José Antonio Apolinário Jr.

QRD-RLS Adaptive Filtering

Springer

QRD-RLS Adaptive Filtering

José Antonio Apolinário Jr.
Editor

QRD-RLS Adaptive Filtering

Foreword by Prof. John G. McWhirter

Springer

Editor
José Antonio Apolinário Jr.
Instituto Militar de Engenharia (IME)
Rio de Janeiro
Brazil
apolin@ime.eb.br

ISBN 978-0-387-09733-6 e-ISBN 978-0-387-09734-3
DOI 10.1007/978-0-387-09734-3

Library of Congress Control Number: 2008936641

Printed on acid-free paper

springer.com

To Ana, Isabela, and Eduardo.

Foreword

I feel very honoured to have been asked to write a brief foreword for this book on QRD-RLS Adaptive Filtering – a subject which has been close to my heart for many years. The book is well written and very timely – I look forward personally to seeing it in print. The editor is to be congratulated on assembling such a highly esteemed team of contributing authors able to span the broad range of topics and concepts which underpin this subject.

In many respects, and for reasons well expounded by the authors, the LMS algorithm has reigned supreme since its inception, as the algorithm of choice for practical applications of adaptive filtering. However, as a result of the relentless advances in electronic technology, the demand for stable and efficient RLS algorithms is growing rapidly – not just because the higher computational load is no longer such a serious barrier, but also because the technological pull has grown much stronger in the modern commercial world of 3G mobile communications, cognitive radio, high speed imagery, and so on.

This book brings together under one cover, and with common notation, the key results from many different strands of research relating to QRD-RLS adaptive filtering over recent years. It succeeds in putting this research into a clear historical perspective which highlights the underpinning theory and common motivating factors that have shaped the subject. This is achieved in the course of providing a very thorough and comprehensive account of the various key topics including numerous up-to-date algorithms in easily accessible form. As such, it should serve as a very good reference text whilst having considerable tutorial value.

Chapter one provides an excellent tutorial review of the fundamental topics in linear algebra which are essential in the context of developing and applying QRD-RLS algorithms. It starts with a very useful historical review and goes on to bring the concept of matrix triangularization and QR decomposition right up-to-date. The Gram–Schmidt orthogonalization technique is included for comparison and it was great to see a clear explanation of the difference between the Gram–Schmidt and modified Gram–Schmidt (MGS) techniques. For this chapter alone, and its extensive bibliography, the book is likely to be very high on the essential reading list for most of my post-graduate students in future. But there is much more to follow.

The second chapter provides a very good overview of adaptive filtering techniques ideal for someone fairly new to the subject. It gives a clear account of the least mean square (LMS) and normalized LMS algorithms before going on to introduce the basic recursive least-square (RLS) algorithm. On the way, it cleverly presents data-reusing versions of the LMS algorithm, typified by the affine projection method. These help to bridge the gap between the LMS and RLS algorithms and provide useful intermediate options. The LMS and data-reusing LMS algorithms are thus seen to be special, simplified cases of the RLS technique.

The QRD approach to adaptive filtering is clearly explained and presented in detail in Chapter 3 where the use of Givens rotations is assumed throughout. Unusually, and very sensibly, it also introduces the inverse QRD technique (based on Givens rotations). This is closely related to the basic QRD technique and best explained at this stage of the book since it is required in later chapters.

The core content of the book is presented in Chapters 4 and 5, which introduce and give a detailed exposition of the fast QRD-RLS algorithms and closely related QRD least squares lattice algorithms. A useful classification of the various QRD-RLS algorithms in Chapter 5 helps to unify and clarify the different variations which have emerged over the years. Similarly, explaining the key distinction between the QRD-RLS and QRD least squares lattice algorithms helps to put the latter class into context. It is worth noting that the author of Chapter 5 adopts a less conventional, but very interesting, approach to deriving QRD lattice algorithms. He does it in the more general context of linear interpolation, from which the conventional linear prediction methods may be deduced whilst other novel algorithms are also derived. A wealth of specific algorithms is presented throughout these two chapters.

Subsequent chapters of the book introduce and develop other important techniques such as multi-channel fast QRD-RLS algorithms (including the generalization to channels with different orders of prediction), QRD-RLS algorithms based on Householder transformations, linearly constrained QRD-RLS algorithms, and techniques for explicit weight extraction from fast QRD-RLS algorithms. The book also moves on to consider some vitally important practical aspects such as numerical stability (a difficult topic which is expertly presented in Chapter 8), the practical effect of finite-precision arithmetic, and the design of pipelined processing architectures to exploit the potential power of parallel computation for higher speed implementation.

In all, this is a very worthwhile text for anyone working, or planning to work, on adaptive filtering or adaptive beamforming. I have thoroughly enjoyed reading it and have no doubt that most readers will find it equally useful and enjoyable.

Wales, UK
September 2008

Prof. John G. McWhirter, FRS FREng
Distinguished Research Professor
School of Engineering
Cardiff University

Preface

The fast growth of the technological resources observed nowadays has triggered the development of new DSP techniques to cope with the requirements of modern industry. The research of efficient algorithms to be used in the ever-increasing applications of adaptive filters has therefore developed tremendously. In such a scenario, the QRD-RLS-based algorithms are a good option in applications where speed of convergence is of paramount importance and an efficient, reliable, and numerically robust adaptive filter is needed.

However, I believe that the nice features of this family of algorithms, in many occasions, are not used simply due to the fact that their matrix equations are not easy to understand. On the other hand, students, researchers, and practitioners need to be constantly up-to-date with the recent developments, not only by attending conferences and reading journal papers, but also by referring to a comprehensive compendium, where all concepts were carefully matured and are presented in such a way as to provide easy understanding. This is the main goal of this book: To provide the reader with the necessary tools to understand and implement a variety of QRD-RLS algorithms suitable to a vast number of applications.

This publication gathers some of the most recent developments as well as the basic concepts for a complete understanding of the QRD-RLS-based algorithms. Although this work does not cover all fronts of research in the field, it tries to bring together the most important topics for those who need an elegant and fast-converging adaptive filter.

QR decomposition has been a pearl in applied mathematics for many years; its use in adaptive filtering is introduced in the first chapter of this book in the form of an annotated bibliography.

The fundamental chapters materialized from lecture notes of a short course given at Helsinki University of Technology in the winter of 2004–2005, a number of conference and journal publications, and some theses I supervised. I was also lucky to receive contributions from many prominent authorities in the field.

This book consists of 12 chapters, going from fundamentals to more advanced aspects. Different algorithms are derived and presented, including basic, fast, lattice, multichannel, and constrained versions. Important issues, such as numerical stability, performance in finite-precision environments, and VLSI oriented implementations are also addressed. All algorithms are derived using Givens rotations, although one chapter deals with implementations using Householder reflections.

I hope the readers will find this book a handy guide to most aspects of theory and implementation details, quite useful in their professional practice. Upon request to the editor, a set of MATLAB®[1] codes for the main algorithms described in this book would be available.

Finally, I express my deep gratitude to all authors for their effort and competence in their timely and high quality contributions. I also thank the people from Springer, always very kind and professional. I am particularly grateful to my former DSc supervisor, Paulo S. R. Diniz, for his support and ability to motivate his pupils, and Marcello L. R. de Campos, the dear friend who, in the middle of a technical meeting on a sunny Friday, suggested this book.

Rio de Janeiro, Brazil
September 2008

José A. Apolinário Jr. D. Sc.
`apolin@ieee.org`

[1] MATLAB is a registered trademark of The MathWorks, Inc.

Contents

List of Contributors

José Antonio Apolinário Jr. (Editor)
Department of Electrical Engineering (SE/3)
Military Institute of Engineering (IME)
Praça General Tibúrcio 80, Rio de Janeiro, RJ, 22290-270 – Brazil
e-mail: `apolin@ieee.org`

Richard C. Le Borne
Department of Mathematics
Tennessee Technological University
Box 5054, Cookeville, TN 38505 – USA
e-mail: `rleborne@tntech.edu`

Marcello L. R. de Campos
Electrical Engineering Program, COPPE
Federal University of Rio de Janeiro (UFRJ)
P. O. Box 68504, Rio de Janeiro, RJ, 21941-972 – Brazil
e-mail: `campos@lps.ufrj.br`

Shiunn-Jang Chern
Department of Electrical Engineering
National Sun-Yat Sen University
70 Lienhai Road, Kaohsiung, Taiwan 80424 – R.O.C.
e-mail: `chern@mail.ee.nsysu.edu.tw`

Paulo S. R. Diniz
Electrical Engineering Program, COPPE
Federal University of Rio de Janeiro (UFRJ)
P. O. Box 68504, Rio de Janeiro, RJ, 21941-972 – Brazil
e-mail: `diniz@lps.ufrj.br`

Jun Ma
School of Microelectronics
Shanghai Jiaotong University
800 Dongchun Road, Shanghai 200240 – China
e-mail: majun@ic.sjtu.edu.cn

Maria D. Miranda
Department of Telecommunications and Control
University of São Paulo (USP)
Avenida Prof. Luciano Gualberto 158, São Paulo, SP, 05508-900 – Brazil
e-mail: maria@lcs.poli.usp.br

Mohammed Mobien
Department of Signal Processing and Acoustics, SMARAD CoE
Helsinki University of Technology
P.O. Box 3000 TKK, FIN-02015 – Finland
e-mail: mobien@signal.tkk.fi

Sergio L. Netto
Electrical Engineering Program, COPPE
Federal University of Rio de Janeiro (UFRJ)
P. O. Box 68504, Rio de Janeiro, RJ, 21941-972 – Brazil
e-mail: sergioln@lps.ufrj.br

Keshab K. Parhi
Department of Electrical and Computer Engineering
University of Minnesota
200 Union Street SE, Minneapolis, MN 55455 – USA
e-mail: parhi@umn.edu

António L. L. Ramos
Department of Technology (ATEK)
Buskerud University College (HIBU)
P. O. Box 251, 3603 Kongsberg – Norway
e-mail: antonio.ramos@hibu.no

Phillip Regalia
Department of Electrical Engineering and Computer Science
Catholic University of America
620 Michigan Avenue NE, Washington, DC 20064 – USA
e-mail: regalia@cua.edu

Athanasios A. Rontogiannis
Institute for Space Applications and Remote Sensing
National Observatory of Athens
Metaxa and Vas. Pavlou Street, Athens 15236 – Greece
e-mail: tronto@space.noa.gr

Marcio G. Siqueira
Cisco Systems
170 West Tasman Drive, San Jose, CA 95134-1706 – USA
e-mail: mgs@cisco.com

Gilbert Strang
Department of Mathematics
Massachusetts Institute of Technology (MIT)
77 Massachusetts Avenue, Cambridge, MA 02139-4307 – USA
e-mail: gs@math.mit.edu

Sergios Theodoridis
Department of Informatics and Telecommunications
University of Athens
Panepistimiopolis, Ilissia, Athens 15784 – Greece
e-mail: stheodor@di.uoa.gr

Stefan Werner
Department of Signal Processing and Acoustics, SMARAD CoE
Helsinki University of Technology
P.O. Box 3000 TKK, FIN-02015 – Finland
e-mail: stefan.werner@tkk.fi

Jenq-Tay Yuan
Department of Electrical Engineering
Fu Jen Catholic University
510 Chung Cheng Road, Hsinchuang, Taiwan 24205 – R.O.C.
e-mail: yuan@ee.fju.edu.tw

Chapter 1
QR Decomposition: An Annotated Bibliography

Marcello L. R. de Campos and Gilbert Strang

Abstract This chapter is divided into two parts. The first one goes back in time and tries to retrace the steps of great mathematicians who laid the foundations of numerical linear algebra. We describe some early methods to compute the eigenvalues and eigenvectors associated to a matrix **A**. The QR decomposition (orthogonalization as in Gram–Schmidt) is encountered in many of these methods as a fundamental tool for the factorization of **A**. The first part describes the QR algorithm, which uses the QR decomposition iteratively for solving the eigenproblem $\mathbf{Ax} = \lambda\mathbf{x}$. The second part of the chapter analyzes the application of the QR decomposition to adaptive filtering.

1.1 Preamble

To tell the story of the QR decomposition, we must go back in history, to an era when electronic calculating machines were first built and the associated algorithmic programming languages were first proposed [1, 2]. Two problems that were common to different applications and of particular importance to the newly created field of applied mathematics were: The solution of large systems of linear simultaneous equations

$$\mathbf{Ax} = \mathbf{b}, \tag{1.1}$$

Marcello L. R. de Campos
Federal University of Rio de Janeiro, Rio de Janeiro – Brazil
e-mail: campos@lps.ufrj.br

Gilbert Strang
Massachusetts Institute of Technology, Cambridge, MA – USA
e-mail: gs@math.mit.edu

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_1, © Springer Science+Business Media, LLC 2009

often arising as consequence of least-squares minimization [3], and the solution of the eigenvalue–eigenvector problem [4, 5]

$$\mathbf{Ax} = \lambda \mathbf{x}. \tag{1.2}$$

These problems may also be seen as particular cases of the problem of Fredholm [6] in the matrix format (see, e.g., [7]),

$$\mathbf{x} - \lambda \mathbf{Ax} = \mathbf{b}, \tag{1.3}$$

which simplifies to the eigenproblem for $\mathbf{b} = \mathbf{0}$ and to the solution of the linear system of equations for $\lambda \to \infty$. The methods to be outlined here are systematic procedures to solve the problem, offering a more economical and sometimes a more stable alternative to matrix inversion or to the Liouville-Neumann expansion

$$\mathbf{x} = (\mathbf{I} - \lambda \mathbf{A})^{-1}\mathbf{b} = (\mathbf{I} + \lambda \mathbf{A} + \lambda^2 \mathbf{A}^2 + \cdots)\mathbf{b}. \tag{1.4}$$

For an account of the early developments in numerical linear algebra related to the different methods for matrix inversion and eigenproblem solution see, e.g., [8–11].

In the following sections, the QR decomposition and the QR algorithm, which have become a standard subject in all linear algebra books (see, e.g., [3, 12–16]), will be placed in the context of the solutions proposed for the Equation (1.2). We will describe the solutions trying to give credit to key researchers that contributed to the development of the methods, as well as their refinement and dissemination.

1.2 Eigenvalues and Eigenvectors

The eigenproblem, as the eigenvalue–eigenvector problem is usually referred to, can be traced back more than 150 years ago to the pioneering work of Jacobi [17] (see also [18–21]). He has not described the problem in matrix notation, which was invented shortly after [22–24]. Jacobi indeed proposed an ingenious solution to the Equation (1.2) for the particular case of symmetric matrices.

A non-zero vector $\mathbf{x} \in \mathbb{C}^N$ is an eigenvector of $\mathbf{A} \in \mathbb{C}^{N\times N}$ and $\lambda \in \mathbb{C}$ is the associated eigenvalue if the Equation (1.2) is true. Since $(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = \mathbf{0}$, the eigenvalues are the roots of the characteristic polynomial, given by

$$\wp_{\mathbf{A}}(z) = \det(z\mathbf{I} - \mathbf{A}), \tag{1.5}$$

where $\mathbf{I}$ is the identity matrix [25]. A scalar λ is an eigenvalue of $\mathbf{A}$ if and only if $\wp_{\mathbf{A}}(\lambda) = 0$ [19, 26]. But the roots of an Nth-degree polynomial have no closed-form formula for $N > 4$ [27–29], and a direct approach yields an ill-conditioned problem. Therefore the solution of the eigenproblem must resort to iterative methods that *reveal* the eigenvalues of $\mathbf{A}$ as they factor or transform the matrix.

As all methods are iterative, their development and constant improvement aim for reduction of computational complexity, increased speed of convergence, and robustness against round-off errors. The QR algorithm, which is based on the QR decomposition of **A**, is still considered one of the most important methods developed so far.

1.3 Iterative Methods for the Solution of the Eigenproblem

All the methods can be broken down to two steps. The first one reduces the general matrix **A** to a more condensed form, usually Hessenberg (tridiagonal when **A** is symmetric or Hermitian). The second step solves the condensed eigenproblem iteratively.

1.3.1 The LR algorithm

Based on the reduction of the matrix to a condensed form $\mathbf{C}^{-1}\mathbf{A}\mathbf{C} = \mathbf{H}$, such as Hessenberg, several methods were proposed to find the roots of the characteristic equation. Perhaps the most significant development to follow was the proposition of the LR algorithm by Rutishauser [30]. The reduction of the matrix, in this case, yields a triangular matrix by means of non-orthogonal transformations:

$$\mathbf{A} = \mathbf{L}\mathbf{R}, \tag{1.6}$$

where **L** is lower triangular and **R** is upper triangular. The LR algorithm was a development of previous methods also proposed by Rutishauser, such as the QD algorithm (see, e.g., [31, 32]).

Based on the Equation (1.6), we may apply a similarity transformation to **A** as

$$\mathbf{A}_2 = \mathbf{L}^{-1}\mathbf{A}\mathbf{L} = \mathbf{L}^{-1}\mathbf{L}\mathbf{R}\mathbf{L} = \mathbf{R}\mathbf{L}, \tag{1.7}$$

which indicates that the lower and upper triangular factors of **A**, when multiplied in the reverse order, yield a matrix similar to **A**. In the LR algorithm, this procedure is applied iteratively and, under certain restrictions, $\mathbf{A}_k$, $k \rightarrow \infty$, converges to a triangular matrix similar to **A**. The eigenvalues of **A** are *revealed* on the diagonal of this new matrix $\mathbf{A}_k$. The LR algorithm may be stated as follows:

$$\begin{aligned}
&\mathbf{A}_1 = \mathbf{A}; \\
&\text{For } k = 1, 2, \ldots \\
&\quad \mathbf{A}_k = \mathbf{L}_k\mathbf{R}_k; \\
&\quad \mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{L}_k.
\end{aligned} \tag{1.8}$$

Although the LR algorithm is usually referenced at a later date, its fundamentals were already clear in [33], although only for the particular cases of $\mathbf{A}$ of Jacobi type or real and symmetric.

1.3.2 The QR algorithm

Proposed by J. G. F. Francis [34] as a modification of the LR algorithm (and also independently by V. N. Kublanovskaya [35]), the QR algorithm decomposes $\mathbf{A}$ into a unitary matrix $\mathbf{Q}$ and a (different!) upper triangular matrix $\mathbf{R}$, instead of the lower and upper triangular matrices from elimination. The QR algorithm uses successive unitary transformations, which render the method superior to its predecessor with respect to numerical stability and computational requirements [34, 36]. For these reasons, together with the more recent development of its parallelizable versions, the QR algorithm still is the dominant method for solving the eigenproblem [37, 38]. Standard public-domain software-library packages, such as EISPACK and LINPACK, and more recently LAPACK [39] and ScaLAPACK [40], have been using the QR algorithm in one form or other. The decomposition of $\mathbf{A}$ becomes

$$\mathbf{A} = \mathbf{QR}. \tag{1.9}$$

Similarly to the LR algorithm, a similarity transformation $\mathbf{Q}$ applied to $\mathbf{A}$ yields

$$\mathbf{A}_2 = \mathbf{Q}^{\mathrm{H}}\mathbf{A}\mathbf{Q} = \mathbf{Q}^{\mathrm{H}}\mathbf{QRQ} = \mathbf{RQ}. \tag{1.10}$$

The QR algorithm (without shift) can be described as follows [34]:

$$\begin{aligned}
&\mathbf{A}_1 = \mathbf{A};\\
&\text{For } k = 1,2,\ldots\\
&\quad \mathbf{A}_k = \mathbf{Q}_k\mathbf{R}_k;\\
&\quad \mathbf{A}_{k+1} = \mathbf{R}_k\mathbf{Q}_k.
\end{aligned} \tag{1.11}$$

For any iteration k, we have

$$\begin{aligned}
\mathbf{A}_k &= \mathbf{Q}_k^{\mathrm{H}}\mathbf{Q}_{k-1}^{\mathrm{H}}\cdots\mathbf{Q}_1^{\mathrm{H}}\mathbf{A}\mathbf{Q}_1\cdots\mathbf{Q}_{k-1}\mathbf{Q}_k\\
&= \bar{\mathbf{Q}}_k^{\mathrm{H}}\mathbf{A}\bar{\mathbf{Q}}_k.
\end{aligned} \tag{1.12}$$

With few exceptions [41], $\mathbf{A}_k$ tends to an upper triangular matrix as $k \to \infty$ [42–44].

Successive pre- and post-multiplications by unitary matrices guarantee that $\mathbf{A}_k$ and $\mathbf{A}$ are similar and therefore share the same eigenvalues, which are revealed on the diagonal of $\mathbf{A}_k$.

The pre-multiplication by the unitary matrix $\mathbf{Q}_k^{\mathrm{H}}$ can conveniently put zeros on the sub-diagonal elements of any column of $\mathbf{A}$, which might lead us to believe that triangularization is possible exactly after a finite number of operations, contradicting the fact that eigenvalue discovery should necessarily be iterative. However, it is precisely the post-multiplication by $\mathbf{Q}$ that destroys the triangular structure. Fortunately the succession of transformations converges and all eigenvalues are resolved [34].

As the iterations progress and $\mathbf{A}_k$ converges to a triangular matrix, we may notice that

$$\begin{aligned}
\mathbf{A} &= \mathbf{Q}_1\mathbf{R}_1 = \bar{\mathbf{Q}}_1\bar{\mathbf{R}}_1 \\
\mathbf{A}^2 &= \mathbf{A}\mathbf{Q}_1\mathbf{R}_1 = \mathbf{Q}_1\mathbf{R}_1\mathbf{Q}_1\mathbf{R}_1 = \mathbf{Q}_1\mathbf{A}_2\mathbf{R}_1 \\
&= \mathbf{Q}_1\mathbf{Q}_2\mathbf{R}_2\mathbf{R}_1 = \bar{\mathbf{Q}}_2\bar{\mathbf{R}}_2 \\
&\vdots \\
\mathbf{A}^k &= \mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_k\mathbf{R}_k\cdots\mathbf{R}_2\mathbf{R}_1 \\
&= \bar{\mathbf{Q}}_k\bar{\mathbf{R}}_k,
\end{aligned} \tag{1.13}$$

which shows that the matrices $\bar{\mathbf{Q}}_k$ and $\bar{\mathbf{R}}_k$ are the Q and R factors of $\mathbf{A}^k$ [44–46].

Reducing $\mathbf{A}$ to a condensed Hessenberg form, prior to the QR steps, usually speeds up the process considerably [38].

Up to this point, we have not worried about the decomposition of $\mathbf{A}$ into its factors $\mathbf{Q}$ and $\mathbf{R}$. This decomposition has been assumed possible and available. In the next section, we will describe some QR decomposition methods that can be used to factor $\mathbf{A}_k$ at the kth iteration of the QR algorithm.

1.4 QR Decomposition for Orthogonalization

The QR decomposition factors $\mathbf{A}$ into the product of a unitary matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$. Given that $\mathbf{Q}$ is unitary, $\mathbf{A}^{\mathrm{H}}\mathbf{A} = \mathbf{R}^{\mathrm{H}}\mathbf{R}$ and therefore the matrix $\mathbf{R}$ is the Cholesky factor of the matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$, but $\mathbf{R}$ is not found that way.

Let $\mathbf{A} \in \mathbb{C}^{N\times N}$ have full rank and let $\mathscr{A}_k = \{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$, $1 \le k \le N$ be the successive spaces spanned by the first k columns of $\mathbf{A}$. The QR decomposition yields a set of orthonormal vectors $\mathbf{q}_i \in \mathbb{C}^N$ that span these successive spaces,

$$\begin{aligned}
\mathbf{a}_1 &= r_{11}\mathbf{q}_1 \\
\mathbf{a}_2 &= r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 \\
&\vdots \\
\mathbf{a}_k &= r_{1k}\mathbf{q}_1 + r_{2k}\mathbf{q}_2 + \cdots + r_{kk}\mathbf{q}_k, \quad 1 \le k \le N,
\end{aligned} \tag{1.14}$$

or, in matrix form,

$$\mathbf{A} = \mathbf{Q}\mathbf{R}, \tag{1.15}$$

where

$$\mathbf{Q} = [\mathbf{q}_1 \; \mathbf{q}_2 \; \cdots \; \mathbf{q}_N] \tag{1.16}$$

and

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ 0 & r_{22} & \cdots & r_{2N} \\ \vdots & & \ddots & \vdots \\ 0 & & & r_{NN} \end{bmatrix}. \tag{1.17}$$

We will describe briefly three methods commonly used to factor the matrix **A**:

1. the classical and the modified versions of the Gram–Schmidt orthogonalization method based on projections,
2. the Householder orthogonalization method based on reflections, and
3. the Givens orthogonalization method based on rotations.

1.4.1 The classical Gram–Schmidt orthogonalization method

A simple, albeit unstable, means to obtain a succession of suitable orthonormal vectors $\mathcal{Q}_k = \{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$, $1 \leq k \leq N$, is the classical Gram–Schmidt method. We subtract from $\mathbf{a}_k$ its projection onto the space $\mathcal{Q}_{k-1}$ already constructed:

$$\mathbf{g}_k = \mathbf{a}_k - (\mathbf{q}_1^{\mathrm{H}}\mathbf{a}_k)\mathbf{q}_1 - (\mathbf{q}_2^{\mathrm{H}}\mathbf{a}_k)\mathbf{q}_2 - \cdots - (\mathbf{q}_{k-1}^{\mathrm{H}}\mathbf{a}_k)\mathbf{q}_{k-1}, \quad 2 \leq k \leq N. \tag{1.18}$$

Then normalize $\mathbf{g}_k$ to the unit vector $\mathbf{q}_k$:

$$\mathbf{q}_k = \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|_2}, \tag{1.19}$$

with $\mathbf{g}_1 = \mathbf{a}_1$. Referring to the Equation (1.14), we realize that

$$\begin{cases} r_{ij} = \mathbf{q}_i^{\mathrm{H}}\mathbf{a}_j, & i \neq j, \\ r_{jj} = \|\mathbf{a}_j - \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i\|_2. \end{cases} \tag{1.20}$$

This orthogonalization method was proposed by Erhard Schmidt in 1907 [47], although he acknowledged that the method is equivalent to the one proposed by J. P. Gram in 1883 [48]. The method was probably first called the Gram–Schmidt orthogonalization method by Y. K. Wong in 1935 [49], where its relation to Gauss's method of substitution was first pointed out. Curiously, the method proposed by Schmidt became known as the Classical Gram–Schmidt orthogonalization method, whereas Gram's earlier version became known as the Modified Gram–Schmidt orthogonalization method. For a thorough account of Gram–Schmidt orthogonalization, including historical aspects and relationship with the QR factorization, the reader is referred to [50] and the references therein.

One may readily see that $\mathbf{q}_k$ comes from the projection of $\mathbf{a}_k$ onto the subspace orthogonal to $\mathscr{Q}_{k-1}$, i.e.,

$$\begin{aligned} \mathbf{q}_1 &= \frac{\mathbf{P}_1\mathbf{a}_1}{\|\mathbf{P}_1\mathbf{a}_1\|_2} \\ &\vdots \\ \mathbf{q}_k &= \frac{\mathbf{P}_k\mathbf{a}_k}{\|\mathbf{P}_k\mathbf{a}_k\|_2}, \end{aligned} \tag{1.21}$$

where

$$\mathbf{P}_i = \mathbf{I} - \mathbf{Q}_{i-1}\mathbf{Q}_{i-1}^{\mathrm{H}} \ \in \mathbb{C}^{N\times N} \tag{1.22}$$

and

$$\mathbf{Q}_{i-1} = [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_{i-1}] \ \in \mathbb{C}^{N\times i-1}. \tag{1.23}$$

Equivalently, the algorithm may be described as post-multiplication by upper triangular matrices $\mathbf{R}_1, \ \mathbf{R}_2, \ \ldots, \ \mathbf{R}_N$. The first vector, $\mathbf{q}_1$, is obtained by post-multiplying $\mathbf{A}$ by $\mathbf{R}_1$:

$$[\mathbf{q}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_N] = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_N] \begin{bmatrix} 1/r_{11} & 0 \cdots 0 \\ 0 & 1 \cdots 0 \\ \vdots & \ddots \vdots \\ 0 & \cdots 1 \end{bmatrix} = \mathbf{A}\mathbf{R}_1. \tag{1.24}$$

The first two vectors, $\mathbf{q}_1$ and $\mathbf{q}_2$, $\mathbf{q}_2 \perp \mathbf{q}_1$, are obtained by post-multiplying $\mathbf{A}$ by the upper-triangular matrices $\mathbf{R}_1$ and $\mathbf{R}_2$:

$$[\mathbf{q}_1 \ \mathbf{q}_2 \ \mathbf{a}_3 \cdots \ \mathbf{a}_N] = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_N] \begin{bmatrix} 1/r_{11} & 0 \cdots 0 \\ 0 & 1 \cdots 0 \\ \vdots & \ddots \vdots \\ 0 & \cdots 1 \end{bmatrix} \begin{bmatrix} 1 & -r_{12}/r_{22} & 0 \cdots 0 \\ 0 & 1/r_{22} & 0 \cdots 0 \\ 0 & 0 & 1 \cdots 0 \\ \vdots & & \ddots \vdots \\ 0 & & \cdots 1 \end{bmatrix} = \mathbf{A}\mathbf{R}_1\mathbf{R}_2, \tag{1.25}$$

where r_{ij} is given in the Equation (1.20). In order to obtain k orthonormal vectors, we must post-multiply the matrix $\mathbf{A}$ by k upper-triangular matrices $\mathbf{R}_i$:

$$\begin{aligned} [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_k \ \mathbf{a}_{k+1} \ \cdots \ \mathbf{a}_N] &= \mathbf{A}\mathbf{R}_1 \cdots \mathbf{R}_{k-1} \begin{bmatrix} 1 \ 0 \ 0 \cdots & -r_{1k}/r_{kk} & 0 \cdots 0 \\ 0 \ 1 \ 0 \cdots & -r_{2k}/r_{kk} & 0 \cdots 0 \\ \vdots \quad \ddots & \vdots & \vdots \\ 0 \quad \cdots & 1/r_{kk} & \\ 0 & & 1 \\ & & \\ 0 & & \ddots \\ 0 & & \quad 1 \end{bmatrix} \\ &= \mathbf{A}\mathbf{R}_1\mathbf{R}_2 \cdots \mathbf{R}_k. \end{aligned} \tag{1.26}$$

The existence of the QR decomposition of any matrix $\mathbf{A} \in \mathbb{C}^{M\times N}$, $M \geq N$, and the uniqueness of this decomposition when $\mathbf{A}$ is of full rank can be readily proved with the aid of the Gram–Schmidt triangular orthogonalization. As a consequence, we may expect that all methods yield exactly the same factors $\mathbf{Q}$ and $\mathbf{R}$, at least in theory, when $\mathbf{A}$ is of full rank.

In practice, performance in terms of robustness and computational complexity differs significantly. The methods to be presented next are superior to the classical Gram–Schmidt method in terms of sensitivity to round-off errors and are, usually, to be preferred in many practical applications.

1.4.2 The modified Gram–Schmidt orthogonalization method

For each column k, $1 \leq k \leq N$, the classical Gram–Schmidt method applies to $\mathbf{a}_k$ a single orthogonal projection of rank $N-k+1$, as seen in the Equation (1.22). The modified Gram–Schmidt method applies to $\mathbf{a}_k$ a succession of $k-1$ projections of rank $N-1$. It subtracts $k-1$ one-dimensional projections onto the known $\mathbf{q}_1, \ldots, \mathbf{q}_{k-1}$:

$$\mathbf{g}_k = \mathbf{P}_{k-1} \cdots \mathbf{P}_1 \mathbf{a}_k \tag{1.27}$$

and

$$\mathbf{q}_k = \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|_2}, \tag{1.28}$$

where

$$\mathbf{P}_{k-1} = \mathbf{I} - \mathbf{q}_{k-1}\mathbf{q}_{k-1}^{\mathrm{H}}. \tag{1.29}$$

The difference between the two methods is in the computation of r_{ij} such that the Equation (1.20) now becomes

$$\begin{cases} r_{ij} = \mathbf{q}_i^{\mathrm{H}} \left(\mathbf{a}_j - \sum_{k=1}^{j-1} r_{kj}\mathbf{q}_k \right), & i \neq j, \\ r_{jj} = \|\mathbf{a}_j - \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i\|_2. \end{cases} \tag{1.30}$$

Both the classical and the modified versions of the Gram–Schmidt method compute the first two columns of $\mathbf{Q}$ and the first row of $\mathbf{R}$ with the exact same arithmetic operations. However, for $j > 2$, the modified Gram–Schmidt method also takes into account cross-products $\mathbf{q}_i^{\mathrm{H}}\mathbf{q}_j$, $i \neq j$, that should ideally be zero. If computations could be performed with infinite precision, both methods would yield the same $\mathbf{Q}$. However, the impact of finite precision arithmetic onto the classical version is more pronounced, mostly in terms of stability and loss of orthogonality [50] (see also [51–53]).

