HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering

Mobien Shoaib

Fast Multichannel Adaptive RLS Algorithms with Applications to System Identification and Equalizer Design

The thesis has been submitted for official examination for the degree of Master of Science in Espoo, Finland on November 11th, 2005.

Supervisor Professor Timo Laakso
Instructor D.Sc. Stefan Werner
Professor José Apolinário Jr.*

* This MSc project was initiated during prof. Apolinário's stay at TKK in November 2004 to January 2005 as a visiting professor and co-instructed by him per e-mail after that. Unfortunately the current TKK regulations do not enable us to credit him officially as the other instructor of the thesis, which he certainly would deserve.

TEKNILLINEN KORKEAKOULU Diplomityön tiivistelmä

Tekijä:	Mobien Shoaib
Työn nimi:	Nopeat monikanavaiset adaptiiviset RLS-algoritmit ja niiden sovellukset systeemin identifiointiin ja korjainsuunnittseluum
Päivämäärä:	7.11.2005 **Sivumäärä:** 122
Osasto:	Sähkö- ja tietoliikennetekniikan osasto
Professuuri:	Signaalinkäsittelytekniikka
Työn valvoja:	Professori Timo Laakso
Työn ohjaaja:	D.Sc (Tech) Stefan Werner Professori José Apolinário Jr.

Nopeat QR-hajotelmaan perustuvat rekursiivista neliösummaa minimoivat adaptiiviset suodinalgoritmit (Fast QR Decomposition Recursive Least Squares, FQRD-RLS) ovat tunnetusti hyvinkäyttäytyviä niin laskentatarkkuuden kuin laskennan määrän puolesta. Tämän takia algoritmit sopivat myös langattoman tietoliikenteen sovelluksiin. Tehokas laskenta perustuu input-datavektorin viivelinjarakenteen hyödyntämiseen. Algoritmin ongelmana on kuitenkin, että laskennan tuloksena saadaan haluttu suodatustulos muttei suotimen kertoimia eksplisiittisessä muodossa. Tämän takia FQRD-RLS-algoritmia ei ole voitu käyttää kaikissa sovelluksissa.

Tässä diplomityössä esitetään uusi menetelmä, jolla suodinkertoimet saadaan uutettua (weight extraction) sekä yksi- että monikanavaisen FQRD-RLS-algoritmin tapauksessa missä tahansa vaiheessa adaptiivista suodatusta. Suodinkertoimien uuttaminen tapahtuu Cholesky-hajotelman avulla. Esitetty uuttotekniikka mahdollistaa algoritmin soveltamisen mm. systeemin identifiointiin, esi- ja jälkikorjaukseen sekä leveäkaistaiseen keilanmuodostukseen.

Avainsanat: Adaptiivinen suodatus, monikanava, QR-hajotelma, rekursiivinen neliösumma, kertoimien uuttaminen.

HELSINKI UNIVERSITY OF TECHNOLOGY Abstract of the Master's Thesis

Author:	Mobien Shoaib
Name of the Thesis:	Fast Multichannel Adaptive RLS Algorithms with Applications to System Identification and Equalizer Design
Date:	1.9.2005 **Pages:** 122
Department:	Department of Electrical and Communications Engineering
Professorship:	Signal Processing
Supervisor:	Professor Timo Laakso
Instructor:	D.Sc (Tech) Stefan Werner
	Professor José Apolinário Jr.

Fast QR decomposition RLS (FQRD-RLS) algorithms are well known for their good numerical properties and low computational complexity. This makes them desirable for wireless applications. The low complexity is due to fact that the FQRD-RLS algorithms exploit the underlying time-shift structure of the input data vector in order to replace matrix update equations with vector updates. However, the FQRD-RLS algorithms do not provide access to the filter weights, and so far their use has been limited to problems seeking an estimate of the output error signal, e.g., noise cancellation and echo cancellation.

In this thesis we present novel techniques to obtain the filter weights of single channel and multichannel FQRD-RLS algorithms at any time instant during adaptation. We construct weight extraction algorithms based on sequential extraction of the columns of the Cholesky factor embedded in the single and multichannel FQRD-RLS algorithms. From the Cholesky factor we can obtain the filter weights in explicit form. The proposed weight extraction techniques for single and multichannel algorithms are applied to system identification, post equalizer, pre-equalizer and broadband beamforming applications.

Keywords: Adaptive filter, multichannel, QR-Decomposition, Recursive Least Squares, weight extraction.

Foreword

This Master of Science thesis was written in Signal Processing Laboratory at the Helsinki University of Technology. This work was funded by Academy of Finland and Smart and Novel Radios (SMARAD) Center of Excellence.

I would like to thank Professor Timo I. Laakso for giving me the opportunity to work under his supervision. Timo has been a great support during my time at Signal Processing Laboratory. His constructive criticism has contributed to the technical quality of the manuscript. He has been very understanding and friendly, especially during the time when I was unable to produce any results.

I wish to express my gratitude to my instructor D.Sc. Stefan Werner. His contributions in this work are countless. He helped me through every stage of my research work, starting from understanding the research problem till arriving at the final results. The present text is the outcome of his numerous suggestions. He taught me how to present research in simple and objective manner. Without his guidance, and support I would never have been able to complete my thesis.

I truly appreciate the support, supervision and friendship of D.Sc. Apolinário Jr. He has been a second instructor, and I have learned a lot from his experience as researcher. It was his teaching at this university that inspired me to carry out this research. I enjoyed his humorous emails with kind suggestions. I hope and believe our co-operation will continue in the future.

From the rest of the laboratory personnel my special thanks go to Anne Jääskeläinen and Mirja Lemetyinen.

The deepest gratitude goes to my family for their love and support.

Helsinki, November 1st, 2005

Mobien Shoaib

Contents

List of Acronyms

FIR	Finite Impulse Response
FQR_POS_B	Fast QR *a POSteriori* Backward prediction error
FQR_POS_F	Fast QR *a POSteriori* Forward prediction error
FQR_PRI_B	Fast QR *a PRIori* Backward prediction error
FQR_PRI_F	Fast QR *a PRIori* Forward prediction error
FQRD	Fast QR-Decomposition
GSM	Global System Mobile
IQRD	Inverse QR decomposition
ISI	Inter-Symbol Interference
LMS	Least Mean Squares
MCFQRD	Multichannel Fast QR decomposition
MIMO	Multi-Input Multi-Output
MSE	Mean Square Error
NLMS	Normalized least mean squares
QRD	QR decomposition
RLS	Recursive Least Squares
WE	Weight Extraction

List of Symbols

$x(k)$	input signal sequence
$\mathbf{x}(k)$	input signal vector
$y(k)$	output signal sequence
$d(k)$	desired signal sequence
$\mathbf{d}(k)$	desired signal vector
$e(k)$	error signal sequence
$\mathbf{e}(k)$	error signal vector
$n(k)$	noise signal sequence
$\mathbf{n}(k)$	noise signal vector
$\mathbf{w}(k)$	vector of coefficients of the adaptive filter
$w_i(k)$	ith coefficient of the adaptive filter
k	iteration/index
$u(k)$	output signal sequence in pre-equalizer
$i(k)$	source signal sequence in noise canceller
$\mathbf{J_w}$	objective function
μ	step size
$\mathbf{R}(k)$	autocorrelation matrix
$\mathrm{tr}\{.\}$	trace
$\|\mathbf{x}\|$	norm of $\mathbf{x}$
$\mathbf{X}(k)$	input signal matrix
$\mathbf{p}(k)$	cross-correlation vector
$\mathbf{U}(k)$	Cholesky factor

$\tilde{\mathbf{Q}}_{\theta}(k)$	orthogonal rotation matrix
$\tilde{\mathbf{Q}}_{\theta_i}(k)$	Givens rotation matrix
$\mathbb{R}^{N\times 1}$	set of real N-dimensional vectors
λ	forgetting factor
$\varepsilon(k)$	*a posteriori* error sequence
N	number of filter coefficients
$\mathbf{w}_b(k)$	coefficient vector of the backward predictor
$\mathbf{w}_f(k)$	coefficient vector of the forward predictor
$\mathbf{d}_b(k)$	vector of backward prediction desired signal
$\mathbf{d}_f(k)$	vector of forward prediction desired signal
$e_b(k)$	*a priori* backward prediction error sequence
$e_f(k)$	*a priori* forward prediction error sequence
$\mathbf{e}_b(k)$	vector of *a priori* backward prediction error
$\mathbf{e}_f(k)$	vector of *a priori* forward prediction error
$\varepsilon_b(k)$	*a posteriori* backward prediction error sequence
$\varepsilon_f(k)$	*a posteriori* forward prediction error sequence
δ_i	kornecker delta function
$\mathbf{x}_N(k)$	multichannel input signal vector
$\mathbf{X}_N(k)$	multichannel input signal matrix
$\mathbf{w}_N(k)$	vector of coefficients of the multichannel adaptive filter
$\mathbf{W}_{Nb}(k)$	coefficient matrix of the multichannel backward predictor
$\mathbf{W}_{Nf}(k)$	coefficient matrix of the multichannel forward predictor
$\mathbf{E}_b(k)$	matrix of the multichannel backward predictor error
$\mathbf{E}_f(k)$	matrix of the multichannel forward predictor error
$\mathbf{D}_b(k)$	matrix of the multichannel backward predictor desired signal
$\mathbf{D}_f(k)$	matrix of the multichannel forward predictor desired signal
$\mathbf{s}(\theta_i)$	steering vector
θ_i	direction of arrival of user i
$\mathbf{Q}$	rotation matrix

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Adaptive filters find many applications in communications systems, e.g., acoustic echo cancellation [1], channel equalization [2], and interference suppression [3]. An adaptive filter takes measurements from the environment and modifies itself based on these measurements; in other words, adaptive filters are *self-designing* [4]. When selecting an adaptive filtering algorithm, it is usually desirable that the algorithm converges to the optimum solution fast and in a computationally cost-efficient manner. The choice of the algorithm is, therefore, based on the computational complexity and the convergence speed. The convergence speed is often a conflicting requirement with the computational complexity of the algorithm. Therefore, the algorithms that successfully combine high convergence speed and low computational complexity are of special interest.

The Recursive Least-Squares (RLS) algorithm and the Least Mean Square (LMS) algorithm are the two most commonly used adaptive filtering algorithms. The LMS algorithm has the desirable property of low computational complexity but suffers from slow convergence speed for correlated input signals. In some applications with a high number of unknown weights, such as Volterra system identification, slow convergence speed is not appreciated. The RLS algorithm on the other hand is one of the fastest converging adaptive filtering algorithms. The convergence speed of the RLS algorithm usually serves as a benchmark for adaptive filtering algorithms. However, there are also numerical stability issues associated with it, mainly when implemented in a finite precision environment [5]. Another issue with this algorithm is a high computational cost. The numerical stability problem can be addressed via QR-decomposition (QRD) based algorithms. By considering the fast QRD based algorithms we can find algorithms with fast convergence, numerical stability, and low computational complexity of $\mathcal{O}(N)$ [6], [7], [8]. Based on their properties, fast QRD-RLS algorithms can be considered most suitable for a wide range of applications.

The main idea in the 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 [9]. The vector update equations are derived from forward and backward predictions. This thesis considers algorithms based on updating of the backward prediction errors, which are known to be numerically robust [10]. The main limitation of the 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, their applications are limited to output error based (e.g., noise or echo cancellation).

The objective of this thesis is to obtain the weights embedded in the internal variables of the FQRD-RLS algorithm, in order to extend the range of applications of the single-channel and multichannel FQRD-RLS algorithm. The proposed method must rely on the knowledge of only vector updates present in the FQRD-RLS algorithms, as opposed to the matrix-embedded structure of the conventional QRD-RLS described in [11]. The knowledge of weights enables new applications for FQRD-RLS algorithm such as, system identification for linear and Volterra based systems, spectral analysis of the channel equalizer weights, and antenna beamforming for

MIMO systems. We also seek efficient techniques that enable output filtering without obtaining the weights explicitly. This technique can be utilized in burst-type training scenarios and pre-equalization using indirect learning architecture [12].

1.2 Organization of the Thesis

A general overview of adaptive filtering is presented in Chapter 2. A few applications of adaptive filtering are mentioned. The Least Mean Square algorithm and the Recursive Least Square based algorithms, basic RLS, QRD-RLS and IQRD-RLS are then introduced. Chapter 3 elaborates the fast QRD-RLS algorithm. First forward and backward prediction are introduced, and later these concepts are used to derive the fast QRD-RLS algorithms based on forward and backward prediction error updates. The derivatives of the FQRD-RLS algorithm based on backward prediction errors are also derived. The main contributions of the thesis are presented in Chapters 4 and 5.

The single channel weight extraction algorithm is derived in Chapter 4. The main idea, summarized by two lemmas, is presented to provide an algorithm that allows, at any time instant during adaptation, to sequentially extract the columns of the Cholesky factor embedded in the FQRD-RLS algorithm. From the Cholesky factor the true weights of the underlying LS problem can be obtained by reusing the known FQRD-RLS variables. In this chapter we also present the equivalent-output filtering algorithms that allow us to reproduce the output signal without obtaining the weights explicitly. As a consequence, the FQRD-RLS algorithm can be used in conjunction with equalizer and pre-equalizer applications. A detailed derivation of the algorithm is presented along with the experimental results.

Chapter 5 elaborates the multichannel concepts and provides the derivation of the multichannel Fast QRD-RLS algorithm based on backward and forward prediction error updates. Next, the weight extraction algorithm for the multichannel FQRD-RLS algorithm is provided with the experimental results. The idea of equivalent-output filtering for the single-channel case is also extended for the multichannel FQRD-RLS algorithm. Conclusions are drawn in Chapter 6.

Chapter 2

Adaptive Filtering and Algorithms

There are many applications in communication systems, digital control and signal processing that require the knowledge of a time-varying unknown system. The objective is either to identify the unknown system or to identify a function of it, e.g., the inverse function. A filter is designed based on the knowledge of statistical characteristics of the input signal. If the statistics of the input signal are changing or missing, an optimum filter cannot be designed. In such scenario pre-designed static filters are not useful, instead we require a special class of self-designing filters known as adaptive filters. The adaptive filters are not pre-designed but they rather use the input signal and a desired signal to design themselves.

Adaptive filtering is a method of recursively finding the estimate of the signal and then updating the filter parameters according to a fixed criteria, based on the estimate. Adaptive filters are therefore time varying, data dependent, and self-designing. Due to the self-designing property, adaptive filters are appropriate for the above mentioned problem that appears in applications such as system identification, channel equalization, pre-equalization, noise cancellation, adaptive beamforming, etc.

In technical literature, we can find a large number of adaptive filtering algorithms. They differ in many distinct ways including the cost function, convergence properties, algorithmic complexity, etc. Here we consider only the recursive least-squares algorithm and the square root least-squares based algorithms including the QRD-RLS, the inverse QRD-RLS and the fast QRD-RLS (FQRD-RLS) algorithms. The RLS algorithms are known for their good performance in terms of fast convergence, but also for high computational complexity.

The purpose of this chapter is to introduce the basic concepts of adaptive filters and the adaptive filtering algorithms to be used in this thesis. In Section 2.1 an adaptive filter setup is presented and the basic notation is introduced. In Section 2.2 applications of adaptive filtering are discussed, which include system identification, post-equalizer, pre-equalizer, and noise canceler. The RLS adaptive filtering algorithms are addressed in Section 2.4 followed by a table of their computational complexity.

2.1 Adaptive Filter Setup

The basic setup of an adaptive filter is depicted in Figure 2.1, where $x(k)$, $y(k)$, $d(k)$, and $e(k)$ are the input, the output, the desired, and the error signals, respectively. The adaptive filter considered in this document has Finite Impulse Response (FIR). The N-element weight vector available at the time instant k (from time instant $k-1$, i.e., assuming one unit delay), is denoted by $\mathbf{w}(k-1)$, where

$$\mathbf{w}(k-1) = \begin{bmatrix} w_0(k-1) & w_1(k-1) & \dots & w_{N-1}(k-1) \end{bmatrix}^{\mathrm{T}} \tag{2.1}$$

and $w_i(k-1)$ denotes the i^{th} element of the weight vector. The filter output $y(k)$ is a linear combination of the current and the previous $N-1$ values of the input $x(k)$, given by

$$y(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) \tag{2.2}$$

where

$$\mathbf{x}(k) = \begin{bmatrix} x(k) & x(k-1) & \dots & x(k-N+1) \end{bmatrix}^{\mathrm{T}} \tag{2.3}$$

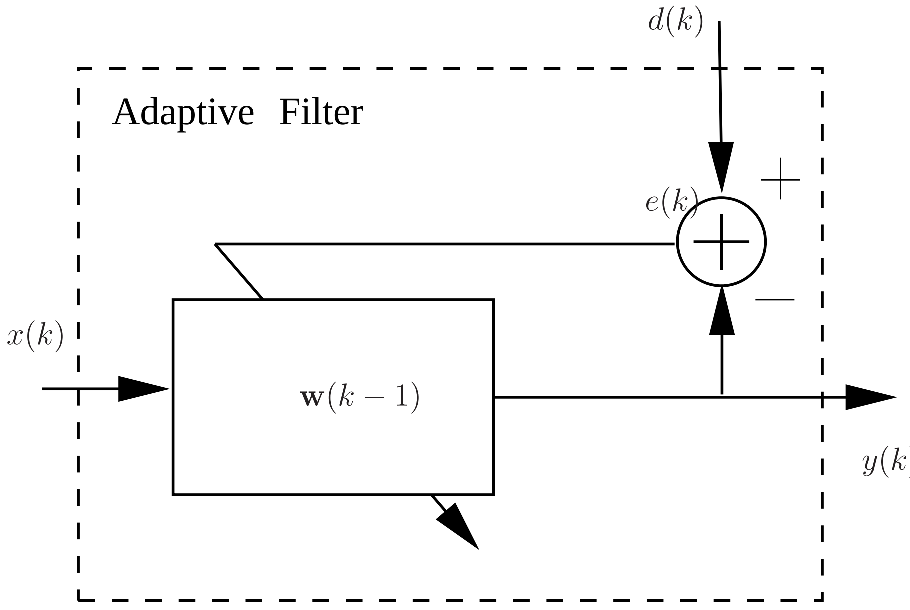


Figure 2.1: Schematic diagram of an adaptive filter

is the input data vector, index k defines the current sample value, while $k-i$ indicates the i^{th} previous value. The definition of the *a priori* error signal is given by

$$e(k) = d(k) - y(k) \tag{2.4}$$

An adaptive filtering algorithm attempts to recursively minimize a cost function often related to the error signal. The objective is to find a weight vector that corresponds to the minimum of the cost function. The weight vector that minimizes the cost function is called the optimum weight vector.

The performance of an adaptive filter is determined by its properties. The most important properties are [4]:

1. *Rate of convergence.* The number of iterations required to converge to the steady-state solution depends on the rate of convergence. Fast convergence means the algorithm reaches the steady-state solution in a small number of iterations.

2. *Misadjustment.* A quantitative measure for how close the obtained solution is to the optimum solution.

3. *Robustness.* An adaptive filter is said to be robust when small disturbances in input distribution result in small estimation errors.

4. *Computational complexity.* The number of operations and the memory required to complete an iteration. The operations may include multiplications, additions, divisions, square roots.

6. *Filter stability.* Adaptive FIR filters are inherently stable. An IIR based adaptive filter algorithm becomes unstable when the poles of the filter are outside the unit circle. An unstable adaptive filter causes the adaptation algorithm to diverge.

7. *Numerical stability.* An algorithm is numerically stable if it converges in finite-precision environments.

In this thesis, these performance criteria are considered for each algorithm.

2.2 Applications

This section presents four applications of adaptive filtering that will be referred to throughout this thesis. The particular applications are selected in order to elaborate on the shortcomings of the FQRD-RLS algorithm. From the applications considered here, only the noise cancellation application is suitable for the FQRD-RLS algorithms, whereas system identification and pre-equalization are applications for which the FQRD-RLS algorithms cannot be used in its present form. Also a post-equalization scenario is discussed for which the conventional FQRD-RLS algorithm is not applicable.

2.2.1 System Identification

System identification is one of key setups of adaptive filtering in signal processing, communications and control systems, as in [1, 13–15]. A system identification setup is depicted in Figure 2.2. The purpose of system identification is to estimate the coefficients of an unknown system or plant. The unknown system, here modelled as an FIR filter, and the adaptive filter have the same input signal. The desired or reference signal corresponds to the output of the unknown system contaminated with measurement noise $n(k)$.

After the adaptation algorithm has converged, the coefficients of the unknown system (in case of perfect modelling) are given by the adaptive filter weight vector. An algorithm incapable of providing the weight vector in an explicit form cannot be used for such applications. For example, the FQRD-RLS algorithm considered in Chapter 3 does not provide the weight vector. Therefore, if system identification is desired using a stable RLS algorithm, we must restore to algorithms with computational complexity of $\mathcal{O}(N^2)$, like the inverse QRD-RLS algorithm (see Section 2.4 for algorithm details).

2.2.2 Post-Equalizer

The purpose of the post-equalizer setup is to find the coefficients of the inverse system to the unknown system. The setup comprises an adaptive filter connected in cascade with the unknown system, hence the name "post-equalizer" or simply equalizer. Furthermore, the input to the unknown system also acts as the desired signal. Channel equalization is an application of the post-equalizer in which a communication channel acts as the unknown system. The channel makes the transmitted signal unrecognizable for the receiver due to intersymbol interference (ISI). Post-equalizer is used to model the inverse channel. The inverse channel is then applied to the received signal so as to undo the distortion due to the ISI [16], [17], [2]. A post-equalizer setup for channel equalization is shown in Figure 2.3, where $x(k)$ is

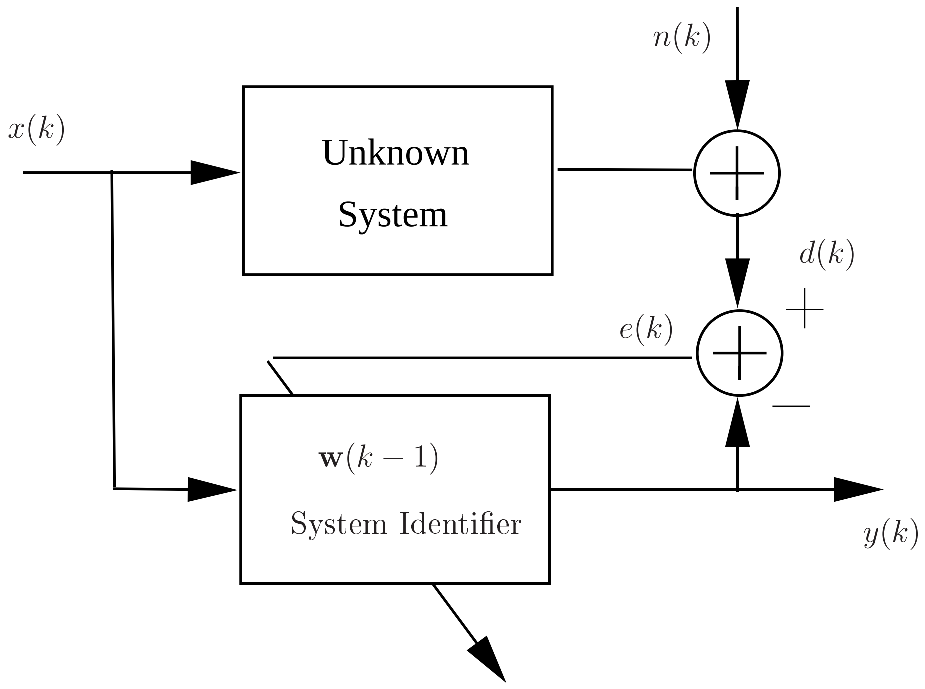


Figure 2.2: System identification using adaptive filtering

the transmitted signal, $d(k)$ is a training signal corresponding to a delayed version of the transmitted signal $x(k)$, $y(k)$ is the output of the adaptive filter, and $e(k)$ is the error between $d(k)$ and $y(k)$. The adaptive filter adjusts the weights $\mathbf{w}(k-1)$ so that $y(k)$ gives an estimate of $x(k-L)$. The delay expressed by z^{-L} is used to provide the correct synchronization of the training sequence.

After the convergence, the adaptive filter weight vector approximates the inverse of the channel. At this stage the adaptation process can be stopped to save computational costs, provided that the channel is not time-varying. The output is obtained by using the adaptive filter coefficients from the inverse filter. As discussed before, this is only possible if the adaptive algorithm can provide the coefficients in explicit form. For example, the Fast QRD-RLS based algorithms do not provide the facility to do so due to unavailability of its coefficients at each iteration.

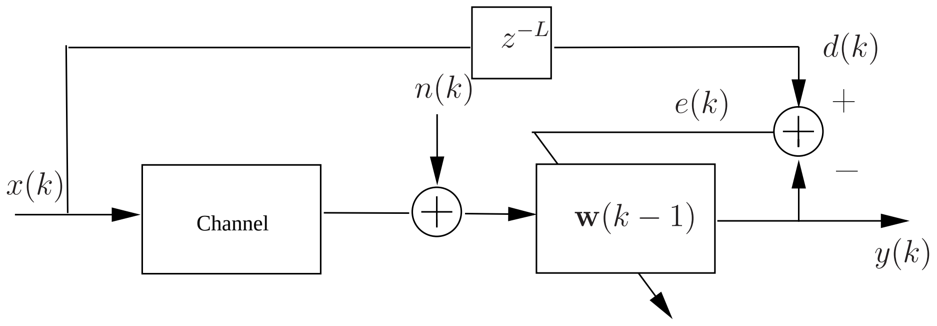


Figure 2.3: The post-equalizer for channel equalization

2.2.3 Pre-Equalizer

The aim of the pre-equalizer is to equalize the unknown linear system using a special setup known as the indirect learning architecture [12]. Consider a pre-equalizer setup as in Figure 2.4. It is desired that the output of the unknown system $y(k)$ matches the input to the pre-equalizer $x(k)$. The input to the adaptive filter is $y(k)$, its output $u(k)$ attempts to match the output of the pre-equalizer $d(k)$. For the adaptive filter, $d(k)$ is the desired signal, and $u(k)$ is the estimate of the desired signal. A cost function related to error signal $e(k) = d(k) - u(k)$ is to be minimized. A consequence of this minimization is that $y(k)$ becomes an estimate of $x(k)$.

The most critical step at each iteration is to copy the adaptive filter weight vector to the pre-equalizer. Without this crucial step, the indirect learning architecture [12] cannot work. The adaptive filtering algorithm should have the provision for copying the weight vector at every iteration. Again the RLS, the QRD-RLS, and the inverse QRD-RLS algorithms are suitable for this application.

2.2.4 Noise Canceller

Noise or interference cancellation has many applications, e.g, in biomedical applications where an adaptive filter is used to remove 60-Hz interference in electrocardiography [18], in cancellation of engine noise in the cockpit of a jet fighter [19],

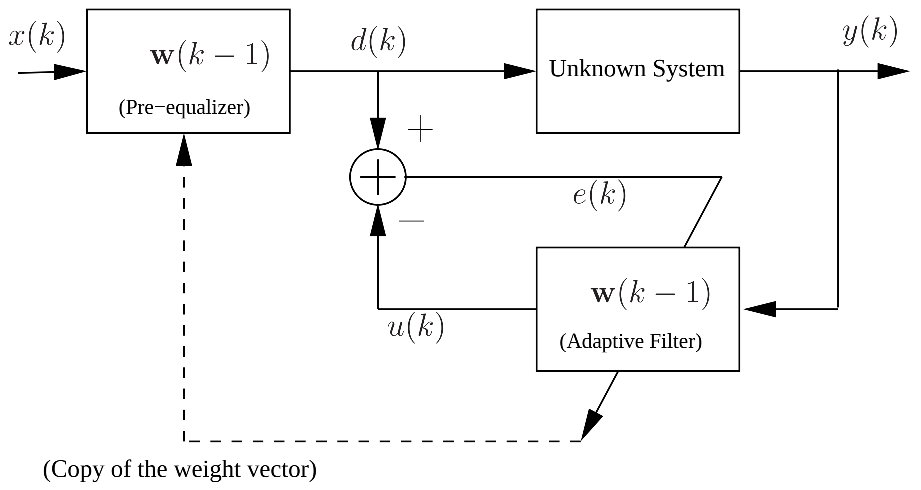


Figure 2.4: The pre-equalizer using indirect learning architecture

etc. A noise cancellation setup is given in Figure 2.5. In the scenario, there is an interference source. The interference signal is given by $i(k)$. The interference source corrupts a wanted signal $x(k)$ after being modified by a system $H_1(z)$. The objective is to remove the interference from the wanted signal. The interference signal after passing through another system $H_2(z)$, or , or $d(k) = i(k) * h_2(k)$ yet correlated to $i(k) * h_1(k)$, is taken as the desired signal. The corrupted signal is given as an input to the adaptive filter. Eventually, the error signal gives the estimate of the signal of interest.

In this application, the FQRD-RLS algorithm can be used, as the weight vector computation step is not necessary. The Fast QRD-RLS algorithm would be preferred over the QRD-RLS or the inverse QRD-RLS because of its lower computational complexity.

2.3 The LMS Algorithm

The LMS algorithm is popular due to its low computational complexity and proven robustness [20]. The LMS algorithm minimizes the objective function that is based

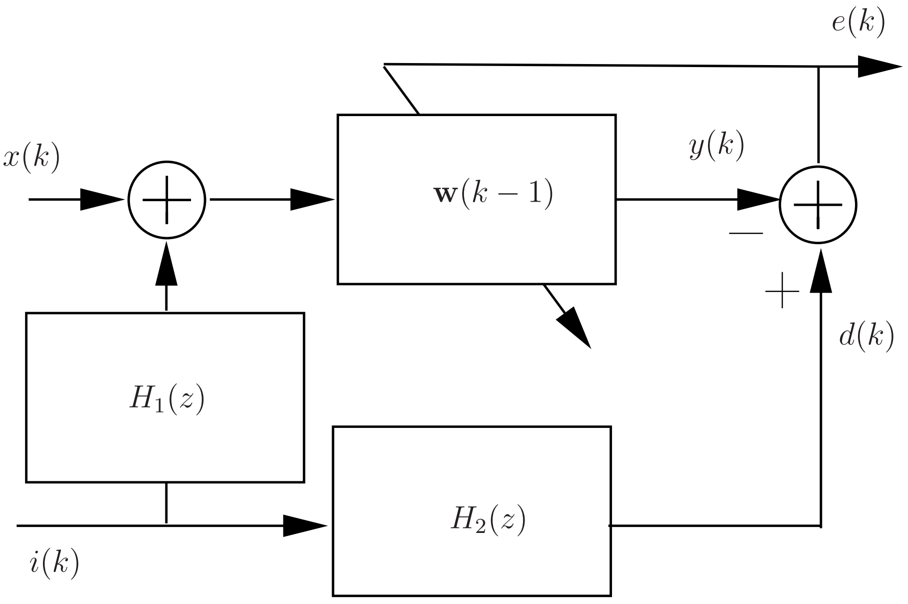


Figure 2.5: A noise cancellation setup

on the MSE, i.e., the instantaneous estimate of the MSE

$$\mathbf{J}_{\mathbf{w}} = e^2(k) \tag{2.5}$$

The coefficient vector is updated by taking a step in the direction of the negative gradient of the objective function

$$\frac{\partial \mathbf{J}_{\mathbf{w}}}{\partial \mathbf{w}(k)} = -2e(k)\mathbf{x}(k) \tag{2.6}$$

Therefore the update equation for the LMS algorithm becomes

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu e(k)\mathbf{x}(k) \tag{2.7}$$

where μ is the step size governing the stability, convergence speed and misadjustment. The value of the step size should be chosen in the range [4]

$$0 < \mu < 2/\mathrm{tr}\{\mathbf{R}\} \tag{2.8}$$

Table 2.1: The LMS Algorithm.

```
for each $k$
{
    $e(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)$
    $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu e(k)\mathbf{x}(k)$
}
```

where tr{.} is the trace operator and

$$\mathbf{R} = \mathrm{E}\{\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\} \tag{2.9}$$

is the input-signal autocorrelation matrix. The upper bound is loose for practical scenarios, therefore much smaller values are recommended [21]. Another variation of the LMS algorithm is the NLMS algorithm, which uses a time varying step size $\mu/\|\mathbf{x}(k)\|^2$. A drawback of the LMS and NLMS algorithms is the slow convergence speed for colored input signal. This thesis focuses on the algorithms with fast converging speed, so that LMS algorithm is not considered further in the thesis. The LMS algorithm is given in Table 2.1.

