NEW ALGORITHMS OF ADAPTIVE FILTERING: LMS WITH DATA-REUSING AND FAST RLS BASED ON QR DECOMPOSITION

José Antonio Apolinário Junior

THESIS SUBMITTED TO THE GRADUATE ENGINEERING SCHOOL (COPPE) OF THE FEDERAL UNIVERSITY OF RIO DE JANEIRO IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF SCIENCE IN ELECTRICAL ENGINEERING.

Approved by:

Prof. Paulo S. R. Diniz, Ph.D.
(Supervisor)

Dr. Mariane R. Petraglia, Ph.D.

Dr. Max Gerken, Dr.Ing.

Dr. Abraham Alcaim, Ph.D.

Dr. Marcello L. R. de Campos, Ph.D.

RIO DE JANEIRO, RJ - BRAZIL

JUNE 1998

To my wife *Ana Luisa*, my daughter *Isabela*, and my son *Eduardo*. I love you all.

Acknowledgments

It has been more than three years I have been working in my D.Sc. program and during this period I was very fortunate to work with very good people and to have the support of them all.

First of all, I thank very much my supervisor, Dr. Paulo Diniz, for his supervision, incentive, and friendship. I am sincerely grateful to my friend and master (*Jedi*) Diniz whose competence, professionalism, and good sense of humor, allowed me to overcome challenges and difficulties with more confidence. I share with him the laurels of this work. I am also thankful to the people at COPPE/UFRJ, the teachers, the students, and the personnel who in one way or another helped me in this difficult task of becoming a doctor. I am also grateful to the Hall of Justice and the super friends Robin, (Supper) He-Man, A.-M., and, more recently, Fyllo-Man. Thanks to Rogério Guedes, Dr. Sérgio Lima Netto, Dr. A.-M., and *tack* to Master of Caipiroskas *in spe* Stefan Werner. In special, thanks to my friend, coauthor, and opponent Marcello Campos for the friendship, partnership, and hopeful mercy (I wrote this before *the* day). It is also my pleasure to thank Prof. Timo Laakso and his team from the Laboratory of Telecommunications Technology, Helsiniki University of Technology, for the fruitful cooperation during the period I was there as a visiting researcher, *kiitos*. I am also indebted to the Military Institute of Engineering, Brazilian Army, for the opportunity I was given to pursue my doctorate.

This work was partially supported by CAPES — Ministry of Education, Brazil.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

NEW ADAPTIVE FILTERING ALGORITHMS: LMS WITH DATA-REUSING AND FAST RLS BASED ON QR DECOMPOSITION

José Antonio Apolinário Junior

June/1998

Advisor: Paulo Sérgio Ramirez Diniz

Department: Electrical Engineering

This work presents two new adaptive filtering algorithms: one of them a least-mean-square type with data-reusing and the other one a fast recursive least-squares based on QR decomposition.

The first part of this work presents and analyzes a new LMS-like algorithm, the binormalized data-reusing least mean-square (BNDR-LMS) algorithm, which compares favorably with other normalized LMS-like algorithms when the input signal is correlated. For this algorithm, convergence analyses in the mean and in the mean-squared are presented and a closed-form formula for the mean-square error

is provided for white input signals as well as its extension to the case of colored input signal. A simple model for the input-signal vector which imparts simplicity and tractability to the analysis of second-order statistics is fully described. The methodology is readily applicable to other adaptation algorithms of difficult analysis. Simulation results show the performance of the BNDR-LMS algorithm for different scenarios and validate the analysis and ensuing assumptions. An extension of the BNDR-LMS algorithm to include constraints is also derived in order to apply this algorithm to a direct-sequence code-division multiple access (DS-CDMA) mobile receiver. Moreover, a step-size optimization is proposed to accomplish this practical application with the conflicting requirements of fast convergence and minimum steady-state mean-square error (MSE).

In the second part of this work, the principles behind the triangularization of the weighted input data matrix via QR decomposition and the type of errors used in the updating process are exploited in order to investigate the relationships among different fast algorithms of the QR decomposition family. The algorithms are classified according to a general framework and a new fast QR algorithm based on Givens rotation using *a priori* forward errors is introduced along with the detailed description of the four classified fast QR algorithms and two lattice versions. Finally, a contribution towards the finite precision analysis of the fast QR algorithms is presented.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BNDR-LMS	binormalized data-reusing LMS
DR-LMS	data-reusing LMS
DS-CDMA	direct sequence code-division multiple access
FIR	finite-duration impulse response
FQR	fast QR
FQR_POS_B	fast QR updating *a posteriori* backward prediction errors
FQR_POS_F	fast QR updating *a posteriori* forward prediction errors
FQR_PRI_B	fast QR updating *a priori* backward prediction errors
FQR_PRI_F	fast QR updating *a priori* forward prediction errors
FQR-L	fast QR lattice
FTRLS	fast transversal RLS
GSC	general sidelobe canceler
IIR	infinite-duration impulse response
IQR	inverse QR
LMS	least mean-square
LRLS	lattice RLS
LS	least-squares
MSE	mean-square error
NLMS	normalized LMS
NNDR-LMS	normalized new data-reusing LMS
QRD	QR decomposition
RLS	recursive least-squares
UNDR-LMS	unnormalized new data-reusing LMS

Chapter 1

Adaptive Filtering

1.1 Introduction

In the last decades, the field of digital signal processing, and particularly adaptive signal processing, has developed enormously due to the increasingly available technology for the implementation of the emerging algorithms. These algorithms have been applied to an extensive number of problems including noise and echo canceling, channel equalization, signal prediction, adaptive arrays as well as many others. A particular application exemplified in Chapter 3 is the adaptive multiuser detection at a mobile DS-CDMA receiver which shows the usefulness of adaptive filtering techniques in mobile communication systems.

An adaptive filter may be defined as a self-modifying digital filter that adjusts its coefficients in order to minimize an error function. This error function or cost function is obtained from the difference between the "reference" or "desired" signal and the filter's output. Adaptive filtering algorithms are in fact closely related to classical optimization techniques although in the latter, all calculations are carried out "offline". Moreover, an adaptive filter is sometimes expected to track the optimum filter (or Wiener filter as is called the optimum filter in the mean-square sense for a stationary environment) in a slowly varying environment.

In order to compare the wide variety of algorithms available in the literature of adaptive filtering, the following aspects must be taken into account [1].

- Structure. The manner in which the algorithm is implemented may be basically divided in two types for the FIR (finite impulse response) adaptive filters: transversal filter (or tapped-delay line) and lattice structure. IIR (infinite impulse response) adaptive filters constitutes another field in adaptive filtering where we can find a number of realizations.

- Rate of convergence, misadjustment and tracking. In a noiseless (no measurement and/or modeling noise) situation, the coefficients of an adaptive filter can converge fast or slowly to the optimum solution. The coefficients, in general, will not reach the optimum values but will stay close to the optimum. Misadjustment is a measure of how close these coefficients (the estimated and the optimum) are in steady-state. It can be taken as a general rule that for a given algorithm the faster you make it converge the higher will be the misadjustment. In nonstationary environments, the algorithm must be fast enough to track the time-varying optimum coefficients.

- Computational aspects. It can be included here the computational complexity as well as the performance of the algorithm in a limited precision environment. The effort in obtaining fast versions of more complex algorithms results from the desire to reduce the computational requirements to a minimal number of operations and to reduce the size of memory necessary to run these algorithms in real time applications. On the other hand, a limited precision environment generates quantization errors which drive the attention of designers to numerical stability, numerical accuracy and robustness of the algorithm.

1.2 Basic Concepts of Adaptive Filters

The definition of the cost function gives rise to the large number of alternative adaptive filtering algorithms. The mean-squared error (MSE) is used in the least mean-square (LMS) and LMS-based algorithms while the least squares leads to the recursive least-squares (RLS) schemes. The RLS algorithms may be subdivided in

conventional, lattice, fast transversal and based on QR-decomposition.

The basic configuration of an adaptive filter is illustrated in Figure 1.1. The input signal is denoted by $x(k)$ where k is the iteration number. The reference signal $d(k)$ may be seen (as in an FIR system identification problem) as the desired signal plus observation noise or $\boldsymbol{x}^T(k)\boldsymbol{w}_0 + n(k)$ where $\boldsymbol{w}_0$ is the optimum coefficient vector and $\boldsymbol{x}(k)$ is the vector $[x(k)\, x(k-1) \cdots x(k-N)]^T$, with N being the order of the adaptive filter. The error signal is $e(k) = d(k) - y(k)$, where $y(k)$ is the output of the adaptive filter. This error will be used by the adaptation algorithm to update the coefficient vector $\boldsymbol{w}(k)$ of the adaptive filter.

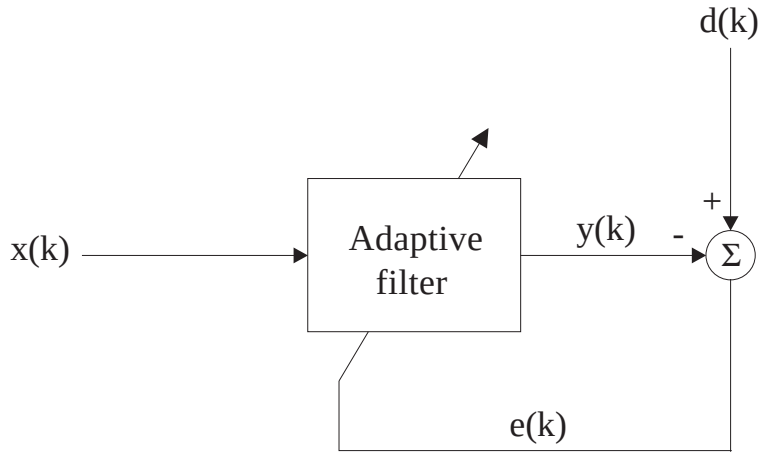

Figure 1.1: Basic configuration of an adaptive filter

1.2.1 The Mean-Square Error and the LMS-based Algorithms

The mean-square error (MSE) is defined as

$$\xi(k) = E[e^2(k)] = E[(d(k) - y(k))^2] \tag{1.1}$$

where $y(k) = \sum_{i=0}^{N} w_i(k)x(k-i) = \boldsymbol{x}^T(k)\boldsymbol{w}(k)$, with $\boldsymbol{w}(k) = [w_0(k)w_1(k) \cdots w_N(k)]^T$ being the coefficient vector.

The gradient vector of the MSE related to the tap-weighted coefficients is

$$\nabla_{\boldsymbol{w}(k)}\xi(k) = -2\boldsymbol{p} + 2\boldsymbol{p}\boldsymbol{w}(k) \tag{1.2}$$

where $\boldsymbol{p} = E[d(k)\boldsymbol{x}(k)]$ is the cross-correlation vector between the desired and the input signals, and $\boldsymbol{R} = E[\boldsymbol{x}(k)\boldsymbol{x}^T(k)]$ is the input signal correlation matrix. The Wiener solution is obtained by equating the gradient vector to zero and assuming that $\boldsymbol{R}$ is nonsingular, and is given by

$$\boldsymbol{w}_0 = \boldsymbol{R}^{-1}\boldsymbol{p} \tag{1.3}$$

We can approach the Wiener solution by searching in the direction of the estimate of the gradient vector (steepest-descent-based algorithm) using a step-size μ. One possible and very simple solution is obtained using instantaneous estimates of $\boldsymbol{R}$ and $\boldsymbol{p}$ given by $\boldsymbol{x}(k)\boldsymbol{x}^T(k)$ and $d(k)\boldsymbol{x}(k)$. The resulting gradient-based algorithm is known as the *least mean-square* (LMS) algorithm.

The LMS algorithm is very popular and has been widely used due to its simplicity. Its convergence speed, however, is highly dependent on the eigenvalue spread (conditioning number) of the input-signal autocorrelation matrix [2, 1]. Alternative schemes which try to improve this performance at the cost of minimum additional computational complexity have been proposed and extensively discussed in the past [2]–[5].

One approach that has been successfully employed in situations where signal statistics are unknown is the online calculation of the convergence factor which takes part in updating the filter coefficients [4, 6]. The normalized LMS (NLMS) algorithm can be included in this category [4, 7]. Also belonging to this class of algorithms are the data-reusing algorithms which will be later described in more details.

1.2.2 The Least-Squares and the RLS Algorithms

Another objective function which is deterministic and convenient to be used in stationary environment is the least squares (LS) given by $\frac{1}{k+1}\sum_{i=0}^{k} e^2(i)$. The computation of the least squares in a recursive form resulted in a family of algorithms known as recursive least-squares (RLS). The RLS algorithms are known to have a fast rate of convergence which is independent of the eigenvalue spread of the input

correlation matrix. They are also very useful in applications where the environment is slowly varying. The price of all the benefits of this algorithm is a considerable increase in the computational complexity.

The objective function of this class of algorithm is given by

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i} e^2(i) = \sum_{i=0}^{k} \lambda^{k-i} [d(i) - \boldsymbol{x}^T(i)\boldsymbol{w}(k)]^2 \tag{1.4}$$

where $e(i)$ is the *a posteriori* output error at instant i and λ is the "forgetting factor".

The optimum solution in the least-squares sense is given after differentiating $\xi(k)$ with respect to $\boldsymbol{w}(k)$ and equating the result to zero. The result is given by the following product of the inverse of a matrix by a vector.

$$\boldsymbol{w}(k) = [\sum_{i=0}^{k} \lambda^{k-i} \boldsymbol{x}(i)\boldsymbol{x}^T(i)]^{-1} [\sum_{i=0}^{k} \lambda^{k-i} \boldsymbol{x}(i)d(i)] \tag{1.5}$$

The straightforward computation of the above equation results in an algorithm with a computational complexity of the order of N^3 multiplications or O$[N^3]$. Nevertheless, the computation of the inverse can be avoided by using the so-called *matrix inversion lemma* [2]. The resulting relation is used in the conventional RLS algorithm whose computational complexity is of order N^2. This computational complexity can drop to O$[N]$ when the input vector consists of delayed versions of the same signal. A number of O$[N]$ or **fast** algorithms are available in the literature including different versions of the lattice RLS (LRLS) algorithm [8] and the fast transversal RLS (FTRLS) algorithm [9] which is considered *the fastest* (in the sense that a minimal number of operations is necessary) although not stable.

1.3 LMS, NLMS and Data-Reusing Algorithms

As remarked before, the LMS algorithm uses estimates of both the input signal correlation matrix and the cross-correlation vector based on the current desired and input signals. The data-reusing LMS (DR-LMS) algorithm [3] uses current desired and input signals repeatedly within each iteration in order to improve its convergence speed. It can be easily shown that, in the limit of infinite data reuses per iteration,

the DR-LMS and the normalized LMS (NLMS) algorithms would yield the same solution [5]. With the recently proposed normalized and unnormalized new data-reusing LMS (NNDR-LMS and UNDR-LMS) algorithms [10] performance can be further improved when data from previous iterations are also used.

In [10], a graphical description of NNDR-LMS and UNDR-LMS algorithms was presented and it was shown that this new class of data-reusing algorithms has prospective better performance than the NLMS algorithm in terms of convergence speed. The graphical description also clarified why improvement is achieved when the number of reuses is increased.

For the LMS algorithm, the coefficient vector $\boldsymbol{w}$ is updated in the opposite direction of the gradient vector ($\nabla_w[\cdot]$) obtained from the instantaneous squared output error [2, 11], i.e.,

$$\boldsymbol{w}_{LMS}(k+1) = \boldsymbol{w}_{LMS}(k) - \frac{\mu}{2}\nabla_w[e^2(k)] \tag{1.6}$$

where

$$e(k) = d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}_{LMS}(k) \tag{1.7}$$

is the output error, $d(k)$ is the desired signal, $\boldsymbol{x}(k)$ is the input-signal vector containing the $N+1$ most recent input-signal samples, i.e.,

$$\boldsymbol{x}(k) = [x(k) \;\; x(k-1) \;\; \cdots \;\; x(k-N)]^T \tag{1.8}$$

and μ is the convergence factor. The coefficient-updating equation is

$$\boldsymbol{w}_{LMS}(k+1) = \boldsymbol{w}_{LMS}(k) + \mu e(k)\boldsymbol{x}(k) \tag{1.9}$$

The NLMS algorithm normalizes the step-size such that the relation

$$\boldsymbol{x}^T(k)\boldsymbol{w}_{NLMS}(k+1) = d(k) \tag{1.10}$$

is always satisfied, i.e.,

$$\boldsymbol{w}_{NLMS}(k+1) = \boldsymbol{w}_{NLMS}(k) + \frac{e(k)}{\|\boldsymbol{x}(k)\|^2 + \epsilon}\boldsymbol{x}(k) \tag{1.11}$$

where ϵ, theoretically equal to zero do satisfy (1.10), is made in practical situations a very small number used to avoid division by zero.

For the DR-LMS with L data reuses, the coefficients are updated as

$$\boldsymbol{w}_{i+1}(k) = \boldsymbol{w}_i(k) + \mu e_i(k)\boldsymbol{x}(k) \tag{1.12}$$

for $i = 0, \ldots, L$, where

$$e_i(k) = d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}_i(k) \tag{1.13}$$

$$\boldsymbol{w}_0(k) = \boldsymbol{w}_{DR-LMS}(k) \tag{1.14}$$

and

$$\boldsymbol{w}_{DR-LMS}(k+1) = \boldsymbol{w}_{L+1}(k) \tag{1.15}$$

Note that if $L = 0$ these equations correspond to the LMS algorithm.

For the NNDR-LMS algorithm with L data reuses, the coefficients are updated as

$$\boldsymbol{w}_{i+1}(k) = \boldsymbol{w}_i(k) + \frac{e_i(k)}{\|\boldsymbol{x}(k-i)\|^2 + \epsilon}\boldsymbol{x}(k-i) \tag{1.16}$$

for $i = 0, \ldots, L$, where

$$e_i(k) = d(k-i) - \boldsymbol{x}^T(k-i)\boldsymbol{w}_i(k) \tag{1.17}$$

$$\boldsymbol{w}_0(k) = \boldsymbol{w}_{NNDR-LMS}(k) \tag{1.18}$$

and

$$\boldsymbol{w}_{NNDR-LMS}(k+1) = \boldsymbol{w}_{L+1}(k) \tag{1.19}$$

Figure 1.2 illustrates geometrically the updating of the coefficient vector in a two-dimensional problem for all algorithms discussed above, starting with an arbitrary $\boldsymbol{w}(k)$. Once we are interested in comparing algorithms of similar complexity, it was considered the case of one reuse, i.e., $L = 1$. $\mathcal{S}(k)$ denotes the hyperplane which

contains all vectors $\boldsymbol{w}$ such that $\boldsymbol{x}^T(k)\boldsymbol{w} = d(k)$. In a noise-free exact-order modeling situation, $\mathcal{S}(k)$ contains the optimal coefficient vector, $\boldsymbol{w}_o$. It can be easily shown that $\boldsymbol{x}(k)$ and, consequently, $\nabla_w[e^2(k)]$ are orthogonal to the hyperplane $\mathcal{S}(k)$.

The conventional LMS algorithm takes a single step towards $\mathcal{S}(k)$ yielding the solution $\boldsymbol{w}_{LMS}(k+1)$, represented by point 2 in Figure 1.2, that is closer to $\mathcal{S}(k)$ than $\boldsymbol{w}_{LMS}(k)$. The DR-LMS algorithm iteratively approaches $\mathcal{S}(k)$ by taking successive steps in the direction given by $\boldsymbol{x}(k)$. The solution $\boldsymbol{w}_{DR-LMS}(k+1)$ is represented by points 2 and 3 in Figure 1.2. It can be shown that $\boldsymbol{w}_{DR-LMS}(k+1)$ would reach $\mathcal{S}(k)$ in the limit, as the number of data reuses approaches infinity [10]. The NLMS algorithm performs a line search in the direction of $\boldsymbol{x}(k)$ to yield in a single step the solution $\boldsymbol{w}_{NLMS}(k+1)$, represented by point 4 in Figure 1.2, which belongs to $\mathcal{S}(k)$.

The algorithms presented in [10] use more than one hyperplane, i.e., use previous data pairs $(\boldsymbol{x}(k-i), d(k-i))$, $i > 0$, in order to produce solutions $\boldsymbol{w}_{UNDR-LMS}(k+1)$ and $\boldsymbol{w}_{NNDR-LMS}(k+1)$ that are closer to $\boldsymbol{w}_o$ than the solution obtained with only the current data pair $(\boldsymbol{x}(k), d(k))$. The solutions obtained with the UNDR-LMS and the NNDR-LMS algorithms are represented by points 5 and 6 in Figure 1.2, respectively. Position 7 of Figure 1.2 corresponds to the new Binormalized Data-Reusing LMS (BNDR-LMS) algorithm which will be derived and analyzed in the next chapter.

For a noise-free exact-order modeling situation, $\boldsymbol{w}_o$ is at the intersection of $N+1$ hyperplanes constructed with linearly independent input-signal vectors. In this case, the *orthogonal-projections algorithm* [12] yields the optimal solution $\boldsymbol{w}_o$ in $N+1$ iterations. This algorithm may be viewed as a normalized data-reusing orthogonal algorithm which utilizes $N+1$ data pairs $(\boldsymbol{x}, d)$, for it performs exact line searches in (N+1) orthogonal directions constructed from present and previous data pairs.

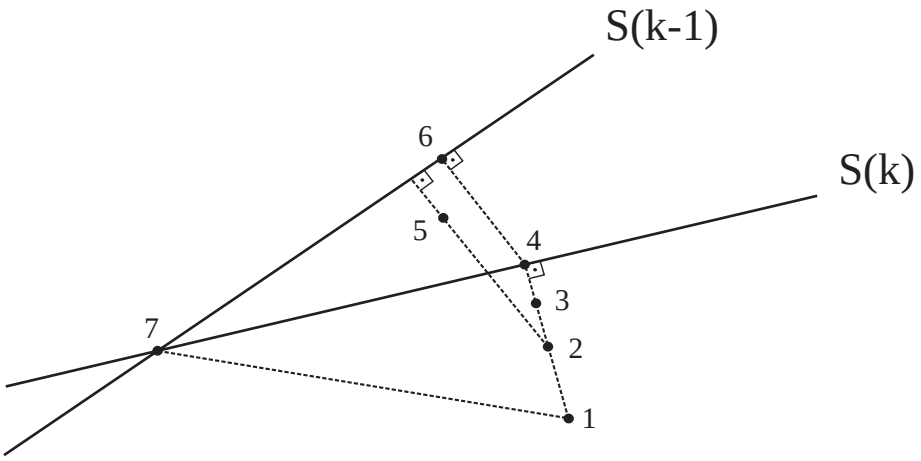

Figure 1.2: Coefficient vector update:

Position 1. $\boldsymbol{w}(k)$;
Position 2. $\boldsymbol{w}_{LMS}(k+1)$ and first step towards $\boldsymbol{w}_{DR-LMS}(k+1)$ and $\boldsymbol{w}_{UNDR-LMS}(k+1)$;
Position 3. $\boldsymbol{w}_{DR-LMS}(k+1)$;
Position 4. $\boldsymbol{w}_{NLMS}(k+1)$ and first step towards $\boldsymbol{w}_{NNDR-LMS}(k+1)$;
Position 5. $\boldsymbol{w}_{UNDR-LMS}(k+1)$;
Position 6. $\boldsymbol{w}_{NNDR-LMS}(k+1)$;
Position 7. $\boldsymbol{w}_{BNDR-LMS}(k+1)$.

1.4 Introducing the QR Decomposition

Another alternative method for the implementation of the recursive least-squares (RLS) is the use of QR decomposition. In (1.5), the matrix which needs to be inverted is usually called deterministic data correlation matrix $\boldsymbol{R}_D = \boldsymbol{X}^T(k)\boldsymbol{X}(k)$ where $\boldsymbol{X}(k)$ is the input data matrix as will be defined later. The basic idea of this QR family of algorithms is the triangularization of the input data matrix through the use of QR decomposition techniques.

It is worth mentioning that the matrix $\boldsymbol{X}(k)$ is $(k+1) \times (N+1)$ which means that it increases its order as the iterations progress. The QR-decomposition process makes use of an orthonormal matrix $\boldsymbol{Q}(k)$ of order $(k+1) \times (k+1)$ in order to

reduce $\boldsymbol{X}(k)$ to a triangular matrix $\boldsymbol{U}(k)$ of order $(N+1)\times(N+1)$ such that

$$\boldsymbol{Q}(k)\boldsymbol{X}(k) = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{U}(k) \end{bmatrix} \tag{1.20}$$

where $\boldsymbol{O}$ is a matrix of order $(k-N)\times(N+1)$ with all elements null.

Matrix $\boldsymbol{Q}(k)$ represents the overall triangularization process and may be implemented in different manners. In this thesis, the numerically well conditioned Givens rotations will be used although other techniques, such as the Householder transformation [13, 14], are available. The main advantages obtained with the QR-decomposition RLS (QRD-RLS) algorithms are the possibility of implementation in systolic arrays and its improved numerical behavior in limited precision environment. The conventional QR-RLS algorithm has a computational requirement of the order of N^2 multiplications or O[N^2]. A number of alternative algorithms such as the inverse QR (IQR) algorithm [15], the so called fast[1] QR (FQR) algorithms [16]–[20] and the fast QR-Lattice (FQR-L) algorithms [21, 22] are also available in the technical literature.

The RLS-based algorithms are summarized in Figure 1.3. These algorithms are used whenever fast convergence is necessary for input signals with a high eigenvalue spread and the increase in the computational load is tolerable. Except for the FTRLS algorithm, which basic version is unstable, all other fast RLS algorithms do not have the coefficient vector of the direct form realization available in every iteration.

Our goal here is the study of the fast recursive least-squares algorithms based on QR decomposition which are among those adaptive algorithms with both the desired characteristics of computational complexity of O[N] and the numerical robustness associated with the use of Givens rotations. Although multichannel and complex versions of most adaptive algorithms exist, this thesis will be concerned only with the single channel with real input case.

[1] "Fast" here means of O[N]

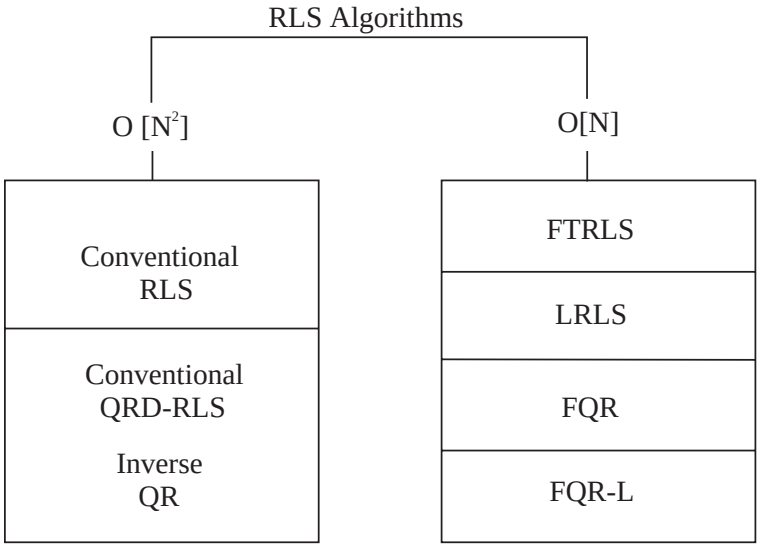

Figure 1.3: The RLS algorithms.

1.5 Original Contributions

Many of the algorithms presented and discussed in the previous sections are still subject of research where simplicity, convergence speed, stability, and robustness are the topics of main interest.

In the next chapter, the new BNDR-LMS algorithm will be described. This algorithm combines data reusing, orthogonal projections of two consecutive gradient directions, and normalization in order to achieve faster convergence when compared to other LMS-like algorithms. On the other hand, the new algorithm is simpler with respect to computational complexity and more robust than the orthogonal-projection algorithm. At each iteration, the BNDR-LMS algorithm yields the solution $\boldsymbol{w}(k+1)$ which is at the intersection of hyperplanes $\mathcal{S}(k)$ and $\mathcal{S}(k-1)$ and at a minimum distance from $\boldsymbol{w}(k)$ (see 7 in Figure 1.2). The algorithm can also be viewed as a simplified version of the orthogonal-projection algorithm which utilizes just two consecutive directions. In Chapter 2, it is also addressed the analyses of the first and second moments of the coefficient vector, respectively, and simulation results.

In Chapter 3 an optimal step-size sequence is proposed and an application of

this algorithm is the field of mobile communications is carried out.

The development of the new fast QR algorithm is done in Chapter 4 where a unified approach is used to classify the members of the fast QR family of algorithms.

In Chapter 5, the lattice version of two of those fast QR algorithms are presented and fully described according to the notation used in this work.

Chapter 6 addresses the analysis in a limited precision environment of the fast QR algorithms using backward prediction errors updating.

The conclusions of the thesis as well as suggestions for further research are summarized in Chapter 7.

Chapter 2

The BNDR-LMS Algorithm

The new binormalized data-reusing LMS (BNDR-LMS) algorithm [23] described in this chapter and briefly introduced in [24] and [25] employs normalization on two orthogonal directions obtained from consecutive data pairs within each iteration. In all simulations carried out with colored input signals, this algorithm presented faster convergence than all other data-reusing algorithms for the case of two data pairs, or, equivalently, one data reuse.

A thorough analysis of convergence in the mean and mean-square of the coefficient vector is provided and the stability limits for the convergence factor as well as closed-form formulas for mean-square error (MSE) after convergence are obtained from the analysis. The inadequacy of the independence assumption [26] for analyses of data-reusing algorithms [10] is overcome by adopting a simplified model for the input-signal vector which is consistent with the first two moments and renders a tractable analysis [4, 27]. This analysis can be readily extended to other data-reusing algorithms (e.g., NNDR-LMS and UNDR-LMS algorithms [10]).

2.1 Problem Statement and Algorithm Derivation

In order to state the problem, we note that the solution which belongs to $\mathcal{S}(k)$ and $\mathcal{S}(k-1)$ at a minimum distance from $\boldsymbol{w}(k)$ is the one that solves

$$\min_{\boldsymbol{w}(k+1)} \|\boldsymbol{w}(k+1) - \boldsymbol{w}(k)\|^2 \tag{2.1}$$

subject to

$$\boldsymbol{x}^T(k)\boldsymbol{w}(k+1) = d(k) \tag{2.2}$$

and

$$\boldsymbol{x}^T(k-1)\boldsymbol{w}(k+1) = d(k-1) \tag{2.3}$$

The function to be minimized is, therefore,

$$\begin{aligned} f[\boldsymbol{w}(k+1)] = \|\boldsymbol{w}(k+1) - \boldsymbol{w}(k)\|^2 &+ \lambda_1[d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k+1)] \\ &+ \lambda_2[d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}(k+1)] \end{aligned} \tag{2.4}$$

which, for linearly independent input-signal vectors $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$, has the unique solution

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \frac{\lambda_1}{2}\boldsymbol{x}(k) + \frac{\lambda_2}{2}\boldsymbol{x}(k-1) \tag{2.5}$$

where

$$\frac{\lambda_1}{2} = \frac{[d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)]\|\boldsymbol{x}(k-1)\|^2 - [d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}(k)]\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \tag{2.6}$$

and

$$\frac{\lambda_2}{2} = \frac{[d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}(k)]\|\boldsymbol{x}(k)\|^2 - [d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)]\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \tag{2.7}$$

2.1.1 Simplified Version

The derivation presented above is valid for any $\boldsymbol{w}(k)$, which may or may not belong to $\mathcal{S}(k-1)$. However, if successive optimized steps are taken for $\boldsymbol{w}(k)$ for all k, then

$$\boldsymbol{x}^T(k-1)\boldsymbol{w}(k) = d(k-1) \tag{2.8}$$

and a simplified set of updating equations for the algorithm results:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \frac{\lambda_1'}{2}\boldsymbol{x}(k) + \frac{\lambda_2'}{2}\boldsymbol{x}(k-1) \tag{2.9}$$

where

$$\frac{\lambda_1'}{2} = \frac{[d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)]\|\boldsymbol{x}(k-1)\|^2}{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \tag{2.10}$$

and

$$\frac{\lambda_2'}{2} = \frac{-[d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)]\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \tag{2.11}$$

2.1.2 Geometrical Derivation

The BNDR-LMS algorithm can be alternatively derived from a purely geometrical reasoning. The first step is to reach a preliminary solution, $\boldsymbol{w}_1(k)$, which belongs to $\mathcal{S}(k)$ and is at a minimum distance from $\boldsymbol{w}(k)$, represented by point 4 in Figure 1.2. This is achieved by the NLMS algorithm starting from $\boldsymbol{w}(k)$, i.e.,

$$\boldsymbol{w}_1(k) = \boldsymbol{w}(k) + \frac{e(k)}{\|\boldsymbol{x}(k)\|^2}\boldsymbol{x}(k) \tag{2.12}$$

In the second step, $\boldsymbol{w}_1(k)$ is updated in a direction orthogonal to the previous one, therefore belonging to $\mathcal{S}(k)$, until the intersection with $\mathcal{S}(k-1)$ is reached. This is achieved by the NLMS algorithm starting from $\boldsymbol{w}_1(k)$ and following the direction $\boldsymbol{x}_1^{\perp}(k)$ which is the projection of $\boldsymbol{x}(k-1)$ onto $\mathcal{S}(k)$, i.e.,

$$\boldsymbol{w}(k+1) = \boldsymbol{w}_1(k) + \frac{e_1(k)}{\|\boldsymbol{x}_1^{\perp T}(k)\|^2}\boldsymbol{x}_1^{\perp}(k) \tag{2.13}$$

where

$$\boldsymbol{x}_1^{\perp}(k) = \left[\boldsymbol{I} - \frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{\|\boldsymbol{x}(k)\|^2}\right]\boldsymbol{x}(k-1) \tag{2.14}$$

and

$$e_1(k) = d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}_1(k) \tag{2.15}$$

The use of $\boldsymbol{x}_1^{\perp}(k)$ obtained from $\boldsymbol{x}(k-1)$ assures that the minimum-distance path is chosen. Note that the requirement of linear independence of consecutive input-signal vectors $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$, necessary to ensure existence of the solution, is also manifested here.

As will be shown in the analysis (c.f. Section 2.3) and will be verified by simulations (c.f. Table 2.2), the excess of the mean-square error (MSE) for the BNDR-LMS algorithm as in (2.5)–(2.7) or in (2.9)–(2.11) is close to the variance of the observation noise when there is no modeling error. Such performance is expected from normalized algorithms. Therefore, in order to control this excess of MSE a step-size μ may be introduced. Although maximum convergence rate is usually obtained with $\mu = 1$, the use of a smaller value for the step-size may be required in applications where measurement error is too high. In this case, we must emphasize that the solution $\boldsymbol{w}(k+1)$ obtained at each iteration is not at the intersection of hyperplanes $\mathcal{S}(k-1)$ and $\mathcal{S}(k)$ and, therefore, the simplified version of the algorithm given by (2.9)–(2.11) should not be used.

If $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$ are linearly dependent, then $\mathcal{S}(k)$ is parallel to $\mathcal{S}(k-1)$, $\boldsymbol{x}_1^{\perp}(k)$ is the null vector and $\boldsymbol{w}(k+1) = \boldsymbol{w}_1(k)$, which corresponds to the NLMS algorithm for any value of step-size. Particularly when $\mu = 1$, it is also correct to say that $\boldsymbol{w}(k)$ is already on the hyperplane $\mathcal{S}(k-1)$.

The BNDR-LMS algorithm is summarized in Table 2.1.

2.2 Convergence Analysis of the Coefficient Vector

In this section, we assume that an unknown FIR filter with coefficient vector given by $\boldsymbol{w}_o$ is to be identified by an adaptive filter of same order employing the BNDR-LMS

Table 2.1: The Binormalized Data-Reusing LMS algorithm

BNDR-LMS

ϵ = small positive value
for each k
{
  $\alpha = \boldsymbol{x}^T(k)\boldsymbol{x}(k-1)$
  $\rho(k) = \boldsymbol{x}^T(k)\boldsymbol{x}(k)$
  $y_1 = \boldsymbol{x}^T(k)\boldsymbol{w}(k)$
  $e_1 = d(k) - y_1$
  $den = \rho(k)\rho(k-1) - \alpha^2$
  if $den < \epsilon$
  {
    $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu e_1 \boldsymbol{x}(k)/\rho(k)$
  }
  else
  {
    $y_2 = \boldsymbol{x}^T(k-1)\boldsymbol{w}(k)$
    $e_2 = d(k-1) - y_2$
    $\frac{\lambda_1}{2} = (e_1\rho(k-1) - e_2\alpha)/den$
    $\frac{\lambda_2}{2} = (e_2\rho(k) - e_1\alpha)/den$
    $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu[\frac{\lambda_1}{2}\boldsymbol{x}(k) + \frac{\lambda_2}{2}\boldsymbol{x}(k-1)]$
  }
}

algorithm, i.e., $d(k)$ can be modeled as

$$d(k) = \boldsymbol{x}^T(k)\boldsymbol{w}_o + n(k) \tag{2.16}$$

where $n(k)$ is measurement noise. It is also assumed that input signal and measurement noise are taken from independent and identically distributed zero-mean white noise processes with variances σ_x^2 and σ_n^2, respectively.