1.4.3 Triangularization via Householder reflections

The Gram–Schmidt method calculates a series of projections in order to reach $\mathbf{A} = \mathbf{QR}$. The operation consists of a succession of multiplications of upper-triangular non-unitary matrices $\mathbf{R}_i$ to the right of $\mathbf{A}$ in order to obtain a unitary matrix $\mathbf{Q}$, such that $\mathbf{AR}_1 \cdots \mathbf{R}_N = \mathbf{AR}^{-1} = \mathbf{Q}$.

The Householder method [54], on the other hand, applies a succession of unitary matrices $\mathbf{Q}_i$ to the left of $\mathbf{A}$ in order to obtain an upper-triangular matrix $\mathbf{R}$, such that $\mathbf{Q}^{\mathrm{H}}\mathbf{A} = \mathbf{R}$ (Figure 1.1).

The matrix $\mathbf{Q}_i$, $i = 1,\ 2,\ \ldots,\ N-1$, is chosen unitary such that

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{I}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_i \end{bmatrix}, \tag{1.31}$$

where the blocks $\mathbf{I}$ and $\mathbf{H}$ have size $i-1$ and $N-i+1$.

The first reflection $\mathbf{H}_1$ reflects the original vector $\mathbf{a}_1$ onto $\mathbf{e}_1^N = [1\ 0\ \cdots\ 0] \in \mathbb{C}^N$ with respect to some hyperplane $\mathscr{H}_1 \in \mathbb{C}^N$ (see Figure 1.2):

$$\mathbf{H}_1^{\mathrm{H}}\mathbf{a}_1 = \|\mathbf{a}_1\|_2\mathbf{e}_1^N, \quad \mathbf{H}_i^{\mathrm{H}} = \mathbf{H}_i, \quad \mathbf{H}_i^2 = \mathbf{I}. \tag{1.32}$$

The second reflection operates on the lower part of the second column of $\mathbf{A}$, still denoted here $\mathbf{a}_2$ although it has been modified by $\mathbf{Q}_1$. This part is reflected onto $\mathbf{e}_1^{N-1} = [1\ 0\ \cdots 0] \in \mathbb{C}^{N-1}$ with respect to some hyperplane $\mathscr{H}_2 \in \mathbb{C}^{N-1}$:

$$\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2^{\mathrm{H}} \end{bmatrix} \mathbf{a}_2 = \begin{bmatrix} \star \\ \|\mathbf{a}_2\|_2\mathbf{e}_1^{N-1} \end{bmatrix}. \tag{1.33}$$

The procedure is applied $N-1$ times, ending with a multiple of $\mathbf{e}_N$ in the last column.

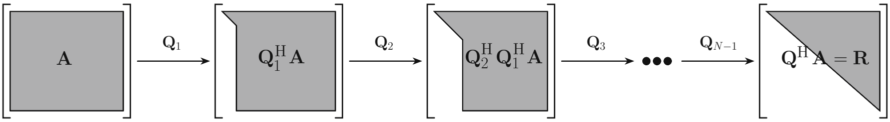


Fig. 1.1 Unitary triangularization of the matrix $\mathbf{A}$: Householder reflections.

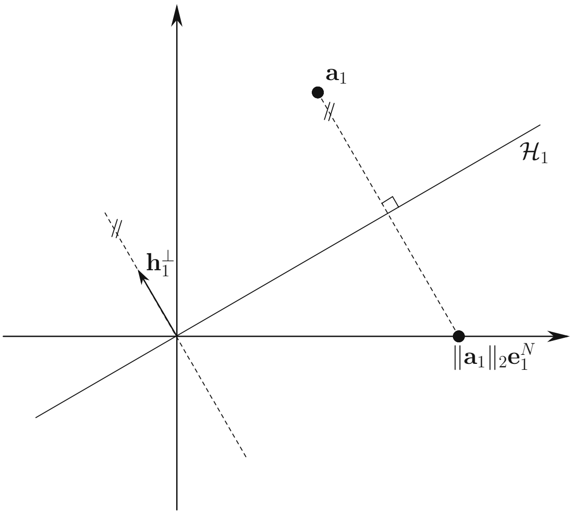


Fig. 1.2 A Householder reflection.

Given a hyperplane $\mathscr{H}_1$ and a unit vector $\mathbf{h}_1^{\perp}$ orthogonal to $\mathscr{H}_1$, the projection of $\mathbf{a}_1$ onto $\mathscr{H}_1$ is given by

$$\mathbf{P}_1 = \mathbf{I} - \mathbf{h}_1^{\perp}\mathbf{h}_1^{\perp\mathrm{H}}. \tag{1.34}$$

In order to project $\mathbf{a}_1$ *across* $\mathscr{H}_1$ and onto $\mathbf{e}_1^N$, we must apply the Householder reflection

$$\mathbf{H}_1^{\mathrm{H}}\mathbf{a}_1 = \left(\mathbf{I} - 2\,\mathbf{h}_1^{\perp}\mathbf{h}_1^{\perp\mathrm{H}}\right)\mathbf{a}_1. \tag{1.35}$$

Householder reflections have become the standard method for orthogonalization, chosen by LAPACK [39] (and by MATLAB® and other popular software systems). One of its advantages is that orthogonalization of **Q** is guaranteed to the working precision [50]. Although originally presented as a triangularization method with immediate application to matrix inversion, soon it was recognized as a fast and accurate method for solving the eigenproblem [55].

1.4.4 Triangularization via Givens plane rotations

Another interesting method to decompose **A** is based on plane rotations, proposed by Givens [56]. The method can be illustrated as follows.

Let $\mathbf{a}_j$ be any column of **A** and a_{jj} be the component on the diagonal of **A**. We wish to place zeros in the positions of a_{ij}, $i = j+1, \ldots, N$, and Givens's proposition is to treat those entries one at a time. To each dyad defined as $\mathbf{x}_m = [a_{jj}\ a_{ij}]$, where

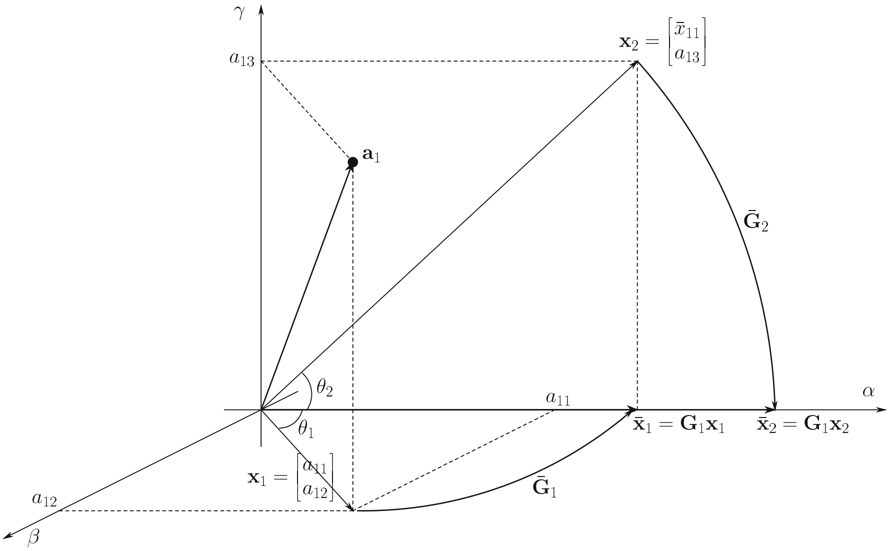

γ
a_{13}
$\mathbf{x}_2 = \begin{bmatrix}\bar{x}_{11}\\ a_{13}\end{bmatrix}$
$\mathbf{a}_1$
$\bar{\mathbf{G}}_2$
θ_2
a_{11}
α
θ_1
$\bar{\mathbf{x}}_1 = \mathbf{G}_1\mathbf{x}_1$
$\bar{\mathbf{x}}_2 = \mathbf{G}_1\mathbf{x}_2$
$\mathbf{x}_1 = \begin{bmatrix}a_{11}\\ a_{12}\end{bmatrix}$
$\bar{\mathbf{G}}_1$
a_{12}
β

Fig. 1.3 Givens rotations. The dyad $\mathbf{x}_1$ lies on the α–β plane and is constructed with the first two elements of $\mathbf{a}_1$. It is rotated by $\bar{\mathbf{G}}_1$ to be aligned with the axis α with length $\bar{x}_{11}$. The second dyad $\mathbf{x}_2$ lies on the α–γ plane and is constructed with $\bar{x}_{11}$ and the third element of $\mathbf{a}_1$. It is rotated by $\bar{\mathbf{G}}_2$ to be also aligned with the axis α.

$m = 1, \ldots, N-j$ is the index of rotations performed and $i = j+m$, we apply a rotation $\bar{\mathbf{G}}_m$ in order to align the dyad with the vector $[1 \;\; 0]$ (see Figure 1.3 for an example with $\mathbf{a}_1 \in \mathbb{C}^3$). Notice that the operation is a rotation, and not a projection; therefore the entry in the position a_{jj} is updated after each rotation, and is denoted as $\bar{x}_{jj}$ in Figure 1.3. The rotations $\bar{\mathbf{G}}_m$ are simply defined as:

$$\bar{\mathbf{G}}_m = \begin{bmatrix} \cos(\theta_m) & \sin^*(\theta_m) \\ -\sin(\theta_m) & \cos(\theta_m) \end{bmatrix}, \tag{1.36}$$

where

$$\theta_m = \tan^{-1} \frac{a_{ij}}{a_{jj}}. \tag{1.37}$$

We may cascade the succession of rotations applied to $\mathbf{a}_j$, not to the dyad, as

$$\mathbf{G}_{j,N-j} \cdots \mathbf{G}_{j,1}\, \mathbf{a}_j = \begin{bmatrix} \star \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{1.38}$$

where the rotations $\mathbf{G}_{j,m}$, $m = 1, \ldots, N-j$, place zeros along the jth-column in (i, j) positions, with $i = j+m$. The rotations $\mathbf{G}_{j,m} \in \mathbb{C}^{N \times N}$ are given by

$$\mathbf{G}_{j,m} = \begin{bmatrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & \cos(\theta_m) & & & & \sin^*(\theta_m) & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & -\sin(\theta_m) & & & & \cos(\theta_m) & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{bmatrix} \qquad \begin{array}{l} j = 1, \ldots, N-1; \\ m = 1, \ldots, N-j. \end{array} \tag{1.39}$$

1.5 QR Decomposition for Linear Least Squares Problems

Linear simultaneous equations will arise when we wish to solve systems of simultaneous equations in the form of the Equation (1.1), arising, for example, when we wish to solve the least squares problem

$$\min_{\mathbf{x}} ||\mathbf{Ax} - \mathbf{b}||_2^2, \tag{1.40}$$

where $\mathbf{A} \in \mathbb{C}^{M \times N}$, $M \geq N$, is of rank N. The solution of the Equation (1.40) above is the vector $\mathbf{x}$ satisfying the normal equations [57]:

$$\mathbf{A}^{\mathrm{H}}\mathbf{Ax} = \mathbf{A}^{\mathrm{H}}\mathbf{b}. \tag{1.41}$$

The conditioning of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ is worse than that of $\mathbf{A}$ [57], and the study of accurate methods to solve the Equation (1.41) has deserved much attention, both due to its challenging aspects and to its importance in many areas (see, e.g., [51, 57–64]. The pioneering works of Gauss [65] and Legendre [66] independently formulated the least squares problem and developed solutions which have withstood the test of time).

As the Euclidean norm is invariant under unitary transformations, the problem stated in the Equation (1.40) above can be rewritten as

$$\min_{\mathbf{x}} ||\mathbf{Q}^{\mathrm{H}}(\mathbf{Ax} - \mathbf{b})||_2^2, \tag{1.42}$$

where $\mathbf{Q} \in \mathbb{C}^{M \times M}$ is unitary and chosen such that $\mathbf{Q}^{\mathrm{H}}\mathbf{A} = \mathbf{R}$, $\mathbf{R} \in \mathbb{C}^{M \times N}$ is upper triangular, forming the $\mathbf{Q}$ and $\mathbf{R}$ factors of the QR decomposition of $\mathbf{A}$. The matrix $\mathbf{Q}$ can be obtained by any of the methods described in the previous section. The matrix $\mathbf{R}$, being upper triangular, may be partitioned as

$$\mathbf{Q}^{\mathrm{H}}\mathbf{A} = \mathbf{R} = \begin{bmatrix} \mathbf{U} \\ \mathbf{0} \end{bmatrix}, \tag{1.43}$$

where $\mathbf{U} \in \mathbb{C}^{N \times N}$ is of full rank provided that $\mathbf{A}$ is of full rank. The vector $\mathbf{Q}^{\mathrm{H}}\mathbf{b}$ may be also partitioned accordingly,

$$\mathbf{Q}^{\mathrm{H}}\mathbf{b} = \begin{bmatrix} \mathbf{b}_u \\ \mathbf{b}_l \end{bmatrix}, \tag{1.44}$$

where $\mathbf{b}_u \in \mathbb{C}^{N}$ and $\mathbf{b}_l \in \mathbb{C}^{M-N}$.

The solution of the Equation (1.42) may be obtained as the solution of

$$\mathbf{U}^{\mathrm{H}}\mathbf{U}\mathbf{x} = \mathbf{U}^{\mathrm{H}}\mathbf{b}_u, \tag{1.45}$$

or, equivalently, that of

$$\mathbf{U}\mathbf{x} = \mathbf{b}_u. \tag{1.46}$$

Notice that the Equation (1.46) describes a triangular system of equations, which can be solved via back-substitution.

Golub [67] has suggested this approach for solving least squares problems employing the Householder transformation for the QR decomposition of $\mathbf{A}$ (see also [68]). Although Golub [67] briefly mentions that many methods do exist to achieve the QR decomposition, for example a series of plane rotations, the use of Givens plane rotations together with the QR decomposition to solve linear least squares problems was fully addressed only later by Gentleman [69] (see also [70–72]).

It is usual, in least squares problems, that the matrix $\mathbf{A}$ is constructed one row at a time, as data from observations are gathered. Furthermore, it is not uncommon that the amount of data gathered becomes prohibitively large, and one should benefit from the zeros already placed in previously transformed versions of $\mathbf{A}$. A method that processes $\mathbf{A}$ one row at a time may offer clear advantages in terms of computational complexity, and the Givens plane rotations seem to be perfect for the job.

Let $\mathbf{A}_k \in \mathbb{C}^{k \times N}$, $k \geq N$, be a matrix containing k observations of some sort organized in the k rows of $\mathbf{A}_k$, and $\mathbf{x}_k$ be the least squares solution that minimizes the Euclidean norm of the residuals,

$$\min_{\mathbf{x}_k} ||\mathbf{A}_k\mathbf{x}_k - \mathbf{b}_k||_2^2. \tag{1.47}$$

Let also $\mathbf{Q}_k$ and $\mathbf{R}_k$ be the QR factors of $\mathbf{A}_k$, such that

$$\mathbf{Q}_k^{\mathrm{H}}\mathbf{A}_k = \mathbf{R}_k = \begin{bmatrix} \mathbf{U}_k \\ \mathbf{0} \end{bmatrix}. \tag{1.48}$$

When new data is incorporated into the system, the matrix $\mathbf{A}_{k+1} \in \mathbb{C}^{k+1 \times N}$ is formed with an additional row, $\mathbf{a}_{k+1}^{\mathrm{H}}$, as

$$\mathbf{A}_{k+1} = \begin{bmatrix} \mathbf{A}_k \\ \mathbf{a}_{k+1}^{\mathrm{H}} \end{bmatrix}. \tag{1.49}$$

The new matrix $\mathbf{A}_{k+1}$ and the QR factors of $\mathbf{A}_k$ are related as

$$\begin{bmatrix} \mathbf{Q}_k^{\mathrm{H}} & \mathbf{0} \\ \mathbf{0}^{\mathrm{H}} & 1 \end{bmatrix} \mathbf{A}_{k+1} = \begin{bmatrix} \mathbf{R}_k \\ \mathbf{a}_{k+1}^{\mathrm{H}} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_k \\ \mathbf{0} \\ \mathbf{a}_{k+1}^{\mathrm{H}} \end{bmatrix}. \tag{1.50}$$

We may now apply the Givens plane rotations to place zeros only on the last row of this matrix, in order to obtain $\mathbf{R}_{k+1}$ upper triangular:

$$\mathbf{Q}_{k+1}^{\mathrm{H}} \begin{bmatrix} \mathbf{U}_k \\ \mathbf{0} \\ \mathbf{a}_{k+1}^{\mathrm{H}} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{k+1} \\ \mathbf{0} \end{bmatrix}. \tag{1.51}$$

Only the upper part of the transformed vector $\mathbf{Q}_k^{\mathrm{H}}\mathbf{b}_k$ needs to be stored, and only the appropriate cosines and sines need to be calculated in order to perform the transformations. Gentleman also suggests in [69] a means to avoid the computation of square roots, supposedly needed for the Givens plane rotations.

1.5.1 QR Decomposition by systolic arrays

Matrix triangularization is certainly a major step in many methods for solving least squares problems. When the solution is required to be obtained online and in real-time, the computational complexity associated with the triangularization may be the bottleneck. The use of systolic arrays [73] for solving least squares problems with QR decomposition was first proposed by Gentleman and Kung [74] (see also [75]). Their approach takes advantage of the inherent parallelism of the Givens plane rotations for matrix triangularization, a fact already mentioned by Gentleman in [72] for the error analysis of the QR decomposition by Givens rotations.

1.6 QR Decomposition for Recursive Least Squares Adaptive Filters

Previous sections have dealt with the QR decomposition and the QR algorithm applied to solving the eigenproblem and the least squares problem. Recursive least squares (RLS) adaptive filters attempt to adjust the system parameters (filter coefficients) as new data are gathered in order to minimize the weighted sum of squares of the residuals (output errors). The QR decomposition is definitely a superb tool for the job, offering increased robustness, reduced complexity, and a possible VLSI implementation in a modular (systolic array) structure.

This section presents some of the early contributions that merged QR decomposition and RLS adaptive filtering. More recent contributions that form the state-of-the-art in the field are the subject of the remaining chapters of the book. We chose to keep our notation consistent with that of the previous sections, and also of many linear algebra texts. However, it is usual in adaptive filtering texts that the observation matrix is denoted by $\mathbf{X}(k)$, the parameter (coefficient) vector is denoted by $\mathbf{w}$, and the constant term (reference) vector is denoted by $\mathbf{d}(k)$. The index, often displayed within parenthesis, is associated with iterations or sampled time instants $t = kT$, where T is the sampling period, usually dropped from the notation for simplicity.

At every iteration or time instant $t = kT$, the adaptive filter is presented with a new vector of observations and a reference signal, which will be denoted here as $\mathbf{a}_k$ and b_k, respectively. The filter coefficients, $\mathbf{x}_k$, must be estimated to produce an output that approximates the reference signal in the sense that the sum of all the previous errors (or residuals) squared, $|e_i|^2$, $1 \leq i \leq k$, is minimized:

$$\min_{\mathbf{x}_k} \left(\xi_k = \sum_{i=1}^{k} \lambda^{k-i} |e_i|^2 \right), \tag{1.52}$$

where $0 < \lambda \leq 1$ and

$$e_i = \mathbf{a}_i^{\mathrm{H}} \mathbf{x}_k - \mathbf{b}_i. \tag{1.53}$$

In matrix notation, the Equation (1.52) can be put in the same form as the Equation (1.47) if we construct the matrix $\mathbf{A}_k$ and the vector $\mathbf{b}_k$ as

$$\mathbf{A}_k = \begin{bmatrix} \lambda^{(k-1)/2} \mathbf{a}_1^{\mathrm{H}} \\ \lambda^{(k-2)/2} \mathbf{a}_2^{\mathrm{H}} \\ \vdots \\ \lambda^{1/2} \mathbf{a}_{k-1}^{\mathrm{H}} \\ \mathbf{a}_k \end{bmatrix} \tag{1.54}$$

and

$$\mathbf{b}_k = \begin{bmatrix} \lambda^{(k-1)/2} b_1 \\ \lambda^{(k-2)/2} b_2 \\ \vdots \\ \lambda^{1/2} b_{k-1} \\ b_k \end{bmatrix}. \tag{1.55}$$

A vector of residuals can be constructed as

$$\begin{aligned} \mathbf{e}_k &= \mathbf{A}_k \mathbf{x}_k - \mathbf{b}_k \\ &= [e_1 \; e_2 \; \cdots \; e_k]^{\mathrm{T}}, \end{aligned} \tag{1.56}$$

so that the Equation (1.47) can also be rewritten as

$$\min_{\mathbf{x}_k} \|\mathbf{e}_k\|_2^2. \tag{1.57}$$

McWhirter [76, 77] and later Ward et al. [78] have proposed adaptive filtering implementations of the RLS algorithm solving the Equation (1.57). They took advantage of the Givens rotations for the matrix triangularization and suggested a systolic array implementation as in [74]. The implementation of the systolic array in [76] was particularly interesting, for the method does not need to solve the triangular linear system in order to extract the coefficients, as is the case in [74]. In [78], the application of interest was adaptive beamforming, for which coefficient extraction may not be needed. If, however, obtaining the coefficients is necessary, the authors propose a "weight flushing" method, simpler than back-substitution, and also applicable when the square-root free Givens rotation method [69] is used. In [79], Varvitsiotis et al. propose a structure that employs the modified Fadeeva's algorithm in order to avoid coefficient extraction via back-substitution. Instead, the unknown coefficients are obtained via Gaussian elimination.

1.6.1 Fast QR decomposition RLS adaptation algorithms

In [80], Cioffi proposed an early version of the first fast implementation (with computational complexity proportional to N) of the RLS algorithm using QR decomposition. A fast pipelined version was presented shortly after in [81]. The method was further developed and improved so that the implementation was based on a pipelined array formed by two types of processing elements: rotational processors (ROTORs) and cosine–sine processors (CISORs) [82]. (For an early account of fast QR decomposition RLS adaptive filters, see [83].)

The search for implementation options that offered robustness and computational complexity that increased only linearly with N has concentrated on methods that avoid the direct back-substitution for calculating the coefficients. One of the options for coefficient extraction without solving the triangular linear system via back-substitution is based on the inverse QR decomposition. In this case, instead of updating the Cholesky factor of the normal matrix $\mathbf{A}^{\mathrm{H}}\mathbf{A}$, the method updates its inverse [84–87].

As a consequence of the derivation of fast QR decomposition methods, a relationship between QR decomposition methods for adaptive filtering and linear prediction was soon established [88, 89]. Furthermore, Bellanger [89] showed that the reflection coefficients of the lattice filter may be obtained directly from the internal variables of the fast QR decomposition implementations. This relationship is further explored in [90], where the authors propose a family of QR decomposition RLS algorithms that reveal the reflection coefficients of an equivalent normalized lattice predictor. The relationship between lattice predictors and QR decomposition least squares methods received much attention not only in the context of adaptive filtering (see, e.g., [91, 92]), but also in the context of statistical signal processing in general (see, e.g., [93–95]).

The use of a modular and systematic procedure to design systolic array implementations of QR decomposition RLS adaptive filters for different applications was also an important research topic. Algorithmic engineering, introduced by McWhirter [96, 97], provided the tools for efficiently prototyping parallel algorithms and architectures for different applications [98–102].

1.7 Conclusion

In this introductory chapter, we attempted to provide an annotated bibliography that will hopefully serve the readers interested in the historical aspects related to the QR decomposition and the QR algorithm, as well as introduce them to the algebraic tools commonly encountered in the analysis and design of QR decomposition RLS adaptive filters.

QRD-RLS adaptive filters do not rely on the QR algorithm, but on the unitary triangularization of the observation matrix via Givens rotations. We presented a brief introductory discussion about the QR decomposition, its application to the solution of the eigenproblem and of the least squares problem, and its implementation options via Gram–Schmidt projections, Householder reflections, and Givens plane rotations. Although Givens rotations preceded Householder reflections, we chose to present the two methods out of their chronological order. Householder reflections have become a standard in software packages, but Givens rotations have proved more suitable and economical when new data is added to the least squares problem and we wish to take advantage of the previously transformed data. We also mentioned the possibility of a VLSI implementation via systolic arrays of the matrix triangularization, with a possible extension to the linear system solution. Here, again, the Givens rotations offer a parallelism which can be efficiently explored by the systolic array architecture.

The application of QR decomposition to RLS adaptive filtering may be regarded as a natural consequence for many constraints, and objectives here are similar to those of applied mathematics (e.g., computational complexity, robustness, and accuracy). However, the investigation of new forms of implementation that provide unforeseen advantages in different modern applications, such as multichannel signal processing and multiple-input multiple-output systems, provide renewed challenges to researchers in the field of signal processing.

We have made an effort to provide key references to give the reader information of when each method or algorithm was first published, as well as books and tutorials that present them with a more modern notation and an alternative form of presentation that may be easier to understand and implement. As all such lists are unavoidably incomplete, we sincerely apologize for any unfairness, which was indeed unintentional.

Acknowledgements

The authors are grateful to Mr. Leonardo Baltar for tirelessly helping us to obtain key references, and to Prof. Gander for kindly providing us a copy of his manuscript to appear on the history of the Gram–Schmidt orthogonalization.

References

1. M. Gutknecht, The pioneer days of scientific computing in Switzerland. Proceedings of the ACM Conference on History of Scientific and Numeric Computation. Princeton, NJ, USA, pp. 63–69 (1987)
2. A. S. Householder, Bibliography on numerical analysis. Journal of the Association for Computing Machinery (JACM), vol. 3, no. 2, pp. 85–100 (April 1956)
3. G. Strang, Computational Science and Engineering. Wellesley-Cambridge Press, Wellesley, MA, USA (2007)
4. H. A. Van der Vorst and G. H. Golub, 150 years old and still alive: Eigenproblems. The State of the Art in Numerical Analysis. I. S. Duff and G. A. Watson, Eds., vol. 63 of Institute of Mathematics and Its Applications Conference Series New Series, pp. 93–120, Oxford University Press (1997)
5. G. H. Golub and H. A. van der Vorst, Eigenvalue computation in the 20th century. Journal of Computational and Applied Mathematics, vol. 123, no. 1–2, pp. 35–65 (November 2000)
6. I. Fredholm, Sur une classe d'équations fonctionnelles. Acta Mathematica, vol. 27, no. 1, pp. 365–390 (December 1903)
7. C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Journal of Research of the National Bureau of Standards, vol. 45, no. 4, pp. 255–282 (October 1950)
8. J. Von Neumann and H. H. Goldstine, Numerical inverting of matrices of high order. Bulletin of the American Mathematical Society, vol. 53, no. 11, pp. 1021–1099 (1947)
9. A. S. Householder, A survey of some closed methods for inverting matrices. Journal of the Society for Industrial and Applied Mathematics, vol. 5, no. 3, pp. 155–169 (September 1957)
10. A. S. Householder, A class of methods for inverting matrices. Journal of the Society for Industrial and Applied Mathematics, vol. 6, no. 2, pp. 189–195 (June 1958)
11. M. R. Hestenes, Inversion of matrices by biorthogonalization and related results. Journal of the Society for Industrial and Applied Mathematics, vol. 6, no. 1, pp. 51–90 (March 1958)
12. J. H. Wilkinson, The Algebraic Eigenvalue Problem. Oxford University Press, New York, NY, USA (1965)
13. R. A. Horn and C. R. Johnson, Matrix Analysis. Cambridge University Press, Cambridge, UK (1985)
14. G. H. Golub and C. F. Van Loan, Matrix Computations. 3rd edition The Johns Hopkins University Press, Baltimore, MD, USA (1996)
15. L. N. Trefethen and D. Bau III, Numerical Linear Algebra. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1997)
16. G. Strang, Linear Algebra and Its Applications. 4th edition Thomson–Brooks/Cole, Belmont, CA, USA (2005)
17. C. G. J. Jacobi, Über ein leichtes verfahren, die in der theorie der säcularstörungen vorkommenden gleichungen numerisch aufzulösen. Journal für die Reine und Angewandte Mathematik, vol. 30, pp. 51–94 (1846)
18. J. Greenstadt, A method for finding roots of arbitrary matrices. Mathematical Tables and Other Aids to Computation, vol. 9, no. 50, pp. 47–52 (April 1955)

19. H. H. Goldstine, F. J. Murray, and J. von Neumann, The Jacobi method for real symmetric matrices. Journal of the ACM, vol. 6, no. 1, pp. 59–96 (January 1959)
20. H. H. Goldstine, A History of Numerical Analysis from the 16th through the 19th Century. Studies in the History of Mathematics and Physical Sciences. Springer-Verlag, New York, NY, USA (1977)
21. G. E. Forsythe and P. Henrici, The cyclic Jacobi method for computing the principal values of a complex matrix. Transactions of the American Mathematical Society, vol. 94, no. 1, pp. 1–23 (January 1960)
22. A. Cayley, A memoir on the theory of matrices. Philosophical Transactions of the Royal Society of London, vol. CXLVIII, pp. 17–37 (1858)
23. I. G. Bashmakova and A. N. Rudakov, The evolution of algebra 1800–1870. The American Mathematical Monthly, vol. 102, no. 3, pp. 266–270 (March 1995)
24. N. J. Higham, Cayley, Sylvester, and early matrix theory. Linear Algebra and Its Applications, vol. 428, no. 1, pp. 39–43 (January 2008)
25. P. Horst, A method for determining the coefficients of a characteristic equation. The Annals of Mathematical Statistics, vol. 6, no. 2, pp. 83–84 (June 1935)
26. W. Givens, The characteristic value-vector problem. Journal of the ACM, vol. 4, no. 3, pp. 298–307 (July 1957)
27. N. H. Abel. Démonstration de l'impossibilité de la résolution algébrique des équations générales qui passent le quatrième degré. Oeuvres Complètes. Suivi de Niels Henrik Abel, Sa Vie et Son Action Scientifique, deuxiéme edition Jacques Gabay, vol. I, pp. 66–87 (1992)
28. N. H. Abel. Mémoire sur les équations algébriques, où l'on démontre l'impossibilité de la résolution de l'équation générale du cinquième degré. Oeuvres Complètes. Suivi de Niels Henrik Abel, Sa Vie et Son Action Scientifique, deuxiéme edition Jacques Gabay, vol. I, pp. 28–33 (1992)
29. N. H. Abel. Sur la resolution algébrique des équations. Oeuvres Complètes. Suivi de Niels Henrik Abel, Sa Vie et Son Action Scientifique, deuxiéme edition Jacques Gabay, vol. II, pp. 217–243 (1992)
30. H. Rutishauser and H. R. Schwarz, The *LR* transformation method for symmetric matrices. Numerische Mathematik, vol. 5, no. 1, pp. 273–289 (December 1963)
31. H. Rutishauser, Der quotienten-differenzen-algorithmus. Zeitschrift für Angewandte Mathematik und Physik, vol. 5, no. 3, pp. 233–251 (May 1954)
32. H. Rutishauser, Anwendungen des quotienten-differenzen-algorithmus. Zeitschrift für Angewandte Mathematik und Physik, vol. 5, no. 6, pp. 496–508 (November 1954)
33. H. Rutishauser, Une méthode pour la détermination des valeurs propres d'une matrice. Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, vol. 240, pp. 34–36 (January-March 1955)
34. J. G. F. Francis, The QR transformation, a unitary analogue to the LR transformation — part 1. The Computer Journal, vol. 4, no. 3, pp. 265–271 (1961)
35. V. N. Kublanovskaya, On some algorithms for the solution of the complete eigenvalue problem. Computational Mathematics and Mathematical Physics, vol. 3, pp. 637–657 (1961)
36. J. G. F. Francis, The QR transformation — part 2. The Computer Journal, vol. 4, no. 4, pp. 332–345 (1962)
37. D. S. Watkins, *QR*-like algorithms for eigenvalue problems. Journal of Computational and Applied Mathematics, vol. 123, no. 1–2, pp. 67–83 (November 2000)
38. D. S. Watkins, The *QR* algorithm revisited. SIAM Review, vol. 50, no. 1, pp. 133–145 (2008)
39. E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK Users' Guide. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, third edition, (1999)
40. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1997)