2.4 The family of RLS Adaptive Filtering Algorithms

As mentioned before, only the weighted least-squares based adaptive filtering algorithms are considered in this thesis. The idea of this family of adaptive filters starts from the weighted least-squares problem. The direct solution to the weighted least squares problem leads to the RLS algorithm [4], [21]. The RLS algorithm uses matrix inversion lemma to compute the matrix inverse which may cause numerical stability problems mainly in a finite-precision environment. The QRD-RLS algorithm avoids the matrix inversion lemma by considering the QR decomposition of the autocorrelation matrix. This results in a numerically stable algorithm. The QRD-RLS algorithm does not show any significant advantage over the RLS algorithm in terms of computational complexity. However there is a disadvantage with the QRD-RLS algorithm: the computation of weight vector requires an extra computational effort. The inverse QRD-RLS algorithm resolves this limitation. In the following subsections the RLS, the QRD-RLS, and the inverse QRD-RLS algorithm are elaborated; for each algorithm a short derivation is given followed by the advantages and limitations. The summary of each algorithm in form of pseudocode is also presented in the tables.

2.4.1 Recursive Least Squares Algorithm

Consider the adaptive filter setup of Figure 2.1. The WLS solution attempts to find the vector $\mathbf{w}$ that minimizes, at each time instant k, the objective function given as

$$\mathbf{J}_{\mathbf{w}} = \sum_{i=0}^{k} \lambda^i e^2(k-i) = \sum_{i=0}^{k} \lambda^i [d(k-i) - \mathbf{x}^{\mathrm{T}}(k-i)\mathbf{w}]^2 \tag{2.10}$$

where λ is the forgetting factor, and $e(k-i) = d(k-i) - \mathbf{x}^{\mathrm{T}}(k-i)\mathbf{w}$ is an *a posteriori* error signal. The objective function in Equation (2.10) can also be written in vector form as

$$\mathbf{J}_{\mathbf{w}} = \|\mathbf{e}(k)\|^2 \tag{2.11}$$

where $\mathbf{e}(k)$ is the error vector containing the weighted past error values $\lambda^{i/2}e(k-i)$

$$\mathbf{e}(k) = \begin{bmatrix} e(k) & \lambda^{1/2}e(k-1) & \ldots & \lambda^{k/2}e(0) \end{bmatrix}^{\mathrm{T}} = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w} \tag{2.12}$$

where $\mathbf{w}$ is the optimum weight vector to be solved for, and the input data matrix $\mathbf{X}(k) \in \mathbb{R}^{(k+1)\times N}$ is

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}(k) & \lambda^{1/2}\mathbf{x}(k-1) & \ldots & \lambda^{k/2}\mathbf{x}(0) \end{bmatrix}^{\mathrm{T}} \tag{2.13}$$

and the desired signal vector $\mathbf{d}(k) \in \mathbb{R}^{(k+1)\times 1}$ is

$$\mathbf{d}(k) = \begin{bmatrix} d(k) & \lambda^{1/2}d(k-1) & \dots & \lambda^{k/2}d(0) \end{bmatrix}^{\mathrm{T}} \tag{2.14}$$

Minimizing the objective function in Equation (2.10) with respect to the weight vector $\mathbf{w}$ results in

$$\mathbf{w}(k) = \arg\min_{\mathbf{w}} \mathbf{J}_{\mathbf{w}} = \mathbf{R}^{-1}(k)\mathbf{p}(k) \tag{2.15}$$

where the sample input-signal autocorrelation matrix $\mathbf{R}(k) \in \mathbb{R}^{N\times N}$ is given by

$$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) \tag{2.16}$$

and the sample cross-correlation vector $\mathbf{p}(k) \in \mathbb{R}^{N\times 1}$ is given by

$$\mathbf{p}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) \tag{2.17}$$

In the RLS algorithm, matrix $\mathbf{R}(k)$ and vector $\mathbf{p}(k)$ are recursively updated as

$$\mathbf{R}(k) = \lambda\mathbf{R}(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) \tag{2.18}$$

$$\mathbf{p}(k) = \lambda\mathbf{p}(k-1) + \mathbf{x}(k)d(k) \tag{2.19}$$

Using Equation (2.19) in Equation (2.15) we find the following recursive update for

Table 2.2: *The RLS Algorithm.*

```
$\mathbf{R}^{-1}(k) = \delta^{-1}\mathbf{I}$, $\delta$ small positive constant.
for each $k$
{
    $\mathbf{k}(k) = \mathbf{R}^{-1}(k-1)\mathbf{x}(k)$
    $\kappa(k) = \frac{\mathbf{k}(k)}{\lambda+\mathbf{x}^{\mathrm{T}}(k)\mathbf{k}(k)}$
    $\mathbf{R}^{-1}(k) = \frac{1}{\lambda}[\mathbf{R}^{-1}(k-1) - \frac{\mathbf{k}(k)\mathbf{k}^{\mathrm{T}}(k)}{\lambda+\mathbf{x}^{\mathrm{T}}(k)\mathbf{k}(k)}]$
    $e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k)$
    $\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\kappa(k)$
}
```

the weight vector

$$\begin{aligned}
\mathbf{w}(k) &= \mathbf{R}^{-1}(k)[\lambda\mathbf{p}(k-1) + d(k)\mathbf{x}(k)] \\
&= \mathbf{R}^{-1}(k)[\lambda\underbrace{\mathbf{R}(k-1)\mathbf{w}(k-1)}_{\mathbf{p}(k-1)} + d(k)\mathbf{x}(k)] \\
&= \mathbf{R}^{-1}(k)[\lambda\mathbf{R}(k-1)\mathbf{w}(k-1) + d(k)\mathbf{x}(k) \\
&\quad + \underbrace{\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1) - \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)}_{\mathbf{0}}] \\
&= \mathbf{R}^{-1}(k)[\lambda\mathbf{R}(k-1)\mathbf{w}(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1) \\
&\quad + d(k)\mathbf{x}(k) - \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)] \\
&= \mathbf{R}^{-1}(k)[\mathbf{R}(k)\mathbf{w}(k-1) + \mathbf{x}(k)[d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)]] \\
&= \mathbf{R}^{-1}(k)[\mathbf{R}(k)\mathbf{w}(k-1) + \mathbf{x}(k)e(k)] \\
&= \mathbf{w}(k-1) + e(k)\mathbf{R}^{-1}(k)\mathbf{x}(k)
\end{aligned} \tag{2.20}$$

The inverse of $\mathbf{R}(k)$, $\mathbf{R}^{-1}(k)$, can be computed using the so-called *matrix inversion* lemma [21] in order to reduce the computational complexity of the matrix inversion. The RLS algorithm is summarized in Table 2.2.

The RLS algorithm can be directly applied in all four applications mentioned in Section 2.2 because of the availability of the weight vector at each iteration. The suitability of the RLS algorithm for practical applications can only be judged by looking at the performance properties of the RLS algorithms that are given as follows:

1. *Rate of convergence:* The RLS algorithm has excellent convergence properties. The convergence rate is in the order of the number of coefficients and is independent of the eigenvalue spread of the input-signal autocorrelation matrix.
2. *Misadjustment:* The RLS algorithm has small misadjustment factor.
3. *Computational complexity:* It requires $\mathcal{O}(N^2)$ computations.
4. *Numerical stability:* The RLS algorithm shows numerical instability [5].

Computational complexity and numerical instability are two major drawbacks of the RLS algorithm. Therefore, despite the excellent convergence rate and small misadjustment, the RLS algorithm is not recommended for finite-precision practical applications.

2.4.2 QRD-RLS Algorithms

As mentioned earlier, the RLS algorithm suffers from numerical problems which are related to the direct computation of the inverse of the autocorrelation matrix via the matrix inversion lemma. In QRD-RLS algorithms, an indirect approach is considered in which the Cholesky factorization of the autocorrelation matrix is exploited, leading to a numerically stable algorithm [4]. In the following subsections, we discuss two important QRD-RLS based algorithms

1. the conventional QRD-RLS algorithm; and
2. the inverse QRD-RLS algorithm

Before going into the detailed description of the algorithms in the next subsections, a background survey is provided as follows.

Preliminaries

The Cholesky decomposition of $\mathbf{R}(k)$ is given by

$$\mathbf{R}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) \tag{2.21}$$

where $\mathbf{U}(k) \in \mathbb{R}^{N\times N}$ is either an upper or lower triangular Cholesky decomposition matrix. The matrix $\mathbf{U}(k)$ is related to $\mathbf{X}(k)$ through an orthogonal rotation matrix $\tilde{\mathbf{Q}}_\theta(k) \in \mathbb{R}^{(k+1)\times(k+1)}$ as [22]

$$\begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_\theta(k)\mathbf{X}(k) \tag{2.22}$$

The QRD-RLS based algorithms use the same error vector for the minimization as given in Equation (2.12). The only difference is that the error vector is multiplied with the rotation matrix $\tilde{\mathbf{Q}}_\theta(k)$ giving the rotated error vector $\mathbf{e}_q(k)$.

$$\mathbf{e}_q(k) = \tilde{\mathbf{Q}}_\theta(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k) \tag{2.23}$$

where

$$\begin{bmatrix} \mathbf{d}_{q1}(k) & \mathbf{d}_{q2}(k) \end{bmatrix}^{\mathrm{T}} = \mathbf{d}^{\mathrm{T}}(k)\tilde{\mathbf{Q}}_\theta^{\mathrm{T}}(k) \tag{2.24}$$

As a result of $\tilde{\mathbf{Q}}_\theta(k)$ being orthogonal, the objective function in the QRD-RLS algorithm is the same as in the RLS algorithm, as can be seen below.

$$\begin{aligned} \|\mathbf{e}_q(k)\|^2 &= \begin{bmatrix} \mathbf{e}_{q1}^{\mathrm{T}}(k) & \mathbf{e}_{q2}^{\mathrm{T}}(k) \end{bmatrix} \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} \\ &= \mathbf{e}^{\mathrm{T}}(k)\tilde{\mathbf{Q}}_\theta^{\mathrm{T}}(k)\tilde{\mathbf{Q}}_\theta(k)\mathbf{e}(k) \\ &= \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k) \\ &= \|\mathbf{e}(k)\|^2 \end{aligned} \tag{2.25}$$

The optimum weight vector for the QRD-RLS adaptive filtering algorithm is given by

$$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k) \tag{2.26}$$

which corresponds to the solution providing $\mathbf{e}_{q2}(k) = \mathbf{0}_{N\times 1}$ in Equation (2.23). As the QRD-RLS minimizes the same objective function as the RLS, Equation (2.26) is equivalent to Equation (2.15).

The RLS algorithm computes the weight vector iteratively. In order to do so, an update equation for important variables such as $\mathbf{R}^{-1}(k)$ are required, see Table 2.2. Similarly, for QRD-RLS based algorithms update equations for $\mathbf{U}^{-1}(k)$ and $\mathbf{d}_{q2}(k)$ are needed. Associated with these updates, a rotation matrix has to be computed at each iteration. Two types of QRD-RLS based algorithms are discussed in the next two subsections: the conventional QRD-RLS, and the inverse QRD-RLS.

QRD-RLS and Givens Rotation Matrices: Description and Implementation

The rotation matrix $\tilde{\mathbf{Q}}_\theta(k)$ can be written as a sequence of Givens rotation matrices. Due to the increasing order of the rotation matrix the representation in Givens rotation form becomes complicated. A much simpler definition can be obtained if the rotation matrix is of fixed order. To get the fixed-order rotation matrix $\mathbf{Q}_\theta(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ we rewrite the Equation (2.22) as

$$\begin{aligned}
\begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} &= \tilde{\mathbf{Q}}_\theta(k) \underbrace{\begin{bmatrix} 1 & \mathbf{0}_{1\times N} \\ \mathbf{0}_{N\times 1} & \tilde{\mathbf{Q}}_\theta^T(k-1) \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}_{1\times N} \\ \mathbf{0}_{N\times 1} & \tilde{\mathbf{Q}}_\theta(k-1) \end{bmatrix}}_{\mathbf{I}_{(k+1)\times(k+1)}} \mathbf{X}(k) \\
&= \underbrace{\tilde{\mathbf{Q}}_\theta(k) \begin{bmatrix} 1 & \mathbf{0}_{1\times k} \\ \mathbf{0}_{k\times 1} & \tilde{\mathbf{Q}}_\theta^T(k-1) \end{bmatrix}}_{\mathbf{Q}(k)} \begin{bmatrix} 1 & \mathbf{0}_{1\times k} \\ \mathbf{0}_{k\times 1} & \tilde{\mathbf{Q}}_\theta(k-1) \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}^{\mathrm{T}}(0) \end{bmatrix}
\end{aligned} \tag{2.27}$$

which, using Equation (2.22) and the fact that the last k rows of $\mathbf{X}(k)$ corresponds to $\lambda^{1/2}\mathbf{X}(k-1)$, results in

$$\begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \mathbf{0}_{(k-N)\times N} \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} \tag{2.28}$$

The fixed-order update equation is obtained once we remove the rows and columns contributing to the ever increasing number of zeros. Finally, we obtain

$$\begin{bmatrix} \mathbf{0}_{1\times N} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} \tag{2.29}$$

The fixed order rotation matrix is responsible for the annihilation of the row vector $\mathbf{x}^{\mathrm{T}}(k)$ and, as a result, updating the matrix $\mathbf{U}(k)$; Figure 2.6 depicts this phenomenon.

The fixed order rotation matrix $\mathbf{Q}_{\theta}(k)$ can be written in the form of a sequence of N Givens rotation matrices as follows

$$\mathbf{Q}_{\theta}(k) = \mathbf{Q}_{\theta_{N-1}}(k)\mathbf{Q}_{\theta_{N-2}}(k)\ldots\mathbf{Q}_{\theta_0}(k) \tag{2.30}$$

where the $(i+1)^{th}$ Givens rotation matrix $\mathbf{Q}_{\theta_i}(k)$ is given by

$$\mathbf{Q}_{\theta_i}(k) = \begin{bmatrix} \cos\theta_i(k) & \mathbf{0}_{1\times(N-i-1)} & -\sin\theta_i(k) & \mathbf{0}_{1\times i} \\ \mathbf{0}_{(N-i-1)\times 1} & \mathbf{I}_{(N-i-1)} & \mathbf{0}_{(N-i-1)\times 1} & \mathbf{0}_{(N-i-1)\times i} \\ \sin\theta_i(k) & \mathbf{0}_{1\times(N-i-1)} & \cos\theta_i(k) & \mathbf{0}_{1\times i} \\ \mathbf{0}_{i\times 1} & \mathbf{0}_{i\times(N-i-1)} & \mathbf{0}_{i\times 1} & \mathbf{I}_i \end{bmatrix} \tag{2.31}$$

Note that applying a single Givens rotation matrix to a vector modifies only two of its elements. Consider a dummy vector $\mathbf{v}(k) \in \mathbb{R}^{(N+1)\times 1}$ given by

$$\mathbf{v}(k) = \begin{bmatrix} v_0(k) & v_1(k) & \ldots & v_N \end{bmatrix}^{\mathrm{T}} \tag{2.32}$$

where $v_i(k)$ is the $(i+1)^{th}$ element of the vector. Multiplying with the $(i+1)^{th}$ rotation matrix will only affect the $v_0(k)$ and $v_{N-i}(k)$ according to the relations given as follows.

$$\begin{aligned} v_0(k) &= v_0(k)\cos\theta_i(k) - v_{N-i}(k)\sin\theta_i(k) \\ v_{N-i}(k) &= v_0(k)\sin\theta_i(k) + v_{N-i}(k)\cos\theta_i(k) \end{aligned} \tag{2.33}$$

which means that applying N Givens rotation vectors results in $4N$ multiplications and $2N$ additions. Also, we do not store all the N Givens rotation matrices; only the values of sine and cosine for each matrix is saved, making a total of $2N$ values. Using the Givens rotation matrix is therefore computationally cost effective as compared to direct multiplication with the rotation matrix $\mathbf{Q}_\theta(k)$.

2.4.3 QRD-RLS Algorithm

The QRD-RLS algorithm consists of two steps,

1. Matrix $\mathbf{U}(k)$ and vector $\mathbf{d}_{q2}(k)$ are updated recursively;
2. Vector $\mathbf{w}(k)$ is calculated using a backward or forward substitution procedure, from the relation $\mathbf{U}(k)\mathbf{w}(k) = \mathbf{d}_{q2}(k)$.

It should be kept in mind that the computation of the weight vector is not an integral part of the QRD-RLS algorithm itself, and in certain applications such as noise cancellation it is not needed.

The update equation for $\mathbf{U}(k)$ was derived in Section 2.4.2 and is given by Equation (2.29).

$$\begin{bmatrix} \mathbf{0}_{1\times N} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} \tag{2.34}$$

The matrices involved in the update are shown in Figure 2.6. The figure shows the process of obtaining the update of Cholesky factor using the rotation matrix. It

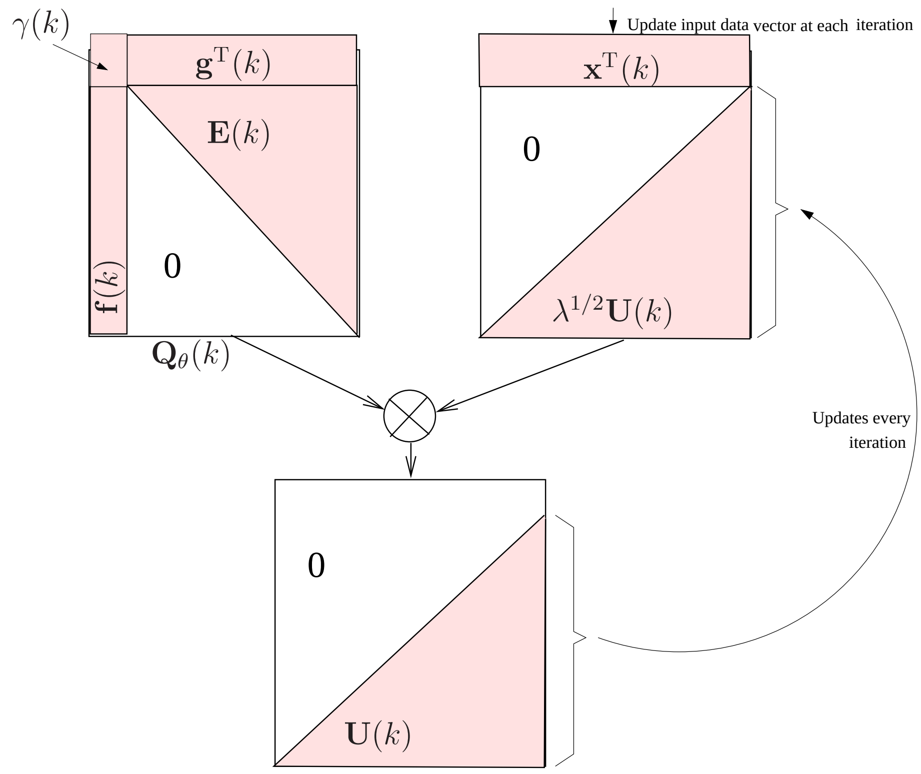

$\gamma(k)$
$\mathbf{g}^{\mathrm{T}}(k)$
$\mathbf{E}(k)$
0
$\mathbf{f}(k)$
$\mathbf{Q}_{\theta}(k)$
Update input data vector at each iteration
$\mathbf{x}^{\mathrm{T}}(k)$
0
$\lambda^{1/2}\mathbf{U}(k)$
Updates every iteration
0
$\mathbf{U}(k)$

Figure 2.6: The update for the lower triangular matrix $\mathbf{U}(k)$ *in the QRD-RLS algorithm.*

can be seen from the figure that the structure (partitions) of the rotation matrix $\mathbf{Q}_\theta(k)$ consists of $\gamma(k)$, $\mathbf{g}(k)$, $\mathbf{f}(k)$, and $\mathbf{E}(k)$. These variables can be identified by manipulating Equation (2.29). Equation (2.35) defines the variables involved in the structure of the rotation matrix.

$$\begin{aligned}\mathbf{g}(k) &= -\lambda^{-1/2}\gamma(k)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) = -\gamma(k)\mathbf{a}(k)\\ \mathbf{f}(k) &= \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k)\\ \mathbf{E}(k) &= \lambda^{1/2}\mathbf{U}^{-\mathrm{T}}(k)\mathbf{U}^{\mathrm{T}}(k-1)\end{aligned} \tag{2.35}$$

Given $\mathbf{Q}_\theta(k)$, the updated vector $\mathbf{d}_{q2}(k)$ is obtained as follows,

$$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix} \tag{2.36}$$

The final step is to get the *a priori* or the *a posteriori* error value. The *a posteriori* error value $\varepsilon(k)$ can be obtained by considering the following definition of $e_{q1}(k)$.

$$\begin{aligned}\varepsilon(k) &= \mathbf{e}_q^{\mathrm{T}}(k)\mathbf{Q}(k)\begin{bmatrix} 1 \\ \mathbf{0}_{k\times 1} \end{bmatrix}\\ &= e_{q1}(k)\gamma(k) + \mathbf{e}_{q2}^{\mathrm{T}}(k)\mathbf{f}(k)\end{aligned} \tag{2.37}$$

It is known that the weight vector $\mathbf{w}(k-1)$ is chosen so that $\mathbf{e}_{q2}(k)$ goes to zero, therefore

$$\varepsilon(k) = e_{q1}(k)\gamma(k) \tag{2.38}$$

and the *a priori* error value is given by

$$e(k) = e_{q1}(k)/\gamma(k) \tag{2.39}$$

The QRD-RLS algorithm is given in Table 2.3.

The properties of the QRD-RLS algorithm are given as follows:

1. *Rate of convergence.* The QRD-RLS algorithm has excellent convergence prop-

Table 2.3: The QRD-RLS Algorithm.

```
for each $k$
{ Obtain $\mathbf{Q}_\theta(k)$ and updating $\mathbf{U}(k)$:
$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix}$
Obtain $\gamma(k)$
$\gamma(k) = \prod_{i=0}^{N} \cos\theta_i(k)$
Obtain $e_{q1}(k)$ and updating $\mathbf{d}_{q2}(k)$:
$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
Obtaining $e(k)$:
$\varepsilon(k) = e_{q1}(k)\gamma(k)$ *a posteriori* error
$e(k) = e_{q1}(k)/\gamma(k)$ *a priori* error
```

erties. The convergence path is identical to that of the RLS algorithm provided that both algorithms are initialized identically.

2. *Misadjustment.* Same as the RLS algorithm.

3. *Computational complexity.* It requires $\mathcal{O}(N^2)$ computations.

4. *Numerical stability.* The QRD-RLS algorithm is numerically stable in finite precision environment [23].

Due to its stable behavior in finite-precision environment the QRD-RLS algorithms is considered to be better than the RLS algorithm. The only drawback is the extra computations required if a weight vector needs to be computed.

2.4.4 Inverse QRD-RLS Algorithm

In the inverse QRD-RLS (IQRD-RLS) algorithm, matrix $\mathbf{U}^{-\mathrm{T}}(k)$ is updated instead of $\mathbf{U}(k)$, and the weight vector $\mathbf{w}(k)$ is explicitly computed as a part of the algorithm [24]. The motivation for computing matrix $\mathbf{U}^{-\mathrm{T}}(k)$ becomes clear if we combine the definition of the autocorrelation matrix $\mathbf{R}(k)$ in Equation (2.21) with the weight update in Equation (2.20)

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\mathbf{U}^{-1}(k)\mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k) \tag{2.40}$$

For the derivation of the IQRD-RLS algorithm, consider the inverse of both sides of the update equation for matrix $\mathbf{U}(k)$ in (2.29). The inversion is possible if we first augment the matrix on the right hand side with the $(N+1)\times 1$ unit vector $\begin{bmatrix}1 & 0\dots & 0\end{bmatrix}^{\mathrm{T}}$ to make the matrix a square matrix. As a result we get,

$$\begin{bmatrix}\mathbf{0}_{1\times N} & \gamma(k)\\ \mathbf{U}(k) & \mathbf{f}(k)\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{x}^{\mathrm{T}} & 1\\ \lambda^{1/2}\mathbf{U}(k-1) & \mathbf{0}\end{bmatrix} \tag{2.41}$$

By inverting and transposing both sides of the equation, assuming that the inverse exists, we get

$$\begin{bmatrix}\frac{-\mathbf{f}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k)}{\gamma(k)} & \frac{1}{\gamma(k)}\\ \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{0}_{N\times 1}\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{0}_{1\times N} & 1\\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) & -\lambda^{1/2}\mathbf{U}^{-T}(k-1)\mathbf{x}(k)\end{bmatrix} \tag{2.42}$$

which can be written in the following compact form

$$\begin{bmatrix}\mathbf{z}^{\mathrm{T}}(k) & \frac{1}{\gamma(k)}\\ \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{0}_{N\times 1}\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{0}_{1\times N} & 1\\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) & -\mathbf{a}(k)\end{bmatrix} \tag{2.43}$$

where vector $\mathbf{z}(k)$ denotes $\frac{-\mathbf{U}^{-1}(k)\mathbf{f}(k)}{\gamma(k)}$. Equation (2.43) can be expressed as two equations,

$$\begin{bmatrix}\frac{1}{\gamma(k)}\\ \mathbf{0}_{N\times 1}\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}1\\ -\mathbf{a}(k)\end{bmatrix} \tag{2.44}$$

and

$$\begin{bmatrix}\mathbf{z}(k)\\ \mathbf{U}^{-\mathrm{T}}(k)\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{0}_{1\times N}\\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\end{bmatrix} \tag{2.45}$$

If we know the value of vector $\mathbf{a}(k)$ in (2.44) we can compute the rotation matrix, which can then be used in Equation (2.45) to compute $\mathbf{z}(k)$. By comparing equations

Table 2.4: *The IQRD-RLS Algorithm.*

```
for each $k$
{ Obtaining $\mathbf{a}(k)$
$\mathbf{a}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$
Obtaining $\mathbf{Q}_\theta(k)$ and $\gamma(k)$
$\begin{bmatrix} 1/\gamma(k) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 \\ -\mathbf{a}(k) \end{bmatrix}$
Obtaining $\mathbf{z}(k)$ and updating $\mathbf{U}^{-\mathrm{T}}(k)$
$\begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k) \\ \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix}$
Obtain $e(k)$:
$e(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)$
Updating the coefficient vector:
$\mathbf{w}(k) = \mathbf{w}(k-1) - e(k)\gamma(k)\mathbf{z}(k)$
}
```

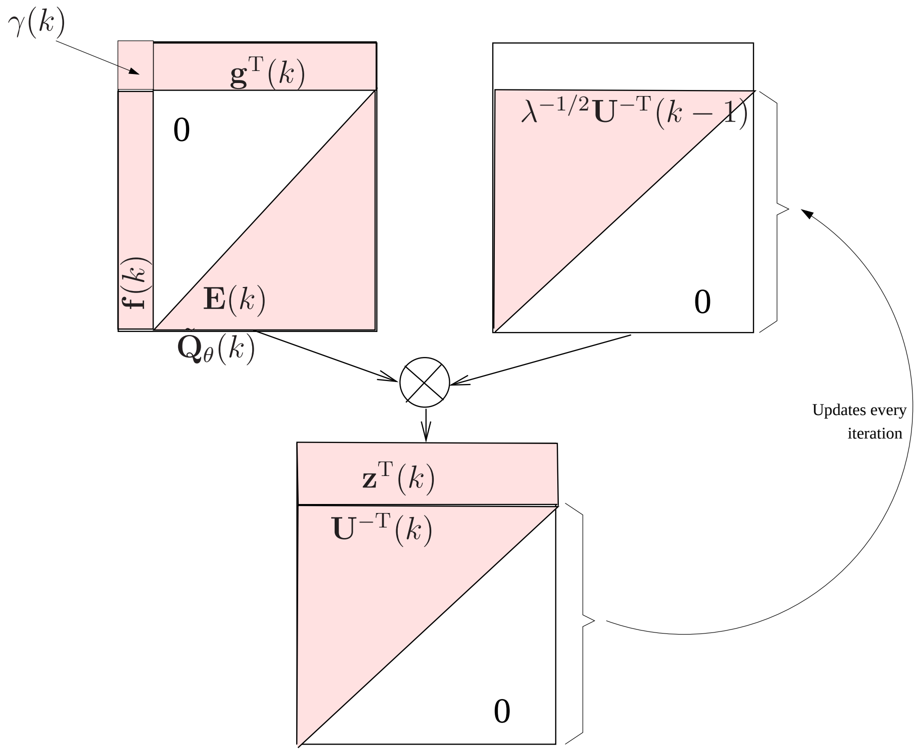
(2.35), (2.40) and (2.45) we can write

$$\mathbf{w}(k) = \mathbf{w}(k-1) - e(k)\gamma(k)\mathbf{z}(k) \tag{2.46}$$

Vector $\mathbf{z}(k)$ is used to update the weight vector. The recursions of the IQRD-RLS algorithm is summarized in Table 2.4.

Figure 2.7 shows the structure of the matrices involved in the update Equation (2.45) when the matrix $\mathbf{U}(k)$ is lower triangular. It can be noticed from the figure that $\mathbf{U}^{-\mathrm{T}}(k)$ is an upper triangular matrix, and the rotation matrix has the same structure as in the QRD-RLS algorithm. The advantage of using IQRD-RLS algorithm is that the computationally expensive backward-substitution step is avoided in the computation of the weight vector. The exact computational complexity of the IQRD-RLS is less than that of the QRD-RLS. However, both algorithms have complexity of $\mathcal{O}(N^2)$.

The properties of the IQRD-RLS algorithm are given as follows:



$\gamma(k)$
$\mathbf{g}^{\mathrm{T}}(k)$
0
$\mathbf{f}(k)$
$\mathbf{E}(k)$
$\tilde{\mathbf{Q}}_{\theta}(k)$
$\lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)$
0
Updates every iteration
$\mathbf{z}^{\mathrm{T}}(k)$
$\mathbf{U}^{-\mathrm{T}}(k)$
0

Figure 2.7: The update for $\mathbf{U}^{-T}(k)$ in IQRD-RLS algorithm.

Table 2.5: Computational complexity of the RLS, QRD-RLS and IQRD-RLS algorithms.

Algorithm	ADD	MULT	DIV	SQRT
RLS	$2N^2 + 2N$	$3N^2 + 3N$	N	0
QRD-RLS	$2(N^2 + N)$	$4(N^2 + N)$	1	N
IQRD-RLS	$5N^2 + 6N$	$3N^2 + 4N + 1$	N	N
Backward Substitution	$(N^2 + 3N)/2$	$(N^2 + N)/2$	N	0

1. *Rate of convergence.* The IQRD-RLS algorithm has excellent convergence properties, the convergence path is identical to the RLS algorithm provided that both algorithms are initialized in the same way.

2. *Robustness.* The IQRD-RLS algorithm is robust in infinite and finite precision environment.

3. *Computational complexity.* It requires $\mathcal{O}(N^2)$ computations.

4. *Numerical stability.* The IQRD-RLS algorithm is numerically stable. [23]

The IQRD-RLS algorithm is considered better than the RLS and the QRD-RLS algorithms because it is stable and does not require extra computations for the weight vector. Even though, the QRD-RLS and the IQRD-RLS algorithms have a complexity of $\mathcal{O}(N^2)$, the latter turns out to be less expensive in terms of number of divisions. The number of operations required for each algorithm are given in Table 2.5.