We are interested in analyzing the convergence behavior of the coefficient vector in terms of a step-size μ. Let

$$\Delta\boldsymbol{w}(k) = \boldsymbol{w}(k) - \boldsymbol{w}_o \tag{2.17}$$

be the error in the adaptive filter coefficients as related to the ideal coefficient vector.

For the BNDR-LMS algorithm as described in (2.5)–(2.7), $\Delta \boldsymbol{w}(k+1)$ is given by

$$\Delta \boldsymbol{w}(k+1) = \Delta \boldsymbol{w}(k) + \mu \left[\frac{\lambda_1}{2} \boldsymbol{x}(k) + \frac{\lambda_2}{2} \boldsymbol{x}(k-1) \right] \tag{2.18}$$

From (2.16) and (2.5)–(2.7), we have

$$\Delta \boldsymbol{w}(k+1) = [\boldsymbol{I} + \mu \boldsymbol{A}] \Delta \boldsymbol{w}(k) + \mu \boldsymbol{b} \tag{2.19}$$

where

$$\begin{aligned} \boldsymbol{A} = & \frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1) + \boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \\ & - \frac{\|\boldsymbol{x}(k-1)\|^2 \boldsymbol{x}(k)\boldsymbol{x}^T(k) + \|\boldsymbol{x}(k)\|^2 \boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \end{aligned} \tag{2.20}$$

and

$$\begin{aligned} \boldsymbol{b} = & \frac{n(k)\|\boldsymbol{x}(k-1)\|^2 - n(k-1)\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \boldsymbol{x}(k) \\ & + \frac{n(k-1)\|\boldsymbol{x}(k)\|^2 - n(k)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \boldsymbol{x}(k-1) \end{aligned} \tag{2.21}$$

By taking the expected value on both sides of (2.19), for $n(k)$ and $x(k)$ samples from independent zero-mean random processes, we have

$$E[\boldsymbol{b}] = 0 \tag{2.22}$$

and

$$\begin{aligned} E[\Delta \boldsymbol{w}(k+1)] &= E[(\boldsymbol{I} + \mu \boldsymbol{A}) \Delta \boldsymbol{w}(k)] \\ &= E\left(\left\{ \boldsymbol{I} + \mu \left[\frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \right.\right. \right. \\ &\quad + \frac{\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k) - \|\boldsymbol{x}(k-1)\|^2 \boldsymbol{x}(k)\boldsymbol{x}^T(k)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \\ &\quad \left.\left. \left. - \frac{\|\boldsymbol{x}(k)\|^2 \boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{\|\boldsymbol{x}(k)\|^2 \|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2} \right] \right\} \Delta \boldsymbol{w}(k) \right) \end{aligned} \tag{2.23}$$

Expression (2.23) can be further simplified if the following assumptions are made:

1. $\Delta \boldsymbol{w}(k)$ is statistically independent of $\boldsymbol{x}(k)\boldsymbol{x}^T(k)$ (independence assumption [26]),

2. $E[num/den] \approx E[num]/E[den]$, where *num* and *den* are the elements in the numerator and denominator of (2.23), respectively, which implies independence between *num* and *den* as well as a first-order approximation[1] in the evaluation of $E[1/den]$.

Moreover, the following relations can be easily verified when the elements of $\boldsymbol{x}(k)$ are samples of a white Gaussian process (see Appendix A):

1.

$$E\{[\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2\} = (N+1)(\sigma_x^2)^2 \tag{2.24}$$

2.

$$E\left\{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - [\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)]^2\right\} = N(N+3)(\sigma_x^2)^2 \tag{2.25}$$

3.

$$\{E[\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)]\}_{ij} = \begin{cases} (\sigma_x^2)^2, & i=j \quad \text{or} \quad i=j-2 \\ 0, & \text{otherwise} \end{cases} \tag{2.26}$$

for $[\cdot]_{ij}$ the (i,j) element of matrix $[\cdot]$.

4.

$$E[\|\boldsymbol{x}(k-1)\|^2\boldsymbol{x}(k)\boldsymbol{x}^T(k)] = (N+3)(\sigma_x^2)^2\boldsymbol{I} \tag{2.27}$$

5.

$$\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k) = (1-\mu)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k-1) + \mu n(k-1) \tag{2.28}$$

[1]For a more in-depth discussion on this approximation, see [4, 6].

Based on these assumptions and relations, (2.23) can be rewritten as

$$\begin{aligned} E[\Delta \boldsymbol{w}(k+1)] \approx E\Bigg(\Bigg\{\boldsymbol{I} + \mu\Bigg[&\frac{\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{N(N+3)(\sigma_x^2)^2} \\ &-\frac{\|\boldsymbol{x}(k-1)\|^2\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{N(N+3)(\sigma_x^2)^2}\Bigg]\Bigg\}\Delta\boldsymbol{w}(k) \\ &+\mu(1-\mu)\Bigg[\frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{N(N+3)(\sigma_x^2)^2} \\ &-\frac{\|\boldsymbol{x}(k)\|^2\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{N(N+3)(\sigma_x^2)^2}\Bigg]\Delta\boldsymbol{w}(k-1)\Bigg) \\ \approx &\left(1-\frac{\mu}{N}\right)E\left[\Delta\boldsymbol{w}(k)\right] - \frac{\mu(1-\mu)}{N}E\left[\Delta\boldsymbol{w}(k-1)\right] \end{aligned} \tag{2.29}$$

The last relation of (2.29) was obtained by considering $\|\boldsymbol{x}(k-1)\|^2$ statistically independent of $\Delta\boldsymbol{w}(k)$ and by making a first order approximation in the calculation of the numerators with the help of relations (2.26) to (2.28). From (2.29), it is clear that convergence in the mean of the BNDR-LMS algorithm to an unbiased solution is guaranteed for values of step size μ such that all elements of $E[\Delta\boldsymbol{w}(k+1)]$ in (2.29) go to zero as $k \to \infty$. This is achieved if the poles of the second-order difference equation are strictly inside the unit circle, i.e.,

$$|z_{1,2}| = \left|\frac{1-\frac{\mu}{N} \pm \sqrt{\left(1-\frac{\mu}{N}\right)^2 - \frac{4\mu(1-\mu)}{N}}}{2}\right| < 1 \tag{2.30}$$

which is always true for $N > 1$ and μ satisfying

$$0 < \mu < 2 \tag{2.31}$$

2.3 Second-Order Statistic Analysis

2.3.1 White Input Signal

Although $\Delta\boldsymbol{w}(k)$ converges in average to zero as k goes to infinite, which characterizes unbiasedness of the estimate, consistency of coefficient estimates, which in other words means negligible instantaneous errors on these coefficients, is only achieved in cases of very small ξ_{min} or values of μ close to zero. In general, an excess of MSE,

which depends on the second-order statistics of vector $\Delta\boldsymbol{w}(k)$, will be present. The excess of MSE is defined as [2, 1]

$$\xi_{exc} = \lim_{k\to\infty} \xi(k) - \xi_{min} \tag{2.32}$$

where $\xi(k) = E[e^2(k)]$ and ξ_{min} is the minimum mean-squared error due to nonexact-modeling or presence of additive noise, or both [2].

The difference $\Delta\xi(k) = \xi(k) - \xi_{min}$ is known as excess in the MSE [2] and can be expressed as

$$\begin{aligned}\Delta\xi(k) &= E\{[n(k) - \Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)]^2\} - \xi_{min} \\ &= E[\Delta\boldsymbol{w}^T(k)\boldsymbol{R}\Delta\boldsymbol{w}(k)] \\ &= \mathrm{tr}\{\boldsymbol{R}\,\mathrm{cov}[\Delta\boldsymbol{w}(k)]\}\end{aligned} \tag{2.33}$$

It is necessary, therefore, to derive an expression for the coefficient-error-vector covariance matrix $\mathrm{cov}[\Delta\boldsymbol{w}(k+1)]$. From (2.19),

$$\begin{aligned}\mathrm{cov}[\Delta\boldsymbol{w}(k+1)] &= E\left[\Delta\boldsymbol{w}(k+1)\Delta\boldsymbol{w}^T(k+1)\right] \\ &= E\left\{[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)[\boldsymbol{I}+\mu\boldsymbol{A}]\right\} \\ &\quad + E\left\{\mu[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\boldsymbol{b}^T\right\} + E\left\{\mu\boldsymbol{b}\Delta\boldsymbol{w}^T(k)[\boldsymbol{I}+\mu\boldsymbol{A}]\right\} \\ &\quad + E\left[\mu^2\boldsymbol{b}\boldsymbol{b}^T\right]\end{aligned} \tag{2.34}$$

Recalling (2.20) and (2.21), we can foresee the enormous complexity to evaluate (2.34) even with a number of assumptions. An interesting alternative is the use of a simplified model for the input-signal vector $\boldsymbol{x}(k)$ which can be consistent with the first- and second-order statistics of a general input signal, but has a reduced and countable number of possible directions of excitation. This model was introduced in [28] and was successfully employed in [4] and [27]. The input-signal vector for the model is

$$\boldsymbol{x}(k) = s_k r_k \boldsymbol{V}_k \tag{2.35}$$

where:

- s_k is ± 1 with probability of occurrence 1/2;
- r_k^2 has the same probability distribution function of $\|\boldsymbol{x}(k)\|^2$, or, for the case of interest, is a sample of an independent process with χ-square distribution of $(N+1)$ degrees of freedom, $E[r_k^2] = (N+1)\sigma_x^2$;
- $\boldsymbol{V}_k$ is equal to one of the $N+1$ orthonormal eigenvectors of $\boldsymbol{R}$, denoted $\mathcal{V}_i$, $i = 1, \ldots, N+1$. We will also assume that for a white Gaussian input signal $\boldsymbol{V}_k$ is uniformly distributed and, consequently, if $P(\cdot)$ denotes the probability of occurrence of event $(\cdot)$, then

$$P(\boldsymbol{V}_k = \mathcal{V}_i) = \frac{1}{N+1} \tag{2.36}$$

For the given input-signal model, we may express $\Delta\xi(k+1)$ as

$$\begin{aligned}\Delta\xi(k+1) = {} & \Delta\xi(k+1)\,|_{\boldsymbol{x}(k)\|\boldsymbol{x}(k-1)} \times P[\boldsymbol{x}(k) \parallel \boldsymbol{x}(k-1)] \\ & + \Delta\xi(k+1)\,|_{\boldsymbol{x}(k)\perp\boldsymbol{x}(k-1)} \times P[\boldsymbol{x}(k) \perp \boldsymbol{x}(k-1)]\end{aligned} \tag{2.37}$$

Conditions $\boldsymbol{x}(k) \parallel \boldsymbol{x}(k-1)$ and $\boldsymbol{x}(k) \perp \boldsymbol{x}(k-1)$ in the adopted model are equivalent to $\boldsymbol{V}_k = \boldsymbol{V}_{k-1}$ and $\boldsymbol{V}_k \neq \boldsymbol{V}_{k-1}$, respectively, such that $\boldsymbol{V}_k$ and $\boldsymbol{V}_{k-1}$ can only be parallel or orthogonal to each other.

As remarked before, the BNDR-LMS algorithm behaves exactly like the NLMS algorithm when the input signal vector at instants k and $k-1$ are parallel. In this case, the excess of MSE is given by [4]

$$\Delta\xi(k+1)_{\|} = \left[1 + \frac{\mu(\mu-2)}{N+1}\right]\Delta\xi(k) + \frac{\mu^2}{(N+2-\nu_x)}\sigma_n^2 \tag{2.38}$$

where $\nu_x = E[x^4(k)/\sigma_x^4]$ is known as the *kurtosis* of the input signal, which varies from 1 for a binary distribution to 3 for a Gaussian distribution to ∞ for a Cauchy distribution [4, 29]. It must be stressed, however, that (2.38) holds only for $\nu_x \ll N+1$ [4].

For the case where $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$ are always orthogonal, from (2.33) and (2.34) we have, for $\boldsymbol{R} = \sigma_x^2\boldsymbol{I}$, i.e., white-noise input signals (see Appendix B),

$$\begin{aligned}\Delta\xi(k+1)_{\perp} = {} & \left[1 + \frac{\mu(\mu-2)}{N+1}\right]\Delta\xi(k) + \frac{\mu(1-\mu)^2(\mu-2)}{N+1}\Delta\xi(k-1) \\ & + \frac{\mu^2(\mu-2)^2}{N+2-\nu_x}\sigma_n^2\end{aligned} \tag{2.39}$$

A final expression for the excess in the MSE may now be obtained from (2.38) and (2.39) combined and weighted accordingly, as suggested in (2.37). For a white input signal, the probabilities of $\boldsymbol{V}_k = \boldsymbol{V}_{k-1}$ and $\boldsymbol{V}_k \neq \boldsymbol{V}_{k-1}$ are equal to $\frac{1}{N+1}$ and $\frac{N}{N+1}$, respectively. The excess in the MSE is, therefore, given by

$$\begin{aligned}\Delta\xi(k+1) = &\left[1 + \frac{\mu(\mu-2)}{N+1}\right]\Delta\xi(k) + \frac{N\mu(1-\mu)^2(\mu-2)}{(N+1)^2}\Delta\xi(k-1) \\ &+ \frac{\mu^2\left[1+N(\mu-2)^2\right]}{(N+1)(N+2-\nu_x)}\sigma_n^2\end{aligned} \tag{2.40}$$

Experiment 1: In order to confront the behavior of the BNDR-LMS algorithm with (2.38) for different values of μ a simple experiment was carried out where input-signal vectors at consecutive time instants are always parallel. A setup was constructed where a $10th$-order unknown plant is to be identified by a $10th$-order adaptive filter when the input-signal vector is $\boldsymbol{x}(k) = s_k\boldsymbol{V}$ with $\boldsymbol{V}$ a constant vector equal to $[1\ 0\ \cdots\ 0]^T$. In this case, the kurtosis ν_x is 1 and the steady-state value of $\Delta\xi(k)$, ξ_{exc}, is

$$\xi_{exc} = \frac{\mu\sigma_n^2}{2-\mu} \tag{2.41}$$

The results, depicted in Figure 2.1, show a comparison between simulations averaged after 50 runs and the theoretical values predicted by (2.41). From the analysis of this experiment, it becomes clear that for the special case when the BNDR-LMS algorithm behaves like the NLMS algorithm, (2.41) is in excellent agreement with simulation results.

Experiment 2: A second experiment was carried out where input-signal vectors at consecutive time instants are always orthogonal. The setup was similar to that of Experiment 1, except that the input signal vector was chosen as $\boldsymbol{x}(k) = s_k\boldsymbol{V}_k$ with $\boldsymbol{V}_k$ always different from $\boldsymbol{V}_{k-1}$ and equal to one of the vectors forming the canonic basis. The kurtosis of the input signal is also equal to 1. Theoretical results and the results from a 50-trial simulation are depicted in Figure 2.2. The results show that the expression in (2.39) is accurate for the assumptions made.

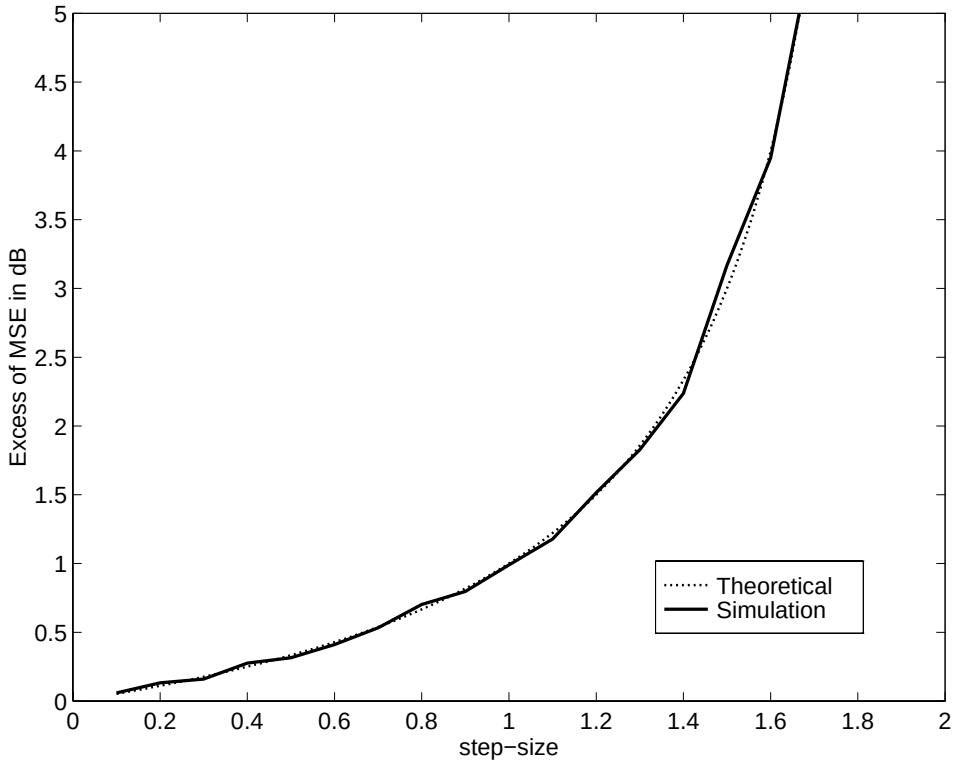

Figure 2.1: Excess of MSE for parallel input signal vectors.

Experiment 3: A third experiment was carried out where the input-signal vectors were randomly chosen among the canonic basis, such that $\boldsymbol{V}_k$ and $\boldsymbol{V}_{k-1}$ could be parallel or orthogonal with probabilities $\frac{1}{N+1}$ and $\frac{N}{N+1}$, respectively. The results from a 50-trial simulation and those from (2.40) are depicted in Figure 2.3 which shows accuracy of the analysis at least for the input-signal model used.

2.3.2 Colored Input Signal

Using the input-signal-vector model given in (2.35), we may now extend the analysis to colored input signals. The angular distribution of $\boldsymbol{x}(k)$ need be changed in order to incorporate different probabilities for the directions given by the $(N+1)$ eigenvectors of $\boldsymbol{R}$. In other words, (2.37)–(2.39) are maintained and only probabilities $P\left[\boldsymbol{x}(k) \parallel \boldsymbol{x}(k-1)\right]$ and $P\left[\boldsymbol{x}(k) \perp \boldsymbol{x}(k-1)\right]$ need be recalculated. Each eigenvector of $\boldsymbol{R}$, denoted as $\mathcal{V}_i,\ i = 1,\ \ldots,\ N+1$ will now have the following probability of

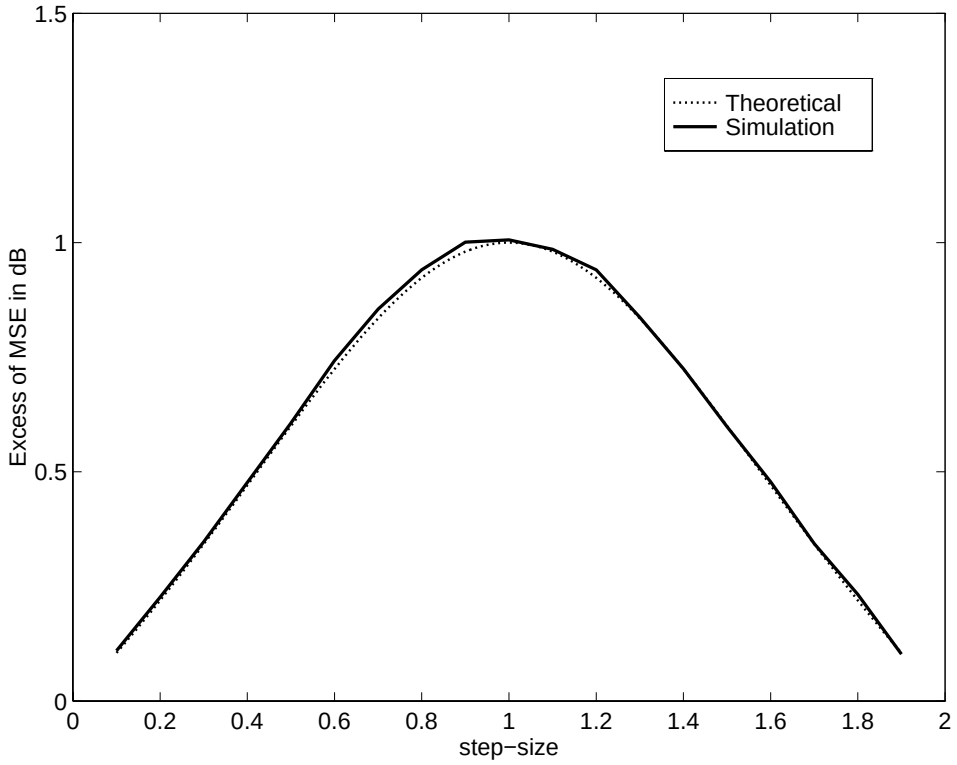

Figure 2.2: Excess of MSE for orthogonal input signal vectors.

occurrence [4]

$$P(\boldsymbol{V}_k = \mathcal{V}_i) = \frac{\lambda_i}{tr(R)} \tag{2.42}$$

where λ_i is the eigenvalue associated to the eigenvector $\mathcal{V}_i$. For an easy association between $P[\boldsymbol{x}(k) \parallel \boldsymbol{x}(k-1)]$ and input-signal correlation, let us suppose the input signal $x(k)$ is correlated by an allpole filter as in

$$x(k) = \gamma x(k-1) + (1-\gamma)\eta(k), \quad 0 \leq \gamma \leq 1 \tag{2.43}$$

where $\eta(k)$ is a sample from an independent zero-mean process with variance given by σ_η^2. The autocorrelation matrix for this input signal can be easily derived and is

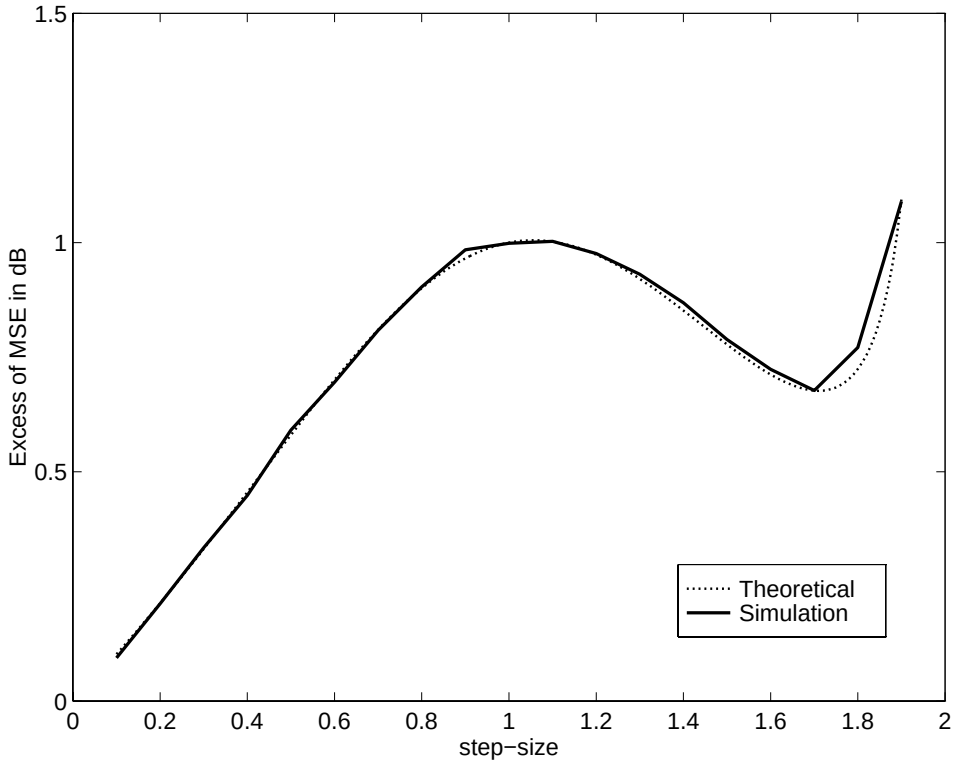

Figure 2.3: Excess of MSE for a modeled input signal vector.

expressed as

$$\boldsymbol{R} = \frac{1-\gamma}{1+\gamma}\sigma_{\eta}^{2}\begin{bmatrix} 1 & \gamma & \gamma^{2} & \cdots & \gamma^{N} \\ \gamma & 1 & \gamma & \cdots & \gamma^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma^{N} & \gamma^{N-1} & \gamma^{N-2} & \cdots & 1 \end{bmatrix} \tag{2.44}$$

From (2.44) we have all necessary eigenvalues and eigenvectors such that we can compute

$$\begin{aligned} P\left[\boldsymbol{x}(k) \parallel \boldsymbol{x}(k-1)\right] &= P\left[\boldsymbol{V}_{k} \parallel \boldsymbol{V}_{k-1}\right] \\ &= P\left[\boldsymbol{V}_{k} \parallel \boldsymbol{V}_{k-1} \mid_{\boldsymbol{V}_{k-1}=\boldsymbol{V}_{1}}\right] \times P\left[\boldsymbol{V}_{k-1} = \boldsymbol{V}_{1}\right] + \cdots \\ &\quad + P\left[\boldsymbol{V}_{k} \parallel \boldsymbol{V}_{k-1} \mid_{\boldsymbol{V}_{k-1}=\boldsymbol{V}_{N+1}}\right] \times P\left[\boldsymbol{V}_{k-1} = \boldsymbol{V}_{N+1}\right] \\ &= \sum_{i=1}^{N+1}\left(\frac{\lambda_{i}}{tr(R)}\right)^{2} \end{aligned} \tag{2.45}$$

and

$$P\left[\boldsymbol{x}(k) \perp \boldsymbol{x}(k-1)\right] = 1 - P\left[\boldsymbol{x}(k) \parallel \boldsymbol{x}(k-1)\right] \tag{2.46}$$

Equations (2.45) and (2.46) are in accordance with the white-input situation, for this case corresponds to $\gamma = 0$ and all eigenvalues will be equal to σ_x^2 such that $P(\boldsymbol{V}_k = \mathcal{V}_i) = \frac{1}{N+1}$ as already described. When the input signal is correlated through a first-order allpole filter and modeled with (2.35) and (2.42), the excess of MSE is given by (2.37)–(2.39) with probabilities given by (2.45) and (2.46). Although (2.38)–(2.39) have been obtained based on a white Gaussian model for the input signal, simulations have shown that our reasoning is valid when the input signal is generated according to (2.35) with probabilities given by (2.42) and λ_i obtained from (2.44). Moreover, for $\mu = 1$ and a modeled input signal where only parallel or perpendicular vectors may occur the BNDR-LMS algorithm degrades to the NLMS algorithm and the steady-state MSE becomes independent of the radial distribution of $\boldsymbol{x}(k)$ [4]. This is perfectly described by (2.38)–(2.39), supporting the validity of our reasoning.

2.4 Simulation Results

In order to test the BNDR-LMS algorithm for more practical situations, simulations were carried out for several system identification problems with input signals not constrained to fit a discrete angular distribution function as in the experiments of the previous section. Initially, the system order was $N = 10$, the input signal was a colored noise with a conditioning number around 187, and the input signal-to-observation-noise ratio (SNR) was set to 60dB and 150dB. The learning curves (MSE in dB) for the NLMS, the NNDR-LMS (one reuse) and the BNDR-LMS algorithms are depicted in Figure 2.4, corresponding to an average of 200 realizations. In this first experiment, the step-size μ was set to 1 in order to achieve the fastest convergence rate of the BNDR-LMS algorithm. The choice of one reuse for the NNDR-LMS algorithm rendered similar computation complexities for the algorithms tested.

In this example we can clearly verify the superior performance of the BNDR-LMS algorithm in terms of speed of convergence when compared to the NLMS and the NNDR-LMS algorithms (with one single reuse) for the case of a high signal-to-noise ratio. This benefit becomes more evident in cases where the input signal is even more correlated. Simulations for the conventional LMS algorithm and for the DR-LMS algorithm were also carried out for the same setup, but their performances were, as expected, inferior to that of the NLMS algorithm and the results were omitted from Figure 2.4. Concerning this example, it is worth mentioning that for the NNDR-LMS algorithm, a similar performance would be obtained if we had used at least four reuses ($L = 4$) with the same two pairs of data (or about two reuses if we increased the information with one extra pair of data). It means that more than double of the computational effort of the BNDR-LMS algorithm would be necessary for the NNDR-LMS algorithm, in the case of the same amount of memory, to have a similar convergence rate.

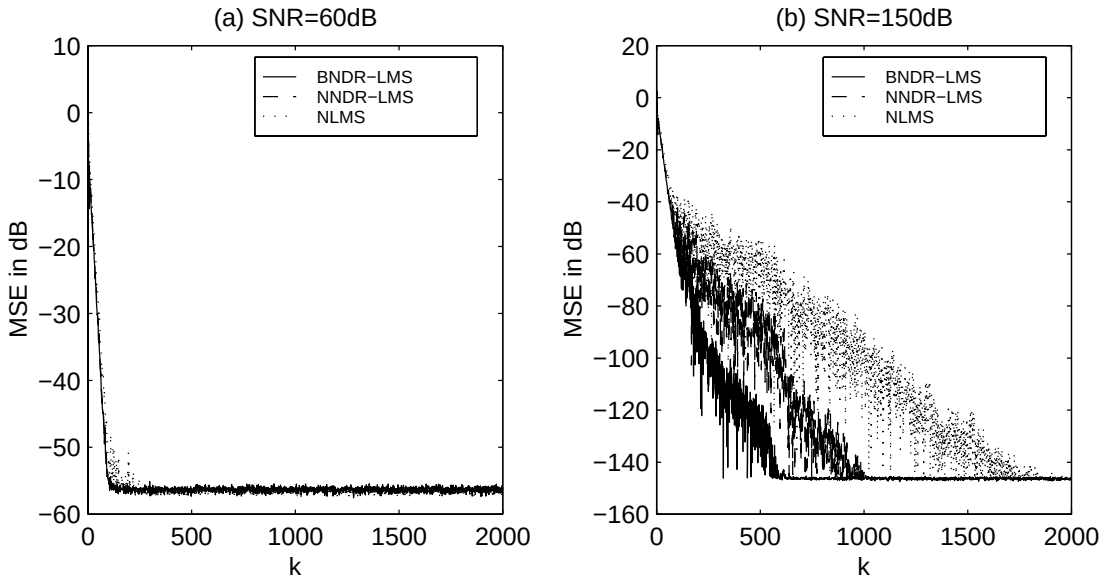

Figure 2.4: MSE for the NLMS, the NNDR-LMS, and the BNDR-LMS algorithms.

In a second experiment, yet with $\mu = 1$ and $N = 10$, the excess of MSE (ξ_{exc} in dB) was measured in order to test the behavior of the BNDR-LMS algorithm in terms of mean-squared error after convergence. ξ_{min}, in this case the variance of the

measurement noise, was set to 10^{-6} while the input signal was made a zero-mean white Gaussian noise process. The results are summarized in Table 2.2 where we can also observe the excess of MSE in dB for a nonstationary environment. In this case, the observation noise was set to zero and the system (plant) coefficients were varied according to a generalized random-walk model, $\boldsymbol{w}(k) = \boldsymbol{w}(k-1) + \boldsymbol{v}$, where $\boldsymbol{v}$ was a random vector with elements of zero-mean and variance equal to 10^{-6}. As we can see from Table 2.2, in both stationary and nonstationary environments, the BNDR-LMS algorithm performed closely to the NLMS and the NNDR-LMS algorithms. Once more the step-size μ was set to one in this experiment.

Table 2.2: Excess of Mean-Square Error

Algorithm Type	$(\xi_{exc})_{dB}$	
	Stationary	Nonstationary
NLMS	-59.09	-39.15
NNDR-LMS	-59.40	-39.42
BNDR-LMS	-58.60	-39.45

Other experiments were carried out in order to test theoretical results obtained from the convergence analysis. The input signal in this case was white noise and the excess of MSE was measured for different values of the step-size (μ varying from 0.1 to 1.9). Once it was shown that the filter order N has a great influence on the theoretical results, the experiment was repeated for $N = 5$, $N = 10$, and $N = 63$. The results are depicted in Figures 2.5, 2.6, and 2.7 respectively, where we can see that the theoretical curve is closer to the experimental curve as N is increased. Furthermore, as N is increased the probability of occurring $\boldsymbol{V}_k = \boldsymbol{V}_{k-1}$ becomes less likely and the curves approach the one obtained in Experiment 2 of the previous section.

A last experiment was designed to test the influence of colored signals on the excess of MSE and the accuracy of the expressions derived in the analysis. Four situations were contemplated corresponding to input signals having different charac-

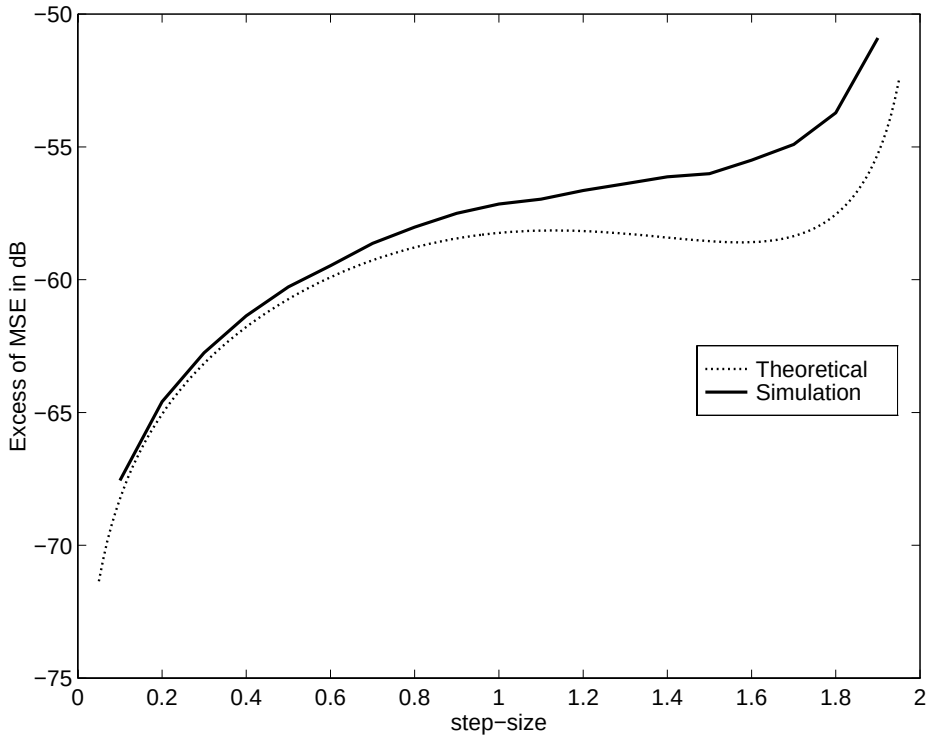

Figure 2.5: Excess of MSE for $N = 5$ as a function of μ.

teristics (all of them with $N = 10$). In the first two situations, signals were obtained from zero-mean white-Gaussian sequences filtered by first-order allpole IIR filters with poles at 0.8 and 0.9, yielding autocorrelation matrices with eigenvalue ratios of 50.85 and 145.44, respectively. In the other two situations, input-signal vectors were generated with discrete radial probability distributions and autocorrelation matrices with eigenvalue spreads also equal to 50.85 and 145.44, respectively. The excess of MSE in dB for these simulations are depicted in Figure 2.8 where simulation results and theoretical curves are confronted. Theoretical values were calculated using (2.37)–(2.39) with probabilities given by (2.45) and (2.46). The analysis for colored input-signals presented very good agreement with the simulations carried out for input-signal vectors presenting discrete angular probability distributions. For the signals obtained by filtering white-Gaussian sequences with first-order allpole IIR filters, only a reasonably accurate qualitative description of the evolution of the ex-

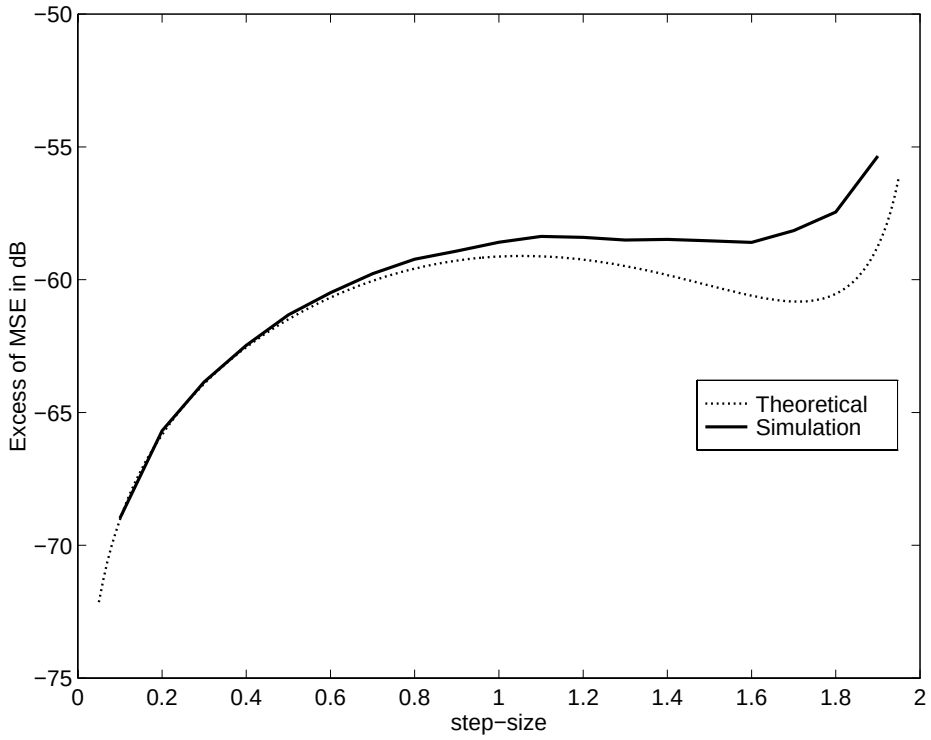

Figure 2.6: Excess of MSE for $N = 10$ as a function of μ.

cess of MSE with respect to the step-size could be observed. This can be explained by the fact that the expressions were derived for a white signal situation. Moreover, in the range of interest ($0 < \mu \leq 1$) the difference between the simulated and the theoretical curves is less than 3 dB. The range of values of μ from 1 to 2 is not used in practical situation since such value would worsen the performance of the algorithm by increasing the misadjustment without improving the convergence rate which is maximum for $\mu = 1$.