41. D. Day. How the QR algorithm fails to converge and how to fix it. Technical Report 96-0913J, Sandia National Laboratory, USA (April 1996)
42. B. N. Parlett, Convergence of the QR algorithm. Numerische Mathematik, vol. 7, no. 2, pp. 187–193 (April 1965)
43. B. N. Parlett, Global convergence of the basic QR algorithm on Hessenberg matrices. Mathematics of Computation, vol. 22, no. 104, pp. 803–817 (October 1968)
44. B. N. Parlett, The Symmetric Eigenvalue Problem. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1998)
45. D. S. Watkins, Understanding the QR algorithm. SIAM Review, vol. 24, no. 4, pp. 427–440 (October 1982)
46. B. N. Parlett and Jr. W. G. Poole, A geometric theory for the QR, LU, and power iterations. SIAM Journal on Numerical Analysis, vol. 10, no. 2, pp. 389–412 (April 1973)
47. E. Schmidt, Zur theorie der linearen und nichtlinearen integralgleichungen i. teil: Entwicklung willkürlicher funktionen nach systemen vorgeschriebener. Mathematische Annalen, vol. 63, no. 4, pp. 433–476 (December 1907)
48. J. P. Gram, Ueber entwickelung reeller functionen in reihen mittelst der methode der kleinsten quadrate. Journal für die reine und angewandte Mathematik, vol. 94, pp. 71–73 (1883)
49. Y. K. Wong, An application of orthogonalization process to the theory of least squares. The Annals of Mathematical Statistics, vol. 6, no. 2, pp. 53–75 (June 1935)
50. S. J. Leon, W. Gander, J. Langou, and Å. Björck, Gram-Schmidt orthogonalization: 100 years and more. Manuscript to appear (2008)
51. Å. Björck, Solving linear least squares problems by Gram-Schmidt orthogonalization. BIT Numerical Mathematics, vol. 7, no. 1, pp. 1–21 (March 1967)
52. Å. Björck, Numerics of Gram-Schmidt orthogonalization. Linear Algebra and its Applications, vol. 197–198, pp. 297–316 (January–February 1994)
53. W. Gander. Algorithms for the QR-decomposition. Research Report 80-02, Eidgenoessische Technische Hochschule, Zurich, Switzerland (April 1980)
54. A. S. Householder, Unitary triangularization of a nonsymmetric matrix. Journal of the ACM, vol. 5, no. 4, pp. 339–342 (October 1958)
55. J. H. Wilkinson, Householder's method for the solution of the algebraic eigenproblem. The Computer Journal, vol. 3, no. 1, pp. 23–27 (1960)
56. W. Givens, Computation of plane unitary rotations transforming a general matrix to triangular form. Journal of the Society for Industrial and Applied Mathematics, vol. 6, no. 1, pp. 26–50 (March 1958)
57. E. E. Osborne, On least squares solutions of linear equations. Journal of the ACM, vol. 8, no. 4, pp. 628–636 (October 1961)
58. A. S. Householder, The approximate solution of matrix problems. Journal of the ACM, vol. 5, no. 3, pp. 205–243 (July 1958)
59. J. H. Wilkinson, Error analysis of direct methods of matrix inversion. Journal of the ACM, vol. 8, no. 3, pp. 281–330 (July 1961)
60. Å. Björck, Iterative refinement of linear least squares solutions I. BIT Numerical Mathematics, vol. 7, no. 4, pp. 257–278 (December 1967)
61. Å. Björck, Iterative refinement of linear least squares solutions II. BIT Numerical Mathematics, vol. 8, no. 1, pp. 8–30 (March 1968)
62. Å. Björck and G. H. Golub, Iterative refinement of linear least squares solutions by Householder transformation. BIT Numerical Mathematics, vol. 7, no. 4, pp. 322–337 (December 1967)
63. Å. Björck, Numerical Methods for Least Squares Problems. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1996)
64. R. W. Farebrother, Linear Least Squares Computations. Marcel Dekker, Inc., New York, NY, USA (1988)
65. C. F. Gauss, Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium. Perthes and Besser, Hamburg (1809)

66. A. M. Legendre, Nouvelles Méthodes pour la Détermination des Orbites des Comètes. Courcier, Paris, France (1805)
67. G. H. Golub, Numerical methods for solving linear least squares problems. Numerische Mathematik, vol. 7, no. 3, pp. 206–216 (June 1965)
68. P. Businger and G. H. Golub, Linear least squares solutions by Householder transformations. Numerische Mathematik, vol. 7, no. 3, pp. 269–276 (June 1965)
69. W. M. Gentleman, Least squares computations by Givens transformations without square roots. IMA Journal of Applied Mathematics, vol. 12, no. 3, pp. 329–336 (1973)
70. W. M. Gentleman, Basic procedures for large, sparse or weighted linear least squares problems (algorithm AS 75). Applied Statistics, vol. 23, no. 3, pp. 448–454 (1974)
71. S. Hammarling, A note on the modifications to the Givens plane rotations. IMA Journal of Applied Mathematics, vol. 13, no. 2, pp. 215–218 (April 1974)
72. W. M. Gentleman, Error analysis of QR decompositions by Givens transformations. Linear Algebra and Its Applications, vol. 10, no. 3, pp. 189–197 (June 1975)
73. H. T. Kung, Why systolic architectures? Computer, vol. 15, no. 1, pp. 37–46 (January 1982)
74. W. M. Gentleman and H. T. Kung, Matrix triangularization by systolic arrays. Real-Time Signal Processing IV, SPIE Proceedings, vol. 298, pp. 19–26 (1981)
75. A. Bojanczyk, R. P. Brent, and H. T. Kung, Numerically stable solution of dense systems of linear equations using mesh-connected processors. SIAM Journal on Scientific and Statistical Computing, vol. 5, no. 1, pp. 95–104 (1984)
76. J. G. McWhirter, Recursive least-squares minimization using a systolic array. Real-Time Signal Processing VI, SPIE Proceedings, vol. 431, pp. 105–112 (1983)
77. J. G. McWhirter, Systolic array for recursive least-squares minimisation. Electronics Letters, vol. 19, no. 18, pp. 729–730 (September 1983)
78. C. R. Ward, P. J. Hargrave, and J. G. McWhirter, A novel algorithm and architecture for adaptive digital beamforming. IEEE Transactions on Antennas and Propagation, vol. AP-34, no. 3, pp. 338–346 (March 1986)
79. A. P. Varvitsiotis, S. Theodoridis, and G. Moustakides, A novel structure for adaptive LS FIR filtering based on QR decomposition. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP' 89, vol. 2, pp. 904–907 (May 1989)
80. J. M. Cioffi, A fast QR/frequency-domain RLS adaptive filter. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'87, vol. 12, pp. 407–410 (April 1987)
81. J. M. Cioffi, High-speed systolic implementation of fast QR adaptive filters. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'88, vol. 3, pp. 1584–1587 (April 1988)
82. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, no. 4, pp. 631–653 (April 1990)
83. M. Bellanger, A survey of QR based fast least squares adaptive filters: From principles to realization. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'91, vol. 3, pp. 1833–1836 (April 1991)
84. A. L. Ghirnikar and S. T. Alexander, Stable recursive least squares filtering using an inverse QR decomposition. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'90, vol. 3, pp. 1623–1626 (April 1990)
85. A. L. Ghirnikar, Performance and implementation of the inverse QR adaptive filter. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'92, vol. 4, pp. 29–32 (March 1992)
86. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)
87. J. G. McWhirter and I. K. Proudler, On the formal derivation of a systolic array for recursive least squares estimation. International Conference on Control, vol. 2, pp. 1272–1277 (March 1994)
88. M. Bellanger, The potential of QR adaptive filter variables for signal analysis. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'89, vol. 4, pp. 2166–2169 (May 1989)

89. M. Bellanger, Sur la dualité entre la prédiction linéaire d'un signal et la decomposition QR. Douzième Colloque Gretsi - Juan-les-Pins, pp. 13–16 (June 1989)
90. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 879–891 (April 1991)
91. I. K. Proudler, T. J. Shepherd, and J. G. McWhirter, Computationally efficient QRD-based wide-band beamforming. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'90, vol. 3, pp. 1799–1802 (April 1990)
92. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, Computationally efficient QR decomposition approach to least squares adaptive filtering. IEE Proceedings F, Radar and Signal Processing, vol. 138, no. 4, pp. 341–353 (August 1991)
93. C. J. Demeure and L. L. Scharf, True lattice algorithms for square root solution of least squares linear prediction problems. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'88, vol. 4, pp. 2312–2315 (April 1988)
94. C. P. Rialan, Fast algorithms for QR and Cholesky factors of Toeplitz operators. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'87, vol. 12, pp. 41–44 (April 1987)
95. C. P. Rialan and L. L. Scharf, Fast algorithms for computing QR and Cholesky factors of Toeplitz operators. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 36, no. 11, pp. 1740–1748 (November 1988)
96. J. G. McWhirter, Algorithmic engineering in digital signal processing. Second International Specialist Seminar on the Design and Application of Parallel Digital Processors, pp. 11–18 (April 1991)
97. J. G. McWhirter, Algorithmic engineering in adaptive signal processing. IEE Proceedings F, Radar and Signal Processing, vol. 139, no. 3, pp. 226–232 (June 1992)
98. I. K. Proudler and J. G. McWhirter, Algorithmic engineering in adaptive signal processing: worked examples. IEE Proceedings — Vision, Image and Signal Processing, vol. 141, no. 1, pp. 19–26 (February 1994)
99. I. K. Proudler, J. G. McWhirter, and M. Moonen, On the formal derivation of a systolic array for recursive least squares estimation. IEEE International Symposium on Circuits and Systems, ISCAS'94, vol. 2, pp. 329–332 (June 1994)
100. I. K. Proudler, J. G. McWhirter, M. Moonen, and G. Hekstra, Formal derivation of a systolic array for recursive least squares estimation. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 43, no. 3, pp. 247–254 (March 1996)
101. M. Moonen and I. K. Proudler, Generating 'fast QR' algorithms using signal flow graph techniques. Thirtieth Asilomar Conference on Signals, Systems and Computers, vol. 1, pp. 410–414 (November 1996)
102. M. Harteneck, J. G. McWhirter, I. K. Proudler, and R. W. Stewart, Algorithmically engineered fast multichannel adaptive filter based on QR-RLS. IEE Proceedings — Vision, Image and Signal Processing, vol. 146, no. 1, pp. 7–13 (February 1999)

Chapter 2
Introduction to Adaptive Filters

José A. Apolinário Jr. and Sergio L. Netto

Abstract This chapter introduces the general concepts of adaptive filtering and its families of algorithms, and settles the basic notation used in the remaining of the book. Section 2.1 presents the fundamentals concepts, highlighting several configurations, such as system identification, interference cancelation, channel equalization, and signal prediction, in which adaptive filters have been successfully applied. The main objective functions associated to optimal filtering are then introduced in Section 2.2, followed, in Section 2.3, by the corresponding classical algorithms, with emphasis given to the least-mean square, data-reusing, and recursive least-squares (RLS) families of algorithms. It is observed that RLS algorithms based on the so-called QR decomposition combines excellent convergence speed with good numerical properties in finite-precision implementations. Finally, computer simulations are presented in Section 2.4, illustrating some convergence properties of the most important adaptation algorithms. For simplicity, all theoretical developments are performed using real variables, whereas the algorithm pseudo-codes are presented in their complex versions, for generality purposes.

2.1 Basic Concepts

In the last decades, the field of digital signal processing, and particularly adaptive signal processing, has developed enormously due to the increasingly availability of technology for the implementation of the emerging algorithms. These algorithms have been applied to an extensive number of problems including noise and

José A. Apolinário Jr.
Military Institute of Engineering (IME), Rio de Janeiro – Brazil
e-mail: apolin@ieee.org

Sergio L. Netto
Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro – Brazil
e-mail: sergioln@lps.ufrj.br

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_2, © Springer Science+Business Media, LLC 2009

echo canceling, channel equalization, signal prediction, adaptive arrays as well as many others.

> An adaptive filter may be understood as a self-modifying digital filter that adjusts its coefficients in order to minimize an error function. This error function, also referred to as the cost function, is a distance measurement between the *reference* or *desired* signal and the output of the adaptive filter.

Adaptive filtering algorithms, which constitute the adjusting mechanism for the filter coefficients, are in fact closely related to classical optimization techniques although, in the latter, all calculations are carried out in an *off-line* manner. Moreover, an adaptive filter, due to its real-time self-adjusting characteristic, is sometimes expected to track the optimum behavior of 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 [1–3]:

- Filter structure: The input–output relationship of the adaptive filter depends on its transfer function implementation. Due to its simplicity and efficacy, the most widely employed adaptive structure is by far the transversal filter (or tapped-delay line) associated to standard finite-duration impulse response (FIR) filters. Other structures comprise FIR lattice and infinite-duration impulse response (IIR) filters. This aspect greatly influences the computational complexity of a given adaptive algorithm and the overall speed of the adaptation process.
- Rate of convergence, misadjustment, and tracking: In a noiseless (no measurement or modeling noise) situation, the coefficients of an adaptive filter can be made to converge fast or slowly to the optimum solution. In practice, the adaptive coefficients do not reach the optimum values but stay close to the optimum. Misadjustment is a measure of excess error associated to how close these coefficients (the estimated and the optimum) are to each other in steady-state. It can be taken as a general rule that, for a given algorithm, a faster convergence yields a higher misadjustment. In non-stationary environments, the algorithm convergence speed is also associated to the tracking ability of the adaptive filter.
- Computational aspects: Due to the desired real-time characteristic, the adaptive filter performance must take into account practical levels of computational complexity and limited-precision representation of associated signals and coefficients. The effort in obtaining fast versions of more complex algorithms results from the desire of reducing the computational requirements to a minimal number of operations, as well as reducing the size of memory necessary to run these algorithms in practical applications. On the other hand, a limited-precision environment generates quantization errors which drive the attention of designers to numerical stability, numerical accuracy, and convergence robustness of the algorithm.

The basic configuration of an adaptive filter, operating in the discrete-time domain k, is illustrated in Figure 2.1. In such a scheme, the input signal is denoted

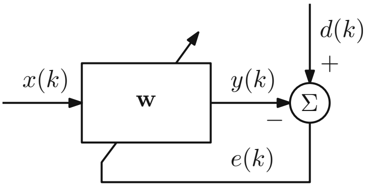


Fig. 2.1 Basic block diagram of an adaptive filter.

by $x(k)$, the reference signal $d(k)$ represents the desired output signal (that usually includes some noise component), $y(k)$ is the output of the adaptive filter, and the error signal is defined as $e(k) = d(k) - y(k)$.

The error signal is used by the adaptation algorithm to update the adaptive filter coefficient vector $\mathbf{w}(k)$ according to some performance criterion. In general, the whole adaptation process aims at minimizing some metric of the error signal, forcing the adaptive filter output signal to approximate the reference signal in a statistical sense.

It is interesting to notice how this basic configuration fits perfectly in several practical applications such as system identification, interference canceling, channel equalization, and signal prediction [1–3], which are detailed as follows.

For instance, Figure 2.2 depicts a typical system identification configuration, where $\mathbf{w}_o$ is an ideal coefficient vector of an unknown plant, whose output is represented by $y_o(k)$, and $n(k)$ denotes the observation or measurement noise. In this setup, the plant and the adaptive filter receive the same input signal. After convergence, the output signals of both systems become similar, and consequently the adaptive transfer function becomes a good model for the input–output relationship of the plant.

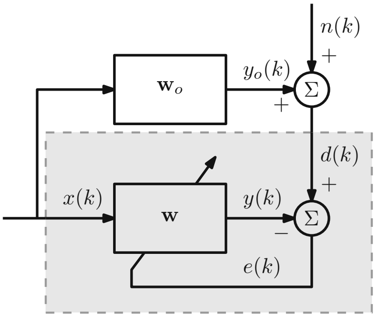


Fig. 2.2 System identification configuration of an adaptive filter: The adaptive coefficient **w** vector estimates the unknown system coefficient vector $\mathbf{w}_o$.

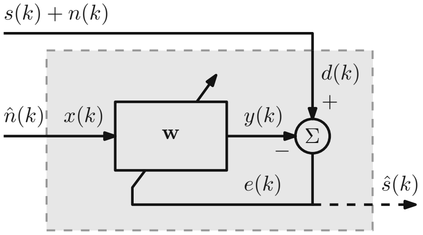


Fig. 2.3 Interference cancelation configuration of an adaptive filter: The error signal $e(k)$ approximates the desired signal component $s(k)$ if $n(k)$ and $\hat{n}(k)$ are correlated.

Another application of an adaptive filter is interference canceling or signal enhancement represented in Figure 2.3. In this problem, a signal of interest $s(k)$ is corrupted by a noise component $n(k)$. A cleaner version of $s(k)$ is desired but cannot be obtained directly in practice. The noisy signal, $s(k)+n(k)$, is then employed as the reference signal for the adaptive filter, whose input must be another version, $\hat{n}(k)$, of the noise signal, strongly correlated to $n(k)$. The adaptive mechanism adjusts the filter coefficients in such a manner that the filter output $y(k)$ approximates $n(k)$, thus forcing the error signal $e(k)$ to resemble signal $s(k)$.

In practical communications systems, a transmitted signal can be heavily distorted by the transmission channel. One may attempt to recover the original signal by employing an adaptive filter in the channel equalization configuration, as depicted in Figure 2.4. In such a framework, a training sequence $s(k)$ known by the receiver is sent via a given channel generating a distorted signal. The same sequence $s(k)$, after a proper time shift to compensate for transmission delays, is used as a reference signal in the receiver for the adaptive filter, whose input is the distorted signal. When the error function approximates zero, the output signal $y(k)$ resembles the transmitted signal $s(k)$, indicating that the adaptive filter is compensating for the channel distortions. After this training process, the desired information can be sent through the channel, which is properly equalized by the adaptive filter.

The adaptive predictor configuration is depicted in Figure 2.5. In this case, the adaptive filter input signal $x(k)$ is a delayed version of the reference signal $d(k)$.

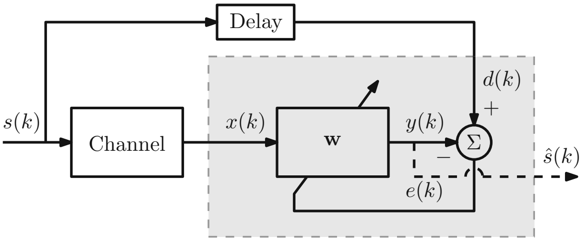


Fig. 2.4 Channel equalization configuration of an adaptive filter: The output signal $y(k)$ estimates the transmitted signal $s(k)$.

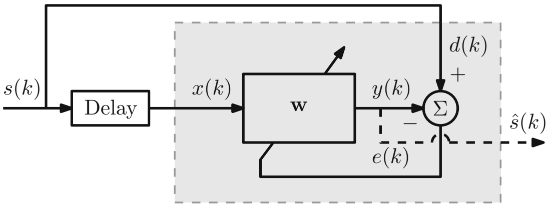


Fig. 2.5 Predictor configuration of an adaptive filter: The output signal $y(k)$ estimates the present input sample $s(k)$ based on past values of this same signal.

Therefore, when the adaptive filter output $y(k)$ approximates the reference, the adaptive filter operates as a predictor system.

From the discussion so far, one observes that the reference signal, through the definition of the error signal, acts as a general guide for the entire adaptation process. The four configurations illustrated above indicate how one can determine the desired output signal in several practical situations. In all cases, one can clearly identify the adaptive filter block given in Figure 2.1. To completely characterize this common basic cell, three main aspects must be defined:

1. Adaptive filter structure: This book will focus on the adaptive transversal FIR structure, whose input–output relationship is described by

$$\begin{aligned} y(k) &= w_0 x(k) + w_1 x(k-1) + \cdots + w_N x(k-N) \\ &= \sum_{i=0}^{N} w_i x(k-i) \\ &= \mathbf{w}^{\mathrm{T}} \mathbf{x}(k), \end{aligned} \tag{2.1}$$

where N is the filter order and $\mathbf{x}(k)$ and $\mathbf{w}$ are vectors composed by the input-signal samples and the filter coefficients, respectively; that is

$$\mathbf{x}(k) = [x(k)\; x(k-1)\; \ldots\; x(k-N)]^{\mathrm{T}}, \tag{2.2}$$

$$\mathbf{w} = [w_0\; w_1\; \ldots\; w_N]^{\mathrm{T}}. \tag{2.3}$$

In cases of complex implementations, the output signal is represented as $\mathbf{w}^{\mathrm{H}}\mathbf{x}(k)$, where the superscript H denotes the Hermitian operator (transpose and complex conjugate).
2. Error metric: As mentioned before, the adaptation algorithms adjust the adaptive filter coefficients in an attempt to minimize a given error norm. Different metrics yield adaptation processes with quite distinct characteristics. The commonly employed processes are discussed in detail in Section 2.2.
3. Adaptation algorithm: Several optimization procedures can be employed to adjust the filter coefficients, including, for instance, the least mean-square (LMS) and its normalized version, the data-reusing (DR) including the affine projection

(AP), and the recursive least-squares (RLS) algorithms. All these schemes are discussed in Section 2.3, emphasizing their main convergence and implementation characteristics. The remaining of the book focuses on the RLS algorithms, particularly, those employing QR decomposition, which achieve excellent overall convergence performance.

2.2 Error Measurements

Adaptation of the filter coefficients follows a minimization procedure of a particular objective or cost function. This function is commonly defined as a norm of the error signal $e(k)$. The three most commonly employed norms are the mean-square error (MSE), the instantaneous square error (ISE), and the weighted least-squares (WLS), which are introduced below.

2.2.1 The mean-square error

The MSE is defined as

$$\xi(k) = E[e^2(k)] = E[|d(k) - y(k)|^2]. \tag{2.4}$$

Writing the output signal $y(k)$ as given in Equation (2.1), one obtains

$$\begin{aligned}\xi(k) &= E[|d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)|^2] \\ &= E[d^2(k)] - 2\mathbf{w}^{\mathrm{T}}E[d(k)\mathbf{x}(k)] + \mathbf{w}^{\mathrm{T}}E[\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)]\mathbf{w} \\ &= E[d^2(k)] - 2\mathbf{w}^{\mathrm{T}}\mathbf{p} + \mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w},\end{aligned} \tag{2.5}$$

where $\mathbf{R}$ and $\mathbf{p}$ are the input-signal correlation matrix and the cross-correlation vector between the reference signal and the input signal, respectively, and are defined as

$$\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)], \tag{2.6}$$

$$\mathbf{p} = E[d(k)\mathbf{x}^{\mathrm{T}}(k)]. \tag{2.7}$$

Note, from the above equations, that $\mathbf{R}$ and $\mathbf{p}$ are not represented as a function of the iteration k or not time-varying, due to the assumed stationarity of the input and reference signals.

From Equation (2.5), the gradient vector of the MSE function with respect to the adaptive filter coefficient vector is given by

$$\nabla_{\mathbf{w}}\xi(k) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}. \tag{2.8}$$

The so-called Wiener solution $\mathbf{w}_o$, that minimizes the MSE cost function, is obtained by equating the gradient vector in Equation (2.8) to zero. Assuming that $\mathbf{R}$ is non-singular, one gets that

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}. \tag{2.9}$$

2.2.2 The instantaneous square error

The MSE is a cost function that requires knowledge of the error function $e(k)$ at all time k. For that purpose, the MSE cannot be determined precisely in practice and is commonly approximated by other cost functions. The simpler form to estimate the MSE function is to work with the ISE given by

$$\hat{\xi}(k) = e^2(k). \tag{2.10}$$

In this case, the associated gradient vector with respect to the coefficient vector is determined as

$$\begin{aligned}\nabla_{\mathbf{w}}\hat{\xi}(k) &= 2e(k)\nabla_{\mathbf{w}}e(k)\\ &= 2e(k)\nabla_{\mathbf{w}}\left[d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)\right]\\ &= -2e(k)\mathbf{x}(k).\end{aligned} \tag{2.11}$$

This vector can be seen as a noisy estimate of the MSE gradient vector defined in Equation (2.8) or as a precise gradient of the ISE function, which, in its own turn, is a noisy estimate of the MSE cost function seen in Section 2.2.1.

2.2.3 The weighted least-squares

Another objective function is the WLS function given by

$$\xi_D(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(i)]^2 \tag{2.12}$$

where $0 \ll \lambda < 1$ is the so-called *forgetting factor*. The parameter λ^{k-i} emphasizes the most recent error samples (where $i \approx k$) in the composition of the deterministic cost function $\xi_D(k)$, giving to this function the ability of modeling non-stationary processes. In addition, since the WLS function is based on several error samples, its stochastic nature reduces in time, being significantly smaller than the noisy ISE nature as k increases.

Defining the auxiliary variables

$$\mathbf{d}(k) = [d(k)\ \lambda^{1/2}d(k-1)\ \ldots\ \lambda^{k/2}d(0)]^{\mathrm{T}}, \tag{2.13}$$

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}^{\mathrm{T}}(0) \end{bmatrix}, \tag{2.14}$$

with $\mathbf{x}(k)$ as defined in Equation (2.2), one may rewrite Equation (2.12) as

$$\xi_D(k) = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k), \tag{2.15}$$

where

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}. \tag{2.16}$$

The corresponding WLS gradient vector is given by

$$\begin{aligned} \nabla_{\mathbf{w}}\xi_D(k) &= -2\sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)\left[d(i) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(i)\right] \\ &= -2\mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) + 2\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\mathbf{w} \\ &= -2\mathbf{p}(k) + 2\mathbf{R}(k)\mathbf{w}, \end{aligned} \tag{2.17}$$

where

$$\mathbf{R}(k) = \sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)\mathbf{x}^{\mathrm{T}}(i) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k), \tag{2.18}$$

$$\mathbf{p}(k) = \sum_{i=0}^{k}\lambda^{k-i}d(i)\mathbf{x}(i) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k), \tag{2.19}$$

are the deterministic counterparts of $\mathbf{R}$ and $\mathbf{p}$ defined in Equations (2.6) and (2.7), respectively. The optimum solution $\mathbf{w}(k)$ in the WLS sense is determined by equating the gradient vector $\nabla_{\mathbf{w}}\xi_D(k)$ to zero, yielding

$$\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k). \tag{2.20}$$

2.3 Adaptation Algorithms

In this section, a number of schemes are presented to find the optimal filter solution for the error functions seen in Section 2.2. Each scheme constitutes an adaptation algorithm that adjusts the adaptive filter coefficients in order to minimize the associated error norm.

The algorithms seen here can be grouped into three families, namely the LMS, the DR, and the RLS classes of algorithms. Each group presents particular characteristics of computational complexity and speed of convergence, which tend to determine the best possible solution to an application at hand.

2.3.1 LMS and normalized-LMS algorithms

Determining the Wiener solution for the MSE problem requires inversion of matrix **R**, which makes Equation (2.9) hard to implement in real time. One can then estimate the Wiener solution, in a computationally efficient manner, iteratively adjusting the coefficient vector **w** at each time instant k, in such a manner that the resulting sequence $\mathbf{w}(k)$ converges to the desired $\mathbf{w}_o$ solution, possibly in a sufficiently small number of iterations.

The so-called steepest-descent scheme searches for the minimum of a given function following the opposite direction of the associated gradient vector. A factor $\mu/2$, where μ is the so-called convergence factor, adjusts the step size between consecutive coefficient vector estimates, yielding the following updating procedure:

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{\mu}{2}\nabla_{\mathbf{w}}\xi(k). \tag{2.21}$$

This iterative procedure is illustrated in Figure 2.6 for the two-dimensional coefficient vector $\mathbf{w}(k) = [w_0(k)\ w_1(k)]^{\mathrm{T}}$ case.

The Wiener solution requires knowledge of the autocorrelation matrix **R** and the cross-correlation vector **p**. To do that, one must have access to the complete second-order statistics of signals $x(k)$ and $d(k)$, what makes Equation (2.9) unsuitable for most practical applications. Naturally, the Wiener solution can be approximated by a proper estimation of **R** and **p** based on sufficiently long time intervals, also assuming ergodicity. A rather simpler approach is to approximate the MSE by the ISE function, using the gradient vector of the latter, given in Equation (2.11), to adjust the coefficient vector in Equation (2.21). The resulting algorithm is the LMS algorithm characterized by

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{\mu}{2}\nabla_{\mathbf{w}}\hat{\xi}(k) = \mathbf{w}(k-1) + \mu e(k)\mathbf{x}(k), \tag{2.22}$$

where, in this iterative case,

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k). \tag{2.23}$$

The LMS algorithm is summarized in Table 2.1, where the superscripts $*$ and H denote the complex-conjugate and the Hermitian operations, respectively. Although behavior analysis of the LMS algorithm is beyond the scope of this work, it is important to mention that the step-size parameter μ plays an important role in the convergence characteristics of the algorithm as well as in its stability condition.

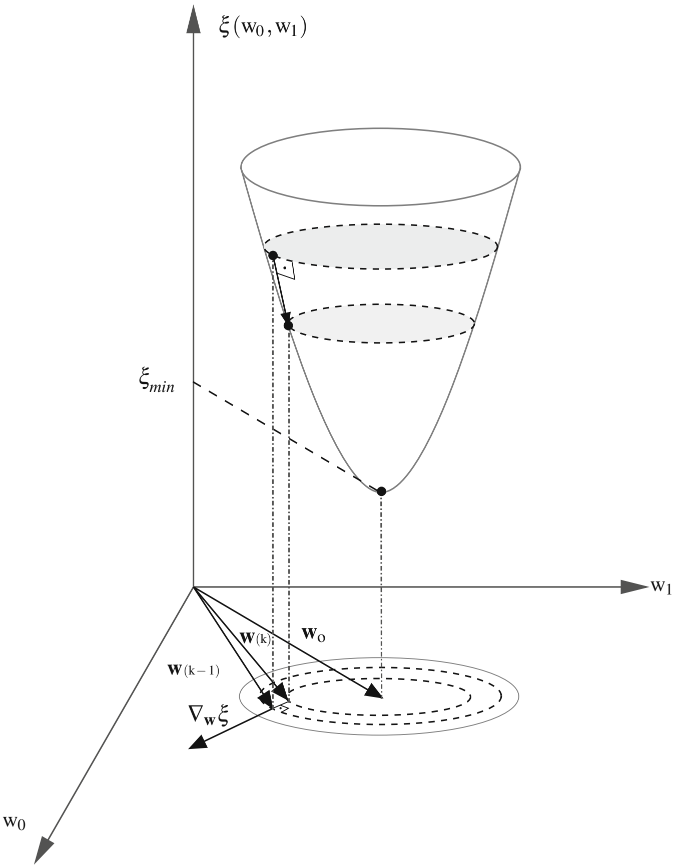


Fig. 2.6 Coefficient updating in a steepest-descent-based algorithm.

Table 2.1 The LMS algorithm.

```
LMS
Initialize $\mu$
for each $k$
{ $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$;
  $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu e^{*}(k)\mathbf{x}(k)$;
}
```

An approximation for the upperbound of this parameter is given in the technical literature and may be stated as [1–3]

$$0 < \mu < \frac{2}{\mathrm{tr}[\mathbf{R}]}, \tag{2.24}$$

where tr[.] denotes the trace operator of a matrix.

The LMS algorithm is very popular and has been widely used due to its extreme simplicity. Its convergence speed, however, is highly dependent on the condition number ρ of the input-signal autocorrelation matrix [1–3], defined as the ratio between the maximum and minimum eigenvalues of this matrix.