Chapter 3

Fast QRD-RLS Algorithms

The QRD-RLS based algorithms exhibit numerical robustness, fast convergence, and computational complexity of $\mathcal{O}(N^2)$. High computational complexity is due to the evaluation of the Cholesky factor at each iteration, which makes the algorithm impractical for implementations involving high-order systems. However, a number of low-complexity derivatives of QRD-RLS algorithm have been successfully discovered, such as the least-squares lattice algorithm [25], the fixed-order fast adaptive ROTOR's algorithm [7], and the backward and forward prediction based algorithms [8–10]. These algorithms have complexity of $\mathcal{O}(N)$ and a convergence speed identical to that of the QRD-RLS algorithm provided that the initialization is equivalent. In this thesis we focus on forward and backward prediction based algorithms that are commonly known as fast QRD-RLS (FQRD-RLS) *a priori* and *a posteriori* algorithms [8, 26].

The main idea in FQRD-RLS algorithms is to identify vectors that can exploit the underlying time-shift structure of the input data vector [9]. The vectors identified are generally related to the Cholesky factor matrix and the input data matrix. As a result, the Cholesky factor update equation involves vectors instead of matrices and, therefore, reduces the computational complexity by one order of magnitude. These vector-based update equations for FQRD-RLS algorithms are derived from the forward and backward prediction algorithms. The introduction of the FQRD-RLS algorithm is therefore incomplete without discussing the forward and backward prediction problems.

An introduction to important variables involved in the FQRD-RLS algorithm is given in Section 3.1. The forward and backward prediction algorithms are introduced in Section 3.2. The FQRD-RLS algorithms based on forward and backward prediction are then derived in Section 3.4 and 3.3, respectively. It is known from the literature that the backward prediction based algorithms are of minimal complexity and backward stable [27], [10]. Consequently, more focus is given on these algorithms, and the forward prediction algorithms are only introduced for the sake of completeness. Finally, the complexity analysis is presented in which the number of multiplications, additions, divisions and square-roots, for each algorithm, are presented in form of a Table.

3.1 FQRD-RLS algorithm: Introduction to key variables

The idea in the FQRD-RLS algorithms is to replace matrix update equations with vector update equations in order to reduce computational complexity. It is observed that the matrix-based update equations in the QRD-RLS and IQRD-RLS algorithms involve Cholesky factor $\mathbf{U}(k)$ (see Eq. (2.29) and (2.45)). This means that the vector update equation, to be used in the FQRD-RLS algorithms, must also be related to the Cholesky factor. The two vectors identified for the vector-based update equations $\mathbf{a}(k)$ and $\mathbf{f}(k)$, partitions of the rotation matrix $\mathbf{Q}_\theta(k)$, are defined as

$$\mathbf{a}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \tag{3.1}$$

and

$$\mathbf{f}(k) = \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k) \tag{3.2}$$

Note that both the vectors not only involve the matrix $\mathbf{U}^{-\mathrm{T}}(k)$ but also the data vector $\mathbf{x}(k)$, which means updating the vectors results in updating both variables. Furthermore, either of vectors $\mathbf{a}(k)$ or $\mathbf{f}(k)$ can be used to compute the rotation matrix $\mathbf{Q}_\theta(k)$, either from

$$\begin{bmatrix} 1/\gamma(k) \\ \mathbf{0}_{N\times 1} \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 \\ -\mathbf{a}(k) \end{bmatrix} \tag{3.3}$$

or

$$\begin{bmatrix} 1 \\ \mathbf{0}_{N\times 1} \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k) \begin{bmatrix} \gamma(k) \\ \mathbf{f}(k) \end{bmatrix} \tag{3.4}$$

Finally, it is important to mention that updating vector $\mathbf{a}(k)$ leads to FQRD-RLS *a priori* (FQR_PRI) algorithms [8], and updating vector $\mathbf{f}(k)$ leads to FQRD-RLS *a posteriori* (FQR_POS) algorithms [8].

3.2 Prediction Problem

In prediction problems we want to estimate a future sample from the past values. Similarly, we can also estimate a past value from the future values. The former is called the *forward prediction* and the latter is known as the *backward prediction.* This concept is depicted in Figure 3.1. The figure shows the extended data vector $\mathbf{x}^{N+1}(k)$. The values in the subvector $\mathbf{x}(k-1)$ are used to predict the future sample $x(k)$ and similarly the values in the subvector $\mathbf{x}(k)$ are used to predict the past sample $x(k-N-1)$.

The estimated sample differs from the actual sample by an estimation error value. In prediction problems, we associate a cost function with the estimation error and try to minimize it. Here the cost function is the weighted least-squares error, which means minimization of the norm of the estimation error vector (see Section 2.4.1). Next, we introduce the backward and forward prediction algorithms.

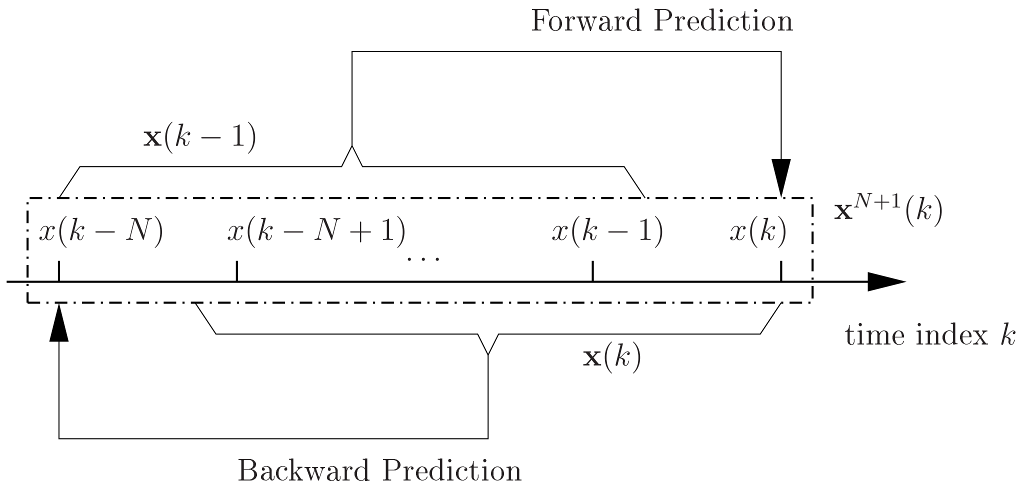


Figure 3.1: The concepts of forward and backward prediction

3.2.1 Backward Prediction

In backward prediction we have a finite set of values taken from a data sequence and we want to estimate the preceding sample value using the values in the set. We generally consider a set of N data values and the estimate is obtained using their weighted sum. A weighted least-squares solution based on the QRD-RLS algorithm for the backward prediction is presented here. Considering Eq. (2.12), the weighted backward prediction error vector is written as

$$\mathbf{e}_b(k) = \mathbf{d}_b(k) - \mathbf{X}(k)\mathbf{w}_b(k) \tag{3.5}$$

where

$$\mathbf{d}_b(k) = \begin{bmatrix} x(k-N) & \lambda^{1/2}x(k-N-1) & \dots & \lambda^{(k-N-1)/2}x(0) & \mathbf{0}_{1\times(N)} \end{bmatrix}^{\mathrm{T}} \tag{3.6}$$

and $\mathbf{w}_b(k)$ is the backward prediction coefficient vector. The above equation is written in compact form as

$$\begin{aligned}\mathbf{e}_b(k) &= \begin{bmatrix}\mathbf{X}(k) & \mathbf{d}_b(k)\end{bmatrix}\begin{bmatrix}-\mathbf{w}_b(k)\\ 1\end{bmatrix}\\ &= \mathbf{X}^{(N+1)}(k)\begin{bmatrix}-\mathbf{w}_b(k)\\ 1\end{bmatrix}\end{aligned} \tag{3.7}$$

where

$$\mathbf{X}^{(N+1)}(k) = \begin{bmatrix}\mathbf{X}(k) & \mathbf{d}_b(k)\end{bmatrix} \tag{3.8}$$

Multiplying Eq. (3.7) with $\tilde{\mathbf{Q}}_\theta(k)$ on both sides results in

$$\mathbf{e}_{bq}(k) = \tilde{\mathbf{Q}}_\theta(k)\mathbf{e}_b(k) = \begin{bmatrix}\mathbf{0}_{(k+1-N)\times N} & \mathbf{e}_{bq1}(k)\\ \mathbf{U}(k) & \mathbf{d}_{bq2}(k)\end{bmatrix}\begin{bmatrix}-\mathbf{w}_b(k)\\ 1\end{bmatrix} \tag{3.9}$$

Similar to Eq. (2.36), the update equation for the vector $\mathbf{d}_{bq2}(k)$ is obtained

$$\begin{bmatrix}e_{bq1}(k)\\ \mathbf{d}_{bq2}(k)\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}d_b(k)\\ \lambda^{1/2}\mathbf{d}_{bq2}(k-1)\end{bmatrix} \tag{3.10}$$

This equation is used in the detailed derivation of the FQRD-RLS algorithms. The least-squares solution for the prediction coefficients that minimizes the backward prediction error is given as

$$\mathbf{w}_b(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{bq2}(k) \tag{3.11}$$

where matrix $\mathbf{U}(k)$ is the same as the one used in the derivation of QRD-RLS algorithm in Section 2.4.2 Eq. (2.22). Finally, we define the *a priori* backward prediction error as

$$e_b(k) = x(k-N) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}_b(k-1) \tag{3.12}$$

and the *a posteriori* backward prediction error as

$$\varepsilon_b(k) = x(k-N) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}_b(k) \tag{3.13}$$

Both the definitions will be referred to in the derivation of the FQRD-RLS algorithms.

3.2.2 Forward Prediction

In forward prediction, we try to estimate a future sample value from the given sample values from the same sequence at the current time instant. Similarly to Eq. (2.12), the weighted forward prediction error vector is written as

$$\mathbf{e}_f(k) = \mathbf{d}_f(k) - \begin{bmatrix} \mathbf{X}(k+1) \\ \mathbf{0}_{1\times N} \end{bmatrix} \mathbf{w}_f(k) \tag{3.14}$$

where

$$\mathbf{d}_f(k) = \begin{bmatrix} x(k) & \lambda^{1/2}x(k-1) & \dots & \lambda^{k/2}x(0) \end{bmatrix}^{\mathrm{T}} \tag{3.15}$$

and $\mathbf{w}_f(k)$ is the forward prediction coefficient vector. The above equation can also be written in compact form as

$$\begin{aligned} \mathbf{e}_f(k) &= \begin{bmatrix} \mathbf{d}_f(k) & \begin{pmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times N} \end{pmatrix} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \\ &= \mathbf{X}^{(N+1)}(k) \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \end{aligned} \tag{3.16}$$

where

$$\mathbf{X}^{(N+1)}(k) = \begin{bmatrix} \mathbf{d}_f(k) & \begin{pmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times N} \end{pmatrix} \end{bmatrix} \tag{3.17}$$

Multiplying Eq. (3.16) with $\begin{bmatrix} \tilde{\mathbf{Q}}_\theta(k-1) & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 1 \end{bmatrix}$ on both sides results in

$$\mathbf{e}_{f_q}(k) = \begin{bmatrix} \tilde{\mathbf{Q}}_\theta(k-1) & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 1 \end{bmatrix} \mathbf{e}_f(k) = \begin{bmatrix} \mathbf{e}_{f_{q1}}(k) & \mathbf{0}_{(k-N)\times(N)} \\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1) \\ \lambda^{k/2}x(0) & \mathbf{0}_{1\times N} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix} \tag{3.18}$$

Similar to Eq. (2.36), the update equation for the vector $\mathbf{d}_{f_{q2}}(k)$ is obtained

$$\begin{bmatrix} e_{f_{q1}}(k) \\ \mathbf{d}_{f_{q2}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} d_f(k) \\ \lambda^{1/2}\mathbf{d}_{f_{q2}}(k-1) \end{bmatrix} \tag{3.19}$$

where $d_f(k) = x(k)$. The above equations are used in the detailed derivation of the fast QRD-RLS algorithms. The solution for the coefficient vector in this case is given by

$$\mathbf{w}_f(k) = \mathbf{U}^{-1}(k-1)\mathbf{d}_{f_{q2}}(k) \tag{3.20}$$

Finally, we define the *a priori* forward prediction error as

$$e_f(k) = x(k) - \mathbf{x}^{\mathrm{T}}(k-1)\mathbf{w}_f(k-1) \tag{3.21}$$

and the *a posteriori* forward prediction error as

$$\varepsilon_f(k) = x(k) - \mathbf{x}^{\mathrm{T}}(k-1)\mathbf{w}_f(k) \tag{3.22}$$

Both the definitions will be referred to in the derivation of the FQRD-RLS algorithms. It is important to note here that both the forward and the backward prediction algorithms involve exactly the same data matrix $\mathbf{X}^{(N+1)}(k)$. This fact will be used in the derivation of the update equations for the FQRD-RLS algorithms.

3.3 Fast QRD-RLS backward prediction algorithms

In this section the FQRD-RLS backward prediction algorithm is derived. Consider the extended input data matrix $\mathbf{X}^{(N+1)}(k)$. The lower triangular Cholesky factor matrix $\mathbf{U}^{(N+1)}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ for the extended data matrix $\mathbf{X}^{(N+1)}(k)$ is obtained as follows

$$\begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} = \tilde{\mathbf{Q}}^{(N+1)}(k)\mathbf{X}^{(N+1)}(k) \tag{3.23}$$

where the matrix $\tilde{\mathbf{Q}}^{(N+1)}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ is the extended rotation matrix. Using the definition of $\mathbf{X}^{(N+1)}(k)$ from Eq. (3.8) we get

$$\begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} = \tilde{\mathbf{Q}}^{(N+1)}(k) \begin{bmatrix} \mathbf{X}(k) & \mathbf{d}_b(k) \end{bmatrix} \tag{3.24}$$

Similarly, Eq. (3.17) leads to

$$\begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} = \tilde{\mathbf{Q}}^{(N+1)}(k) \begin{bmatrix} \mathbf{d}_f(k) & \begin{pmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times N} \end{pmatrix} \end{bmatrix} \tag{3.25}$$

Next we write the rotation matrix $\tilde{\mathbf{Q}}^{(N+1)}(k)$ as a sequential product of rotation matrices as follows

$$\begin{aligned} \tilde{\mathbf{Q}}^{(N+1)}(k) &= \mathbf{Q}'_b(k)\tilde{\mathbf{Q}}_\theta(k) \\ \tilde{\mathbf{Q}}^{(N+1)}(k) &= \tilde{\mathbf{Q}}_f(k)\mathbf{Q}'_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}_\theta(k-1) & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 1 \end{bmatrix} \end{aligned} \tag{3.26}$$

where the subscripts b and f denote backward and forward prediction, respectively. Applying the definitions of the rotation matrix to the Eqs. (3.24) and (3.25) results in

$$\begin{aligned} \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} &= \mathbf{Q}'_b(k)\tilde{\mathbf{Q}}_\theta(k) \begin{bmatrix} \mathbf{X}(k) & \mathbf{d}_b(k) \end{bmatrix} \\ &= \mathbf{Q}'_b(k) \begin{bmatrix} \mathbf{0}_{(k-N+1)\times N} & \mathbf{e}_{b_{q1}}(k) \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} \end{aligned} \tag{3.27}$$

and

$$\begin{aligned} \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} &= \tilde{\mathbf{Q}}_f(k)\mathbf{Q}'_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}_\theta(k-1) & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{d}_f(k) & \begin{pmatrix} \mathbf{X}(k-1) \\ \mathbf{0}_{1\times N} \end{pmatrix} \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_f(k)\mathbf{Q}'_f(k) \begin{bmatrix} \mathbf{e}_{f_{q1}}(k) & \mathbf{0}_{(k-N)\times N} \\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1) \\ \lambda^{k/2}x(0) & \mathbf{0}_{1\times N} \end{bmatrix} \end{aligned} \tag{3.28}$$

The lower triangularization in Eq. (3.27) is complete when the rotation matrix $\mathbf{Q}'_b(k)$ is applied. The rotation matrix operates only on the error vector $\mathbf{e}_{b_{q1}}(k)$ in the equation, resulting in

$$\begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{(k-N)\times N} & \mathbf{0}_{(k-N)\times 1} \\ \mathbf{0}_{1\times N} & \|\mathbf{e}_b(k)\| \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} \tag{3.29}$$

In Eq. (3.27), the rotation matrix $\mathbf{Q}'_f(k)$ operates on the error vector $\mathbf{e}_{fq1}(k)$ and $\lambda^{1/2}x(0)$ and rotates the vector so that the magnitude of the vector is obtained in the last element of the vector and rest of the elements are zero.

$$\begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \mathbf{0}_{(k-N)\times 1} & \mathbf{0}_{(k-N)\times N} \\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1) \\ \|\mathbf{e}_f(k)\| & \mathbf{0}_{1\times N} \end{bmatrix} \tag{3.30}$$

>From Eq. (3.30) and (3.29) we can write

$$\begin{bmatrix} \mathbf{0}_{(k-N)\times N} & \mathbf{0}_{(k-N)\times 1} \\ \mathbf{0}_{1\times N} & \|\mathbf{e}_b(k)\| \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_f(k) \begin{bmatrix} \mathbf{0}_{(k-N)\times 1} & \mathbf{0}_{(k-N)\times N} \\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1) \\ \|\mathbf{e}_f(k)\| & \mathbf{0}_{1\times N} \end{bmatrix} \tag{3.31}$$

The increasing order of the equation is avoided by removing the redundant zeros and the corresponding rows and columns from the rotation matrix, resulting in

$$\begin{bmatrix} \mathbf{0}_{1\times N} & \|\mathbf{e}_b(k)\| \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1) \\ \|\mathbf{e}_f(k)\| & \mathbf{0}_{N\times 1} \end{bmatrix} \tag{3.32}$$

It is assumed that the inverse of the matrices involved in the above equation always exists. Taking the inverse transpose of both sides of Eq. (3.32) results in

$$\begin{bmatrix} \frac{-\mathbf{d}^{\mathrm{T}}_{b_{q2}}(k)\mathbf{U}^{-\mathrm{T}}(k)}{\|\mathbf{e}_b(k)\|} & \frac{1}{\|\mathbf{e}_b(k)\|} \\ \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{0}_{N\times 1} \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-1) \\ \frac{1}{\|\mathbf{e}_f(k)\|} & \frac{-\mathbf{d}^{\mathrm{T}}_{f_{q2}}(k)\mathbf{U}^{-\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k)\|} \end{bmatrix} \tag{3.33}$$

>From here on there are two approaches to follow. Depending on whether we derive an update equation for the vector $\mathbf{f}(k)$ or $\mathbf{a}(k)$, defined in (3.2) and (3.1), we set different versions of the FQRD-RLS algorithm. Updating $\mathbf{a}(k)$ results in the so-called *a priori* algorithm, while updating $\mathbf{f}(k)$ results in the so-called *a posteriori* algorithm. Due to its stability [26], the backward algorithm is treated in detail. The forward algorithm is briefly addressed in the following section only for the sake of completeness, and will not be used further in the thesis.

3.3.1 Fast QRD-RLS *a posteriori* backward prediction algorithm

The FQR_POS_B algorithm updates vector $\mathbf{f}(k)$. The update equation is obtained by multiplying Eq. (3.33) with the extended input data vector $\mathbf{x}^{N+1}(k)$:

$$\begin{bmatrix} \frac{-\mathbf{d}_{b_{q2}}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k)}{\|\mathbf{e}_b(k)\|} + \frac{\mathbf{x}(k-N)}{\|\mathbf{e}_b(k)\|} \\ \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k-1) \\ \frac{x(k)}{\|\mathbf{e}_f(k)\|} - \frac{\mathbf{d}_{f_{q2}}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k-1)}{\|\mathbf{e}_f(k)\|} \end{bmatrix} \tag{3.34}$$

Using the definition of $\mathbf{f}(k)$ and the *a posteriori* forward and backward errors from Eqs. (3.2), (3.13) and (3.22) we get

$$\begin{bmatrix} \frac{\varepsilon_b(k)}{\|\mathbf{e}_b(k)\|} \\ \mathbf{f}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{f}(k-1) \\ \frac{\varepsilon_f(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix} \tag{3.35}$$

where the vector $\mathbf{f}(k-1)$ is assumed to be known from the previous iteration, and the *a posteriori* forward prediction error value $\varepsilon_f(k)$ can be computed from $e_{fq1}(k)$ as follows

$$\varepsilon_f(k) = e_{fq1}(k)\gamma(k) \tag{3.36}$$

and $e_{fq1}(k)$ is computed from Eq. (3.19).

There are two ways for solving Eq. (3.35)

1. $\varepsilon_f(k)$ is not known, the last element of vector $\mathbf{f}(k)$ is known to be $\frac{x(k)}{\|\mathbf{e}_f^{(0)}(k)\|}$. The algorithm based on this way is called FQR_POS_B Version 1.

2. Compute $\varepsilon_f(k)$ from Eq. (3.36). No knowledge of the variables on the left is required. The algorithm based on this approach is called FQR_POS_B Version 2.

In version 1, $\|\mathbf{e}_f^{(0)}(k)\|$ is given by the relation

$$\begin{bmatrix} \mathbf{0}_{N\times 1} \\ \|\mathbf{e}_f^{(0)}(k)\| \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix} \tag{3.37}$$

However, a simplified expression for $\|\mathbf{e}_f^{(0)}(k)\|$ exists and can be obtained by first noting that the above equation gives the first column vector on both sides of Eq. (3.32). Furthermore, it is known that the rotation matrix $\mathbf{Q}_{\theta f}(k)$ results in lower triangularization of the right hand side. Therefore only the last element of the first column vector on the left hand side is non-zero. This is only possible if and only if

$$\|\mathbf{e}_f^{(0)}(k)\| = \sqrt{\|\mathbf{d}_{fq2}(k)\|^2 + \|\mathbf{e}_f(k)\|^2} \tag{3.38}$$

The updated $\mathbf{f}(k)$ is then used for updating rotation matrix $\mathbf{Q}_\theta(k)$ according to Eq. (3.4). >From the rotation matrix computed using the above equation, the vector $\mathbf{f}(k)$ is updated. Final Equation (2.36) is used to obtain $e_{q1}(k)$ from which the *a posteriori* error is computed as

$$\varepsilon(k) = e_{q1}(k)\gamma(k) \tag{3.39}$$

The complete FQR_POS_B algorithm is given in Table 3.1.

3.3.2 Fast QRD-RLS *a priori* backward prediction algorithm

The FQR_PRI_B algorithm is obtained in a similar way to that of the FQR_POS_B algorithm. The only difference is that the time index of Eq. (3.33) is decremented by

Table 3.1: FQR_POS_B Algorithm based on backward prediction errors.

```
for each $k$
{ 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 $\|\mathbf{e}_f(k)\|$:
    $\|\mathbf{e}_f(k)\| = \sqrt{e_{fq1}^2(k) + \lambda\|\mathbf{e}_f(k-1)\|^2}$
  Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k)$:
    $\begin{bmatrix} \mathbf{0} \\ \mathbf{e}_f^{(0)}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix}$
  Obtaining $\mathbf{f}(k)$
    $\begin{bmatrix} \frac{\varepsilon_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k)\|} \\ \mathbf{f}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{f}(k-1) \\ \frac{\varepsilon_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k)\|} \end{bmatrix}$
  Obtaining $\mathbf{Q}_\theta(k)$:
    $\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \gamma(k) \\ \mathbf{f}(k) \end{bmatrix}$
  Joint Process Estimation:
    $\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
    $\varepsilon(k) = e_{q1}(k)\gamma(k)$
}
```

one and that the resulting equation is multiplied with the scaled input data vector $\mathbf{x}^{N+1}(k)/\lambda^{1/2}$. As a result of these two operations, we get

$$\begin{bmatrix} \frac{-\mathbf{d}_{b_{q2}}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} + \frac{\mathbf{x}(k-N)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \frac{\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)}{\lambda^{1/2}} \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \frac{\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)}{\lambda^{1/2}} \\ \frac{x(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} - \frac{\mathbf{d}_{fq2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)}{\lambda^{1/2}\|\mathbf{e}_f(k)\|} \end{bmatrix} \tag{3.40}$$

Using the definition of $\mathbf{a}(k)$, $e_b(k)$ and $e_f(k)$ from Eq. (3.1), (3.12), and (3.21), the above expression in compact form is written as

$$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{3.41}$$

The *a priori* forward prediction error $e_f(k)$ is computed using,

$$e_f(k) = e_{fq1}(k)/\gamma(k) \tag{3.42}$$

The rotation matrix $\mathbf{Q}_\theta(k)$ is then updated using Eq. (3.3). Following the same steps as in the FQR_POS_B algorithm, we finally get the *a posteriori* error value using Eq. (3.39). The FQR_PRI_B algorithm is given in Table 3.2. Similar to the FQR_POS_B algorithm there are two versions of FQR_PRI_B algorithm. Again the derivation of the two algorithms is done following the same steps as for the FQR_POS_B algorithm.

3.4 Fast QRD-RLS forward prediction algorithms

The steps for the derivation of the FQRD-RLS algorithms based on forward prediction are similar to those of the FQRD-RLS backward algorithms, with the exception that the matrix $\mathbf{U}(k)$ is an upper triangular matrix and the rotation matrix $\mathbf{Q}^{(N+1)}(k)$ is defined with a different sequence of rotation matrices. Therefore, only the important equations that elaborate the difference of the backward algorithm from the forward algorithm are mentioned.

In the forward algorithms we change the definition of the rotation matrix $\tilde{\mathbf{Q}}_\theta^{(N+1)}(k)$ to

$$\begin{aligned} \tilde{\mathbf{Q}}^{(N+1)}(k) &= \tilde{\mathbf{Q}}_b(k)\mathbf{Q}'_b(k)\tilde{\mathbf{Q}}_\theta(k) \\ \tilde{\mathbf{Q}}^{(N+1)}(k) &= \mathbf{Q}'_f(k) \begin{bmatrix} \tilde{\mathbf{Q}}_\theta(k-1) & \mathbf{0}_{N\times 1} \\ \mathbf{0}_{1\times N} & 1 \end{bmatrix} \end{aligned} \tag{3.43}$$

Table 3.2: FQR_PRI_B based on backward prediction errors.

```
for each $k$
{ 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 $\mathbf{a}(k)$
    $\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
    Obtaining $\|\mathbf{e}_f(k)\|$:
    $\|\mathbf{e}_f(k)\| = \sqrt{e_{fq1}^2(k) + \lambda\|\mathbf{e}_f(k-1)\|^2}$
    Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k)$:
    $\begin{bmatrix} \mathbf{0} \\ \|\mathbf{e}_f^{(0)}(k)\| \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix}$
    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:
    $\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
    $\varepsilon(k) = e_{q1}(k)\gamma(k)$
}
```

Also the Cholesky factor matrix is lower triangular in this case. Therefore, Eqs. (3.24) and (3.25) result in

$$\begin{aligned} \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}^{(N+1)}(k) \end{bmatrix} &= \tilde{\mathbf{Q}}_b(k)\mathbf{Q}_b'(k)\tilde{\mathbf{Q}}_\theta(k) \begin{bmatrix} \mathbf{X}(k) & \mathbf{d}_b(k) \end{bmatrix} \\ &= \tilde{\mathbf{Q}}_b(k)\mathbf{Q}_b'(k) \begin{bmatrix} \mathbf{0}_{(k+1-N)\times N} & \mathbf{e}_{b_{q1}}(k) \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} \end{aligned} \qquad (3.44)$$

and

$$\begin{aligned}\begin{bmatrix}\mathbf{0}_{(k-N)\times(N+1)}\\ \mathbf{U}^{(N+1)}(k)\end{bmatrix} &= \mathbf{Q}'_f(k)\begin{bmatrix}\tilde{\mathbf{Q}}_\theta(k-1) & \mathbf{0}_{N\times 1}\\ \mathbf{0}_{1\times N} & 1\end{bmatrix}\begin{bmatrix}\mathbf{d}_f(k) & \begin{pmatrix}\mathbf{X}(k-1)\\ \mathbf{0}_{1\times N}\end{pmatrix}\end{bmatrix}\\ &= \mathbf{Q}'_f(k)\begin{bmatrix}\mathbf{e}_{f_{q1}}(k) & \mathbf{0}_{(k-N)\times N}\\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1)\\ \lambda^{k/2}x(0) & \mathbf{0}_{1\times N}\end{bmatrix}\end{aligned} \tag{3.45}$$

Next we apply the rotation matrix $\mathbf{Q}'_b(k)$ and $\tilde{\mathbf{Q}}'_f(k)$ to their respective equations. From the derivation of the backward algorithm it is known that matrix $\mathbf{Q}'_b(k)$ operates only on the vector $\mathbf{e}_{bq1}(k)$. Similarly, the rotation matrix $\mathbf{Q}'_f(k)$ operates on $\mathbf{e}_{f_{q1}}(k)$ and $x(0)$. Both the matrices rotate the vectors so that only one element in the vector remains non-zero. Following the derivation of the backward algorithm similar to Eq. (3.29) and (3.30), we can write

$$\begin{bmatrix}\mathbf{0}_{(k-N)\times(N+1)}\\ \mathbf{U}^{(N+1)}(k)\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{(k-N)\times 1} & \mathbf{0}_{(k-N)\times N}\\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1)\\ \|\mathbf{e}_f(k)\| & \mathbf{0}_{1\times N}\end{bmatrix} \tag{3.46}$$

and

$$\begin{bmatrix}\mathbf{0}_{(k-N-1)\times(N+1)}\\ \mathbf{U}^{(N+1)}(k)\end{bmatrix} = \tilde{\mathbf{Q}}_b(k)\begin{bmatrix}\mathbf{0}_{(k-N)\times N} & \mathbf{0}_{(k-N)\times 1}\\ \mathbf{0}_{1\times N} & \|\mathbf{e}_b(k)\|\\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k)\end{bmatrix} \tag{3.47}$$

By comparing Eq. (3.46) and Eq. (3.47) we get

$$\begin{bmatrix}\mathbf{0}_{(k-N)\times 1} & \mathbf{0}_{(k-N)\times N}\\ \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1)\\ \|\mathbf{e}_f(k)\| & \mathbf{0}_{1\times N}\end{bmatrix} = \tilde{\mathbf{Q}}_b(k)\begin{bmatrix}\mathbf{0}_{(k-N)\times N} & \mathbf{0}_{(k-N)\times 1}\\ \mathbf{0}_{1\times N} & \|\mathbf{e}_b(k)\|\\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k)\end{bmatrix} \tag{3.48}$$

We can get fixed-order matrices by removing the rows and columns contributing to

Table 3.3: Computational complexity of FQRD-RLS algorithms.

Algorithm	ADD	MULT	DIV	SQRT
FQR_POS_F	$10N+3$	$26N+10$	$3N+2$	$2N+1$
FQR_PRI_F	$10N+3$	$26N+11$	$4N+4$	$2N+1$
FQR_POS_B (VERSION 1)	$8N+1$	$19N+4$	$4N+1$	$2N+1$
FQR_POS_B (VERSION 2)	$8N+1$	$20N+5$	$3N+1$	$2N+1$
FQR_PRI_B (VERSION 1)	$8N-1$	$19N+2$	$5N+1$	$2N+1$
FQR_PRI_B (VERSION 2)	$8N+1$	$20N+6$	$4N+2$	$2N+1$

the increasing number of zeros:

$$\begin{bmatrix} \mathbf{d}_{f_{q2}}(k) & \mathbf{U}(k-1) \\ \|\mathbf{e}_f(k)\| & \mathbf{0}_{1\times N} \end{bmatrix} = \mathbf{Q}_{\theta b}(k) \begin{bmatrix} \mathbf{0}_{1\times N} & \|\mathbf{e}_b(k)\| \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} \tag{3.49}$$

Finally, taking the inverse transpose of both sides with the assumption that the inverse always exists, we write

$$\begin{bmatrix} \mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-1) \\ \frac{1}{\|\mathbf{e}_f(k)\|} & \frac{-\mathbf{d}^{\mathrm{T}}_{f_{q2}}(k)\mathbf{U}^{-\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k)\|} \end{bmatrix} = \mathbf{Q}_{\theta b}(k) \begin{bmatrix} \frac{-\mathbf{d}^{\mathrm{T}}_{b_{q2}}(k)\mathbf{U}^{-\mathrm{T}}(k)}{\|\mathbf{e}_b(k)\|} & \frac{1}{\|\mathbf{e}_b(k)\|} \\ \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{0}_{N\times 1} \end{bmatrix} \tag{3.50}$$

This equation is the basis for the forward algorithms. The next step will be the derivation of the *a priori* version based on vector $\mathbf{a}(k)$ and the *a posteriori* version based on vector $\mathbf{f}(k)$. In order to do so, we would need to multiply Eq. (3.33) with an appropriate input data matrix. As mentioned before, the forward algorithm is not discussed further.