In terms of computational complexity, Table 2.3 shows the comparisons among the three normalized algorithms mentioned before. Note that $p = N + 1$ is the number of coefficients. By observing this table, we can conclude that the computational load of the BNDR-LMS algorithm is slightly higher than the computational load of the NNDR-LMS algorithm (which is equal to $L + 1 = 2$ times the complexity of the NLMS algorithm). We stress the fact that this table is relative to one only re-use

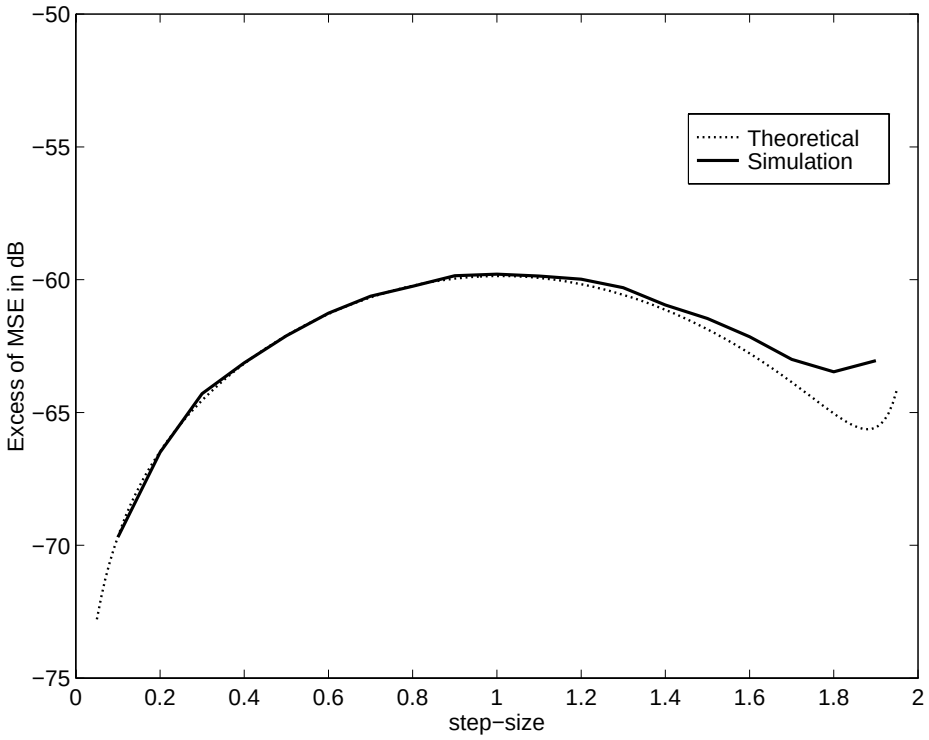

Figure 2.7: Excess of MSE for $N = 63$ as a function of μ.

($L = 1$) and that, in the case of the first experiment, for the same performance of the NNDR-LMS algorithm as compared to the BNDR-LMS algorithm, a complexity $L + 1 = 5$ times the complexity of the NLMS algorithm would be required.

Table 2.3: Comparison of computational complexity

ALG.	ADD	MULT.	DIV.
NLMS	3p-1	3p	1
NNDR-LMS	6p-2	6p	2
BNDR-LMS	6p+1	6p+8	2

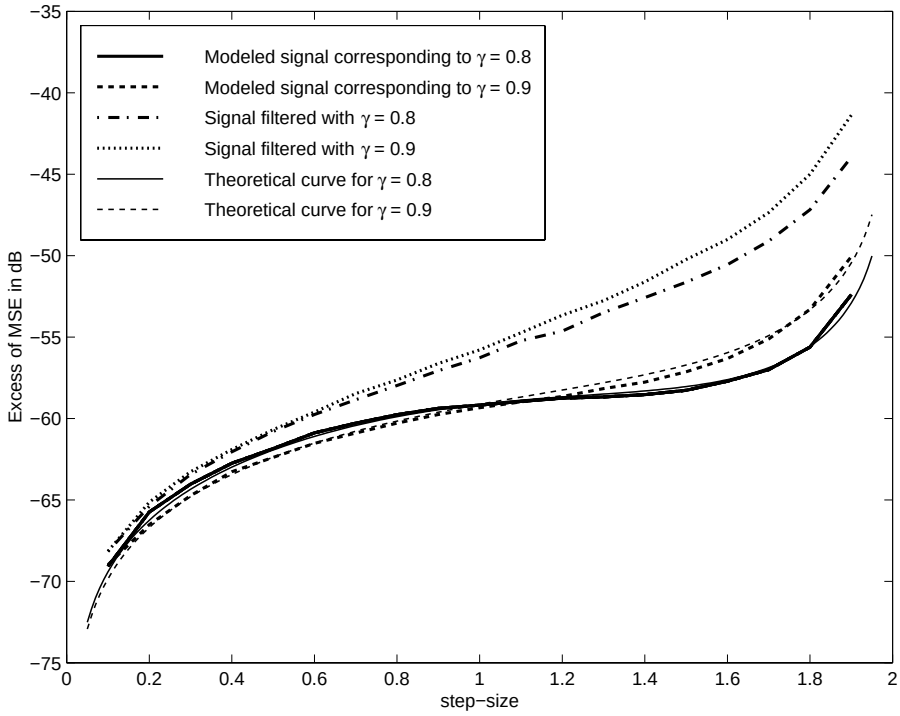

Figure 2.8: Excess of MSE for colored input signals.

2.5 Conclusions

This chapter presented the BNDR-LMS algorithm along with its analyses of convergence and mean-square error. A geometric interpretation of the algorithm was also provided showing that the coefficients are updated in two normalized steps following orthogonal directions. The relationship between the BNDR-LMS algorithm and the orthogonal-projection algorithm was clarified.

Simulations carried out in a system identification application showed that the BNDR-LMS algorithm compares favorably with other LMS-like algorithms in terms of speed of convergence. Moreover, the more correlated is the input signal, the better is the performance of the new algorithm when compared with other LMS-like algorithms. This improvement is more clearly observed in cases of small measurement noise.

Analyses in the mean and the covariance were carried out being the latter based

on a simplified model for the input signal which rendered tractable expressions for the complex problem of analyzing data-reusing algorithms. Consistency with the first two moments of the input signal are maintained by the model. For white input signals, analysis of mean-square error, which is in excellent agreement with simulations, was carried out. Limits for convergence in the mean and the covariance of the coefficient vector were also established. Moreover, a closed-form expression for the excess of MSE as a function of the step-size was derived for the case of white input signals. The applicability of this expression for the case of colored input signals was also addressed. The model and the analyses can be readily extended to other data-reusing algorithms that have not been considered in the past due to exceeding complexity.

Chapter 3

A Practical Application of the BNDR-LMS Algorithm

3.1 Introduction

In order to illustrate a practical application of the BNDR-LMS algorithm, this chapter presents a constrained version of this algorithm as well as proposes an optimal step-size sequence which allows fast convergence and minimum misadjustment. The new algorithm is applied to a direct-sequence code-division multiple access (DS-CDMA) mobile receiver and the results has shown a considerable speed up of the convergence rate compared to the traditional approach using the LMS algorithm.

A constrained adaptive filter has several fields of applications such as antenna array processing and interference cancellation in DS-CDMA mobile communication systems. The two traditional methods used in these applications are the so called Frost [30] approach and the general sidelobe canceler (GSC) [31] approach. The GSC approach converts the constrained problem into an unconstrained problem. The Frost scheme is probably the most widely used technique due to the simplicity of the LMS algorithm. Nevertheless, the main drawback of the LMS algorithm is also present in Frost scheme; that is, its performance is strongly dependent on the eigenvalue spread of the input-signal autocorrelation matrix. An alternative approach is the use of fast least-squares techniques as proposed in [32]. Since the

Frost algorithm turns out to be the projection of the conventional LMS result onto a constrained hyperplane, a natural step would be to use the traditional normalized LMS-like algorithms [2, 33] followed by a projection just like in the LMS Frost case. This approach results in a superior convergence rate compared to the LMS Frost algorithm when the input signals are strongly correlated. This intuitive approach, however, lacks an optimization criterion and performs worse than the corresponding GSC structure in some cases.

It was observed in our experiments that the constrained LMS algorithm in both implementations, Frost and GSC schemes, give identical results [31] when applied to the same set of data. Our goal in this chapter is the derivation of the constrained BNDR-LMS algorithm, Frost-like structure, such that it presents identical results when compared to the results obtained by employing the BNDR-LMS algorithm in the GSC structure.

This chapter is organized as follows. Section 3.2 presents an alternative derivation of the conventional NLMS and BNDR-LMS algorithms. In Section 3.3 the constrained normalized algorithms are derived based on the approach used in Section 3.2. In Section 3.4 the optimal step-sized sequence is derived. Section 3.5 shows some simulation results in the typical field of application proposed here, followed by conclusions.

3.2 Re-Derivation of the NLMS and the BNDR-LMS Algorithms

In this section we show an alternative way of deriving the NLMS and the BNDR-LMS algorithms. Let us start with the normalized LMS. Suppose we have an LMS-like algorithm that updates the coefficient vector according to the following expression.

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu_k \boldsymbol{x}(k) \tag{3.1}$$

where $\boldsymbol{w}(k)$ is the coefficient vector (of size $(N+1) \times 1$ where N is the order of the adaptive filter) at instant k, $\boldsymbol{x}(k)$ is the input signal vector and μ_k is the variable step-

size (or convergence factor) which must be chosen with the objective of achieving faster convergence. The strategy used here is to reduce the instantaneous squared error as much as possible since this is a good and simple estimate of the mean squared error (MSE) [2]. Since the instantaneous error is given by $e(k) = d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)$, the instantaneous squared error at instant k after the *updating* of the coefficient vector can be written as

$$\begin{aligned} e^{\star 2}(k) &\triangleq [d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k+1)]^2 \\ &= [d(k) - \boldsymbol{x}^T(k)(\boldsymbol{w}(k) + \mu_k \boldsymbol{x}(k))]^2 \end{aligned} \tag{3.2}$$

where the star (*) indicates the *a posteriori* error. In order to increase the convergence rate by choosing an appropriate step-size, we take the partial derivative of $e^{\star 2}(k)$ with respect to μ_k and make it equal to zero, obtaining

$$\mu_k = \frac{d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)}{\boldsymbol{x}^T(k)\boldsymbol{x}(k)} \tag{3.3}$$

which corresponds, as expected, to the traditional normalized LMS algorithm.

In the BNDR-LMS algorithm, we update the coefficient vector by adding the input-signal vectors $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$ weighted by two step-sizes, μ_{1k} and μ_{2k}, respectively.

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu_{1k}\boldsymbol{x}(k) + \mu_{2k}\boldsymbol{x}(k-1) \tag{3.4}$$

In this case, we minimize the cost function $F(k)$ which corresponds to the instantaneous squared error at instant k plus the instantaneous squared error at instant $k-1$ calculated with the coefficient vector of instant k or $F(k) = [d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)]^2 + [d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}(k)]^2$. We next define $F^\star(k)$ as $F(k)$ calculated with the updated coefficient vector.

$$\begin{aligned} F^\star(k) \triangleq\ & [d(k) - \boldsymbol{x}^T(k)(\boldsymbol{w}(k) + \mu_{1k}\boldsymbol{x}(k) + \mu_{2k}\boldsymbol{x}(k-1))]^2 \\ +& [d(k-1) - \boldsymbol{x}^T(k-1)(\boldsymbol{w}(k) + \mu_{1k}\boldsymbol{x}(k) + \mu_{2k}\boldsymbol{x}(k-1))]^2 \end{aligned} \tag{3.5}$$

In the next step, we take the partial derivatives of $F^\star(k)$ with respect to μ_{1k} and

μ_{2k} and make them equal to zero. After some algebraic manipulations we obtain

$$\begin{aligned} e_1 &= d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k) \\ e_2 &= d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}(k) \\ den &= \boldsymbol{x}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k-1) - (\boldsymbol{x}^T(k-1)\boldsymbol{x}(k))^2 \\ \mu_{1k} &= \frac{e_1\boldsymbol{x}^T(k-1)\boldsymbol{x}(k-1) - e_2\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)}{den} \\ \mu_{2k} &= \frac{e_2\boldsymbol{x}^T(k)\boldsymbol{x}(k) - e_1\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)}{den} \end{aligned} \tag{3.6}$$

which together with (3.4) correspond to the binormalized data-reusing LMS (BNDR-LMS) algorithm.

3.3 The Constrained Algorithm

In linearly constrained adaptive filtering, the J constraints are represented by the following linear system.

$$\boldsymbol{C}^T\boldsymbol{w}(k) = \boldsymbol{f} \tag{3.7}$$

where $\boldsymbol{C}$ is a $(N+1)\times J$ matrix containing the constraint vectors, and $\boldsymbol{f}$ is a vector of J elements containing the constraint values (one single constraint means that $\boldsymbol{C}$ is a vector and $\boldsymbol{f}$ is a scalar).

In the LMS case (Frost structure), the resulting algorithm is given by the projection of the coefficient vector — $\boldsymbol{w}(k+1)$ unconstrained — onto the hyperplane defined by (3.7). The constrained coefficient vector is obtained by first projecting the unconstrained solution onto the homogeneous hyperplane $\boldsymbol{C}^T\boldsymbol{w}(k) = \boldsymbol{0}$ with the help of the projection matrix $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T$. Finally, the resulting vector is moved back to the constraint hyperplane by adding the vector $\boldsymbol{F} = \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{f}$.

$$\begin{aligned} \boldsymbol{w}(k+1) &= \boldsymbol{P}\boldsymbol{w}_{LMS}(k+1) + \boldsymbol{F} \\ &= \boldsymbol{P}[\boldsymbol{w}(k) + \mu e(k)\boldsymbol{x}(k)] + \boldsymbol{F} \end{aligned} \tag{3.8}$$

where $e(k) = d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k)$.

Our approach here for both the NLMS and the BNDR-LMS algorithms is the projection of the unconstrained solution followed by the optimization of the step-size(s) similar to what was done in the previous section. Let us start with the NLMS algorithm by taking $\boldsymbol{w}_{NLMS}(k+1)$ as in 3.1 where μ_k is the variable step size we wish to obtain.

$$\begin{aligned} \boldsymbol{w}(k+1) &= \boldsymbol{P}\boldsymbol{w}_{NLMS}(k+1) + \boldsymbol{F} \\ &= \boldsymbol{P}[\boldsymbol{w}(k) + \mu_k \boldsymbol{x}(k)] + \boldsymbol{F} \end{aligned} \tag{3.9}$$

If we remember that $\boldsymbol{w}(k)$ was forced to satisfy the constraint in (3.7) which means that $\boldsymbol{P}\boldsymbol{w}(k) + \boldsymbol{F} = \boldsymbol{w}(k)$, it follows that (3.9) can be written as

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \mu_k \boldsymbol{P}\boldsymbol{x}(k) \tag{3.10}$$

If we now compare (3.1) and (3.10) we can see that they are the same problem if we substitute the input vector by a rotated version $\boldsymbol{x}'(k) = \boldsymbol{P}\boldsymbol{x}(k)$. Moreover, recalling that $\boldsymbol{P}^2 = \boldsymbol{P}$, it follows from the same approach used in the previous section that

$$\begin{aligned} e(k) &= d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k) \\ \boldsymbol{w}(k+1) &= \boldsymbol{P}\left[\boldsymbol{w}(k) + \frac{e(k)\boldsymbol{x}(k)}{\boldsymbol{x}^T(k)\boldsymbol{P}\boldsymbol{x}(k)}\right] + \boldsymbol{F} \end{aligned} \tag{3.11}$$

which correspond to the constrained NLMS algorithm [34].

The same approach can be applied to the BNDR-LMS if we make

$$\begin{aligned} \boldsymbol{w}(k+1) &= \boldsymbol{P}\boldsymbol{w}_{BNDR-LMS}(k+1) + \boldsymbol{F} \\ &= \boldsymbol{P}[\boldsymbol{w}(k) + \mu_{1k}\boldsymbol{x}(k) + \mu_{2k}\boldsymbol{x}(k-1)] + \boldsymbol{F} \\ &= \boldsymbol{w}(k) + \mu_{1k}\boldsymbol{P}\boldsymbol{x}(k) + \mu_{2k}\boldsymbol{P}\boldsymbol{x}(k-1) \end{aligned} \tag{3.12}$$

and compare with (3.4). The equations of the constrained BNDR-LMS algorithm

are obtained as

$$
\begin{aligned}
e_1 &= d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k) \\
e_2 &= d(k-1) - \boldsymbol{x}^T(k-1)\boldsymbol{w}(k) \\
\boldsymbol{P} &= \boldsymbol{I} - \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{C}^T \\
\boldsymbol{F} &= \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{C})^{-1}\boldsymbol{f} \\
den &= \boldsymbol{x}^T(k)\boldsymbol{P}\boldsymbol{x}(k)\boldsymbol{x}^T(k-1)\boldsymbol{P}\boldsymbol{x}(k-1) - (\boldsymbol{x}^T(k-1)\boldsymbol{P}\boldsymbol{x}(k))^2 \\
\mu_{1k} &= \frac{e_1\boldsymbol{x}^T(k-1)\boldsymbol{P}\boldsymbol{x}(k-1) - e_2\boldsymbol{x}^T(k-1)\boldsymbol{P}\boldsymbol{x}(k)}{den} \\
\mu_{2k} &= \frac{e_2\boldsymbol{x}^T(k)\boldsymbol{P}\boldsymbol{x}(k) - e_1\boldsymbol{x}^T(k-1)\boldsymbol{P}\boldsymbol{x}(k)}{den} \\
\boldsymbol{w}(k+1) &= \boldsymbol{P}[\boldsymbol{w}(k) + \mu_{1k}\boldsymbol{x}(k) + \mu_{2k}\boldsymbol{x}(k-1)] + \boldsymbol{F}
\end{aligned}
\tag{3.13}
$$

It is worth mentioning that these two constrained algorithms present identical results when compared to the NLMS and BNDR-LMS algorithms used in the GSC structure. It is also interesting to remark that (3.11) and (3.13) can be simplified by admitting that $\boldsymbol{P}\boldsymbol{w}(k) + \boldsymbol{F} = \boldsymbol{w}(k)$. This simplification, however, can produce round-off error accumulation when the algorithm is implemented in finite-precision environment.

3.4 Step-Size Optimization of the BNDR-LMS Algorithm

We have seen that the BNDR-LMS algorithm offers faster convergence than a number of other normalized LMS algorithms for a highly correlated input signals at the cost of a small additional complexity. The MSE after convergence for this algorithm is controlled by a step-size parameter μ. For $\mu = 1$, we have the fastest convergence and also the highest steady-state MSE when compared to the cases where the values of the step-size are closer to zero. In the previous chapter, it was shown that the BNDR-LMS algorithm convergences if the step-size is in the range from zero to two. For practical reasons, the value of μ is kept between zero and one since it was

observed that the steady-state MSE was higher and the convergence slower when the step-size was set to a value between one and two.

In this section, the expression for the MSE developed in Chapter 2 is used to propose an optimal step-size sequence which allows fast convergence and minimum misadjustment. The final expression for the convergence behavior of the BNDR-LMS algorithm is rewritten here in terms of the excess in the MSE.

$$\begin{aligned}\Delta\xi(k+1) &= \left[1+\frac{\mu(\mu-2)}{N+1}\right]\Delta\xi(k) \\ &\quad +\frac{N\mu(1-\mu)^2(\mu-2)}{(N+1)^2}\Delta\xi(k-1) \\ &\quad +\frac{(1+N(\mu-2)^2)\mu^2}{(N+1)(N+2-\nu_x)}\sigma_n^2 \end{aligned} \tag{3.14}$$

From the expression of $\Delta\xi(k+1)$ above, we will follow an approach similar to that used in [4] and we start by rewriting (3.14) assuming that up to instant k we have the optimal sequence $\mu^*(0)$ to $\mu^*(k-1)$ already available and also the optimal quantities $\Delta\xi^*(k)$ and $\Delta\xi^*(k-1)$.

$$\begin{aligned}\Delta\xi(k+1) &= \left[1+\frac{\mu(k)(\mu(k)-2)}{N+1}\right]\Delta\xi^*(k) \\ &\quad +\frac{N\mu(k)(1-\mu(k))^2(\mu(k)-2)}{(N+1)^2}\Delta\xi^*(k-1) \\ &\quad +\frac{(1+N(\mu(k)-2)^2)\mu(k)^2}{(N+1)^2}\sigma_n^2 \end{aligned} \tag{3.15}$$

If we now compute the derivative of $\Delta\xi(k+1)$ with respect to $\mu(k)$ and make it equal to zero, we obtain after some algebraic manipulation

$$\begin{aligned}\mu^*(k) &= 1-\sqrt{1-\frac{\Delta\xi^*(k)+\Delta\xi^*(k-1)}{2(\Delta\xi^*(k-1)+\sigma_n^2)}} \\ &= 1-\sqrt{1-\frac{\xi^*(k)+\xi^*(k-1)-2\sigma_n^2}{2\xi^*(k-1)}} \end{aligned} \tag{3.16}$$

It is worth mentioning that (3.16) is in accordance with the situation corresponding to when the convergence is reached; in that case we have $\xi^*(k)=\xi^*(k-1)=\sigma_n^2$ and therefore we have $\mu^*(k)=0$ as expected. Moreover, if we have $\sigma_n^2=0$ the value of $\mu^*(k)$ will be close to one (admitting that $\Delta\xi^*(k)\approx\Delta\xi^*(k-1)$) even after

convergence, which means that we should have maximum speed of convergence with minimum misadjustment if the noise is zero.

For the normalized LMS (NLMS) algorithm, a recursive formula for $\mu^*(k)$ in terms if $\mu^*(k-1)$ and the order N was obtained in [4]. In the case of the BNDR-LMS algorithm, a simple recursive expression was not obtained and a small algorithm was used to produce the optimal step-size sequence. This algorithm is presented in Table 3.1 [1] and has one important initialization parameter with a strong influence on the behavior of $\mu^*(k)$. This parameter is the ratio $\frac{\sigma_d^2}{\sigma_n^2}$ where the numerator is the variance of the reference signal.

Table 3.1: Algorithm for computing the optimal step-size sequence.

$\mu(k)$ of the BNDR-LMS algorithm
$\Delta\xi(0) = \Delta\xi(-1) = \sigma_d^2$
$\sigma_n^2 =$ noise variance
$N =$ adaptive filter order
$\mu(0) = 1$
for each k
$\{\ \mu(k) = 1 - \sqrt{1 - \frac{\Delta\xi(k)+\Delta\xi(k-1)}{2(\Delta\xi(k-1)+\sigma_n^2)}}$
$aa = \left[1 + \frac{\mu(k)(\mu(k)-2)}{N+1}\right]$
$bb = \frac{N\mu(k)(1-\mu(k))^2(\mu(k)-2)}{(N+1)^2}$
$cc = \frac{(1+N(\mu(k)-2)^2)\mu(k)^2}{(N+1)^2}\sigma_n^2$
$\Delta\xi(k+1) = aa\Delta\xi(k) + bb\Delta\xi(k-1) + cc$
$\}$

We next present in Figure 3.1 the curves of $\mu(k)$ for different values of what should be called in this case (desired) signal to noise ratio or $SNR = 10log\frac{\sigma_d^2}{\sigma_n^2}$ from 0 to 40 dB. Note that for $\sigma_n^2 = 0$ (noiseless case), the SNR approaches infinity whereas the step-size remains fixed at unity.

[1] Note that the asterisk (*) was dropped from the optimal values for simplicity only.

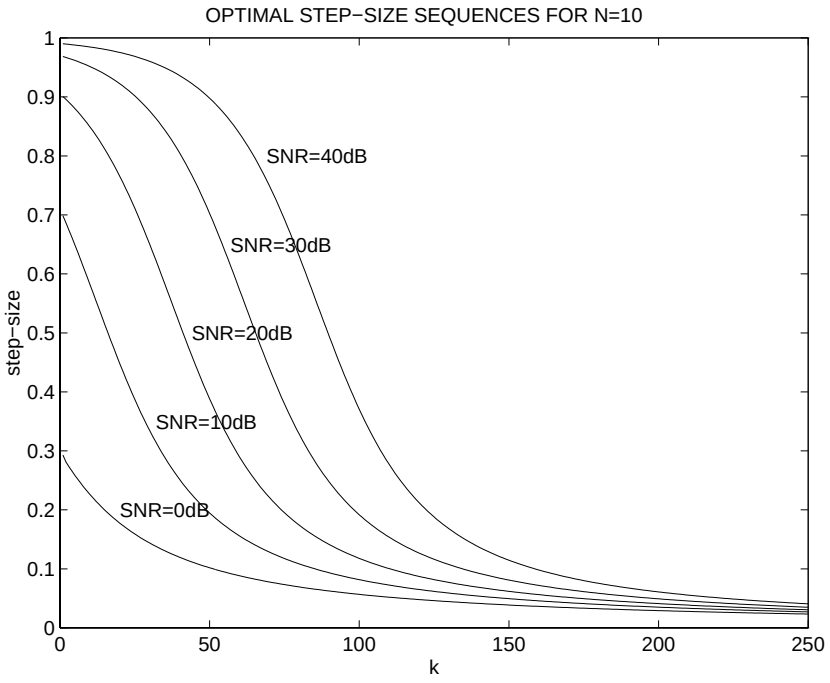

Figure 3.1: Optimal $\mu(k)$ sequences for the BNDR-LMS algorithm.

In a practical implementation the optimal sequence can be computed *a priori* and stored in memory or computed on the fly. For this last option, since a recursive and compact formula is not available, an approximation of the curve is of great interest. We will use here two classes of sequences also proposed in [4]. They were chosen due to their simplicity and, as will be seen later, lead to good results. The first class is the optimal sequence for the NLMS algorithm. It is given by

$$\mu(k) = \mu(k-1)\frac{1 - \frac{\mu(k-1)}{N+1}}{1 - \frac{\mu^2(k-1)}{N+1}} \tag{3.17}$$

For the NLMS algorithm, the correct initialization for this sequence is given by $\mu(0) = 1 - \frac{\sigma_n^2}{\sigma_d^2}$. However, in our case we can choose an initial value for the step-size such that the two sequences are close, as will be seen.

The second class of sequences (referred to hereafter as the $1/k$ approximation) is quite simple and was also used in [4]. This sequence is given by

$$\mu(k) = \begin{cases} 1 & \text{if } 0 \leq k \leq c(N+1) \\ max\{\mu_{min}, \frac{1}{1-c+\frac{k}{N+1}}\} & \text{if } k > c(N+1) \end{cases} \tag{3.18}$$

The parameter c will be related to the SNR of the optimal sequence. A minimum step-size was introduced here (it can be used in all sequences as well) in order to provide a tracking capability to the algorithm.

The results of a few experiments will demonstrate the superior performance obtained with the proposed adaptive step-size scheme. For the first simulation, we used a white noise input signal in a system identification setup with $N = 10$, $\sigma_n^2 = 10^{-2}$ and $SNR = 20dB$. Figure 3.2 shows the optimal step-size sequence obtained with the algorithm described in Table 3.1 and other curves from the two classes of approximations used.

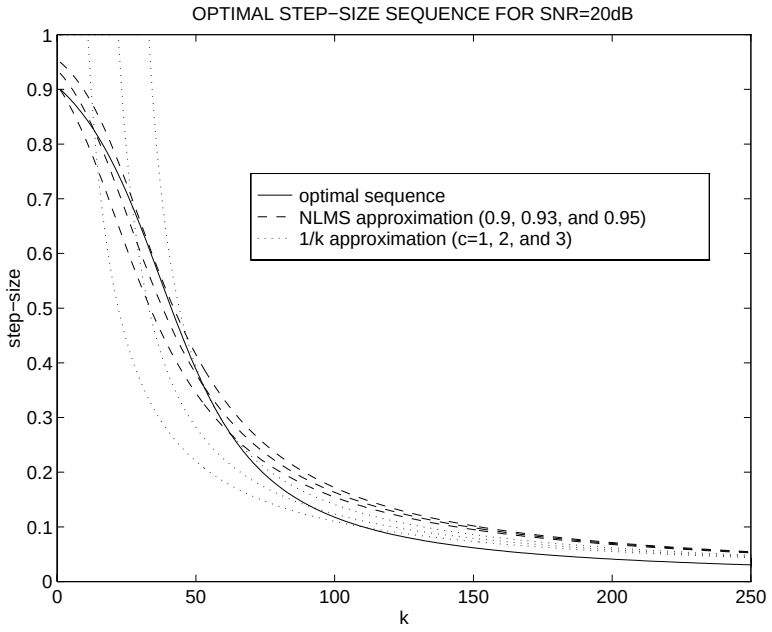

Figure 3.2: Optimal step-size sequence and two classes of approximation sequences.

From Figure 3.2, we can conjecture which curve to use. If we use the least norm of the difference between the optimal and the approximation sequences as a criterion to decide which curve to implement, the chosen parameters for this example would be $\mu(0) = 0.93$ and $c = 3$.

With these parameters we have run a simulation with a fixed step-size, an optimal step-size and the two approximations. The learning curves (average of 1000 runs) are depicted in Figure 3.3 where we can see that the same fast convergence and

the same small steady-state MSE are shared by the three time-varying step-size sequences used. The fixed step-size was set to one and, as expected, has the highest misadjustment.

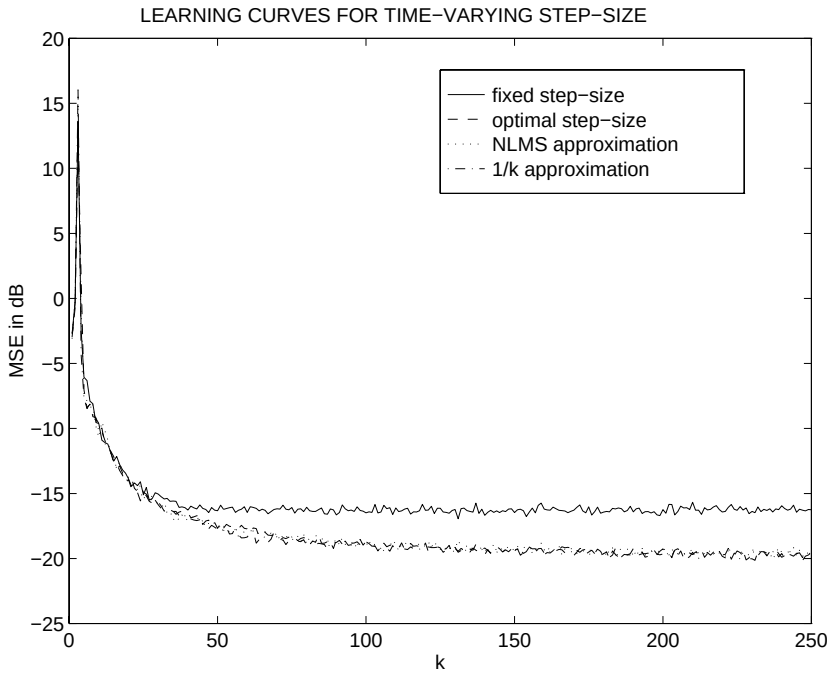

Figure 3.3: Learning curves for the fixed step-size, the optimal step-size and its two approximations.

A second experiment was carried out in order to evaluate the performance of this optimal sequence in case where the input signal is correlated. The same setup was used but with an input signal having a condition number (ratio between the largest and the smallest eigenvalue of the input signal autocorrelation matrix) around 180. Figure 3.4 shows us that, even for a correlated input signal, the proposed step-size sequence has a good performance. We observe on this same figure that, in this given example, the BNDR-LMS algorithm using optimal step-size sequence has better performance than the NLMS algorithm also using its optimal step-size sequence.

A final remark is the possibility to use an estimator for $\xi(k)$ instead of calculating $\Delta\xi(k)$ using (3.15) as described in the algorithm of Table 3.1. We have also made

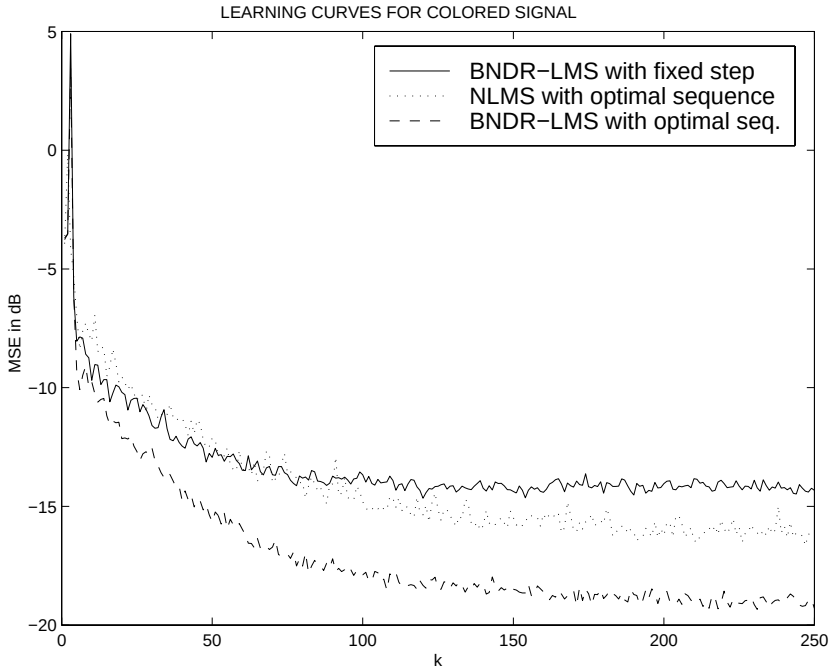

Figure 3.4: Comparing the learning curves for the case of colored input signal.

an experiment using the following estimator:

$$\xi(k+1) = \lambda\xi(k) + (1-\lambda)e^2(k) \tag{3.19}$$

This experiment has shown us that a reasonable value for λ is around 0.96. The advantage of this alternative approach is the possibility of fast tracking of sudden and strong changes in the environment. In this case, the instantaneous error becomes high and the estimated $\xi(k+1)$ is increased such that the value of μ approaches the unity again and a fast re-adaptation starts.

When using this approach, it is worth remembering that, since equation (3.16) is of the type $1-\sqrt{1-x}$, the step-size $\mu(k)$ can be written as $\frac{x}{1+\sqrt{1-x}}$ which is a numerically less sensitive expression. Equation (3.20) shows this expression.

$$\mu(k) = \frac{\frac{\xi(k)+\xi(k-1)-2\sigma_n^2}{2\xi(k-1)}}{1+\sqrt{1-\frac{\xi(k)+\xi(k-1)-2\sigma_n^2}{2\xi(k-1)}}} \tag{3.20}$$

3.5 Simulation Results

In this section, we apply the constrained adaptive algorithms to the case of a DS-CDMA downlink transmission system. The received signal for a system with K simultaneous users can be written as

$$\boldsymbol{x}(k) = \sum_{i=1}^{K} A_i b_i(k) \boldsymbol{s}_i + \boldsymbol{n}(k) \tag{3.21}$$

where A_i is the signal amplitude of user i, $\boldsymbol{s}_i$ is the signature sequence of the ith user, and $b_i(k) \in \{\pm 1\}$ is the transmitted bit of the ith user. At the mobile receiver we are only interested in detecting one user (here assumed to be $i = 1$). One way of constructing the receiver coefficients is to minimize its output energy under the constraint that the desired user's code sequence can pass with unity response. The problem is then to find a coefficient vector $\boldsymbol{w}(k)$ such that solves

$$\min_{\boldsymbol{w}(k)} \|\boldsymbol{x}^T(k)\boldsymbol{w}(k)\|^2 \quad subject\ to \quad \boldsymbol{s}_1^T \boldsymbol{w}(k) = 1 \tag{3.22}$$

where, using the notation of the previous section, we see that the reference signal $d(k) = 0$, $\boldsymbol{C} = \boldsymbol{s}_1$ and $\boldsymbol{f} = 1$. The system used in our experiment consists of $K = 5$ users whose spreading sequences were Gold codes of length 7 [35]. The signal-to-noise ratio (SNR) for the desired user was set to $8dB$ and the interfering users power was set to 20 times stronger than the desired user power ($A_i = \sqrt{20}$ for $i \neq 1$). In the simulation, we have used the optimal step-size sequence [33] described in the previous section.