Alternative schemes which attempt to improve performance at the cost of minimum additional computational complexity have been proposed and are extensively discussed in [3, 4]. One approach that has been successfully employed in situations where signal statistics are unknown is the on-line calculation of the convergence factor which takes part in updating the filter coefficients [5, 6]. The normalized LMS (NLMS) algorithm can be included in this category [5, 7]. The NLMS algorithm normalizes the convergence factor such that the relation

$$\mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k) = d(k) = \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k) + \mu e(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) \tag{2.25}$$

is always satisfied. This results in a variable step-size parameter given by

$$\mu(k) = \frac{1}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k)}. \tag{2.26}$$

In practice, this parameter is modified to

$$\mu(k) = \frac{\bar{\mu}}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \varepsilon}, \tag{2.27}$$

where another fixed step-size, usually within the range $0 < \bar{\mu} \leq 1$, is used to control misadjustment[1] and convergence speed, and the parameter ε is a very small positive number that avoids possible divisions by zero.[2] Therefore, the NLMS updating equation is given by

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu(k)e(k)\mathbf{x}(k), \tag{2.28}$$

which is summarized in Table 2.2. In the NLMS algorithm, when $\bar{\mu} = 0$, one has $\mathbf{w}(k) = \mathbf{w}(k-1)$ and the updating halts. When $\bar{\mu} = 1$, the fastest convergence is attained at the price of a higher misadjustment then the one obtained for $0 < \bar{\mu} < 1$. Using $\bar{\mu} > 1$ is not a practical choice (although, theoretically, $\bar{\mu}$ can vary within the range $0 < \bar{\mu} < 2$), since it yields slower convergence rate and an even higher misadjustment than when $\bar{\mu} = 1$. Figure 2.7 depicts the theoretical misadjustment of the NLMS algorithm as a function of $\bar{\mu}$ in two cases of exact-order system identification: with small ($N = 5$) and large ($N = 50$) values of filter order. In both cases, the

[1] The misadjustment is defined as $M = (\xi(\infty) - \xi_{min})/\xi_{min}$ [3], where $\xi(\infty)$ and ξ_{min} denote the steady-state MSE yielded by the adaptation algorithm and the theoretical minimum value for the MSE, respectively.

[2] Some authors name this algorithm the ε-NLMS algorithm when this constant is employed [8].

Table 2.2 The NLMS algorithm.

```
NLMS
Initialize $\varepsilon \approx 0_+$ and $0 < \bar{\mu} \leq 1$
for each $k$
{ $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$;
  $\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\bar{\mu}}{\mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k)+\varepsilon} e^*(k)\mathbf{x}(k)$;
}
```

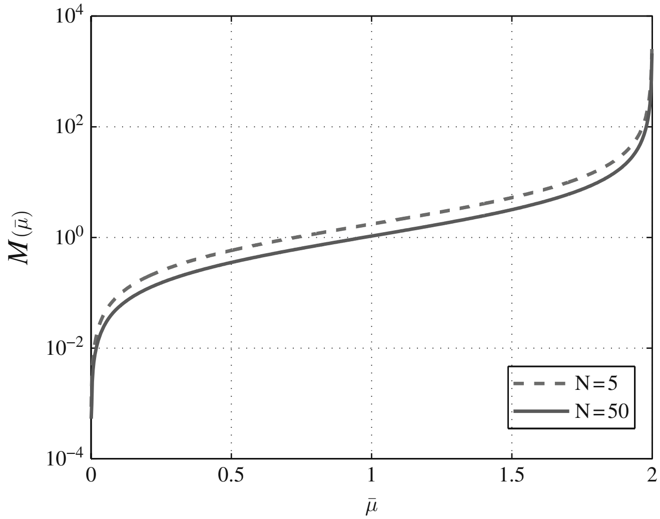


Fig. 2.7 Misadjustment value, as a function of the NLMS convergence factor, using white noise as input signal.

input signal consisted of a zero-mean Gaussian white noise with variance σ_x^2 and the misadjustment was approximated by [5]

$$M(\bar{\mu}) \approx \frac{\bar{\mu}}{2-\bar{\mu}} \frac{N+2}{N-1}. \tag{2.29}$$

2.3.2 *Data-reusing LMS algorithms*

As remarked before, the LMS algorithm estimates the MSE function with the current ISE value, yielding a noisy adaptation process. In this algorithm, information from each time sample k is disregarded in future coefficient updates. DR algorithms [9–11] employ present and past samples of the reference and input signals to improve convergence characteristics of the overall adaptation process.

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

$$\mathbf{w}_{i+1}(k) = \mathbf{w}_i(k) + \mu e_i(k)\mathbf{x}(k), \tag{2.30}$$

for $i = 0, 1, \ldots, L$, where

$$e_i(k) = d(k) - \mathbf{w}_i^{\mathrm{T}}(k)\mathbf{x}(k), \tag{2.31}$$

and

$$\mathbf{w}_0(k) = \mathbf{w}(k-1), \tag{2.32}$$

$$\mathbf{w}_{L+1}(k) = \mathbf{w}(k). \tag{2.33}$$

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

As for the LMS algorithm, the DR-LMS also has a normalized-DR (NDR) version, whose updating equation, for L data reuses, is given by

$$\mathbf{w}_{i+1}(k) = \mathbf{w}_i(k) + \frac{e_i(k)}{\mathbf{x}^{\mathrm{T}}(k-i)\mathbf{x}(k-i) + \varepsilon}\mathbf{x}(k-i) \tag{2.34}$$

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

$$e_i(k) = d(k-i) - \mathbf{w}_i^{\mathrm{T}}(k)\mathbf{x}(k-i), \tag{2.35}$$

with $\mathbf{w}_0(k) = \mathbf{w}(k-1)$ and $\mathbf{w}_{L+1}(k) = \mathbf{w}(k)$ as before.

Figure 2.8 provides a geometric interpretation for the coefficient vector upgrade for the LMS, NLMS, DR-LMS, and NDR-LMS algorithms in the two-dimensional case ($N = 1$). In this figure, $\mathscr{S}(k)$ denotes the hyperplane which contains all vectors

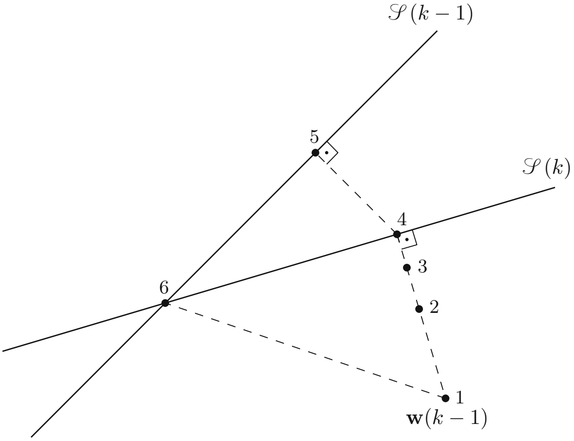


Fig. 2.8 Coefficient vector update for several LMS-type algorithms: 1. Initial vector $\mathbf{w}(k-1)$; 2. LMS and intermediary DR-LMS; 3. DR-LMS ($L = 1$); 4. NLMS and DR-LMS (for $L \to \infty$); 5. NDR-LMS; 6. BNDR-LMS.

$\mathbf{w}$ such that $\mathbf{w}^{\mathrm{T}}\mathbf{x}(k) = d(k)$ and the initial coefficient vector $\mathbf{w}(k-1)$ is indicated by position 1. In a noise-free exact-order modeling situation, $\mathscr{S}(k)$ would contain the optimal coefficient vector $\mathbf{w}_o$. In this scenario, it can be verified that $\mathbf{x}(k)$ and, consequently, the ISE gradient vector, is orthogonal to the hyperplane $\mathscr{S}(k)$. The LMS algorithm takes a single step towards $\mathscr{S}(k)$ yielding a new position represented by point 2 in Figure 2.8. The NLMS algorithm performs a line search in the direction of $\mathbf{x}(k)$ to reach the position represented by point 4, which belongs to $\mathscr{S}(k)$, in a unique iteration. The DR-LMS algorithm iteratively approaches $\mathscr{S}(k)$ by taking successive steps (within a single iteration) in the direction of $\mathbf{x}(k)$, as indicated in Equations (2.30) and (2.31). In Figure 2.8, with $L = 1$, positions 2 and 3 indicate the intermediary and final DR-LMS positions. It can be verified that the DR-LMS algorithm reaches $\mathscr{S}(k)$ in the limit, as the number of data reuses L approaches infinity [10, 11], turning this algorithm similar to the NLMS algorithm. From Equations (2.34) and (2.35), the NDR algorithm [11] employs more than one hyperplane, that is, uses more data pairs $\{\mathbf{x}(k-i), d(k-i)\}$, with $i > 0$, to adjust the coefficient vector $\mathbf{w}(k)$, closer to $\mathbf{w}_o$ than the adjustment obtained with only the current data pair $\{\mathbf{x}(k), d(k)\}$. In Figure 2.8, the NDR update is represented by point 5. Position 6 corresponds to the binormalized data-reusing (BNDR) LMS [4] algorithm addressed below.

For a noise-free exact-order modeling situation, the MSE optimal solution $\mathbf{w}_o$ is at the intersection of $(N+1)$ hyperplanes determined by $(N+1)$ linearly independent input-signal vectors. In this case, an orthogonal-projections algorithm [12] would reach $\mathbf{w}_o$ in exact $(N+1)$ iterations. This algorithm may be viewed as an orthogonal-NDR algorithm that performs exact line searches in $(N+1)$ orthogonal directions determined by the data pairs $\{\mathbf{x}(k-i), d(k-i)\}$, for $i = 0, 1, \ldots, N$. The BNDR algorithm [4] employs normalization on two orthogonal directions obtained from consecutive data pairs within each iteration. In simulations carried out with colored input signals, this algorithm presents faster convergence than all other data-reusing algorithms for the case of two data pairs, or, equivalently, $L = 1$ data reuse.

In order to state the BNDR problem, one may note that the solution which belongs to $\mathscr{S}(k)$ and $\mathscr{S}(k-1)$ at a minimum distance from $\mathbf{w}(k-1)$ is the one characterized by

$$\mathbf{w}(k) = \min_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}(k-1)\|^2, \tag{2.36}$$

subject to

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}(k) = d(k), \tag{2.37}$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}(k-1) = d(k-1). \tag{2.38}$$

Using Lagrange multipliers, the constraints above can be incorporated into Equation (2.36), resulting in the modified objective function

$$\begin{aligned} f[\mathbf{w}] = \|\mathbf{w} - \mathbf{w}(k-1)\|^2 &+ \lambda_1 \left[d(k) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k)\right] \\ &+ \lambda_2 \left[d(k-1) - \mathbf{w}^{\mathrm{T}}\mathbf{x}(k-1)\right], \end{aligned} \tag{2.39}$$

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

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\lambda_1}{2}\mathbf{x}(k) + \frac{\lambda_2}{2}\mathbf{x}(k-1), \tag{2.40}$$

where

$$\frac{\lambda_1}{2} = \frac{e(k)\|\mathbf{x}(k-1)\|^2 - \varepsilon(k-1)\mathbf{x}^{\mathrm{T}}(k-1)\mathbf{x}(k)}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}, \tag{2.41}$$

$$\frac{\lambda_2}{2} = \frac{\varepsilon(k-1)\|\mathbf{x}(k)\|^2 - e(k)\mathbf{x}^{\mathrm{T}}(k-1)\mathbf{x}(k)}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}, \tag{2.42}$$

with $e(k)$ as in Equation (2.23) and

$$\varepsilon(k-1) = d(k-1) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k-1). \tag{2.43}$$

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

$$\mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k-1) = d(k-1), \tag{2.44}$$

corresponding to $\varepsilon(k-1) = 0$, and a simplified version of the BNDR algorithm results:

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\lambda_1'}{2}\mathbf{x}(k) + \frac{\lambda_2'}{2}\mathbf{x}(k-1), \tag{2.45}$$

where

$$\frac{\lambda_1'}{2} = \frac{e(k)\|\mathbf{x}(k-1)\|^2}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}, \tag{2.46}$$

$$\frac{\lambda_2'}{2} = \frac{-e(k)\mathbf{x}^{\mathrm{T}}(k-1)\mathbf{x}(k)}{\|\mathbf{x}(k)\|^2\|\mathbf{x}(k-1)\|^2 - [\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k-1)]^2}. \tag{2.47}$$

It is worth mentioning that the excess MSE for either implementation of the BNDR-LMS algorithm, as in (2.40, 2.41, 2.42) or in (2.45, 2.46, 2.47), is close to the observation noise power when there is no modeling error, as expected from normalized algorithms. In order to control this excess MSE, a convergence parameter μ may be introduced into the algorithm, forcing a trade-off between convergence rate, maximized with $\mu = 1$, and lower steady-state MSE, associated to smaller values of μ, which may be required in applications with significant measurement error. With the introduction of μ, the coefficient vector $\mathbf{w}(k)$ at each iteration is not at the exact intersection of hyperplanes $\mathscr{S}(k-1)$ and $\mathscr{S}(k)$ and, therefore, the simplified version of the algorithm given by (2.45, 2.46, 2.47) should not be used.

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

Table 2.3 The BNDR-LMS algorithm.

```
BNDR-LMS
Initialize $\varepsilon \approx 0_+$
for each $k$
{ $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$;
  $\alpha = \mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k-1)$;
  $\rho(k) = \mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k)$;
  $D = \rho(k)\rho(k-1) - |\alpha|^2$;
  if $D < \varepsilon$
  { $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu e^*(k)\mathbf{x}(k)/\rho(k)$;
  }
  else
  { $\varepsilon(k-1) = d(k-1) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k-1)$;
    $\frac{\lambda_1}{2} = \left[e^*(k)\rho(k-1) - \varepsilon^*(k-1)\alpha\right]/D$;
    $\frac{\lambda_2}{2} = \left[\varepsilon^*(k-1)\rho(k) - e^*(k)\alpha^*\right]/D$;
    $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu\left[\frac{\lambda_1}{2}\mathbf{x}(k) + \frac{\lambda_2}{2}\mathbf{x}(k-1)\right]$;
  }
}
```

The BNDR-LMS algorithm is summarized in Table 2.3, where the denominator D in Equations (2.46) and (2.47) is determined recursively in time.

In Figure 2.8, the updating process was represented for a two-dimensional ($N = 1$) weight vector. In such a case, only one solution for $\mathbf{w}(k)$ was available as the intersection of two lines. In a more practical situation, the BNDR-LMS algorithm decreases in two the degree of freedom of $\mathbf{w}(k)$ by choosing the closest solution to $\mathbf{w}(k-1)$, following the least perturbation property [8] or the principle of minimal disturbance [2]. The three-dimensional case is depicted in Figure 2.9, where the updated coefficient vector $\mathbf{w}(k)$ is chosen closest to $\mathbf{w}(k-1)$

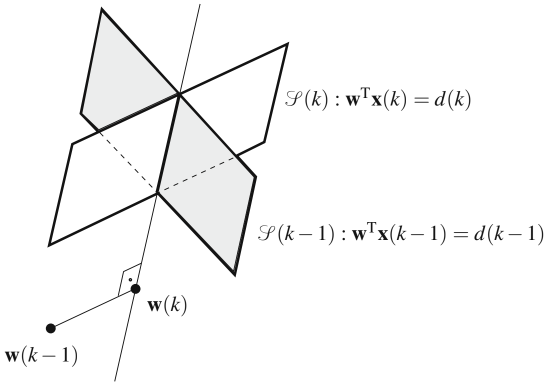

$\mathscr{S}(k) : \mathbf{w}^{\mathrm{T}}\mathbf{x}(k) = d(k)$
$\mathscr{S}(k-1) : \mathbf{w}^{\mathrm{T}}\mathbf{x}(k-1) = d(k-1)$
$\mathbf{w}(k)$
$\mathbf{w}(k-1)$

Fig. 2.9 Choice of the BNDR-LMS updated coefficient vector as the intersection of two hyperplanes, according to the minimal disturbance principle, assuming that $\mathbf{w}(k-1)$ does not belong to $\mathscr{S}(k-1)$.

and belonging to the intersection of the hyperplanes $\mathscr{S}(k) \cap \mathscr{S}(k-1)$. If one considers a third hyperplane, $\mathscr{S}(k-2)$ for instance, performing normalization in three directions, this would determine one single possibility for $\mathbf{w}(k)$, given by $\mathscr{S}(k) \cap \mathscr{S}(k-1) \cap \mathscr{S}(k-2)$.

As a generalization of the previous idea, the AP algorithm [13–15] is among the prominent adaptation algorithms that allow trade-off between fast convergence and low computational complexity. By adjusting the number of projections, or alternatively, the number of data reuses, one obtains adaptation processes ranging from that of the NLMS algorithm to that of the sliding-window RLS algorithm [16, 17].

The AP algorithm updates its coefficient vector such that the new solution belongs to the intersection of L hyperplanes defined by the present and the $(L-1)$ previous data pairs. The optimization criterion used for the derivation of the AP algorithm is given by

$$\mathbf{w}(k) = \min_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}(k-1)\|^2, \tag{2.48}$$

subject to

$$\mathbf{d}_L(k) = \mathbf{X}_L^{\mathrm{T}}(k)\mathbf{w}, \tag{2.49}$$

where

$$\mathbf{d}_L(k) = [d(k)\ d(k-1)\ \ldots\ d(k-L+1)]^{\mathrm{T}}, \tag{2.50}$$

$$\mathbf{X}_L(k) = [\mathbf{x}(k)\,\mathbf{x}(k-1)\ \ldots\ \mathbf{x}(k-L+1)]. \tag{2.51}$$

The updating equations for the AP algorithm obtained as the solution to the minimization problem in (2.48) are presented in Table 2.4 [13, 14]. To control stability, convergence speed, and misadjustment, a convergence factor, usually constrained to $0 < \mu < 1$, is introduced. In order to improve robustness, a diagonal matrix $\varepsilon\mathbf{I}$, with $\varepsilon > 0$, is employed to regularize the inverse operation required by the AP algorithm.

Table 2.4 The AP algorithm.

```
APA
Initialize $\varepsilon \approx 0_+$
for each $k$
{ $\mathbf{e}_L(k) = \mathbf{d}_L(k) - \mathbf{X}_L^{\mathrm{T}}(k)\mathbf{w}^*(k-1)$;
  $\mathbf{t}_k = \left[\mathbf{X}_L^{\mathrm{H}}(k)\mathbf{X}_L(k) + \varepsilon\mathbf{I}\right]^{-1}\mathbf{e}_L^*(k)$;
  $\mathbf{w}(k) = \mathbf{w}(k-1) + \mu\mathbf{X}_L(k)\mathbf{t}_k$;
}
```

2.3.3 RLS-type algorithms

This subsection presents the basic versions of the RLS family of adaptive algorithms. Importance of the expressions presented here cannot be overstated for they allow an easy and smooth reading of the forthcoming chapters.

> The RLS-type algorithms have a high convergence speed which is independent of the eigenvalue spread of the input correlation matrix. These algorithms are also very useful in applications where the environment is slowly varying. The price of all these benefits is a considerable increase in the computational complexity of the algorithms belonging to the RLS family.

In order to obtain the equations of the conventional RLS algorithm, the deterministic correlation matrix and cross-correlation vector defined in Equations (2.18) and (2.19), respectively, are rewritten as

$$\mathbf{R}(k) = \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda \mathbf{R}(k-1), \tag{2.52}$$

$$\mathbf{p}(k) = \mathbf{x}(k)d(k) + \lambda \mathbf{p}(k-1). \tag{2.53}$$

Using these recursive expressions into Equation (2.20), the following development can be made:

$$\begin{aligned}
\mathbf{w}(k) &= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)d(k) + \lambda \mathbf{p}(k-1)\right] \\
&= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)d(k) + \lambda \mathbf{R}(k-1)\mathbf{w}(k-1)\right] \\
&= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)d(k) + \lambda \mathbf{R}(k-1)\mathbf{w}(k-1)\right. \\
&\quad \left. + \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1) - \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)\right] \\
&= \mathbf{R}^{-1}(k)\left\{\mathbf{x}(k)\left[d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)\right] + \mathbf{R}(k)\mathbf{w}(k-1)\right\} \\
&= \mathbf{R}^{-1}(k)\left[\mathbf{x}(k)e(k) + \mathbf{R}(k)\mathbf{w}(k-1)\right],
\end{aligned} \tag{2.54}$$

and then

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\mathbf{R}^{-1}(k)\mathbf{x}(k). \tag{2.55}$$

In this updating expression, the computational burden for determining the inverse matrix $\mathbf{R}^{-1}(k)$ can be reduced significantly by employing the *matrix inversion lemma*[3] [3], which, in this case, yields that

$$\mathbf{R}^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \frac{\mathbf{R}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)}{\lambda + \mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}\right]. \tag{2.56}$$

[3] For suitable matrix orders, $[\mathbf{A} + \mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1}$.

For convenience of notation, consider the following auxiliary vectors:

$$\boldsymbol{\kappa}(k) = \mathbf{R}^{-1}(k)\mathbf{x}(k), \tag{2.57}$$
$$\mathbf{k}(k) = \mathbf{R}^{-1}(k-1)\mathbf{x}(k), \tag{2.58}$$

which in conjunction to Equation (2.56) yield that

$$\boldsymbol{\kappa}(k) = \frac{\mathbf{k}(k)}{\lambda + \mathbf{x}^{\mathrm{T}}(k)\mathbf{k}(k)}. \tag{2.59}$$

Hence, by substituting Equation (2.59) into Equation (2.56), one gets that

$$\begin{aligned}\mathbf{R}^{-1}(k) &= \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\right] \\ &= \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \boldsymbol{\kappa}(k)\mathbf{k}^{\mathrm{T}}(k)\right],\end{aligned} \tag{2.60}$$

and that the conventional RLS algorithm, requiring $\mathscr{O}[N^2]$ multiplications, can be implemented as indicated in Table 2.5. A few RLS-type algorithms requiring only $\mathscr{O}[N]$ multiplications, such as the fast transversal (FT) [18] and the lattice (L) [19] RLS algorithms, can be found in the technical literature.

An alternative RLS implementation with good numerical properties employs the so-called QR decomposition in the triangularization of the input data matrix combined with a back-substitution procedure [3] to obtain the coefficient vector.

It is worth mentioning that matrix $\mathbf{X}(k)$ is $(k+1)\times(N+1)$, which means that its order increases as the iterations progress. The QR-decomposition process applies an orthogonal matrix $\mathbf{Q}(k)$ of order $(k+1)\times(k+1)$ to transform $\mathbf{X}(k)$ into a triangular matrix $\mathbf{U}(k)$ of order $(N+1)\times(N+1)$ such that

$$\mathbf{Q}(k)\mathbf{X}(k) = \begin{bmatrix}\mathbf{O} \\ \mathbf{U}(k)\end{bmatrix} \tag{2.61}$$

where $\mathbf{O}$ is a null matrix of order $(k-N)\times(N+1)$. Matrix $\mathbf{Q}(k)$ represents the overall triangularization process and may be implemented in different ways as, for

Table 2.5 The RLS algorithm.

```
RLS
Initialize $0 \ll \lambda < 1$, $\varepsilon \approx 1/\sigma_x^2$, and $\mathbf{R}^{-1}(0) = \varepsilon\mathbf{I}$
for each $k$
{  $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$;
   $\mathbf{k}(k) = \mathbf{R}^{-1}(k-1)\mathbf{x}(k)$;
   $\boldsymbol{\kappa}(k) = \frac{\mathbf{k}(k)}{\lambda+\mathbf{x}^{\mathrm{H}}(k)\mathbf{k}(k)}$;
   $\mathbf{R}^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{R}^{-1}(k-1) - \frac{\boldsymbol{\kappa}(k)\boldsymbol{\kappa}^{\mathrm{H}}(k)}{\lambda+\mathbf{x}^{\mathrm{H}}(k)\mathbf{k}(k)}\right]$;
   $\mathbf{w}(k) = \mathbf{w}(k-1) + e^{*}(k)\boldsymbol{\kappa}(k)$;
}
```

instance, the numerically well-conditioned Givens rotations [2, 3, 8] or the Householder transformation [20, 21].

The main advantages associated to the QR-decomposition RLS (QRD-RLS) algorithms, as opposed to their conventional RLS counterpart, are the possibility of implementation in systolic arrays and the improved numerical behavior in limited precision environment.

The basic QRD [2, 3] and the so-called inverse QR (IQR) [2, 3, 22] RLS algorithms have a computational requirement of $\mathscr{O}[N^2]$ multiplications. A number of QR-based RLS versions requiring only $\mathscr{O}[N]$ multiplications, such as the fast QR (FQR) [23–28], the fast QR-Lattice (FQR-L) [29–31], are also available in the technical literature.

Since the scope of the next chapter encompasses both the basic (or conventional) and the inverse QRD-RLS algorithms, tables with their equations were not included at this point but left to be presented with their derivations.

2.4 Computer Simulations

This section presents simulations of system identification using some of the adaptation algorithms previously presented. In this framework, a few issues, such as misadjustment, convergence speed, tracking performance, and algorithm stability, are addressed in a practical point of view.

In the system identification setup implemented here, the plant is described by the transfer function $H(z) = -1 + 3z^{-1}$, such that $N = 1$ and $\mathbf{w}_o = [-1 \;\; 3]^{\mathrm{T}}$. An adaptive filter of correct order (exact modeling) is employed to identify $H(z)$. Therefore, $\mathbf{w}(k) = [w_0(k) \;\; w_1(k)]^{\mathrm{T}}$. In practical situations, under-modeling or over-modeling is usually a problem that requires special attention of the system designer [3]. In all cases, measurement noise $n(k)$, consisting of a zero-mean white Gaussian noise with variance $\sigma_n^2 = 10^{-6}$ and uncorrelated to the input signal $x(k)$, is added to the plant output signal.

2.4.1 Example 1: Misadjustment of the LMS algorithm

In a first example, consider that a zero-mean unit-variance white Gaussian noise is used as the input signal $x(k)$. The average ISE over an ensemble of 400 experiments is shown in Figure 2.10 for the LMS algorithm with $\mu = 0.22$ and $\mu = 0.013$. From this figure, one clearly notices how the step-size μ controls the trade-off between convergence speed and steady-state excess of average ISE (which approximates the MSE if the ensemble is large enough) which characterizes the misadjustment.

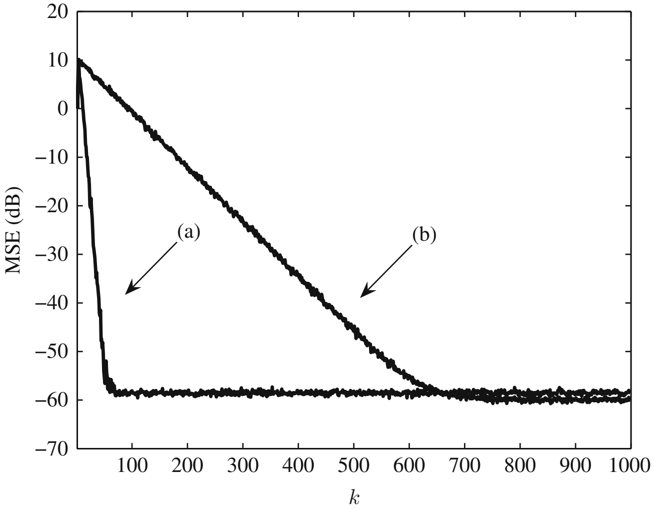


Fig. 2.10 Example 1: MSE convergence (*learning curve*) for the LMS algorithm with: (a) $\mu = 0.22$; (b) $\mu = 0.013$. The larger step-size parameter yields faster convergence and higher misadjustment.

2.4.2 Example 2: Convergence trajectories

In the previous case, since $x(k)$ consisted of a white noise, the LMS algorithm could achieve a very fast convergence regardless the initial position of the adaptive filter coefficient vector. If, however, the input signal is colored, the LMS convergence process becomes highly dependent on the initial value of $\mathbf{w}(k)$. Let then $x(k)$ be obtained by processing a zero-mean unit-variance white Gaussian noise with the filter $H(z) = 1/(1+1.2z^{-1}+0.81z^{-2})$. The resulting input signal autocorrelation matrix $\mathbf{R}$ presents a condition number around $\rho = 5$.

The LMS ($\mu = 0.005$) and RLS ($\lambda = 0.95$, $\varepsilon = 1$) coefficient trajectories for this setup are seen in Figure 2.11, where the crosses indicate distinct initial conditions for the adaptive coefficient vector and the circle indicates the optimum value $\mathbf{w}_o = [-1 \;\; 3]^{\mathrm{T}}$. From this figure, one observes how the LMS trajectories converge first to the main axis of the elliptic MSE contour lines, whereas the RLS trajectories follow a straight path to $\mathbf{w}_o$ regardless the initial value of $\mathbf{w}(k)$.

2.4.3 Example 3: Tracking performance

In this example, the adaptive filter performance is evaluated in a non-stationary environment. The idea is to perform an adaptive equalization of a channel, whose impulse response is initially given by

$$h_1(k) = e^{-k/5}[u(k) - u(k-5)], \tag{2.62}$$

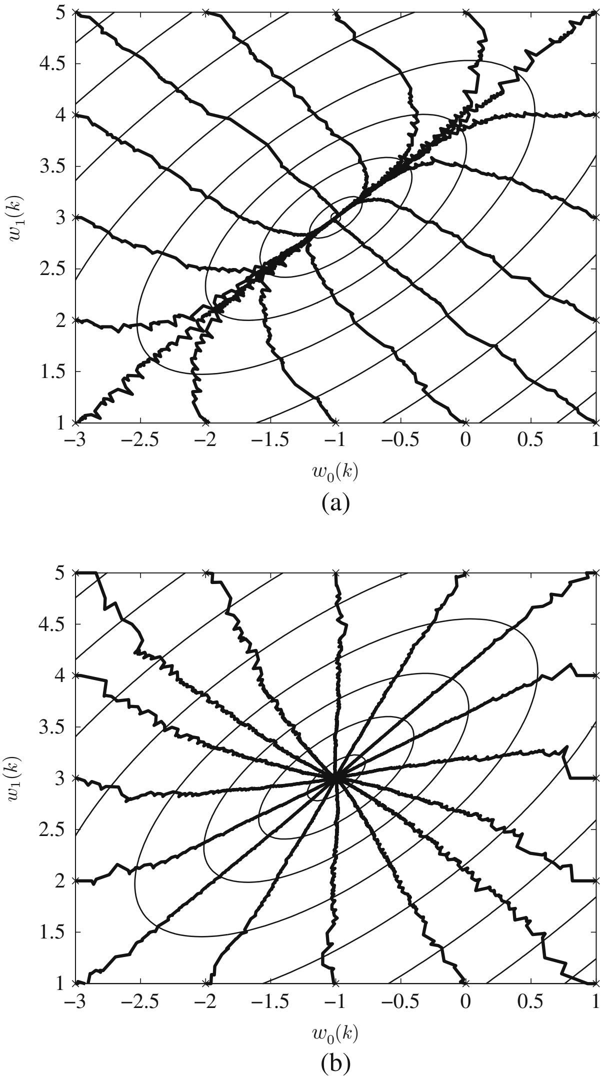


Fig. 2.11 Example 2: Convergence of adaptive coefficient vector – crosses indicate the initial position and circle indicates the MSE optimal solution: (a) LMS algorithm; (b) RLS algorithm.

where $u(k)$ is the unitary step sequence. It is then assumed that, for iteration 3000 on, the channel impulse response suddenly changes to

$$h_2(k) = h_1(k)e^{j\pi k}. \tag{2.63}$$

The adaptive filter order was set to $N = 49$, and the delay block, as in Figure 2.4, was set to $\Delta = 5$. A very small amount of additive noise (zero-mean white Gaussian noise with variance $\sigma_n^2 = 10^{-8}$) was included in the reference signal $d(k)$ to highlight the performance of the equalizer. The resulting MSE (average ISE over an ensemble of 1000 independent runs) is presented in Figure 2.12 for the NLMS, the BNDR-LMS, the AP (with $L = 5$), and the RLS (with $\lambda = 0.98$) algorithms, where in all cases the step-size was set to 1.

From Figure 2.12, one can conclude that all algorithms converge properly despite the sudden change in the environment. As expected, the algorithms considered converge with different speeds (the RLS algorithm being the fastest one) and misadjustment levels (the AP algorithm presenting the highest misadjustment amongst the LMS-type algorithms, since it uses $L = 5$, whereas the BNDR-LMS has $L = 2$ and the NLMS corresponds to having $L = 1$).

To verify the algorithm tracking ability, the channel was made to vary continuously. In this case, the impulse response was set to correspond to a five-tap Rayleigh fading channel with sampling frequency 1.25 MHz and Doppler frequency equal to 50 Hz. The 500 transmitted symbols are equalized by the same adaptive algorithms used in Figure 2.12, the only difference being that the RLS forgetting factor was set to $\lambda = 0.995$ in the present case. The resulting time-varying channel taps and the algorithm learning curves are shown in Figure 2.13. From this figure, one may observe that the RLS algorithm, although presenting a fast convergence, is not so well tuned for this application, since its MSE increases as time goes by. The reason for this is the forgetting factor λ, which should be decreased in order for the algorithm to remember only the most recent samples. However, this reduction cannot be implemented for the conventional RLS algorithm, as will be verified in Section 2.4.4.