The numbers of additions, multiplications, additions, and square-roots for the FQRD-RLS algorithms regarding one output sample are given in Table 3.3 as a function of N, the number of filter coefficients. From the table it becomes clear that all the algorithm are of complexity $\mathcal{O}(N)$. All algorithms show only small computational differences from each other, so that no single algorithm can be picked as most efficient in terms of computational complexity from the set. Nevertheless, backward algorithms are preferred due to their numerical robustness.

Chapter 4

Weight Extraction and Fixed Filtering Techniques for FQRD-RLS Algorithms

4.1 Introduction

The main limitation of the FQRD-RLS recursion is the fact that the coefficient vector of the adaptive filter is not known explicitly. As a consequence, applications like system identification and pre-equalizers discussed in Chapter 2 have not been associated with FQRD-RLS algorithms. In order to compensate for this shortcoming of the FQRD-RLS algorithm, we develop tools that enable the FQRD-RLS algorithm to be used in such applications. We first show in Section 4.2.1 how to extract the coefficient weights through the so-called *weight extraction* method. This weight extraction technique can be used in a system identification application to obtain the plant coefficients at any stage of convergence. Thereafter, in Section 4.2.2 a method for output filtering is discussed in which the adaptive filter coefficients are kept constant. We refer to this method as "output filtering for burst type training", since it can be used for periodic update applications, such as equalizer design in GSM systems. Section 4.2.3 describes an output filtering method referred to as "output filtering for pre-equalizer type setup" which enables filtering of a different sequence than the one associated with the filter update. The method is particularly useful in a pre-equalizer application. The computational complexity of the proposed tools in terms of multiplications, additions, divisions and square-roots is also discussed. Section 4.3 describes the experiments and illustrates the results. The three examples chosen in order to verify the proposed tools are system identification, pre-equalization and post-equalization. Section 4.4 draws conclusions from the experimental results.

4.2 Weight Extraction and output filtering techniques

4.2.1 Weight Extraction

The novel weight extraction (WE) technique to be presented in the following can be invoked at any iteration of the conventional FQRD-RLS algorithm. The internal variables of the FQRD-RLS algorithm at the time of interest are computed in a serial manner, i.e., N iterations for an N coefficient vector.

Consider the output of the adaptive filter $y(k)$ given as

$$y(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \tag{4.1}$$

Let us define $\boldsymbol{\delta}_i = \begin{bmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}^{\mathrm{T}}$ to be a vector of zeros containing a '1' at the i^{th} position. We can now get the i^{th} coefficient of vector $\mathbf{w}(k-1)$ as

$$w_i(k-1) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\boldsymbol{\delta}_i = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1) \tag{4.2}$$

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

Lemma 1. *Let $\mathbf{u}_i^T(k) = \begin{bmatrix} u_{i,0}(k) & \dots & u_{i,N-1}(k) \end{bmatrix}^T \in \mathbb{R}^{N\times 1}$ denote the i^{th} column of the upper triangular matrix $\mathbf{U}^{-T}(k) \in \mathbb{R}^{N\times N}$. Given $\mathbf{Q}_\theta(k-1) \in \mathbb{R}^{(N+1)\times(N+1)}$ from Table 4.1, then $\mathbf{u}_i(k-2)$ can be obtained from $\mathbf{u}_i(k-1)$ using the relation below*

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-2) \end{bmatrix} = \mathbf{Q}_\theta^T(k-1) \begin{bmatrix} z_i \\ \mathbf{u}_i(k-1) \end{bmatrix}, \quad i = 0, \dots, N-1 \tag{4.3}$$

where $z_i = -\mathbf{f}^T(k-1)\mathbf{u}_i(k-1)/\gamma(k-1)$.

Lemma 2. *Let $\mathbf{u}_i(k) = \begin{bmatrix} u_{i,0}(k) & \dots & u_{i,N-1}(k) \end{bmatrix}^T \in \mathbb{R}^{N\times 1}$ denote the i^{th} column of the upper triangular matrix $\mathbf{U}^{-T}(k-1) \in \mathbb{R}^{N\times N}$. Given $\tilde{\mathbf{Q}}_{\theta f}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ from Table 4.1, then $\mathbf{u}_i(k-1)$ can be obtained from $\mathbf{u}_{i-1}(k-2)$ using the following relation*

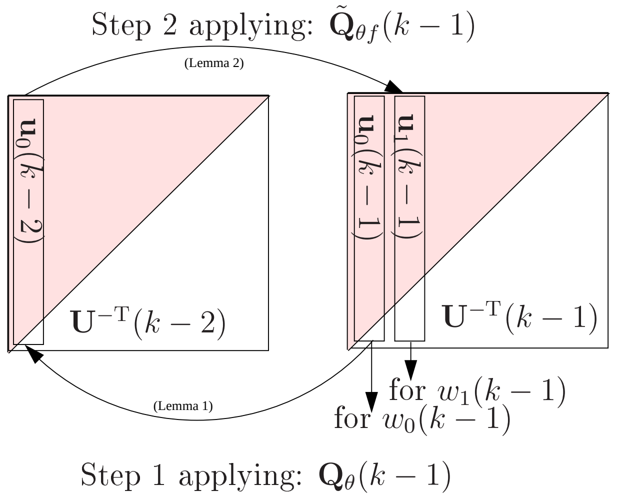
$$\begin{bmatrix} \frac{-w_{b,i-1}(k-1)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{u}_i(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i-1}(k-2) \\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}, \quad i = 0, \dots, N-1 \tag{4.4}$$

where

$$w_{f,i-1} = \begin{cases} -1 & \text{for } i = 0 \\ \mathbf{u}_{i-1}^T(k-2)\mathbf{d}_{fq2}(k-1) & \text{otherwise} \end{cases} \tag{4.5}$$

and $\mathbf{u}_{-1}(k-2) = \mathbf{0}_{N\times 1}$.

The proofs of Lemmas 1 and 2 are in the Appendices A1 and A2. Another set of alternative lemmas and their proofs are given in Appendices B1 and B2, these lemmas require division operations and are mentioned only for the sake of completeness. In order to extract the implicit weights $\mathbf{w}(k-1)$ of the FQRD-RLS, Lemma 2 is initialized with $\mathbf{u}_{-1}(k-2)$, and as a result we get column vector $\mathbf{u}_0(k-1)$. Lemma 1 is then invoked to compute column vector $\mathbf{u}_0(k-2)$. From $\mathbf{u}_0(k-2)$ we can compute $\mathbf{u}_1(k-1)$ using Lemma 2. By induction we can conclude that all $\mathbf{u}_i(k-1)$ can be obtained. The procedure is illustrated in Fig 4.1. As a consequence, the elements



Step 2 applying: $\tilde{\mathbf{Q}}_{\theta f}(k-1)$
(Lemma 2)
$\mathbf{u}_0(k-2)$
$\mathbf{U}^{-\mathrm{T}}(k-2)$
$\mathbf{u}_0(k-1)$
$\mathbf{u}_1(k-1)$
$\mathbf{U}^{-\mathrm{T}}(k-1)$
for $w_1(k-1)$
for $w_0(k-1)$
(Lemma 1)
Step 1 applying: $\mathbf{Q}_\theta(k-1)$

Figure 4.1: The procedure for updating $\mathbf{u}_i(k-1)$ *for weight extraction.*

of $\mathbf{w}(k-1)$ can be obtained from Eq. (4.2) in a serial manner. The WE algorithm is summarized in Table 4.1.

The number of operations required to completely extract all the coefficients is given in Table 4.2. For comparison, the computational cost of the FQRD-RLS algorithm based on *a priori* backward prediction errors and the Inverse QRD-RLS algorithm updates are also given.

4.2.2 Output filtering for burst type setup

The output filtering problem under consideration is illustrated in Figure 4.2. 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

$$y(k) = \begin{cases} \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) & k < k_F \\ \mathbf{w}_F^{\mathrm{T}}\tilde{\mathbf{x}}(k) & k \geq k_F \end{cases} \tag{4.6}$$

where $\mathbf{w}_F = \mathbf{w}(k_F - 1)$ is the coefficient vector of the adaptive filter "frozen" at time instant immediately before $k = k_F$ and $\tilde{x}(k)$ is the input signal for the time instant $k \geq k_F$.

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

If the FQRD-RLS algorithm is used for updating $\mathbf{w}(k)$, one alternative for carrying out the filtering for the data mode is to flush the FQRD-RLS filter coefficients (see Section 4.2.1) and, thereafter, perform the filtering of $\tilde{x}(k)$ with a simple transversal structure. Alternatively, to avoid the increased peak complexity of this operation, we can reuse the variables from the FQRD-RLS update at time instant $k = k_F - 1$ to reproduce the equivalent output signal without explicitly extracting the weights $\mathbf{w}_F$. For this purpose, the output after *weight freezing* is written as

$$\begin{aligned} y(k) &= \mathbf{d}_{q2}^{\mathrm{T}}(k_F - 1)\mathbf{U}^{-\mathrm{T}}(k_F - 1)\tilde{\mathbf{x}}(k) \\ &= \mathbf{d}_{q2}^{\mathrm{T}}(k_F - 1)\mathbf{r}(k), \;\; k \geq k_F \end{aligned} \tag{4.7}$$

where, $\mathbf{d}_{q2}(k_F - 1)$ is the vector $\mathbf{d}_{q2}(k)$ and $\mathbf{U}^{-\mathrm{T}}(k_F - 1)$ is the matrix $\mathbf{U}^{-\mathrm{T}}(k)$ at time instant $k = k_F - 1$, respectively, and $\mathbf{r}(k) = \mathbf{U}^{-\mathrm{T}}(k_F - 1)\tilde{\mathbf{x}}(k)$. The following lemmas are required in order to compute Eq. (4.7) without explicitly computing matrix $\mathbf{U}^{-\mathrm{T}}(k_F - 1)$.

Lemma 3. *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 4.3, then $\mathbf{U}^{-T}(k-2)\mathbf{x}(k)$ can be obtained from $\mathbf{U}^{-T}(k -$*

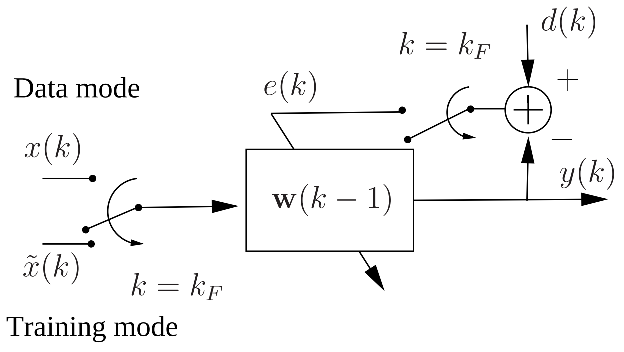


Figure 4.2: The output filtering with fixed weights using adaptive filter setup.

1)$\mathbf{x}(k)$ *using the relation below*

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-T}(k-2)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_\theta^T(k-1) \begin{bmatrix} z_i(k) \\ \mathbf{U}^{-T}(k-1)\mathbf{x}(k) \end{bmatrix} \tag{4.8}$$

where $z_i(k) = \frac{-\mathbf{f}^T(k-1)\mathbf{U}^{-T}(k-1)\mathbf{x}(k)}{\gamma(k)}$.

Lemma 4. *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* $\mathbf{Q}_{\theta f}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ *from Table 4.3, then* $\mathbf{U}^{-T}(k-1)\mathbf{x}(k)$ *can be obtained from* $\mathbf{U}^{-T}(k-2)\mathbf{x}(k-1)$ *using the following relation*

$$\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-T}(k-1)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-T}(k-2)\mathbf{x}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{4.9}$$

where $e_f(k) = x(k) - \mathbf{d}_{fq2}^T(k-1)\mathbf{U}^{-T}(k-2)\mathbf{x}(k-1)$.

The proofs for Lemmas 3 and 4 are in the Appendices A3 and A4. Another set of alternative lemmas and their proofs are given in Appendices B3 and B4, these lemmas require division operations and are mentioned only for the sake of completeness. 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

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\tilde{\mathbf{r}}(k-1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k_F-1) \begin{bmatrix} z_i(k) \\ \mathbf{r}(k-1) \end{bmatrix} \tag{4.10}$$

and

$$\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{r}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k_F - 1) \begin{bmatrix} \tilde{\mathbf{r}}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{4.11}$$

where $\tilde{\mathbf{r}}(k-1)$ is an intermediate value required to get $\mathbf{r}(k)$. At a time instant after $k = k_F$ we have

$$\mathbf{r}(k-1) = \mathbf{U}^{-\mathrm{T}}(k_F - 1)\tilde{\mathbf{x}}(k-1) \tag{4.12}$$

Applying Eq. (4.10) we get

$$\tilde{\mathbf{r}}(k-1) = \mathbf{U}^{-\mathrm{T}}(k_F - 2)\tilde{\mathbf{x}}(k-1) \tag{4.13}$$

and applying Eq. (4.11) to $\tilde{\mathbf{r}}(k-1)$, we have

$$\mathbf{r}(k) = \mathbf{U}^{-\mathrm{T}}(k_F - 1)\tilde{\mathbf{x}}(k) \tag{4.14}$$

The output $y(k)$ can then be obtained by using the updated $\mathbf{r}(k)$ in Eq. (4.7). Note that when $\mathbf{r}(k)$ is obtained from $\mathbf{r}(k-1)$, only the 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 detailed algorithm for the filtering operation is given in Table 4.3 along with the initialization procedure. Note that matrix multiplications are avoided; instead Givens rotations are used, rendering a low-complexity solution. 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 4.4 for comparison.

4.2.3 Output filtering for pre-equalizer type setup

Consider the pre-equalizer setup described in Section 2.2.3; for clarity the setup is illustrated in Figure 4.3. As discussed before, in the pre-equalizer setup, the weights of the adaptive filters are copied at each iteration to the pre-equalizer block with input signal $\tilde{x}(k)$ which is independent of the input to the adaptive filter $x(k)$. The output $\tilde{y}(k)$ is then computed with the updated copy of the weight vector. The

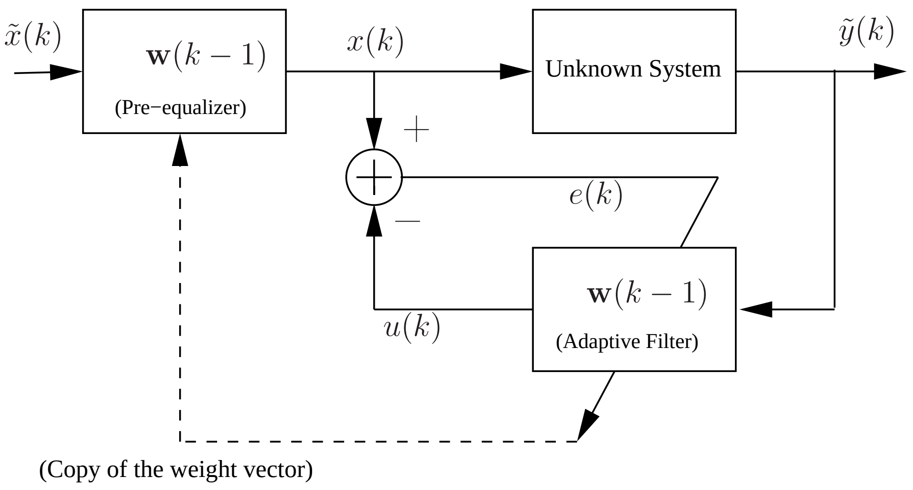


Figure 4.3: The pre-equalizer using indirect learning architecture

output of the copied weight vector $\tilde{y}(k)$ is given as

$$\tilde{y}(k) = \mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{x}}(k) \tag{4.15}$$

Let $\tilde{x}(k)$ be an input sequence independent of $x(k)$. Then Eq. (4.15) can be modified as

$$\tilde{y}(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\tilde{\mathbf{x}}(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\bar{\mathbf{r}}(k) \tag{4.16}$$

where $\bar{\mathbf{r}}(k) = \mathbf{U}^{-\mathrm{T}}(k-1)\tilde{\mathbf{x}}(k)$. As can be seen from Eq. (4.16), the variables $\mathbf{d}_{q2}(k)$ and $\mathbf{U}^{-\mathrm{T}}(k-1)$ are changing throughout the adaptation of $\mathbf{w}(k)$, contrary to output filtering mentioned in Section 4.2.2 where the weight vector is not changing. Lemma 4 can be used to update $\bar{\mathbf{r}}(k)$ as follows

$$\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \bar{\mathbf{r}}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \bar{\mathbf{r}}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{4.17}$$

For output filtering we also require to update the rotation matrix $\tilde{\mathbf{Q}}_{\theta f}(k)$ along with $\mathbf{d}_{q2}(k)$. This is only possible if a FQRD-RLS algorithm is running in parallel as in case of a pre-equalizer. In summary, the rotation matrix $\tilde{\mathbf{Q}}_{\theta f}(k)$ and the vector $\mathbf{d}_{q2}(k)$ are provided by the FQRD-RLS algorithm, and the update for $\bar{\mathbf{r}}(k)$ can be computed from Lemma 4 as described above. The algorithm for output filtering for the pre-equalizer is given in Table 4.5.

The computational complexity of the algorithm is given in Table 4.6 along with FQRD-RLS and IQRD-RLS algorithm for comparison. Note that for pre-equalizer the output filtering algorithm also includes the FQRD-RLS algorithm.

4.3 Experimental Results

The weight extraction and output filtering techniques for the FQRD-RLS algorithms allow it to be used in applications such as system identification, pre-equalization and post-equalization. In the following subsections the experimental setups for system identification, post-equalization, and pre-equalization are given along with the experimental results. For comparison purpose, the inverse QRD-RLS algorithm is selected. It is shown that the difference in results from the FQRD-RLS algorithm to the IQRD-RLS algorithms are within machine precision.

4.3.1 System Identification

The proposed WE algorithm is applied to the FQRD-RLS algorithm in a system identification setup. The plant has $N = 11$ coefficients and a colored noise input signal was used with SNR set to 30 dB. The condition number of the input-signal autocorrelation matrix is 821. The extracted weights of the FQRD-RLS algorithm are compared to those of the IQRD-RLS algorithm [21] which, with proper initialization, provides an identical solution in an infinite precision environment. As a measure of accuracy, the squared weight-difference from both algorithms was calculated and averaged over $K = 100$ ensemble using

$$\Delta \bar{w}_i = \frac{1}{K} \sum_{j=0}^{K-1} [w^j_{IQRD,i} - w^j_{FQRD,i}]^2 \tag{4.18}$$

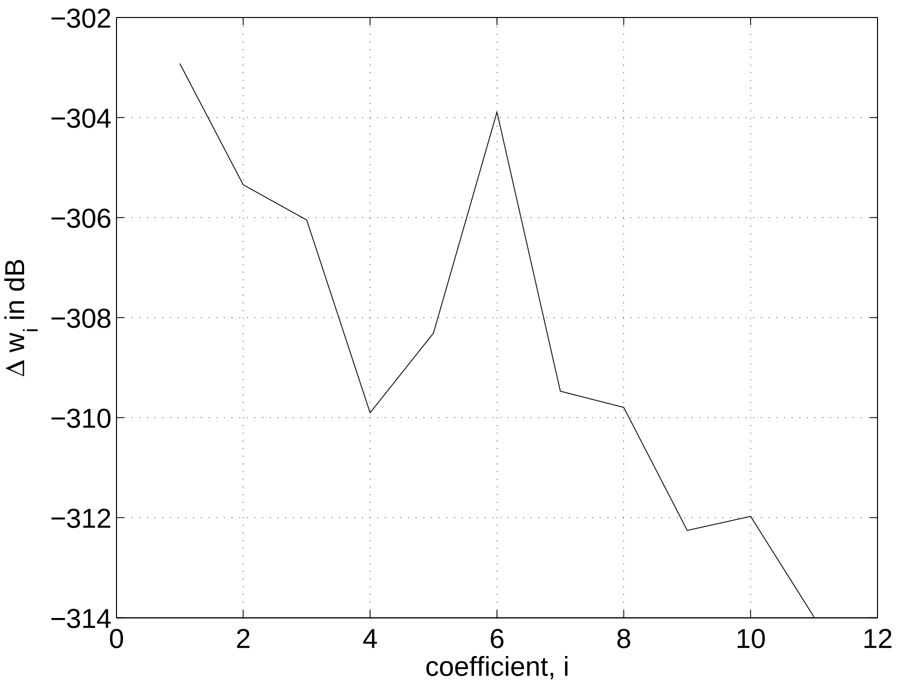


Figure 4.4: The difference between the weight vector from QRD-RLS and "Weight Extraction method". The variance of the measurement noise is 1×10^{-5}.

where for the j^{th} simulation run, $w^j_{IQRD,i}$ and $w^j_{FQRD,i}$ are the i^{th} coefficients of the IQRD-RLS and the FQRD-RLS algorithms, respectively. Figure 4.4 shows that the difference between the extracted weights of the FQRD-RLS and those of the IQRD-RLS are within machine precision. The learning-curves for both algorithms are plotted in Figure 4.5. As can be seen from the figure, they are identical up to numerical accuracy. Also the difference of the MSE curves is given in Figure 4.6.

4.3.2 Post-Equalization

The channel equalization example is taken from [28], where the channel taps are given as $\begin{bmatrix} 0.5 & 1.2 & 1.5 & -1 \end{bmatrix}^{\mathrm{T}}$. 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 training mode 150 symbols from a 4-QAM are used whereas in data mode 750 symbols from a 16-QAM constellation are processed. For comparison,

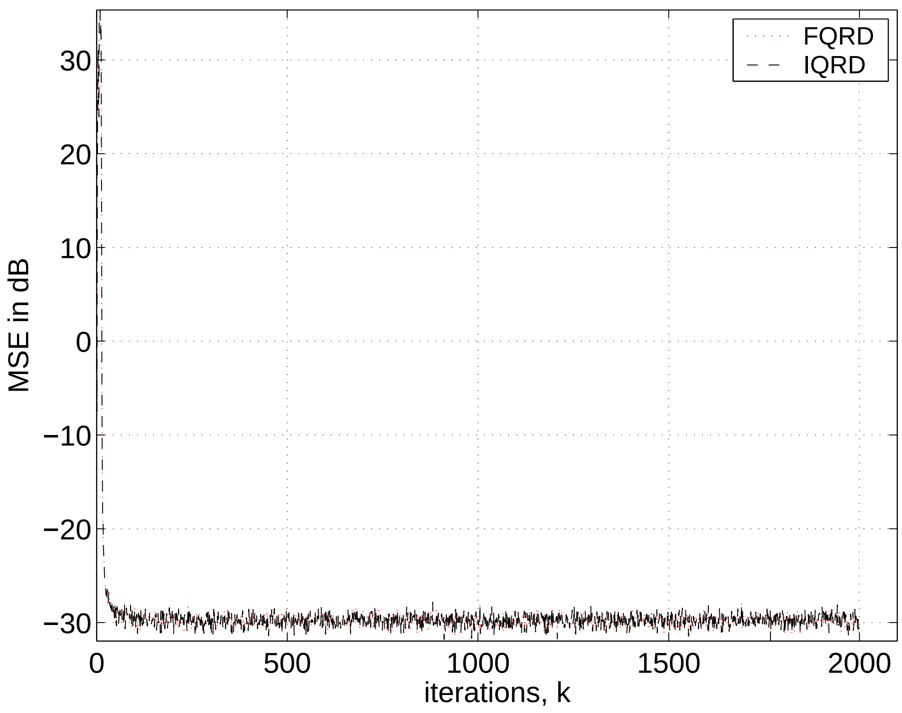


Figure 4.5: MSE curve for FQRD-RLS and IQRD-RLS algorithm.

the inverse QRD-RLS algorithm [29] is implemented along with the output filtering based FQRD-RLS algorithm. The MSE of the algorithms are shown in Figure 4.8, 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 4.7 for both algorithms. The constellation points achieved with inverse QRD-RLS and the proposed method cannot be distinguished in the figure (difference lower than -250 dB). Hence, the proposed technique works as desired. The difference of the learning curves is also plotted in Figure 4.9 to show that it goes down to numerical accuracy.

4.3.3 Pre-Equalization

The pre-equalizer is discussed in Section 2.2.3. The pre-equalizer experiment is done to verify that the FQRD-RLS algorithm can be used for pre-equalizer setup with the

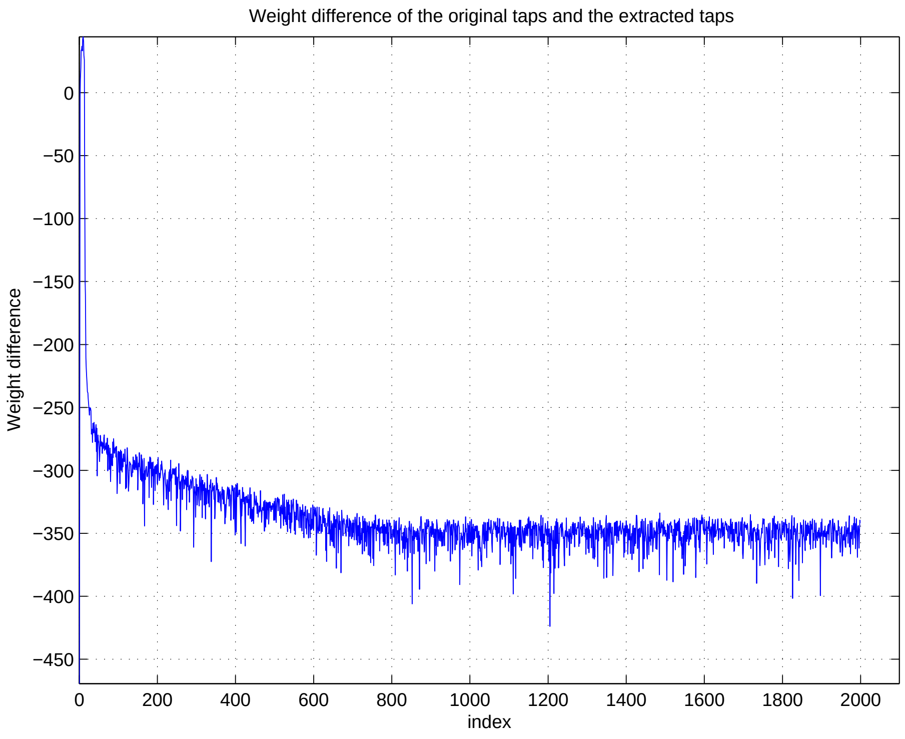


Figure 4.6: The difference between the weight vector from QRD-RLS and "Weight Extraction method". The variance of the measurement noise is 1×10^{-5}.