Figure 3.5 shows the learning curves for the LMS, NLMS, and BNDR-LMS algorithms (average of 500 runs). The step-size for the LMS algorithm was chosen to be $\mu = 5.10^{-4}$ such that its misadjustment is comparable with the other normalized algorithms using optimal sequences. As can be seen from the figure, the performance of the normalized algorithms are superior to the LMS algorithm in terms of convergence rate. Probably due to the input signal which in this example is not correlated enough, the BNDR-LMS algorithm could not have the best performance and the NLMS algorithm is suggested in these cases. This assertion is supported by the fact that even the RLS algorithm have not shown a much better performance

than the NLMS algorithm in this particular example. It is worth mentioning that the general sidelobe canceler (GSC) structure using the NLMS and the BNDR-LMS algorithms was also simulated and presented identical learning curves.

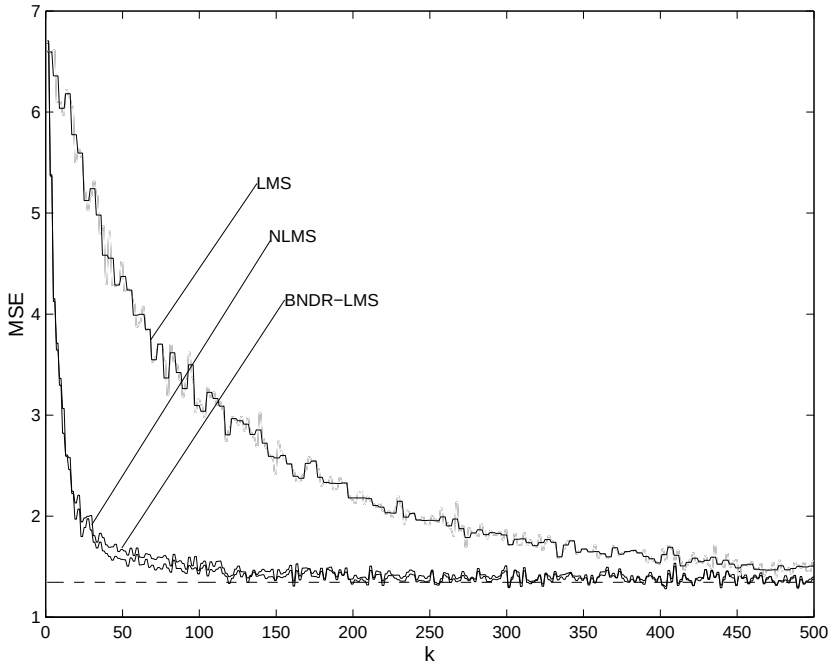

Figure 3.5: Learning curves of the constrained algorithms.

3.6 Conclusions

This chapter introduced the constrained version of the BNDR-LMS algorithm using the structure developed by Frost [30].

A straightforward method of obtaining the normalized (NLMS and BNDR-LMS) algorithms was presented and it was shown that this method is also valid for the constrained versions of these algorithms. The resulting constrained BNDR-LMS algorithm using the Frost structure presented identical results then the unconstrained counterpart using the GSC structure. An optimal step-size sequence for the BNDR-LMS was also obtained. The algorithm was applied to CDMA mobile reception and the simulation results showed faster convergence rate as well as a small misadjustment when the optimal time-varying step-size is used.

Chapter 4

Fast QR Algorithms: a Unified Approach

4.1 Introduction

This section deals with the basic concepts used in the RLS algorithms employing conventional and inverse QR decomposition. The methods of triangularizing the input data matrix and the meaning of the internal variables of these algorithms are emphasized in order to establish the notation used in this work as well as to introduce the most important relations used hereafter.

4.1.1 The Conventional QR Algorithm

As in the conventional RLS algorithm, we are interested in minimizing the following cost function

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i} e^2(i) = \boldsymbol{e}^T(k)\boldsymbol{e}(k) = \| \boldsymbol{e}(k) \|^2 \tag{4.1}$$

where each component of the vector $\boldsymbol{e}(k)$ is the *a posteriori* error at instant i weighted by $\lambda^{(k-i)/2}$ (λ is the forgetting factor and i varies from 0 to k). The vector $\boldsymbol{e}(k)$ is given by

$$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{X}(k)\boldsymbol{w}(k) \tag{4.2}$$

In the above equation, the weighted desired signal vector $\boldsymbol{d}(k)$, the coefficient vector $\boldsymbol{w}(k)$, and the input data matrix $\boldsymbol{X}(k)$ are defined by

$$\boldsymbol{d}(k) = \begin{bmatrix} d(k) \\ \lambda^{1/2} d(k-1) \\ \vdots \\ \lambda^{k/2} d(0) \end{bmatrix} \tag{4.3}$$

$$\boldsymbol{w}(k) = \begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix} \tag{4.4}$$

$$\boldsymbol{X}(k) = \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2} \boldsymbol{x}^T(k-1) \\ \vdots \\ \lambda^{k/2} \boldsymbol{x}^T(0) \end{bmatrix} \tag{4.5}$$

where N is the filter order (number of coefficients minus one), $\boldsymbol{x}(k)$ is the input signal vector $[x(k)\ x(k-1)\ \cdots\ x(k-N)]^T$ and the samples before instant $k = 0$ are considered zeros.

The optimal solution to the least-squares problem at a given instant k can be found by differentiating the cost function with respect to $\boldsymbol{w}(k)$ and equating to zero, resulting in

$$\boldsymbol{w}(k) = \boldsymbol{R}_D^{-1}(k) \boldsymbol{p}_D(k) \tag{4.6}$$

where $\boldsymbol{R}_D(k) = \boldsymbol{X}^T(k)\boldsymbol{X}(k)$ is the deterministic data correlation matrix and $\boldsymbol{p}_D(k) = \boldsymbol{X}^T(k)\boldsymbol{d}(k)$ is the deterministic cross-correlation vector between the input and the desired signal.

Expression (4.6) is used in the conventional RLS approach. The inverse of $\boldsymbol{R}_D(k)$ can become ill-conditioned, e.g., due to loss of persistence of excitation of the input signal or due to quantization effects [2]. In order to avoid possible inaccurate solu-

tions, the QR decomposition approach triangularizes the input data matrix through the use of Givens rotation matrices.

The premultiplication of (4.2) by $\boldsymbol{Q}(k)$ (matrix which represents the overall triangularization process via elementary Givens rotations matrices) triangularizes $\boldsymbol{X}(k)$ and since $\boldsymbol{Q}(k)$ is orthogonal (actually orthonormal), it will not affect the cost function.

$$\boldsymbol{Q}(k)\boldsymbol{e}(k) = \begin{bmatrix} \boldsymbol{e}_{q_1}(k) \\ \boldsymbol{e}_{q_2}(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{d}_{q_1}(k) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} - \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{U}(k) \end{bmatrix} \boldsymbol{w}(k) \tag{4.7}$$

where $\boldsymbol{U}(k)$ is the Cholesky factor of $\boldsymbol{X}^T(k)\boldsymbol{X}(k)$ (i.e. $\boldsymbol{U}^T(k)\boldsymbol{U}(k) = \boldsymbol{X}^T(k)\boldsymbol{X}(k)$) and the subscripts 1 and 2 indicate the first $k-N$ and the last $N+1$ components of the vector, respectively.

The weighted-square error (or cost function) can be minimized by choosing $\boldsymbol{w}(k)$ such that the term $\boldsymbol{d}_{q_2}(k) - \boldsymbol{U}(k)\boldsymbol{w}(k)$ is zero. The tap-weight coefficients are then calculated using a backsubstitution procedure (see [2, 36] for more details).

Using once more the fact that $\boldsymbol{Q}(k)$ is orthonormal and the definition in (4.5), we can write the product $\boldsymbol{Q}(k)\boldsymbol{X}(k)$ as

$$\boldsymbol{Q}(k) \begin{bmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{Q}^T(k-1) \end{bmatrix} \begin{bmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{Q}(k-1) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{X}(k-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{U}(k) \end{bmatrix} \tag{4.8}$$

In the above equation, if we call $\tilde{\boldsymbol{Q}}(k)$ the product of the first two matrices on the left and execute the multiplication of the remaining two matrices, we have

$$\tilde{\boldsymbol{Q}}(k) \begin{bmatrix} \boldsymbol{x}^T(k) \\ \boldsymbol{O} \\ \lambda^{1/2}\boldsymbol{U}(k-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{U}(k) \end{bmatrix} \tag{4.9}$$

From (4.9) we see that $\tilde{\boldsymbol{Q}}(k)$ is a product of Givens rotations matrices that annihilates the current input vector. Moreover, the recursive nature of $\boldsymbol{Q}(k)$ may be expressed by

$$\boldsymbol{Q}(k) = \tilde{\boldsymbol{Q}}(k) \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \boldsymbol{Q}(k-1) \end{bmatrix} \tag{4.10}$$

Once $\tilde{\boldsymbol{Q}}(k)$ is responsible for zeroing $\boldsymbol{x}^T(k)$ as shown in (4.9), its structure includes a submatrix $\boldsymbol{I}_{k-N-1}$. Fortunately, we can avoid the ever increasing order characteristic by deleting this section of $\tilde{\boldsymbol{Q}}(k)$ and rewriting (4.9) as

$$\begin{bmatrix} \mathbf{0}^T(k) \\ \boldsymbol{U}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{U}(k-1) \end{bmatrix} \tag{4.11}$$

where $\boldsymbol{Q}_\theta(k)$ is $\tilde{\boldsymbol{Q}}(k)$ after removing the $\boldsymbol{I}_{k-N-1}$ section along with the corresponding rows and columns.

Recalling (4.7), we see that $\boldsymbol{e}_{q_1}(k) = \boldsymbol{d}_{q_1}(k)$ and the product $\boldsymbol{Q}(k)\boldsymbol{d}(k)$ can be written as

$$\begin{aligned} \begin{bmatrix} \boldsymbol{e}_{q_1}(k) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} &= \tilde{\boldsymbol{Q}}(k) \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \boldsymbol{Q}(k-1) \end{bmatrix} \begin{bmatrix} d(k) \\ \lambda^{1/2}\boldsymbol{d}(k-1) \end{bmatrix} \\ &= \tilde{\boldsymbol{Q}}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\boldsymbol{e}_{q_1}(k-1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} e_{q_1}(k) \\ \lambda^{1/2}\boldsymbol{e}_{q_1}(k-1) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} \end{aligned} \tag{4.12}$$

where the last multiplication can be easily understood if we note in (4.9) that $\tilde{\boldsymbol{Q}}(k)$ will alter only the first and the last $N+1$ components of a vector.

From (4.12), it is also possible to remove the increasing section of $\tilde{\boldsymbol{Q}}(k)$ resulting in the following expression:

$$\begin{bmatrix} e_{q_1}(k) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k-1) \end{bmatrix} \tag{4.13}$$

where $e_{q_1}(k)$ is the first element of the *rotated* error signal and $\boldsymbol{d}_{q_2}(k)$ is a vector with the last $N+1$ elements of the *rotated* desired signal vector.

At this point, it is important to emphasize the structure of $\boldsymbol{Q}_\theta(k)$ as a product of Givens rotation matrices given by $\prod_{i=N}^{0} \boldsymbol{Q}_{\theta_i}(k)$. This structure will depend on the type of triangularization of $\boldsymbol{U}(k)$: upper or lower triangular matrix. This choice, as will be seen later, will also determine two classes of fast algorithms. The way by which matrix $\boldsymbol{U}(k)$ is triangularized is depicted in Fig. 4.1 with the corresponding $\boldsymbol{Q}_{\theta_i}(k)$ being given by

$$UPPER: \quad \boldsymbol{Q}_{\theta_i}(k) = \begin{bmatrix} cos\theta_i(k) & \mathbf{0}^T & -sin\theta_i(k) & \mathbf{0}^T \\ \mathbf{0} & \boldsymbol{I}_i & \mathbf{0} & \mathbf{0}\cdots\mathbf{0} \\ sin\theta_i(k) & \mathbf{0}^T & cos\theta_i(k) & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0}\cdots\mathbf{0} & \mathbf{0} & \boldsymbol{I}_{N-i} \end{bmatrix} \tag{4.14}$$

$$LOWER: \quad \boldsymbol{Q}_{\theta_i}(k) = \begin{bmatrix} cos\theta_i(k) & \mathbf{0}^T & -sin\theta_i(k) & \mathbf{0}^T \\ \mathbf{0} & \boldsymbol{I}_{N-i} & \mathbf{0} & \mathbf{0}\cdots\mathbf{0} \\ sin\theta_i(k) & \mathbf{0}^T & cos\theta_i(k) & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0}\cdots\mathbf{0} & \mathbf{0} & \boldsymbol{I}_i \end{bmatrix} \tag{4.15}$$

It is important to remark that the angles $\theta_i(k)$ in (4.14) and (4.15) are not the same although written in the same way for the sake of simplicity. It is also relevant to mention that the two possibilities for a upper triangular matrix (zeros triangle on the lower right side as in the figure or zeros triangles on the lower left side) as well as the two possibilities for a lower triangular matrix (zeros triangle on the upper left side as in the figure or zeros triangle on the upper right side) lead to the identical algorithms.

As an example, the structures of $\boldsymbol{Q}_\theta(k)$ for upper and lower triangularizations of $\boldsymbol{U}(k)$ with $N=2$ are given by

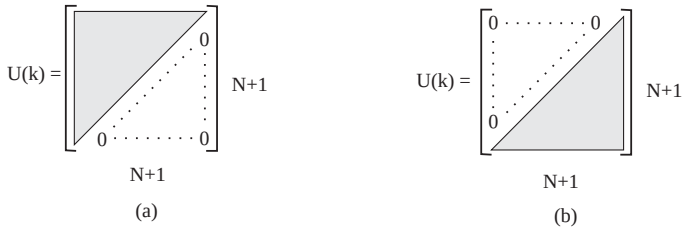

Figure 4.1: The different triangularizations of $\boldsymbol{U}(k)$: (a) UPPER and (b) LOWER.

$$UPPER: \quad \boldsymbol{Q}_\theta(k) = \begin{bmatrix} c\theta_2 c\theta_1 c\theta_0 & -c\theta_2 c\theta_1 s\theta_0 & -c\theta_2 s\theta_1 & -s\theta_2 \\ s\theta_0 & c\theta_0 & 0 & 0 \\ s\theta_1 c\theta_0 & -s\theta_1 s\theta_0 & c\theta_1 & 0 \\ s\theta_2 c\theta_1 c\theta_0 & -s\theta_2 c\theta_1 s\theta_0 & -s\theta_2 s\theta_1 & c\theta_2 \end{bmatrix} \tag{4.16}$$

$$LOWER: \quad \boldsymbol{Q}_\theta(k) = \begin{bmatrix} c\theta_2 c\theta_1 c\theta_0 & -s\theta_2 & -c\theta_2 s\theta_1 & -c\theta_2 c\theta_1 s\theta_0 \\ s\theta_2 c\theta_1 c\theta_0 & c\theta_2 & -s\theta_2 s\theta_1 & -s\theta_2 c\theta_1 s\theta_0 \\ s\theta_1 c\theta_0 & 0 & c\theta_1 & -s\theta_1 s\theta_0 \\ s\theta_0 & 0 & 0 & c\theta_0 \end{bmatrix} \tag{4.17}$$

where $c\theta_i = cos\theta_i(k)$ and $s\theta_i = sin\theta_i(k)$.

The last two equations suggest that $\boldsymbol{Q}_\theta(k)$ in both cases can be partitioned as

$$\boldsymbol{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \boldsymbol{g}^T(k) \\ \boldsymbol{f}(k) & \boldsymbol{E}(k) \end{bmatrix} \tag{4.18}$$

where $\gamma(k) = \prod_{i=0}^{N} cos\theta_i(k)$ and the elements of $\boldsymbol{f}(k)$, $\boldsymbol{g}(k)$ and $\boldsymbol{E}(k)$ depend on the type of triangularization.

In order to have all equations of the traditional (O[N^2]) QR algorithm, let us postmultiply the transposed vector $\boldsymbol{e}_q^T(k)\boldsymbol{Q}(k)$ by the pinning vector $[1\ 0\ \cdots\ 0]^T$.

$$\boldsymbol{e}_q^T(k)\boldsymbol{Q}(k) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \boldsymbol{e}^T(k)\boldsymbol{Q}^T(k)\boldsymbol{Q}(k) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = e(k) \tag{4.19}$$

On the other hand, from equations (4.10) and (4.18) and the fact that $\boldsymbol{Q}_\theta(k)$ is $\tilde{\boldsymbol{Q}}(k)$ after removing the $k-N-1$ increasing columns and rows, we can conclude that $\boldsymbol{Q}(k)[1\ 0\ \cdots\ 0]^T = [\gamma(k)\ 0\ \cdots\ 0\ \boldsymbol{f}^T(k)]^T$. Once $\boldsymbol{e}_{q_2}(k)$ is a null vector (keep in mind that $\boldsymbol{w}(k)$ in (4.7) was chosen in order to make it zero), it is easy to show that

$$e(k) = e_{q_1}(k)\gamma(k) \tag{4.20}$$

This last equation is particularly useful in applications where the coefficients of the adaptive filter are not necessary since we can obtain $e(k)$ without calculating $\boldsymbol{w}(k)$.

The equations of the conventional QR algorithm are presented in Table 4.1. It is worth mentioning that in this case the type of triangularization used has no influence on the performance of the algorithm and also that if the tap-weight coefficients are required, they can be calculated using the backsubstitution procedure [2].

Table 4.1: The conventional QR equations.

QR
for each k
{ Obtaining $\boldsymbol{Q}_\theta(k)$ and updating $\boldsymbol{U}(k)$:
$\begin{bmatrix} \mathbf{0}^T \\ \boldsymbol{U}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{U}(k-1) \end{bmatrix}$
Obtaining $\gamma(k)$:
$\gamma(k) = \prod_{i=0}^{N} cos\theta_i(k)$
Obtaining $e_{q_1}(k)$ and updating $\boldsymbol{d}_{q_2}(k)$:
$\begin{bmatrix} e_{q_1}(k) \\ \boldsymbol{d}_{q_2}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k-1) \end{bmatrix}$
Obtaining $e(k)$:
$e(k) = e_{q_1}(k)\gamma(k)$
}

4.1.2 Interpreting the Internal Variables

Examining the structure of $\boldsymbol{Q}_\theta(k)$ as expressed in (4.18) and recalling (4.11) and the fact that this matrix is orthonormal, we can write the next two equations.

$$\begin{bmatrix} \gamma(k) & \boldsymbol{g}^T(k) \\ \boldsymbol{f}(k) & \boldsymbol{E}(k) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{U}(k-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}^T \\ \boldsymbol{U}(k) \end{bmatrix} \tag{4.21}$$

$$\begin{aligned} \boldsymbol{I}_{N+2} &= \boldsymbol{Q}_\theta(k)\boldsymbol{Q}_\theta^T(k) = \begin{bmatrix} \gamma(k) & \boldsymbol{g}^T(k) \\ \boldsymbol{f}(k) & \boldsymbol{E}(k) \end{bmatrix} \begin{bmatrix} \gamma(k) & \boldsymbol{f}^T(k) \\ \boldsymbol{g}(k) & \boldsymbol{E}^T(k) \end{bmatrix} \\ &= \boldsymbol{Q}_\theta^T(k)\boldsymbol{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \boldsymbol{f}^T(k) \\ \boldsymbol{g}(k) & \boldsymbol{E}^T(k) \end{bmatrix} \begin{bmatrix} \gamma(k) & \boldsymbol{g}(k)^T \\ \boldsymbol{f}(k) & \boldsymbol{E}(k) \end{bmatrix} \end{aligned} \tag{4.22}$$

From (4.21) and (4.22), a number of relations, which are common to both triangularization methods, can be derived. Let us highlight the following two relations

$$\boldsymbol{f}(k)\boldsymbol{x}^T(k) + \lambda^{1/2}\boldsymbol{E}(k)\boldsymbol{U}(k-1) = \boldsymbol{U}(k) \tag{4.23}$$

$$\gamma(k)\boldsymbol{f}(k) + \boldsymbol{E}(k)\boldsymbol{g}(k) = \boldsymbol{0} \tag{4.24}$$

Now, if we multiply the transpose of (4.11) by itself, we obtain

$$\boldsymbol{U}^T(k)\boldsymbol{U}(k) = \boldsymbol{x}(k)\boldsymbol{x}^T(k) + \lambda\boldsymbol{U}^T(k-1)\boldsymbol{U}(k-1) \tag{4.25}$$

Premultiplying (4.23) by $\boldsymbol{U}^T(k)$ and comparing to (4.25) we find that

$$\boldsymbol{f}(k) = \boldsymbol{U}^{-T}(k)\boldsymbol{x}(k) \tag{4.26}$$

$$\boldsymbol{E}(k) = \lambda^{1/2}\boldsymbol{U}^{-T}(k)\boldsymbol{U}^T(k-1) \tag{4.27}$$

By substituting (4.26) and (4.27) in (4.24), it follows that

$$\boldsymbol{g}(k) = -\gamma(k)\boldsymbol{U}^{-T}(k-1)\boldsymbol{x}(k)/\sqrt{\lambda} \tag{4.28}$$

We will see that (4.26), (4.27) and (4.28) are key relations for the comprehension of other algorithms of the QR family. In order to understand the meaning of these

variables, which depend on the triangularization method, it is necessary to introduce the application of the QR decomposition to the forward and backward prediction problems as well as to compare the QR method discussed so far to the Gram-Schmidt Orthogonalization procedure.

The Backward Prediction Problem

In the backward prediction problem, we try to obtain an estimate of a past sample of a given input sequence using the currently available information of the sequence. The signal $x(K-N-1)$ is the desired signal and the prediction is based on $\boldsymbol{x}(k)$. The weighted backward error vector is

$$\boldsymbol{e}_b(k) = \boldsymbol{d}_b(k) - \boldsymbol{X}(k)\boldsymbol{w}_b(k) = \begin{bmatrix} x(k-N-1) \\ \lambda^{1/2}x(k-N-2) \\ \vdots \\ \lambda^{(k-N-1)/2}x(0) \\ \mathbf{0}_{N+1} \end{bmatrix} - \boldsymbol{X}(k)\boldsymbol{w}_b(k) \tag{4.29}$$

where $\boldsymbol{w}_b(k)$ is the backward prediction coefficient vector.

The last equation can be rewritten in terms of $\boldsymbol{X}^{(N+2)}(k)$, the input data matrix of order $N+1$.

$$\boldsymbol{e}_b(k) = [\boldsymbol{X}(k)\ \boldsymbol{d}_b(k)] \begin{bmatrix} -\boldsymbol{w}_b(k) \\ 1 \end{bmatrix} = \boldsymbol{X}^{(N+2)}(k) \begin{bmatrix} -\boldsymbol{w}_b(k) \\ 1 \end{bmatrix} \tag{4.30}$$

The rotated weighted backward error vector is defined below and it will be used later in the derivation of the fast QR algorithms.

$$\boldsymbol{e}_{b_q}(k) = \boldsymbol{Q}(k)\boldsymbol{e}_b(k) = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{e}_{b_{q_1}}(k) \\ \boldsymbol{U}(k) & \boldsymbol{d}_{b_{q_2}}(k) \end{bmatrix} \begin{bmatrix} -\boldsymbol{w}_b(k) \\ 1 \end{bmatrix} \tag{4.31}$$

The Forward Prediction Problem

In the forward prediction problem, the desired signal is $x(k)$ and the prediction will be carried out with $\boldsymbol{x}(k-1)$. In terms of weighted vectors, we have

$$\boldsymbol{e}_f(k) = \boldsymbol{d}_f(k) - \begin{bmatrix} \boldsymbol{X}(k-1) \\ \boldsymbol{0}^T \end{bmatrix} \boldsymbol{w}_f(k) = \begin{bmatrix} x(k) \\ \lambda^{1/2}x(k-1) \\ \vdots \\ \lambda^{k/2}x(0) \end{bmatrix} - \begin{bmatrix} \boldsymbol{X}(k-1) \\ \boldsymbol{0}^T \end{bmatrix} \boldsymbol{w}_f(k) \tag{4.32}$$

where $\boldsymbol{w}_f(k)$ is the forward prediction coefficient vector.

The last equation can also be rewritten in terms of $\boldsymbol{X}^{(N+2)}(k)$, the input data matrix of order $N+1$.

$$\boldsymbol{e}_f(k) = \begin{bmatrix} \boldsymbol{d}_f(k) & \boldsymbol{X}(k-1) \\ & \boldsymbol{0}^T \end{bmatrix} \begin{bmatrix} 1 \\ -\boldsymbol{w}_f(k) \end{bmatrix} = \boldsymbol{X}^{(N+2)}(k) \begin{bmatrix} 1 \\ -\boldsymbol{w}_f(k) \end{bmatrix} \tag{4.33}$$

The rotated weighted forward error vector is defined as

$$\boldsymbol{e}_{f_q}(k) = \begin{bmatrix} \boldsymbol{Q}(k-1) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \boldsymbol{e}_f(k) = \begin{bmatrix} \boldsymbol{e}_{f_{q_1}}(k) & \boldsymbol{O} \\ \boldsymbol{d}_{f_{q_2}}(k) & \boldsymbol{U}(k-1) \\ \lambda^{k/2}x(0) & \boldsymbol{0}^T \end{bmatrix} \begin{bmatrix} 1 \\ -\boldsymbol{w}_f(k) \end{bmatrix} \tag{4.34}$$

It is worth mentioning that all variables of the backward and forward predictors are related to order N predictors ($N+1$ prediction coefficients). As such $\boldsymbol{e}_b(k) = \boldsymbol{e}_b^{(N+1)}(k)$ and $\boldsymbol{e}_f(k) = \boldsymbol{e}_f^{(N+1)}(k)$.

Gram-Schmidt Orthogonalization for Lower Triangular $\boldsymbol{U}(k)$

The Gram-Schmidt technique searches a set of orthogonal vectors $\{\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_N\}$ spanning the same space as another set of vectors $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ which are not

mutually orthogonal. This is usually accomplished by making $\boldsymbol{e}_0 = \boldsymbol{x}_0$, $\boldsymbol{e}_1 = \boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1$ (where $\hat{\boldsymbol{x}}_1$ is the projection of $\boldsymbol{x}_1$ in $\boldsymbol{e}_0$) and so on, such that $\boldsymbol{e}_i = \boldsymbol{x}_i - \hat{\boldsymbol{x}}_i$, $\hat{\boldsymbol{x}}_i = \sum_{j=0}^{i-1} k_{ji}\boldsymbol{e}_j$ with $k_{ji} = \boldsymbol{e}_j^T\boldsymbol{x}_i / \parallel \boldsymbol{e}_j \parallel^2$, $j < i$. After finding $\boldsymbol{e}_N$, the orthonormality is forced by replacing $\boldsymbol{e}_i$ by $\boldsymbol{e}_i / \parallel \boldsymbol{e}_i \parallel$. This procedure triangularizes a matrix consisting of vectors $\boldsymbol{x}_i$ as its columns. The result is an upper triangular matrix.

We will choose another set of orthogonal vectors such that matrix $\boldsymbol{X}(k)$ (rewritten below) is correctly triangularized.

$$\begin{aligned}\boldsymbol{X}(k) &= [\boldsymbol{x}_0\ \boldsymbol{x}_1 \cdots \boldsymbol{x}_N] = \begin{bmatrix} \boldsymbol{x}^T(k) \\ \lambda^{1/2}\boldsymbol{x}^T(k-1) \\ \vdots \\ \lambda^{k/2}\boldsymbol{x}^T(0) \end{bmatrix} \\ &= \begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-N) \\ \lambda^{1/2}x(k-1) & \lambda^{1/2}x(k-2) & & \vdots \\ & & & \lambda^{(k-N)/2}x(0) \\ \vdots & \vdots & & 0 \\ \lambda^{(k-1)/2}x(1) & \lambda^{(k-1)/2}x(0) & & \vdots \\ \lambda^{k/2}x(0) & 0 & \cdots & 0 \end{bmatrix}\end{aligned} \tag{4.35}$$

By making $\boldsymbol{e}_N = \boldsymbol{x}_0, \boldsymbol{e}_{N-1} = \boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1, \ldots, \boldsymbol{e}_0 = \boldsymbol{x}_N - \hat{\boldsymbol{x}}_N$, we have $\boldsymbol{x}_0 = \boldsymbol{e}_N, \boldsymbol{x}_1 = \boldsymbol{e}_{N-1} + k_{N1}\boldsymbol{e}_N, \ldots, \boldsymbol{x}_N = \boldsymbol{e}_0 + k_{1N}\boldsymbol{e}_1 + \cdots + k_{NN}\boldsymbol{e}_N$, which means that

$$\boldsymbol{X}(k) = [\boldsymbol{e}_0\ \boldsymbol{e}_1 \cdots \boldsymbol{e}_N] \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & & k_{1N} \\ 0 & & & \vdots \\ 1 & k_{N1} & \cdots & k_{NN} \end{bmatrix} = \boldsymbol{G}(k)\boldsymbol{K}(k) \tag{4.36}$$

Defining $\boldsymbol{D}^2(k) = \boldsymbol{G}^T(k)\boldsymbol{G}(k)$ and being this product a diagonal matrix whose elements are $\parallel \boldsymbol{e}_i \parallel^2$, $\boldsymbol{D}(k)$ is a diagonal matrix whose elements are $\parallel \boldsymbol{e}_i \parallel$. We can, then, write

$$\boldsymbol{U}^T(k)\boldsymbol{U}(k) = \boldsymbol{X}^T(k)\boldsymbol{X}(k) = \boldsymbol{K}^T(k)\boldsymbol{G}^T(k)\boldsymbol{G}(k)\boldsymbol{K}(k) = [\boldsymbol{D}(k)\boldsymbol{K}(k)]^T[\boldsymbol{D}(k)\boldsymbol{K}(k)] \tag{4.37}$$

Once $\boldsymbol{U}(k) = \boldsymbol{D}(k)\boldsymbol{K}(k)$ [37] and using (4.18), (4.26), (4.27) and (4.28), we have the following expression for $\boldsymbol{Q}_\theta(k)$

$$\boldsymbol{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & -\gamma(k)\lambda^{-1/2}\left[\boldsymbol{D}^{-1}(k-1)\boldsymbol{K}^{-T}(k-1)\boldsymbol{x}(k)\right]^T \\ \boldsymbol{D}^{-1}(k)\boldsymbol{K}^{-T}(k)\boldsymbol{x}(k) & \lambda^{1/2}\boldsymbol{D}^{-1}(k)\boldsymbol{K}^{-T}(k)\boldsymbol{K}^T(k-1)\boldsymbol{D}^T(k-1) \end{bmatrix} \tag{4.38}$$

It is possible to find a physical meaning to the expression $\boldsymbol{K}^{-T}(k)\boldsymbol{x}(k)$ if (4.36) is first rewritten as

$$\boldsymbol{G}^T(k) = \boldsymbol{K}^{-T}(k)\boldsymbol{X}^T(k) = \boldsymbol{K}^{-T}(k)\left[\boldsymbol{x}(k)\ \lambda^{1/2}\boldsymbol{x}(k-1)\ \cdots\right] = \begin{bmatrix} \boldsymbol{e}_0^T \\ \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_N^T \end{bmatrix} \tag{4.39}$$

From the backward prediction problem (4.29), we know that

$$\boldsymbol{e}_b^{(i)}(k) = \boldsymbol{d}_b^{(i)}(k) - \boldsymbol{X}^{(i)}(k)\boldsymbol{w}_b^{(i)}(k) \tag{4.40}$$

where $\boldsymbol{d}_b^{(i)}(k)$ is the weighted backward desired signal vector of order $i-1$ or $\left[x(k-i)\ \ \lambda^{1/2}x(k-i-1)\ \ \cdots\ \ \lambda^{(k-i)/2}x(0)\ \ \boldsymbol{0}_i^T\right]^T$.

By differentiating $\boldsymbol{e}_b^{(i)^T}(k)\boldsymbol{e}_b^{(i)}(k)$ with respect to $\boldsymbol{w}_b^{(i)}(k)$, it is straightforward to show that the optimum backward prediction coefficients vector is given by

$$\boldsymbol{w}_b^{(i)}(k) = \left[\boldsymbol{X}^{(i)^T}(k)\boldsymbol{X}^{(i)}(k)\right]^{-1}\boldsymbol{X}^{(i)^T}(k)\boldsymbol{d}_b^{(i)}(k) \tag{4.41}$$

If we recognize $\boldsymbol{d}_b^{(i)}(k)$ from (4.29) as $\boldsymbol{x}_i$ in (4.35) and substitute first (4.41) in (4.40) and then $\boldsymbol{X}^{(i)}(k)$ by $\boldsymbol{G}^{(i)}(k)\boldsymbol{K}^{(i)}(k)$ (of order $i-1$), we obtain, after some manipulations, the expression

$$\boldsymbol{e}_b^{(i)}(k) = \boldsymbol{x}_i - \sum_{j=0}^{i-1} \frac{\boldsymbol{e}_j \boldsymbol{e}_j^T \boldsymbol{x}_i}{\| \boldsymbol{e}_j \|^2} \tag{4.42}$$

which corresponds to $\boldsymbol{e}_{N-i} = \boldsymbol{x}_i - \hat{\boldsymbol{x}}_i$, i.e., one of the vectors of the new base shown in (4.36).

Once we know that $\boldsymbol{e}_i$ is equal to $\boldsymbol{e}_b^{(N-i)}(k)$, these values can be used in (4.39) and it follows that

$$\boldsymbol{K}^{-T}(k)\boldsymbol{x}(k) = \begin{bmatrix} e_b^{(N)}(k) \\ e_b^{(N-1)}(k) \\ \vdots \\ e_b^{(0)}(k) \end{bmatrix} \tag{4.43}$$

where the above product is the *a posteriori* backward prediction error (with different number of coefficients) vector at instant k.

It is worth mentioning that (4.43) brings an interpretation of the non-zero elements of the rows of $\boldsymbol{K}^{-T}(k)$ as the coefficients of backward prediction filters of different orders. If we now recall that the elements of the diagonal matrix $\boldsymbol{D}(k)$ are given by $\| \boldsymbol{e}_i(k) \| = \| \boldsymbol{e}_b^{(N-i)}(k) \|$, the vector $\boldsymbol{f}(k)$ of (4.18) as seen in (4.38) is given by

$$\boldsymbol{f}(k) = \begin{bmatrix} e_b^{(N)}(k) / \| \boldsymbol{e}_b^{(N)}(k) \| \\ e_b^{(N-1)}(k) / \| \boldsymbol{e}_b^{(N-1)}(k) \| \\ \vdots \\ e_b^{(0)}(k) / \| \boldsymbol{e}_b^{(0)}(k) \| \end{bmatrix} \tag{4.44}$$

and will be referred as the normalized *a posteriori* backward prediction error vector at instant k.

Moreover, using the same interpretation of $\boldsymbol{D}(k-1)$ and $\boldsymbol{K}^{-T}(k-1)$, the vector $\boldsymbol{g}(k)$ of (4.18) can be shown to correspond to

$$\boldsymbol{g}(k) = -\gamma(k)\boldsymbol{a}(k) = -\gamma(k)\lambda^{-1/2} \begin{bmatrix} e'^{(N)}_b(k)/ \parallel \boldsymbol{e}^{(N)}_b(k-1) \parallel \\ e'^{(N-1)}_b(k)/ \parallel \boldsymbol{e}^{(N-1)}_b(k-1) \parallel \\ \vdots \\ e'^{(0)}_b(k)/ \parallel \boldsymbol{e}^{(0)}_b(k-1) \parallel \end{bmatrix} \tag{4.45}$$

where $\boldsymbol{a}(k)$ is the *a priori* backward prediction error vector at instant k normalized by the *a posteriori* backward error energies at instant $k-1$ and weighted by $\lambda^{-1/2}$.