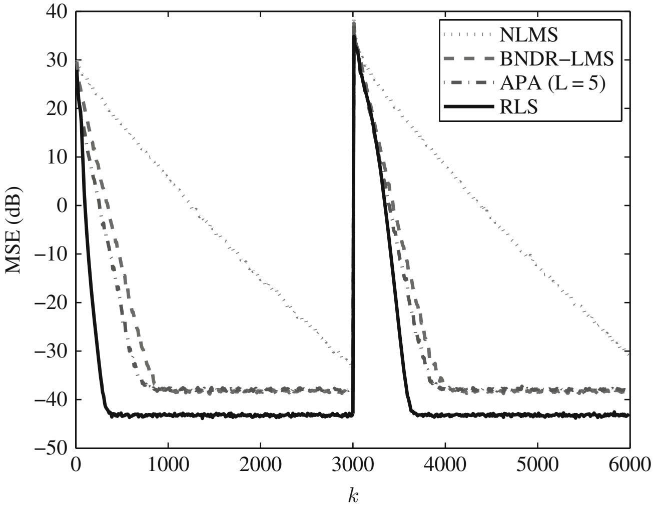
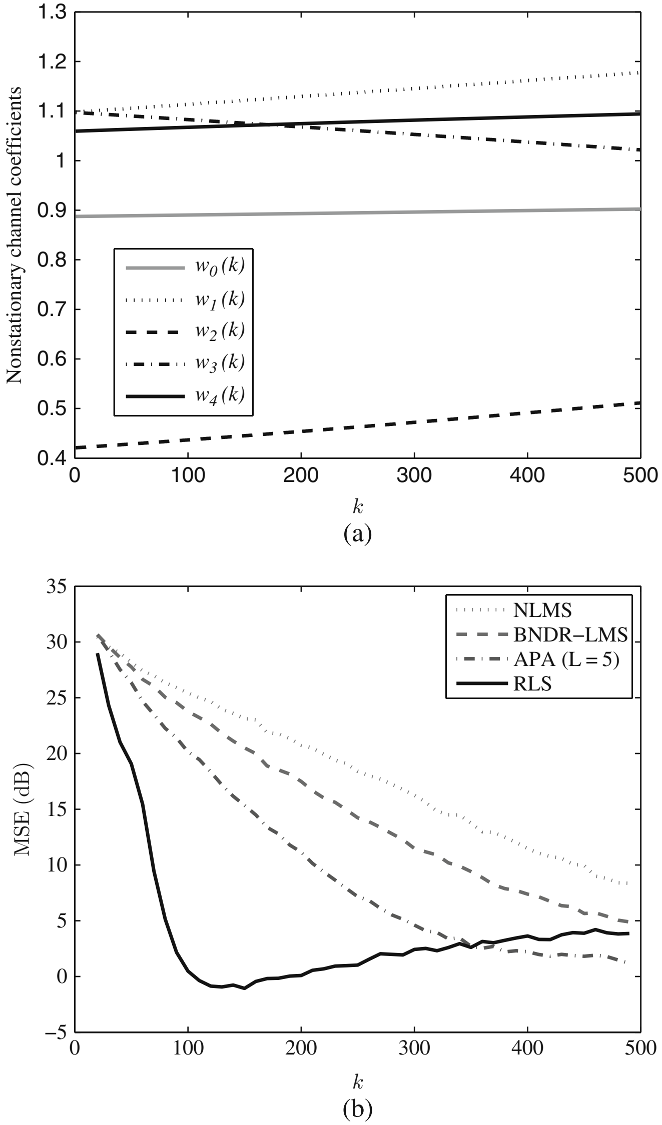


Fig. 2.12 Example 3: Resulting MSE in non-stationary channel equalization for the NLMS, BND-LMS, AP, and RLS algorithms.



(a)

(b)

Fig. 2.13 Example 3: (a) Time-varying channel coefficients; (b) Resulting MSE in non-stationary channel equalization for the NLMS, BND-LMS, AP, and RLS algorithms.

2.4.4 Example 4: Algorithm stability

Although, as previously seen, the conventional RLS algorithm is fast converging, presents low misadjustment, and is suitable for time-varying environments, a last experiment shows its vulnerability: the lack of numerical stability. In an

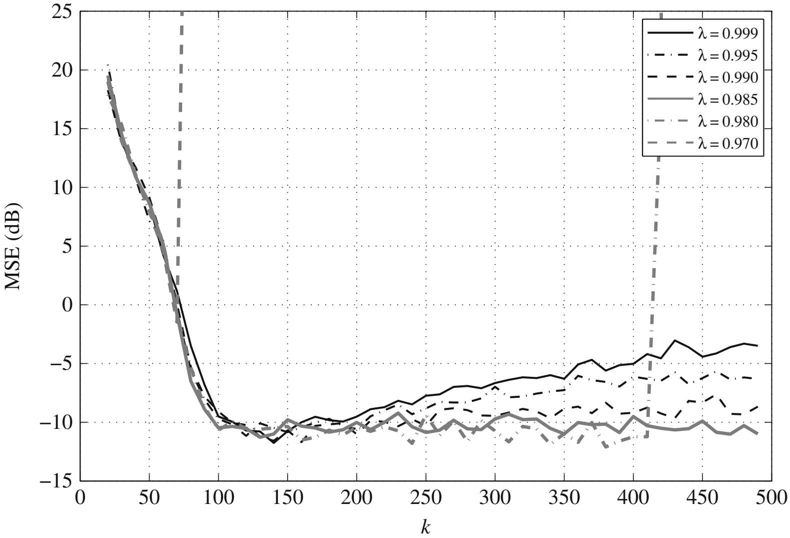


Fig. 2.14 Example 4: Resulting MSE for the RLS algorithm, with different values of the forgetting factor, illustrating how small values of λ may lead to algorithm divergence.

attempt to improve its performance in the non-stationary channel equalization of the last experiment, the RLS algorithm was employed with different values of forgetting factor. The associated learning curves for each value of λ within the set $\{0.999, 0.995, 0.99, 0.985, 0.98, 0.97\}$ are presented in Figure 2.14. From this figure, one can conclude that decreasing the value of λ improves the RLS performance (meaning that the algorithm can track better the channel variation) in this case. However, if one greatly reduces the value of λ, the RLS algorithm may diverge due to the time-varying nature of this example. For instance, when $\lambda = 0.98$, the adaptation process diverges a bit after 400 iterations, whereas the divergence occurs even sooner when $\lambda = 0.97$.

This example brings to light an important issue of an adaptive algorithm: its numerical stability. As seen above, the conventional RLS algorithm, besides its high computational complexity, also presents numerical stabilities problems even in double precision floating point arithmetic. This fact calls for one's attention to the need of a stable RLS version, as, for instance, the family of RLS algorithms based on the QR decomposition, which constitutes an elegant and efficient answer to the algorithm stability problem.

2.5 Conclusion

This chapter introduced the most basic concepts associated to adaptive filtering establishing the notation used throughout the book. It was verified how adaptive algorithms are employed to adjust the coefficients of a digital filter to achieve a desired time-varying performance in several practical situations. Emphasis was given on the description of several adaptation algorithms. In particular, the LMS and the NLMS algorithms were seen as iterative schemes for optimizing the ISE, an instantaneous approximation of the MSE objective function. Data-reuse algorithms introduced the concept of utilizing data from past time samples, resulting in a faster convergence of the adaptive process. Finally, the RLS family of algorithms, based on the WLS function, was seen as the epitome of fast adaptation algorithms, which use all available signal samples to perform the adaptation process. In general, RLS algorithms are used whenever fast convergence is necessary, for input signals with a high eigenvalue spread, and when the increase in the computational load is tolerable. A detailed discussion on the RLS family of algorithms based on the QR decomposition, which also guarantees good numerical properties in finite-precision implementations, constitutes the main goals of this book. Practical examples of adaptive system identification and channel equalization were presented, allowing one to visualize convergence properties, such as misadjustment, speed, and stability, of several distinct algorithms discussed previously.

References

1. B. Widrow and S. D. Stearns, Adaptive Signal Processing. Prentice-Hall, Englewood-Cliffs, NJ, USA (1985)
2. S. Haykin, Adaptive Filter Theory. 2nd edition Prentice-Hall, Englewood Cliffs, NJ, USA (1991)
3. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
4. J. A. Apolinário Jr., M. L. R. de Campos, and P. S. R. Diniz, Convergence analysis of the binormalized data-reusing LMS algorithm. IEEE Transactions on Signal Processing, vol. 48, no. 11, pp. 3235–3242 (November 2000)
5. D. T. Slock, On the convergence behavior of the LMS and the normalized LMS algorithms. IEEE Transactions on Signal Processing, vol. 41, no. 9, pp. 2811–2825 (September 1993)
6. P. S. R. Diniz, M. L. R. de Campos, and A. Antoniou, Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor. IEEE Transactions on Signal Processing, vol. 43, no. 3, pp. 617–627 (March 1995)
7. F. F. Yassa, Optimality in the choice of the convergence factor for gradient-based adaptive algorithms. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-35, no. 1, pp. 48–59 (January 1987)
8. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley & Sons, Hoboken, NJ, USA (2003)
9. S. Roy and J. J. Shynk, Analysis of the data-reusing LMS algorithm, 32nd Midwest Symposium on Circuits and Systems, MWSCAS'89, Urbana-Champaign, USA, pp. 1127–1130 (1989)
10. W. K. Jenkins, A. W. Hull, J. C. Strait, B. A. Schnaufer, and X. Li, Advanced Concepts in Adaptive Signal Processing. Kluwer Academic Publishers, Norwell, MA, USA (1996)
11. B. A. Schnaufer, *Practical Techniques for Rapid and Reliable Real-Time Adaptive Filtering*. Ph.D. thesis, Adviser: W. K. Jenkins, Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL, USA (1995)
12. G. C. Goodwin and S. K. Sin, Adaptive Filtering Prediction and Control. Prentice-Hall, Englewood-Cliffs, NJ, USA (1984)

13. K. Ozeki and T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. Transactions IECE Japan, vol. J67-A, no. 5, pp. 126–132 (1984)
14. S. L. Gay and S. Tavathia, The fast affine projection algorithm. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 3023–3036 (May 1995)
15. S. G. Sankaran and A. A. Beex, Convergence behavior of affine projection algorithms. IEEE Transactions on Signal Processing, vol. 48, no. 4, pp. 1086–1096 (April 2000)
16. M. L. R. de Campos, P. S. R. Diniz, and J. A. Apolinário Jr., On normalized data-reusing and affine-projections algorithms. IEEE International Conference on Electronics, Circuits, and Systems, ICECS'99, Pafos, Cyprus, pp. 843–846 (1999)
17. M. Montazeri and P. Duhamel, A set of algorithms linking NLMS and block RLS algorithms. IEEE Transactions on Signal Processing, vol. 43, no. 2, pp. 444–453 (February 1995)
18. J. M. Cioffi and T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32, no. 2, pp. 302–337 (April 1984)
19. D. L. Lee, M. Morf, and B. Friedlander, Recursive least squares ladder estimation algorithms. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-29, no. 3, pp. 627–641 (June 1981)
20. J. M. Cioffi, The fast Householder filters RLS adaptive filter. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'90, Albuquerque, USA, pp. 1619–1622 (1990)
21. K. J. R. Liu, S.-F. Hsieh, and K. Yao, Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation. IEEE Transactions on Signal Processing, vol. 40, no. 4, pp. 946–957 (April 1992)
22. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares adaptive filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. SP-41, no. 1, pp. 20–30 (January 1993)
23. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-38, no. 4, pp. 631–653 (April 1990)
24. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. SP-39, no. 4, pp. 879–891 (April 1991)
25. M. D. Miranda and M. Gerken, A hybrid QR-lattice least squares algorithm using a priori errors. 38th Midwest Symposium on Circuits and Systems, MWSCAS'95, Rio de Janeiro, Brazil, pp. 983–986 (August 1995)
26. A. A. Rontogiannis and S. Theodoridis, New fast inverse QR least squares adaptive algorithms. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 1412–1415 (May 1995)
27. J. A. Apolinário Jr. and P. S. R. Diniz, A new fast QR algorithm based on a priori errors. IEEE Signal Processing Letters, vol. 4, no. 11, pp. 307–309 (November 1997)
28. J. A. Apolinário Jr., M. G. Siqueira, and P. S. R. Diniz, Fast QR algorithms based on backward prediction errors: a new implementation and its finite precision performance. Birkhäuser Circuits Systems and Signal Processing, vol. 22, no. 4, pp. 335–349 (July/August 2003)
29. F. Ling, Givens rotation based least squares lattice and related algorithms. IEEE Transactions on Signal Processing, vol. SP-39, no. 7, pp. 1541–1551 (July 1991)
30. I. K. Proudler, J. G. McWhirter, and T. J. Shepard, Computationally efficient QR decomposition approach to least squares adaptive filtering. IEE Proceedings-F, vol. 138, no. 4, pp. 341–353 (August 1991)
31. F. Desbouvries and P. A. Regalia, A minimal, rotation-based FRLS lattice algorithm. IEEE Transactions on Signal Processing, vol. 45, no. 5, pp. 1371–1374 (May 1997)

Chapter 3
Conventional and Inverse QRD-RLS Algorithms

José A. Apolinário Jr. and Maria D. Miranda

Abstract This chapter deals with the basic concepts used in the recursive least-squares (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 provide details of their most important relations. The notation and variables used herein will be exactly the same used in the previous introductory chapter. For clarity, all derivations will be carried out using real variables and the final presentation of the algorithms (tables and pseudo-codes) will correspond to their complex-valued versions.

3.1 The Least-Squares Problem and the QR Decomposition

We start by introducing the weighted least-squares (WLS) filtering problem for the identification of a linear system [1]. To this end, we consider two sets of variables, $d(\ell)$ and $x(\ell)$, and the errors $\bar{e}(\ell)$, for $0 \leq \ell \leq k$. The set $d(\ell)$ is the response of an unknown system at time-instant ℓ when the input is the set of variables $x(\ell)$, with $x(\ell) = 0$ for $\ell < 0$. The error at time-instant ℓ is defined as $\bar{e}(\ell) = d(\ell) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(\ell)$, $\mathbf{w}(k)$ being the filter coefficient vector. These errors and the variables $d(\ell)$ and $x(\ell)$ are attenuated by the factor $\lambda^{(k-\ell)/2}$, $0 \ll \lambda < 1$.

José A. Apolinário Jr.
Military Institute of Engineering (IME), Rio de Janeiro – Brazil
e-mail: apolin@ieee.org

Maria D. Miranda
University of São Paulo (USP), São Paulo – Brazil
e-mail: maria@lcs.poli.usp.br

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_3,

Now, to facilitate the problem formulation, we collect the attenuated estimation errors into a $(k+1)\times 1$ vector, written as

$$\mathbf{e}(k) = \left[\bar{e}(k) \;\; \lambda^{1/2}\bar{e}(k-1) \;\cdots\; \lambda^{k/2}\bar{e}(0)\right]^{\mathrm{T}}. \tag{3.1}$$

This vector can take the form

$$\mathbf{e}(k) = \mathbf{d}(k) - \widehat{\mathbf{d}}(k), \tag{3.2}$$

where the desired response vector is given by

$$\mathbf{d}(k) = \left[d(k) \;\; \lambda^{1/2}d(k-1) \;\cdots\; \lambda^{k/2}d(0)\right]^{\mathrm{T}}, \tag{3.3}$$

and its estimate, obtained from linear weighted averages of observation sequences, is given as

$$\widehat{\mathbf{d}}(k) = \underbrace{\begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-N) \\ \lambda^{1/2}x(k-1) & \lambda^{1/2}x(k-2) & & \vdots \\ & & & \lambda^{(k-N)/2}x(0) \\ \vdots & \vdots & & 0 \\ \lambda^{(k-1)/2}x(1) & \lambda^{(k-1)/2}x(0) & & \vdots \\ \lambda^{k/2}x(0) & 0 & \cdots & 0 \end{bmatrix}}_{\mathbf{X}(k)} \underbrace{\begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix}}_{\mathbf{w}(k)}. \tag{3.4}$$

The input data matrix, denoted here as $\mathbf{X}(k)$, has dimension $(k+1)\times(N+1)$ and can be represented in terms of its $N+1$ columns or its $k+1$ rows, that is,

$$\mathbf{X}(k) = \left[\mathbf{x}^{(0)}(k)\,\mathbf{x}^{(1)}(k)\cdots\mathbf{x}^{(N)}(k)\right] = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}^{\mathrm{T}}(0) \end{bmatrix}. \tag{3.5}$$

Note that $\mathbf{x}(\ell) = [x(\ell)\; x(\ell-1) \;\cdots\; x(\ell-N)]^{\mathrm{T}}$ represents the input regression vector at instant ℓ, $0 \le \ell \le k$, with $N+1$ elements, and $\mathbf{x}^{(i)}(k)$ for $(i = 0,\cdots,N)$ represents the $(i+1)$th column of $\mathbf{X}(k)$.

The Euclidean norm of the weighted estimation error vector corresponds to the deterministic cost function $\xi_D(k)$, the same one used for the recursive least-squares (RLS) algorithm, which is given by

$$\xi_D(k) = \|\mathbf{e}(k)\|^2 = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k) = \sum_{\ell=0}^{k} \lambda^{k-\ell}\bar{e}^2(\ell). \tag{3.6}$$

The WLS filtering problem consists in determining, at instant k, the coefficient vector $\mathbf{w}(k)$ that minimizes $\xi_D(k)$. The optimal solution is obtained when the coefficient vector satisfies [1, 2]

$$\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k), \tag{3.7}$$

where

$$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) \tag{3.8}$$

is the $(N+1)\times(N+1)$ input-data deterministic autocorrelation matrix, and

$$\mathbf{p}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) \tag{3.9}$$

is the $(N+1)\times 1$ deterministic cross-correlation vector.

The deterministic autocorrelation matrix $\mathbf{R}(k)$ is considered non-singular; nevertheless, the inverse of $\mathbf{R}(k)$, used in (3.7), can become ill-conditioned, e.g., due to loss of persistence of excitation of the input signal or due to quantization effects [1, 2].

The WLS problem may be solved by means of a QR decomposition which is numerically well-conditioned [3, 4]. This approach is based on the following triangularization of the input data matrix:

$$\mathbf{Q}(k)\mathbf{X}(k) = \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \\ \mathbf{U}(k) \end{bmatrix}. \tag{3.10}$$

Matrix $\mathbf{U}(k)$ has dimension $(N+1)\times(N+1)$ and a triangular structure as, for example, shown in Figure 3.1. $\mathbf{Q}(k)$ is an unitary matrix with dimension $(k+1)\times(k+1)$ which represents the overall triangularization process. The unicity of the decomposition is yielded through the condition that all elements of the main anti-diagonal of $\mathbf{U}(k)$ are non-negative [4].

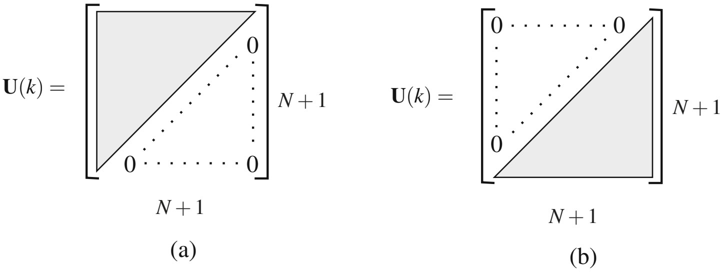


Fig. 3.1 The different triangularizations of $\mathbf{U}(k)$: (a) UPPER and (b) LOWER.

As $\mathbf{X}(k)$ is a $(k+1)\times(N+1)$ matrix, it is interesting to consider the following partition of matrix $\mathbf{Q}(k)$:

$$\mathbf{Q}(k) = \begin{bmatrix} \mathbf{Q}_1(k) \\ \\ \mathbf{Q}_2(k) \end{bmatrix}, \tag{3.11}$$

with $\mathbf{Q}_1(k)$ and $\mathbf{Q}_2(k)$ having dimensions $(k-N)\times(k+1)$ and $(N+1)\times(k+1)$, respectively. Since matrix $\mathbf{Q}(k)$ is unitary, i.e.,

$$\mathbf{Q}^{\mathrm{T}}(k)\mathbf{Q}(k) = \mathbf{Q}(k)\mathbf{Q}^{\mathrm{T}}(k) = \mathbf{I}_{k+1}, \tag{3.12}$$

it follows that

$$\begin{aligned} &\mathbf{Q}_1^{\mathrm{T}}(k)\mathbf{Q}_1(k) + \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k) = \mathbf{I}_{k+1}, \\ &\mathbf{Q}_1(k)\mathbf{Q}_1^{\mathrm{T}}(k) = \mathbf{I}_{k-N}, \text{ and} \\ &\mathbf{Q}_2(k)\mathbf{Q}_2^{\mathrm{T}}(k) = \mathbf{I}_{N+1}. \end{aligned} \tag{3.13}$$

The pre-multiplication of (3.10) by $\mathbf{Q}^{\mathrm{T}}(k)$, with (3.12) and (3.11) yields

$$\mathbf{X}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{U}(k). \tag{3.14}$$

Therefore, the deterministic autocorrelation matrix satisfies

$$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k), \tag{3.15}$$

and matrix $\mathbf{U}(k)$ is referred to as the Cholesky factor of $\mathbf{R}(k)$ [1, 3].

The pre-multiplication of (3.2) by $\mathbf{Q}(k)$, i.e.,

$$\mathbf{e}_q(k) = \mathbf{Q}(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q_1}(k) \\ \mathbf{e}_{q_2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{O} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k), \tag{3.16}$$

triangularizes $\mathbf{X}(k)$ and, being $\mathbf{Q}(k)$ a unitary matrix, it will not affect the cost function, i.e., $\|\mathbf{e}_q(k)\|^2 = \|\mathbf{e}(k)\|^2$. 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 when, by a proper choice of $\mathbf{w}(k)$, the term $\mathbf{d}_{q_2}(k) - \mathbf{U}(k)\mathbf{w}(k)$ becomes zero. The tap-weight coefficients can then be computed as

$$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k). \tag{3.17}$$

Note that (3.7) and (3.17) represent different ways to find the same WLS solution. Next, with geometric and linear algebraic arguments, we show how these two representations are related.

When the coefficient vector $\mathbf{w}(k)$ satisfies (3.7), it is easy to figure out that the estimated desired response $\widehat{\mathbf{d}}(k) = \mathbf{X}(k)\mathbf{w}(k)$ and the estimation error vector

$\mathbf{e}(k) = \mathbf{d}(k) - \widehat{\mathbf{d}}(k)$ can be expressed as

$$\widehat{\mathbf{d}}(k) = \mathbf{X}(k)\underbrace{\mathbf{R}^{-1}(k)\mathbf{p}(k)}_{\mathbf{w}(k)} = \mathscr{P}(k)\mathbf{d}(k) \tag{3.18}$$

and

$$\mathbf{e}(k) = [\mathbf{I}_{k+1} - \mathscr{P}(k)]\,\mathbf{d}(k), \tag{3.19}$$

where $\mathscr{P}(k) = \mathbf{X}(k)\mathbf{R}^{-1}(k)\mathbf{X}^{\mathrm{T}}(k)$.

Matrix $\mathscr{P}(k)$ is a projection operator, responsible for projecting the desired response vector in the space spanned by the data matrix columns of $\mathbf{X}(k)$. This projection is orthogonal to the estimation error (orthogonality principle). In this condition, $\mathbf{e}(k)$ is known as the optimum estimation error.

When the norm of $\mathbf{e}(k)$ is minimum, the coefficient vector $\mathbf{w}(k)$ also satisfies (3.17). Then, vector $\widehat{\mathbf{d}}(k) = \mathbf{X}(k)\mathbf{w}(k)$ can also be rewritten using (3.17) and (3.14) as

$$\widehat{\mathbf{d}}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{d}_{q_2}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k)\mathbf{d}(k), \tag{3.20}$$

and the optimum estimation error vector is given as

$$\mathbf{e}(k) = \left[\mathbf{I}_{k+1} - \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k)\right]\mathbf{d}(k) = \mathbf{Q}_1^{\mathrm{T}}(k)\mathbf{Q}_1(k)\mathbf{d}(k). \tag{3.21}$$

Therefore, the projection operator can be defined by matrix $\mathbf{Q}_2(k)$ as

$$\mathscr{P}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k), \tag{3.22}$$

and the complementary projection operator by matrix $\mathbf{Q}_1(k)$ as

$$\mathscr{P}^{\perp}(k) = \mathbf{I}_{k+1} - \mathscr{P}(k) = \mathbf{Q}_1^{\mathrm{T}}(k)\mathbf{Q}_1(k). \tag{3.23}$$

From the mathematical relationships presented so far, we can observe that matrix $\mathbf{Q}(k)$ rotates the subspace spanned by the columns of matrix $\mathbf{X}(k)$ and by its optimum estimation error $\mathbf{e}(k)$, which is orthogonal to the columns of $\mathbf{X}(k)$. In this case, matrix $\mathbf{Q}(k)$ forces the subspace spanned by the columns of matrix $\mathbf{X}(k)$, of dimension $(N+1)$, to coincide with the subspace spanned by the last $(N+1)$ vectors of the Euclidean space basis, of dimension $(k+1)\times(k+1)$, used in the representation. Hence, the subspace of dimension $(k-N)$, where the optimum estimation error $\mathbf{e}(k)$ lies and which is orthogonal to the space spanned by the columns of $\mathbf{X}(k)$, coincides with the subspace spanned by the other $(k-N)$ vectors from the Euclidean basis. Figure 3.2 illustrates these interpretations. Such transformation affects the input signal, the desired signal, and the projection operator, without modifying the autocorrelation matrix, the cross-correlation vector, and the optimum coefficient vector.

Table 3.1 presents the relationship between the conventional LS (or WLS) method and its QR decomposition counterpart. In the second column, the representations of the main results and the operators in the rotated domain are presented. The

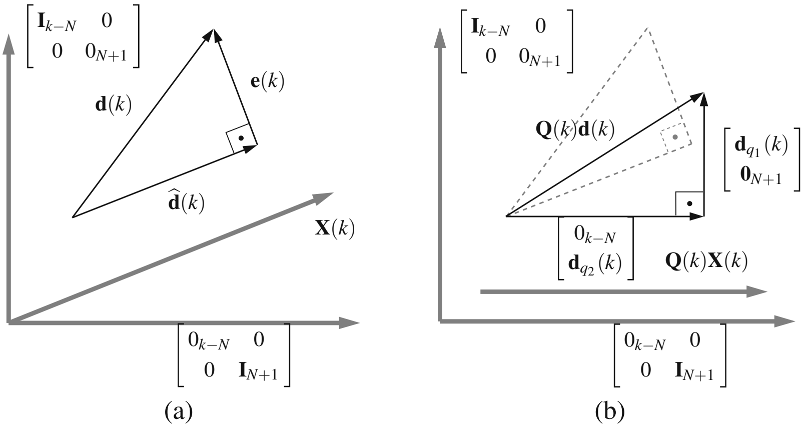

$\begin{bmatrix} \mathbf{I}_{k-N} & 0 \\ 0 & 0_{N+1} \end{bmatrix}$
$\mathbf{d}(k)$
$\mathbf{e}(k)$
$\widehat{\mathbf{d}}(k)$
$\mathbf{X}(k)$
$\begin{bmatrix} 0_{k-N} & 0 \\ 0 & \mathbf{I}_{N+1} \end{bmatrix}$
(a)
$\begin{bmatrix} \mathbf{I}_{k-N} & 0 \\ 0 & 0_{N+1} \end{bmatrix}$
$\mathbf{Q}(k)\mathbf{d}(k)$
$\begin{bmatrix} \mathbf{d}_{q_1}(k) \\ \mathbf{0}_{N+1} \end{bmatrix}$
$\begin{bmatrix} 0_{k-N} \\ \mathbf{d}_{q_2}(k) \end{bmatrix}$
$\mathbf{Q}(k)\mathbf{X}(k)$
$\begin{bmatrix} 0_{k-N} & 0 \\ 0 & \mathbf{I}_{N+1} \end{bmatrix}$
(b)

Fig. 3.2 Spacial visualization of the rotated vectors. (a) Estimation of $\mathbf{d}(k)$ projected onto the subspace spanned by the columns of $\mathbf{X}(k)$. (b) Estimation of $\mathbf{d}_q(k)$ projected onto the Euclidean basis space.

last three rows indicate how to calculate the results, which values will not change due to the transformation, using whether the original or the rotated variables. It is worth mentioning that, in the domain of the signals rotated by the matrix $\mathbf{Q}(k)$, the projection operator and its complement assume very simple forms.

Table 3.1 Relationships between methods LS and QRD-LS.

Method	LS	QRD-LS
Data matrix	$\mathbf{X}(k)$	$\begin{bmatrix} \mathbf{O} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{X}(k)$
Desired response vector	$\mathbf{d}(k)$	$\begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}(k)\mathbf{d}(k)$
Projection operator	$\mathbf{X}(k)\mathbf{R}^{-1}(k)\mathbf{X}^{\mathrm{T}}(k)$	$\begin{bmatrix} \mathbf{O}_{k-N} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{N+1} \end{bmatrix}$
Estimated response	$\mathscr{P}(k)\mathbf{d}(k)$	$\begin{bmatrix} 0 \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{O} \\ \mathbf{Q}_2(k) \end{bmatrix} \mathbf{d}(k)$
Complementary projection operator	$\mathscr{P}^{\perp}(k)$	$\begin{bmatrix} \mathbf{I}_{k-N} & \mathbf{O} \\ \mathbf{O} & \mathbf{O}_{N+1} \end{bmatrix}$
Estimation error	$\mathscr{P}^{\perp}(k)\mathbf{d}(k)$	$\begin{bmatrix} \mathbf{d}_{q1}(k) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_1(k) \\ 0 \end{bmatrix} \mathbf{d}(k)$
Autocorrelation matrix	$\mathbf{R}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k)$	
Cross-correlation vector	$\mathbf{p}(k) = \mathbf{X}^{\mathrm{T}}(k)\mathbf{d}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{d}_{q2}(k)$	
Optimum coefficients vector	$\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q2}(k)$	

The solution for the least-squares problem using the QR decomposition consists basically in the data matrix decomposition (3.10) and, depending on the particular method employed, in executing some of the calculation indicated in (3.20), (3.17), and (3.21). Obviously, the procedure is numerically complex due to the data matrix QR decomposition. But, in practical adaptive filtering applications, the data matrix can be performed recursively. Also, if $\mathbf{U}(k)$ and $\mathbf{d}_{q2}(k)$ are available, the optimum vector $\mathbf{w}(k)$ can be computed with a reduced computational complexity. This is due to the triangular nature of matrix $\mathbf{U}(k)$ and the possibility to use the so-called back-substitution procedure (if assuming a lower triangular matrix). In Appendix 1, we present this procedure as well as the forward substitution procedure, its counterpart for the case of an upper triangular matrix $\mathbf{U}(k)$. It is important to note that, apart from the reduction in the computational burden, if the main diagonal elements are non-zero, the existence of the inverse of $\mathbf{U}(k)$ is ensured [3, 4].

3.2 The Givens Rotation Method

The orthogonal triangularization process may be carried out using various techniques such as Gram–Schmidt orthogonalization, Householder transformation, or Givens rotations. Particularly, Givens rotations leads to an efficient algorithm whereby the triangularization process is updated recursively. As we are interested in the iterative least-squares solution, we consider a triangularization procedure carried out through Givens rotations.