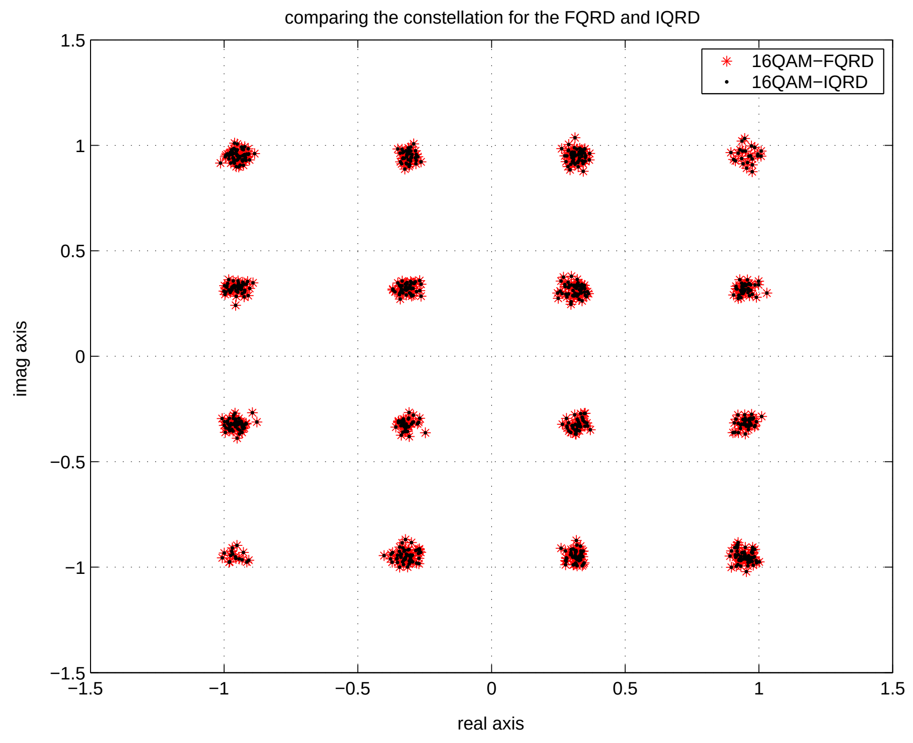


Figure 4.7: The constellation for the IQRD-RLS and the fast QRD-RLS algorithm after applying the output filtering technique

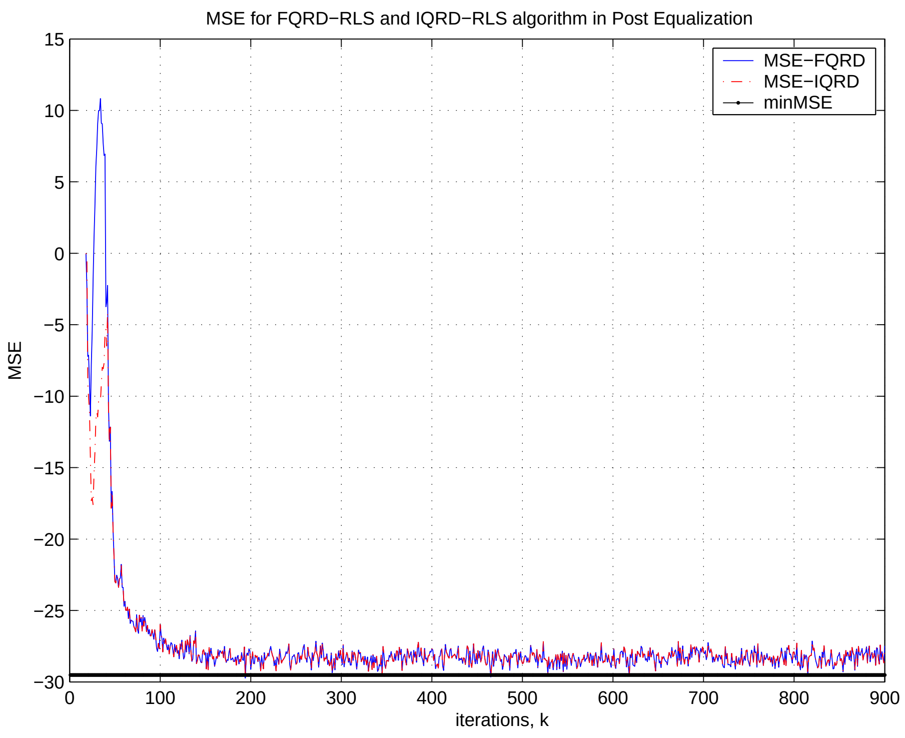


Figure 4.8: MSE curve for FQRD-RLS and IQRD-RLS algorithm in post-equalizer setup

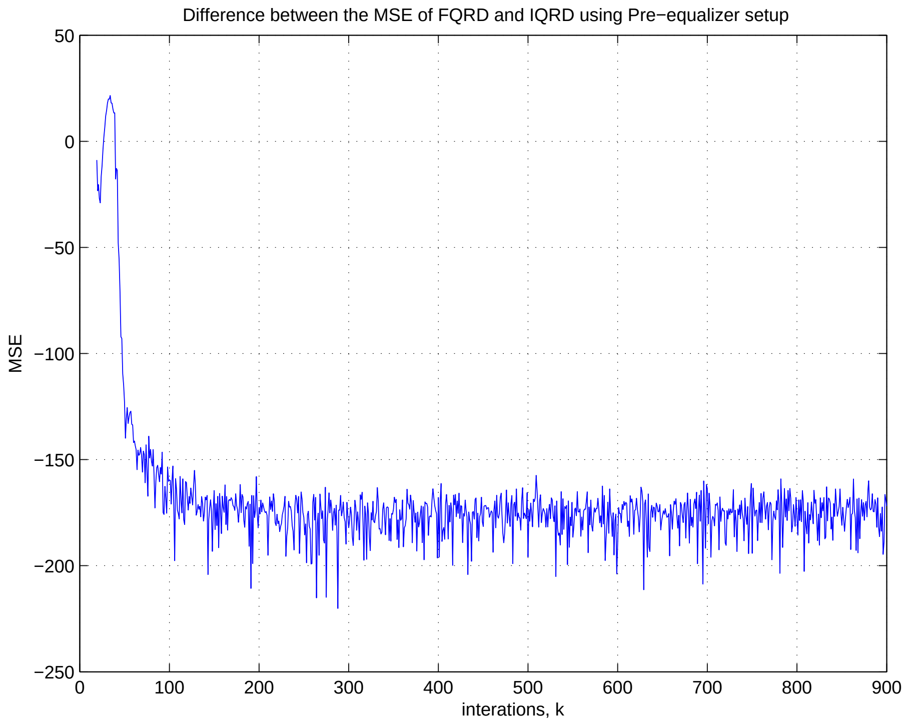


Figure 4.9: The difference between the MSE curves for the FQRD-RLS and the IQRD-RLS algorithms in post-equalizer setup

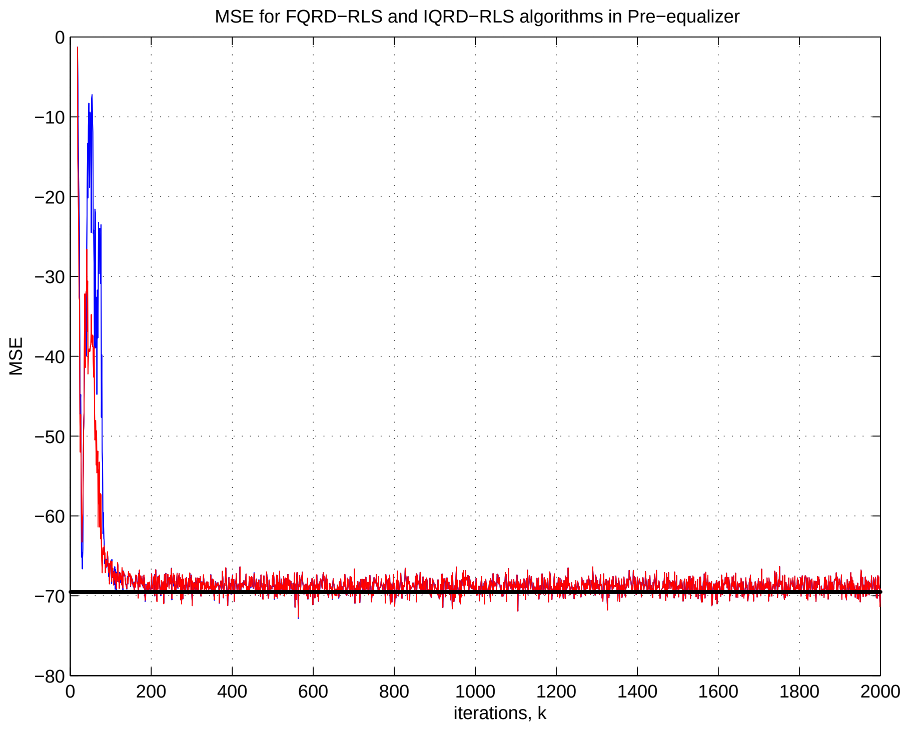


Figure 4.10: The MSE curves for IQRD-RLS and FQRD-RLS algorithm in pre-equalizer setup, the average taken over 50 *runs*

help of the proposed output filtering algorithm. The inverse QRD-RLS algorithm is also used in the experiment as the standard algorithm. Due to different initial values, the algorithm will be given different results in the transient. Therefore, the experiment is considered successful if both the algorithms converge to the same solution. The input signal to the pre-equalizer setup is the same as defined in the previous section, with 2000 samples. Zero-mean and 1×10^{-7} variance noise is added after the channel. The channel taps are same as in the previous section. For this experiment a 35 taps adaptive filter is used. The pre-equalizer results are obtained after 50 runs. To provide the evidence for the fact that both the algorithms converge to the same solution, the MSE curves, the difference of MSE curves and the difference of weight vectors after 2000 samples are given in Figure 4.10, 4.11 and 4.12 respectively.

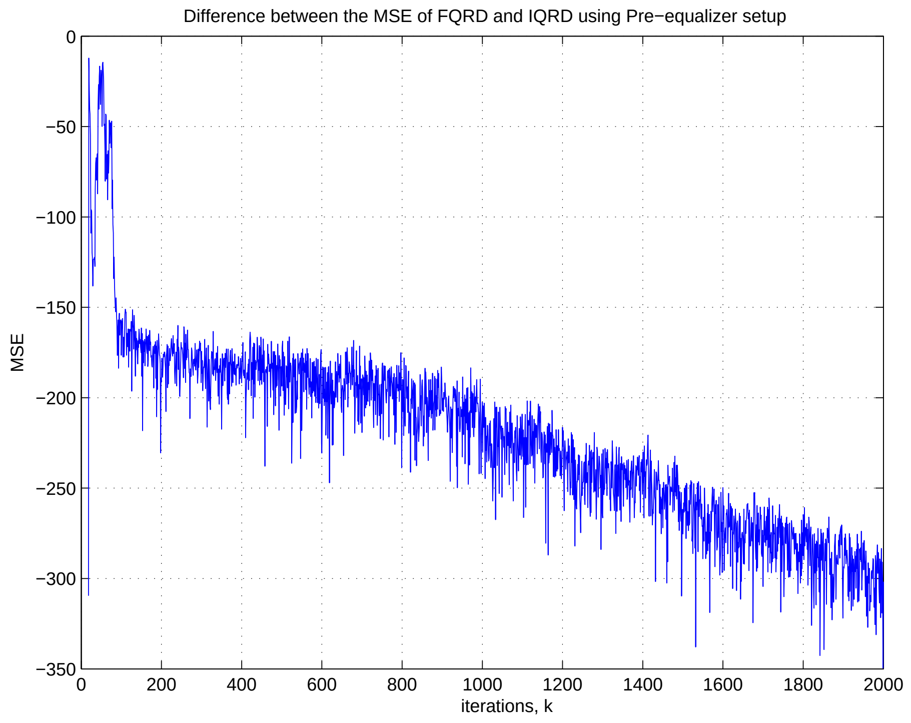


Figure 4.11: The difference between the MSE curves of the IQRD-RLS and the FQRD-RLS algorithms in pre-equalizer setup, the average taken over 50 *runs*

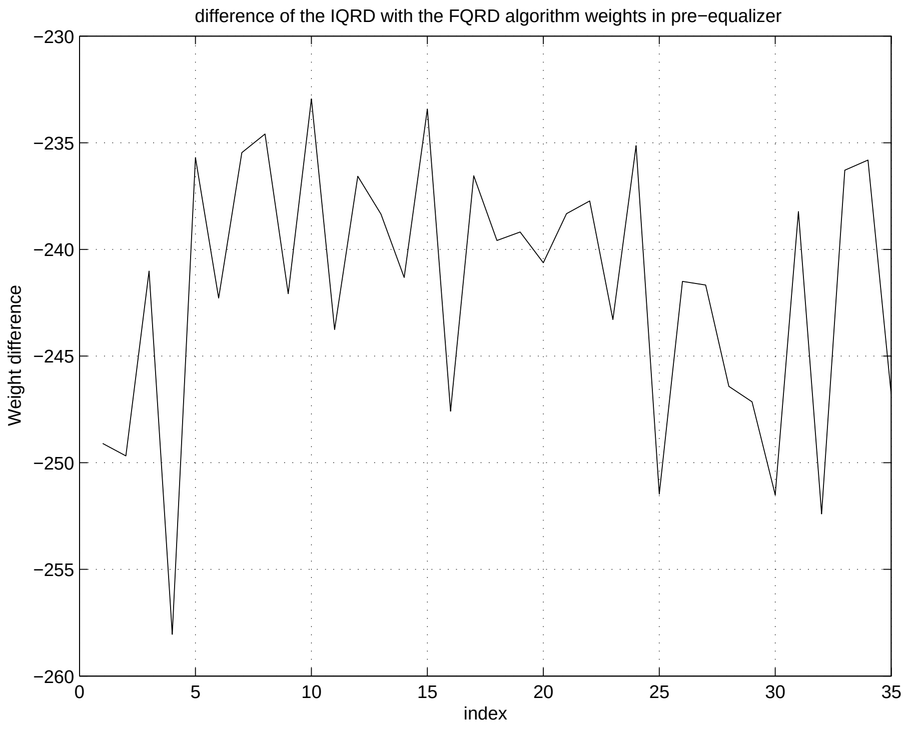


Figure 4.12: The difference between the weight vector from the QRD-RLS and the equivalent output filtering algorithm in pre-equalizer setup, the average was taken over 50 runs

4.4 Conclusion

In this chapter three novel algorithms were presented which extend the applications of the FQRD-RLS algorithms. In the weight extraction algorithm the weight coefficients for the FQRD-RLS algorithm are obtained, which means that the FQRD-RLS algorithms can be used for applications such as system identification and burst-type training of the post-equalizer. However, for the burst type training a more efficient way in terms of peak complexity has been proposed by introducing the output filtering. In this algorithm the output of the coefficient vector is obtained directly without extracting the weights and it has computational advantages over the weight extraction method when the number of training symbols are close to the number of coefficients. The third algorithm is the output filtering for the pre-equalizer setup. The FQRD-RLS algorithm can be used for pre-equalizers using this algorithm. The validity of the algorithms is first established by mathematical proofs and their proper functionality in the practice was verified by experimental results.

Table 4.1: Weight extraction algorithm.

```
Conventional FQR_PRI_B algorithm
--------------------------------
for each $k$
{ Obtain $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix} e_{fq1}(k) \\ \mathbf{d}_{fq2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} x(k) \\ \lambda^{1/2}\mathbf{d}_{fq2}(k-1) \end{bmatrix}$
  Obtain $\mathbf{a}(k)$
    $\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
  Obtain $\|\mathbf{e}_f(k)\|$:
    $\|\mathbf{e}_f(k)\| = \sqrt{e_{fq1}^2(k) + \lambda\|\mathbf{e}_f(k-1)\|^2}$
  Obtaining $\mathbf{Q}_{\theta f}(k)$:
    $\begin{bmatrix} \mathbf{0} \\ \|\mathbf{e}_f^{(0)}(k)\| \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix}$
  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:
    $\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
  $e(k) = e_{q1}(k)/\gamma(k)$ }
--------------------------------
Weight extraction at any chosen time instant $k$
--------------------------------
initializing $w_{f,-1}(k-1)$ and obtaining $\mathbf{f}(k-1)$
$w_{f,-1}(k-1) = -1$
$\begin{bmatrix} \gamma(k-1) \\ \mathbf{f}(k-1) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} 1 \\ \mathbf{0}_{N\times 1} \end{bmatrix}$
for each $i = 0 : N-1$
{   Obtaining $\mathbf{u}_i(k-1)$
    $\begin{bmatrix} \frac{-w_{b,i}(k-1)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{u}_i(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i-1}(k-2) \\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
  Obtaining $z_i(k-1)$ and $\mathbf{u}_i(k-2)$
    $z_i(k-1) = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1)/\gamma(k-1)$
    $\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-2) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k-1) \begin{bmatrix} z_i(k-1) \\ \mathbf{u}_i(k-1) \end{bmatrix}$
  Obtaining the coefficients $w_{f,i-1}(k-1)$
    $w_{f,i}(k-1) = \mathbf{u}_i^{\mathrm{T}}(k-2)\mathbf{d}_{fq2}(k-1)$
  Obtaining the coefficients
    $w_i(k-1) = \mathbf{u}_i^{\mathrm{T}}(k-1)\mathbf{d}_{q2}(k-1)$ }
```

Table 4.2: Computational complexity of weight extraction (WE).

$ALG. \times OPER.$	MULT	DIV	SQRT
FQR_PRI_B	$19N + 4$	$4N + 1$	$2N + 1$
WE (per weight i)	$16N - 6 - 14i$	1	0
WE (total)	$7N^2 + N$	1	0
IQRD-RLS	$3N^2 + 4N + 1$	N	N

Table 4.3: Equivalent output-filtering method for burst type training

Conventional FQRD-RLS algorithm with input signal $x(k)$

```
for each $0 \leq k < k_F$
{ Obtain $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix} e_{fq1}(k) \\ \mathbf{d}_{fq2}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k-1) \begin{bmatrix} x(k) \\ \lambda^{1/2}\mathbf{d}_{fq2}(k-1) \end{bmatrix}$
  Obtain $\mathbf{a}(k)$
    $\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
  Obtain $\|\mathbf{e}_f(k)\|$:
    $\|\mathbf{e}_f(k)\| = \sqrt{e_{fq1}^2(k) + \lambda\|\mathbf{e}_f(k-1)\|^2}$
  Obtaining $\mathbf{Q}_{\theta f}(k)$:
    $\begin{bmatrix} \mathbf{0} \\ \|\mathbf{e}_f^{(0)}(k)\| \end{bmatrix} = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix}$
  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:
    $\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
    $e(k) = e_{q1}(k)/\gamma(k)$
}
```

Equivalent output-filtering method with input signal $\tilde{x}(k)$

```
Initialization:
$\tilde{\mathbf{r}}(k_F - 1) = \mathbf{0}$
for each $k \geq k_F$
{
  Obtaining $\mathbf{r}(k)$ from $\tilde{\mathbf{r}}(k-1)$
    $\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{r}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k_F - 1) \begin{bmatrix} \tilde{\mathbf{r}}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
  Obtaining $z_i(k-1)$
    $z_i(k-1) = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{r}(k)/\gamma(k-1)$
  Updating $\tilde{\mathbf{r}}(k)$
    $\begin{bmatrix} 0 \\ \lambda^{-1/2}\tilde{\mathbf{r}}(k) \end{bmatrix} = \mathbf{Q}_{\theta}^{\mathrm{T}}(k_F - 1) \begin{bmatrix} z_i(k) \\ \mathbf{r}(k) \end{bmatrix}$
  Obtaining $e_f(k-1)$
    $e_f(k-1) = \tilde{\mathbf{r}}^{\mathrm{T}}(k)\mathbf{d}_{fq2}(k-1)$
  Obtaining the output
    $y(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k_F - 1)\mathbf{r}(k)$
}
```

Table 4.4: Computational complexity in terms of number of operations.

Algorithm	MULT	DIV	SQRT
FQR_PRI_B training mode	$20N + 5$	$3N + 1$	$2N + 1$
FQR_PRI_B data mode	$16N - 6$	1	0
Inverse QRD-RLS	$3N^2 + 4N + 1$	N	N

Table 4.5: Output Filtering for pre-equalizer type setup

```
for each $k$
{ Obtain $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix} e_{fq1}(k) \\ \mathbf{d}_{fq2}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k-1) \begin{bmatrix} x(k) \\ \lambda^{1/2}\mathbf{d}_{fq2}(k-1) \end{bmatrix}$
    Obtain $\mathbf{a}(k)$
    $\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
    Obtain $\|\mathbf{e}_f(k)\|$:
    $\|\mathbf{e}_f(k)\| = \sqrt{e_{fq1}^2(k) + \lambda\|\mathbf{e}_f(k-1)\|^2}$
   Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k)$:
    $\begin{bmatrix} \mathbf{0} \\ \mathbf{e}_f^{(0)}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq2}(k) \\ \|\mathbf{e}_f(k)\| \end{bmatrix}$
    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:
    $\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}$
    $\varepsilon(k) = e_{q1}(k)\gamma(k)$
}
```

The output filtering algorithm

```
Initialization:
$\bar{\mathbf{r}}(k-1) = \mathbf{0}$
for each $k$
{
 Obtaining $\bar{\mathbf{r}}(k)$:
 $\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \bar{\mathbf{r}}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \bar{\mathbf{r}}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix}$
  Obtaining $e_f(k-1)$
  $e_f(k-1) = \tilde{\mathbf{r}}^{\mathrm{T}}(k)\mathbf{d}_{fq2}(k-1)$
 Obtaining the output:
 $y(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\bar{\mathbf{r}}(k)$
}
```

Table 4.6: Computational complexity in terms of number of operations.

Algorithm	MULT	DIV	SQRT
FQR_PRI_B	$20N+5$	$3N+1$	$2N+1$
FQR_PRI_B Output filtering for Pre-equalizer	$4N-3$	1	0
Inverse QRD-RLS	$3N^2+4N+1$	N	N

Chapter 5

Multichannel Fast QRD-RLS Algorithm - Weight Extraction

In this chapter FQRD-RLS algorithms for multichannel systems are considered. As in the case of a single-channel systems, the multichannel FQRD-RLS algorithm does not provide direct access to the filter weights. For this purpose, we propose weight extraction for multichannel algorithms and equivalent-output filtering. The proposed algorithms can be seen as multichannel extensions of the single-channel algorithms presented in Chapter 4. They can be used for multichannel system identification and multichannel pre-equalization. The multichannel FQRD-RLS algorithms can be based on the updating of backward or forward prediction errors in the same way as the single-channel algorithms. As in the single-channel case, only the FQRD-RLS algorithm based on updating backward prediction errors is considered here because of its numerical robustness.

First, the multichannel system is introduced and the concept of multichannel adaptive filtering is presented in Section 5.1. The applications for the multichannel adaptive filtering are discussed in Section 5.2. The multichannel FQRD-RLS algorithm is derived in Section 5.3. The weight extraction and the equivalent-output filtering algorithms for the multichannel case are discussed in Section 5.4.1, 5.4.2, and 5.4.3, respectively. Section 5.5 shows experimental results for the weight extraction and the equivalent-output filtering algorithms in a multichannel scenario. Finally, Section 5.6 draws the conclusions.

5.1 The Multichannel Adaptive Filtering

A multichannel system has multiple input signals, each signal being fed to an independent set of weights. In this chapter, the number of coefficients in each channel is constrained to be the same. Let M be the number of channels and N be the number of coefficients in each channel. The input signal for the i^{th} channel is denoted by $x_i(k)$. The M-channel input signal vector at an instant k is defined as

$$\mathbf{x}_k^{\mathrm{T}} = \begin{bmatrix} x_1(k) & x_2(k) & \dots & x_M(k) \end{bmatrix}. \tag{5.1}$$

The input signal vector for M filters with N coefficients is, therefore, defined as

$$\mathbf{x}_N^{\mathrm{T}}(k) = \begin{bmatrix} \mathbf{x}_k^{\mathrm{T}} & \mathbf{x}_{k-1}^{\mathrm{T}} & \dots & \mathbf{x}_{k-N+1}^{\mathrm{T}} \end{bmatrix} \tag{5.2}$$

We denote the $NM \times 1$ coefficient vector as $\mathbf{w}_N(k)$. The output of the multichannel system $y_N(k)$ at instant k is obtained as

$$y_N(k) = \mathbf{x}_N^{\mathrm{T}}(k)\mathbf{w}_N(k) \tag{5.3}$$

A multichannel system is depicted in Figure 5.1. The adaptive filter equations of multichannel system are similar in form to the adaptive filter based on a single-channel system. The most notable difference is that vector updates are changed to matrix updates, whose dimensions relates to the number of channels M. In this thesis we focus on the least-squares based adaptive filtering algorithms for the multichannel systems. The error-signal vector for the weighted least-squares solution is written as

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}_N(k)\mathbf{w}_N(k) \tag{5.4}$$

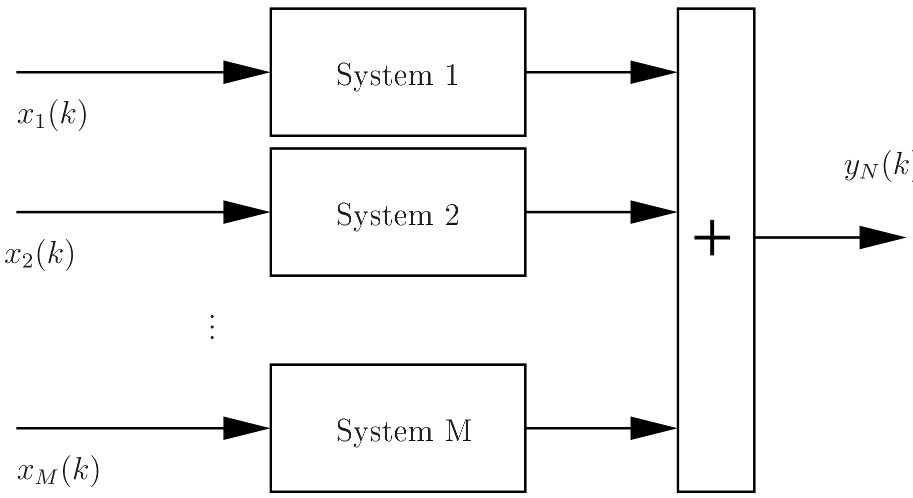


Figure 5.1: A multichannel system

where the multichannel input data matrix is given by

$$\mathbf{X}_N(k) = \begin{bmatrix}\mathbf{x}_N(k) & \lambda^{1/2}\mathbf{x}_N(k-1) & \dots & \lambda^{k/2}\mathbf{x}_N(0)\end{bmatrix}^{\mathrm{T}} \tag{5.5}$$

and the desired signal vector is expressed as

$$\mathbf{d}(k) = \begin{bmatrix}d(k) & \lambda^{1/2}d(k-1) & \dots & \lambda^{k/2}d(0)\end{bmatrix}^{\mathrm{T}} \tag{5.6}$$

The minimization of the weighted least-squares error results in the solution for the optimum weights given as

$$\mathbf{w}_N(k) = [\mathbf{X}_N^{\mathrm{T}}(k)\mathbf{X}_N(k)]^{-1}[\mathbf{X}_N^{\mathrm{T}}(k)\mathbf{d}(k)] \tag{5.7}$$

It is important to note that the solution is of the same form as Eq. (2.15) in the single-channel case. They are identical if $M = 1$.

For multi-channel adaptive filters, the same RLS and QRD-RLS algorithms can be used as in the single channel case; only the dimensions of the vectors and matrices are increased and the input signal is multi-channel. However, for the FQRD-RLS algorithms some modifications are required.

5.2 Adaptive Multichannel Filtering Applications

As noted before, there are several applications of multichannel adaptive filtering. In this section system identification and broadband adaptive beamforming applications are discussed in detail.

5.2.1 Multichannel System Identification

Single-channel system identification was explained in Section 2.2.1. The system identification setup for multichannel system is shown in Figure 5.2, where $\mathbf{x}_k$ is the multidimensional input signal, $d(k)$ is the desired signal, $y_N(k)$ is the output of the adaptive filter, $\mathbf{h}_N$ is the unknown system, $\mathbf{w}_N(k)$ is coefficient vector of the multichannel adaptive filter, $e(k)$ is the error signal, and $n(k)$ is the measurement noise.

After the convergence, the multichannel adaptive filter weight vector gives the estimate of the coefficients of the unknown multichannel system. The multichannel FQRD-RLS algorithm does not provide the weight vector at each iteration. Therefore, system identification is not possible with this algorithm. The weight extraction algorithm for the FQRD-RLS algorithm will be presented later in this chapter.

5.2.2 Broadband beamformer

A beamformer is inherently a multichannel system. It usually comprises a uniformly spaced linear array of M sensors. The sensors receive K signals from different directions resulting in a multichannel signal at the input of the beamformer. The received signal $\mathbf{x}(k)$ consists of unwanted interference signal and a desired signal given as

$$\mathbf{x}(k) = \mathbf{SAu}(k) + \mathbf{n}(k) \tag{5.8}$$

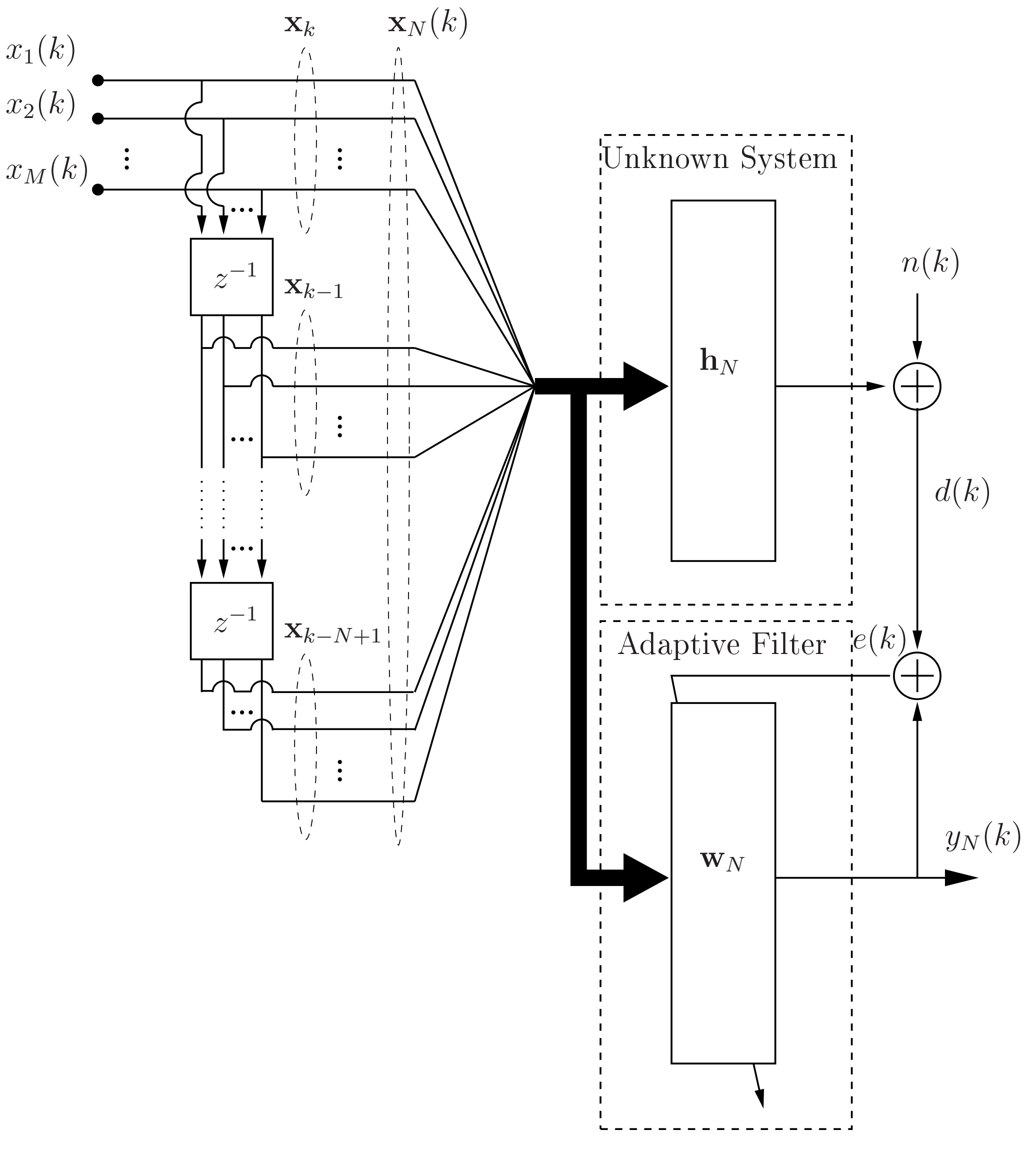


Figure 5.2: Multichannel system identification setup

where

$$\mathbf{S} = \begin{bmatrix}\mathbf{s}(\theta_1) & \mathbf{s}(\theta_2) & \dots & \mathbf{s}(\theta_K)\end{bmatrix}^{\mathrm{T}} \tag{5.9}$$

is the steering matrix, containing the steering vectors of the users of the form

$$\mathbf{s}(\theta_i) = \begin{bmatrix}1 & e^{j\pi \sin\theta_i} & \dots & e^{jM\pi \sin\theta_i}\end{bmatrix}^{\mathrm{T}}, \tag{5.10}$$

θ_i denotes the direction of arrival,

$$\mathbf{A} = \mathrm{diag}\begin{bmatrix}A_1 & A_2 & \dots & A_K\end{bmatrix} \tag{5.11}$$

contains the amplitudes of the signals,

$$\mathbf{u}(k) = \begin{bmatrix}u_1(k) & u_2(k) & \dots & u_K(k)\end{bmatrix} \tag{5.12}$$

is the vector of the transmitted signal and the interferers, and $\mathbf{n}(k)$ is the sampled noise sequence across the antenna array. The signal is assumed to be broadband so that N coefficients per sensor are considered, with the overall coefficient vector $\mathbf{w}_N$ for all sensors. The objective of the beamformer is to suppress the interferer signals by adjusting the coefficients so that nulls are placed in the directions of the interference signals.

The antenna beam pattern can be obtained from the coefficients after convergence, and the attenuation factor for the nulls placed for each interfering signal can be observed. The beamformer setup is illustrated in Fig 5.3.

5.3 The Multichannel FQRD-RLS Algorithms

In the derivation of the multichannel FQRD-RLS algorithms we have a multichannel input data matrix. This means that the equations become more complicated but are still very similar to those of the single-channel FQRD-RLS algorithm. Here we first introduce the forward and backward prediction equations and then we derive

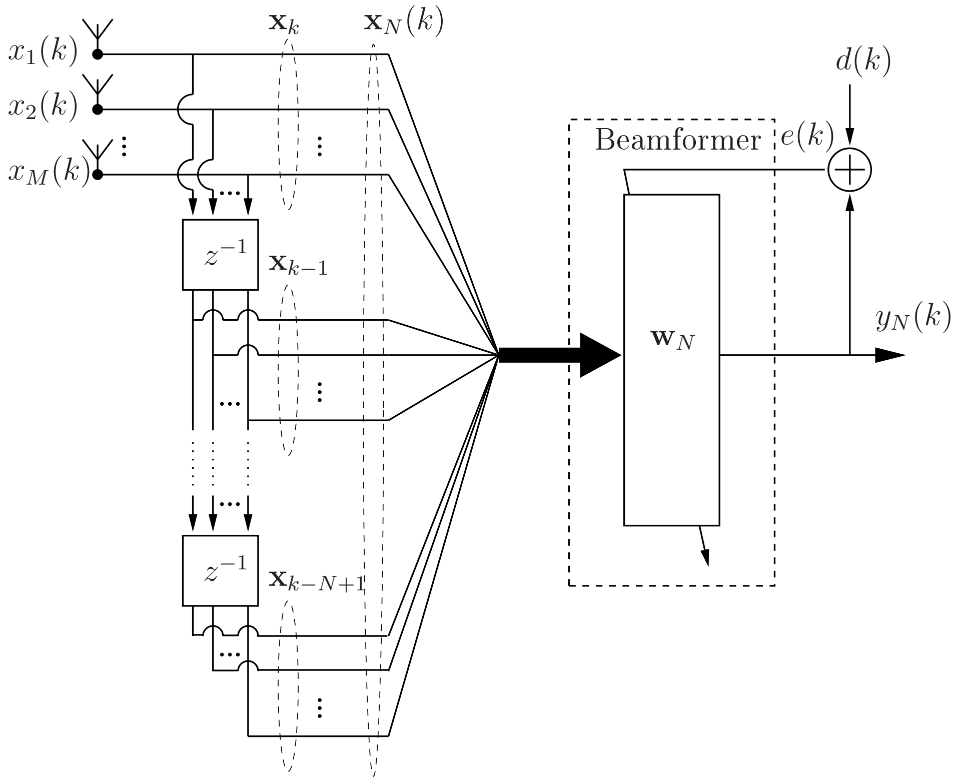


Figure 5.3: Broadband adaptive beamformer

the complete FQRD-RLS algorithm.