The expression for $\boldsymbol{E}(k)$ given in (4.38), however, does not lead to a relevant physical interpretation and (4.27) remains the best representation for $\boldsymbol{E}(k)$. Another alternative representation to $\boldsymbol{E}(k)$ (yet with no physical meaning) is $\boldsymbol{A}^{-T}(k)$ where $\boldsymbol{A}(k)$ is the Cholesky factor of $\boldsymbol{I} + \boldsymbol{a}(k)\boldsymbol{a}^T(k)$ [38].

Gram-Schmidt Orthogonalization for Upper Triangular $\boldsymbol{U}(k)$

The informations given in (4.44) and (4.45) are exclusively valid for the lower triangularization of $\boldsymbol{U}(k)$. For the upper triangularization of matrix $\boldsymbol{U}(k)$ we shall choose a different set of orthogonal vectors $\{\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_N\}$ such that $\boldsymbol{e}_0 = \boldsymbol{x}_N, \boldsymbol{e}_1 = \boldsymbol{x}_{N-1} - \hat{\boldsymbol{x}}_{N-1}, \ldots, \boldsymbol{e}_N = \boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0$. In this case, we have $\boldsymbol{x}_N = \boldsymbol{e}_0, \boldsymbol{x}_{N-1} = \boldsymbol{e}_1 + k_{0\,N-1}\boldsymbol{e}_0, \ldots, \boldsymbol{x}_0 = \boldsymbol{e}_N + k_{00}\boldsymbol{e}_0 + k_{10}\boldsymbol{e}_1 + \cdots + k_{N-1\,0}\boldsymbol{e}_{N-1}$, which means that

$$\boldsymbol{X}(k) = [\boldsymbol{e}_0\ \boldsymbol{e}_1 \cdots \boldsymbol{e}_N] \begin{bmatrix} k_{00} & \cdots & k_{0\,N-1} & 1 \\ \vdots & & & 0 \\ & & & \vdots \\ k_{N-1\,0} & & & \\ 1 & 0 & \cdots & \end{bmatrix} = \boldsymbol{G}(k)\boldsymbol{K}(k) \tag{4.46}$$

Equations (4.37) to (4.39) still hold in this upper triangularization case. From the forward prediction problem (4.32), we have

$$\boldsymbol{e}^{(i)}_f(k) = \boldsymbol{d}^{(i)}_f(k) - \begin{bmatrix} \boldsymbol{X}^{(i)}(k-1) \\ \boldsymbol{0}^T \end{bmatrix} \boldsymbol{w}^{(i)}_f(k) \tag{4.47}$$

where $\boldsymbol{d}_f^{(i)}(k-i) = \boldsymbol{d}_f(k-i)$ and $\boldsymbol{d}_f(k-i) = \boldsymbol{x}_i$ from (4.35).

By differentiating $[\boldsymbol{e}_f^{(i)}(k)]^T \boldsymbol{e}_f^{(i)}(k)$ with respect to $\boldsymbol{w}_f^{(i)}(k)$ and equating the result to zero, we find the optimum forward prediction coefficient vector of order $(i-1)$ and at instant k which is given by

$$\boldsymbol{w}_f^{(i)}(k) = \left\{ [\boldsymbol{X}^{(i)}(k-1)]^T \boldsymbol{X}^{(i)}(k-1) \right\}^{-1} \begin{bmatrix} \boldsymbol{X}^{(i)}(k-1) \\ \boldsymbol{0}^T \end{bmatrix}^T \boldsymbol{d}_f^{(i)}(k) \tag{4.48}$$

By substituting this equation in (4.47) and making $\boldsymbol{X}^{(i)}(k-1) = \boldsymbol{G}^{(i)}(k-1)\boldsymbol{K}^{(i)}(k-1)$, we have

$$\boldsymbol{e}_f^{(i)}(k) = \boldsymbol{d}_f^{(i)}(k) - \begin{bmatrix} \sum_{j=0}^{i-1} \boldsymbol{e}_j^{(i)}(k-1)[\boldsymbol{e}_j^{(i)}(k-1)]^T & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix} \boldsymbol{d}_f^{(i)}(k) \tag{4.49}$$

from which it is possible to obtain the vectors of $\boldsymbol{G}(k)$ in (4.46) which are given by

$$\boldsymbol{e}_i = \begin{bmatrix} \boldsymbol{e}_f^{(i)}(k-N+i) \\ \boldsymbol{0}_{N-i} \end{bmatrix} \tag{4.50}$$

This expression for $\boldsymbol{e}_i$ can then be used in (4.39) for the upper triangularization case and it follows that

$$\boldsymbol{K}^{-T}(k)\boldsymbol{x}(k) = \begin{bmatrix} e_f^{(0)}(k-N) \\ e_f^{(1)}(k-N+1) \\ \vdots \\ e_f^{(N)}(k) \end{bmatrix} \tag{4.51}$$

where the above product is the *a posteriori* forward prediction errors (with different orders and at distinct instants of time) vector.

It is also worth mentioning that (4.51) brings an interpretation of the non-zero elements of the rows of $\boldsymbol{K}^{-T}(k)$ as the coefficients of forward prediction filters of different orders at distinct instants of time. Recalling that the elements of the

diagonal matrix $\boldsymbol{D}(k)$ are given by $\| \boldsymbol{e}_i(k) \| = \| \boldsymbol{e}_f^{(i)}(k-N+i) \|$, vector $\boldsymbol{f}(k)$ is now given by

$$\boldsymbol{f}(k) = \begin{bmatrix} e_f^{(0)}(k-N) / \| \boldsymbol{e}_f^{(0)}(k-N) \| \\ e_f^{(1)}(k-N+1) / \| \boldsymbol{e}_f^{(1)}(k-N+1) \| \\ \vdots \\ e_f^{(N)}(k) / \| \boldsymbol{e}_f^{(N)}(k) \| \end{bmatrix} \tag{4.52}$$

and will be referred as the normalized *a posteriori* forward prediction error vector.

By using the same interpretation of $\boldsymbol{D}(k-1)$ and $\boldsymbol{K}^{-T}(k-1)$, the vector $\boldsymbol{g}(k)$ corresponds in the upper triangularization case to

$$\boldsymbol{g}(k) = -\gamma(k)\boldsymbol{a}(k) = -\gamma(k)\lambda^{-1/2} \begin{bmatrix} e'^{(0)}_f(k-N) / \| \boldsymbol{e}_f^{(0)}(k-N-1) \| \\ e'^{(1)}_f(k-N+1) / \| \boldsymbol{e}_f^{(1)}(k-N) \| \\ \vdots \\ e'^{(N)}_f(k) / \| \boldsymbol{e}_f^{(N)}(k-1) \| \end{bmatrix} \tag{4.53}$$

where $\boldsymbol{a}(k)$ in this case is the *a priori* forward prediction error vector normalized by the *a posteriori* forward error energies and weighted by $\lambda^{-1/2}$.

We saw that in the case of upper triangularization, the normalized errors present in $\boldsymbol{Q}_\theta(k)$ are of different orders at distinct instants of time (order and time updating) and this fact seems to be the cause of the extra computational effort of the fast algorithms derived by using this type of triangularization.

4.1.3 The Inverse QR Algorithm

An alternative approach based on the update of the inverse Cholesky factor was presented in [15]. This algorithm known as inverse QR (IQR) allows the calculation of the weight vector without backsubstitution and some of its relations will be used later in this work.

Starting from (4.6) and using $\left[\boldsymbol{x}(k)\ \ \lambda^{1/2}\boldsymbol{X}^T(k-1)\right]^T$ instead of $\boldsymbol{X}(k)$ on the definition of $\boldsymbol{R}_D(k)$ and $\boldsymbol{p}_D(k)$, after some manipulations, we can show that

$$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) + \boldsymbol{U}^{-1}(k)\boldsymbol{U}^{-T}(k)\boldsymbol{x}(k)e'(k) \tag{4.54}$$

where $e'(k)$ is the *a priori* error or $d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k-1)$ and the term multiplying this variable is known as the *Kalman Gain*.

The product $\boldsymbol{U}^{-1}(k)\boldsymbol{U}^{-T}(k)$ in (4.54) can be found from previous quantities if we invert (4.25) and use the so called matrix inversion lemma [2]. After some algebraic operations, the result is

$$\begin{aligned}\boldsymbol{U}^{-1}(k)\boldsymbol{U}^{-T}(k) &= \lambda^{-1}\boldsymbol{U}^{-1}(k-1)\boldsymbol{U}^{-T}(k-1) - \\ &\quad -\lambda^{-1}\gamma^2(k)\boldsymbol{U}^{-1}(k-1)\boldsymbol{a}(k)\boldsymbol{a}^T(k)\boldsymbol{U}^{-T}(k-1)\end{aligned} \tag{4.55}$$

Defining the vector $\boldsymbol{u}(k)$ as

$$\boldsymbol{u}(k) = -\lambda^{-1/2}\gamma(k)\boldsymbol{U}^{-1}(k-1)\boldsymbol{a}(k) \tag{4.56}$$

we can rewrite (4.54) and (4.55) as

$$\boldsymbol{U}^{-1}(k)\boldsymbol{U}^{-T}(k) + \boldsymbol{u}(k)\boldsymbol{u}^T(k) = \lambda^{-1}\boldsymbol{U}^{-1}(k-1)\boldsymbol{U}^{-T}(k-1) \tag{4.57}$$

$$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \gamma(k)\boldsymbol{u}(k)e'(k) \tag{4.58}$$

where the Kalman vector is now expressed as $-\gamma(k)\boldsymbol{u}(k)$.

It was noted in [15] that (4.57) implies the existence of an orthogonal matrix $\boldsymbol{Q}_I(k)$ such that

$$\boldsymbol{Q}_I(k)\begin{bmatrix}\boldsymbol{0}^T \\ \lambda^{-1/2}\boldsymbol{U}^{-T}(k-1)\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}^T(k) \\ \boldsymbol{U}^{-T}(k)\end{bmatrix} \tag{4.59}$$

This can be verified by taking the transpose of (4.59) and multiplying by itself, resulting in equation (4.57). Fortunately, $\boldsymbol{Q}_I(k)$ is already known. By admitting a partition of $\boldsymbol{Q}_I(k)$ similar to the one used in (4.18) and imposing orthonormality, we can see that (4.59) yields $\boldsymbol{Q}_I(k) = \boldsymbol{Q}_\theta(k)$.

Since we know that $\boldsymbol{Q}_\theta(k)$ is orthonormal, if we postmultiply this matrix by its first row transposed, we shall have

$$\boldsymbol{Q}_\theta(k) \begin{bmatrix} \gamma(k) \\ -\gamma(k)\boldsymbol{a}(k) \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \tag{4.60}$$

By combining (4.59) (with $\boldsymbol{Q}_I(k) = \boldsymbol{Q}_\theta(k)$) and (4.60) (after dividing both terms by $\gamma(k)$) in one only equation, we have

$$\begin{bmatrix} 1/\gamma(k) & \boldsymbol{u}^T(k) \\ \mathbf{0} & \boldsymbol{U}^{-T}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} 1 & \mathbf{0}^T \\ -\boldsymbol{a}(k) & \lambda^{-1/2}\boldsymbol{U}^{-T}(k-1) \end{bmatrix} \tag{4.61}$$

Equation (4.61) is a key relation to the inverse QR algorithm. In order to have all the necessary equations, let us now analyze the relation between the *a posteriori* and the *a priori* errors. Replacing $\boldsymbol{Q}_\theta(k)$ by its partition (given in (4.18), (4.26), (4.27) and (4.28)) in (4.13), it follows that

$$e(k) = \gamma(k)e_{q_1}(k) = \gamma^2(k)e'(k) \tag{4.62}$$

By following similar procedures in the backward and forward prediction problems, it is possible to show that

$$e_b(k) = \gamma(k)e_{b_{q_1}}(k) = \gamma^2(k)e'_b(k) \tag{4.63}$$

$$e_f(k) = \gamma(k-1)e_{f_{q_1}}(k) = \gamma^2(k-1)e'_f(k) \tag{4.64}$$

The equations of the inverse QR are presented in Table 4.2 while the detailed algorithm description can be found in [15].

Table 4.2: The inverse QR equations.

IQR
for each k
{ Obtaining $\boldsymbol{a}(k)$:
$\boldsymbol{a}(k) = \lambda^{-1/2}\boldsymbol{U}^{-T}(k-1)\boldsymbol{x}(k)$
Obtaining $\boldsymbol{Q}_\theta(k)$ and $\gamma(k)$:
$\begin{bmatrix} 1/\gamma(k) \\ \mathbf{0} \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} 1 \\ -\boldsymbol{a}(k) \end{bmatrix}$
Obtaining $\boldsymbol{u}(k)$ and updating $\boldsymbol{U}^{-T}(k)$:
$\begin{bmatrix} \boldsymbol{u}^T(k) \\ \boldsymbol{U}^{-T}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} \mathbf{0}^T \\ \lambda^{-1/2}\boldsymbol{U}^{-T}(k-1) \end{bmatrix}$
Obtaining $e(k)$:
$e(k) = \left[d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(k-1)\right]\gamma^2(k)$
Updating the coefficient vector:
$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \boldsymbol{u}(k)e(k)$
}

4.2 Classification of the Fast QR Algorithms

From the conventional (O[N^2]) QR decomposition method [1, 2] a number of fast algorithms (O[N]) were derived [16]–[19]. These algorithms can be classified in terms of the type of triangularization applied to the input data matrix (upper or lower triangular) and type of error vector (*a posteriori* or *a priori*) involved in the updating process. It was clear from the Gram-Schmidt orthogonalization procedure that an upper triangularization (in the notation adopted in this work) involves the updating of forward prediction errors while a lower triangularization involves the updating of backward prediction errors. This section presents the equations of these algorithms as well as a new one, referred as FQR_PRI_F, which is a fast QR using upper triangularization (of the Cholesky factor of the data correlation matrix) and

updating *a priori* forward errors (vector $\boldsymbol{a}(k)$). Table 4.3 presents the classification and introduces how these algorithms will be designated hereafter.

Table 4.3: Classification of the fast QR algorithms.

Error Type	Prediction	
	Forward	Backward
A Posteriori	FQR_POS_F [16]	FQR_POS_B [17]
A Priori	FQR_PRI_F new [20]	FQR_PRI_B [19, 18]

It is worth mentioning that the FQR_PRI_B algorithm was independently developed in [18] and [19] using different approaches. The approach which will be used here was derived [19] from concepts used in the inverse QR algorithm [15]. The same algorithm was also derived in [39] as a lattice extension of the inverse QR algorithm [40].

In the derivation of fast QR algorithms, we start by applying the QR decomposition to the backward and forward prediction problems whose prediction errors were defined in (4.29) and (4.32). Our aim is to triangularize $\boldsymbol{X}^{(N+2)}(k)$ from equations (4.30) and (4.33) in order to obtain $\boldsymbol{Q}^{(N+2)}(k)$ such that

$$\boldsymbol{Q}^{(N+2)}(k)\boldsymbol{X}^{(N+2)}(k) = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{U}^{(N+2)}(k) \end{bmatrix} \tag{4.65}$$

4.3 Upper Triangularization Algorithms (Updating Forward Prediction Errors)

We will derive here the FQR_POS_F algorithm [16] and the new FQR_PRI_F algorithm. Initially, if we premultiply the backward weighted desired vector $\boldsymbol{d}_b(k)$ defined in (4.29) by $\boldsymbol{Q}(k)$ and use (4.10), two important relations will follow

$$\| \boldsymbol{e}_b(k) \|^2 = e_{bq_1}^2(k) + \lambda \| \boldsymbol{e}_b(k-1) \|^2 \tag{4.66}$$

$$\begin{bmatrix} e_{bq_1}(k) \\ \boldsymbol{d}_{bq_2}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} d_b(k) \\ \lambda^{1/2} \boldsymbol{d}_{bq_2}(k-1) \end{bmatrix} \tag{4.67}$$

where $d_b(k) = x(k-N-1)$.

In the backward prediction problem, the triangularization as seen in (4.65) is achieved using three matrices, $\boldsymbol{Q}^{(N+2)}(k) = \boldsymbol{Q}_b'(k)\boldsymbol{Q}_b(k)\boldsymbol{Q}(k)$, where $\boldsymbol{Q}_b(k)$ and $\boldsymbol{Q}_b'(k)$ are two sets of Givens rotations applied to generate, respectively, $\| \boldsymbol{e}_b(k) \|$ and $\| \boldsymbol{e}_b^{(0)}(k) \|$. As a result we have

$$\begin{aligned} \boldsymbol{U}^{(N+2)}(k) &= \boldsymbol{Q}_{\theta b}'(k) \begin{bmatrix} \boldsymbol{0}^T & \| \boldsymbol{e}_b(k) \| \\ \boldsymbol{U}(k) & \boldsymbol{d}_{bq_2}(k) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{z}^T(k) & \| \boldsymbol{e}_b^{(0)}(k) \| \\ \boldsymbol{R}(k) & \boldsymbol{0} \end{bmatrix} \end{aligned} \tag{4.68}$$

where $\boldsymbol{Q}_{\theta b}'(k)$ is a submatrix of $\boldsymbol{Q}_b'(k)$, $[\boldsymbol{z}(k)\boldsymbol{R}^T(k)]^T$ is the left part of $\boldsymbol{U}^{(N+2)}(k)$ and $\| \boldsymbol{e}_b^{(0)}(k) \|$ is the norm of the backward error of a zero-coefficient predictor.

In the forward prediction problem, the premultiplication of the forward weighted desired vector, $\boldsymbol{d}_f(k)$ as defined in (4.32), by $\begin{bmatrix} \boldsymbol{Q}(k-1) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$ and the use of (4.10) will lead to two other important relations given by

$$\| \boldsymbol{e}_f(k) \|^2 = e_{fq_1}^2(k) + \lambda \| \boldsymbol{e}_f(k-1) \|^2 \tag{4.69}$$

$$\begin{bmatrix} e_{fq_1}(k) \\ \boldsymbol{d}_{fq_2}(k) \end{bmatrix} = \boldsymbol{Q}_\theta(k-1) \begin{bmatrix} d_f(k) \\ \lambda^{1/2} \boldsymbol{d}_{fq_2}(k-1) \end{bmatrix} \tag{4.70}$$

where $d_f(k) = x(k)$.

The upper triangularization of $\boldsymbol{U}^{(N+2)}(k)$ in the forward prediction problem is implemented by premultiplying $\boldsymbol{e}_f(k)$ by the product $\boldsymbol{Q}_f(k) \begin{bmatrix} \boldsymbol{Q}(k-1) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$,

where $\boldsymbol{Q}_f(k)$ is a set of Givens rotations generating $\| \boldsymbol{e}_f(k) \|$ by eliminating the first $k-N$ elements of the rotated desired vector of the forward predictor. The result is

$$\boldsymbol{U}^{(N+2)}(k) = \begin{bmatrix} \boldsymbol{d}_{fq_2}(k) & \boldsymbol{U}(k-1) \\ \| \boldsymbol{e}_f(k) \| & \boldsymbol{0}^T \end{bmatrix} \tag{4.71}$$

Working with nonincreasing dimensions, it is easy to show that [2]

$$\boldsymbol{Q}_\theta^{(N+2)}(k) = \boldsymbol{Q}_{\theta f}(k) \begin{bmatrix} \boldsymbol{Q}_\theta(k-1) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \tag{4.72}$$

where $\boldsymbol{Q}_{\theta f}(k)$ is a single Given rotation generating $\| \boldsymbol{e}_f(k) \|$ as in (4.69).

If we take the inverses of (4.68) and (4.71), the results are

$$\begin{aligned} [\boldsymbol{U}^{(N+2)}(k)]^{-1} &= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{R}^{-1}(k) \\ \frac{1}{\|\boldsymbol{e}_b^{(0)}(k)\|} & \frac{-\boldsymbol{z}^T(k)\boldsymbol{R}^{-1}(k)}{\|\boldsymbol{e}_b^{(0)}(k)\|} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{0}^T & \frac{1}{\|\boldsymbol{e}_f(k)\|} \\ \boldsymbol{U}^{-1}(k-1) & \frac{-\boldsymbol{U}^{-1}(k-1)\boldsymbol{d}_{fq_2}(k)}{\|\boldsymbol{e}_f(k)\|} \end{bmatrix} \end{aligned} \tag{4.73}$$

We can use the expressions of $[\boldsymbol{U}^{(N+2)}(k)]^{-1}$ given in (4.73) to obtain the vectors $\boldsymbol{f}^{(N+2)}(k+1)$ and $\boldsymbol{a}^{(N+2)}(k+1)$. The choice of one of these vectors will determine the algorithm: updating $\boldsymbol{f}(k)$ (*a posteriori* forward errors) will lead to the FQR_POS_F algorithm [16] and updating $\boldsymbol{a}(k)$ (*a priori* forward errors) will lead to the new FQR_PRI_F algorithm [20].

4.3.1 The FQR_POS_F Algorithm

In the FQR_POS_F algorithm, the vector $\boldsymbol{f}^{(N+2)}(k+1) = [\boldsymbol{U}^{(N+2)}(k+1)]^{-T}\boldsymbol{x}^{(N+2)}(k+1)$ is expressed in terms of the relations obtained in the forward and backward prediction problems. We shall first use the expression for $[\boldsymbol{U}^{(N+2)}(k)]^{-1}$ in (4.73) that comes from the backward prediction evaluated at instant $k+1$ to calculate

$\boldsymbol{f}^{(N+2)}(k+1)$. In this case we replace $\boldsymbol{x}^{(N+2)}(k+1)$ by $[\boldsymbol{x}^T(k+1)\ \ x(k-N)]^T$ and then premultiply the result by $\boldsymbol{Q}'^T_{\theta b}(k+1)$. The final outcome after some algebraic manipulation (using equation (4.68) to help with the simplification of the expression) is

$$\boldsymbol{f}^{(N+2)}(k+1) = \boldsymbol{Q}'_{\theta b}(k+1) \begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1) \end{bmatrix} \tag{4.74}$$

Using the other expression for $[\boldsymbol{U}^{(N+2)}(k)]^{-1}$ (from the forward prediction) at instant $k+1$ and replacing $\boldsymbol{x}^{(N+2)}(k+1)$ by $[x(k+1)\ \ \boldsymbol{x}^T(k)]^T$, we have

$$\boldsymbol{f}^{(N+2)}(k+1) = \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix} \tag{4.75}$$

By combining (4.74) and (4.75), we have an expression to update $\boldsymbol{f}(k)$ given by

$$\begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1)] \end{bmatrix} = \boldsymbol{Q}'^T_{\theta b}(k+1) \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix} \tag{4.76}$$

Once we have $\boldsymbol{f}(k+1)$, we can find the angles of $\boldsymbol{Q}_\theta(k+1)$ by postmultiplying this matrix by the pinning vector $[1\ 0\ \cdots\ 0]^T$. From (4.18), we can see that the result is

$$\boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \boldsymbol{f}(k+1) \end{bmatrix} \tag{4.77}$$

However, the quantities required to compute the angles of $\boldsymbol{Q}'_{\theta_b}(k+1)$ are not available at instant k, and a special strategy is required. The updated $\boldsymbol{Q}'_{\theta_b}(k+1)$ is obtained [2, 41] with the use of the vector $\boldsymbol{c}(k+1)$ defined as

$$\begin{aligned} \boldsymbol{c}(k+1) &= \hat{\boldsymbol{Q}}^{(N+2)}_\theta(k+1)\boldsymbol{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} \\ &= \boldsymbol{Q}'_{\theta b}(k+1) \begin{bmatrix} b \\ \boldsymbol{0} \end{bmatrix} \end{aligned} \tag{4.78}$$

The submatrix $\hat{\boldsymbol{Q}}_{\theta}^{(N+2)}(k+1)$ consisting of the last $(N+2)\times(N+2)$ elements of $\boldsymbol{Q}_{\theta}^{(N+2)}(k+1)$ is available from (4.72) (forward prediction) and b does not need to be explicitly calculated in order to obtain the angles θ'_{b_i}.

Finally, the joint process estimation is calculated with (4.13) and (4.20), and the FQR_POS_F equations are presented in Table 4.4. A detailed description of this algorithm is found in Appendix C.

4.3.2 The New FQR_PRI_F Algorithm

Expressing $\boldsymbol{a}^{(N+2)}(k+1) = [\boldsymbol{U}^{(N+2)}(k)]^{-T}\boldsymbol{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the matrices in (4.73) and premultiplying the one that comes from the backward prediction problem by $\boldsymbol{Q}'_{\theta b}(k)\boldsymbol{Q}'^{T}_{\theta b}(k)$ yields

$$\begin{bmatrix} \frac{e'_b(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_b(k)\|} \\ \boldsymbol{a}(k+1) \end{bmatrix} = \boldsymbol{Q}'^{\,T}_{\theta b}(k) \begin{bmatrix} \boldsymbol{a}(k) \\ \frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|} \end{bmatrix} \tag{4.79}$$

Once we have $\boldsymbol{a}(k+1)$, the angles of $\boldsymbol{Q}_{\theta}(k+1)$ are found through the following relation obtained by postmultiplying $\boldsymbol{Q}_{\theta}^{T}(k+1)$ by the pinning vector.

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \boldsymbol{Q}_{\theta}(k+1) \begin{bmatrix} 1 \\ -\boldsymbol{a}(k+1) \end{bmatrix} \tag{4.80}$$

Since the angles of $\boldsymbol{Q}'_{\theta_b}(k+1)$ can be updated with the same procedure used in the FQR_POS_F algorithm, we already have all the necessary equations of the new fast QR-RLS algorithm presented in Table 4.5. The detailed description of this algorithm is found in Appendix C.

4.4 Lower Triangularization Algorithms (Updating Backward Prediction Errors)

Following similar steps as in the upper triangularization, it is possible to obtain the lower triangular matrix $\boldsymbol{U}^{(N+2)}(k)$ from the forward and backward prediction

problems.

In the backward prediction problem, the lower triangular $\boldsymbol{U}^{(N+2)}(k)$ is obtained through the use of $\boldsymbol{Q}^{(N+2)}(k) = \boldsymbol{Q}_b(k)\boldsymbol{Q}(k)$, where $\boldsymbol{Q}_b(k)$ is a set of Givens rotations applied to generate $\| \boldsymbol{e}_b(k) \|$. The resulting Cholesky factor is

$$\boldsymbol{U}^{(N+2)}(k) = \begin{bmatrix} \boldsymbol{0}^T & \| \boldsymbol{e}_b(k) \| \\ \boldsymbol{U}(k) & \boldsymbol{d}_{bq_2}(k) \end{bmatrix} \tag{4.81}$$

On the other hand, in the forward prediction problem, the lower triangular of $\boldsymbol{U}^{(N+2)}(k)$ is implemented by premultiplying $\boldsymbol{e}_f(k)$ by the product $\boldsymbol{Q}'_f(k)\boldsymbol{Q}_f(k) \begin{bmatrix} \boldsymbol{Q}(k-1) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$, where $\boldsymbol{Q}_f(k)$ and $\boldsymbol{Q}'_f(k)$ are two sets of Givens rotations generating $\| \boldsymbol{e}_f(k) \|$ and $\| \boldsymbol{e}_f^{(0)}(k) \|$, respectively. The resulting expression is

$$\boldsymbol{U}^{(N+2)}(k) = \boldsymbol{Q}'_{\theta f}(k) \begin{bmatrix} \boldsymbol{d}_{fq_2}(k) & \boldsymbol{U}(k-1) \\ \| \boldsymbol{e}_f(k) \| & \boldsymbol{0}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{R}(k) \\ \| \boldsymbol{e}_f^{(0)}(k) \| & \boldsymbol{z}^T(k) \end{bmatrix} \tag{4.82}$$

where $[\boldsymbol{R}^T(k) \ \ \boldsymbol{z}(k)]^T$ is the right part of $\boldsymbol{U}^{(N+2)}(k)$. Keeping in mind that (4.66), (4.67), (4.69) and (4.70) hold, $\| \boldsymbol{e}_f(k) \|$ can be recursively computed using (4.69).

Taking the inverse of (4.81) and (4.82) we have the following results

$$\begin{aligned} [\boldsymbol{U}^{(N+2)}(k)]^{-1} &= \begin{bmatrix} \frac{-\boldsymbol{U}^{-1}(k)\boldsymbol{d}_{bq_2}(k)}{\|\boldsymbol{e}_b(k)\|} & \boldsymbol{U}^{-1}(k) \\ \frac{1}{\|\boldsymbol{e}_b(k)\|} & \boldsymbol{0}^T \end{bmatrix} \\ &= \begin{bmatrix} \frac{-\boldsymbol{z}^T(k)\boldsymbol{R}^{-1}(k)}{\|\boldsymbol{e}_f^{(0)}(k)\|} & \frac{1}{\|\boldsymbol{e}_f^{(0)}(k)\|} \\ \boldsymbol{R}^{-1}(k) & \boldsymbol{0} \end{bmatrix} \end{aligned} \tag{4.83}$$

With the results obtained in (4.83), we can once more express the vectors $\boldsymbol{f}^{(N+2)}(k+1)$ and $\boldsymbol{a}^{(N+2)}(k+1)$ in terms of the partitions of $\left[\boldsymbol{U}^{(N+2)}(k+1)\right]^{-1}$. If we update $\boldsymbol{f}(k)$, the resulting algorithm will be the FQR_POS_B while updating $\boldsymbol{a}(k)$ we will have the FQR_PRI_B algorithm.

4.4.1 The FQR_POS_B Algorithm

Expressing $\boldsymbol{f}^{(N+2)}(k+1) = [\boldsymbol{U}^{(N+2)}(k+1)]^{-T}\boldsymbol{x}^{(N+2)}(k+1)$ in terms of the matrices in (4.83) and premultiplying the one that comes from the forward prediction problem by $\boldsymbol{Q}'_{\theta f}(k+1)\boldsymbol{Q}'^{T}_{\theta f}(k+1)$ yields

$$\begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta_f}(k+1) \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix} \tag{4.84}$$

During the derivation of (4.84), it was observed that the last element of $\boldsymbol{f}(k+1)$ is $\frac{x(k+1)}{\|\boldsymbol{e}_f^{(0)}(k+1)\|}$. The term $\frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$ can be calculated as $\gamma(k)sin\theta_f(k+1)$ where $sin\theta_f(k+1) = \frac{e_{fq_1}(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$ is the sine of the angle of rotation matrix $\boldsymbol{Q}_f(k+1)$.

Once we have $\boldsymbol{f}(k+1)$, we find $\boldsymbol{Q}_\theta(k+1)$ with the same relation used in the upper triangularization algorithms, (4.77). Moreover, the joint process estimation is carried out in the same way and the FQR_POS_B equations are presented in Table 4.6. The detailed descriptions of two slightly different versions of this algorithm is found in Appendix C.

4.4.2 The FQR_PRI_B Algorithm

This last algorithm of this family is obtained by expressing the vector $\boldsymbol{a}^{(N+2)}(k+1)) = [\boldsymbol{U}^{(N+2)}(k)]^{-T}\boldsymbol{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the matrices in (4.83) and premultiplying the one that comes from the forward prediction problem by $\boldsymbol{Q}'_{\theta f}(k)\boldsymbol{Q}'^{T}_{\theta f}(k)$. The updating equation is

$$\begin{bmatrix} \frac{e'_b(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_b(k)\|} \\ \boldsymbol{a}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k) \begin{bmatrix} \boldsymbol{a}(k) \\ \frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|} \end{bmatrix} \tag{4.85}$$

It is again important to mention that, during the derivation, it was observed that the last element of $\boldsymbol{a}(k+1)$ in (4.85) is already known to be equal to $\frac{x(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f^{(0)}(k)\|}$. This fact leads to two slightly different versions of the same algorithm.

Once more, if we have $\boldsymbol{a}(k+1)$, we can find $\boldsymbol{Q}_\theta(k+1)$ using (4.80) and the joint process estimation is carried out with (4.13) and (4.20). The FQR_PRI_B

equations are presented in Table 4.7. The detailed descriptions of two versions of this algorithm is found in Appendix C.

4.5 Conclusions

This chapter derived a new algorithm named FQR_PRI_F or fast QR decomposition RLS algorithm using *a priori* forward errors (based on an upper-triangularization of the input data matrix according to the notation used here) and its relations with other members of the fast QR algorithms family. A comprehensive framework was used to classify the four fast QR algorithms of this family. Their derivations in simple (only matrices equations) and detailed algorithmic descriptions were also provided.

In terms of computational complexity, Table 4.8 shows the comparisons among the four fast QR algorithms according to the algorithms detailed in Appendix C. Note that $p = N + 1$ is the number of coefficients.

Finally, it is important to remark that the fast QR algorithms with lower triangularization of the input data matrix or, equivalently, updating backward prediction errors are of minimal complexity and backward stable under persistent excitation [17, 40].

Table 4.4: The FQR_POS_F equations.

FQR_POS_F

for each k

{ Obtaining $\| \boldsymbol{e}_f(k+1) \|$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq_2}(k) \end{bmatrix}$$

$$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$$

Obtaining $\boldsymbol{Q}_{\theta f}(k+1)$:

$$cos\theta_f(k+1) = \lambda^{1/2} \| \boldsymbol{e}_f(k) \| / \| \boldsymbol{e}_f(k+1) \|$$

$$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \boldsymbol{e}_f(k+1) \|$$

Obtaining $\boldsymbol{c}(k+1)$:

$$\boldsymbol{Q}_\theta^{(N+2)}(k+1) = \boldsymbol{Q}_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{Q}_\theta(k) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

$$\hat{\boldsymbol{Q}}_\theta^{(N+2)}(k+1) = \text{last } (N+2) \times (N+2) \text{ elements of } \boldsymbol{Q}_\theta^{(N+2)}(k+1)$$

$$\boldsymbol{c}(k+1) = \hat{\boldsymbol{Q}}_\theta^{(N+2)}(k+1) \boldsymbol{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}$$

Obtaining $\boldsymbol{Q}'_{\theta b}(k+1)$:

$$\begin{bmatrix} b \\ \boldsymbol{0} \end{bmatrix} = {\boldsymbol{Q}'_{\theta b}}^T(k+1) \boldsymbol{c}(k+1)$$

Obtaining $\boldsymbol{f}(k+1)$:

$$\begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1) \end{bmatrix} = {\boldsymbol{Q}'_{\theta b}}^T(k+1) \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix}$$

Obtaining $\boldsymbol{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_\theta^T(k+1) \begin{bmatrix} \gamma(k+1) \\ \boldsymbol{f}(k+1) \end{bmatrix}$$

Joint Process Estimation:

$$\begin{bmatrix} e_{q_1}(k+1) \\ \boldsymbol{d}_{q_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k) \end{bmatrix}$$

$$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$$

}

Table 4.5: The FQR_PRI_F equations.

FQR_PRI_F

for each k

{ Obtaining $e'_f(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq_2}(k) \end{bmatrix}$$

$$e'_f(k+1) = e_{fq_1}(k+1)/\gamma(k)$$

Obtaining $\boldsymbol{a}(k+1)$:

$$\begin{bmatrix} \frac{e'_b(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_b(k)\|} \\ \boldsymbol{a}(k+1) \end{bmatrix} = \boldsymbol{Q'_{\theta b}}^T(k) \begin{bmatrix} \boldsymbol{a}(k) \\ \frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|} \end{bmatrix}$$

Obtaining $\boldsymbol{Q}_{\theta f}(k+1)$:

$$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$$

$$cos\theta_f(k+1) = \lambda^{1/2} \| \boldsymbol{e}_f(k) \| / \| \boldsymbol{e}_f(k+1) \|$$

$$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \boldsymbol{e}_f(k+1) \|$$

Obtaining $\boldsymbol{c}(k+1)$:

$$\boldsymbol{Q}^{(N+2)}_\theta(k+1) = \boldsymbol{Q}_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{Q}_\theta(k) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix}$$

$$\hat{\boldsymbol{Q}}^{(N+2)}_\theta(k+1) = \text{last } (N+2) \times (N+2) \text{ elements of } \boldsymbol{Q}^{(N+2)}_\theta(k+1)$$

$$\boldsymbol{c}(k+1) = \hat{\boldsymbol{Q}}^{(N+2)}_\theta(k+1) \boldsymbol{Q'_{\theta b}}(k) \begin{bmatrix} 1 \\ \boldsymbol{0} \end{bmatrix}$$

Obtaining $\boldsymbol{Q}'_{\theta b}(k+1)$:

$$\begin{bmatrix} b \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q'_{\theta b}}^T(k+1) \boldsymbol{c}(k+1)$$

Obtaining $\boldsymbol{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1/\gamma(k+1) \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\boldsymbol{a}(k+1) \end{bmatrix}$$

Joint Process Estimation:

$$\begin{bmatrix} e_{q_1}(k+1) \\ \boldsymbol{d}_{q_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k) \end{bmatrix}$$

$$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$$

}

Table 4.6: The FQR_POS_B equations.