In order to introduce the Givens rotation method, we consider vectors $\mathbf{v}$ and $\mathbf{v}'$ as shown in Figure 3.3. If we represent vector $\mathbf{v}$ as

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix}, \tag{3.24}$$

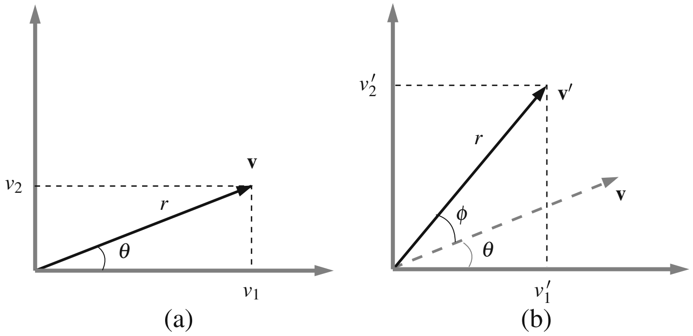


Fig. 3.3 (a) Vector $\mathbf{v}$ projected onto the Euclidean space; (b) Vector $\mathbf{v}$ rotated by ϕ and projected onto the Euclidean space.

vector $\mathbf{v}'$, obtained from a plane rotation of ϕ degrees counterclockwise, can be written as

$$\mathbf{v}' = \begin{bmatrix} v_1^{'} \\ v_2^{'} \end{bmatrix} = \begin{bmatrix} r\cos(\theta+\phi) \\ r\sin(\theta+\phi) \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{3.25}$$

For convenience, we rewrite $\mathbf{v}' = G\mathbf{v}$, where

$$G = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{3.26}$$

is the elementary Givens rotation matrix.

When the plane rotation is clockwise, $\mathbf{v}'$ is given as

$$\mathbf{v}' = \begin{bmatrix} r\cos(\theta-\phi) \\ r\sin(\theta-\phi) \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{3.27}$$

In this case, the elementary Givens rotation matrix takes the form

$$G = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}. \tag{3.28}$$

In both cases (counterclockwise and clockwise), the rotation matrix is unitary, that is, $G^TG = GG^T = I$, and therefore it preserves the norm r of vector $\mathbf{v}$. We consider the counterclockwise rotation throughout this chapter.

If $\phi + \theta = \pi/2$, then $v_1^{'} = 0$, $v_2^{'} = r$ and

$$\begin{bmatrix} 0 \\ r \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{3.29}$$

The above development suggests that it is possible to annihilate an element in a two-dimensional vector by using plane rotation. From the relation in (3.29) and noting that $\sin\phi$ and $\cos\phi$ are always constrained by the trigonometric relation

$$\sin^2\phi + \cos^2\phi = 1, \tag{3.30}$$

we have: $\cos\phi = v_2/r$, $\sin\phi = v_1/r$, and $r = \sqrt{v_1^2 + v_2^2}$.

The annihilation using the Givens rotations, as seen above, can be extended to annihilate a specific element in a vector of $N+1$ elements, or a sequence of elements in a vector or in a matrix [5]. Various choices of rotation orders can be used to solve the problem.

In order to illustrate, for a 4×3 matrix formed by a first row and a lower triangular matrix, the rotation angles to annihilate all elements of the first row, we consider

$$\mathbf{A} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix}. \tag{3.31}$$

In this example, we have to find a transformation matrix $\mathbf{Q}_\theta(k)$ such that $\mathbf{B} = \mathbf{Q}_\theta(k)\mathbf{A}$. Matrix $\mathbf{Q}_\theta(k)$ must annihilate all elements of the first row of $\mathbf{A}$ from left to right. Three Givens rotations are shown in the following steps.

$$\textbf{Step 1:} \begin{bmatrix} \cos\theta_0 & 0 & 0 & -\sin\theta_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta_0 & 0 & 0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}_{1,2} & \bar{x}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} \tag{3.32}$$

$$\textbf{Step 2:} \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \bar{x}_{1,2} & \bar{x}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & v_{2,2} & v_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \bar{\bar{x}}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} \tag{3.33}$$

$$\textbf{Step 3:} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & \bar{\bar{x}}_{1,3} \\ 0 & 0 & v_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix}. \tag{3.34}$$

This completes the lower triangularization of matrix $\mathbf{A}$.

At this point, it is important to note that the transformation matrix $\mathbf{Q}_\theta(k)$ can be written as a product of Givens rotation matrices given by

$$\mathbf{Q}_\theta(k) = \mathbf{Q}_{\theta_N}(k) \cdots \mathbf{Q}_{\theta_1}(k)\,\mathbf{Q}_{\theta_0}(k) = \prod_{i=0}^{N} \mathbf{Q}_{\theta_i}(k). \tag{3.35}$$

If matrix $\mathbf{U}(k)$ is triangularized as depicted in Figure 3.1, the corresponding $\mathbf{Q}_{\theta_i}(k)$, for $0 \le I \le N$, are given as follows:

$$UPPER: \ \mathbf{Q}_{\theta_i}(k) = \begin{bmatrix} \cos\theta_i(k) & \mathbf{0}^{\mathrm{T}} & -\sin\theta_i(k) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I}_i & \mathbf{0} & \mathbf{0}\cdots\mathbf{0} \\ \sin\theta_i(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta_i(k) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0}\cdots\mathbf{0} & \mathbf{0} & \mathbf{I}_{N-i} \end{bmatrix} \tag{3.36}$$

$$LOWER: \ \mathbf{Q}_{\theta_i}(k) = \begin{bmatrix} \cos\theta_i(k) & \mathbf{0}^{\mathrm{T}} & -\sin\theta_i(k) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I}_{N-i} & \mathbf{0} & \mathbf{0}\cdots\mathbf{0} \\ \sin\theta_i(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta_i(k) & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0}\cdots\mathbf{0} & \mathbf{0} & \mathbf{I}_i \end{bmatrix}. \tag{3.37}$$

The row and column of each $\cos\theta_i$ and $\sin\theta_i$ are related with the element that we want to annihilate. Only the elements of matrix $\mathbf{A}$ at a position related to the row and column of $\cos\theta_i$ and $\sin\theta_i$, respectively, are affected by the transformation imposed by $\mathbf{Q}_{\theta_i}$. The angles $\theta_i(k)$ in (3.36) and (3.37) are not the same although written, for the sake of simplicity, using the same notation.

3.3 The Conventional QRD-RLS Algorithm

One of the first works that used QR decomposition to solve the RLS problem was proposed by Gentleman [6]. He used a triangular array in order to avoid matrix inversion and proposed a pipelined sequence of Givens rotations to perform the back-substitution process required to solve the associated system of equations. From [6], the conventional QRD-RLS algorithm, as we know it today, was conceived by McWhirter [7]. He was the first to describe a systolic array, using a pipelined sequence of Givens rotations, performing an orthogonal triangularization of the input data matrix and generating the estimated error without having to resort to back-substitution. This section presents the most basic equations of the QRD-RLS algorithms.

Let us start by using the fact that the data matrix in (3.5) can be rewritten as

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{X}(k-1) \end{bmatrix} \tag{3.38}$$

and also that $\mathbf{Q}(k-1)$ is unitary. Thus, the product $\mathbf{Q}(k)\mathbf{X}(k)$ can be written as

$$\mathbf{Q}(k)\underbrace{\begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ 0 & \mathbf{Q}^{\mathrm{T}}(k-1) \end{bmatrix}\begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ 0 & \mathbf{Q}(k-1) \end{bmatrix}}_{\mathbf{I}}\underbrace{\begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{X}(k-1) \end{bmatrix}}_{\mathbf{X}(k)} = \begin{bmatrix} \mathbf{O} \\ \mathbf{U}(k) \end{bmatrix}. \tag{3.39}$$

In the above equation, if we denote the product of the first two matrices on the left side by $\tilde{\mathbf{Q}}(k)$ and compute the multiplication of the remaining two matrices, we obtain

$$\tilde{\mathbf{Q}}(k)\begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \mathbf{0}_{(k-N-1)\times(N+1)} \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{(k-N)\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}. \tag{3.40}$$

From (3.40), we see that $\tilde{\mathbf{Q}}(k)$ is a product of Givens rotations matrices that annihilates the current input vector. Moreover, the recursive nature of $\mathbf{Q}(k)$ may be expressed by

$$\mathbf{Q}(k) = \tilde{\mathbf{Q}}(k)\begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ 0 & \mathbf{Q}(k-1) \end{bmatrix}. \tag{3.41}$$

Once $\tilde{\mathbf{Q}}(k)$ is responsible for zeroing $\mathbf{x}^{\mathrm{T}}(k)$, as shown in (3.40), its structure includes a sub-matrix $\mathbf{I}_{k-N-1}$, such that it can be represented as

$$\tilde{\mathbf{Q}}(k) = \left[\begin{array}{ccccccc} * & 0 & \cdots & 0 & * & \cdots & * \\ 0 & & & & 0 & & \\ \vdots & & \mathbf{I}_{k-N-1} & & \vdots & & \\ 0 & & & & 0 & & \\ * & 0 & \cdots & 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ * & 0 & \cdots & 0 & * & \cdots & * \end{array}\right], \tag{3.42}$$

with the asterisks ($*$) corresponding to the non-zero elements. Although $\mathbf{Q}(k)$ is $(k+1)\times(k+1)$, we can avoid the ever-increasing order characteristic rewriting (3.40) as

$$\begin{bmatrix} 0^{\mathrm{T}} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix}, \tag{3.43}$$

where $\mathbf{Q}_{\theta}(k)$ is $(N+2)\times(N+2)$ and corresponds to $\tilde{\mathbf{Q}}(k)$ after removing the $\mathbf{I}_{k-N-1}$ section along with the corresponding rows and columns.

Recalling (3.16), we see that $\mathbf{e}_{q_1}(k) = \mathbf{d}_{q_1}(k)$; moreover, (3.41) shows that the product $\mathbf{Q}(k)\mathbf{d}(k)$ can be written as

$$\begin{aligned} \mathbf{Q}(k)\mathbf{d}(k) = \begin{bmatrix} \mathbf{e}_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} &= \underbrace{\tilde{\mathbf{Q}}(k) \begin{bmatrix} 1 & 0^{\mathrm{T}} \\ 0 & \mathbf{Q}(k-1) \end{bmatrix}}_{\mathbf{Q}(k)} \underbrace{\begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}(k-1) \end{bmatrix}}_{\mathbf{d}(k)} \\ &= \tilde{\mathbf{Q}}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{e}_{q_1}(k-1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} e_{q_1}(k) \\ \lambda^{1/2}\mathbf{e}_{q_1}(k-1) \\ \mathbf{d}_{q_2}(k) \end{bmatrix}, \end{aligned} \tag{3.44}$$

where the last multiplication can be easily understood if we note in (3.40) and in (3.42) that $\tilde{\mathbf{Q}}(k)$ alter only the first and the last $N+1$ components of a $(k+1)\times 1$ vector.

From (3.44), it is also possible to remove the increasing section of $\tilde{\mathbf{Q}}(k)$, resulting in the following expression:

$$\begin{bmatrix} e_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix}, \tag{3.45}$$

where $e_{q_1}(k)$ is the first element of the *rotated* error vector $\mathbf{e}_q(k)$ and $\mathbf{d}_{q_2}(k)$ is a vector with the last $N+1$ elements of the *rotated* desired signal vector. The rotated error $e_{q_1}(k)$, with the help of a conversion factor, allows the computation of the prediction error—the difference between $d(k)$ and its prediction from $\mathbf{x}(k)$—without computing the estimate of the desired response. This process is usually termed *joint-process estimation*.

It is important to note that $\mathbf{Q}_\theta(k)$, being responsible for the update of $\mathbf{U}(k)$ as in (3.43), is formed by a product of Givens rotation matrices as shown in (3.35) and its structure will depend on the type of triangularization of $\mathbf{U}(k)$, i.e., whether it is an **upper** or a **lower** triangular matrix. This choice determines two classes of *fast* (reduced complexity) algorithms, as will be seen in Chapter 4. From the example of the previous section, it is possible to obtain, using (3.35), the structure of $\mathbf{Q}_\theta(k)$ for upper and lower triangularization of $\mathbf{U}(k)$; the results, for $N = 2$, are given by the following two matrices:

$$UPPER: \;\; \mathbf{Q}_\theta(k) = \begin{bmatrix} c\theta_2 c\theta_1 c\theta_0 & -c\theta_2 c\theta_1 s\theta_0 & -c\theta_2 s\theta_1 & -s\theta_2 \\ s\theta_0 & c\theta_0 & 0 & 0 \\ s\theta_1 c\theta_0 & -s\theta_1 s\theta_0 & c\theta_1 & 0 \\ s\theta_2 c\theta_1 c\theta_0 & -s\theta_2 c\theta_1 s\theta_0 & -s\theta_2 s\theta_1 & c\theta_2 \end{bmatrix}, \tag{3.46}$$

and

$$LOWER: \;\; \mathbf{Q}_\theta(k) = \begin{bmatrix} c\theta_2 c\theta_1 c\theta_0 & -s\theta_2 & -c\theta_2 s\theta_1 & -c\theta_2 c\theta_1 s\theta_0 \\ s\theta_2 c\theta_1 c\theta_0 & c\theta_2 & -s\theta_2 s\theta_1 & -s\theta_2 c\theta_1 s\theta_0 \\ s\theta_1 c\theta_0 & 0 & c\theta_1 & -s\theta_1 s\theta_0 \\ s\theta_0 & 0 & 0 & c\theta_0 \end{bmatrix}, \tag{3.47}$$

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

The structure of matrix $\mathbf{Q}_\theta(k)$ suggests, in both cases, that it can be partitioned as

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}, \tag{3.48}$$

where the structures of vectors $\mathbf{f}(k)$, $\mathbf{g}(k)$, and matrix $\mathbf{E}(k)$ depend on the type of triangularization of the data matrix. On the other hand, for both types of triangularization, the scalar $\gamma(k)$ denotes the product of successive cosine terms, i.e.,

$$\gamma(k) = \prod_{i=0}^{N} \cos\theta_i(k). \tag{3.49}$$

It was shown in [1], and will be detailed later in Section 3.5, that $\gamma(k)$ represents the squared root of the conversion factor between the *a priori* and *a posteriori* output errors for the degree $N+1$ filtering problem.

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

$$\mathbf{e}_q^{\mathrm{T}}(k)\mathbf{Q}(k)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} = \mathbf{e}^{\mathrm{T}}(k)\mathbf{Q}^{\mathrm{T}}(k)\mathbf{Q}(k)\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix} = \bar{e}(k), \tag{3.50}$$

where $\bar{e}(k)$ is the first element of the error vector defined in (3.1) and represents the *a posteriori* estimation error $\varepsilon(k)$, defined in Chapter 2 as

$$\varepsilon(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k) = \bar{e}(k). \tag{3.51}$$

From Equations (3.41) and (3.48) and the fact that $\mathbf{Q}_\theta(k)$ is $\tilde{\mathbf{Q}}(k)$ after removing the $k-N-1$ increasing columns and rows, we can conclude that $\mathbf{Q}(k)[1\ 0\ \cdots\ 0]^{\mathrm{T}} = [\gamma(k)\ 0\ \cdots\ 0\ \mathbf{f}^{\mathrm{T}}(k)]^{\mathrm{T}}$. Once $\mathbf{e}_{q_2}(k)$ is a null vector (keep in mind that $\mathbf{w}(k)$ in (3.16) was chosen in order to make it zero), it is possible to see that

$$\varepsilon(k) = e_{q_1}(k)\gamma(k). \tag{3.52}$$

This equation was first noted by McWhirter in [7] and is particularly useful in applications where the coefficients of the adaptive filter are not explicitly necessary since we can obtain $\varepsilon(k)$, the *a posteriori* output error, without computing $\mathbf{w}(k)$. If, however, the coefficient vector $\mathbf{w}(k)$ is needed, it can be computed by solving $\mathbf{U}(k)\mathbf{w}(k) = \mathbf{d}_{q2}(k)$. In this case, due to the fact that the Cholesky matrix $\mathbf{U}(k)$ is triangular, a matrix inversion can be avoided, e.g., with a *forward* or a *back-substitution* procedure (see Appendix 1 for further details).

The equations of the conventional QR algorithm (complex version) are presented in Table 3.2 where the type of triangularization used has no influence on the performance of the conventional QRD-RLS algorithm. A pseudo-code of a lower triangularization implementation is available in Appendix 3 (Table 3.5).

In the next section, we provide hints about the initialization of the triangularization procedure. Appendix 2 presents ways of avoiding the square-root operation in

Table 3.2 The conventional QRD-RLS equations.

```
QRD-RLS
for each $k$
{ Obtaining $\mathbf{Q}_\theta(k)$ and updating $\mathbf{U}(k)$:
  $\begin{bmatrix}\mathbf{0}^{\mathrm{T}}\\ \mathbf{U}(k)\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}\mathbf{x}^{\mathrm{H}}(k)\\ \lambda^{1/2}\mathbf{U}(k-1)\end{bmatrix}$
  Obtaining $\gamma(k)$:
  $\gamma(k) = \prod_{i=0}^{N}\cos\theta_i(k)$
  Obtaining $e_{q_1}(k)$ and updating $\mathbf{d}_{q_2}(k)$:
  $\begin{bmatrix}e_{q_1}(k)\\ \mathbf{d}_{q_2}(k)\end{bmatrix} = \mathbf{Q}_\theta(k)\begin{bmatrix}d^*(k)\\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1)\end{bmatrix}$
  Obtaining $\varepsilon(k)$:
  $\varepsilon(k) = e^*_{q_1}(k)\gamma(k)$  % a posteriori error: $d(k) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k)$
}
```

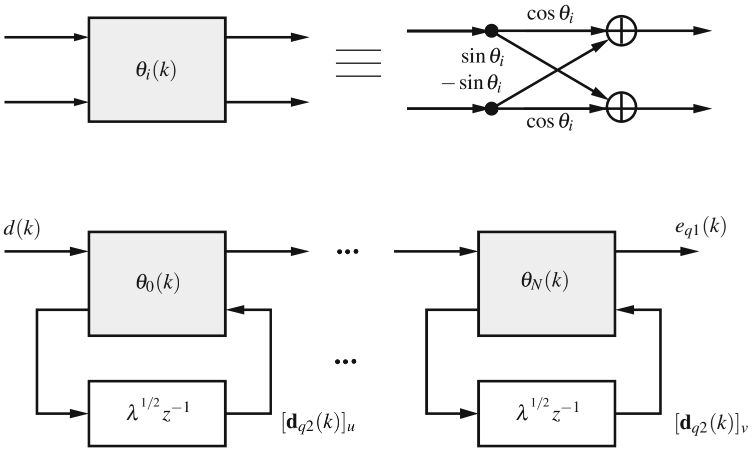


Fig. 3.4 McWhirter structure: Equation (3.45) implemented as a cascade of first-order orthogonal filters.

QRD-RLS algorithms. Now, to close this section, it is worth noting that, since $\mathbf{Q}_\theta(k)$ is a product of Givens rotation matrices, the equation system in (3.45) becomes the cascade of first-order orthogonal filters. Figure 3.4 depicts the operation carried out in (3.45). Due to the work of McWhirter [7], each orthogonal filter in [8] was named a McWhirter structure. In this figure, when we use an upper triangular Cholesky factor, we make $u = 0$ and $v = N$; this means that we update the elements of vector $\mathbf{d}_{q2}(k)$ from the first to the last one. Otherwise, when a lower triangular matrix is used, the updating is from the last element ($u = N$) to the first one ($v = 0$).

3.4 Initialization of the Triangularization Procedure

In order to run the QRD-RLS algorithm at time-instant $k = 0$, we need vector $\mathbf{d}_{q_2}(-1)$ and matrix $\mathbf{U}(-1)$. Assuming pre-windowing, a natural choice would be $\mathbf{d}_{q_2}(-1) = \mathbf{0}_{N+1}$ and $\mathbf{U}(-1) = \mathbf{0}_{(N+1)\times(N+1)}$. In that case, the choice $\mathbf{U}(-1) = \mathbf{0}_{(N+1)\times(N+1)}$ would lead to a non-singular matrix. In order to solve this problem, two strategies are possible: the *exact initialization* and the *soft-start*, as in the RLS algorithm [1, 9].

The **exact initialization** procedure comprises a period of $N+1$ samples, from $k = 0$ to N, during which the estimation error is zero. At $k = N+1$, the initialization period is completed and the estimation error may assume a value different than zero by executing the steps of the algorithm in Table 3.2.

If we are interested in an upper triangularization procedure, since $\mathbf{X}(N)$ is already upper triangular, nothing needs to be done. The exact initialization in this case is carried out as detailed in [2, Chapter 9] which corresponds to the exact initialization of the RLS algorithm, using back-substitution to obtain the coefficient vector. At $k = N+1$, when $x(N+1)$ is available, the matrix is no longer triangular and $N+1$ Givens rotations are necessary to annihilate all elements of the first row and the QRD-RLS equations start to be used with $\mathbf{U}(N) = \mathbf{X}(N)$.

If we are working with a lower triangular Cholesky factor, then the data matrix needs to be transformed before $k = N+1$. The complete transformation can be carried out with $N+1$ Givens rotations at $k = N$. In this case, we begin by annihilating the elements of column one until column N. For each column i, only the elements of row $j = 1, \ldots (N-i+1)$ are annihilated. The exact initialization procedure of a 3×3 lower triangular Cholesky matrix, that is, order $N = 2$, for $k = 2$, is described below.

$$\begin{bmatrix} c_0 & 0 & -s_0 \\ 0 & 1 & 0 \\ s_0 & 0 & c_0 \end{bmatrix} \begin{bmatrix} x(2) & x(1) & x(0) \\ \lambda^{1/2}x(1) & \lambda^{1/2}x(0) & 0 \\ \lambda\, x(0) & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ \lambda^{1/2}x(1) & \lambda^{1/2}x(0) & 0 \\ \bar{u}(3,1) & \bar{u}(3,2) & \bar{u}(3,3) \end{bmatrix} \tag{3.53}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ \lambda^{1/2}x(1) & \lambda^{1/2}x(0) & 0 \\ \bar{u}(3,1) & \bar{u}(3,2) & \bar{u}(3,3) \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ 0 & \bar{u}(2,2) & \bar{u}(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix} \tag{3.54}$$

$$\begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \bar{x}(1) & \bar{x}(0) \\ 0 & \bar{u}(2,2) & \bar{u}(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & u(1,3) \\ 0 & u(2,2) & u(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix} \tag{3.55}$$

As vector $\mathbf{e}(N) = \mathbf{d}(N) - \mathbf{X}(N)\mathbf{w}(N)$ is $(N+1) \times 1$, then $\mathbf{d}_{q_2}(N) = \mathbf{Q}(N)\mathbf{d}(N)$ and

$$\mathbf{dq}(2) = \underbrace{\begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \begin{bmatrix} c_0 & 0 & -s_0 \\ 0 & 1 & 0 \\ s_0 & 0 & c_0 \end{bmatrix}}_{\mathbf{Q}(N)} \underbrace{\begin{bmatrix} d(2) \\ \lambda^{1/2}d(1) \\ \lambda\, d(0) \end{bmatrix}}_{\mathbf{d}(N)}. \tag{3.56}$$

The initialization of the lower triangular Cholesky factor can also be carried out in a recursive way [1]. For this case, it is assumed that $x(k) = 0$ for $k < 0$ and, for $k = 0, \ldots, N$, and the same update procedures for the lower triangular $\mathbf{U}(k)$ and vector $\mathbf{dq}(k)$ described in Table 3.2 are used. The iterative exact initialization procedure of a 3×3 lower triangular matrix ($N = 2$), for $k = 2$ is described as follows:
At $k = 0$:

$$\underbrace{\mathbf{Q}_{\theta_0}(0)}_{\mathbf{Q}_\theta(0)} \begin{bmatrix} x(0) & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ x(0) & 0 \end{bmatrix} \tag{3.57}$$

At $k = 1$:

$$\underbrace{\mathbf{Q}_{\theta_1}(1)\, \mathbf{Q}_{\theta_0}(1)}_{\mathbf{Q}_\theta(1)} \begin{bmatrix} x(1) & x(0) \\ 0 & 0 \\ \lambda^{1/2}x(0) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \bar{\bar{x}} \\ \bar{\bar{x}} & \bar{\bar{x}} \end{bmatrix} \tag{3.58}$$

At $k = 2$:

$$\underbrace{\mathbf{Q}_{\theta_2}(2)\,\mathbf{Q}_{\theta_1}(2)\,\mathbf{Q}_{\theta_0}(2)}_{\mathbf{Q}_\theta(2)}\begin{bmatrix} x(2) & x(1) & x(0) \\ 0 & 0 & 0 \\ 0 & \lambda^{1/2}\bar{\bar{x}} & 0 \\ \lambda^{1/2}\bar{\bar{x}} & \lambda^{1/2}\bar{\bar{x}} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & u(1,3) \\ 0 & u(2,2) & u(2,3) \\ u(3,1) & u(3,2) & u(3,3) \end{bmatrix}. \qquad (3.59)$$

It is worth mentioning that, in an infinite-precision environment, this exact initialization of the QRD-RLS algorithm is equivalent to the exact initialization of the RLS algorithm.

For the case of the **soft-start** procedure, one should choose $\mathbf{U}(-1) = \delta\mathbf{J}$, where $\mathbf{J}$ is the reversal matrix and δ is a regularization parameter with a value proportional to the standard deviation of the input signal. Note that $\mathbf{Jv}$ reverses vector $\mathbf{v}$: Its first element becomes the last one and vice-versa. The *soft-start* strategy is simpler than the exact initialization and its effect becomes negligible when k increases.

Regarding the soft-constrained initialization, it is relevant to note that:

- As in the conventional RLS algorithm [1], the soft initialization causes a bias and, if $\lambda < 1$, this bias tends to zero as k increases.
- In [10], the full quantitative analysis of the dynamic range of all internal quantities of the QRD-RLS algorithm was presented; it was also shown in this reference, for the case of fixed-point arithmetics, that when the value of the regularization parameter is approximately $\delta = \sigma_x/\sqrt{1-\lambda}$, σ_x^2 being the variance of the input signal, overflow in the internal variable with soft initialization can be avoided.

3.5 On the $\mathbf{Q}_\theta(k)$ Matrix

Different versions of QRD-RLS algorithms can be obtained from adequate interpretation of matrix $\mathbf{Q}_\theta(k)$. Below, based on (3.48), we present the key relations to allow such interpretation.

- Observing the fact that $\mathbf{Q}_\theta(k)$ is unitary, that is,

$$\begin{aligned} \mathbf{I}_{N+2} = \mathbf{Q}_\theta(k)\mathbf{Q}_\theta^{\mathrm{T}}(k) &= \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}\begin{bmatrix} \gamma(k) & \mathbf{f}^{\mathrm{T}}(k) \\ \mathbf{g}(k) & \mathbf{E}^{\mathrm{T}}(k) \end{bmatrix} \\ = \mathbf{Q}_\theta^{\mathrm{T}}(k)\mathbf{Q}_\theta(k) &= \begin{bmatrix} \gamma(k) & \mathbf{f}^{\mathrm{T}}(k) \\ \mathbf{g}(k) & \mathbf{E}^{\mathrm{T}}(k) \end{bmatrix}\begin{bmatrix} \gamma(k) & \mathbf{g}(k)^{\mathrm{T}} \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}, \end{aligned} \qquad (3.60)$$

  then

$$\mathbf{f}(k) = -\gamma^{-1}(k)\mathbf{E}(k)\mathbf{g}(k), \text{ and} \qquad (3.61)$$

$$\mathbf{g}(k) = -\gamma^{-1}(k)\mathbf{E}^{\mathrm{T}}(k)\mathbf{f}(k). \qquad (3.62)$$

- Replacing $\mathbf{Q}_\theta(k)$, as in (3.48), in (3.43), we have:

$$\begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} 0^{\mathrm{T}} \\ \mathbf{U}(k) \end{bmatrix}; \tag{3.63}$$

then, the next two relations follow:

$$\mathbf{f}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda^{1/2}\mathbf{E}(k)\mathbf{U}(k-1) = \mathbf{U}(k), \text{ and} \tag{3.64}$$

$$\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k). \tag{3.65}$$

- Observing the fact that $\mathbf{Q}_\theta(k)$ is unitary, we see that the norm of the expression in (3.63) obeys

$$\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1). \tag{3.66}$$

If we pre-multiply (3.66) by $\mathbf{U}^{-\mathrm{T}}(k)$, and confront the result

$$\mathbf{U}(k) = \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) + \lambda\mathbf{U}^{-\mathrm{T}}(k)\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1) \tag{3.67}$$

with (3.64), we see that $\mathbf{f}(k)$ and $\mathbf{E}(k)$ can be given by

$$\mathbf{f}(k) = \mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k), \text{ and} \tag{3.68}$$
$$\mathbf{E}(k) = \lambda^{1/2}\mathbf{U}^{-\mathrm{T}}(k)\mathbf{U}^{\mathrm{T}}(k-1). \tag{3.69}$$

The above relations are common to both triangularization methods. Especially (3.65), (3.68), and (3.69) are key relations for the comprehension of other algorithms of the QR family.

With the above, we can relate some expressions of the RLS algorithm, from the previous chapter, with their corresponding QRD-RLS counterparts. These relations, linking both algorithms, are shown in Table 3.3 where the expressions of the first column rise naturally from the RLS algorithm while the equations in the second column were basically obtained from (3.60).

Table 3.3 Relating equivalent expressions from the RLS and the QRD-RLS algorithms.

RLS	QRD-RLS
$\gamma^2(k) = 1 - \mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)\mathbf{x}(k)$	$\gamma^2(k) = 1 - \mathbf{f}^{\mathrm{T}}(k)\mathbf{f}(k) = 1 - \lVert\mathbf{f}(k)\rVert^2$
$\gamma^{-2}(k) = 1 + \lambda^{-1}\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)$	$\gamma^{-2}(k) = 1 + \lVert\gamma^{-1}(k)\mathbf{g}(k)\rVert^2$
$\lambda^{-1}\gamma^2(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k) = \mathbf{R}^{-1}(k)\mathbf{x}(k)$	$\mathbf{g}(k) = -\gamma^{-1}(k)\mathbf{E}^{\mathrm{T}}(k)\mathbf{f}(k)$
$\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) = \mathbf{R}(k) - \lambda\mathbf{R}(k-1)$	$\mathbf{f}(k)\mathbf{f}^{\mathrm{T}}(k) = \mathbf{I} - \mathbf{E}(k)\mathbf{E}^{\mathrm{T}}(k)$
$\mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) = \mathbf{I} - \lambda\mathbf{R}^{-1}(k)\mathbf{R}(k-1)$	$\mathbf{g}(k)\mathbf{g}^{\mathrm{T}}(k) = \mathbf{I} - \mathbf{E}^{\mathrm{T}}(k)\mathbf{E}(k)$

We next interpret the elements of $\mathbf{Q}_\theta(k)$ beginning with $\gamma(k)$. If $\mathbf{Q}_\theta(k)$ is regarded as in (3.48), the first element of the equation system in (3.45) can be written as

$$e_{q1}(k) = \gamma(k)d(k) + \mathbf{g}^{\mathrm{T}}(k)\lambda^{1/2}\mathbf{d}_{q_2}(k-1). \tag{3.70}$$

From (3.65) and using (3.17), the rotated error can be expressed as

$$e_{q1}(k) = \gamma(k)e(k), \tag{3.71}$$

where $e(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)$ is the *a priori* estimation error as defined in Chapter 2. With (3.52) and (3.71), it follows that

$$\varepsilon(k) = \gamma^2(k)e(k), \tag{3.72}$$

and $\gamma^2(k)$ represents the conversion factor between the *a priori* and the *a posteriori* output errors for the degree $N+1$ filtering problem. Note that, this same conversion factor is also used in the context of the RLS algorithm.

In order to interpret the other elements of $\mathbf{Q}_\theta(k)$, it is necessary to apply the QR decomposition to the problems of forward and backward predictions. In Sections 3.5.1 and 3.5.2, we consider the scheme shown in Figure 3.5 to present the prediction of a past sample $x(k-N-1)$ from vector $\mathbf{x}(k)$ (backward prediction) and the prediction of the current sample $x(k)$ from vector $\mathbf{x}(k-1)$ (forward prediction).