For the multichannel input data matrix $\mathbf{X}_N(k)$, there exists a unitary rotation matrix $\tilde{\mathbf{Q}}_{N\theta}(k)$ such that

$$\begin{bmatrix} \mathbf{0}_{(k-MN+1)\times(k-MN+1)} \\ \mathbf{U}_N(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{N\theta}(k)\mathbf{X}_N(k) \tag{5.13}$$

This is due to the fact that the matrix $\mathbf{X}_N(k)$ consists of $M \times N$ independent column vectors, the lower triangular matrix $\mathbf{U}_N(k) \in \mathbb{R}^{NM\times NM}$ is called the Cholesky factor of $\mathbf{X}_N^{\mathrm{T}}(k)\mathbf{X}_N(k)$. Consider the forward and backward prediction equation for the multichannel case

$$\mathbf{E}_f(k) = \mathbf{D}_f(k) - \begin{bmatrix} \mathbf{X}_N(k-1) \\ \mathbf{0}_{1\times NM} \end{bmatrix} \mathbf{W}_{Nf}(k) \tag{5.14}$$

$$\mathbf{E}_b(k) = \mathbf{D}_b(k) - \mathbf{X}_N(k)\mathbf{W}_{Nb}(k) \tag{5.15}$$

this can also be written as

$$\mathbf{E}_f(k) = \begin{bmatrix} \mathbf{D}_f(k) & \begin{pmatrix} \mathbf{X}_N(k-1) \\ \mathbf{0}_{1\times NM} \end{pmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ -\mathbf{W}_{Nf}(k) \end{bmatrix} \tag{5.16}$$

$$\mathbf{E}_b(k) = \begin{bmatrix} \mathbf{X}_N(k) & \mathbf{D}_b(k) \end{bmatrix} \begin{bmatrix} -\mathbf{W}_{Nb}(k) \\ \mathbf{I} \end{bmatrix} \tag{5.17}$$

where the matrix $\mathbf{E}_f(k)$, and $\mathbf{E}_b(k)$ are the forward and the backward prediction error matrices, respectively, and the forward and backward reference signal matrices are

$$\mathbf{D}_f(k) = \begin{bmatrix} \mathbf{x}_k & \lambda^{1/2}\mathbf{x}_{k-1} & \dots & \lambda^{k/2}\mathbf{x}_0 \end{bmatrix}^{\mathrm{T}} \tag{5.18}$$

$$\mathbf{D}_b(k) = \begin{bmatrix} \mathbf{x}_{k-N} & \lambda^{1/2}\mathbf{x}_{k-N-1} & \dots & \lambda^{(k-N)/2}\mathbf{x}_0 & \mathbf{0}_{M\times(N+1)} \end{bmatrix}^{\mathrm{T}} \tag{5.19}$$

Note that the forward and backward prediction equations are decoupled because a sample value for each channel is predicted separately. Applying the rotation matrix

$\tilde{\mathbf{Q}}_{N\theta}(k)$ to Eq. (5.16) and (5.17) results in

$$\begin{aligned}&\begin{bmatrix}\tilde{\mathbf{Q}}_{N\theta}(k-1) & \mathbf{0}_{k\times M}\\ \mathbf{0}_{M\times k} & \mathbf{I}_{M\times M}\end{bmatrix}\mathbf{E}_f(k)\\ &=\begin{bmatrix}\tilde{\mathbf{Q}}_{N\theta}(k-1) & \mathbf{0}_{k\times M}\\ \mathbf{0}_{M\times k} & \mathbf{I}_{M\times M}\end{bmatrix}\begin{bmatrix}\mathbf{D}_f(k) & \begin{pmatrix}\mathbf{X}_N(k-1)\\ \mathbf{0}_{1\times NM}\end{pmatrix}\end{bmatrix}\begin{bmatrix}1\\ -\mathbf{W}_{Nf}(k)\end{bmatrix}\end{aligned} \tag{5.20}$$

carrying out the multiplication gives

$$\begin{bmatrix}\mathbf{E}_{fq1}(k)\\ \mathbf{E}_{fq2}(k)\end{bmatrix}=\begin{bmatrix}\mathbf{D}_{fq1}(k) & \mathbf{0}_{(k-NM)\times(k-NM)}\\ \mathbf{D}_{fq2}(k) & \begin{pmatrix}\mathbf{U}_N(k-1)\\ \mathbf{0}_{1\times NM}\end{pmatrix}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ -\mathbf{W}_{Nf}(k)\end{bmatrix} \tag{5.21}$$

$$\begin{aligned}&\tilde{\mathbf{Q}}_{N\theta}(k)\mathbf{E}_b(k)=\tilde{\mathbf{Q}}_{N\theta}(k)\begin{bmatrix}\mathbf{X}_N(k) & \mathbf{D}_b(k)\end{bmatrix}\begin{bmatrix}-\mathbf{W}_{Nb}(k)\\ \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{E}_{bq1}(k)\\ \mathbf{E}_{bq2}(k)\end{bmatrix}\\ &=\begin{bmatrix}\mathbf{0}_{(k-NM)\times(k-NM)} & \mathbf{D}_{bq1}(k)\\ \mathbf{U}_N(k-1) & \mathbf{D}_{bq2}(k)\end{bmatrix}\begin{bmatrix}-\mathbf{W}_{Nb}(k)\\ \mathbf{I}\end{bmatrix}\end{aligned} \tag{5.22}$$

where it is obvious that $\mathbf{E}_{fq1}(k)=\mathbf{D}_{fq1}(k)$ and $\mathbf{E}_{bq1}(k)=\mathbf{D}_{bq1}(k)$. And the forward and backward prediction weight vectors are, therefore, given by

$$\mathbf{W}_{Nf}(k)=\mathbf{U}_N^{-\mathrm{T}}(k-1)\mathbf{D}_{fq2}(k) \tag{5.23}$$

$$\mathbf{W}_{Nb}(k)=\mathbf{U}_N^{-\mathrm{T}}(k)\mathbf{D}_{bq2}(k) \tag{5.24}$$

>From Eq. (5.16) and Eq. (5.17) we can write matrix $\mathbf{X}_{N+1}(k)$ as

$$\mathbf{X}_{N+1}(k)=\begin{bmatrix}\mathbf{D}_f(k)\begin{pmatrix}\mathbf{X}_N(k-1)\\ \mathbf{0}_{1\times NM}\end{pmatrix}\\ \mathbf{0}_{(M-1)\times(M)}\mathbf{0}_{(M-1)\times(MN)}\end{bmatrix} \tag{5.25}$$

and

$$\mathbf{X}_{N+1}(k)=\begin{bmatrix}\mathbf{X}_N(k) & \mathbf{D}_b(k)\\ \mathbf{0}_{(M-1)\times(MN)} & \mathbf{0}_{(M-1)\times(M)}\end{bmatrix} \tag{5.26}$$

Similar to Eq. (5.13) we can find a unitary rotation matrix $\tilde{\mathbf{Q}}_{(N+1)\theta}(k)$ to determine the Cholesky factor of the matrix $\mathbf{X}^{\mathrm{T}}_{N+1}(k)\mathbf{X}_{N+1}(k)$

$$\begin{aligned}\begin{bmatrix}\mathbf{0}_{(k+1-NM-M)\times(NM)} & \mathbf{0}_{(k+1-NM-M)\times(M)}\\ \mathbf{0}_{M\times NM} & \mathbf{E}_{bq1}(k)\\ \mathbf{U}_N(k) & \mathbf{D}_{bq2}(k)\end{bmatrix} &= \begin{bmatrix}\mathbf{0}_{(k+1-NM-M)\times(NM+M)}\\ \mathbf{U}_{N+1}(k)\end{bmatrix} \\ &= \tilde{\mathbf{Q}}_{(N+1)\theta}(k)\mathbf{X}_{N+1}(k)\end{aligned} \qquad (5.27)$$

where the lower triangular matrix $\mathbf{U}_{N+1}(k) \in \mathbb{R}^{NM\times NM}$ is the Cholesky factor of $\mathbf{X}^{\mathrm{T}}_{N+1}(k)\mathbf{X}_{N+1}(k)$. The unitary rotation matrix can be written as a series of Givens rotation matrices as follows

$$\begin{bmatrix}\mathbf{0}_{(k+1-NM-M)\times(NM+M)}\\ \mathbf{U}_{N+1}(k)\end{bmatrix} = \mathbf{Q}'_f(k)\mathbf{Q}_f(k)\begin{bmatrix}\mathbf{Q}(k-1) & \mathbf{0}_{k\times M}\\ \mathbf{0}_{M\times k} & \mathbf{I}_{M\times M}\end{bmatrix}\mathbf{X}_{N+1}(k) \qquad (5.28)$$

The triangularization of matrix $\mathbf{X}_{N+1}(k)$ is done in three steps. Firstly, we obtain the rightmost rotation matrix as

$$\begin{aligned}\begin{bmatrix}\mathbf{0}_{(k+1-NM-M)\times(NM+M)}\\ \mathbf{U}_{N+1}(k)\end{bmatrix} &= \mathbf{Q}'_f(k)\mathbf{Q}_f(k)\begin{bmatrix}\mathbf{Q}(k-1) & \mathbf{0}_{k\times M}\\ \mathbf{0}_{M\times k} & \mathbf{I}_{M\times M}\end{bmatrix}\begin{bmatrix}\mathbf{D}_f(k)\begin{pmatrix}\mathbf{X}_N(k)\\ \mathbf{0}_{1\times NM}\end{pmatrix}\\ \mathbf{0}_{(M-1)\times(MN+M)}\end{bmatrix}\\ &= \mathbf{Q}'_f(k)\mathbf{Q}_f(k)\begin{bmatrix}\mathbf{E}_{fq1} & \mathbf{0}_{(k-NM-1)\times(NM)}\\ \mathbf{D}_{fq2}(k) & \mathbf{U}_N(k-1)\\ \lambda^{k/2}\mathbf{x}_0^{\mathrm{T}} & \mathbf{0}_{1\times NM}\\ \mathbf{0}_{(M-1)\times(M)} & \mathbf{0}_{(M-1)\times(MN)}\end{bmatrix}\end{aligned} \qquad (5.29)$$

Next we apply rotation matrix $\mathbf{Q}_f(k)$. As a result, the values at the upper part of the vectors are rotated down to the lower part, resulting in

$$\begin{bmatrix}\mathbf{0}_{(k+1-NM-M)\times(NM+M)}\\ \mathbf{U}_{N+1}(k)\end{bmatrix} = \mathbf{Q}'_f(k)\begin{bmatrix}\mathbf{0}_{(k-NM-1)\times(M)} & \mathbf{0}_{(k-NM-1)\times(NM+M)}\\ \mathbf{D}_{fq2}(k) & \mathbf{U}_N(k-1)\\ \mathbf{E}'_f(k) & \mathbf{0}_{M\times NM}\end{bmatrix}. \qquad (5.30)$$

The ever increasing size of the matrices can be avoided by removing the redundant zeros from both sides of Eq. (5.30) and the corresponding rows and columns from the rotation matrix. Therefore we get

$$\begin{bmatrix} \mathbf{0}_{M\times NM} & \mathbf{E}_{bq1}(k) \\ \mathbf{U}_N(k) & \mathbf{D}_{bq2}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{D}_{fq2}(k) & \mathbf{U}_N(k-1) \\ \mathbf{E}'_f(k) & \mathbf{0}_{M\times NM} \end{bmatrix} \tag{5.31}$$

where the rotation matrix $\mathbf{Q}'_{\theta f}(k)$ is the matrix $\mathbf{Q}'_f(k)$ with the redundant rows and columns removed. Finally, the rotation matrix $\mathbf{Q}'_{\theta f}(k)$ results in complete triangularization, which means we can obtain a lower triangular matrix $\mathbf{E}_f^{(0)}(k)$ as follows

$$\begin{bmatrix} \mathbf{0}_{1\times M} \\ \mathbf{E}_f^{(0)}(k) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{D}_{fq2}(k) \\ \mathbf{E}'_f(k) \end{bmatrix} \tag{5.32}$$

where matrix $\mathbf{E}_f^{(0)}(k)$ is referred to as the forward error covariance matrix. In the following we derive the update equations for $\mathbf{E}'_f(k)$ and $\mathbf{D}_{fq2}(k)$. The rotation matrices $\mathbf{Q}_f(k)$ and $\mathbf{Q}(k-1)$ can also be written as

$$\begin{aligned} \mathbf{Q}_f(k) &= \underbrace{\mathbf{Q}_f(k) \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}_f^{\mathrm{T}}(k-1) \end{bmatrix}}_{\hat{\mathbf{Q}}_f(k)} \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}_f(k-1) \end{bmatrix} \\ &= \hat{\mathbf{Q}}_f(k) \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}_f(k-1) \end{bmatrix} \end{aligned} \tag{5.33}$$

and

$$\begin{aligned} \mathbf{Q}(k-1) &= \underbrace{\mathbf{Q}(k-1) \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}^{\mathrm{T}}(k-2) \end{bmatrix}}_{\hat{\mathbf{Q}}(k-1)} \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}(k-2) \end{bmatrix} \\ &= \hat{\mathbf{Q}}(k-1) \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}(k-2) \end{bmatrix} \end{aligned} \tag{5.34}$$

It is also important to mention that the two rotation matrices are commutative [30], i.e.,

$$\begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}_f(k-1) \end{bmatrix} \begin{bmatrix} \hat{\mathbf{Q}}(k-1) & \mathbf{0}_{M\times k} \\ \mathbf{0}_{k\times M} & \mathbf{I}_{M\times M} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{Q}}(k-1) & \mathbf{0}_{M\times k} \\ \mathbf{0}_{k\times M} & \mathbf{I}_{M\times M} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}_{1\times k+M-1} \\ \mathbf{0}_{k+M-1\times 1} & \mathbf{Q}_f(k-1) \end{bmatrix} \tag{5.35}$$

Using the commutative property Eq. (5.28) can be written as

$$\begin{bmatrix} \mathbf{0}_{k-N(1+M)\times 1} \\ \mathbf{D}_{fq2}(k) \\ \mathbf{E}'_f(k) \end{bmatrix} = \mathbf{Q}_f(k) \begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0}_{M\times k} \\ \mathbf{0}_{k\times M} & \mathbf{I}_{M\times M} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k^{\mathrm{T}} \\ \lambda^{1/2}\mathbf{x}_{k-1}^{\mathrm{T}} \\ \vdots \\ \lambda^{k/2}\mathbf{x}_0^{\mathrm{T}} \\ \mathbf{0}_{M-1\times M} \end{bmatrix} \tag{5.36}$$

Combining Eq. (5.36) with Eqs. (5.33)-(5.34) we get

$$\begin{aligned} \begin{bmatrix} \mathbf{0}_{k-N(1+M)\times 1} \\ \mathbf{D}_{fq2}(k) \\ \mathbf{E}'_f(k) \end{bmatrix} &= \hat{\mathbf{Q}}_f(k) \begin{bmatrix} \hat{\mathbf{Q}}(k-1) \begin{pmatrix} \mathbf{x}_k^{\mathrm{T}} \\ \mathbf{0}_{k-N(M+1)+1\times 1} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k-1) \end{pmatrix} \\ \lambda^{1/2}\mathbf{E}_f(k-1) \end{bmatrix} \\ &= \hat{\mathbf{Q}}_f(k) \begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k) \\ \mathbf{0} \\ \mathbf{D}_{fq2}(k) \\ \lambda^{1/2}\mathbf{E}_f(k-1) \end{bmatrix} \end{aligned} \tag{5.37}$$

Eq. (5.37) results in two update equations, one for the vector $\mathbf{D}_{fq2}(k)$ and one for matrix $\mathbf{E}'_f(k)$. These equations are given by

$$\begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}_{k+1}^{\mathrm{T}} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix} \tag{5.38}$$

and

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{E}_f(k+1) \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{E}_f(k) \end{bmatrix} \tag{5.39}$$

where $\bar{\mathbf{Q}}_f(k+1)$ is obtained from $\hat{\mathbf{Q}}(k+1)$ by removing the redundant rows and columns contributing to the ever increasing size of it. Finally, the update equation for vector $\mathbf{d}_{q2}(k)$ is given by

$$\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k+1) \end{bmatrix} \tag{5.40}$$

where the *a priori* error is given as

$$e(k) = e_{q1}(k)/\gamma(k) \tag{5.41}$$

Next the *a priori* and the *a posteriori* algorithms are derived based on the update equation obtained in this section.

5.3.1 The Multichannel FQR_PRI_B Algorithm

The multichannel *a priori* FQRD-RLS algorithm updates vector $\mathbf{a}_N(k)$ defined as

$$\mathbf{a}_N(k) = \lambda^{-1/2}\mathbf{U}_N^{-T}(k-1)\mathbf{x}_N(k) \tag{5.42}$$

As discussed before, the basic idea in the FQRD based algorithms is to eliminate the matrix updates for the matrix $\mathbf{U}_N^{-\mathrm{T}}(k)$. Therefore, the update equation for vector $\mathbf{a}_N(k)$ is considered. The extended Cholesky matrix $\mathbf{U}_{N+1}(k)$ is defined in Eq. (5.31). Taking the inverse and the transpose of both sides we get

$$\mathbf{U}_{N+1}^{-\mathrm{T}}(k) = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{0}_{NM\times M} & \mathbf{U}_N^{-\mathrm{T}}(k-1) \\ [\mathbf{E}'_f(k)]^{-\mathrm{T}} & -[\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}_{fq2}^{\mathrm{T}}(k)\mathbf{U}_N^{-\mathrm{T}}(k-1) \end{bmatrix} \tag{5.43}$$

By post multiplying both sides of Eq. (5.43) with the extended multichannel input data vector $\mathbf{x}_{N+1}(k)$ and $\lambda^{-1/2}$ we obtain the update equation for vector $\mathbf{a}_N(k)$

$$\mathbf{a}_{N+1}(k+1) = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}_N(k) \\ \mathbf{r}(k+1) \end{bmatrix} \tag{5.44}$$

where $\mathbf{r}(k+1) = \lambda^{1/2}[\mathbf{E}'_f(k)]^{-\mathrm{T}}\tilde{\mathbf{e}}_f(k+1)$ and $\tilde{\mathbf{e}}_f(k+1) = \gamma(k)\tilde{\mathbf{e}}_{fq1}(k+1)$. >From the updated vector $\mathbf{a}_N(k)$, the update equation for the rotation matrix $\mathbf{Q}_\theta(k+1)$ is obtained

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0}_{NM\times 1} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_N(k+1) \end{bmatrix} \tag{5.45}$$

In order to avoid the computation of matrix inversion in $\mathbf{r}(k+1)$ we can use [30]

$$\begin{bmatrix} * \\ \mathbf{0}_{M\times 1} \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} 1/\gamma(k+1) \\ -\mathbf{r}(k+1) \end{bmatrix} \tag{5.46}$$

The algorithm is summarized in Table 5.1.

5.3.2 The Multichannel FQR_POS_B Algorithm

The derivation for the *a posteriori* algorithm follows the same lines as those for the *a priori* case. The vector $\mathbf{f}_N(k+1)$ is defined as

$$\mathbf{f}_N(k+1) = \mathbf{U}_N^{-\mathrm{T}}(k+1)\mathbf{x}_N(k+1) \tag{5.47}$$

By post multiplying Eq. (5.31) with the multichannel input data vector $\mathbf{x}_{N+1}(k)$ we get the update equation for the vector $\mathbf{f}_N(k)$ as

$$\mathbf{f}_{N+1}(k+1) = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{f}_N(k) \\ \mathbf{p}(k+1) \end{bmatrix} \tag{5.48}$$

Table 5.1: The FQR_PRI_B Algorithm based on backward prediction errors.

```
for each $k$
{ Obtaining $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix} \tilde{\mathbf{e}}^{\mathrm{T}}_{fq1}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}_{k+1} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}$
    Obtaining $\|\mathbf{E}_f(k+1)\|$:
    $\begin{bmatrix} \mathbf{0}_{1\times M} \\ \mathbf{E}'_f(k+1) \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} \tilde{\mathbf{e}}^{\mathrm{T}}_{fq1}(k+1) \\ \lambda^{1/2}\mathbf{E}'_f(k) \end{bmatrix}$
    Obtaining $\mathbf{r}(k+1)$
    $\begin{bmatrix} * \\ \mathbf{0}_{M\times 1} \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} 1/\gamma(k+1) \\ -\mathbf{r}(k+1) \end{bmatrix}$
    Obtaining $\mathbf{a}_N(k+1)$
    $\mathbf{a}_{N+1}(k+1) = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{a}_N(k) \\ \mathbf{r}(k+1) \end{bmatrix}$
    Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k+1)$:
    $\begin{bmatrix} \mathbf{0}_{NM\times M} \\ \mathbf{E}^{(0)}_f(k+1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \|\mathbf{E}'_f(k+1)\| \end{bmatrix}$
    Obtaining $\mathbf{Q}_{\theta}(k+1)$:
    $\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_{\theta}(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_N(k+1) \end{bmatrix}$
    Joint Process Estimation:
    $\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}$
    $e(k) = e_{q1}(k)/\gamma(k)$
}
```

where vector $\mathbf{p}(k+1) = [\mathbf{E}'_f(k+1)]^{-\mathrm{T}}\tilde{\mathbf{e}}'_f(k+1)$. We can avoid the backward substitution using the following relation [30]

$$\begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} \gamma(k) \\ \mathbf{0}_{M\times 1} \end{bmatrix} \tag{5.49}$$

Table 5.2: The FQR_POS_B Algorithm based on backward prediction errors.

```
for each $k$
{ Obtain $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix} \tilde{\mathbf{e}}^{\mathrm{T}}_{fq1}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}_{k+1} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}$
    Obtain $\|\mathbf{E}_f(k+1)\|$:
    $\begin{bmatrix} \mathbf{0}_{1\times M} \\ \mathbf{E}'_f(k+1) \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} \tilde{\mathbf{e}}^{\mathrm{T}}_{fq1}(k+1) \\ \lambda^{1/2}\mathbf{E}'_f(k) \end{bmatrix}$
    Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k+1)$:
    $\begin{bmatrix} \mathbf{0}_{NM\times M} \\ \mathbf{E}^{(0)}_f(k+1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \|\mathbf{E}'_f(k+1)\| \end{bmatrix}$
    Obtaining $\mathbf{p}_N(k+1)$
    $\begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} \gamma(k) \\ \mathbf{0}_{M\times 1} \end{bmatrix}$
    Obtain $\mathbf{f}_N(k+1)$
    $\mathbf{f}_{N+1}(k+1) = \tilde{\mathbf{Q}}_{\theta f}(k+1) \begin{bmatrix} \mathbf{f}_N(k) \\ \mathbf{p}(k+1) \end{bmatrix}$
    Obtaining $\mathbf{Q}_\theta(k+1)$:
    $\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}^{\mathrm{T}}_\theta(k+1) \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}_N(k+1) \end{bmatrix}$
    Joint Process Estimation:
    $\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}$
    $e(k) = e_{q1}(k)/\gamma(k)$
}
```

The update for the rotation matrix can be obtained from the vector $\mathbf{f}_N(k)$ with the help of the following equation

$$\begin{bmatrix} \gamma(k+1) \\ \mathbf{f}_N(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \mathbf{0}_{NM\times 1} \end{bmatrix} \tag{5.50}$$

The complete algorithm is summarized in Table 5.2.

5.4 Weight-Extraction and Output Filtering Approach for MC-FQRD-RLS algorithms

In Chapter 4, new applications for the single channel FQRD-RLS algorithm were presented that required the knowledge of the coefficients. The objective of this section is to show that, with slight modifications, the lemmas of Chapter 4 can be extended for the MCFQRD-RLS algorithm cases.

In Subsection 5.4.1, the idea of weight extraction for the multichannel case is presented. Next the output filtering for burst-type errors and the equivalent output filtering for pre-equalizer setup is elaborated with the help of the modified lemmas. The proofs for the lemmas are presented in Appendix A.

5.4.1 Weight Extraction for the *a priori* Multichannel FQRD-RLS Algorithm

Consider the output of the multichannel adaptive filter $y_N(k)$ given by

$$\begin{aligned} y_N(k) &= \mathbf{w}_N^{\mathrm{T}}(k-1)\mathbf{x}_N(k) \\ &= \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}_N^{\mathrm{T}}(k-1)\mathbf{x}_N(k) \end{aligned} \tag{5.51}$$

Let us define an impulse vector $\delta_i = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{NM\times 1}$ to be a vector with a "1" at the i^{th} position ($1 \le i \le NM$). Note that vector $\mathbf{x}_N(k)$ comprises of input vectors from M channels. From the definition of vector $\mathbf{x}_N(k)$ given in Section 5.1 it can be seen that the elements corresponding to one channel are placed at every M^{th} instant. The j^{th} element of the weight vector for the i^{th}

channel is given as

$$\begin{aligned} w_{N,i+jM}(k) &= \mathbf{w}_N^{\mathrm{T}}(k-1)\delta_{i+jM} \\ &= \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}_N^{-\mathrm{T}}(k-1)\delta_{i+jM} \\ &= \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{u}_{i+jM}(k-1) \end{aligned} \tag{5.52}$$

Similar to the single channel case in Chapter 4, the weight coefficient vector for the multichannel algorithm is obtained by computing the columns of the matrix $\mathbf{U}_N^{-\mathrm{T}}(k-1)$. For this purpose, we need to extend Lemma 2 of Chapter 4 to cover the multichannel case. Lemma 1 will still be valid here.

Lemma 5. *(Multichannel extension of Lemma 2)*
Let $\mathbf{u}_i(k) = \begin{bmatrix} u_{i,0}(k) & \dots & u_{i,NM-1}(k) \end{bmatrix}^T \in \mathbb{R}^{NM\times 1}$ *denote the* i^{th} *column of the upper triangular matrix* $\mathbf{U}_N^{-T}(k-1) \in \mathbb{R}^{NM\times NM}$. *Given* $\tilde{\mathbf{Q}}_{\theta f}(k) \in \mathbb{R}^{(NM+1)\times(NM+1)}$ *from Table 5.1, then* $\mathbf{u}_{i+jM}(k-1)$ *can be obtained from* $\mathbf{u}_{i+(j-1)M}(k-2)$ *using the following relation*

$$\begin{bmatrix} \tilde{\mathbf{r}}'(k) \\ \mathbf{u}_{i+jM}(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i+(j-1)M}(k-2) \\ \tilde{\mathbf{r}}_{i+(j-1)M}(k) \end{bmatrix}, \; i = 0, \dots, M-1; j = 0, \dots, N-1 \tag{5.53}$$

where $\tilde{\mathbf{r}}_i(k) = -[\mathbf{E}'_f(k)]^{-T}\mathbf{D}_{fq2}^T(k)\mathbf{u}_i(k-1)$. *Also for* $j = 1, \dots, M$ $\mathbf{u}_{-j}(k-2) = \mathbf{0}_{NM\times 1}$ *and* $\tilde{\mathbf{r}}_{-j}(k) = \mathbf{e}_{f,-j}(k)$, *where* $\mathbf{e}_{f,-j}(k)$ *is the* j^{th} *column of* $-[\mathbf{E}'_f(k)]^{-T}$.

Proof: See Appendix A5

Assuming vector $\mathbf{u}_{i+(j-1)M}(k-1)$ to be known, Lemmas 1 and 5 can be used to compute vector $\mathbf{u}_{i+(j-1)M}(k-2)$ and $\mathbf{u}_{i+jM}(k-1)$, respectively. Therefore all the column vectors corresponding to the i^{th} channel are obtained by iterating through all the possible values of j. Consequently, we obtain all the weights for the i^{th} channel. Note that in order to obtain the column vector $\mathbf{u}_{i+jM}(k-1)$ corresponding to a particular channel, we need to initialize Eq. (5.53) given in Lemma 5 properly, which means choosing the appropriate column of matrix $[\mathbf{E}_f^{(0)}(k-1)]^{-1}$. A schematic

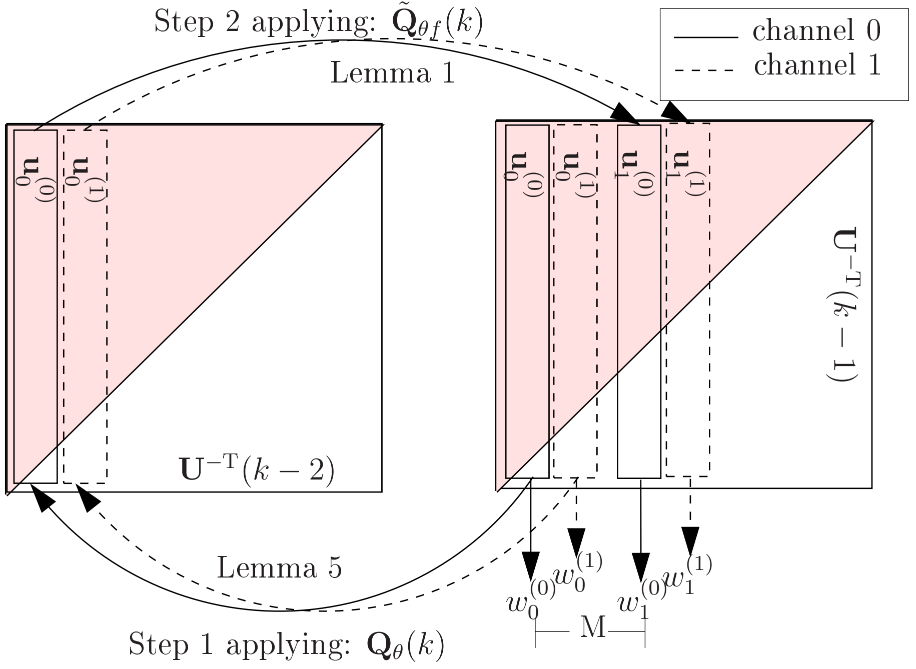


Figure 5.4: The procedure for updating $\mathbf{u}_i^n(k-1)$ for weight extraction in first the two channels, in MCFQRD-RLS a priori algorithm. Note that indices for some variables have been omitted

Table 5.3: "Weight Extraction" algorithm

```
$\tilde{\mathbf{r}}_l(k) = \mathbf{e}_{f,l}(k)$ for $l = -M, \ldots, -1$
$\mathbf{u}_l(k-2) = \mathbf{0}_{NM\times 1}$ for $l = -M, \ldots, -1$
for each $i = 0 : N-1$
 for each $j = 0 : M-1$
{
  Obtaining $\mathbf{u}_i(k-1)$
  $\begin{bmatrix} \tilde{\mathbf{r}}'(k) \\ \mathbf{u}_{i+jM}(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{u}_{i+(j-1)M}(k-2) \\ \tilde{\mathbf{r}}_{i+(j-1)M}(k) \end{bmatrix}, \begin{bmatrix} i = 0, \ldots, M-1 \\ j = 0, \ldots, N-1 \end{bmatrix}$
  Obtaining $z_{i+jM}(k)$
  $z_{i+jM}(k) = \frac{\mathbf{f}(k)\mathbf{u}_{i+jM}(k-1)}{\gamma(k)}$
  Obtaining $\mathbf{u}_i(k-2)$, to get the column vector of $\mathbf{U}^{-\mathrm{T}}(k-1)$
  $\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_{i+jM}(k-2) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k-1) \begin{bmatrix} z_{i+jM}(k) \\ \mathbf{u}_{i+jM}(k-1) \end{bmatrix}$
  Obtaining $\tilde{\mathbf{r}}_{i+jM}(k)$
  $\tilde{\mathbf{r}}_{i+jM}(k) = -[\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}_{fq2}^{\mathrm{T}}(k)\mathbf{u}_{i+jM}(k-2)$
  Obtaining the coefficients
  $w_{i,j}(k-1) = \mathbf{u}_{i+jM}^{\mathrm{T}}(k-1)\mathbf{d}_{q2}(k-1)$
}
```

for obtaining the column vectors is given in Figure 5.4. It is shown that, starting from $\mathbf{u}_0(k-1)$, first the column $\mathbf{u}_0(k-2)$ is obtained using Lemma 1 and then $\mathbf{u}_M(k-1)$ using Lemma 5. Both of them correspond to channel 1. These column vectors can be used to compute the weight coefficients $w_0(k-1)$ and $w_M(k-1)$ respectively. The other channels from 2 to M are treated in the same way to compute the corresponding weight coefficients. There are a total of NM weight coefficients so that it takes NM iterations to compute the whole coefficient vector. The weight extraction algorithm is summarized in Table 5.3.