FQR_POS_B

for each k

{ Obtaining $\boldsymbol{d}_{fq_2}(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq_2}(k) \end{bmatrix}$$

Obtaining $\| \boldsymbol{e}_f(k+1) \|$:

$$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$$

Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$:

$$\begin{bmatrix} \mathbf{0} \\ \| \boldsymbol{e}_f^{(0)}(k+1) \| \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq_2}(k+1) \\ \| \boldsymbol{e}_f(k+1) \| \end{bmatrix}$$

Obtaining $\boldsymbol{f}(k+1)$:

$$\begin{bmatrix} \frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} \\ \boldsymbol{f}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{f}(k) \\ \frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|} \end{bmatrix}$$

Obtaining $\boldsymbol{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \boldsymbol{Q}_\theta^T(k+1) \begin{bmatrix} \gamma(k+1) \\ \boldsymbol{f}(k+1) \end{bmatrix}$$

Joint Process Estimation:

$$\begin{bmatrix} e_{q_1}(k+1) \\ \boldsymbol{d}_{q_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k) \end{bmatrix}$$

$$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$$

}

Table 4.7: The FQR_PRI_B equations.

FQR_PRI_B

for each k

{ Obtaining $\boldsymbol{d}_{fq_2}(k+1)$:

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq_2}(k) \end{bmatrix}$$

Obtaining $\boldsymbol{a}(k+1)$:

$$\begin{bmatrix} \frac{e'_b(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_b(k)\|} \\ \boldsymbol{a}(k+1) \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k) \begin{bmatrix} \boldsymbol{a}(k) \\ \frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|} \end{bmatrix}$$

Obtaining $\| \boldsymbol{e}_f(k+1) \|$:

$$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$$

Obtaining $\boldsymbol{Q}'_{\theta f}(k+1)$:

$$\begin{bmatrix} \boldsymbol{0} \\ \| \boldsymbol{e}_f^{(0)}(k+1) \| \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq_2}(k+1) \\ \| \boldsymbol{e}_f(k+1) \| \end{bmatrix}$$

Obtaining $\boldsymbol{Q}_\theta(k+1)$:

$$\begin{bmatrix} 1/\gamma(k+1) \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\boldsymbol{a}(k+1) \end{bmatrix}$$

Joint Process Estimation:

$$\begin{bmatrix} e_{q_1}(k+1) \\ \boldsymbol{d}_{q_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{q_2}(k) \end{bmatrix}$$

$$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$$

}

Table 4.8: Comparison of computational complexity.

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

Chapter 5

The Lattice Versions of the Fast QR Algorithms

5.1 Introduction

The fast QR algorithms employing lower triangularization of the input data matrix are known as "hybrid QR-lattice least squares algorithms". It was clear from the previous chapter that these algorithms may update the *a posteriori* or the *a priori* backward prediction errors. Moreover, they are known for their robust numerical behavior and minimal complexity but lack the pipelining property of the lattice algorithms.

The main goal of this chapter is the presentation of the lattice versions of the fast QR algorithms using *a posteriori* and *a priori* backward errors or FQR_POS_B and FQR_PRI_B algorithms according to our classification. The equations of these algorithms are combined in an order recursive manner such that they may be represented as increasing order single loop lattice algorithms. These lattice versions can then be implemented with a modular structure which utilizes a unique type of lattice stage for each algorithm.

Before the derivation of these lattice versions, let us specify on Table 5.1 the meaning of each variable used in both algorithms.

It is worth mentioning here that a variable with no superscript implies in an

N-th order quantity or, equivalently, is related to a $N+1$ coefficients filtering. Let us take as an example the norm of the forward energy: $\| \boldsymbol{e}_f(k) \| = \| \boldsymbol{e}_f^{(N+1)}(k) \|$.

Table 5.1: Variables used in FQR_POS_B and FQR_PRI_B algorithms.

$\boldsymbol{d}_{fq}(k)$:	rotated forward desired vector
$\boldsymbol{d}_{fq2}(k)$:	last N+1 elements of $\boldsymbol{d}_{fq}(k)$
$\boldsymbol{e}_f(k)$:	forward error vector
$\| \boldsymbol{e}_f(k) \|$:	norm of $\boldsymbol{e}_f(k)$
$\boldsymbol{e}_{fq}(k)$:	rotated $\boldsymbol{e}_f(k)$
$e_{fq_1}(k)$:	first element of $\boldsymbol{e}_{fq}(k)$
$\boldsymbol{Q}_\theta(k)$:	Givens matrix (updates the Cholesky factor)
$x(k)$:	input signal
λ :	forgetting factor
$\boldsymbol{Q}'_{\theta f}(k+1)$:	Givens matrix (annihilates $\boldsymbol{d}_{fq2}(k+1)$ in (E3))
$\| \boldsymbol{e}_f^{(0)}(k) \|$:	norm of $\boldsymbol{e}_f(k)$ in a 0 coefficient case
$\boldsymbol{f}(k)$:	*a posteriori* normalized errors
$\boldsymbol{a}(k)$:	*a priori* normalized errors
$e_b(k)$:	backward prediction error
$\| \boldsymbol{e}_b(k) \|$:	norm of $e_b(k)$
$e_f(k)$:	*a posteriori* forward prediction error
$e'_f(k)$:	*a priori* forward prediction error
$e_b(k)$:	*a posteriori* backward prediction error
$e'_b(k)$:	*a priori* backward prediction error
$\gamma(k)$:	product of cosines of the angles of $\boldsymbol{Q}_\theta(k)$
$\boldsymbol{e}_q(k)$:	rotated error vector

(Continuation of Table 5.1)

$e_{q_1}(k)$:	first element of $\boldsymbol{e}_q(k)$
$\boldsymbol{d}_q(k)$:	rotated desired vector
$\boldsymbol{d}_{q_2}(k)$:	last N+1 elements of $\boldsymbol{d}_q(k)$
$d(k)$:	desired signal
$e(k)$:	output error

5.2 Deriving the Lattice Versions

The internal variables found in fast QR algorithms are closely related to those found in conventional lattice algorithms. This was indeed the approach used in [17, 18] to develop these algorithms originally and the implications are well explained in those two references. As pointed out in [17], within this framework the solution to the parameter identification problem was first addressed using fast QR algorithms. The work of [37] stresses the fact that $sin\theta'_{f_i}(k)$ and $sin\theta'_{f_i}(k-1)$ represent the reflection coefficients of the normalized lattice RLS algorithms (*a priori* and *a posteriori*).

On the other hand, the main idea behind the generation of a lattice (or *fully* lattice) version of the fast QR algorithms is the merging of their equations using order updating instead of fixed order variables. This can be done when partial results possess this order updating property. This is indeed the case of the lower triangularization type algorithms since the internal variables are synchronized at instant k or $k-1$ (only order updating). The same facility in obtaining lattice versions is not observed in those algorithms employing upper triangularization (FQR_POS_F and FQR_PRI_F) since since the normalized errors present in the orthogonal matrix $\boldsymbol{Q}_\theta(k)$ are of different orders **at distinct instants of time** (order and time updating).

We next show how to combine the equations of FQR_POS_B in order to obtain its lattice version. Starting from (4.70), we rewrite this equation evaluated at $k+1$,

with an explicit form of $\boldsymbol{Q}_\theta(k)$ in terms of a product of $N+1$ Givens rotations $\boldsymbol{Q}_{\theta_i}(k)$ — see (4.15) — and with $e^{(0)}_{fq_1}(k+1) = x(k+1)$.

$$\begin{bmatrix} e_{fq_1}(k+1) \\ d_{fq2_1}(k+1) \\ \vdots \\ d_{fq2_{N+1}}(k+1) \end{bmatrix} = \begin{bmatrix} cos\theta_N(k) & -sin\theta_N(k) & \mathbf{0}^T \\ sin\theta_N(k) & cos\theta_N(k) & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_N \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} cos\theta_0(k) & \mathbf{0}^T & -sin\theta_0(k) \\ \mathbf{0} & \boldsymbol{I}_N & \mathbf{0} \\ sin\theta_0(k) & \mathbf{0}^T & cos\theta_0(k) \end{bmatrix} \begin{bmatrix} e^{(0)}_{fq_1}(k+1) \\ \lambda^{1/2} d_{fq2_1}(k) \\ \vdots \\ \lambda^{1/2} d_{fq2_{N+1}}(k) \end{bmatrix} \tag{5.1}$$

The product of the first two terms, from right to left, results

$$\begin{bmatrix} cos\theta_0(k) e^{(0)}_{fq_1}(k+1) - sin\theta_0 \lambda^{1/2} d_{fq2_{N+1}}(k) \\ \lambda^{1/2} d_{fq2_1}(k) \\ \vdots \\ \lambda^{1/2} d_{fq2_N}(k) \\ sin\theta_0(k) e^{(0)}_{fq_1}(k+1) + cos\theta_0(k) \lambda^{1/2} d_{fq2_{N+1}}(k) \end{bmatrix} \tag{5.2}$$

The first and last terms of the above equation are, respectively, $e^{(1)}_{fq_1}(k+1)$ and $d_{fq2_{N+1}}(k+1)$. If the other products are computed, one can reach the following relations:

$$e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) - sin\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k) \tag{5.3}$$

$$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) + cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k) \tag{5.4}$$

where i belongs to the closed interval between 1 and $N+1$.

If we use a similar procedure with the equation $\begin{bmatrix} \mathbf{0} \\ \| \boldsymbol{e}^{(0)}_f(k+1) \| \end{bmatrix} = \boldsymbol{Q}'_{\theta f}(k+1) \begin{bmatrix} \boldsymbol{d}_{fq_2}(k+1) \\ \| \boldsymbol{e}_f(k+1) \| \end{bmatrix}$ (part of (4.82) used in the FQR_POS_B algorithm), we will find

$$cos\theta'_{f_{i-1}}(k+1) = \frac{\| \boldsymbol{e}_f^{(i)}(k+1) \|}{\| \boldsymbol{e}_f^{(i-1)}(k+1) \|} \tag{5.5}$$

$$sin\theta'_{f_{i-1}}(k+1) = \frac{d_{fq2_{N+2-i}}(k+1)}{\| \boldsymbol{e}_f^{(i-1)}(k+1) \|} \tag{5.6}$$

In the last equation, i varies from 1 to $N+1$ and the updating of the forward error energy is done by the following generalization of (4.69)

$$\| \boldsymbol{e}_f^{(i)}(k+1) \| = \sqrt{\lambda \| \boldsymbol{e}_f^{(i)}(k) \|^2 + [e_{fq_1}^{(i)}(k+1)]^2} \tag{5.7}$$

All other equations are joined in a single loop by computing partial results from the partial results of the previous equations. The resulting algorithm is shown in Table 5.2 and, although not identical, is similar to the one presented in [42]. A stage of its lattice structure is depicted in Figure 5.1 where the rotation and angle processors can be easily understood from the algorithmic description.

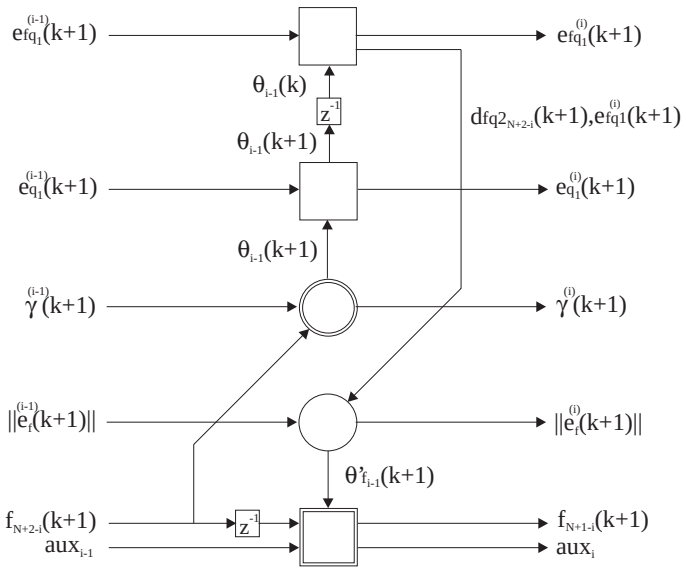

Figure 5.1: One stage of the FQR_POS_B lattice structure.

Finally, the lattice version of the FQR_PRI_B algorithm is obtained in a way which is very similar to the one used to derive the lattice version of the FQR_POS_B algorithm [19]. The algorithm is shown in Table 5.3 and Figure 5.2 depicts one stage of the lattice structure for this algorithm.

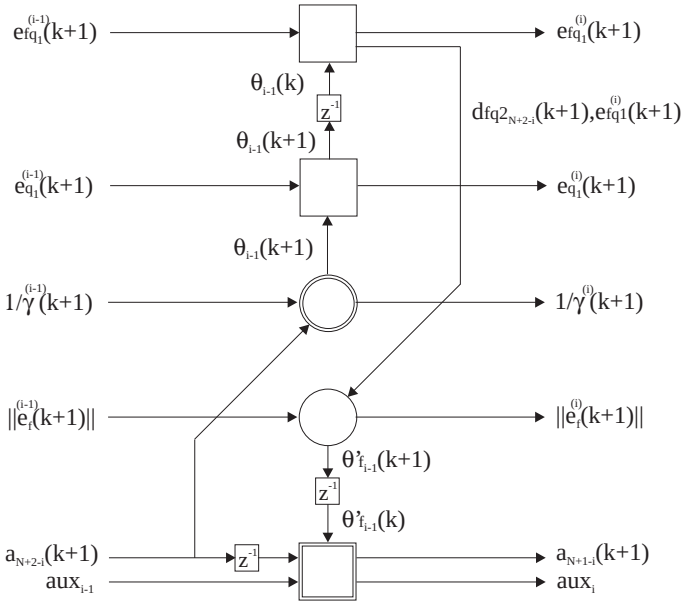

Figure 5.2: One stage of the FQR_PRI_B lattice structure.

5.3 Simulation Results

This section presents some simulation results to test the fast QR algorithms in a finite-precision environment. The setup is a system identification problem with a system order of $N = 10$. The input signal was a colored noise whose eigenvalue spread of its autocorrelation matrix is around 187 and $SNR = 40dB$. The mean-square error (MSE) in dB was measured running the algorithm with a floating-point arithmetic with quantization applied to the mantissa in all operations. The mantissa was rounded excluding the sign bit and assuming the exponent wordlength was sufficient to represent all dynamic ranges. In all algorithms, the constraint of passive rotations ($\sin^2\theta + \cos^2\theta \leq 1$) was imposed. In the first experiment, the mantissa wordlength was varied (8 to 16 bits excluding the sign bit) while keeping fixed the

value of the forgetting factor ($\lambda = 0.98$). Next, the lambda was varied (0.90 to 1.0) for a fixed mantissa wordlength (10 bits). The results, which correspond to the average of ten independent runs, can be observed in Figure 5.3 and Figure 5.4. The figures show that the latice version of the FQR_PR_B algorithm has a performance in finite-precision which is close to the other fast QR algorithms specially when λ is not too close to one. It is also interesting to note that, although the *a priori* algorithms seem to show worse performance, these algorithms do not require the constraint of passive rotations to have the backward consistency guaranteed. It is also claimed in [43] that they have better performance for small mantissa wordlength and λ not too close to 1.

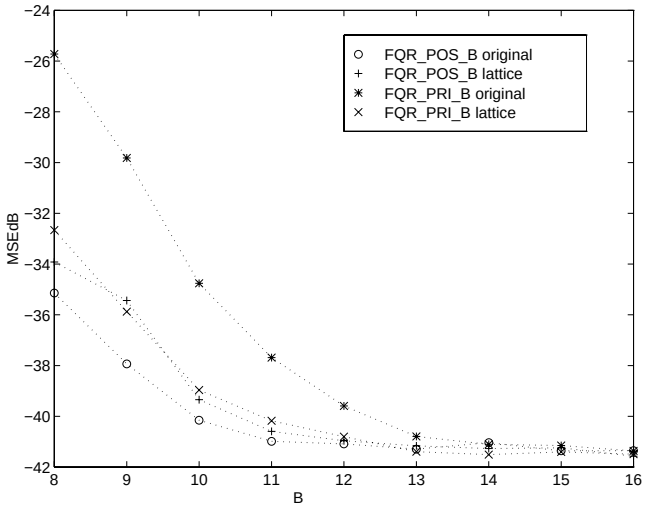

Figure 5.3: Performance of the algorithms in a finite-precision environment (varying B, the number of bits in the mantissa).

5.4 Conclusions

This chapter presented the fully lattice versions of the fast QR algorithms that update *a posteriori* and *a priori* backward errors. The results from the Gram-Schmidt orthogonalization used in Chapter 4 were used to conjecture the reason why only the fast QR algorithms using lower triangularization would have their lattice versions easily implementable.

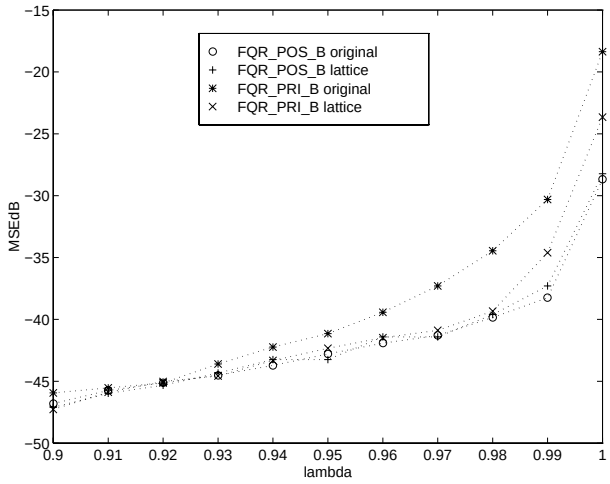

Figure 5.4: MSE in db for different values of λ.

The detailed description of the lattice versions of the FQR_POS_B and FQR_PRI_B algorithms were presented.

The simulation results showed that the performance of the lattice versions in a finite-precision implementation is comparable with the original algorithms. The lattice versions have the same complexity of their original algorithms (FQR_POS_B and FQR_PRI_B) and a lower complexity than the fast QR lattices algorithms previously proposed in [22].

Table 5.2: The lattice version of the FQR_POS_B algorithm.

LATTICE FQR_POS_B

Soft-constrained initialization:

ϵ = small positive value;

for $i = 0 : N + 1$

{ $\| \boldsymbol{e}_f^{(i)}(k) \| = \epsilon$;

}

$\boldsymbol{d}_{fq2}(k) = \text{zeros}(N + 1, 1)$;

$\boldsymbol{d}_{q2}(k) = \text{zeros}(N + 1, 1)$;

$cos\boldsymbol{\theta}(k) = \text{ones}(N + 1, 1)$;

$sin\boldsymbol{\theta}(k) = \text{zeros}(N + 1, 1)$;

$\boldsymbol{f}(k) = \text{zeros}(N + 1, 1)$;

for each k

{ $e_{fq_1}^{(0)}(k + 1) = x(k + 1)$;

$\| \boldsymbol{e}_f^{(0)}(k + 1) \| = \sqrt{[e_{fq_1}^{(0)}(k + 1)]^2 + \lambda \| \boldsymbol{e}_f^{(0)}(k) \|^2}$;

$aux_0 = \frac{x(k+1)}{\|\boldsymbol{e}_f^{(0)}(k+1)\|}$;

$f_{N+1}(k + 1) = aux_0$;

$\gamma^{(0)}(k + 1) = 1$;

$e_{q_1}^{(0)}(k + 1) = d(k + 1)$;

for $i = 1 : N + 1$

{ $d_{fq2_{N+2-i}}(k + 1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k + 1) +$

$cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$e_{fq_1}^{(i)}(k + 1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k + 1) -$

$sin\theta_{i-1}(k) \lambda^{1/2} \boldsymbol{d}_{fq2_{N+2-i}}(k)$;

(Continuation of Table 5.2)

$\| \boldsymbol{e}_f^{(i)}(k+1) \| = \sqrt{[e_{fq_1}^{(i)}(k+1)]^2 + \lambda \| \boldsymbol{e}_f^{(i)}(k) \|^2}$;

$cos\theta'_{f_{i-1}}(k+1) = \| \boldsymbol{e}_f^{(i)}(k+1) \| / \| \boldsymbol{e}_f^{(i-1)}(k+1) \|$;

$sin\theta'_{f_{i-1}}(k+1) = d_{fq2_{N+2-i}}(k+1) / \| \boldsymbol{e}_f^{(i-1)}(k+1) \|$;

$f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k+1) aux_{i-1}}{cos\theta'_{f_{i-1}}(k+1)}$;

$aux_i = -sin\theta'_{f_{i-1}}(k+1) f_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k+1) aux_{i-1}$;

$\gamma^{(i)}(k+1) = \sqrt{[\gamma^{(i-1)}(k+1)]^2 - [f_{N+2-i}(k+1)]^2}$;

$cos\theta_{i-1}(k+1) = \frac{\gamma^{(i)}(k+1)}{\gamma^{(i-1)}(k+1)}$;

$sin\theta_{i-1}(k+1) = \frac{f_{N+2-i}(k+1)}{\gamma^{(i-1)}(k+1)}$;

$dq2_{N+2-i}(k+1) = sin\theta_{i-1}(k+1) e_{q1}^{(i-1)}(k+1) +$
$cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) -$
$sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;

}

$e(k+1) = e_{q_1}^{(N+1)}(k+1) \gamma^{(N+1)}(k+1)$;

}

Table 5.3: The lattice version of the FQR_PRI_B algorithm.

LATTICE FQR_PRI_B

Soft-constrained initialization:

ϵ = small positive value;

for $i = 0 : N + 1$

{ $\| \boldsymbol{e}_f^{(i)}(k) \| = \epsilon$;

}

$\boldsymbol{d}_{fq2}(k) = \text{zeros}(N + 1, 1)$;

$\boldsymbol{d}_{q2}(k) = \text{zeros}(N + 1, 1)$;

$cos\boldsymbol{\theta}(k) = \text{ones}(N + 1, 1)$;

$cos\boldsymbol{\theta}'_f(k) = \text{ones}(N + 1, 1)$;

$sin\boldsymbol{\theta}(k) = \text{zeros}(N + 1, 1)$;

$sin\boldsymbol{\theta}'_f(k) = \text{zeros}(N + 1, 1)$;

$\boldsymbol{a}(k) = \text{zeros}(N + 1, 1)$;

for each k

{ $aux_0 = \frac{x(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f^{(0)}(k)\|}$;

$a_{N+1}(k + 1) = aux_0$;

$e_{fq_1}^{(0)}(k + 1) = x(k + 1)$;

$\| \boldsymbol{e}_f^{(0)}(k + 1) \| = \sqrt{[e_{fq_1}^{(0)}(k + 1)]^2 + \lambda \| \boldsymbol{e}_f^{(0)}(k) \|^2}$;

$1/\gamma^{(0)}(k + 1) = 1$;

$e_{q_1}^{(0)}(k + 1) = d(k + 1)$;

(Continuation of Table 5.3)

for $i = 1 : N + 1$

{ $a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k) aux_{i-1}}{cos\theta'_{f_{i-1}}(k)}$;

$aux_i = -sin\theta'_{f_{i-1}}(k) a_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k) aux_{i-1}$;

$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) +$

$cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k)$;

$e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k+1) -$

$sin\theta_{i-1}(k) \lambda^{1/2} \boldsymbol{d}_{fq2_{N+2-i}}(k)$;

$\| \boldsymbol{e}^{(i)}_f(k+1) \| = \sqrt{[e^{(i)}_{fq_1}(k+1)]^2 + \lambda \| \boldsymbol{e}^{(i)}_f(k) \|^2}$;

$cos\theta'_{f_{i-1}}(k+1) = \| \boldsymbol{e}^{(i)}_f(k+1) \| / \| \boldsymbol{e}^{(i-1)}_f(k+1) \|$;

$sin\theta'_{f_{i-1}}(k+1) = d_{fq2_{N+2-i}}(k+1) / \| \boldsymbol{e}^{(i-1)}_f(k+1) \|$;

$1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + [a_{N+2-i}(k+1)]^2}$;

$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;

$sin\theta_{i-1}(k+1) = \frac{a_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;

$dq2_{N+2-i}(k+1) = sin\theta_{i-1}(k+1) e^{(i-1)}_{q1}(k+1) +$

$cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;

$e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) -$

$sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;

}

$e(k+1) = e^{(N+1)}_{q_1}(k+1) / [1/\gamma^{(N+1)}(k+1)]$;

}

Chapter 6

Contributions to the Finite-Precision Analysis of the Fast QR Algorithms

6.1 Introduction

In chapter 4, it was remarked that the fast QR algorithms with lower triangularization (according to the notation used in this work) of the input data matrix are minimal in a system theory sense and backward stable [40]. It was also shown that they have lower computational load when compared to the fast QR algorithms using upper triangularization. Moreover, it was observed in [43] that the FQR_PRI_B algorithm performs better for forgetting factors not too close to one due to the fact that passive rotations are not necessary.

These facts turn these algorithms natural candidates for practical implementation and the finite-precision effect is then a topic of major concern. This chapter deals with the determination of the steady-state quantization error of the internal variables of both FQR_POS_B and FQR_PRI_B algorithms. A complete finite-precision analysis of both algorithms are beyond the scope of this chapter but the process of obtaining the mean-squared value of accumulated quantization errors of

some variables are presented as a contribution towards the solution of this problem. The validation of the expressions obtained are carried out through computer simulations.

6.2 Infinite-Precision Analysis

This section reviews the dynamic range of the internal quantities of the FQR_POS_B and FQR_PRI_B algorithms. Most of the results presented in this section are available in the technical literature but are reviewed here, for they are required in the finite-precision analysis.

6.2.1 Infinite-Precision Results for the FQR_POS_B Algorithm

All variables have the same notation used in Appendix C. Note that the version 2 of the FQR_POS_B algorithm will be used hereafter.

Mean Squared Values of $cos\theta_i(k)$ and $sin\theta_i(k)$

The following results can be found in [44].

$$E[cos^2\theta_i(k)] \approx \lambda \tag{6.1}$$

$$E[sin^2\theta_i(k)] \approx 1 - \lambda \tag{6.2}$$

Mean Squared Value of $e_{fq_1}^{(i)}(k)$

The following result can be found in [45].

$$E\left\{[e_{fq_1}^{(i)}(k)]^2\right\} \approx \sigma_x^2 \left(\frac{2\lambda}{1+\lambda}\right)^i \tag{6.3}$$

Mean Squared Value of $d_{fq2_i}(k)$

The following result can be found in [45].

$$E\left\{[d_{fq2_i}(k)]^2\right\} \approx \frac{\sigma_x^2}{1+\lambda}\left(\frac{2\lambda}{1+\lambda}\right)^{N+1-i} \tag{6.4}$$

Mean Squared Value of $\| \boldsymbol{e}_f^{(i)}(k) \|$

The following result can be found in [45].

$$E[\| \boldsymbol{e}_f^{(i)}(k) \|^2] \approx \frac{\sigma_x^2}{1-\lambda}\left(\frac{2\lambda}{1+\lambda}\right)^{i} \tag{6.5}$$

Mean Squared Values of $cos\theta'_{f_i}(k)$ and $sin\theta'_{f_i}(k)$

The following results can be found in [45].

$$E[cos^2\theta'_{f_i}(k)] \approx \frac{2\lambda}{1+\lambda} \tag{6.6}$$

$$E[sin^2\theta'_{f_i}(k)] \approx \frac{1-\lambda}{1+\lambda} \tag{6.7}$$

Mean Squared Value of $\gamma^{(i)}(k)$

If we recall from Chapter 4 that $\gamma(k) = \prod_{i=0}^{N} cos\theta_i(k)$, use (6.1) and (6.2), and assume independence between $cos\theta_i(k)$ and $cos\theta_j(k)$, $i \neq j$, it is easy to find the next expression also obtained in [43] using a different approach.

$$E\left\{[\gamma^{(i)}(k)]^2\right\} \approx \lambda^i \tag{6.8}$$

Mean Squared Value of $f_i(k)$

If we take the square of $\gamma^{(i)}(k+1)$ from version 2 of the FQR_POS_B algorithm in Appendix C and substitute $cos^2\theta_{i-1}(k+1)$ and then $sin^2\theta_{i-1}(k+1)$ for the expressions also found there, we obtain

$$[\gamma^{(i)}(k+1)]^2 = [\gamma^{(i-1)}(k+1)]^2 - f_{N+2-i}^2(k+1) \tag{6.9}$$

By taking the expected value of (6.9) and the approximation of (6.8), we find the following expression also available in [43].

$$E[f_i^2(k)] \approx \lambda^{N+1-i}(1-\lambda) \tag{6.10}$$

If we take from Appendix C the expressions for $f_{i-1}(k+1)$ and aux_i, and use them to calculate $E[f_{i-1}^2(k)]+E[aux_i^2]$, it is straightforward to obtain $E[aux_i^2]+E[f_{i-1}^2(k)] = E[aux_{i-1}^2]+E[f_i^2(k)]$. Seeing that $f_{N+1}(k+1) = aux_{N+1}$, it is easy to figure out that $E[aux_i^2] = E[f_i^2(k)]$ and, therefore, (6.10) can also be used as a good approximation for $E[aux_i^2]$.

Mean Squared Value of $d_{q2_i}(k)$

The following results can be found in [44].

$$E[d_{q2_{N+1-i}}^2(k)] \approx \left[\frac{2\lambda}{1+\lambda}\right]^i \left[\frac{\sigma_x^2}{1-\lambda} w_{0,i}^2 + \frac{\sigma_x^2}{1+\lambda} \sum_{j=i+1}^{N} w_{0,j}^2\right] \tag{6.11}$$

where $w_{0,i}^2 = E[w_i^2(k)]$. Observe that although $w_{0,i}$ is not available, a rough estimate of $\sigma_x^2 w_{0,i}^2$ can be obtained based on the power of the reference signal [44].

Mean Squared Value of $e_{q_1}^{(i)}(k)$

From the Joint Process Estimation part of the FQR_POS_B algorithm we take the expressions of $e_{q_1}^{(i)}(k+1)$ and $d_{q2_{N+2-i}}(k+1)$, and use them to derive the expected value of $[e_{q_1}^{(i)}(k+1)]^2 + [d_{q2_{N+2-i}}(k+1)]^2$. By assuming stationarity, we find the following relation.

$$E\left\{[e_{q_1}^{(i)}(k)]^2\right\} = E\left\{[e_{q_1}^{(i-1)}(k)]^2\right\} - (1-\lambda)E[d_{q2_{N+2-i}}^2(k)] \tag{6.12}$$

where $E\left\{[e_{q_1}^{(0)}(k)]^2\right\} = \sigma_d^2 = \sigma_x^2 \sum_{i=0}^{N} w_{o,i}^2 + \sigma_n^2$ is the variance of the reference signal and σ_n^2 is the variance of the measurement noise (it is assumed here that the algorithm is applied in a sufficient-order identification problem, i.e., the unknown FIR system has the same order of the adaptive filter).

Finally, from the last equation of the algorithm, we have $E[e^2(k)] \approx \lambda^{N+1} E[e_{q_1}^2(k)]$. Since from (6.12) and (6.11) we have that $E[e_{q_1}^2(k)] = \sigma_n^2$, the following expression

results:

$$E[e^2(k)] \approx \lambda^{N+1}\sigma_n^2 \tag{6.13}$$

6.2.2 Infinite-Precision Results for the FQR_PRI_B Algorithm

The FQR_PRI_B algorithm is similar to the FQR_POS_B algorithm in a sense that five of its seven equations are identical (three of them are matrix equations) as can be seen from Tables 4.6 and 4.7. It means that except (6.9) and (6.10) all other equations presented in this chapter remain valid and only the expression for the mean squared value of $a_i(k)$ is left for the infinite-precision analysis of this algorithm. It is worth mentioning that only version 2 of the FQR_PRI_B algorithm, as described in Appendix C, is under investigation here.

Mean Squared Value of $a_i(k)$

From the implementation of (4.80) described in version 2 of the FQR_PRI_B algorithm (step "Obtaining $\boldsymbol{Q}_\theta(k+1)$") we have (see Appendix C)

$$a_{N+2-i}^2(k+1) = [\gamma^{(i)}(k+1)]^{-2} - [\gamma^{(i-1)}(k+1)]^{-2} \tag{6.14}$$

By taking the expected value of (6.14), using the approximation of (6.8), and employing the averaging principle [46, 47], it follows the next expression also available in [43].

$$E[a_i^2(k)] \approx \lambda^{-(N+2-i)}(1-\lambda) \tag{6.15}$$

If we take from Appendix C the expressions for $a_{i-1}(k+1)$ and aux_i, and use them to calculate $E[a_{i-1}^2(k)] + E[aux_i^2]$, it is straightforward to obtain $E[aux_i^2] + E[a_{i-1}^2(k)] = E[aux_{i-1}^2] + E[a_i^2(k)]$. Since $a_{N+1}(k+1) = aux_{N+1}$, it is easy to figure out that $E[aux_i^2] = E[a_i^2(k)]$ and (6.15) can also be used as a good approximation for $E[aux_i^2]$.

6.3 Contribution to the Finite-Precision Analysis

This section first presents the fixed-point quantization error model to be used in the rest of the chapter. The first matrix equation of the FQR_PRI_B algorithm is then analyzed as an example of the method used. The results of this analysis are very good as will be seen in the next section. On the other hand, the analysis expression for the initialization of the second matrix equation of the same algorithm presented poor result when the so called averaging principle was used. An alternative to the averaging principle is then derived and applied to develop a more accurate approximation.

6.3.1 Fixed-Point Quantization Error Model

A number of assumptions are made here in order to have a simple model of the quantization error. We assume that no overflow due to additions and subtractions occurs. Two's complement arithmetic is used for numeric representations of the internal variables which are assumed properly scaled to avoid overflow. It is assumed that the operation $*$ (multiplication, division or exponentiation) introduces the following quantization error.

$$\eta_*(a,b) \triangleq a * b - Q[a * b] \tag{6.16}$$

where a and b are scalars and it is assumed that the instantaneous quantizations are performed by rounding such that the quantization error is a white noise with zero mean and variance $\frac{2^{-2B}}{12}$, B being the number of bits excluding the sign bit.

The accumulated quantization error is defined as the difference between the infinite-precision implementation and the finite-precision implementation of a variable. If $r(k)$ is a variable, its *accumulated quantization error* is given by

$$\Delta r(k) \triangleq r(k) - r_Q(k) \tag{6.17}$$

and we are interested here in finding the mean squared value of this quantity.

6.3.2 The FQR_PRI_B Algorithm: Mean Squared Value of $\Delta e_{fq_1}^{(i)}(k)$ and $\Delta d_{fq2_i(k)}$

Let us start by analyzing (4.70) in finite-precision. Note that the analysis expressions that will be obtained from this matrix equation is valid for both FQR_POS_B and FQR_PRI_B algorithms (and are expected to be quite similar to the FQR_POS_F and FQR_PRI_F counterparts) since the following expression is common for all the four FQR algorithms studied.

$$\begin{bmatrix} e_{fq_1}(k+1) \\ \boldsymbol{d}_{fq_2}(k+1) \end{bmatrix} = \boldsymbol{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\boldsymbol{d}_{fq_2}(k) \end{bmatrix} \tag{6.18}$$

where $x(k+1)$ is assumed to be already quantized as well as all other external signals and constants. The previous equation is implemented as

$$\begin{aligned} e_{fq_1}^{(i)}(k+1) &= cos\theta_{i-1}(k)e_{fq_1}^{(i-1)}(k+1) \\ &\quad -sin\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \qquad (6.19) \\ d_{fq2_{N+2-i}}(k+1) &= sin\theta_{i-1}(k)e_{fq_1}^{(i-1)}(k+1) \\ &\quad +cos\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \qquad (6.20) \end{aligned}$$

where $i = 1 : N+1$ and $e_{fq_1}^{(0)}(k+1) = x(k+1)$. Equations (6.19) and (6.20) above implemented in finite-precision are given by

$$\begin{aligned} e_{fq_{1;Q}}^{(i)}(k+1) &= cos\theta_{i-1;Q}(k)e_{fq_1;Q}^{(i-1)}(k+1) \\ &\quad -sin\theta_{i-1;Q}(k)\lambda^{1/2}d_{fq2_{N+2-i};Q}(k) - \eta_1(k) \qquad (6.21) \\ d_{fq2_{N+2-i};Q}(k+1) &= sin\theta_{i-1;Q}(k)e_{fq_1;Q}^{(i-1)}(k+1) \\ &\quad +cos\theta_{i-1;Q}(k)\lambda^{1/2}d_{fq2_{N+2-i};Q}(k) - \eta_2(k) \qquad (6.22) \end{aligned}$$

where $\eta_1(k) = \eta_M(cos\theta_{i-1;Q}(k), e_{fq_1;Q}^{(i-1)}(k+1)) - \eta_M(sin\theta_{i-1;Q}(k), \lambda^{1/2}, d_{fq2_{N+2-i};Q}(k))$ and $\eta_2(k) = \eta_M(sin\theta_{i-1;Q}(k), e_{fq_1;Q}^{(i-1)}(k+1)) + \eta_M(cos\theta_{i-1;Q}(k), \lambda^{1/2}, d_{fq2_{N+2-i};Q}(k))$ are the quantization errors introduced by the multiplications.