Throughout the next subsections, the $(k+1) \times i$ input data matrix is denoted as $\mathbf{X}^{(i)}(k)$ and all variables of backward and forward predictors, related to order i predictors ($i+1$ prediction coefficients), are indicated with the superscript $^{(i+1)}$,

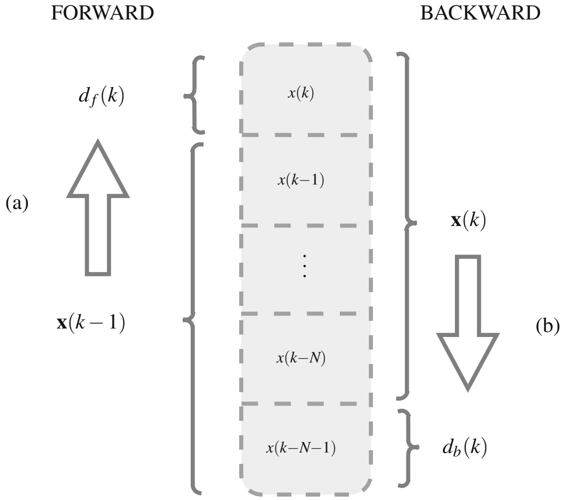


Fig. 3.5 Signal prediction at instant k and order N. (a) Forward prediction: sample $x(k)$ is predicted from vector $\mathbf{x}(k-1)$. (b) Backward prediction: sample $x(k-N-1)$ is predicted from vector $\mathbf{x}(k)$.

e.g., $\mathbf{e}_b^{(i+1)}(k)$ and $\mathbf{e}_f^{(i+1)}(k)$. However, for convenience of notation, order N predictors are indicated without the superscript ${}^{(N+1)}$, i.e., $\mathbf{e}_b(k) = \mathbf{e}_b^{(N+1)}(k)$ and $\mathbf{e}_f(k) = \mathbf{e}_f^{(N+1)}(k)$.

3.5.1 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 more recent available information of this sequence. In the problem of order N at instant k, the prediction of the desired backward sample $x(k-N-1)$ is based on vector $\mathbf{x}(k)$. The weighted backward prediction error vector is defined as

$$\mathbf{e}_b(k) = \underbrace{\begin{bmatrix} x(k-N-1) \\ \lambda^{1/2}x(k-N-2) \\ \vdots \\ \lambda^{(k-N-1)/2}x(0) \\ 0_{N+1} \end{bmatrix}}_{\mathbf{d}_b(k)} - \underbrace{\begin{bmatrix} x(k) & \cdots & x(k-N) \\ \lambda^{1/2}x(k-1) & \cdots & \lambda^{1/2}x(k-N-1) \\ \vdots & \ddots & \vdots \\ \lambda^{(k-1)/2}x(1) & \cdots & 0 \\ \lambda^{k/2}x(0) & \cdots & 0 \end{bmatrix}}_{\mathbf{X}(k)} \mathbf{w}_b(k), \quad (3.73)$$

where $\mathbf{w}_b(k)$ is the backward prediction coefficient vector and $\mathbf{d}_b(k)$ the weighted backward desired signal vector.

In the backward prediction problem of order $i-1$ the weighted backward prediction error vector is

$$\mathbf{e}_b^{(i)}(k) = \mathbf{d}_b^{(i)}(k) - \mathbf{X}^{(i)}(k)\mathbf{w}_b^{(i)}(k), \quad (3.74)$$

where the weighted desired signal vector

$$\mathbf{d}_b^{(i)}(k) = \left[x(k-i) \;\; \lambda^{1/2}x(k-i-1) \;\; \cdots \;\; \lambda^{(k-i)/2}x(0) \;\; 0_i^{\mathrm{T}} \right]^{\mathrm{T}} \quad (3.75)$$

represents the $(i+1)$th column of the data matrix in (3.5) and is denoted as $\mathbf{x}^{(i)}(k)$.

By differentiating $\mathbf{e}_b^{(i)^{\mathrm{T}}}(k)\mathbf{e}_b^{(i)}(k)$ with respect to $\mathbf{w}_b^{(i)}(k)$, the optimum backward prediction coefficient vector is given by

$$\mathbf{w}_b^{(i)}(k) = \left[\mathbf{X}^{(i)^{\mathrm{T}}}(k)\boldsymbol{X}^{(i)}(k) \right]^{-1} \mathbf{X}^{(i)^{\mathrm{T}}}(k)\mathbf{d}_b^{(i)}(k). \quad (3.76)$$

From the relations used in backward prediction, it is worth noting that:

- For $i = 0$, $\mathbf{e}_b^{(0)}(k) = \mathbf{d}_b^{(0)}(k) = \mathbf{x}^{(0)}(k)$. For $\{i = 1, \ldots, N\}$, the data matrix can be denoted as $\mathbf{X}^{(i+1)}(k) = \left[\mathbf{X}^{(i)}(k) \;\; \mathbf{d}_b^{(i)}(k) \right]$ and (3.74) can be rewritten as

$$\mathbf{e}_b^{(i)}(k) = \mathbf{X}^{(i+1)}(k) \begin{bmatrix} -\mathbf{w}_b^{(i)}(k) \\ 1 \end{bmatrix}. \quad (3.77)$$

- The error vectors in (3.77) for $\{i=0,\ldots,N\}$ can be collected in matrix

$$\mathbf{G}(k)=\mathbf{X}(k)\mathbf{K}^{-1}(k), \tag{3.78}$$

where

$$\mathbf{G}(k)=\left[\mathbf{e}_b^{(N)}(k)\;\mathbf{e}_b^{(N-1)}(k)\;\cdots\;\mathbf{e}_b^{(0)}(k)\right], \tag{3.79}$$

and

$$\mathbf{K}^{-1}(k)=\begin{bmatrix} -w_{b,0}^{(N)}(k) & -w_{b,0}^{(N-1)}(k) & \cdots & -w_{b,0}^{(1)}(k) & 1 \\ -w_{b,1}^{(N)}(k) & -w_{b,1}^{(N-1)}(k) & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -w_{b,N-1}^{(N)}(k) & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{3.80}$$

- The $(i+1)$th column of $\mathbf{K}^{-1}(k)$ for $i=0,\ldots,N$, represents the coefficients of the backward prediction errors filters of order $N-i$;
- The first row of matrix $\mathbf{G}(k)$ in (3.78) corresponds to the *a posteriori* backward prediction error (with different orders and at instant k) vector transposed, i.e.,

$$\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k)=\begin{bmatrix} \varepsilon_b^{(N)}(k) \\ \varepsilon_b^{(N-1)}(k) \\ \vdots \\ \varepsilon_b^{(1)}(k) \\ \varepsilon_b^{(0)}(k) \end{bmatrix}. \tag{3.81}$$

- When $\mathbf{w}_b^{(i)}(k)$ fulfills (3.76) for $\{i=N,N-1,\ldots,1,0\}$, the product $\mathbf{D}^2(k)=\mathbf{G}^{\mathrm{T}}(k)\mathbf{G}(k)$ is a diagonal matrix whose elements $||\mathbf{e}_b^{(i)}(k)||^2$ represent the minimum backward prediction errors energy.
- If we replace first (3.76) in (3.74) and then $\mathbf{X}^{(i)}(k)$ by $\mathbf{G}^{(i)}(k)\mathbf{K}^{(i)}(k)$ (of order $i-1$), we obtain the expression

$$\mathbf{e}_b^{(i)}(k)=\mathbf{x}^{(i)}(k)-\mathbf{G}(k)\mathbf{D}^{-2}(k)\mathbf{G}^{\mathrm{T}}(k)\mathbf{x}^{(i)}(k). \tag{3.82}$$

For convenience, we rewrite the last equation as

$$\mathbf{e}_b^{(i)}(k)=\mathbf{x}^{(i)}(k)-\sum_{j=0}^{i-1}\mathbf{e}_j\;\kappa_{ji}(k), \tag{3.83}$$

with $\mathbf{e}_j=\mathbf{e}_b^{(N-j)}(k)$, $\kappa_{ji}=\mathbf{e}_j^{\mathrm{T}}\mathbf{x}^{(i)}(k)/\parallel\mathbf{e}_j\parallel^2$ and $j<i$.
- The set of error vectors for $\{i=0,\ldots,N\}$ can be rewritten as

$$\mathbf{X}(k) = [\mathbf{e}_0\,\mathbf{e}_1 \cdots \mathbf{e}_N] \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & & \kappa_{1N} \\ 0 & & & \vdots \\ 1 & \kappa_{N1} & \cdots & \kappa_{NN} \end{bmatrix} = \mathbf{G}(k)\mathbf{K}(k). \tag{3.84}$$

- Equation (3.84) represents the Gram–Schmidt orthogonalization procedure [3] of the columns of matrix $\mathbf{X}(k)$ of special interest for the case of a lower triangular Cholesky factor.

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

$$\mathbf{e}_{b_q}(k) = \mathbf{Q}(k)\mathbf{e}_b(k) = \begin{bmatrix} \mathbf{O} & \mathbf{e}_{b_{q1}}(k) \\ \mathbf{U}(k) & \mathbf{d}_{b_{q2}}(k) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b(k) \\ 1 \end{bmatrix} \tag{3.85}$$

By following similar procedure as the one used in the WLS estimation problem, it is possible to show that

$$\varepsilon_b(k) = \gamma(k)e_{b_{q1}}(k) = \gamma^2(k)e_b(k). \tag{3.86}$$

3.5.2 The forward prediction problem

In the forward prediction problem, we obtain an estimate of a sample of a given input sequence using the past available information of this sequence. In the problem of order N at instant k, the prediction of the desired signal $x(k)$ is based on vector $\mathbf{x}(k-1)$ and the weighted forward prediction error vector is defined as

$$\mathbf{e}_f(k) = \underbrace{\begin{bmatrix} x(k) \\ \lambda^{1/2}x(k-1) \\ \vdots \\ \lambda^{(k-1)/2}x(1) \\ \lambda^{k/2}x(0) \end{bmatrix}}_{\mathbf{d}_f(k)} - \underbrace{\begin{bmatrix} x(k-1) & \cdots & x(k-N-1) \\ \lambda^{1/2}x(k-2) & \cdots & \lambda^{1/2}x(k-N-2) \\ \vdots & \ddots & \vdots \\ \lambda^{(k-1)/2}x(0) & \cdots & 0 \\ 0 & \cdots & 0 \end{bmatrix}}_{\begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix}} \mathbf{w}_f(k) \tag{3.87}$$

where $\mathbf{w}_f(k)$ is the forward prediction coefficient vector. Note that the last row of the data matrix, which contains zeros, appears due to the fact that we assume $x(k) = 0$ for $k < 0$.

The weighted forward prediction error vector, of order $i-1$ and instant k, is given as

$$\mathbf{e}_f^{(i)}(k) = \mathbf{d}_f^{(i)}(k) - \begin{bmatrix} \mathbf{X}^{(i)}(k-1) \\ 0^{\mathrm{T}} \end{bmatrix} \mathbf{w}_f^{(i)}(k), \tag{3.88}$$

where $\mathbf{d}_f^{(i)}(k)$ is the weighted desired signal and represents the first column of the data matrix denoted in (3.5) as $\mathbf{x}^{(0)}(k)$, that is,

$$\mathbf{d}_f^{(i)}(k) = \left[x(k) \;\; \lambda^{1/2} x(k-1) \;\; \cdots \;\; \lambda^{k/2} x(0) \right]^{\mathrm{T}}. \tag{3.89}$$

By differentiating $[\mathbf{e}_f^{(i)}(k)]^{\mathrm{T}} \mathbf{e}_f^{(i)}(k)$ with respect to $\mathbf{w}_f^{(i)}(k)$ and equating the result to zero, we find the optimum forward prediction coefficient vector of order $(i-1)$, i.e.,

$$\mathbf{w}_f^{(i)}(k) = \left\{ \left[\mathbf{X}^{(i)}(k-1) \right]^{\mathrm{T}} \mathbf{X}^{(i)}(k-1) \right\}^{-1} \begin{bmatrix} \mathbf{X}^{(i)}(k-1) \\ 0^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \mathbf{d}_f^{(i)}(k). \tag{3.90}$$

From the relations used in forward prediction, it is relevant to note that:

- At instant $\ell = k - N$ and for $i = 0$, the weighted forward prediction error vector is $\mathbf{e}_f^{(0)}(k-N) = \mathbf{d}_f^{(0)}(k-N) = \mathbf{x}^{(0)}(k-N)$. At instant $\ell + i$ with $\{i = 1, \ldots, N\}$ the data matrix with dimension $(\ell + i + 1) \times (i + 1)$ can be represented as $\mathbf{X}^{(i+1)}$ $(\ell + i) = \left[\mathbf{d}_f^{(i)}(\ell + i) \;\; \begin{matrix} \mathbf{X}^{(i)}(\ell + i - 1) \\ 0^{\mathrm{T}} \end{matrix} \right]$, and (3.88) can be rewritten as

$$\mathbf{e}_f^{(i)}(\ell + i) = \mathbf{X}^{(i+1)}(\ell + i) \begin{bmatrix} 1 \\ -\mathbf{w}_f^{(i)}(\ell + i) \end{bmatrix}. \tag{3.91}$$

- The error vectors in (3.91) for $\{i = 0, \ldots, N\}$ can be collected into matrix

$$\mathbf{G}(k) = \mathbf{X}(k) \mathbf{K}^{-1}(k), \tag{3.92}$$

where

$$\mathbf{G}(k) = \left[\mathbf{e}_0 \; \mathbf{e}_1 \; \cdots \; \mathbf{e}_N \right], \tag{3.93}$$

with

$$\mathbf{e}_i = \begin{bmatrix} \mathbf{e}_f^{(i)}(k - N + i) \\ 0_{N-i} \end{bmatrix}, \tag{3.94}$$

and

$$\mathbf{K}^{-1}(k) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & -w_{f,0}^{(N)}(k) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & -w_{f,N-3}^{(N-1)}(k-1) & -w_{f,N-2}^{(N)}(k) \\ 1 & -w_{f,0}^{(1)}(k-N+1) & \cdots & -w_{f,N-2}^{(N-1)}(k-1) & -w_{f,N-1}^{(N)}(k) \end{bmatrix}. \tag{3.95}$$

- The $(i+1)$th column of $\mathbf{K}^{-1}(k)$ for $i = 0,\ldots,N$, represents the coefficients of the forward prediction error filters of order i at instant $k-N+i$.
- The first row of the system of equations in (3.92) is the *a posteriori* forward prediction error vector, with different orders, and at distinct time instants, i.e.,

$$\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k) = \begin{bmatrix} \varepsilon_f^{(0)}(k-N) \\ \varepsilon_f^{(1)}(k-N+1) \\ \vdots \\ \varepsilon_f^{(N)}(k) \end{bmatrix}. \tag{3.96}$$

- When $\mathbf{w}_f^{(i)}(k-N+i)$ obeys (3.90) for $\{i = N,\ldots,0\}$, the product $\mathbf{D}^2(k) = \mathbf{G}^{\mathrm{T}}(k)\mathbf{G}(k)$ is a diagonal matrix whose elements are the minimum forward prediction error energy $||\mathbf{e}_f^{(i)}(k-N+i)||^2$.
- If we replace (3.90) in (3.88) and rewrite the data matrix $\mathbf{X}^{(i)}(k-1)$ as $\mathbf{G}^{(i)}(k-1)\mathbf{K}^{(i)}(k-1)$, we obtain, at instant $\ell+i$ with $\ell = k-N$,

$$\mathbf{e}_f^{(i)}(\ell+i) = \mathbf{d}_f^{(i)}(\ell+i) - \begin{bmatrix} \sum_{j=0}^{i-1} \mathbf{e}_f^{(j)}(\ell+j-1)\ \kappa_{ij} \\ 0 \end{bmatrix}, \tag{3.97}$$

where $\kappa_{ij} = \begin{bmatrix} \mathbf{e}_f^{(j)}(\ell+j-1) \\ 0 \end{bmatrix}^{\mathrm{T}} \mathbf{d}_f^{(i)}(\ell+i)/||\mathbf{e}_f^{j}(\ell+j)||^2$ and $j < i$.

- The set of weighted forward prediction error vectors for $\{i = 0,\ldots,N\}$ can be rewritten as

$$\mathbf{X}(k) = [\mathbf{e}_0\ \mathbf{e}_1 \cdots \mathbf{e}_N] \begin{bmatrix} \kappa_{00} & \cdots & \kappa_{0\,N-1} & 1 \\ \vdots & & & 0 \\ & & & \vdots \\ \kappa_{N-1\,0} & & & \\ 1 & 0 & \cdots & \end{bmatrix} = \mathbf{G}(k)\mathbf{K}(k). \tag{3.98}$$

- Equation (3.98) represents the Gram–Schmidt orthogonalization procedure [3] of the columns of $\mathbf{X}(k)$ matrix of special interest for the case of an upper triangular Cholesky factor.

The rotated weighted forward prediction error vector, obtained from (3.88) with $i = N+1$, is then defined as

$$\mathbf{e}_{f_q}(k) = \begin{bmatrix} \mathbf{Q}(k-1) & 0 \\ 0^{\mathrm{T}} & 1 \end{bmatrix} \mathbf{e}_f(k) = \begin{bmatrix} \mathbf{e}_{f_{q_1}}(k) & \mathbf{O} \\ \mathbf{d}_{f_{q_2}}(k) & \mathbf{U}(k-1) \\ \lambda^{k/2}x(0) & 0^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k) \end{bmatrix}. \tag{3.99}$$

By following similar procedures as the one used in the WLS estimation problem, it is possible to show that

$$\varepsilon_f(k) = \gamma(k-1)e_{f_{q_1}}(k) = \gamma^2(k-1)e_f(k). \tag{3.100}$$

3.5.3 Interpreting the elements of $\mathbf{Q}_\theta(k)$ for a lower triangular Cholesky factor

This subsection provides insightful information about the variables used by the QR algorithms. To start, we observe that both representations of the input data matrix, in (3.84) and in (3.98), can be used to rewrite the autocorrelation matrix as

$$\begin{aligned}\mathbf{R}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) &= \mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\\ &= \mathbf{K}^{\mathrm{T}}(k)\mathbf{G}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{K}(k)\\ &= [\mathbf{D}(k)\mathbf{K}(k)]^{\mathrm{T}}[\mathbf{D}(k)\mathbf{K}(k)],\end{aligned} \tag{3.101}$$

and the projection operator as

$$\mathscr{P}(k) = \mathbf{X}(k)\mathbf{R}^{-1}(k)\mathbf{X}^{\mathrm{T}}(k) = \mathbf{Q}_2^{\mathrm{T}}(k)\mathbf{Q}_2(k) = \mathbf{G}(k)\mathbf{D}^{-2}(k)\mathbf{G}^{\mathrm{T}}(k). \tag{3.102}$$

The term $\mathbf{D}(k)\mathbf{K}(k)$ may represent the Cholesky factor of the deterministic autocorrelation matrix, defined in (3.15) as $\mathbf{U}(k)$, and the columns of matrix $\mathbf{Q}_2^{\mathrm{T}}(k) = \mathbf{G}(k)\mathbf{D}^{-1}(k)$ represent the prediction error vectors normalized by the prediction error energy.

Once $\mathbf{U}(k) = \mathbf{D}(k)\mathbf{K}(k)$, using (3.48), (3.68), (3.69) and (3.65), yields the following expression for $\mathbf{Q}_\theta(k)$:

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & -\gamma(k)\lambda^{-1/2}\left[\mathbf{D}^{-1}(k-1)\boldsymbol{K}^{-\mathrm{T}}(k-1)\mathbf{x}(k)\right]^{\mathrm{T}} \\ \mathbf{D}^{-1}(k)\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k) & \lambda^{1/2}\mathbf{D}^{-1}(k)\mathbf{K}^{-\mathrm{T}}(k)\mathbf{K}^{\mathrm{T}}(k-1)\mathbf{D}^{\mathrm{T}}(k-1) \end{bmatrix}. \tag{3.103}$$

Equations (3.101, 3.102, 3.103) are valid for both types of triangularization. Nevertheless, from the backward prediction problem, where the product $\mathbf{D}(k)\mathbf{K}(k)$ corresponds to a lower triangular matrix, if we recall the physical interpretation of $\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k)$ as in (3.81) and that the elements of the diagonal matrix $\mathbf{D}(k)$ are given by $\parallel \mathbf{e}_b^{(N-i)}(k) \parallel$, it follows that

$$\mathbf{f}(k) = \mathbf{D}^{-1}(k)\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k) = \begin{bmatrix} \varepsilon_b^{(N)}(k)/\parallel \mathbf{e}_b^{(N)}(k) \parallel \\ \varepsilon_b^{(N-1)}(k)/\parallel \mathbf{e}_b^{(N-1)}(k) \parallel \\ \vdots \\ \varepsilon_b^{(0)}(k)/\parallel \mathbf{e}_b^{(0)}(k) \parallel \end{bmatrix}. \tag{3.104}$$

Thus, $\mathbf{f}(k)$, for the case of lower triangularization of the Cholesky factor, is the *a posteriori* backward prediction error vector at instant k normalized by backward error energies at the same instant.

Moreover, from the same interpretation of $\mathbf{D}(k-1)$ and $\mathbf{K}^{-\mathrm{T}}(k-1)$, vector $\mathbf{g}(k)$, from (3.48) and (3.65), may be given as

$$\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2}\begin{bmatrix} e_b^{(N)}(k)/\parallel \mathbf{e}_b^{(N)}(k-1)\parallel \\ e_b^{(N-1)}(k)/\parallel \mathbf{e}_b^{(N-1)}(k-1)\parallel \\ \vdots \\ e_b^{(0)}(k)/\parallel \mathbf{e}_b^{(0)}(k-1)\parallel \end{bmatrix}. \tag{3.105}$$

Thus, $\mathbf{g}(k)$ is the *a priori* backward prediction error vector at instant k weighted by $-\gamma(k)\lambda^{-1/2}$ and normalized by backward error energies at instant $k-1$.

3.5.4 Interpreting the elements of $\mathbf{Q}_\theta(k)$ for an upper triangular Cholesky factor

For the case of an upper triangular Cholesky factor, the product $\mathbf{D}(k)\mathbf{K}(k)$ comes from the forward prediction orthogonalization procedure. It is also worth mentioning that (3.95) brings an interpretation of the non-zero elements of the rows of $\mathbf{K}^{-\mathrm{T}}(k)$ as the coefficients of forward prediction filters of different orders at distinct time instants. Recalling the physical interpretation of $\mathbf{K}^{-\mathrm{T}}(k)\mathbf{x}(k)$ as (3.96) and that the elements of the diagonal matrix $\mathbf{D}(k)$ are also given by $\parallel \mathbf{e}_f^{(i)}(k-N+i)\parallel$, it follows that

$$\mathbf{f}(k) = \begin{bmatrix} \varepsilon_f^{(0)}(k-N)/\parallel \mathbf{e}_f^{(0)}(k-N)\parallel \\ \varepsilon_f^{(1)}(k-N+1)/\parallel \mathbf{e}_f^{(1)}(k-N+1)\parallel \\ \vdots \\ \varepsilon_f^{(N)}(k)/\parallel \mathbf{e}_f^{(N)}(k)\parallel \end{bmatrix}. \tag{3.106}$$

In this case, $\mathbf{f}(k)$ is the *a posteriori* forward prediction error vector normalized by forward error energies at different time instants.

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

$$\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2}\begin{bmatrix} e_f^{(0)}(k-N)/\parallel \mathbf{e}_f^{(0)}(k-N-1)\parallel \\ e_f^{(1)}(k-N+1)/\parallel \mathbf{e}_f^{(1)}(k-N)\parallel \\ \vdots \\ e_f^{(N)}(k)/\parallel \mathbf{e}_f^{(N)}(k-1)\parallel \end{bmatrix}. \tag{3.107}$$

Vector $\mathbf{g}(k)$ is then an *a priori* forward prediction error vector normalized by the *a posteriori* forward error energies and weighted by $-\gamma(k)\lambda^{-1/2}$.

We note that, in the case of upper triangularization, the normalized errors present in $\mathbf{Q}_\theta(k)$ are of different orders at distinct instants of time (order and time updating); this fact seems to be the cause of the extra computational effort of the fast—or $\mathscr{O}[N]$ (order of N)—algorithms derived from this type of triangularization.

With the physical interpretation of vectors $\mathbf{g}(k)$ and $\mathbf{f}(k)$ presented above and taking into account Equations (3.61) and (3.62), matrix $\mathbf{E}(k)$ can be interpreted as a *conversion factor matrix* between the *a priori* and the *a posteriori* prediction error vectors.

3.6 The Inverse QRD-RLS Algorithm

An alternative approach to the conventional QRD-RLS algorithm based on the update of the inverse Cholesky factor was presented in [11]. This algorithm, known as the Inverse QR decomposition (IQRD-RLS) algorithm, allows the calculation of the weight vector without back-substitution. In the following, based on the structure of $\mathbf{Q}_\theta(k)$ and on the relations (3.62), (3.65), (3.68), and (3.69), we present the IQRD-RLS algorithm.

Starting from the RLS solution $\mathbf{w}(k) = \mathbf{R}^{-1}(k)\mathbf{p}(k)$ with $\mathbf{R}(k)$ and $\mathbf{p}(k)$ defined as in (3.9) and in (3.8), respectively, and using the expression in (3.38) instead of $\mathbf{X}(k)$, after some manipulations, we can show that

$$\mathbf{w}(k) = \mathbf{w}(k-1) + e(k)\mathbf{U}^{-1}(k)\mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k), \tag{3.108}$$

where $e(k) = d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k-1)$ is the *a priori* error and the term multiplying this variable is known as the *Kalman Gain*. Also note, knowing that $\mathbf{R}(k) = \mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k)$, that (3.108) corresponds to (2.55), from the previous chapter.

Since we know that $\mathbf{Q}_\theta(k)$ is unitary, if we post-multiply this matrix by its first row transposed, it follows that

$$\mathbf{Q}_\theta(k)\begin{bmatrix}\gamma(k)\\ \mathbf{g}(k)\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix}. \tag{3.109}$$

From (3.65), we have $\mathbf{g}(k) = -\gamma(k)\lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k)$. For convenience, we define

$$\mathbf{a}(k) = -\gamma^{-1}(k)\mathbf{g}(k) = \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1)\mathbf{x}(k) \tag{3.110}$$

and rewrite (3.109) as

$$\mathbf{Q}_\theta(k)\begin{bmatrix}1\\ -\mathbf{a}(k)\end{bmatrix} = \begin{bmatrix}\gamma^{-1}(k)\\ 0\end{bmatrix}. \tag{3.111}$$

This expression, if we know $\mathbf{a}(k)$, provides $\mathbf{Q}_\theta(k)$. At this point, it is relevant to observe that (3.69), rewritten as $\lambda^{-1/2}\mathbf{E}(k)\mathbf{U}^{-\mathrm{T}}(k-1) = U^{-\mathrm{T}}(k)$, suggests that $\mathbf{U}^{-\mathrm{T}}(k-1)$ can be updated with the same matrix that updates $\mathbf{U}(k-1)$. In fact, if we rotate $\left[0 \;\; \lambda^{-1/2}\mathbf{U}^{-1}(k-1)\right]^{\mathrm{T}}$ with $\mathbf{Q}_\theta(k)$, we obtain

$$\begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix} \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \begin{bmatrix} \lambda^{-1/2}\mathbf{g}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k-1) \\ \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix}. \tag{3.112}$$

For convenience, we define $\mathbf{u}(k) = \lambda^{-1/2}\mathbf{U}^{-1}(k-1)\mathbf{g}(k)$. Using vector $\mathbf{a}(k)$ from (3.110), this vector can be expressed as

$$\mathbf{u}(k) = -\lambda^{-1/2}\gamma(k)\mathbf{U}^{-1}(k-1)\mathbf{a}(k), \tag{3.113}$$

or, using (3.62), (3.68), and (3.69), as

$$\mathbf{u}(k) = -\gamma^{-1}(k)\mathbf{U}^{-1}(k)\mathbf{U}^{-\mathrm{T}}(k)\mathbf{x}(k). \tag{3.114}$$

Finally, with this last equation, (3.108) can be rewritten as

$$\mathbf{w}(k) = \mathbf{w}(k-1) - e(k)\gamma(k)\mathbf{u}(k), \tag{3.115}$$

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

By combining (3.112) and (3.111) in one single equation, we have

$$\begin{bmatrix} 1/\gamma(k) & \mathbf{u}^{\mathrm{T}}(k) \\ 0 & \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ -\mathbf{a}(k) & \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix}. \tag{3.116}$$

Equation (3.116) is a key relation to the inverse QRD-RLS algorithm, which equations are presented in Table 3.4. A pseudo-code of the (lower-triangularization version) inverse QRD-RLS algorithm is available in Appendix 3 (Table 3.6).

To close this section, we note that Equation (3.45) can be added to (3.116), i.e,

$$\begin{bmatrix} \gamma^{-1}(k) & \mathbf{u}^{\mathrm{T}}(k) & e_{q_1}(k) \\ 0 & \mathbf{U}^{-\mathrm{T}}(k) & \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} & d(k) \\ -\mathbf{a}(k) & \lambda^{-1/2}\mathbf{U}^{-\mathrm{T}}(k-1) & \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} \tag{3.117}$$

The resulting set of equations is known as the extended QRD-RLS algorithm. It was presented in [12], before the inverse QRD-RLS algorithm [11].

3.7 Conclusion

This chapter presented concepts and derivations of the basic algorithms belonging to the QRD-RLS family: the conventional and the inverse QRD-RLS algorithms. We started by noting that the QR decomposition, when applied to solve the LS

Table 3.4 The inverse QRD-RLS equations.

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

problem, comprises the rotation of the space spanned by the columns of the input data matrix. The relationship between the conventional LS and the QRD-LS methods was established and the orthogonality principle was shown in the rotate signal domain.

Using the recursive structure of the data matrix, the rotated RLS solution of (3.45) follows. It is worth mentioning that, if we apply the transformation

$$\begin{bmatrix} \gamma(k) & \mathbf{0}^{\mathrm{T}} \\ 0 & \mathbf{U}(k) \end{bmatrix} \tag{3.118}$$

in the filtering and adaptation operations of the RLS algorithm, i.e.,

$$\begin{bmatrix} e(k) \\ \mathbf{w}(k) \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{x}^{\mathrm{T}}(k) \\ \mathbf{R}^{-1}(k)\mathbf{x}(k) & \mathbf{I} - \mathbf{R}^{-1}(k)\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k) \end{bmatrix} \begin{bmatrix} d(k) \\ \mathbf{w}(k-1) \end{bmatrix}, \tag{3.119}$$

after some algebra, the same conventional QRD-RLS algorithm of (3.45) follows, that is,

$$\begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q2}(k-1) \end{bmatrix}. \tag{3.120}$$

In fact, with the change of coordinates, the system described in (3.119) can be transformed into another system, with different internal descriptions, but with the same input–output behavior for solving the LS problem. Although theoretically equivalent, the resulting system may have different numerical behavior when implemented in finite precision arithmetics [13]. Concerning specially the transformed system in (3.120), it is possible to find a numerically robust algorithm, e.g., those in [8, 14].

In Section 3.5, the structure of matrix $\mathbf{Q}_\theta(k)$ and the physical interpretation of its internal variables were presented. From the known structure of matrix $\mathbf{Q}_\theta(k)$, the IQRD-RLS algorithm was easily obtained. This algorithm, besides the inherited numerical robustness of its family, provides the coefficient vector at every iteration, without having to resort to the computationally onerous backward or forward substitution procedures. With the interpretation of the internal variables of $\mathbf{Q}_\theta(k)$, expressed in terms of backward and forward prediction errors, we have also provided all necessary tools to help readers understand the forthcoming chapters of this book.

It is worth mentioning that the QRD-RLS algorithms addressed in this chapter present a computational complexity of $\mathscr{O}[N^2]$ and preserve the desirable fast convergence property of the conventional RLS algorithm. Hence, assuming an infinite-precision environment, the outcomes of both algorithms (the RLS and the QRD-RLS, conventional or inverse), once initialized in an equivalent form, are identical in terms of speed of convergence and quality of estimation. Since the solution obtained by the QRD-RLS algorithm corresponds to the solution of the RLS algorithm from a transformed domain, the good numerical behavior can be presumed as a direct consequence of this transformation.

Appendix 1 - Forward and Back-Substitution Procedures

In order to show two possible procedures used to obtaining the coefficient vector with one order of magnitude less computational complexity than matrix inversion [15], consider the following system of equations:

$$\mathbf{U}\mathbf{x} = \mathbf{y}, \tag{3.121}$$

where $\mathbf{U}$ is a $(N+1) \times (N+1)$ triangular matrix, as illustrated in Figure 3.1, and $\mathbf{x}$ is the $(N+1) \times 1$ vector we need to obtain.

When $\mathbf{U}$ is upper triangular, we use the***forward substitution procedure***:

$$\begin{aligned} x_1 &= \frac{y_{N+1}}{U_{N+1,1}} \\ x_i &= \frac{1}{U_{\ell,i}}\left(y_\ell - \sum_{j=1}^{i-1} U_{\ell,j} x_j\right) \end{aligned} \tag{3.122}$$

for $i = 2, 3, \ldots, N+1$ and $\ell = N-i+2$.