5.4.2 Output filtering for burst type setup

One important application for output filtering is adaptive beamforming. The beamformer considered here is first adapted using training signals. After convergence the weight vector is not updated, and the output of interest is obtained by giving a different input to the weights. The equivalent output filtering algorithm can be used here in conjunction with the multichannel FQRD-RLS algorithm. The advantage of equivalent output filtering is that explicit weight extraction is not necessary.

In this section we extend the output filtering method for single channel in Section 4.2.2 to the multichannel case. Instead of having a single set of N coefficients, we now have M different coefficient vectors of N taps. There are also M input vectors. The multichannel counterpart of Eq. (4.6) is given as

$$y_N(k) = \begin{cases} \mathbf{w}_N^{\mathrm{T}}(k)\mathbf{x}_N(k) & k < k_F \\ \mathbf{w}_{N,F}^{\mathrm{T}}\tilde{\mathbf{x}}_N(k) & k \geq k_F \end{cases} \tag{5.54}$$

where k_F is the time instant after which the adaptive filter coefficient vector is not updated, i.e., $\mathbf{w}_{N,F} = \mathbf{w}_N(k_F - 1)$, and the multichannel input vector $\tilde{\mathbf{x}}_N(k)$ is independent of the input vector $\mathbf{x}_N(k)$. The output after weight freezing is given by

$$y_N(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k_F)\mathbf{U}_N^{-\mathrm{T}}(k_F)\tilde{\mathbf{x}}_N(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k_F)\mathbf{r}_N(k), \;\; k \geq k_F \tag{5.55}$$

As we are interested in using the multichannel FQRD-RLS algorithm to compute the output of the adaptive filter with frozen weights, the lemmas for the single-channel case have to be modified. Lemma 3 requires slight modifications; the matrix $\mathbf{U}^{-\mathrm{T}}(k)$ is replaced with the multichannel upper triangular matrix $\mathbf{U}_N^{-\mathrm{T}}(k)$, the input vector is replaced with the multichannel input vector $\mathbf{x}_N(k)$, and the rotation matrix is taken from Table 5.1. Similarly, Lemma 4 also needs to be modified.

Lemma 6. *(Multichannel extension of Lemma 4)*
Let $\mathbf{x}_N(k) \in \mathbb{R}^{NM\times 1}$ be the input data vector and let $\mathbf{u}_{r,i}(k) \in \mathbb{R}^{1\times NM}$ denote the i^{th} column vector of the upper triangular matrix $\mathbf{U}_N^{-1}(k) \in \mathbb{R}^{NM\times NM}$. Given $\tilde{\mathbf{Q}}_{\theta f}(k) \in \mathbb{R}^{(NM+1)\times(NM+1)}$ from Table 5.1, then $\mathbf{U}_N(k-1)^{-T}\mathbf{x}_N(k)$ can be obtained from $\mathbf{U}_N^{-T}(k-2)\mathbf{x}_N(k-1)$ using the following relation

$$\begin{bmatrix} \tilde{\mathbf{r}}'(k) \\ \mathbf{U}_N^{-T}(k-1)\mathbf{x}_N(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-T}(k-2)\mathbf{x}_N(k-1) \\ \tilde{\mathbf{r}}(k) \end{bmatrix} \tag{5.56}$$

where $\tilde{\mathbf{r}}(k) = [\mathbf{E}'_f(k)]^{-T}\mathbf{x}_{k+1} - [\mathbf{E}'_f(k)]^{-T}\mathbf{D}^T_{fq2}(k)\mathbf{U}_N^{-T}(k-1)\mathbf{x}_N(k)$.

Proof: See Appendix A6.

Lemma 6 can be considered for any input vector $\mathbf{x}_N(k)$. For sake of simplicity we define vectors $\mathbf{p}(k) = \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ and $\bar{\mathbf{p}}(k) = \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k)$. If vector $\mathbf{p}(k)$ is assumed known, the problem is to obtain its update such that the weights remain constant. In order to do so we first invoke Lemma 3 which results in vector $\bar{\mathbf{p}}(k)$. Then using Lemma 6 with $\bar{\mathbf{p}}(k)$ on the right hand side we can compute vector $\mathbf{p}(k+1)$. The algorithm is given in Table 5.4.

5.4.3 Output filtering for pre-equalizer type setup

This section describes the multichannel pre-equalizer using a multichannel FQRD-RLS algorithm. The output filtering for pre-equalizer has already been presented for the single-channel case in Section 4.2.3. The multichannel output filtering algorithm for the pre-equalizer setup uses only Lemma 6. The output of a multichannel adaptive filter is defined as

$$\tilde{y}_N(k) = \mathbf{w}_N^{\mathrm{T}}(k)\tilde{\mathbf{x}}_N(k) \tag{5.57}$$

Eq. (5.57) can also be written as,

$$\tilde{y}_N(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\mathbf{U}_N^{-\mathrm{T}}(k-1)\tilde{\mathbf{x}}_N(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k-1)\bar{\mathbf{p}}(k) \tag{5.58}$$

where $\bar{\mathbf{p}} = \mathbf{U}_N^{-\mathrm{T}}(k-1)\tilde{\mathbf{x}}_N(k)$. Using this definition in Lemma 6 we get

$$\begin{bmatrix} \tilde{\mathbf{r}}'(k) \\ \bar{\mathbf{p}}(k+1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \bar{\mathbf{p}}(k) \\ \tilde{\mathbf{r}}(k) \end{bmatrix} \tag{5.59}$$

To obtain the updated value, we use the updated rotation matrix $\tilde{\mathbf{Q}}_{\theta f}(k+1)$ as follows

$$\begin{bmatrix} \tilde{\mathbf{r}}'(k) \\ \bar{\mathbf{p}}(k+2) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k+1) \begin{bmatrix} \bar{\mathbf{p}}(k+1) \\ \tilde{\mathbf{r}}(k) \end{bmatrix} \tag{5.60}$$

Equivalent-output filtering for pre-equalizer requires a multichannel FQRD-RLS algorithm running in parallel in order to use its state variables i.e., the rotation matrix $\mathbf{Q}_\theta(k)$ and the matrix $[\mathbf{E}_f^{(0)}(k)]^{-\mathrm{T}}$ at each iteration. The algorithm is summarized in Table 5.5.

5.5 Experimental Results

5.5.1 Multichannel System Identification

The multichannel system consists of $M = 3$ channels with each channel having $N = 6$ taps. The SNR is 30 dBs. The multichannel FQRD-RLS algorithm was used to identify the system. After convergence the weight extraction algorithm was run to compute the filter weights. In order to verify how close the weights are to the true ones, an IQRD-RLS algorithm is used to identify the system. The weights obtained from the weight extraction method are then compared with those obtained by the IQRD-RLS algorithm. After 4000 iteration the difference of weights from both the algorithms is seen to be approximately -300 dB, which is within the numerical accuracy of the software used in simulation (MATLAB), as shown in Fig. 5.5.

5.5.2 Broadband beamformer

A uniform linear array with $M = 4$ antenna elements with spacing equal to half wavelength is used in a system with $K = 4$ signals, one being the desired signal and rest interference signals with the direction of arrivals 0^o, -35^o, 45^o, and 50^o, and $N = 6$ coefficients per channel. The SNR for the interfering signals was set to 40dB

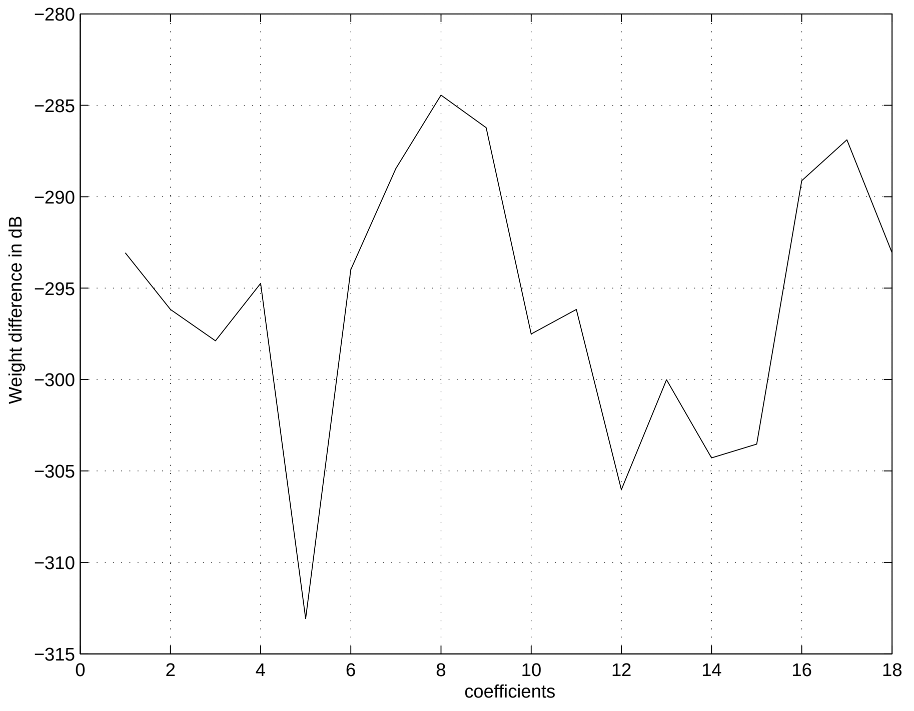


Figure 5.5: Comparison of weights obtained with the IQRD-RLS and the WE algorithm

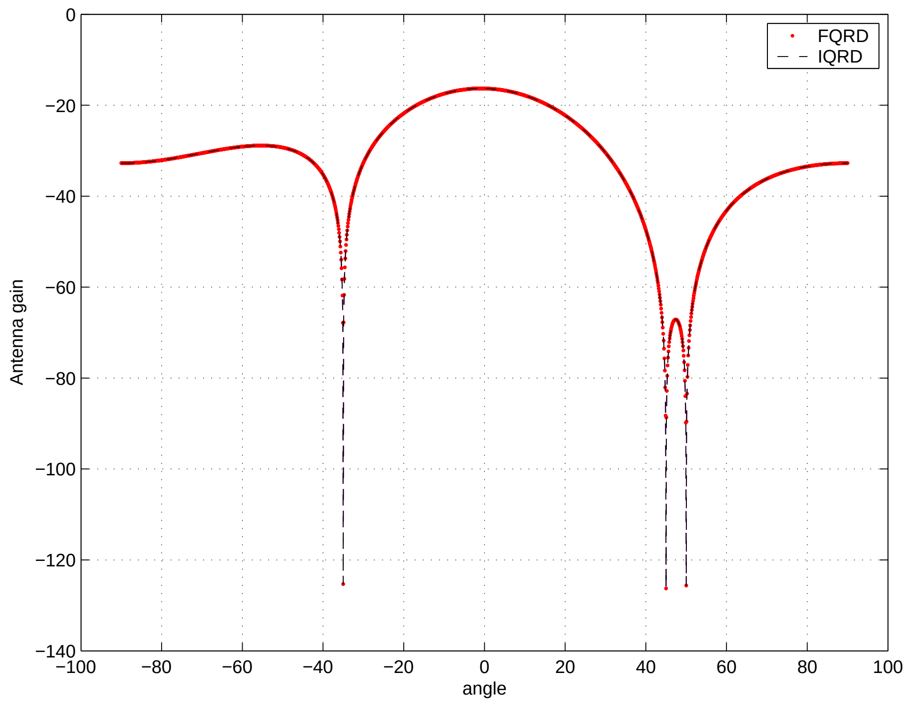


Figure 5.6: The beam pattern obtained from IQRD-RLS and FQRD-RLS algorithm

and 5 dB for the desired signal. The RLS and the FQRD-RLS algorithms are used for adapting the beamformer. The Weight Extraction algorithm is used to extract the weights of the FQRD-RLS algorithm. The beam pattern for both algorithms is shown for comparison in Fig. 5.6, this validates the weight extraction procedure.

5.6 Conclusions

In this chapter we have shown how multichannel FQRD-RLS algorithms can be used for applications other than the output error based ones (i.e., noise, echo cancellation etc.). First, we presented a literature review of the *a priori* multichannel FQRD-RLS algorithms based on backward prediction error. Next, three novel algorithms were derived to extend the range of applications of the multichannel FQRDR-RLS algorithm. These algorithms are based on lemmas that are generalized from those presented in Chapter 4. The weight extraction algorithm enables multichannel applications that require the explicit knowledge of the weights. The accuracy of the weight extraction algorithm is validated by a system identification and a broadband beamforming application. The example shows that the weight extraction algorithm and the IQRD-RLS algorithm give identical results.

Table 5.4: Equivalent-output filtering algorithm for pre-equalizer.

```
for each $0 \leq k < k_F$
{ Obtain $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}_{k+1}^{\mathrm{T}} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}$
    Obtain $\|\mathbf{E}_f(k+1)\|$:
    $\begin{bmatrix} \mathbf{0}_{1\times M} \\ \mathbf{E}'_f(k+1) \end{bmatrix} = \bar{\mathbf{Q}}_f(k+1) \begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{E}'_f(k) \end{bmatrix}$
    Obtain $\mathbf{a}_N(k)$
    $\mathbf{a}_{N+1}(k+1) = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{a}_N(k) \\ \mathbf{r}(k+1) \end{bmatrix}$
    Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k+1)$:
    $\begin{bmatrix} \mathbf{0}_{NM\times M} \\ \mathbf{E}_f^{(0)}(k+1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \|\mathbf{E}'_f(k+1)\| \end{bmatrix}$
    Obtaining $\mathbf{Q}_\theta(k+1)$:
    $\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_N(k+1) \end{bmatrix}$
    Joint Process Estimation:
    $\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}$
    $\varepsilon(k) = e_{q1}(k)\gamma(k)$
}
```

Output-filtering for pre-equalizer with input signal $\tilde{x}(k)$

```
Initialization:
$\bar{\mathbf{p}}(k_F) = \mathbf{0}$
for each $k \geq k_F$
{
  Obtaining $\mathbf{r}(k+1)$ from $\bar{\mathbf{r}}(k)$:
  $\begin{bmatrix} \tilde{\mathbf{r}}'(k) \\ \mathbf{p}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta f}(k_F-1) \begin{bmatrix} \bar{\mathbf{p}}(k) \\ \tilde{\mathbf{r}}_{i+jM}(k) \end{bmatrix}$
  Obtaining $z_i(k-1)$
  $z_i(k-1) = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{p}(k+1)/\gamma(k-1)$
  Updating $\bar{\mathbf{r}}(k+1)$
  $\begin{bmatrix} 0 \\ \lambda^{-1/2}\bar{\mathbf{p}}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k_F-1) \begin{bmatrix} z_i(k) \\ \mathbf{p}(k+1) \end{bmatrix}$
  Obtaining $tilde\mathbf{r}_{i+jM}(k)$
  $\tilde{\mathbf{r}}_{i+jM}(k) = -[\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}_{fq2}^{\mathrm{T}}(k)\bar{\mathbf{p}}(k+1)$
  Obtaining the output:
  $y_N(k) = \mathbf{d}_{q2}^{\mathrm{T}}(k_F)\mathbf{p}(k+1)$
}
```

Table 5.5: Equivalent-output filtering algorithm for pre-equalizer.

```
for each $k$
{ Obtain $\mathbf{d}_{fq2}(k)$:
    $\begin{bmatrix}\tilde{\mathbf{e}}^{\mathrm{T}}_{fq1}(k+1)\\ \mathbf{D}_{fq2}(k+1)\end{bmatrix} = \mathbf{Q}_{\theta}(k)\begin{bmatrix}\mathbf{x}^{\mathrm{T}}_{k+1}\\ \lambda^{1/2}\mathbf{D}_{fq2}(k)\end{bmatrix}$
    Obtain $\|\mathbf{E}_f(k+1)\|$:
    $\begin{bmatrix}\mathbf{0}_{1\times M}\\ \mathbf{E}'_f(k+1)\end{bmatrix} = \bar{\mathbf{Q}}_f(k+1)\begin{bmatrix}\tilde{\mathbf{e}}^{\mathrm{T}}_{fq1}(k+1)\\ \lambda^{1/2}\mathbf{E}'_f(k)\end{bmatrix}$
    Obtain $\mathbf{a}_N(k)$
    $\mathbf{a}_{N+1}(k+1) = \tilde{\mathbf{Q}}_{\theta f}(k)\begin{bmatrix}\mathbf{a}_N(k)\\ \mathbf{r}(k+1)\end{bmatrix}$
    Obtaining $\tilde{\mathbf{Q}}_{\theta f}(k+1)$:
    $\begin{bmatrix}\mathbf{0}_{NM\times M}\\ \mathbf{E}^{(0)}_f(k+1)\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k)\begin{bmatrix}\mathbf{D}_{fq2}(k+1)\\ \|\mathbf{E}'_f(k+1)\|\end{bmatrix}$
    Obtaining $\mathbf{Q}_{\theta}(k+1)$:
    $\begin{bmatrix}1/\gamma(k+1)\\ \mathbf{0}\end{bmatrix} = \mathbf{Q}_{\theta}(k+1)\begin{bmatrix}1\\ -\mathbf{a}_N(k+1)\end{bmatrix}$
    Joint Process Estimation:
    $\begin{bmatrix}e_{q1}(k+1)\\ \mathbf{d}_{q2}(k+1)\end{bmatrix} = \mathbf{Q}_{\theta}(k+1)\begin{bmatrix}d(k+1)\\ \lambda^{1/2}\mathbf{d}_{q2}(k)\end{bmatrix}$
    $\varepsilon(k) = e_{q1}(k)\gamma(k)$
}
```

Output-filtering for pre-equalizer with input signal $\tilde{x}(k)$

```
Initialization:
$\bar{\mathbf{r}}(k_F) = \mathbf{0}$
for each $k$
{
    Obtaining $\mathbf{r}(k+1)$ from $\bar{\mathbf{r}}(k)$:
    $\begin{bmatrix}\tilde{\mathbf{r}}'(k)\\ \mathbf{p}(k+1)\end{bmatrix} = \mathbf{Q}_{\theta f}(k_F-1)\begin{bmatrix}\bar{\mathbf{p}}(k)\\ \tilde{\mathbf{r}}_{i+jM}(k)\end{bmatrix}$
    Obtaining $\tilde{\mathbf{r}}_{i+jM}(k)$
    $\tilde{\mathbf{r}}_{i+jM}(k) = -[\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}^{\mathrm{T}}_{fq2}(k)\bar{\mathbf{p}}(k+1)$
    Obtaining the output:
    $y_N(k) = \mathbf{d}^{\mathrm{T}}_{q2}(k_F)\bar{\mathbf{r}}(k+1)$
}
```

Chapter 6

Conclusions and Future Work

This chapter concludes the results of the thesis and suggests future research topics.

6.1 Conclusion

The objective of this thesis was to obtain the weights embedded in the internal variables of the FQRD-RLS algorithm, in order to extend the range of applications of the single-channel and multichannel FQRD-RLS algorithms. The knowledge of weights enables new applications for FQRD-RLS algorithms such as system identification for linear and non-linear Volterra based systems, spectral analysis of the channel equalizer weights, and antenna beamforming for MIMO systems. In order to achieve the objective, a literature survey of QRD-RLS, Inverse QRD-RLS (IQRD-RLS), single channel and multichannel FQRD-RLS algorithms were presented. Thereafter, we provided new theoretical results that lead to algorithms which allow us to extract the weights of the single-channel and multichannel algorithms.

It was shown that the weight extraction method provides identical solution to that of any RLS-type algorithms, e.g., the IQRD-RLS algorithm used in this work. The theoretical results presented in the thesis were verified with the help of several experiments and the results were compared with those of the IQRD-RLS algorithm. The results from both approaches were found to be equal up to machine precision. The single-channel weight extraction algorithm for the FQRD-RLS algorithm was successfully applied in three applications, i.e., system identification, channel equalization and pre-equalization. The multichannel weight extraction algorithm was verified using multichannel system identification and broadband beamforming.

It can be concluded that the objective of the thesis was achieved. A novel technique for weight extraction was developed for both single and multichannel algorithms which extends the range of applications of the FQRD-RLS algorithms.

6.2 Future Work

There are two interesting directions where to further develop the results of the thesis:

1. The proposed method for weight extraction was verified using only the FQRD-RLS algorithm based on updating the *a priori* backward prediction errors. An immediate task would be to develop a common framework for all other fast QRD-RLS algorithms.

2. The solution proposed for the multichannel case can also be generalized by modifying it for the case of a multiple order filter. This is expected to enable efficient implementation of Volterra-based applications, such as Volterra system identification or Volterra based indirect learning architecture for nonlinear predistortion.

Appendix A

Proof of Lemmas

A.1 Proof of Lemma 1

The update equation for $\mathbf{U}^{-\mathrm{T}}(k-2)$ in the IQRD-RLS algorithm is given by

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

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

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

where $z_i(k-1)$ is the i^{th} element of vector $\mathbf{z}(k)$

$$z_i(k-1) = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1)/\gamma(k-1) \tag{A.3}$$

and the elements of vector $\mathbf{f}(k-1)$ and $\gamma(k-1)$ are obtained from the rotation matrix $\mathbf{Q}_{\theta}(k-1)$ as

$$\begin{bmatrix}\gamma(k-1)\\ \mathbf{f}(k-1)\end{bmatrix} = \mathbf{Q}_{\theta}(k-1)\begin{bmatrix}1\\ \mathbf{0}_{N\times 1}\end{bmatrix} \tag{A.4}$$

A.2 Proof of Lemma 2

The FQRD-RLS algorithm of Table 4.1 updates $\mathbf{a}(k)$ at every iteration as follows

$$\begin{bmatrix}\frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|}\\ \mathbf{a}(k)\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{a}(k-1)\\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|}\end{bmatrix} \tag{A.5}$$

where $e_b(k)$ and $e_f(k)$ are the backward and the forward prediction error values given by [21]:

$$\begin{aligned} e_f(k) &= x(k) - \mathbf{w}_f^{\mathrm{T}}(k-1)\mathbf{x}(k-1)\\ e_b(k) &= x(k-N-1) - \mathbf{w}_b^{\mathrm{T}}(k)\mathbf{x}(k) \end{aligned} \tag{A.6}$$

with $\mathbf{w}_f(k) = \mathbf{U}^{-T}(k)\mathbf{d}_{fq2}(k)$ [21] and $\mathbf{w}_b(k)$ denoting the forward and backward prediction weight vectors, respectively. Using Equation (A.6), the definition of $\mathbf{a}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$, and removing scalars and vectors related to input signal $x(k)$, the following relation is obtained from Equation (A.5)

$$\begin{aligned} &\begin{bmatrix}\frac{-\mathbf{w}_b^{\mathrm{T}}(k)}{\|\mathbf{e}_b(k-1)\|} & \frac{1}{\|\mathbf{e}_b(k-1)\|}\\ \mathbf{U}^{-\mathrm{T}}(k-1) & \mathbf{0}\end{bmatrix}\\ &= \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-2)\\ \frac{1}{\|\mathbf{e}_f(k-1)\|} & \frac{-\mathbf{w}_f^{\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k-1)\|}\end{bmatrix} \end{aligned} \tag{A.7}$$

Considering the partitioning of matrix $\mathbf{U}^{-\mathrm{T}}(k-1)$ into its column vectors $\mathbf{u}_i(k-1)$, the column version of (A.7) becomes

$$\begin{bmatrix}\frac{-w_{b,i}(k-1)}{\|\mathbf{e}_b(k-1)\|}\\ \mathbf{u}_i(k-1)\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{u}_{i-2}(k-2)\\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|}\end{bmatrix}, \quad i = 0,\dots,N-1 \tag{A.8}$$

where $w_{b,i}(k)$ and $w_{f,i}(k-1)$ are the i^{th} elements of the forward and backward prediction weight vectors, respectively. To account for the first column of (A.7) we initialize with $\mathbf{u}_{-1}(k-2) = \mathbf{0}_{N\times 1}$ and $w_{f,-1}(k-1) = -1$.

A.3 Proof of Lemma 3

It can be shown that the following relation holds for the QRD-RLS algorithms [29]

$$\begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \mathbf{Q}_{\theta}(k-1) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2) \end{bmatrix} \tag{A.9}$$

Pre-multiplying (A.9) with $\mathbf{Q}_{\theta}^{\mathrm{T}}(k-1)$ followed by post-multiplication with $\mathbf{x}(k)$ leads to

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta}^{\mathrm{T}}(k-1) \begin{bmatrix} \mathbf{z}^{T}(k-1)\mathbf{x}(k) \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} \tag{A.10}$$

where $\mathbf{z}(k) = -\frac{\mathbf{f}^{\mathrm{T}}(\mathbf{k})\mathbf{U}^{-\mathrm{T}}(k)}{\gamma(k)}$. The elements of vector $\mathbf{f}(k-1)$ and $\gamma(k-1)$ are obtained using Eq. (A.4).

A.4 Proof of Lemma 4

Combining the definition of $\mathbf{a}(k)$ with (3.41), we get

$$\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{A.11}$$

where $e_b(k)$ and $e_f(k)$ are the backward and the forward prediction error values given by [21]:

$$\begin{aligned} e_f(k) &= x(k) - \mathbf{w}_f^{\mathrm{T}}(k-1)\mathbf{x}(k-1) \\ e_b(k) &= x(k-N-1) - \mathbf{w}_b^{\mathrm{T}}(k)\mathbf{x}(k) \end{aligned} \tag{A.12}$$

with $\mathbf{w}_f(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{fq2}(k)$ [21] and $\mathbf{w}_b(k)$ denoting the forward and backward prediction weight vectors, respectively. All the values on the right hand side are known, therefore, Eq. (A.11) can be evaluated. This concludes the proof.

A.5 Proof of Lemma 5

The multichannel FQRD-RLS algorithm of Table 5.1 updates $\mathbf{a}_N(k) = \lambda^{-1/2}\mathbf{U}_N^{-\mathrm{T}}(k-1)\mathbf{x}_N(k)$ at every iteration as follows

$$\mathbf{a}_{N+1}(k+1) = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{a}_N(k)\\ \mathbf{r}(k+1)\end{bmatrix} \tag{A.13}$$

where $\mathbf{r}(k+1) = \lambda^{1/2}[\mathbf{E}'_f(k)]^{-\mathrm{T}}\tilde{\mathbf{e}}'_f(k+1)$ and

$$\tilde{\mathbf{e}}'_f(k+1) = x_{k+1} - \mathbf{W}_{Nf}^{\mathrm{T}}(k)\mathbf{x}_N(k) \tag{A.14}$$

with $\mathbf{w}_{Nf}(k) = \mathbf{U}_N^{-T}(k)\mathbf{D}_{fq2}(k)$ from Eq. (5.23). Using Equation (A.14), the definition of $\mathbf{a}_N(k)$, and removing vectors related to input signal $x(k)$, the following relation is obtained from Equation (A.5)

$$\begin{aligned}&\begin{bmatrix}[-\mathbf{E}_{bq1}(k)]^{-\mathrm{T}}\mathbf{D}_{bq2}^{\mathrm{T}}(k)\mathbf{U}_N^{-\mathrm{T}}(k-1) & [\mathbf{E}_{bq1}(k)]^{-\mathrm{T}}\\ \mathbf{U}_N^{-\mathrm{T}}(k) & \mathbf{0}_{NM\times M}\end{bmatrix}\\ &= \mathbf{Q}'_{\theta f}(k)\begin{bmatrix}\mathbf{0}_{NM\times M} & \mathbf{U}_N^{-\mathrm{T}}(k-1)\\ [\mathbf{E}'_f(k)]^{-\mathrm{T}} & -[\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}_{fq2}^{\mathrm{T}}(k)\mathbf{U}_N^{-\mathrm{T}}(k-1)\end{bmatrix}\end{aligned} \tag{A.15}$$

Considering the partition of matrix $\mathbf{U}_N^{-\mathrm{T}}(k-1)$ into its column vectors $\mathbf{u}_i(k-1)$, the column version of (A.7) becomes

$$\begin{bmatrix}\tilde{\mathbf{r}}'(k)\\ \mathbf{u}_{i-1+M}(k)\end{bmatrix} = \mathbf{Q}'_{\theta f}(k)\begin{bmatrix}\mathbf{u}_{i-1}(k-1)\\ \tilde{\mathbf{r}}_{i-1}(k)\end{bmatrix} \tag{A.16}$$

where $\tilde{\mathbf{r}}_{i-1}(k) = -[\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}^{\mathrm{T}}_{fq2}(k)\mathbf{u}_{i-1}(k-1)$. From Eq. (A.15), the first M columns correspond to initialization. In Eq. (A.16) we have $\mathbf{u}_{-j}(k-2) = \mathbf{0}_{NM\times 1}$ and $\tilde{\mathbf{r}}_{-j}(k) = \mathbf{e}_{f,-j}(k)$, where $\mathbf{e}_{f,-j}(k)$ is the j^{th} column of $-[\mathbf{E}'_f(k)]^{-\mathrm{T}}$.

A.6 Proof of Lemma 6

The multichannel FQRD-RLS algorithm of Table 5.1 updates $\mathbf{a}_N(k) = \lambda^{-1/2}\mathbf{U}_N^{-\mathrm{T}}(k-1)\mathbf{x}_N(k)$ at every iteration as follows

$$\mathbf{a}_{N+1}(k+1) = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{a}_N(k)\\ \mathbf{r}(k+1)\end{bmatrix} \tag{A.17}$$

where $\mathbf{r}(k+1) = \lambda^{1/2}[\mathbf{E}'_f(k)]^{-\mathrm{T}}\tilde{\mathbf{e}}'_f(k+1)$ and

$$\tilde{\mathbf{e}}'_f(k+1) = x_{k+1} - \mathbf{W}^{\mathrm{T}}_{Nf}(k)\mathbf{x}_N(k) \tag{A.18}$$

with $\mathbf{w}_{Nf}(k) = \mathbf{U}_N^{-T}(k)\mathbf{D}_{fq2}(k)$ from Eq. (5.23). Using Equation (A.18), the definition of $\mathbf{a}_N(k)$ the following relation is obtained from Equation (A.5)

$$\begin{aligned}&\begin{bmatrix}[-\mathbf{E}_{bq1}(k)]^{-\mathrm{T}}\mathbf{D}^{\mathrm{T}}_{bq2}(k)\mathbf{U}_N^{-\mathrm{T}}(k)\mathbf{x}_N(k+1) + [\mathbf{E}_{bq1}(k)]^{-\mathrm{T}}\mathbf{x}_{k-N}\\ \mathbf{U}_N^{-\mathrm{T}}(k)\mathbf{x}_N(k+1)\end{bmatrix}\\ &= \mathbf{Q}'_{\theta f}(k)\begin{bmatrix}\mathbf{U}_N^{-\mathrm{T}}(k-1)\mathbf{x}_N(k)\\ [\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{x}_{k+1} - [\mathbf{E}'_f(k)]^{-\mathrm{T}}\mathbf{D}^{\mathrm{T}}_{fq2}(k)\mathbf{U}_N^{-\mathrm{T}}(k-1)\mathbf{x}_N(k)\end{bmatrix}\end{aligned} \tag{A.19}$$

All the variables at the right hand side are known therfore the expression can be evaluated. This concludes the proof.