From (6.19) and (6.21), we derive the expression for the accumulated quantiza-

tion error of $e_{fq_1}^{(i)}(k+1)$ as follows.

$$\begin{aligned}
\Delta e_{fq_1}^{(i)}(k+1) &= e_{fq_1}^{(i)}(k+1) - e_{fq_1;Q}^{(i)}(k+1) \\
&= cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) \\
&\quad -sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k) \\
&\quad -cos\theta_{i-1;Q}(k) e_{fq_1;Q}^{(i-1)}(k+1) \\
&\quad +sin\theta_{i-1;Q}(k)\lambda^{1/2} d_{fq2_{N+2-i};Q}(k) + \eta_1(k) \qquad (6.23)
\end{aligned}$$

In (6.23) by replacing the four quantized values by the infinite-precision value minus the accumulated quantization error (as in $r_Q(k) = r(k) - \Delta r(k)$) and admitting that the absolute values of the cross products of the accumulated quantization errors are much smaller than the absolute values of the other terms, we find

$$\begin{aligned}
\Delta e_{fq_1}^{(i)}(k+1) &\approx cos\theta_{i-1}(k)\Delta e_{fq_1}^{(i-1)}(k+1) + \Delta cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) \\
&\quad -sin\theta_{i-1}(k)\lambda^{1/2}\Delta d_{fq2_{N+2-i}}(k) \\
&\quad -\Delta sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k) + \eta_1(k) \qquad (6.24)
\end{aligned}$$

If we now assume that $\eta_1(k)$ and the accumulated quantization errors are zero mean and have a very small cross-correlation between each other, the following expression can be easily obtained.

$$\begin{aligned}
E\left\{[\Delta e_{fq_1}^{(i)}(k+1)]^2\right\} &\approx E\left\{[cos\theta_{i-1}(k)\Delta e_{fq_1}^{(i-1)}(k+1)]^2\right\} \\
&\quad +E\left\{[\Delta cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1)]^2\right\} \\
&\quad +\lambda E\left\{[sin\theta_{i-1}(k)\Delta d_{fq2_{N+2-i}}(k)]^2\right\} \\
&\quad +\lambda\left\{[\Delta sin\theta_{i-1}(k) d_{fq2_{N+2-i}}(k)]^2\right\} + E[\eta_1^2(k)] \qquad (6.25)
\end{aligned}$$

The final expression for the mean squared value of $\Delta e_{fq_1}^{(i)}(k+1)$ is obtained by assuming stationarity, by assuming that the variables are uncorrelated with the accumulated quantization errors, and by using the results from the infinite-precision

analysis. That is

$$\begin{aligned} E\left\{[\Delta e_{fq_1}^{(i)}(k)]^2\right\} &\approx \lambda E\left\{[\Delta e_{fq_1}^{(i-1)}(k)]^2\right\} \\ &+\sigma_x^2\left(\frac{2\lambda}{1+\lambda}\right)^{(}i-1))E\left\{[\Delta cos\theta_{i-1}(k)]^2\right\} \\ &+\lambda(1-\lambda)E\left\{[\Delta d_{fq2_{N+2-i}}(k)]^2\right\} \\ &+\lambda\frac{\sigma_x^2}{1+\lambda}\left(\frac{2\lambda}{1+\lambda}\right)\left\{[\Delta sin\theta_{i-1}(k)]^2\right\}+E[\eta_1^2(k)] \quad (6.26) \end{aligned}$$

where $E[\eta_1^2(k)] = 2\frac{2^{-2B}}{12}$.

The accumulated quantization error of $d_{fq2_{N+2-i}}(k+1)$ will be next derived from (6.20) and (6.22).

$$\begin{aligned} \Delta d_{fq2_{N+2-i}}(k+1) &= d_{fq2_{N+2-i}}(k+1) - d_{fq2_{N+2-i};Q}(k+1) \\ &= sin\theta_{i-1}(k)e_{fq_1}^{(i-1)}(k+1) \\ &+cos\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \\ &-sin\theta_{i-1;Q}(k)e_{fq_1;Q}^{(i-1)}(k+1) \\ &-cos\theta_{i-1;Q}(k)\lambda^{1/2}d_{fq2_{N+2-i};Q}(k)+\eta_2(k) \quad (6.27) \end{aligned}$$

By using the same approach and assumptions used for obtaining $E\left\{[\Delta e_{fq_1}^{(i)}(k)]^2\right\}$ we find the mean squared value of the accumulated quantization error of $d_{fq2_{N+2-i}}(k+1)$ given by

$$\begin{aligned} E\left\{[\Delta d_{fq2_{N+2-i}}(k)]^2\right\} &\approx \frac{\sigma_x^2}{2\lambda(1-\lambda)}\left(\frac{2\lambda}{1+\lambda}\right)^i E\left\{[\,\Delta sin\theta_{i-1}(k)]^2\right\} \\ &+\frac{1}{1+\lambda}E\left\{[\Delta e_{fq_1}^{(i-1)}(k)]^2\right\} \\ &+\frac{\sigma_x^2}{2(1-\lambda^2)}\left(\frac{2\lambda}{1+\lambda}\right)^i E\left\{[\Delta cos\theta_{i-1}(k)]^2\right\} \\ &+\frac{E[\eta_2^2(k)]}{1-\lambda^2} \quad (6.28) \end{aligned}$$

where $E[\eta_2^2(k)] = 2\frac{2^{-2B}}{12}$.

6.3.3 The FQR_PRI_B Algorithm: Mean Squared Value of Δaux_0

In the former section, we have given as an example of finite-precision analysis, the expressions for the mean squared values of the accumulated quantization errors of the first matrix equation of the FQR_PRI_B algorithm. The next matrix equation of this algorithm has an initialization (parameter aux_0) given by

$$aux_0 = \frac{e_{fq_1}(k+1)}{\gamma(k)\lambda^{1/2} \parallel \boldsymbol{e}_f(k) \parallel} \tag{6.29}$$

In a finite-precision environment, the previous equation is implemented as

$$aux_{0;Q} = \frac{e_{fq_1;Q}(k+1)}{\gamma_Q(k)\lambda^{1/2} \parallel \boldsymbol{e}_f(k) \parallel_Q - \eta_3(k)} - \eta_4(k) \tag{6.30}$$

where $\eta_3(k)$ and $\eta_4(k)$ are the quantization errors due to the multiplication and division, respectively.

From (6.29) and (6.30), we write the accumulated quantization error of aux_0 as

$$\begin{aligned} \Delta aux_0 &= aux_0 - aux_{0;Q} \\ &= \frac{e_{fq_1}(k+1)}{\gamma(k)\lambda^{1/2} \parallel \boldsymbol{e}_f(k) \parallel} \\ &\quad - \frac{e_{fq_1;Q}(k+1)}{\gamma_Q(k)\lambda^{1/2} \parallel \boldsymbol{e}_f(k) \parallel_Q - \eta_3(k)} + \eta_4(k) \\ &= \frac{e_{fq_1}(k+1)}{r} - \frac{e_{fq_1;Q}(k+1)}{r - \Delta r} + \eta_4(k) \end{aligned} \tag{6.31}$$

where $r = \gamma(k)\lambda^{1/2} \parallel \boldsymbol{e}_f(k) \parallel$.

If we use the approximation $\frac{1}{r-\Delta r} \approx \frac{1}{r}\left(1 + \frac{\Delta r}{r}\right)$ in (6.31) and substitute $e_{fq_1;Q}(k+1)$ by $e_{fq_1}(k+1) - \Delta e_{fq_1}(k+1)$ we obtain

$$\Delta aux_0 \approx \frac{\Delta e_{fq_1}(k+1)}{r} - \frac{[e_{fq_1}(k+1) - \Delta e_{fq_1}(k+1)]\Delta r}{r^2} + \eta_4(k) \tag{6.32}$$

By substituting the expressions of r and Δr in (6.32) and assuming that the term with the product of the two accumulated errors is much smaller in absolute value than the other terms, we obtain

$$\begin{aligned} \Delta aux_0 &\approx \frac{\Delta e_{fq_1}(k+1)}{\lambda^{1/2}\gamma(k) \parallel \boldsymbol{e}_f(k) \parallel} - \frac{e_{fq_1}(k+1)\Delta \parallel \boldsymbol{e}_f(k) \parallel}{\lambda^{1/2}\gamma(k) \parallel \boldsymbol{e}_f(k) \parallel^2} \\ &\quad - \frac{e_{fq_1}(k+1)\Delta\gamma(k)}{\lambda^{1/2}\gamma^2(k) \parallel \boldsymbol{e}_f(k) \parallel} - \frac{e_{fq_1}(k+1)\eta_3(k)}{\lambda\gamma^2(k) \parallel \boldsymbol{e}_f(k) \parallel^2} + \eta_4(k) \end{aligned} \tag{6.33}$$

If we now assume that the instantaneous errors ($\eta_3(k)$ and $\eta_4(k)$) and the accumulated errors ($\Delta e_{fq_1}(k+1)$, $\Delta \parallel \boldsymbol{e}_f(k) \parallel$, and $\Delta\gamma(k)$) are zero mean with small cross-correlation between each other, the mean squared quantization error of aux_o results.

$$\begin{aligned} E\left\{[\Delta aux_0]^2\right\} \approx & \frac{1}{\lambda} E\left\{[\Delta e_{fq_1}(k)]^2\right\} E\left[\frac{1}{\gamma^2(k)}\right] E\left[\frac{1}{\parallel \boldsymbol{e}_f(k) \parallel^2}\right] \\ & + \frac{E[e_{fq_1}^2(k)]}{\lambda} E\left\{[\Delta \parallel \boldsymbol{e}_f(k) \parallel]^2\right\} E\left[\frac{1}{\gamma^2(k)}\right] E\left[\frac{1}{\parallel \boldsymbol{e}_f(k) \parallel^4}\right] \\ & + \frac{E[e_{fq_1}^2(k)]}{\lambda} E\left\{[\Delta\gamma(k)]^2\right\} E\left[\frac{1}{\gamma^4(k)}\right] E\left[\frac{1}{\parallel \boldsymbol{e}_f(k) \parallel^2}\right] \\ & + \frac{E[e_{fq_1}^2(k)] E[\eta_3^2(k)]}{\lambda^2} E\left[\frac{1}{\gamma^4(k)}\right] E\left[\frac{1}{\parallel \boldsymbol{e}_f(k) \parallel^4}\right] \\ & + E[\eta_4^2(k)] \end{aligned} \tag{6.34}$$

From the infinite-precision analysis, we have $E[e_{fq_1}^2(k)]$, $E[\gamma^2(k)]$, and $E[\parallel \boldsymbol{e}_f(k) \parallel^2]$. If we use the averaging principle and approximate $E[1/x^2]$ and $E[1/x^4]$ by $1/E[x^2]$ and $1/E[x^4]$, respectively, and substitute the results from the infinite-precision analysis, we find

$$\begin{aligned} E\left\{[\Delta aux_0]^2\right\} \approx & \frac{(1-\lambda)}{\lambda^{N+2}\sigma_x^2} \left(\frac{1+\lambda}{2\lambda}\right)^{N+1} E\left\{[\Delta e_{fq_1}(k)]^2\right\} \\ & + \frac{(1-\lambda)^2}{\lambda^{N+2}\sigma_x^2} \left(\frac{1+\lambda}{2\lambda}\right)^{N+1} E\left\{[\Delta \parallel \boldsymbol{e}_f(k) \parallel]^2\right\} \\ & + \frac{1-\lambda}{\lambda^{2N+3}} E\left\{[\Delta\gamma(k)]^2\right\} \\ & + \frac{(1-\lambda)^2}{\lambda^{2N+3}\sigma_x^2} \left(\frac{1+\lambda}{2\lambda}\right)^{N+1} E[\eta_3^2(k)] \\ & + E[\eta_4^2(k)] \end{aligned} \tag{6.35}$$

where $E[\eta_3^2(k)] = E[\eta_4^2(k)] = \frac{2^{-2B}}{12}$.

As will be seen from the simulations, (6.35) presents a value in dB which defers more than one dB from the simulated result. Nevertheless, we still claim that (6.34) is a reasonable approximation for $E\left\{[\Delta aux_0]^2\right\}$. Let us see in the next subsection how to validate (6.34).

6.3.4 Refining the approximations of $E[1/x^2]$ and $E[1/x^4]$

The so-called averaging principle [46] has been widely used in the derivation of relations used in infinite and finite-precision analysis and is stated as

$$E\left\{\frac{f[x(k)]}{g[y(k)]}\right\} \approx \frac{E\{f[x(k)]\}}{E\{g[y(k)]\}} \tag{6.36}$$

which in the present case is valid for large k and forgetting factor close to unity [44].

One reason of the poor result of (6.35) is due to the approximation of $E[1/x^2(k)]$ by $1/E[x^2(k)]$ and $E[1/x^4(k)]$ by $1/E^2[x^2(k)]$. The purpose of this subsection is to derive more reliable approximations for these expressions in order to validate (6.34).

We started to search an alternative solution to this problem by taking the *Gaussian Moment Factoring Theorem* [1] for four samples of a zero-mean, real Gaussian process[1] and admitting they are all equal to $x(k)$. The resulting expression is

$$E[x^4(k)] = 3E^3[x^2(k)] \tag{6.37}$$

If we now admit that $x(k)$ has a mean value different from zero ($E[x(k)] = \bar{x}$) and model it as $x(k) = x'(k) + \bar{x}$ with $x'(k)$ being zero-mean real Gaussian, we have

$$\begin{aligned} E[x^4(k)] &= E[(x'(k) + \bar{x})^4] \\ &= 3E^2[x'^2(k)] + 6E[x'^2(k)]\bar{x}^2 + \bar{x}^4 \\ &= 3E^2[x^2(k)] - 2\bar{x}^4 \end{aligned} \tag{6.38}$$

By using the same approach of [4] and equation (6.38) we will derive an expression

[1] $E[z_1 z_2 z_3 z_4] = E[z_1 z_2]E[z_3 z_4] + E[z_1 z_3]E[z_2 z_4] + E[z_1 z_4]E[z_2 z_3]$

for $E[1/x^2(k)]$.

$$\begin{aligned} E\left[\frac{1}{x^2(k)}\right] &= \frac{1}{E[x^2(k)]}E\left[\frac{1}{1-\left(1-\frac{x^2(k)}{E[x^2(k)]}\right)}\right] \\ &= \frac{1}{E[x^2(k)]}E\left[\sum_{i=0}^{\infty}\left(1-\frac{x^2(k)}{E[x^2(k)]}\right)^i\right] \\ &= \frac{1}{E[x^2(k)]}\left(1+0+\frac{E[x^4(k)]-E^2[x^2(k)]}{E^2[x^2(k)]}+\cdots\right) \\ &= \frac{1}{E[x^2(k)]}\left(\frac{E[x^4(k)]}{E^2[x^2(k)]}+\cdots\right) \\ &\approx \frac{3E^2[x^2(k)]-2\bar{x}^4}{E^3[x^2(k)]} \end{aligned} \tag{6.39}$$

Finally, we can find an approximation for $E[1/x^4(k)]$ if we use an auxiliary variable $y = x^2(k)$ and (6.39). It is easy to see that $E[y] = E[x^2(k)]$ and that $E[y^2] = E[x^4(k)]$ is available from (6.38).

$$E\left[\frac{1}{x^4(k)}\right] \approx \frac{3\left\{3E^2[x^2(k)]-2\bar{x}^4\right\}^2-2\left\{E[x^2(k)]\right\}^4}{\left\{3E^2[x^2(k)]-2\bar{x}^4\right\}^3} \tag{6.40}$$

where $\bar{x} = E[x(k)]$.

It is worth mentioning that the above approximations are valid whenever the series expansion used in the derivation is valid. The critical case happens when $x(k)$ tends to zero (division by zero) and the best results come when the $\bar{x}/\sigma_x$ is not so low. A small simulation was carried out to check the results of the expressions derived here. In this simulation, $x(k)$ was a Gaussian random variable with variance $\sigma_x^2 = 10^{-3}$ and mean value set to 0.1, 0.3, 0.5, and 1.0. The following Table 6.1 displays the results of a 100000 samples simulation and it is clear from there that these new expressions outperform old approximations for different values of $\bar{x}/\sigma_x$. Also note that for case of the smallest mean value ($\bar{x} = 0.1$), the variable crosses the zero several times and this is the reason for the large values of $E\left[\frac{1}{x^4(k)}\right]$.

By using equations (6.38), (6.39), and (6.40), we can obtain the following expressions to be used in (6.34) whose results, as will be seen from the simulations, are more accurate than the results obtained with (6.35). Note that the new relations obtained here were used specifically with the purpose of validating (6.34) since we do not have $E[\gamma(k)]$ and $E[\| \boldsymbol{e}_f(k) \|]$ available from the infinite-precision analysis.

Table 6.1: Comparison of Performance of the New Expressions.

EXPRESSION	$\bar{x} = 0.1$	$\bar{x} = 0.3$	$\bar{x} = 0.5$	$\bar{x} = 1.0$
$E[x^4(k)]$	$1.6313\ 10^{-4}$	0.0086	0.0640	1.0060
$E^2[x^2(k)]$	$1.2100\ 10^{-4}$	0.0083	0.0630	1.0020
(6.38)	$1.6300\ 10^{-4}$	0.0086	0.0640	1.0060
$E[\frac{1}{x^2(k)}]$	$2.8489\ 10^{4}$	11.5041	4.0490	1.0030
$\frac{1}{E[x^2(k)]}$	90.9092	10.9890	3.9841	0.9990
(6.39)	122.4642	11.4694	4.0474	1.0030
$E[\frac{1}{x^4(k)}]$	$4.1738\ 10^{13}$	139.0497	16.6687	1.0101
$\frac{1}{E^2[x^2(k)]}$	$8.2645\ 10^{3}$	120.7584	15.8728	0.9980
(6.40)	$1.1644\ 10^{4}$	134.6784	16.5959	1.0098

$$
\begin{aligned}
E\left[\frac{1}{\gamma^2(k)}\right] &\approx \frac{3E^2[\gamma^2(k)] - 2E^4[\gamma(k)]}{E^3[\gamma^2(k)]} \\
E\left[\frac{1}{\gamma^4(k)}\right] &\approx \frac{3\{3E^2[\gamma^2(k)] - 2E^4[\gamma(k)]\}^2 - 2E^4[\gamma^2(k)]}{\{3E^2[\gamma^2(k)] - 2E^4[\gamma(k)]\}^3} \\
E\left[\frac{1}{\| \boldsymbol{e}_f(k) \|^2}\right] &\approx \frac{3E^2[\| \boldsymbol{e}_f(k) \|^2] - 2E^4[\| \boldsymbol{e}_f(k) \|]}{E^3[\| \boldsymbol{e}_f(k) \|^2]} \\
E\left[\frac{1}{\| \boldsymbol{e}_f(k) \|^4}\right] &\approx \frac{3\{3E^2[\| \boldsymbol{e}_f(k) \|^2] - 2E^4[\| \boldsymbol{e}_f(k) \|]\}^2 - 2E^4[\| \boldsymbol{e}_f(k) \|^2]}{\{3E^2[\| \boldsymbol{e}_f(k) \|^2] - 2E^4[\| \boldsymbol{e}_f(k) \|]\}^3}
\end{aligned}
\tag{6.41}
$$

6.4 Simulation Results

In order to verify the accuracy of the formulae presented in this chapter, a computer simulation of the FQR_PRI_B algorithm running in a fixed-point arithmetic environment was carried out. The experiment consisted of a system identification set up with input signal and measurement noise with normal distribution, zero-mean and variances $\sigma_x^2 = 10^{-3}$ and $\sigma_n^2 = 10^{-7}$, respectively. The unknown FIR system has

order $N = 4$ and the forgetting factor was chosen equal to $\lambda = 0.95$. In the 500 independent runs, the FQR_PRI_B algorithm was simulated with 1000 iterations and the last 800 were averaged to obtain the results presented here. Two's complement rounding was used with $B = 15$ bits (excluding the sign bit) while infinite-precision was implemented with 64 bits floating-point.

All theoretical results of the accumulated quantization errors shown in the following tables were obtained from the analysis relations where the "unknown" quantities — variables not analyzed — were taken from the simulations with the purpose of verifying the expression. Moreover, the column DIFFERENCE was obtained as the absolute value of the difference of the simulated value in dB minus the theoretical value in dB.

Table 6.2 below shows the results of (6.26).

Table 6.2: Mean Squared Value of $\Delta e_{fq_1}^{(i)}(k)$.

EXPRESSION	SIMULATED	THEORETICAL	DIFFERENCE
$E\left\{[\Delta e_f^{(1)} q_1(k)]^2\right\}$	$0.0347\ 10^{-8}$	$0.0330\ 10^{-8}$	0.2260
$E\left\{[\Delta e_f^{(2)} q_1(k)]^2\right\}$	$0.0692\ 10^{-8}$	$0.0655\ 10^{-8}$	0.2397
$E\left\{[\Delta e_f^{(3)} q_1(k)]^2\right\}$	$0.1041\ 10^{-8}$	$0.0982\ 10^{-8}$	0.2538
$E\left\{[\Delta e_f^{(4)} q_1(k)]^2\right\}$	$0.1389\ 10^{-8}$	$0.1307\ 10^{-8}$	0.2643
$E\left\{[\Delta e_f^{(5)} q_1(k)]^2\right\}$	$0.1730\ 10^{-8}$	$0.1630\ 10^{-8}$	0.2576

Following, Table 6.3 shows the results of (6.28).

Table 6.4 displays the results of $E\{[\Delta aux_0]^2\}$ with the help of different relations. Note the improvement of the results with the refined expression. It is important to mention that $E[\gamma(k)]$ and $E[\| \boldsymbol{e}_f(k) \|]$ in (6.41) were not taken from the simulation results but calculated from $E[\gamma^2(k)]$, $E[\| \boldsymbol{e}_f(k) \|^2]$ and the variances of $\gamma(k)$ and $\| \boldsymbol{e}_f(k) \|$. These variances were actually obtained from the simulations. This procedure was used in order to guarantee that $E[x^2] > E^2[x]$ for $E[x^2] = \sigma_x^2 + E^2[x]$.

Table 6.3: Mean Squared Value of $\Delta d_{fq2_i}(k)$.

EXPRESSION	SIMULATED	THEORETICAL	DIFFERENCE
$E\{[\Delta d_{fq2_1}(k)]^2\}$	$0.3330\ 10^{-8}$	$0.3813\ 10^{-8}$	0.5875
$E\{[\Delta d_{fq2_2}(k)]^2\}$	$0.3128\ 10^{-8}$	$0.3531\ 10^{-8}$	0.5258
$E\{[\Delta d_{fq2_3}(k)]^2\}$	$0.2914\ 10^{-8}$	$0.3252\ 10^{-8}$	0.4771
$E\{[\Delta d_{fq2_4}(k)]^2\}$	$0.2608\ 10^{-8}$	$0.2976\ 10^{-8}$	0.5739
$E\{[\Delta d_{fq2_5}(k)]^2\}$	$0.2458\ 10^{-8}$	$0.2704\ 10^{-8}$	0.4151

Table 6.4: Mean Squared Value of Δaux_0.

TYPE OF RESULT	$E\{[\Delta aux_0]^2\}$
SIMULATED	$3.2229\ 10^{-7}$
EQUATION (6.35)	$2.4923\ 10^{-7}$
DIFFERENCE	1.1165
EQUATIONS (6.34) AND (6.41)	$2.7751\ 10^{-7}$
DIFFERENCE	0.6497
EQUATIONS (6.34) AND (6.41) IMPROVED[2]	$2.8572\ 10^{-7}$
DIFFERENCE	0.5230

6.5 Conclusions

This chapter aimed an introduction to the topic finite-precision analysis of the fast QR algorithms based on the backward prediction errors (FQR_POS_B and FQR_PRI_B algorithms). A survey on the infinite-precision analysis for both algorithms were presented and, although not mentioned before, every expression was checked with computer simulations and validated for a mean squared value of all variables within one dB. The fixed-point quantization analysis was carried out for a matrix equation which is common for both algorithms and it was tested for

[2]Using results of simulations instead of its corresponding theoretical values.

the FQR_PRI_B algorithm with excellent results. Another expression was analyzed showing a possible problem typically found in the analysis procedure. An expression for this analysis was developed using the averaging principle and an alternative approximation was derived to improve the results if the first-order statistics (mean) of some variables are known. In the present case the approximations derived here were only useful to validate part of the analysis since the means of some variables are not available. A simulation was carried out to evaluate the partial analysis presented and the results show that the theoretical formulae agree well with those obtained in the experiment. The author intends to complete the analysis starting from the contribution given here as well as investigate its extension to the floating-point case.

Chapter 7

Conclusions and Suggestions

This chapter summarizes the results of the thesis and highlights a variety of worth investigating problems for possible future research. The work focused on two families of adaptive filters and two new algorithms have been introduced in this thesis: the BiNormalized Data-Reusing Least Mean-Squares (BNDR-LMS) algorithm and the Fast QR based on the updating of the *a PRIori* Forward prediction errors (FQR_PRI_F) algorithm. The investigation of these algorithms led to the study of a number of relevant research results such as mean-square error analysis, constrained version of the BNDR-LMS algorithm and its application, lattice QR based version, and finite precision analysis of the fast QR algorithms; all of them addressed throughout the thesis.

7.1 Conclusions

The first chapter of the thesis presented a brief review of the adaptive filtering basic theory with special attention to the LMS-like and QR decomposition algorithms.

Chapter 2 presented the BNDR-LMS algorithm and it was verified through computer simulations that this algorithm compares favorably with other normalized LMS-like algorithms when the input signal is correlated. For this algorithm, convergence analysis in the mean-squared was presented for white input signals as well as its extension to the case of colored input signal. A simple model for the input-

signal vector which imparts simplicity and tractability to the analysis of second-order statistics is employed. The simulation results validated the analysis and the ensuing assumptions. It can be concluded that the proposed algorithm (BNDR-LMS) presents higher convergence speed for colored input signals than other LMS based algorithms employing data-reusing with similar computational complexity and that this higher efficiency is better observed in low noise level environment.

In Chapter 3, a constrained version of the BNDR-LMS algorithm was derived in order to apply this algorithm to the field of mobile communications. In particular, an example using this algorithm in a direct-sequence code-division multiple access (DS-CDMA) mobile receiver scenario was carried out, and a step-size optimization was proposed to accomplish the requirements of fast convergence and minimum MSE. We conclude that, in environments where the (observation or modeling) noise level is too high, the use of a variable step-size is necessary.

In chapter 4, the QR decomposition algorithms using Givens rotations were presented in a tutorial form with the special objective of classifying the existing algorithms in a comprehensive framework. This chapter derived the new FQR_PRI_F algorithm and its relations with other members of the fast QR algorithms family. The previously proposed algorithms were also rederived using the same notation. The equations for the four fast QR algorithms classified in this chapter were also provided along with their detailed algorithmic descriptions in appendix. According to what was studied, the conclusion is that the FQR_PRI_B algorithm presents the best performance (in terms of computational load and numerical behavior) among the fast QR algorithms classified.

Chapter 5 presented the fully lattice versions of the FQR_POS_B and FQR_PRI_B algorithms. The simulation results showed that the performance of the lattice versions in a finite-precision implementation is comparable with the original algorithms.

Finally, Chapter 6 addresses the finite precision analysis of the fast QR algorithms using backward prediction errors (*a priori* and a posteriori). The finite precision analysis results already available from previous works are summarized with the same uniform notation used throughout the thesis as well as a contribution to the

fixed point error analysis is given with the study of the behavior of some expressions in this finite precision environment.

7.2 Suggestions for Future Research

This section gives some suggestions for further research.

Concerning the BNDR-LMS algorithm, it seems to be interesting the investigation of the link between this algorithm and its extensions (i-normalized DR-LMS) to the orthogonal-projection algorithm in addition to their relationships with the RLS algorithm.

Another promising field of interest is the finite precision performance and analysis of this algorithm.

From Chapter 2, it was clear that the model and the analyses used there can be readily extended to other data-reusing algorithms that have not been considered in the past due to exceeding complexity in the analysis expressions. This is another contribution of this work that can be subject of future research.

Another suggestion is the possible improvement of the performance of the adaptive step-size if the colored input signal is considered in the derivation of the optimal sequence.

Last, but not least, a few suggestions for further investigation concerning the fast QR algorithms are pointed out. The completion of the finite precision analysis is a topic of major interest.

Also the same notation and framework used in Chapter 4 could be extended to derive the QR lattices [21, 22, 48] and verify their relationship with the Lattice RLS algorithms.

Some variants of the FQR algorithms (some of them already developed and available in the technical literature) also require some attention such as multichannel versions and constrained versions with their applications.

References

[1] HAYKIN, S., *Adaptive Filter Theory*. 2 ed. Englewood Cliffs, NJ, USA, Prentice Hall, 1991.

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

[3] ROY, S., SHYNK, J. J., "Analysis of the data-reusing LMS algorithm". In: *Proceedings of the 32 Midwest Symposium on Circuits and Systems*, pp. 1127–1130, Urbana-Champaign, IL, USA, 1989.

[4] SLOCK, D. T., "On the convergence behavior of the LMS and the normalized LMS algorithms", *IEEE Transactions on Signal Processing*, v. 41, pp. 2811–2825, September 1993.

[5] JENKINS, W. K, HULL, A. W., STRAIT, J. C. et al., *Advanced Concepts in Adaptive Signal Processing*. Norwell, MA, USA, Kluwer Academic Publishers, 1996.

[6] DINIZ, P. S. R., DE CAMPOS, M. L. R., ANTONIOU, A., "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor", *IEEE Transactions on Signal Processing*, v. 43, pp. 617–627, March 1995.

[7] YASSA, F. F., "Optimality in the choice of the convergence factor for gradient-based adaptive algorithms", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, v. ASSP-35, pp. 48–59, January 1987.

[8] LEE, D. L., MORF, M., FRIEDLANDER, B., "Recursive least squares ladder estimation algorithms", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, v. ASSP-29, pp. 627–641, June 1981.

[9] CIOFFI, J. M., KAILATH, T., "Fast, recursive-least-squares transversal filters for adaptive filtering", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, v. ASSP-32, pp. 302–337, April 1984.

[10] SCHNAUFER, B. A., *Practical Techniques for Rapid and Reliable Real-Time Adaptive Filtering*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, USA, 1995.

[11] WIDROW, B., STEARNS, S. D., *Adaptive Signal Processing*. Englewood-Cliffs, NJ, USA, Prentice-Hall, 1985.

[12] GOODWIN, G. C., SIN, S. K., *Adaptive Filtering Prediction and Control*. Englewood-Cliffs, NJ, USA, Prentice-Hall, 1984.

[13] CIOFFI, J. M., "The fast Householder filters RLS adaptive filter". In: *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, pp. 1619–1622, Albuquerque, NM, USA, 1990.

[14] LIU, K. J. R.,HSIEH, S.-F., YAO, K., "Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation", *IEEE Transactions on Signal Processing*, v. 40, pp. 946–957, April 1992.

[15] ALEXANDER, S. T., GHIRNIKAR, A. L., "A method for recursive least squares adaptive filtering based upon an inverse QR decomposition", *IEEE Transactions on Signal Processing*, v. SP-41, pp. 20–30, January 1993.

[16] CIOFFI, J. M., "The fast adaptive ROTOR's RLS algorithm", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, v. ASSP-38, pp. 631–653, April 1990.

[17] REGALIA, P. A., BELLANGER, M. G., "On the duality between fast QR methods and lattice methods in least squares adaptive filtering", *IEEE Transactions on Signal Processing*, v. SP-39, pp. 879–891, April 1991.

[18] MIRANDA, M. D., GERKEN, M., "A hybrid QR-lattice least squares algorithm using a priori errors". In: *Proceedings of the 38 Midwest Symposium on Circuits and Systems*, pp. 983–986, Rio de Janeiro, RJ, Brazil, August 1995.

[19] RONTOGIANNIS, A. A., THEODORIDIS, S., "New fast inverse QR least squares adaptive algorithms". In: *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, pp. 1412–1415, Detroit, MI, USA, 1995.

[20] APOLINÁRIO JR., J. A., DINIZ, P. S. R., "A new fast QR algorithm based on a priori errors", *IEEE Signal Processing Letters*, v. 4, pp. 307–309, November 1997.

[21] LING, F., "Givens rotation based least squares lattice and related algorithms", *IEEE Transactions on Signal Processing*, v. SP-39, pp. 1541–1551, July 1991.

[22] PROUDLER, I. K., MCWHIRTER, J. G., SHEPARD, T. J., "Computationally efficient QR decomposition approach to least squares adaptive filtering", *IEE Proceedings-F*, v. 138, pp. 341–353, August 1991.

[23] APOLINÁRIO JR., J. A., DE CAMPOS, M. L. R., DINIZ, P. S. R., "The binormalized data-reusing LMS algorithm", submitted to the *IEEE Transactions on Signal Processing*, 1998.

[24] APOLINÁRIO JR., J. A., DE CAMPOS, M. L. R., DINIZ, P. S. R., "The binormalized data-reusing LMS algorithm". In: *Proceedings of the XV Simpósio Brasileiro de Telecomunicações*, pp. 77–80, Recife, PE, Brazil, September 1997.

[25] APOLINÁRIO JR., J. A., DE CAMPOS, M. L. R., DINIZ, P. S. R., "Convergence analysis of the binormalized data-reusing LMS algorithm". In: *Proceedings of the European Conference on Circuit Theory and Design*, pp. 972–977, Budapest, Hungary, September 1997.

[26] MAZO, J. E., "On the independence theory of equalizer convergence", *The Bell System Technical Journal*, v. 58, pp. 962–993, May–June 1979.

[27] DE CAMPOS, M. L. R., ANTONIOU, A., "A new quasi-Newton adaptive filtering algorithm", *IEEE Transactions on Circuits and Systems — Part II*, v. 44, pp. 924–934, November 1997.

[28] D. T. SLOCK, "On the convergence behavior of the LMS and NLMS algorithms". In: *V European Signal Processing Conference*, (Barcelona, Spain), pp. 197–200, 1990.

[29] PAPOULIS, A., *Probability, Random Variables, and Stochastic Processes*. 3 ed., McGraw-Hill, 1991.

[30] FROST, III, O. L., "An algorithm for linearly constrained adaptive array processing", *Proceedings of the IEEE*, v. 60, pp. 926–935, August 1972.

[31] GRIFFITHS, L. J., JIM, C. W., "An alternative approach to linearly constrained adaptive beamforming", *IEEE Transactions on Antennas and Propagation*, v. AP-30, pp. 27–34, January 1982.

[32] RESENDE, L. S., ROMANO, J. M. T., BELLANGER, M. G., "A fast least-squares algorithm for linearly constrained adaptive filtering", *IEEE Transactions on Signal Processing*, v. 44, pp. 1168–1174, May 1996.

[33] APOLINÁRIO JR., J. A., DINIZ, P. S. R., LAAKSO, T. I. et al., "Step-size optimization of the BNDR-LMS algorithm". (accepted) *IX European Signal Processing Conference*, Island of Rhodes, Greece, August 1998.

[34] APOLINÁRIO JR., J. A., WERNER, S., DINIZ, P. S. R. et al., "Constrained normalized adaptive filters for CDMA mobile communications". (accepted) *IX European Signal Processing Conference*, Island of Rhodes, Greece, August 1998.

[35] SCHLEGEL, C., ROY, S., ALEXANDER, P. D. et al., "Multiuser projection receivers", *IEEE Journal on Selected Areas in communications*, v. 14, October 1996.

[36] GOLUB, G. H., VAN LOAN, C. F., *Matrix Computations*. 2 ed. Baltimore, MD, USA, John Hopkins University Press, 1989.

[37] MIRANDA, M. D., *Sobre algoritmos dos mínimos quadrados rápidos, recursivos na ordem, que utilizam triangularização ortogonal.* PhD Thesis, Universidade de São Paulo, São Paulo, SP, Brazil, 1996.

[38] PAN, C.-T., PLEMMONS, R. J., "Least squares modifications with inverse factorizations: parallel implications", *Journal of Computational and Applied Mathematics*, v. 27, pp. 109–127, 1989.

[39] PARK, P. G., KAILATH, T., "A lattice algorithm dual to the extended inverse QR algorithm", *Signal Processing*, v. 47, pp. 115–133, November 1995.

[40] MIRANDA, M. D., GERKEN, M., "A hybrid leat squares QR-lattice algorithm using a priori errors", *IEEE Transactions on Signal Processing*, v. 45, pp. 2900–2911, December 1997.

[41] BELLANGER, M. G., "The FLS-QR algorithm for adaptive filtering", *Signal Processing*, v. 17, pp. 291–304, August 1989.

[42] TERRÉ, M., BELLANGER, M. G., "A sytolic QRD-based algorithm for adaptive filtering and its implementation". In: *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, Adelaide, Australia, 1994.

[43] MIRANDA, M. D., AGUAYO, L., GERKEN, M., "Performance of the a priori and a posteriori QR-LSL algorithms in a limited precision environment". In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, Germany, April 1997.

[44] DINIZ, P. S. R., SIQUEIRA, M. G., "Fixed-point error analysis of the QR-recursive least square algorithm", *IEEE Transactions on Circuits and Systems — Part II*, v. 42, pp. 334–348, May 1995.