Appendix B

Alternative Proof of Lemmas

In this Appendix we provide lemmas that lead to different form of weight extraction algorithm. This approach differs from previous approach from computational aspect. In this case the algorithm will have less multiplications. However, divisions will be needed. The need for divisions my prohibit the use of these algorithms. They are included here for sake of completeness.

B.1 Proof of Lemma 1b

The proof of this lemma is given in two parts. The first part proves the existence of the relation given by Equation (4.3). The solution to Equation (4.3) without the *a priori* knowledge of the variable $*$ is given in the second part.

Part 1: In IQRD-RLS algorithm, the update equation for $\mathbf{U}^{-\mathrm{T}}(k-2)$ is given as

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

The above equation can also be written as follows

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

or, alternatively, for each column

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{u}_i(k-2) \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k-1) \begin{bmatrix} z_i(k-1) \\ \mathbf{u}_i(k-1) \end{bmatrix}, \text{where } i = 0, \ldots, N-1 \tag{B.3}$$

where, $z_i(k-1)$ is the i^{th} element of vector $\mathbf{z}(k)$ and it corresponds to the variable $*$ in Equation (4.3) which is unknown *a priori*. This concludes the proof of Part 1.
Part 2: Before starting the proof, it is important here to mention that the rotation matrix $\mathbf{Q}_\theta(k-1)$ can be written in the form of a sequence of rotation matrices, the details are mentioned in Section 2.4.2. The value of unknown $z_i(k-1)$ can be computed in two ways. The first approach requires the explicit construction of parts of the rotation matrix $\mathbf{Q}_\theta(k-1)$, therefore the solution provided is not attractive from a pratical point of view. However, the second approach computes the unknown iteratively and does not require the extra computation of the first approach. Therefore, second approach provides a practical solution.

Approach 1:
It is known that the partition of the rotation matrix is given as

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \tag{B.4}$$

Therefore, using Equation (4.3), the value of $*$ can be computed from the known values with the help of the following expression

$$* = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{u}_i(k-1)/\gamma(k-1) \tag{B.5}$$

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) and it requires $N(N+1)/2$ multiplications. The vector can then be stored for further use. This concludes the first approach

Approach 2:
Premultiplying Equation (4.3) with matrix $\mathbf{Q}_\theta(k-1)$ gives

$$\mathbf{Q}_\theta(k-1)\begin{bmatrix}0\\ \lambda^{-1/2}\mathbf{u}_i(k-2)\end{bmatrix} = \begin{bmatrix}z_i(k-1)\\ \mathbf{u}_i(k-1)\end{bmatrix} \tag{B.6}$$

or equivalently

$$\mathbf{Q}_{\theta_{N-1}}(k-1)\mathbf{Q}_{\theta_{N-2}}(k-1)\dots\mathbf{Q}_{\theta_0}(k-1)\begin{bmatrix}0\\ \lambda^{-1/2}\mathbf{u}_i(k-2)\end{bmatrix} = \begin{bmatrix}z_i(k-1)\\ \mathbf{u}_i(k-1)\end{bmatrix} \tag{B.7}$$

By carrying out the sequence of rotations in Equation (B.7) the value $z_i(k-1)$ is obtained by recursively updating $z_i^{(n)}(k-1)$ where n corresponds to the index of rotation matrix $\mathbf{Q}_{\theta_n}(k-1)$ in Equation (B.7). Therefore, after the last rotation has been applied, we have $z_i(k-1) = z_i^{(N-1)}(k-1)$. After applying the rotation matrix $\mathbf{Q}_{\theta_0}(k-1)$ onto vector $\begin{bmatrix}0 & \lambda^{-1/2}\mathbf{u}_i^{\mathrm{T}}(k-2)\end{bmatrix}^{\mathrm{T}}$ we get

$$\begin{bmatrix}\cos\theta_0 & \mathbf{0}^{\mathrm{T}} & -\sin\theta_0\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \sin\theta_0 & \mathbf{0}^{\mathrm{T}} & \cos\theta_0\end{bmatrix}\begin{bmatrix}0\\ \lambda^{-1/2}\mathbf{u}_i(k-2)\end{bmatrix} = \begin{bmatrix}z_i^{(0)}(k-1)\\ \lambda^{-1/2}u_{i,0}(k-2)\\ \vdots\\ \lambda^{-1/2}u_{i,N-2}(k-2)\\ u_{i,N-1}(k-1)\end{bmatrix} \tag{B.8}$$

From Equation (B.7) and Equation (B.8) the values of $z_i^{(0)}$ and $u_{i,N-1}(k-1)$ are identified as

$$\begin{aligned}-\sin\theta_0\lambda^{-1/2}u_{i,N-1}(k-2) &= z_i^{(0)}(k-1)\\ \cos\theta_0\lambda^{-1/2}u_{i,N-1}(k-2) &= u_{i,N-1}(k-1)\end{aligned} \tag{B.9}$$

Rearranging the order of the expressions in Equation (B.9) and solving for the unknowns $u_{i,N-1}(k-2)$ and $z_i^{(0)}(k-1)$ we get

$$\begin{aligned} u_{i,N-1}(k-2) &= \lambda^{1/2} u_{i,N-1}(k-1)/\cos\theta_0 \\ z_i^{(0)} &= -\sin\theta_0 \lambda^{-1/2} u_{i,N-1}(k-2) \end{aligned} \tag{B.10}$$

Next, $\mathbf{Q}_{\theta_1}(k-1)$ is applied to Equation (B.8) in order to solve $u_{i,N-2}(k-2)$ and $z_i^{(1)}(k-1)$, i.e.,

$$\begin{bmatrix} \cos\theta_1 & \mathbf{0}^{\mathrm{T}} & -\sin\theta_1 & 0 \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & 0 \\ \sin\theta_1 & \mathbf{0}^{\mathrm{T}} & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z_i^{(0)}(k-1) \\ \lambda^{-1/2} u_{i,0}(k-2) \\ \vdots \\ \lambda^{-1/2} u_{i,N-2}(k-2) \\ u_{i,N-1}(k-1) \end{bmatrix} = \begin{bmatrix} z_i^{(1)}(k-1) \\ \lambda^{-1/2} u_{i,0}(k-2) \\ \vdots \\ \lambda^{-1/2} u_{i,N-3}(k-2) \\ u_{i,N-2}(k-1) \\ u_{i,N-1}(k-1) \end{bmatrix} \tag{B.11}$$

As a result, we have

$$\begin{aligned} u_{i,N-2}(k-2) &= [\lambda^{1/2} u_{i,N-2}(k-1) - z_i^{(0)}(k-1)\sin\theta_1]/\cos\theta_1 \\ z_i^{(1)}(k-1) &= z_i^{(0)}(k-1)\cos\theta_1 - \lambda^{-1/2} u_{i,N-2}(k-2)\sin\theta_1 \end{aligned} \tag{B.12}$$

In general, after applying the j^{th} rotation we get

$$\begin{bmatrix} \cos\theta_j(k-1) & \mathbf{0}_{1\times(N-j-1)} & -\sin\theta_j(k-1) & \mathbf{0}_{1\times j} \\ \mathbf{0}_{(N-j-1)\times 1} & \mathbf{I}_{(N-j-1)} & \mathbf{0}_{(N-j-1)\times 1} & \mathbf{0}_{(N-j-1)\times j} \\ \sin\theta_j(k-1) & \mathbf{0}_{1\times(N-j-1)} & \cos\theta_i(k-1) & \mathbf{0}_{1\times j} \\ \mathbf{0}_{j\times 1} & \mathbf{0}_{i\times(N-j)} & \mathbf{0}_{j\times 1} & \mathbf{I}_j \end{bmatrix} \begin{bmatrix} z_i^{(j-1)}(k-1) \\ \lambda^{-1/2}u_{i,0}(k-2) \\ \vdots \\ \lambda^{-1/2}u_{i,N-1-j}(k-2) \\ u_{i,N-2-j}(k-1) \\ \vdots \\ u_{i,N-1}(k-1) \end{bmatrix}$$

$$= \begin{bmatrix} z_i^{(j)}(k-1) \\ \lambda^{-1/2}u_{i,0}(k-2) \\ \vdots \\ \lambda^{-1/2}u_{i,N-j}(k-2) \\ u_{i,N-1-j}(k-1) \\ \vdots \\ u_{i,N-1}(k-1) \end{bmatrix} \tag{B.13}$$

where the values $u_{i,N-1-j}(k-2)$ and $z_i^{(j)}(k-1)$ are computed as follows

$$\begin{aligned} u_{i,N-1-j}(k-2) &= [\lambda^{1/2}u_{i,N-1-j}(k-1) - z_i^{(j-1)}\sin\theta_j]/\cos\theta_j \\ z_i^{(j)} &= z_i^{(j-1)}\cos\theta_j - \lambda^{-1/2}u_{i,N-1-j}(k-2)\sin\theta_j \end{aligned} \tag{B.14}$$

Thus, after N rotations, all elements of $\mathbf{u}_i(k-2)$ are computed, and $z_i^{(N-1)}$ is the value of unknown $z_i(k-1)$. This concludes the proof of part 2.

B.2 Proof of Lemma 2b

The proof is given in two parts. In the first part the existence of the relation given in Lemma 2 is proved. In the second part, a solution to the relation without *a priori* knowledge of variables $*$ is given.

Part 1: The FQRD-RLS algorithm of Table 3.2 updates $\mathbf{a}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ at every iteration as follows

$$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{B.15}$$

where $e_b(k)$ and $e_f(k)$ are the backward and the forward prediction error values, respectively:

$$\begin{aligned} e_f(k) &= x(k) - \mathbf{w}_f^{\mathrm{T}}(k-1)\mathbf{x}(k-1) \\ e_b(k) &= x(k-N-1) - \mathbf{w}_b^{\mathrm{T}}(k)\mathbf{x}(k) \end{aligned} \tag{B.16}$$

with $\mathbf{w}_f(k)$ and $\mathbf{w}_b(k)$ denoting the forward and backward prediction weight vectors, respectively. Using Equation (B.16) and the definition of $\mathbf{a}(k)$, Equation (B.15) can be written as

$$\begin{bmatrix} \frac{-\mathbf{w}_b^{\mathrm{T}}(k)\mathbf{x}(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} + \frac{x(k-N-1)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)/\lambda^{1/2} \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)/\lambda^{1/2} \\ \frac{x(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} - \frac{\mathbf{w}_f^{\mathrm{T}}(k-1)\mathbf{x}(k-1)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{B.17}$$

or further simplified

$$\begin{aligned} &\begin{bmatrix} \frac{-\mathbf{w}_b^{\mathrm{T}}(k)}{\|\mathbf{e}_b(k-1)\|} & \frac{1}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \lambda^{-1/2}\mathbf{x}(k) \\ \lambda^{-1/2}x(k-N-1) \end{bmatrix} = \\ \tilde{\mathbf{Q}}_{\theta f}(k-1) &\begin{bmatrix} \mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-2) \\ \frac{1}{\|\mathbf{e}_f(k-1)\|} & \frac{-\mathbf{w}_f^{\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \begin{bmatrix} \lambda^{-1/2}x(k) \\ \lambda^{-1/2}\mathbf{x}(k-1) \end{bmatrix} \end{aligned} \tag{B.18}$$

In Equation (B.18) the two vectors $\lambda^{-1/2}\begin{bmatrix} x(k) & \mathbf{x}^{\mathrm{T}}(k-1) \end{bmatrix}^{\mathrm{T}}$ and

$\lambda^{-1/2}\begin{bmatrix}\mathbf{x}^{\mathrm{T}}(k) & x(k-N-1)\end{bmatrix}^{\mathrm{T}}$ are identical, leading to the following relation

$$\begin{bmatrix}\frac{-\mathbf{w}_b^{\mathrm{T}}(k)}{\|\mathbf{e}_b(k-1)\|} & \frac{1}{\|\mathbf{e}_b(k-1)\|}\\ \mathbf{U}^{-\mathrm{T}}(k-1) & \mathbf{0}\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{0} & \mathbf{U}^{-\mathrm{T}}(k-2)\\ \frac{1}{\|\mathbf{e}_f(k-1)\|} & \frac{-\mathbf{w}_f^{\mathrm{T}}(k-1)}{\|\mathbf{e}_f(k-1)\|}\end{bmatrix} \tag{B.19}$$

Considering the partition of matrix $\mathbf{U}^{-\mathrm{T}}(k-1)$ into its column vectors $\mathbf{u}_i(k-1)$, we have

$$\begin{bmatrix}\frac{-w_{b,i}(k)}{\|\mathbf{e}_b(k-1)\|}\\ \mathbf{u}_i(k-1)\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1)\begin{bmatrix}\mathbf{u}_{i-1}(k-2)\\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|}\end{bmatrix}, i=0,\ldots,N-1 \tag{B.20}$$

where $w_{b,i}(k)$ and $w_{f,i}(k-1)$ are the i^{th} elements of the forward and backward prediction weight vectors, respectively. Also, note that $\mathbf{u}_{-1}(k-1)=\mathbf{0}$ and $w_{f,-1}(k)=1$. This concludes the proof of the first part.

Part 2: In the second part we prove how Equation (B.20) can be evaluated without the *a priori* knowledge of $\frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|}$ and $\frac{-w_{b,i}(k)}{\|\mathbf{e}_b(k-1)\|}$.

The rotation matrix $\tilde{\mathbf{Q}}_{\theta f}(k-1)$ can be written as a sequence of rotation matrices as

$$\tilde{\mathbf{Q}}_{\theta f}(k-1) = \tilde{\mathbf{Q}}_{\theta_{N-1} f}(k-1)\tilde{\mathbf{Q}}_{\theta_{N-2} f}(k-1)\ldots\tilde{\mathbf{Q}}_{\theta_0 f}(k-1) \tag{B.21}$$

where $\tilde{\mathbf{Q}}_{\theta_j f}(k-1)$ is given as

$$\tilde{\mathbf{Q}}_{\theta_j f}(k-1) = \begin{bmatrix}\mathbf{I}_j & \mathbf{0}_{j\times 1} & \mathbf{0}_{i\times(N-j-1)} & \mathbf{0}_{j\times 1}\\ \mathbf{0}_{1\times j} & \cos\theta_j(k-1) & \mathbf{0}_{1\times(N-j-1)} & -\sin\theta_j(k-1)\\ \mathbf{0}_{(N-j-1)\times j} & \mathbf{0}_{(N-j-1)\times 1} & \mathbf{I}_{(N-j-1)} & \mathbf{0}_{(N-j-1)\times 1}\\ \mathbf{0}_{1\times j} & \sin\theta_j(k-1) & \mathbf{0}_{1\times(N-j-1)} & \cos\theta_j(k-1)\end{bmatrix} \tag{B.22}$$

with the definition of $\tilde{\mathbf{Q}}_{\theta f}(k-1)$ in Equation (B.21), Equation (B.20) becomes

$$\begin{bmatrix}\frac{-w_{b,i}(k)}{\|\mathbf{e}_b(k-1)\|}\\ \mathbf{u}_i(k-1)\end{bmatrix} = \tilde{\mathbf{Q}}_{\theta_{N-1} f}(k-1)\tilde{\mathbf{Q}}_{\theta_{N-2} f}(k-1)\ldots\tilde{\mathbf{Q}}_{\theta_0 f}(k-1)\begin{bmatrix}\mathbf{u}_{i-1}(k-2)\\ \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|}\end{bmatrix} \tag{B.23}$$

Let $z_f^{(j)}$ denote the recursively updated value of $z_f = \frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|}$ after the j^{th} rotation and let z_b be equal to $\frac{-w_{b,i}(k)}{\|\mathbf{e}_b(k-1)\|}$. Premultiplying Equation (B.23) with $\tilde{\mathbf{Q}}_{\theta f}^{\mathrm{T}}(k-1)$ gives

$$\begin{bmatrix} \mathbf{u}_{i-1}(k-2) \\ z_f(k-1) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta_0 f}^{\mathrm{T}}(k-1)\tilde{\mathbf{Q}}_{\theta_1 f}^{\mathrm{T}}(k-1)\ldots\tilde{\mathbf{Q}}_{\theta_{N-1} f}^{\mathrm{T}}(k-1) \begin{bmatrix} z_b(k-1) \\ \mathbf{u}_i(k-1) \end{bmatrix} \tag{B.24}$$

After applying rotation matrix $\tilde{\mathbf{Q}}_{\theta_{N-1} f}^{\mathrm{T}}(k-1)$ onto the right hand side of Equation (B.24), we get

$$\begin{bmatrix} z_b(k-1) \\ u_{i-1,0}(k-1) \\ \vdots \\ u_{i-1,N-2}(k-1) \\ u_{i-1,N-1}(k-2) \\ z_f^{(0)}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \cos\theta_{N-1}(k-1) & \sin\theta_{N-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & -\sin\theta_{N-1}(k-1) & \cos\theta_{N-1}(k-1) \end{bmatrix} \begin{bmatrix} z_b(k-1) \\ u_{i,0}(k-1) \\ \vdots \\ u_{i,N-2}(k-1) \\ u_{i,N-1}(k-1) \end{bmatrix} \tag{B.25}$$

The rotation matrix modifies only the last two terms of the vector $\begin{bmatrix} \mathbf{u}_{i-1}^{\mathrm{T}}(k-2) & z_f(k-1) \end{bmatrix}^{\mathrm{T}}$ From Equations (B.24) and (B.25), the values of $z_f^{(0)}(k-1)$ and $u_{i,N-2}(k-1)$ are identified as,

$$\begin{aligned} u_{i,N-2}(k-1) &= [u_{i-1,N-1}(k-2) - u_{i,N-1}(k-1) sin\theta_{N-1}(k-1)]/\cos\theta_{N-1}(k-1) \\ z_f^{(0)}(k-1) &= -u_{i,N-2}(k-1)\sin\theta_{N-1}(k-1) + u_{i,N-1}(k-1)\cos\theta_{N-1}(k-1) \end{aligned} \tag{B.26}$$

The value of $u_{i,N-1}(k-1)$ for $i > 0$ is known to be zero from the fact that $\mathbf{U}^{-\mathrm{T}}(k)$ is an upper triangular matrix. Next, the rotation matrix $\mathbf{Q}_{\theta_{N-2} f}(k-1)$ is applied to Equation (B.25), yielding

$$\begin{bmatrix} z_b(k-1) \\ u_{i-1,0}(k-1) \\ \vdots \\ u_{i-1,N-3}(k-1) \\ u_{i-1,N-2}(k-2) \\ u_{i-1,N-1}(k-2) \\ z_f^{(1)}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \cos\theta_{N-2}(k-1) & 0 & \sin\theta_{N-2}(k-1) \\ 0 & 0 & 1 & 0 \\ \mathbf{0}^{\mathrm{T}} & -\sin\theta_{N-2}(k-1) & 0 & \cos\theta_{N-2}(k-1) \end{bmatrix} \begin{bmatrix} z_b(k-1) \\ u_{i-1,0}(k-1) \\ \vdots \\ u_{i-1,N-2}(k-1) \\ u_{i-1,N-1}(k-2) \\ z_f^{(0)}(k-1) \end{bmatrix} \tag{B.27}$$

From the above equation the unknowns can be calculated

$$\begin{aligned} u_{i,N-3}(k-1) &= [u_{i-1,N-2}(k-2) - z_f^{(0)}(k-1) sin\theta_{N-2}(k-1)]/\cos\theta_{N-2}(k-1) \\ z_f^{(1)}(k-1) &= -u_{i,N-3}(k-1)\sin\theta_{N-2}(k-1) + z_f^{(0)}(k-1)\cos\theta_{N-2}(k-1) \end{aligned} \tag{B.28}$$

In general, after applying the j^{th} rotation we get

$$\begin{aligned} u_{i,N-1-j}(k-1) &= [u_{i-1,N-j}(k-2) - z_f^{(i-2)}(k-1) sin\theta_{N-j}(k-1)]/\cos\theta_{N-j}(k-1) \\ z_f^{(j-1)}(k-1) &= -u_{i,N-1-j}(k-1)\sin\theta_{N-j}(k-1) + z_f^{(j-2)}(k-1)\cos\theta_{N-j}(k-1) \end{aligned} \tag{B.29}$$

The value of the unknown vector $\mathbf{u}_i(k-1)$ can, therefore, be computed without the *a priori* knowledge of variables $\frac{-w_{f,i-1}(k-1)}{\|\mathbf{e}_f(k-1)\|}$ and $\frac{-w_{b,i}(k)}{\|\mathbf{e}_b(k-1)\|}$. This concludes the proof for the second part.

B.3 Proof of Lemma 3b

The proof of this lemma is given in two parts. The first part proves the existence of the relation given by Equation (4.8). The solution to Equation (4.8) without the *a priori* knowledge of the variable $*$ is given in the second part.

Part 1: In IQRD-RLS algorithm, the update equation for $\mathbf{U}^{-\mathrm{T}}(k-2)$ is given as

$$\begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \mathbf{Q}_{\theta}(k-1) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2) \end{bmatrix} \tag{B.30}$$

The above equation can also be written as follows

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2) \end{bmatrix} = \mathbf{Q}_{\theta}^{\mathrm{T}}(k) \begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} \tag{B.31}$$

post-multiplying Equation (B.31) with $\mathbf{x}(k)$ yields

$$\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k) \end{bmatrix} = \mathbf{Q}_{\theta}^{\mathrm{T}}(k-1) \begin{bmatrix} \mathbf{z}^{T}(k-1)\mathbf{x}(k) \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} \tag{B.32}$$

where, $\mathbf{z}^{T}(k-1)\mathbf{x}(k)$ corresponds to the variable $*$ in Equation (4.8) which is unknown *a priori*. This concludes the proof of Part 1.

Part 2: Before starting the proof it is important here to mention that the rotation matrix $\mathbf{Q}_{\theta}(k-1)$ can be written in the form of a sequence of rotation matrices, the details are mentioned in Section 2.4.2. Below we show how $\mathbf{z}^{\mathrm{T}}(k-1)\mathbf{x}(k)$ can be computed using two approaches. The first approach requires the explicit construction of vector $\mathbf{f}(k)$ related to the rotation matrix $\mathbf{Q}_{\theta}(k-1)$. Therefore the solution provided is not attractive from a computational complexity point-of-view. However, the second approach computes the unknowns iteratively and does not require the extra computation of the first approach. Therefore, second approach provides a practical solution. For simplicity, let us call $z(k-1) = \mathbf{z}^{\mathrm{T}}(k-1)\mathbf{x}(k)$; also let the i^{th} column of $\mathbf{U}^{-1}(k)$ be denoted by $\mathbf{u}_{r,i}(k)$.

Approach 1:

It is known that the partition of the rotation matrix is given as

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \tag{B.33}$$

Therefore, using Equation (4.8), the value of $*$ can be computed from the known values with the help of the following expression

$$* = -\mathbf{f}^{\mathrm{T}}(k-1)\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)/\gamma(k-1) \tag{B.34}$$

the elements of vector $\mathbf{f}(k-1)$ are given by $f_j(k-1) = \sin\theta_{N-j-1}(k-1)\prod_{i=0}^{N-j-2}\cos\theta_i(k-1)$ and it requires $N(N+1)/2$ multiplications. The vector can then be stored for further use. This concludes the first approach

Approach 2:

Premultiplying Equation (4.8) with matrix $\mathbf{Q}_\theta(k-1)$ gives

$$\mathbf{Q}_\theta(k-1)\begin{bmatrix} 0 \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k) \end{bmatrix} = \begin{bmatrix} z(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} \tag{B.35}$$

The proof of this approach follows the same steps mentioned in the proof of approach 2 in lemma 1 by replacing the vector $\mathbf{u}_i(k-1)$ and $\mathbf{u}_i(k-2)$ by vectors $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ and $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k)$ respectively. As a result we obtain the following recursion.

$$\begin{aligned} \mathbf{u}_{r,N-1-j}^{\mathrm{T}}(k-2)\mathbf{x}(k) &= [\lambda^{1/2}\mathbf{u}_{r,N-1-j}^{\mathrm{T}}(k-1)\mathbf{x}(k) - z^{(j-1)}\sin\theta_j]/\cos\theta_j \\ z^{(j)} &= z^{(j-1)}\cos\theta_j - \lambda^{-1/2}\mathbf{u}_{r,N-1-j}(k-2)\mathbf{x}(k)\sin\theta_j \end{aligned} \tag{B.36}$$

where $\mathbf{u}_{r,i}^{\mathrm{T}}(k-1)$ corresponds to the i^{th} row vector of $\mathbf{U}^{-\mathrm{T}}(k-1)$. Thus, after N rotations, all elements of $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k)$ are computed, and $z^{(N-1)}$ is the value of unknown $z(k-1)$. This concludes the proof of Part 2.

B.4 Proof of Lemma 4b

The proof is given in two parts. In the first part the existence of the relation given in Lemma is proved. In the second part, a solution to the relation without *a priori* knowledge of variables $*$ is given.

Part 1: The FQRD-RLS algorithm of Table 3.2 updates $\mathbf{a}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ at every iteration as follows

$$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{a}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{a}(k-1) \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{B.37}$$

where $e_b(k)$ and $e_f(k)$ are the backward and the forward prediction error values, respectively. Using the definition of $\mathbf{a}(k)$, Equation (B.37) can be written as

$$\begin{bmatrix} \frac{e_b(k)}{\lambda^{1/2}\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)/\lambda^{1/2} \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1)/\lambda^{1/2} \\ \frac{e_f(k)}{\lambda^{1/2}\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{B.38}$$

or further simplified

$$\begin{bmatrix} \frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|} \\ \mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \end{bmatrix} = \tilde{\mathbf{Q}}_{\theta f}(k-1) \begin{bmatrix} \mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k-1) \\ \frac{e_f(k)}{\|\mathbf{e}_f(k-1)\|} \end{bmatrix} \tag{B.39}$$

This concludes the proof of the first part.

Part 2: The part 2 can be proved following the same steps as mentioned in the proof of part 2 for lemma 2 by replacing the vector $\mathbf{u}_i(k-1)$ and $\mathbf{u}_i(k-2)$ with $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ and $\mathbf{U}^{-\mathrm{T}}(k-2)\mathbf{x}(k)$ respectively. As a result we obtain the following recursion.

$$\begin{aligned} \mathbf{u}^{\mathrm{T}}_{r,N-1-j}(k-1)\mathbf{x}(k) &= [\mathbf{u}^{\mathrm{T}}_{r,N-j}(k-2)\mathbf{x}(k) - z_f^{(i-2)} sin\theta_{N-j}(k-1)]/\cos\theta_{N-j}(k-1) \\ z_f^{(j-1)} &= -\mathbf{u}_{r,N-1-j}(k-1)\mathbf{x}(k)\sin\theta_{N-j}(k-1) + z_f^{(j-2)}\cos\theta_{N-j}(k-1) \end{aligned} \tag{B.40}$$

where $\mathbf{u}_{r,i}^{\mathrm{T}}(k-1)$ corresponds to the i^{th} row vector of $\mathbf{U}^{-\mathrm{T}}(k-1)$. The value of the unknown vector $\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$ can therefore be computed without the *a priori* knowledge of variables $\frac{e_f(k-1)}{\|\mathbf{e}_f(k-1)\|}$ and $\frac{e_b(k)}{\|\mathbf{e}_b(k-1)\|}$. This concludes the proof for part 2.

Bibliography

[1] C. Breining *et al.*, “Acoustic echo control: An application of very-high-order adaptive filters,” *IEEE Signal Processing Magazine*, vol. 16, pp. 42–69, July 1999.

[2] S. U. Qureshi, “Adaptive Equalization,” in *Proc. IEEE*, vol. 73, no. 9, Sept. 1985, pp. 1349–1387.

[3] U. Madhow and M. L. Honig, “MMSE interference suppression for direct sequence spread-spectrum CDMA,” *IEEE Trans. on Comm.*, vol. 42, no. 12, pp. 3178–3188, 1994.

[4] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice Hall, 1991.

[5] J. Cioffi, “Limited-precision effects in adaptive filtering,” *IEEE Transactions on Circuits and Systems*, vol. 34, no. 7, pp. 821 – 833, Jul 1987.

[6] M. G. Bellanger, “The FLS-QR algorithm for adaptive filtering,” *Signal Processing*, vol. 17, pp. 291–304, March 1984.

[7] J. M. Cioffi, “The fast adaptive ROTOR’s RLS algorithm,” *IEEE Trans. on Acoust., Speech and Signal Processing*, vol. 38, pp. 631–653, April 1990.

[8] J. A. Apolinário. Jr. and P. S. R. Diniz, “A new fast QR algorithm based on a priori erros,” *IEEE Signal Processing Letters*, vol. 4, no. 11, pp. 307–309, November 1997.

[9] A. A. Rontogiannis and S. Theodoridis, “New fast inverse QR least squares adaptive algorithms,” in *IEEE International Conference on Acoustic, Speech and Signal Processing.* Detroit, MI, USA: IEEE, 1995, pp. 1412–1415.

[10] M. D. Miranda and M. Gerken, “A hybrid leat squares QR-lattice algorithm using a priori errors,” *IEEE Transactions on Signal Processing*, vol. 45, no. 12, pp. 2900–2911, December 1997.

[11] C. R. Ward *et al.*, “Application of a systolic array to adaptive beamforming,” in *IEE Proceedings*, vol. 131, London, England, 1984, pp. 638–645.

[12] D. Psaltis, A. Sideris, and A. A. Yamamura, “A multilayered neural network controller,” *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 17 – 21, 1988.

[13] S. L. Gay and S. Tavathia, “The fast affine projection algorithm,” in *International conference of Acoustics, Speech and Signal Processing*, Detroit, USA, May 1995, pp. 3023–3026.

[14] S. M. Kuo and D. R. Morgan, “Active noise control: a tutorial overview,” in *Proceedings of the IEEE*, vol. 87, no. 6, 1999, pp. 943–975.

[15] D. Gesbert and P. Duhamel, “Unbiased blind adpative channel identification,” *IEEE Transaction on Signal Processing*, vol. 48, no. 1, pp. 148–158, January 2000.

[16] D. P. Taylor, G. M. Vitetta, B. D. Hart, and A. Mammela, “Wireless channel equalization,” *European Transactions on Telecommunications*, vol. 9, no. 2, pp. 117–143, 1998.

[17] J. G. Proakis, *Digital Communications.* McGraw-Hill, 1995.

[18] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Englewood-Cliffs, NJ, USA: Prentice-Hall, 1985.

[19] W. Harrison, L. Jae, and E. Singer, “Adaptive noise cancellation in a fighter cockpit environment,” in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP*, vol. 9, no. 1, Mar 1984, pp. 61–64.

[20] S. Werner, Reduced Complexity Adaptive Filtering Algorithms with Application to Communication Systems, D.Sc. (Tech) Thesis, Helsinki University of Technology, Department of Electrical Engineering, Signal Processing Laboratory, Espoo, Finland, 2002.

[21] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation.* Norwell, MA, USA: Kluwer Academic Press, 1997.

[22] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[23] H. Leung and S. Haykin, "Stability of recursive QRD-LS algorithms using finite precision systolic array implementation," *IEEE Trans. on Acoust., Speech and Signal Processing*, vol. 37, pp. 760–763, 1989.

[24] S. Ghirnikar, A.; Alexander, "Stable recursive least squares filtering using an inverse QR decomposition," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 1990, pp. 1623 – 1626.

[25] F. Ling, "Givens rotation based least squares lattice and related algorithms," *IEEE Transactions on Signal Processing*, vol. SP-39, no. 7, pp. 1541–1551, July 1991.

[26] J. A. Apolinário. Jr., M. G. Siqueira, and 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.

[27] 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. SP-39, no. 4, pp. 879–891, April 1991.

[28] A. Sayed, *Fundamentals of Adaptive Filtering.* NY: Wiley, 2003.

[29] S. T. Alexander and 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.

[30] A. Ramos, “Novos algoritmos adaptiativos QRD-RLS multicanais rápidos e suas applicacões LCMV,” M.Sc. Thesis, Instituto Militar de Engenharia, Rio de Janeiro, Brazil, 2004.