[45] SIQUEIRA, M. G., DINIZ, P. S. R., ALWAN, A., "Infinite precision analysis of the fast QR decomposition RLS algorithm". In: *Proceedings of the IEEE International Symposium on Circuits and Systems*, London, UK, 1994.

[46] SAMSON, C., REDDY, V., "Fixed point error analysis of the normalized ladder algorithm", *IEEE Transactions on Acoustic, Speech, and Signal Processing*, v. 31, pp. 1177–1191, October 1983.

[47] HUBING, N. E., ALEXANDER, S. T., "Statistical analysis of initialization methods for RLS adaptive filters", *IEEE Transactions on Signal Processing*, v. 39, pp. 1793–1804, August 1991.

[48] DESBOUVRIES, F., REGALIA, P. A., "A minimal, Givens rotation based FRLS lattice algorithm". In: *Proceedings of the VIII European Signal Processing Conference*, 1996.

Appendix A

1. Equation (2.24):

$$\begin{aligned} & E\left[\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\right] \\ & \quad = E\left[\sum_{i=0}^{N}\sum_{j=0}^{N} x(k-i)x(k-1-i)x(k-1-j)x(k-j)\right] \\ & \quad = \sum_{i=0}^{N} E\left[x^2(k-i)x^2(k-1-i)\right] \\ & \quad = (N+1)(\sigma_x^2)^2 \end{aligned} \tag{A.1}$$

2. Equation (2.25):

$$\begin{aligned} & E\left\{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - \left[\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\right]^2\right\} \\ & = E\left[\sum_{i=0}^{N}\sum_{j=0}^{N} x^2(k-i)x^2(k-1-j)\right] - E\left\{\left[\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\right]^2\right\} \\ & = \sum_{i=0}^{N} E\left[x^2(k-i)\right] \sum_{j=0,\, j\neq i-1}^{N} E\left[x^2(k-1-j)\right] \\ & \quad + \sum_{i=1}^{N} E\left[x^4(k-i)\right] - E\left\{\left[\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\right]^2\right\} \\ & = (N^2+N+1)(\sigma_x^2)^2 + NE[x^4(k)] - (N+1)(\sigma_x^2)^2 \end{aligned} \tag{A.2}$$

   For stationary Gaussian-distributed signals, using the fourth-moment factoring theorem we have $E[x^4(k)] = 3(\sigma_x^2)^2$ [29] and, therefore,

$$E\left\{\|\boldsymbol{x}(k)\|^2\|\boldsymbol{x}(k-1)\|^2 - \left[\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\right]^2\right\} = N(N+3)(\sigma_x^2)^2 \tag{A.3}$$

3. Equation (2.26):

   Let $[\cdot]_{ij}$ be the (i,j) element of matrix $[\cdot]$, then

$$\begin{aligned} & E\left\{\left[\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right]_{ij}\right\} \\ & \qquad = E\left[x(k-1-i)x(k-j)\sum_{l=0}^{N} x(k-1-l)x(k-l)\right] \end{aligned} \tag{A.4}$$

For Gaussian-distributed signals we may use the fourth-moment factoring theorem to obtain

$$
\begin{aligned}
&E\left\{\left[\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right]_{ij}\right\} \\
&\quad = \sum_{l=0}^{N} \{E\left[x(k-1-i)x(k-j)\right] E\left[x(k-1-l)x(k-l)\right] \\
&\quad\quad + E\left[x(k-1-i)x(k-1-l)\right] E\left[x(k-j)x(k-l)\right] \\
&\quad\quad + E\left[x(k-1-i)x(k-l)\right] E\left[x(k-j)x(k-1-l)\right]\} \\
&\quad = \begin{cases} (\sigma_x^2)^2, & i=j \quad \text{or} \quad i=j-2 \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{A.5}
$$

4. Equation (2.27):

   Once again, using the fact that for stationary Gaussian-distributed signals $E[x^4(k)] = 3(\sigma_x^2)^2$ [29], we have

$$
\begin{aligned}
E\left\{\left[\|\boldsymbol{x}(k-1)\|^2\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right]_{ij}\right\} &= E\left[\sum_{l=0}^{N} x^2(k-1-l)x(k-i)x(k-j)\right] \\
&= \begin{cases} N(\sigma_x^2)^2 + E[x^4(k)], & i=j \\ 0, & \text{otherwise} \end{cases} \\
&= \begin{cases} (N+3)(\sigma_x^2)^2, & i=j \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{A.6}
$$

Appendix B

1. Equation (2.39):

   In the derivation of (2.39) $\boldsymbol{x}(k)$ and $\boldsymbol{x}(k-1)$ were replaced by $s_k r_k V_k$ and $s_{k-1} r_{k-1} V_{k-1}$, respectively, with $V_k \perp V_{k-1}$. Therefore, $\boldsymbol{x}^T(k)\boldsymbol{x}(k-1) = 0$. Furthermore, a second-order approximation for $E[1/r_k^2]$ was used [4], i.e.,

   $$E\left[\frac{1}{\|\boldsymbol{x}(k)\|^2}\right] = E\left[\frac{1}{\|\boldsymbol{x}(k-1)\|^2}\right] = E\left[\frac{1}{r_k^2}\right] \approx \frac{1}{(N+2-\nu_x)\sigma_x^2} \tag{B.1}$$

   where ν_x is the kurtosis of the input signal.

   For $\boldsymbol{R} = \sigma_x^2 \boldsymbol{I}$, using (2.33) and (2.34) the expression for $\Delta\xi(k)$ may be rewritten as

   $$\begin{aligned}
   \Delta\xi(k+1) &= \sigma_x^2 \mathrm{tr}\left\{\mathrm{cov}[\Delta\boldsymbol{w}(k+1)]\right\} \\
   &= \sigma_x^2 \mathrm{tr}\left\{E\left[\Delta\boldsymbol{w}(k+1)\Delta\boldsymbol{w}^T(k+1)\right]\right\} \\
   &= \sigma_x^2 \mathrm{tr}\left(E\left\{[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)[\boldsymbol{I}+\mu\boldsymbol{A}]\right\}\right) \\
   &\quad + \sigma_x^2 \mathrm{tr}\left(E\left\{\mu[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\boldsymbol{b}^T\right\}\right) \\
   &\quad + \sigma_x^2 \mathrm{tr}\left(E\left\{\mu\boldsymbol{b}\Delta\boldsymbol{w}^T(k)[\boldsymbol{I}+\mu\boldsymbol{A}]\right\}\right) \\
   &\quad + \sigma_x^2 \mathrm{tr}\left(E\left[\mu^2\boldsymbol{b}\boldsymbol{b}^T\right]\right) \\
   &= \rho_1 + \rho_2 + \rho_3 + \rho_4
   \end{aligned} \tag{B.2}$$

Evaluating each of these terms separately we obtain

$$
\begin{aligned}
\rho_1 &= \sigma_x^2 \mathrm{tr}\left(E\left\{[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)[\boldsymbol{I}+\mu\boldsymbol{A}]\right\}\right) \\
&= \sigma_x^2 \mathrm{tr}\left\{\mathrm{cov}[\Delta\boldsymbol{w}(k)]\right\} \\
&\quad - \mu\sigma_x^2 \mathrm{tr}\left\{E\left[\frac{\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)}{\|\boldsymbol{x}(k-1)\|^2}\right]\right\} \\
&\quad - \mu\sigma_x^2 \mathrm{tr}\left\{E\left[\frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)}{\|\boldsymbol{x}(k)\|^2}\right]\right\} \\
&\quad - \mu\sigma_x^2 \mathrm{tr}\left\{E\left[\frac{\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{\|\boldsymbol{x}(k-1)\|^2}\right]\right\} \\
&\quad - \mu\sigma_x^2 \mathrm{tr}\left\{E\left[\frac{\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{\|\boldsymbol{x}(k)\|^2}\right]\right\} \\
&\quad + \mu^2\sigma_x^2 \mathrm{tr}\left(E\left\{\frac{\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{\left[\|\boldsymbol{x}(k-1)\|^2\right]^2}\right\}\right) \\
&\quad + \mu^2\sigma_x^2 \mathrm{tr}\left\{E\left[\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\right]\right\} \\
&\quad + \mu^2\sigma_x^2 \mathrm{tr}\left\{E\left[\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right]\right\} \\
&\quad + \mu^2\sigma_x^2 \mathrm{tr}\left(E\left\{\frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{\left[\|\boldsymbol{x}(k)\|^2\right]^2}\right\}\right) \\
&= \psi_1 + \psi_2 + \cdots + \psi_9
\end{aligned}
\tag{B.3}
$$

where

$$
\begin{aligned}
\psi_1 &= \sigma_x^2 \mathrm{tr}\left\{\mathrm{cov}[\Delta\boldsymbol{w}(k)]\right\} \\
&= \Delta\xi(k)
\end{aligned}
\tag{B.4}
$$

$$
\begin{aligned}
\psi_2 &= -\mu\sigma_x^2 E\left\{\mathrm{tr}\left[\frac{\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)}{\|\boldsymbol{x}(k-1)\|^2}\right]\right\} \\
&= -\mu\sigma_x^2 E\left[\frac{\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)}{\|\boldsymbol{x}(k-1)\|^2}\right] \\
&= -\mu\sigma_x^2 E\left\{\frac{\left[\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\right]^2}{\|\boldsymbol{x}(k-1)\|^2}\right\} \\
&= -\mu\sigma_x^2 E\left\{\frac{\left[(1-\mu)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k-1)+\mu n(k-1)\right]^2}{\|\boldsymbol{x}(k-1)\|^2}\right\} \\
&= -\mu(1-\mu)^2\sigma_x^2 E\left[\Delta\boldsymbol{w}^T(k-1)\boldsymbol{V}_{k-1}\boldsymbol{V}_{k-1}^T\Delta\boldsymbol{w}(k-1)\right]-\mu^3\sigma_x^2 E\left[\frac{n^2(k-1)}{r_k^2}\right] \\
&= -\frac{\mu(1-\mu)^2\sigma_x^2}{(N+1)}\mathrm{tr}\left\{\mathrm{cov}\left[\Delta\boldsymbol{w}(k-1)\right]\right\}-\mu^3\sigma_n^2\sigma_x^2 E\left[\frac{1}{r_k^2}\right] \\
&= -\frac{\mu(1-\mu)^2}{(N+1)}\Delta\xi(k-1)-\frac{\mu^3\sigma_n^2}{(N+2-\nu_x)}
\end{aligned}
\tag{B.5}
$$

$$
\begin{aligned}
\psi_3 &= -\mu\sigma_x^2\mathrm{tr}\left\{E\left[\frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)}{\|\boldsymbol{x}(k)\|^2}\right]\right\} \\
&= -\mu\sigma_x^2\mathrm{tr}\left\{E\left[\boldsymbol{V}_k\boldsymbol{V}_k^T\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\right]\right\} \\
&= -\frac{\mu\sigma_x^2}{N+1}\mathrm{tr}\left\{\mathrm{cov}\left[\Delta\boldsymbol{w}(k)\right]\right\} \\
&= -\frac{\mu}{(N+1)}\Delta\xi(k)
\end{aligned}
\tag{B.6}
$$

Recalling that $\mathrm{tr}[\boldsymbol{AB}] = \mathrm{tr}[\boldsymbol{BA}]$ for any square matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we find that

$$
\psi_4 = \psi_2 \tag{B.7}
$$

$$
\psi_5 = \psi_3 \tag{B.8}
$$

$$\begin{aligned}
\psi_6 &= \mu^2\sigma_x^2 \mathrm{tr}\left(E\left\{\frac{\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)}{\left[\|\boldsymbol{x}(k-1)\|^2\right]^2}\right\}\right) \\
&= \mu^2\sigma_x^2 E\left\{\frac{\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)}{\left[\|\boldsymbol{x}(k-1)\|^2\right]^2}\right\} \\
&= \mu^2\sigma_x^2 E\left[\frac{\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)}{\|\boldsymbol{x}^T(k-1)\|^2}\right] \\
&= -\mu\psi_2
\end{aligned} \tag{B.9}$$

$$\begin{aligned}
\psi_7 &= \mu^2\sigma_x^2 \mathrm{tr}\left\{E[\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)]\right\} \\
&= \mu^2\sigma_x^2 \mathrm{tr}\left\{E[\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)]\right\} \\
&= 0
\end{aligned} \tag{B.10}$$

$$\begin{aligned}
\psi_8 &= \mu^2\sigma_x^2 \mathrm{tr}\left\{E[\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)]\right\} \\
&= \mu^2\sigma_x^2 \mathrm{tr}\left\{E[\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{x}(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)]\right\} \\
&= 0
\end{aligned} \tag{B.11}$$

$$\begin{aligned}
\psi_9 &= \mu^2\sigma_x^2 \mathrm{tr}\left(E\left\{\frac{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)}{\left[\|\boldsymbol{x}(k)\|^2\right]^2}\right\}\right) \\
&= \mu^2\sigma_x^2 E\left\{\frac{\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)}{\left[\|\boldsymbol{x}(k)\|^2\right]^2}\right\} \\
&= \mu^2\sigma_x^2 E\left[\frac{\Delta\boldsymbol{w}^T(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)}{\|\boldsymbol{x}(k)\|^2}\right] \\
&= \mu^2\sigma_x^2 E\left[\Delta\boldsymbol{w}^T(k)\boldsymbol{V}_k\boldsymbol{V}_k^T\Delta\boldsymbol{w}(k)\right] \\
&= \mu^2\sigma_x^2 \frac{\mathrm{tr}\left\{\mathrm{cov}\left[\Delta\boldsymbol{w}(k)\right]\right\}}{N+1} \\
&= \frac{\mu^2}{N+1}\Delta\xi(k)
\end{aligned} \tag{B.12}$$

Therefore,

$$\begin{aligned}
\rho_1 &= \sigma_x^2 \mathrm{tr}\left(E\left\{[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\Delta\boldsymbol{w}^T(k)[\boldsymbol{I}+\mu\boldsymbol{A}]\right\}\right) \\
&= \left(1+\frac{\mu(\mu-2)}{N+1}\right)\Delta\xi(k) + \frac{(1-\mu)^2\mu(\mu-2)}{N+1}\Delta\xi(k-1) + \frac{\mu^3(\mu-2)}{N+2-\nu_x}\sigma_n^2
\end{aligned} \tag{B.13}$$

Similarly,

$$\begin{aligned}
\rho_2 &= \mu\sigma_x^2 E\left\{\boldsymbol{b}^T[\boldsymbol{I}+\mu\boldsymbol{A}]\Delta\boldsymbol{w}(k)\right\} \\
&= \mu\sigma_x^2 E\left[\frac{n(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)}{\|\boldsymbol{x}(k)\|^2}\right] \\
&\quad + \mu\sigma_x^2 E\left[\frac{n(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)}{\|\boldsymbol{x}(k-1)\|^2}\right] \\
&\quad - \mu^2\sigma_x^2 E\left[\frac{n(k)\boldsymbol{x}^T(k)\Delta\boldsymbol{w}(k)}{\|\boldsymbol{x}(k)\|^2}\right] \\
&\quad - \mu^2\sigma_x^2 E\left[\frac{n(k-1)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k)}{\|\boldsymbol{x}(k-1)\|^2}\right] \\
&= \mu^2(1-\mu)\sigma_x^2 E\left\{\frac{n(k-1)\left[(1-\mu)\boldsymbol{x}^T(k-1)\Delta\boldsymbol{w}(k-1)+\mu n(k-1)\right]}{\|\boldsymbol{x}(k-1)\|^2}\right\} \\
&= \mu^2(1-\mu)\sigma_x^2 E\left[n^2(k-1)\right]E\left[\frac{1}{r_{k-1}^2}\right] \\
&= \frac{\mu^2(1-\mu)}{(N+2-\nu_x)}\sigma_n^2 \\
&= \rho_3
\end{aligned} \tag{B.14}$$

$$\begin{aligned}
\rho_4 &= \mu^2\sigma_x^2 E\left[\frac{n^2(k)}{\|\boldsymbol{x}(k)\|^2}+\frac{n^2(k-1)}{\|\boldsymbol{x}(k-1)\|^2}\right] \\
&= \frac{2\mu^2\sigma_n^2}{(N+2-\nu_x)}
\end{aligned} \tag{B.15}$$

From (B.13)–(B.15) the difference equation for $\Delta\xi(k)$ is finally obtained as in (2.39)

$$\begin{aligned}
\Delta\xi(k+1) = \left[1+\frac{\mu(\mu-2)}{N+1}\right]\Delta\xi(k) + \frac{(1-\mu)^2\mu(\mu-2)}{N+1}\Delta\xi(k-1) \\
+ \frac{\mu^2(\mu-2)^2}{N+2-\nu_x}\sigma_n^2
\end{aligned} \tag{B.16}$$

Appendix C

1. The FQR_POS_F Algorithm:

FQR_POS_F

Initialization:
$\| \boldsymbol{e}_f(k) \| = \epsilon$ = small positive value;
$\boldsymbol{d}_{fq2}(k) = \boldsymbol{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\boldsymbol{\theta}(k) = cos\boldsymbol{\theta}'_b(k) = \text{ones}(N+1,1)$;
$sin\boldsymbol{\theta}(k) = sin\boldsymbol{\theta}'_b(k) = \text{zeros}(N+1,1)$;
$\gamma(k) = 1$;
$\boldsymbol{f}(k) = \text{zeros}(N+1,1)$;
for $k = 1, 2, \ldots$
{ $e^{(0)}_{fq_1}(k+1) = x(k+1)$;
for $i = 1 : N+1$
{ $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) - sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_i}(k)$;
$d_{fq2_i}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_i}(k)$;
}
$e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;
$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$;
$cos\theta_f(k+1) = \lambda^{1/2} \| \boldsymbol{e}_f(k) \| / \| \boldsymbol{e}_f(k+1) \|$;
$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \boldsymbol{e}_f(k+1) \|$;
$\boldsymbol{c}(k+1) = [1; \text{zeros}(N+1,1)]$;
for $i = 1 : N+1$
{ $c_{N+3-i} = -sin\theta'_{b_{N+1-i}}(k) c_1$;
$c_1 = cos\theta'_{b_{N+1-i}}(k) c_1$;
}
$\boldsymbol{c}_{aux} = [0;\ \boldsymbol{c}(k+1)]$;
for $i = 1 : N+1$
{ $oldvalue = c_{aux_{i+1}}$;
$c_{aux_{i+1}} = sin\theta_{i-1}(k) c_{aux_1} + cos\theta_{i-1}(k) c_{aux_{i+1}}$;
$c_{aux_1} = cos\theta_{i-1}(k) c_{aux_1} - sin\theta_{i-1}(k) oldvalue$;
}
$oldvalue = c_{aux_1}$;

(Continuation of the FQR_POS_F Algorithm)

$c_{aux_1} = cos\theta_f(k+1)c_{aux_1} - sin\theta_f(k+1)c_{aux_{N+3}}$;
$c_{aux_{N+3}} = sin\theta_f(k+1)oldvalue + cos\theta_f(k+1)c_{aux_{N+3}}$;
$\boldsymbol{c}(k+1) = \boldsymbol{c}_{aux}(2 : N+3)$;
for $i = 1 : N+1$
{ $oldvalue = c_1$;
$c_1 = \sqrt{c_1^2 + c_{i+1}^2}$;
$cos\theta'_{b_{i-1}}(k+1) = oldvalue/c_1$;
$sin\theta'_{b_{i-1}}(k+1) = -c_{i+1}/c_1$;
}
$\boldsymbol{f}^{(N+2)}(k+1) = [\boldsymbol{f}(k);\ sin\theta_f(k+1)\gamma(k)]$;
$aux_0 = f_1^{(N+2)}(k+1)$;
for $i = 1 : N+1$
{ $aux_i = cos\theta'_{b_{i-1}}(k+1)aux_{i-1} - sin\theta'_{b_{i-1}}(k+1)f_{i+1}^{(N+2)}(k+1)$;
$f_i(k+1) = sin\theta'_{b_{i-1}}(k+1)aux_{i-1} + cos\theta'_{b_{i-1}}(k+1)f_{i+1}^{(N+2)}(k+1)$;
}
$\gamma^{(0)}(k+1) = 1$;
for $i = 1 : N+1$
{ $sin\theta_{i-1}(k+1) = f_i(k+1)/\gamma^{(i-1)}(k+1)$;
$cos\theta_{i-1}(k+1) = \sqrt{1 - sin^2\theta_{i-1}(k+1)}$;
$\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1)\gamma^{(i-1)}(k+1)$;
}
$\gamma(k+1) = \gamma^{(N+1)}(k+1)$;
$e_{q_1}^{(0)}(k+1) = d(k+1)$;
for $i = 1 : N+1$
{ $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1)e_{q_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_i}(k)$;
$d_{q2_i}(k+1) = sin\theta_{i-1}(k+1)e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_i}(k)$;
}
$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$;
$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$;
}

2. The FQR_PRI_F Algorithm:

FQR_PRI_F

Initialization:
$\| \boldsymbol{e}_f(k) \| = \epsilon$ = small positive value;
$\boldsymbol{d}_{fq2}(k) = \boldsymbol{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\boldsymbol{\theta}(k) = cos\boldsymbol{\theta}'_b(k) = \text{ones}(N+1,1)$;
$sin\boldsymbol{\theta}(k) = sin\boldsymbol{\theta}'_b(k) = \text{zeros}(N+1,1)$;
$1/\gamma(k) = 1$;
$\boldsymbol{a}(k) = \text{zeros}(N+1,1)$;
for $k = 1, 2, \ldots$
{ $e^{(0)}_{fq_1}(k+1) = x(k+1)$;
for $i = 1 : N+1$
{ $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) - sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_i}(k)$;
$d_{fq2_i}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_i}(k)$;
}
$e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;
$e'_f(k+1) = e_{fq_1}(k+1)[1/\gamma(k)]$;
$\boldsymbol{a}^{(N+2)}(k+1) = [\boldsymbol{a}(k);\ \frac{e'_f(k+1)}{\lambda^{1/2}\|\boldsymbol{e}_f(k)\|}]$;
$aux_0 = a^{(N+2)}_1(k+1)$;
for $i = 1 : N+1$
{ $aux_i = cos\theta'_{b_{i-1}}(k) aux_{i-1} - sin\theta'_{b_{i-1}}(k) a^{(N+2)}_{i+1}(k+1)$;
$a_i(k+1) = sin\theta'_{b_{i-1}}(k) aux_{i-1} + cos\theta'_{b_{i-1}}(k) a^{(N+2)}_{i+1}(k+1)$;
}
$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$;
$cos\theta_f(k+1) = \lambda^{1/2} \| \boldsymbol{e}_f(k) \| / \| \boldsymbol{e}_f(k+1) \|$;
$sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \boldsymbol{e}_f(k+1) \|$
$\boldsymbol{c}(k+1) = [1; \text{zeros}(N+1,1)]$;
for $i = 1 : N+1$
{ $c_{N+3-i} = -sin\theta'_{b_{N+1-i}}(k) c_1$;
$c_1 = cos\theta'_{b_{N+1-i}}(k) c_1$;
}

(Continuation of the FQR_PRI_F Algorithm)

$\boldsymbol{c}_{aux} = [0;\ \boldsymbol{c}(k+1)]$;
for $i = 1 : N+1$
{ $oldvalue = c_{aux_{i+1}}$;
$c_{aux_{i+1}} = sin\theta_{i-1}(k)c_{aux_1} + cos\theta_{i-1}(k)c_{aux_{i+1}}$;
$c_{aux_1} = cos\theta_{i-1}(k)c_{aux_1} - sin\theta_{i-1}(k)oldvalue$;
}
$oldvalue = c_{aux_1}$;
$c_{aux_1} = cos\theta_f(k+1)c_{aux_1} - sin\theta_f(k+1)c_{aux_{N+3}}$;
$c_{aux_{N+3}} = sin\theta_f(k+1)oldvalue + cos\theta_f(k+1)c_{aux_{N+3}}$;
$\boldsymbol{c}(k+1) = \boldsymbol{c}_{aux}(2 : N+3)$;
for $i = 1 : N+1$
{ $oldvalue = c_1$;
$c_1 = \sqrt{c_1^2 + c_{i+1}^2}$;
$cos\theta'_{b_{i-1}}(k+1) = oldvalue/c_1$;
$sin\theta'_{b_{i-1}}(k+1) = -c_{i+1}/c_1$;
}
$1/\gamma^{(0)}(k+1) = 1$;
for $i = 1 : N+1$
{ $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + a_i^2(k+1)}$;
$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
$sin\theta_{i-1}(k+1) = \frac{a_i(k+1)}{1/\gamma^{(i)}(k+1)}$;
}
$\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$;
$e_{q_1}^{(0)}(k+1) = d(k+1)$;
for $i = 1 : N+1$
{ $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1)e_{q_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_i}(k)$;
$d_{q2_i}(k+1) = sin\theta_{i-1}(k+1)e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_i}(k)$;
}
$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$;
$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$;
}

3. The FQR_POS_B Algorithm:

The first version of this algorithm is based on the fact that the last element of $\boldsymbol{f}(k+1)$ (or $f_{N+1}(k+1) = \frac{x(k+1)}{\| \boldsymbol{e}_f^{(0)}(k+1) \|}$) is known in advance or prior to its calculation.

FQR_POS_B - Version 1

Initialization:
ϵ = small positive value;
$\| \boldsymbol{e}_f(k) \| = \epsilon$;
$\boldsymbol{d}_{fq2}(k) = \text{zeros}(N+1, 1)$;
$\boldsymbol{d}_{q2}(k) = \text{zeros}(N+1, 1)$;
$cos\boldsymbol{\theta}(k) = \text{ones}(N+1, 1)$;
$sin\boldsymbol{\theta}(k) = \text{zeros}(N+1, 1)$;
$\boldsymbol{f}(k) = \text{zeros}(N+1, 1)$;
for $k = 1, 2, \ldots$
{ $e_{fq_1}^{(0)}(k+1) = x(k+1)$;
for $i = 1 : N+1$
{ $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
}
$e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;
$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e_{fq_1}^2(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$;
$\| \boldsymbol{e}_f^{(N+1)}(k+1) \| = \| \boldsymbol{e}_f(k+1) \|$;
for $i = 1 : N+1$
{ $\| \boldsymbol{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \boldsymbol{e}_f^{(N+2-i)}(k+1) \|^2 + d_{fq2_i}^2(k+1)}$;
$cos\theta'_{f_{N+1-i}}(k+1) = \| \boldsymbol{e}_f^{(N+2-i)}(k+1) \| \; / \; \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
$sin\theta'_{f_{N+1-i}}(k+1) = d_{fq2_i}(k+1) / \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
}

(Continuation of the FQR_POS_B Algorithm - Version 1)

$aux_0 = x(k+1)/ \parallel \boldsymbol{e}_f^{(0)}(k+1) \parallel;$
$f_{N+1}(k+1) = aux_0;$
for $i = 1 : N$
{ $f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k+1)aux_{i-1}}{cos\theta'_{f_{i-1}}(k+1)};$
$aux_i = -sin\theta'_{f_{i-1}}(k+1)f_{N+1-i}(k) + cos\theta'_{f_{i-1}}(k+1)aux_{i-1};$
}
$\gamma^{(0)}(k+1) = 1;$
for $i = 1 : N+1$
{ $sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1)/\gamma^{(i-1)}(k+1);$
$cos\theta_{i-1}(k+1) = \sqrt{1 - sin^2\theta_{i-1}(k+1)};$
$\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1)\gamma^{(i-1)}(k+1);$
}
$\gamma(k+1) = \gamma^{(N+1)}(k+1);$
$e_{q_1}^{(0)}(k+1) = d(k+1);$
for $i = 1 : N+1$
{ $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1)e_{q_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_{N+2-i}}(k);$
$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1)e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_{N+2-i}}(k);$
}
$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1);$
$e(k+1) = e_{q_1}(k+1)\gamma(k+1);$
}

The second version of the FQR_POS_B algorithm is based on the straightforward computation of $\boldsymbol{f}(k+1)$ according to (4.84) and requires the calculation of $\frac{e_f(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$.

FQR_POS_B - Version 2

Initialization:
ϵ = small positive value;
$\| \boldsymbol{e}_f(k) \| = \epsilon$;
$\boldsymbol{d}_{fq2}(k) = \text{zeros}(N+1,1)$;
$\boldsymbol{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\boldsymbol{\theta}(k) = \text{ones}(N+1,1)$;
$sin\boldsymbol{\theta}(k) = \text{zeros}(N+1,1)$;
$\boldsymbol{f}(k) = \text{zeros}(N+1,1)$;
for $k = 1, 2, \ldots$
{ $e_{fq_1}^{(0)}(k+1) = x(k+1)$;
for $i = 1 : N+1$
{ $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
}
$e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;
$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e_{fq_1}^2(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$;
$\| \boldsymbol{e}_f^{(N+1)}(k+1) \| = \| \boldsymbol{e}_f(k+1) \|$;
for $i = 1 : N+1$
{ $\| \boldsymbol{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \boldsymbol{e}_f^{(N+2-i)}(k+1) \|^2 + d_{fq2_i}^2(k+1)}$;
$cos\theta'_{f_{N+1-i}}(k+1) = \| \boldsymbol{e}_f^{(N+2-i)}(k+1) \| \; / \; \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
$sin\theta'_{f_{N+1-i}}(k+1) = d_{fq2_i}(k+1) / \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
}

(Continuation of the FQR_POS_B Algorithm - Version 2)

$aux_0 = \frac{\gamma(k) e_{fq1}(k+1)}{\|\boldsymbol{e}_f(k+1)\|}$;
for $i = 1 : N+1$
{ $f_{i-1}(k+1) = cos\theta'_{f_{N+1-i}}(k+1) f_i(k) - sin\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;
$aux_i = sin\theta'_{f_{N+1-i}}(k+1) f_i(k) + cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;
}
$\frac{e_b(k+1)}{\|\boldsymbol{e}_b(k+1)\|} = f_0(k+1)$;
$f_{N+1}(k+1) = aux_{N+1}$;
$\gamma^{(0)}(k+1) = 1$;
for $i = 1 : N+1$
{ $sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1)/\gamma^{(i-1)}(k+1)$;
$cos\theta_{i-1}(k+1) = \sqrt{1 - sin^2\theta_{i-1}(k+1)}$;
$\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1)\gamma^{(i-1)}(k+1)$;
}
$\gamma(k+1) = \gamma^{(N+1)}(k+1)$;
$e_{q_1}^{(0)}(k+1) = d(k+1)$;
for $i = 1 : N+1$
{ $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;
$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;
}
$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$;
$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$;
}

4. The FQR_PRI_B Algorithm:

The first version of this algorithm is based on the fact that the last element of $\boldsymbol{a}(k+1)$ (or $a_{N+1}(k+1) = \frac{x(k+1)}{\sqrt{\lambda}\| \boldsymbol{e}_f^{(0)}(k)\|}$) is known in advance or prior to its calculation. This version was presented in [19].

FQR_PRI_B - Version 1

Initialization:
ϵ = small positive value;
$\| \boldsymbol{e}_f^{(0)}(k) \| = \epsilon$;
$\| \boldsymbol{e}_f(k) \| = \epsilon$;
$\boldsymbol{d}_{fq2}(k) = \text{zeros}(N+1, 1)$;
$\boldsymbol{d}_{q2}(k) = \text{zeros}(N+1, 1)$;
$cos\boldsymbol{\theta}(k) = \text{ones}(N+1, 1)$;
$cos\boldsymbol{\theta}'_f(k) = \text{ones}(N+1, 1)$;
$sin\boldsymbol{\theta}(k) = \text{zeros}(N+1, 1)$;
$sin\boldsymbol{\theta}'_f(k) = \text{zeros}(N+1, 1)$;
$\boldsymbol{a}(k) = \text{zeros}(N+1, 1)$;
for $k = 1, 2, \ldots$
{ $e_{fq_1}^{(0)}(k+1) = x(k+1)$;
  for $i = 1 : N+1$
  { $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
    $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
  }
  $e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;
  $aux_0 = \frac{x(k+1)}{\lambda^{1/2}\| \boldsymbol{e}_f^{(0)}(k)\|}$;
  $a_{N+1}(k+1) = aux_0$;
  for $i = 1 : N$
  { $a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k) aux_{i-1}}{cos\theta'_{f_{i-1}}(k)}$;
    $aux_i = -sin\theta'_{f_{i-1}}(k) a_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k) aux_{i-1}$;
  }
  $\| \boldsymbol{e}_f(k+1) \| = \sqrt{e_{fq_1}^2(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$;

(Continuation of the FQR_PRI_B Algorithm - Version 1)

$\| \boldsymbol{e}_f^{(N+1)}(k+1) \| = \| \boldsymbol{e}_f(k+1) \|$;
for $i = 1 : N+1$
{ $\| \boldsymbol{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \boldsymbol{e}_f^{(N+2-i)}(k+1) \|^2 + d_{fq2_i}^2(k+1)}$;
$cos\theta'_{f_{N+1-i}}(k+1) = \| \boldsymbol{e}_f^{(N+2-i)}(k+1) \| / \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
$sin\theta'_{f_{N+1-i}}(k+1) = d_{fq2_i}(k+1) / \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
}
$1/\gamma^{(0)}(k+1) = 1$;
for $i = 1 : N+1$
{ $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + a_{N+2-i}^2(k+1)}$;
$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
$sin\theta_{i-1}(k+1) = \frac{a_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;
}
$\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$;
$e_{q_1}^{(0)}(k+1) = d(k+1)$;
for $i = 1 : N+1$
{ $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;
$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;
}
$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$;
$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$;
}

The second version of the FQR_PRI_B algorithm is based on the straightforward computation of $\boldsymbol{a}(k+1)$ according to (4.85) and requires the calculation of $\frac{e'_f(k+1)}{\sqrt{\lambda}\|\boldsymbol{e}_f(k)\|}$. This version was presented in [18].

FQR_PRI_B - Version 2

Inicialization:
ϵ = small positive value;
$\| \boldsymbol{e}_f^{(0)}(k) \| = \epsilon$;
$\| \boldsymbol{e}_f(k) \| = \epsilon$;
$\boldsymbol{d}_{fq2}(k) = \text{zeros}(N+1,1)$;
$\boldsymbol{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\boldsymbol{\theta}(k) = \text{ones}(N+1,1)$;
$cos\boldsymbol{\theta}'_f(k) = \text{ones}(N+1,1)$;
$sin\boldsymbol{\theta}(k) = \text{zeros}(N+1,1)$;
$sin\boldsymbol{\theta}'_f(k) = \text{zeros}(N+1,1)$;
$\boldsymbol{a}(k) = \text{zeros}(N+1,1)$;
for $k = 1, 2, \ldots$
{ $e_{fq_1}^{(0)}(k+1) = x(k+1)$;
for $i = 1 : N+1$
{ $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
$d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
}
$e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;
$aux_0 = \frac{e_{fq_1}(k+1)}{\gamma(k)\lambda^{1/2}\|\boldsymbol{e}_f(k)\|}$;
for $i = 1 : N+1$
{ $a_{i-1}(k+1) = cos\theta'_{f_{N+1-i}}(k) a_i(k) - sin\theta'_{f_{N+1-i}}(k) aux_{i-1}$;
$aux_i = sin\theta'_{f_{N+1-i}}(k) a_i(k) + cos\theta'_{f_{N+1-i}}(k) aux_{i-1}$;
}
$\frac{e'_b(k+1)}{\lambda^{1/2}\|\boldsymbol{e}_b(k)\|} = a_0(k+1)$;
$a_{N+1}(k+1) = aux_{N+1}$;
$\| \boldsymbol{e}_f(k+1) \| = \sqrt{e_{fq_1}^2(k+1) + \lambda \| \boldsymbol{e}_f(k) \|^2}$;

(Continuation of the FQR_PRI_B Algorithm - Version 2)

$\| \boldsymbol{e}_f^{(N+1)}(k+1) \| = \| \boldsymbol{e}_f(k+1) \|$;
for $i = 1 : N+1$
{ $\| \boldsymbol{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \boldsymbol{e}_f^{(N+2-i)}(k+1) \|^2 + d_{fq2_i}^2(k+1)}$;
$cos\theta'_{f_{N+1-i}}(k+1) = \| \boldsymbol{e}_f^{(N+2-i)}(k+1) \| / \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
$sin\theta'_{f_{N+1-i}}(k+1) = d_{fq2_i}(k+1) / \| \boldsymbol{e}_f^{(N+1-i)}(k+1) \|$;
}
$1/\gamma^{(0)}(k+1) = 1$;
for $i = 1 : N+1$
{ $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + a_{N+2-i}^2(k+1)}$;
$cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
$sin\theta_{i-1}(k+1) = \frac{a_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;
}
$\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$;
$e_{q_1}^{(0)}(k+1) = d(k+1)$;
for $i = 1 : N+1$
{ $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) - sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;
$d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;
}
$e_{q_1}(k+1) = e_{q_1}^{(N+1)}(k+1)$;
$e(k+1) = e_{q_1}(k+1)\gamma(k+1)$;
}