When $\mathbf{U}$ is lower triangular, we use the ***back-substitution procedure***:

$$\begin{aligned} x_{N+1} &= \frac{y_1}{U_{1,N+1}} \\ x_i &= \frac{1}{U_{\ell,i}}\left(y_\ell - \sum_{j=i+1}^{N+1} U_{\ell,j} x_j\right) \end{aligned} \tag{3.123}$$

for $i = N, N-1, \ldots, 1$ and $\ell = N-i+2$.

Appendix 2 - Square-Root-Free QRD-RLS Algorithms

The two-dimensional vector rotations, necessary to execute the QRD-RLS algorithm, can be efficiently implemented using a CORDIC (COrdinate Rotation DIgital Computer) processor [16, 17]. In systems based on general-purpose programable DSPs, vector rotations requires a SQuare-RooT (SQRT) operation, which may eventually represent a bottleneck due to its significant computational burden. To circumvent this problem, many authors have proposed square-root-free methods for performing Givens rotations. Gentleman, in his pioneer work [18], shows how to perform the plane rotations without SQRT. After that, different versions of Givens rotations without SQRT were introduced (see, e.g. [19–23]).

Among different implementations of rotations without SQRT, for the sake of simplicity, we address the version introduced in [19]. We thus start with the following rotation.

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ u_1 & u_2 \end{bmatrix} = \begin{bmatrix} 0 & x_2' \\ u_1' & u_2' \end{bmatrix} \tag{3.124}$$

In order to have the first element of the first column nulled, the rotation angle is such that $\cos\theta_1 = u_1/u_1'$, $\sin\theta_1 = x_1/u_1'$ and $u_1' = \sqrt{x_1^2 + u_1^2}$, which requires a SQRT operation.

The main "trick" behind this class of algorithm is to scale the rows as follows.

$$\begin{aligned} x_i &= \delta_1^{1/2}\,\bar{x}_i \\ u_i &= d_1^{1/2}\,\bar{u}_i \\ u_i' &= d_2^{1/2}\,\bar{u}_i' \end{aligned} \tag{3.125}$$

for $i = 1,2$ and, also, $x_2' = \delta_2^{1/2}\,\bar{x}_2'$. In terms of the new quantities, the Givens rotation in (3.124) can be rewritten as

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \delta_1^{1/2} & 0 \\ 0 & d_1^{1/2} \end{bmatrix} \begin{bmatrix} \bar{x}_1 & \bar{x}_2 \\ \bar{u}_1 & \bar{u}_2 \end{bmatrix} = \begin{bmatrix} \delta_2^{1/2} & 0 \\ 0 & d_2^{1/2} \end{bmatrix} \begin{bmatrix} 0 & \bar{x}_2' \\ \bar{u}_1' & \bar{u}_2' \end{bmatrix}. \tag{3.126}$$

By rearranging the transformations, we obtain

$$\bar{G} \begin{bmatrix} \bar{x}_1 & \bar{x}_2 \\ \bar{u}_1 & \bar{u}_2 \end{bmatrix} = \begin{bmatrix} 0 & \bar{x}_2' \\ \bar{u}_1' & \bar{u}_2' \end{bmatrix}, \tag{3.127}$$

where

$$\bar{G} = \begin{bmatrix} \cos\theta_1 \sqrt{\delta_1/\delta_2} & -\sin\theta_1 \sqrt{d_1/\delta_2} \\ \sin\theta_1 \sqrt{\delta_1/d_2} & \cos\theta_1 \sqrt{d_1/d_2} \end{bmatrix}. \tag{3.128}$$

We now rewrite $\cos\theta_1$ in terms of the new quantities:

$$\cos\theta_1 = \frac{u_1}{u_1'} = \frac{d_1^{1/2}\,\bar{u}_1}{d_2^{1/2}\,\bar{u}_1'}. \tag{3.129}$$

To find an adequate scale factor, we set

$$\cos\theta_1 = \sqrt{d_2/d_1} = \sqrt{\delta_2/\delta_1}. \tag{3.130}$$

From (3.129) and (3.130), it follows that $d_2\bar{u}'_1 = d_1\bar{u}_1$. Therefore

$$\sin\theta_1 = \frac{x_1}{u'_1} = \frac{\delta_1^{1/2}\bar{x}_1}{d_2^{1/2}\bar{u}'_1} = \frac{d_2^{1/2}\delta_1^{1/2}\bar{x}_1}{d_1\bar{u}_1}, \tag{3.131}$$

$$\sin\theta_1\,\sqrt{\delta_1/d_2} = \frac{d_2^{1/2}\delta_1^{1/2}\bar{x}_1}{d_1\bar{u}_1}\,\frac{\delta_1^{1/2}}{d_2^{1/2}} = \frac{\delta_1\bar{x}_1}{d_1\bar{u}_1}, \text{ and} \tag{3.132}$$

$$\sin\theta_1\,\sqrt{d_1/\delta_2} = \frac{d_2^{1/2}\delta_1^{1/2}\bar{x}_1}{d_1\bar{u}_1}\,\frac{d_1^{\;1/2}}{\delta_2^{1/2}} = \frac{\bar{x}_1}{\bar{u}_1}\,\frac{d_2^{1/2}}{d_1^{1/2}}\,\frac{\delta_1^{1/2}}{\delta_2^{1/2}} = \frac{\bar{x}_1}{\bar{u}_1}. \tag{3.133}$$

Thus, we can write (3.128) as

$$\bar{G} = \begin{bmatrix} 1 & -\frac{\bar{x}_1}{\bar{u}_1} \\ \frac{\delta_2\bar{x}_1}{d_2\bar{u}_1} & 1 \end{bmatrix}. \tag{3.134}$$

From (3.130), we have

$$d_2 = d_1\cos^2\theta_1, \text{ and} \tag{3.135}$$

$$\delta_2 = \delta_1\cos^2\theta_1. \tag{3.136}$$

From (3.130) and (3.131), it appears that

$$\frac{\sin^2\theta_1}{\cos^2\theta_1} = \frac{\delta_1\bar{x}_1^2}{d_1\bar{u}_1^2}, \tag{3.137}$$

such that $\cos^2\theta_1$ can be obtained with

$$\cos^2\theta_1 = (1+\sin^2\theta_1/\cos^2\theta_1)^{-1} = (1+\frac{\delta_1\bar{x}_1^2}{d_1\bar{u}_1^2})^{-1}. \tag{3.138}$$

As a result, the update formula in (3.127) with $\bar{G}$ as in (3.134) and their elements computed with (3.138, 3.135, 3.136) shall avoid the use of SQRT.

Appendix 3 - Pseudo-Codes

In the following, we present the pseudo-codes for the conventional and the inverse QRD-RLS algorithms. In both cases, we present their complex versions employing the Cholesky vector with lower triangular matrices.

Table 3.5 Pseudo-code for the conventional QRD-RLS algorithm.

QRD-RLS

```
% Initialization:
  $N$ (filter order),  $\delta$ (small constant),  $\lambda$ (forgetting factor)
  $\mathbf{U}(k-1) = \delta \mathbf{J}_{N+1}$; ($\mathbf{J}_{N+1}$ being the reversal matrix)
  $\mathbf{w}(k) = 0_{(N+1)\times 1}$; (if necessary)  $\mathbf{d}_{q2}(k-1) = 0_{(N+1)\times 1}$;
for $k = 1, 2, \ldots$
{  xaux $= \mathbf{x}^{\mathrm{H}}(k)$;            % xaux$(n) = x^*(k-n-1)$ for $n = 1 : N+1$
  Uaux $= \lambda^{1/2}\mathbf{U}(k-1)$;       % Uaux(n,m) = $\lambda^{1/2}[\mathbf{U}(k-1)]_{n,m}$ for $n,m = 1 : N+1$
  daux $= d^*(k)$;
  dq2aux $= \lambda^{1/2}\mathbf{d}_{q2}(k-1)$;
  gamma $= 1$;
  for $n = 1 : N+1$
  {  % Obtaining $\mathbf{Q}_\theta(k)$ and updating $\mathbf{U}(k)$:
     $cos\theta_{n-1}(k) = \frac{|\text{Uaux}(N+2-n,n)|}{\sqrt{|\text{xaux}(n)|^2 + |\text{Uaux}(N+2-n,n)|^2}}$;
     $sin\theta_{n-1}(k) = \left(\frac{\text{xaux}(n)}{\text{Uaux}(N+2-n,n)}\right)^* cos\theta_{n-1}(k)$;
     xaux$(n) = 0$;
     Uaux$(N+2-n,n) = sin\theta_{n-1}(k)$xaux$(n) + cos\theta_{n-1}(k)$Uaux$(N+2-n,n)$;
     for $m = n+1 : N+1$
     {  oldxaux = xaux$(m)$;
        xaux$(m) = cos\theta_{n-1}(k)$oldxaux $- sin^*\theta_{n-1}(k)$Uaux$(N+2-n,m)$;
        Uaux$(N+2-n,m)$= $sin\theta_{n-1}(k)$oldxaux $+ cos\theta_{n-1}(k)$Uaux$(N+2-n,m)$;
     }
     % Obtaining $\gamma(k)$:
     gamma = gamma $cos\theta_{n-1}(k)$;
     % Obtaining $eq_1(k)$ and updating $\mathbf{d}_{q2}(k)$:
     olddaux = daux;
     daux $= cos\theta_{n-1}(k)$.olddaux $- sin^*\theta_{n-1}(k)$dq2aux$(N+2-n)$;
     dq2aux$(N+2-n) = sin\theta_{n-1}(k)$olddaux $+ cos\theta_{n-1}(k)$dq2aux$(N+2-n)$;
     % Back-substitution, if necessary:
     summa= 0;
     for $m = 1 : (n-1)$
     {  summa = summa+Uaux$(n, N+2-m)[\mathbf{w}(k)]_{N+2-n}$;
     }
     $[\mathbf{w}(k)]_{N+2-n}$ = (dq2aux$(n)$-summa)/Uaux$(n, N+2-n)$;
  }
  $\mathbf{U}(k) =$ Uaux;
  $\gamma(k) =$ gamma;
  $\mathbf{d}_{q2}(k) =$ dq2aux;
  $e_{q1}(k) =$ daux;
  % Obtaining the estimation errors:
  $\varepsilon(k) = e^*_{q1}(k)\gamma(k)$;     % a posteriori error, i.e, $d(k) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k)$
  $e(k) = e^*_{q1}(k)/\gamma(k)$;     % a priori error, i.e, $d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
}
```

Table 3.6 Pseudo-code for the inverse QRD-RLS algorithm.

IQRD-RLS

```
% Initialization:
  $N$ (filter order),  $\delta$ (small constant),  $\lambda$ (forgetting factor)
  $\mathbf{U}^{\mathrm{H}}(k-1) = \frac{1}{\delta}\mathbf{J}_{N+1}$;  ($\mathbf{J}_{N+1}$ being the reversal matrix)
  $\mathbf{w}(k) = 0_{(N+1)\times 1}$;  $\mathbf{d}_{q2}(k-1) = 0_{(N+1)\times 1}$;
for $k = 1, 2, \ldots$
{  % Obtaining $\mathbf{a}(k)$:
  akaux $= 0_{(N+1)\times 1}$;
  xaux $= \lambda^{-1/2}\mathbf{x}(k)$;
  for $n = 1 : N+1$
    for $m = 1 : (N+2-n)$
    { akaux($n$) = akaux($n$)+ $[\mathbf{U}^{-\mathrm{H}}(k-1)]_{n,m}$xaux($m$);
    }
  }
  $\mathbf{a}(k) =$ akaux;
  % Obtaining $\mathbf{Q}_{\theta}(k)$ and $\gamma(k)$:
  igamma $= 1$;
  for $n = 1 : N+1$
    aux1 $= \sqrt{|\text{igamma}|^2 + |\text{akaux}(N+2-n)|^2}$;
    $cos\theta_{n-1}(k) = \frac{|\text{igamma}|}{\text{aux1}}$;
    $sin\theta_{n-1}(k) = \frac{\text{akaux}(N+2-n)}{\text{igamma}} cos\theta_{n-1}(k)$;
    igamma = aux1;    % or $cos\theta_{n-1}(k)$igamma$+ sin^*\theta_{n-1}(k)$akaux($N+2-m$);
  }
  $\gamma(k) = 1/$igamma;
  % Obtaining $\mathbf{u}(k)$ and updating $\mathbf{U}^{-\mathrm{H}}(k)$:
  uHaux $= 0_{(N+1)\times 1}$;
  UmHaux $= \lambda^{-1/2}\mathbf{U}^{-\mathrm{H}}(k-1)$;
  for $n = 1 : N+1$
    for $m = 1 : n$
    { aux2 = uHaux($m$);
      uHaux($m$) $= cos\theta_{n-1}(k)$aux2$- sin^*\theta_{n-1}(k)$UmHaux($N+2-n, m$);
      UmHaux($N+2-n, m$) $= sin\theta_{n-1}(k)$aux2$+ \cdots$
                                              $\cdots + cos\theta_{n-1}(k)$UmHaux($N+2-n, m$);
    }
  }
  $\mathbf{u}(k) =$ uHaux$^*$;
  $\mathbf{U}^{-\mathrm{H}}(k) =$ UmHaux;
  % Obtaining $e(k)$:
  $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$;   % a priori error
  % Updating the coefficient vector:
  $\mathbf{w}(k) = \mathbf{w}(k-1) - \gamma(k)e^*(k)\mathbf{u}(k)$;
}
```

In order to provide a better understanding of a Givens rotation matrix, for the case of a complex vector, we give a simple example: consider vector $\mathbf{z} = [a\ b]^{\mathrm{T}}$ where a and b are complex numbers and we want to rotate this vector by pre-multiplying matrix $\mathbf{Q}_\theta$ such that the resulting vector has the same norm but one element was annihilated. This can be carried out as follows.

1. $\mathbf{Q}_\theta \mathbf{z} = \begin{bmatrix} \cos\theta & -\sin^*\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$

   For this case, the values of the cosine and the sine of θ are given by

   $$\begin{cases} \cos\theta = \frac{|a|}{\sqrt{|a|^2+|b|^2}}, \text{ and} \\ \sin\theta = \left(-\frac{b}{a}\right)^* cos\theta. \end{cases}$$

   This definition was used in the conventional QRD-RLS algorithm in order to obtain $\mathbf{Q}_\theta$.

2. $\mathbf{Q}_\theta \mathbf{z} = \begin{bmatrix} \cos\theta & -\sin^*\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$

   For this second case, the values of the cosine and the sine of θ are given by

   $$\begin{cases} \cos\theta = \frac{|b|}{\sqrt{|a|^2+|b|^2}}, \text{ and} \\ \sin\theta = \left(-\frac{a}{b}\right)^* cos\theta. \end{cases}$$

   These expressions were used in inverse QRD-RLS algorithm to obtain $\mathbf{Q}_\theta$.

In both cases, α has the same norm of $\mathbf{z}$ and may be expressed as $\pm e^{j\phi}\sqrt{|a|^2+|b|^2}$ where ϕ is the phase of a (first case) or b (second case). $\mathbf{Q}_\theta$ can also be chosen slightly different in order to compensate this phase and produce a real number, the norm of z, instead of $\pm e^{j\phi}||\mathbf{z}||$.

A pseudo-code for the conventional QRD-RLS algorithm is presented in Table 3.5 while a pseudo-code for the inverse QRD-RLS algorithm is presented in Table 3.6, both employing lower triangular Cholesky factors.

References

1. S. Haykin, Adaptive Filter Theory. 4th edition Prentice-Hall, Englewood Cliffs, NJ, USA (2002)
2. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
3. G. H. Golub and C. F. Van Loan, Matrix Computations. 2nd edition John Hopkins University Press, Baltimore, MD, USA (1989)
4. R. A. Horn and C. R. Johnson, Matrix Analysis. Cambridge University Press, New York, NY, USA (1986)

5. W. Givens, Computation of plane unitary rotations transforming a general matrix to triangular form. Journal of the Society for Industrial and Applied Mathematics, vol. 6, no. 1, pp. 26–50 (March 1958)
6. W. H. Gentleman and H. T. Kung, Matrix triangularization by systolic arrays. SPIE Real-Time Signal Processing IV, vol. 298, pp. 19–26 (January 1981)
7. J. G. McWhirter, Recursive least-squares minimization using a systolic array. SPIE Real-Time Signal Processing VI, vol. 431, pp. 105–112 (January 1983)
8. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 879–891 (April 1991)
9. N. E. Hubing and S. T. Alexander, Statistical analysis of initialization methods for RLS adaptive filters. IEEE Transactions on Signal Processing, vol. 39, no. 8, pp. 1793–1804 (August 1991)
10. P. S. R. Diniz and M. G. Siqueira, Fixed-point error analysis of the QR-recursive least square algorithm. IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing, vol. 42, no. 5, pp. 334–348 (May 1995)
11. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)
12. J. E. Hudson and T. J. Shepherd, Parallel weight extraction by a systolic least squares algorithm. SPIE Advanced Algorithms and Architectures for Signal Processing IV, vol. 1152, pp. 68–77 (August 1989)
13. S. Ljung and L. Ljung, Error propagation properties of recursive least-squares adaptation algorithms. Automatica, vol. 21, no. 2, pp. 157–167 (March 1985)
14. M. D. Miranda and M. Gerken, Hybrid least squares QR-lattice algorithm using a priori errors. IEEE Transactions on Signal Processing, vol. 45, no. 12, pp. 2900–2911 (December 1997)
15. G. Strang, Computational Science and Engineering. Wellesly-Cambridge Press, Wellesley, MA, USA (2007)
16. B. Haller, J. Götze and J. R. Cavallaro, Efficient implementation of rotation operations for high performance QRD-RLS filtering. IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP'97, Zurich, Switzerland, pp. 162–174 (July 1997)
17. J. E. Volder, The CORDIC Trigonometric Computing Technique. IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334 (September 1959)
18. W. M. Gentleman, Least squares computations by Givens transformations without square roots. IMA Journal of Applied Mathematics, vol. 12, no. 3, pp. 329–336 (December 1973)
19. S. Hammarling, A note on modifications to the Givens plane rotation. IMA Journal of Applied Mathematics vol. 13, no. 2, pp. 215–218 (April 1974)
20. J. L. Barlow and I. C. F. Ipsen, Scaled Givens rotations for the solution of linear least squares problems on systolic arrays. SIAM Journal on Scientific and Statistical Computing, vol. 8, no. 5, pp. 716–733 (September 1987)
21. J. Götze and U. Schwiegelshohn, A square root and division free Givens rotation for solving least squares problems on systolic arrays. SIAM Journal on Scientific and Statistical Computing, vol. 12, no. 4, pp. 800–807 (July 1991)
22. E. N. Frantzeskakis and K. J. R. Liu, A class of square root and division free algorithms and architectures for QRD-based adaptive signal processing. IEEE Transactions on Signal Processing, vol. 42, no. 9, pp. 2455–2469 (September 1994)
23. S. F. Hsieh, K. J. R. Liu, and K. Yao, A unified square-root-free approach for QRD-based recursive-least-squares estimation. IEEE Transactions on Signal Processing, vol. 41, no. 3, pp. 1405–1409 (March 1993)

Chapter 4
Fast QRD-RLS Algorithms

José A. Apolinário Jr. and Paulo S. R. Diniz

Abstract Although numerically robust, the QR-decomposition recursive least-squares (QRD-RLS) algorithms studied in the previous chapter are computationally intensive, requiring a number of mathematical operations in the order of N^2, or $\mathscr{O}[N^2]$, N being the order of the adaptive filter. This chapter describes the so-called *fast* QRD-RLS algorithms, i.e., those computationally efficient algorithms that, besides keeping the attractive numerical robustness of the family, benefit from the fact that the input signal is a delay line, reducing the complexity to $\mathscr{O}[N]$. The fast versions of the QRD-RLS algorithms using real variables are classified and derived. For each algorithm, we present the final set of equations as well as their pseudo-codes in tables. For the main algorithms, their descriptions are given utilizing complex variables.

4.1 Introduction

Usually the choice of a given adaptive filtering algorithm for an application relies on a number of properties such as speed of convergence, steady-state behavior in stationary environments, and tracking capability in non-stationary environments. However, very often the algorithm choice is highly correlated to its computational complexity. In the case of the recursive least-squares(RLS) family of algorithms, their distinctive features are behavior in finite wordlength implementations and computational burden.

José A. Apolinário Jr.
Military Institute of Engineering (IME), Rio de Janeiro – Brazil
e-mail: apolin@ieee.org

Paulo S. R. Diniz
Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro – Brazil
e-mail: diniz@lps.ufrj.br

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_4, © Springer Science+Business Media, LLC 2009

In case the application at hand has some structure related to the input signal data vector, such as being composed by a tapped delay line, it is possible to derive a number of algorithms with lower computational complexity collectively known as *fast* algorithms.

From the conventional ($\mathscr{O}[N^2]$) QR decomposition method [1, 2], a number of *fast* algorithms ($\mathscr{O}[N]$) has been derived [3, 4]. 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. As it can be verified from the Gram–Schmidt orthogonalization procedure, an *upper triangularization* of the data correlation matrix Cholesky factor (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 chapter presents a classification of a family of fast QRD-RLS algorithms along with their corresponding equations. Specifically, Table 4.1 shows how the algorithms discussed here will be designated hereafter as well as the classification of the type of prediction problem the algorithms rely.

Table 4.1 Classification of the fast QRD-RLS algorithms.

Error type	Prediction	
	Forward	Backward
A posteriori	FQR_POS_F [3]	FQR_POS_B [5, 6]
A priori	FQR_PRI_F [7]	FQR_PRI_B [4, 8]

It is worth mentioning that the FQR_PRI_B algorithm was independently developed in [8] and in [4] using distinct approaches and leading to two different versions. The approach which will be used here [4] was derived from concepts used in the inverse QRD-RLS algorithm [9]. The same algorithm (FQR_PRI_B) was also derived in [10] as a lattice extension of the inverse QRD-RLS algorithm [11].

In the derivation of fast QRD-RLS algorithms, we start by applying the QR decomposition to the backward and forward prediction problems whose prediction errors were defined in the previous chapter. We aim the triangularization of the extended order input data matrix $\mathbf{X}^{(N+2)}(k)$, from the expressions involving backward and forward predictions, in order to obtain $\mathbf{Q}^{(N+2)}(k)$, such that

$$\mathbf{Q}^{(N+2)}(k)\mathbf{X}^{(N+2)}(k) = \begin{bmatrix} \mathbf{0} \\ \mathbf{U}^{(N+2)}(k) \end{bmatrix}, \tag{4.1}$$

where the null matrix $\mathbf{0}$ above has dimension $(k-N-1)\times(N+2)$.

4.2 Upper Triangularization Algorithms (Updating Forward Prediction Errors)

The first algorithms derived here are those based on forward prediction errors, namely the FQR_POS_F [3] and the FQR_PRI_F [7] algorithms. These algorithms are presented here for completeness, due to their historical importance, and to pave the way for the next section which deals with a more attractive class of algorithm in terms of complexity and stability. As such, we suggest to a reader more interested in practical implementation to skip this section, and focus on the algorithms based on the updating of backward prediction errors.

Let us start, from the definition of the weighted backward prediction error vector $\mathbf{e}_b(k) = \mathbf{d}_b(k) - \mathbf{X}(k)\mathbf{w}_b(k)$, by pre-multiplying the weighted backward desired vector $\mathbf{d}_b(k) = \begin{bmatrix} x(k-N-1) & \cdots & \lambda^{(k-N-1)/2}x(0) & \mathbf{0}^{\mathrm{T}}_{N+1}\end{bmatrix}^{\mathrm{T}}$ by $\mathbf{Q}(k)$ and use the recursive structure of $\mathbf{Q}(k)$.[1] As a result, two important relations follow.

$$\| \mathbf{e}_b(k) \|^2 = e^2_{bq_1}(k) + \lambda \| \mathbf{e}_b(k-1) \|^2, \text{ and} \tag{4.2}$$

$$\begin{bmatrix} e_{bq_1}(k) \\ \mathbf{d}_{bq_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d_b(k) \\ \lambda^{1/2}\mathbf{d}_{bq_2}(k-1) \end{bmatrix}, \tag{4.3}$$

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

The (upper) triangularization, as seen in (4.1), of $\mathbf{X}^{(N+2)}(k)$, as defined in the previous chapter for the backward prediction problem, is achieved using three distinct matrices: $\mathbf{Q}^{(N+2)}(k) = \mathbf{Q}'_b(k)\mathbf{Q}_b(k)\mathbf{Q}(k)$, where $\mathbf{Q}_b(k)$ and $\mathbf{Q}'_b(k)$ are two sets of Givens rotations applied to generate, respectively, $\| \mathbf{e}_b(k) \|$ and $\| \mathbf{e}^{(0)}_b(k) \|$. The latter quantities are defined in the sequel. As a result, we have

$$\begin{aligned} \mathbf{U}^{(N+2)}(k) &= \mathbf{Q}'_{\theta b}(k) \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \| \mathbf{e}_b(k) \| \\ \mathbf{U}(k) & \mathbf{d}_{bq_2}(k) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k) & \| \mathbf{e}^{(0)}_b(k) \| \\ \mathbf{R}(k) & \mathbf{0} \end{bmatrix}, \end{aligned} \tag{4.4}$$

where $\mathbf{Q}'_{\theta b}(k)$ is a submatrix of $\mathbf{Q}'_b(k)$, $[\mathbf{z}(k)\mathbf{R}^{\mathrm{T}}(k)]^{\mathrm{T}}$ is the left part of $\mathbf{U}^{(N+2)}(k)$, just excluding the last column, and $\| \mathbf{e}^{(0)}_b(k) \|$ is the norm of the backward error of a predictor whose number of coefficients is zero.

In the forward prediction problem, the pre-multiplication of the forward weighted desired vector, $\mathbf{d}_f(k) = \begin{bmatrix} x(k) & \cdots & \lambda^{k/2}x(0)\end{bmatrix}^{\mathrm{T}}$, by $\begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$ and the use of the recurse expression of $\mathbf{Q}(k)$ in the weighted forward error vector $\mathbf{e}_f(k) = \mathbf{d}_f(k) -$

[1] The recursive structure of $\mathbf{Q}(k)$ is expressed by $\mathbf{Q}(k) = \tilde{\mathbf{Q}}(k) \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{Q}(k-1) \end{bmatrix}$.

$\begin{bmatrix} \mathbf{X}(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{w}_f(k)$, leads to two other important relations given by

$$\| \mathbf{e}_f(k) \|^2 = e^2_{fq_1}(k) + \lambda \| \mathbf{e}_f(k-1) \|^2, \text{ and} \tag{4.5}$$

$$\begin{bmatrix} e_{fq_1}(k) \\ \mathbf{d}_{fq_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k-1) \begin{bmatrix} d_f(k) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k-1) \end{bmatrix}, \tag{4.6}$$

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

The upper triangularization of $\mathbf{U}^{(N+2)}(k)$ in the forward prediction problem is implemented by pre-multiplying $\mathbf{e}_f(k)$ by the product $\mathbf{Q}_f(k) \begin{bmatrix} \mathbf{Q}(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$, where $\mathbf{Q}_f(k)$ is a set of Givens rotations generating $\| \mathbf{e}_f(k) \|$ by eliminating the first $k-N$ elements of the rotated desired vector of the forward predictor. The result is

$$\mathbf{U}^{(N+2)}(k) = \begin{bmatrix} \mathbf{d}_{fq_2}(k) & \mathbf{U}(k-1) \\ \| \mathbf{e}_f(k) \| & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{4.7}$$

In order to avoid working with matrices of increasing dimensions, it is possible to eliminate the identity matrices that are part of the rotation matrices and are the source of the dimension increase. By eliminating these internal identity matrices, one can show that [2]

$$\mathbf{Q}_\theta^{(N+2)}(k) = \mathbf{Q}_{\theta f}(k) \begin{bmatrix} \mathbf{Q}_\theta(k-1) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}, \tag{4.8}$$

where $\mathbf{Q}_{\theta f}(k)$ is a single Givens rotation generating $\| \mathbf{e}_f(k) \|$ as in (4.5).

If we take the inverses of (4.4) and (4.7), the results are

$$\begin{aligned}[\mathbf{U}^{(N+2)}(k)]^{-1} &= \begin{bmatrix} \mathbf{0} & \mathbf{R}^{-1}(k) \\ \frac{1}{\|\mathbf{e}_b^{(0)}(k)\|} & \frac{-\mathbf{z}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)}{\|\mathbf{e}_b^{(0)}(k)\|} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \frac{1}{\|\mathbf{e}_f(k)\|} \\ \mathbf{U}^{-1}(k-1) & \frac{-\mathbf{U}^{-1}(k-1)\mathbf{d}_{fq_2}(k)}{\|\mathbf{e}_f(k)\|} \end{bmatrix}. \end{aligned} \tag{4.9}$$

We can use the expressions of $[\mathbf{U}^{(N+2)}(k)]^{-1}$ given in (4.9) to obtain the vectors $\mathbf{f}^{(N+2)}(k+1)$ and $\mathbf{a}^{(N+2)}(k+1)$. The choices of these vectors generate distinct algorithms, that is, updating $\mathbf{f}(k)$ (*a posteriori* forward errors) leads to the FQR_POS_F algorithm [3] whereas updating $\mathbf{a}(k)$ (*a priori* forward errors) leads to the FQR_PRI_F algorithm [7].

4.2.1 The FQR_POS_F algorithm

In the FQR_POS_F algorithm, vector $\mathbf{f}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k+1)]^{-\mathrm{T}}$ $\mathbf{x}^{(N+2)}(k+1)$ is expressed in terms of the relations obtained in the forward and backward prediction problems. We shall first use the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ in (4.9) that comes from the backward prediction evaluated at instant $k+1$ to calculate $\mathbf{f}^{(N+2)}(k+1)$. In this case, we replace $\mathbf{x}^{(N+2)}(k+1)$ by $[\mathbf{x}^{\mathrm{T}}(k+1)\ x(k-N)]^{\mathrm{T}}$ and then pre-multiply the result by $\mathbf{Q}'^{\mathrm{T}}_{\theta b}(k+1)$. The outcome, after some algebraic manipulations (using Equation (4.4) to help with the simplification of the expression), is

$$\mathbf{f}^{(N+2)}(k+1) = \mathbf{Q}'_{\theta b}(k+1)\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix}. \tag{4.10}$$

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

$$\mathbf{f}^{(N+2)}(k+1) = \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}. \tag{4.11}$$

By combining (4.10) and (4.11), it is possible to derive an expression to update $\mathbf{f}(k)$, which is given by

$$\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1)] \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k+1)\begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}. \tag{4.12}$$

Once we have $\mathbf{f}(k+1)$, we can extract the angles of $\mathbf{Q}_\theta(k+1)$ by post-multiplying this matrix by the pinning vector $[1\,0\,\cdots\,0]^{\mathrm{T}}$. From the partitioned expression of $\mathbf{Q}_\theta(k)$, we can see that the result is

$$\mathbf{Q}_\theta(k+1)\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}. \tag{4.13}$$

However, the quantities required to compute the angles of $\mathbf{Q}'_{\theta_b}(k+1)$ are not available at instant k so that a special strategy is required. The updated $\mathbf{Q}'_{\theta_b}(k+1)$ is obtained [2, 12] with the use of vector $\mathbf{c}(k+1)$ defined as

$$\begin{aligned}\mathbf{c}(k+1) &= \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k)\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \\ &= \mathbf{Q}'_{\theta b}(k+1)\begin{bmatrix} b \\ \mathbf{0} \end{bmatrix}. \end{aligned} \tag{4.14}$$

The submatrix $\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)$ consisting of the last $(N+2)\times(N+2)$ elements of $\mathbf{Q}_\theta^{(N+2)}(k+1)$ is available from (4.8) (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 the same expressions previously used for the conventional QRD-RLS algorithm. The FQR_POS_F equations are presented in Table 4.2. A detailed description of this algorithm is found in Appendix 1.

Table 4.2 The FQRD_POS_F equations.

```
FQR_POS_F
for each $k$
{  Obtaining $e_{f_{q_1}}(k+1)$:
   $\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$
   Obtaining $\mathbf{Q}_{\theta f}(k+1)$:
   $\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$
   $cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| \, / \, \| \mathbf{e}_f(k+1) \|$
   $sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$
   Obtaining $\mathbf{c}(k+1)$:
   $\mathbf{Q}_\theta^{(N+2)}(k+1) = \mathbf{Q}_{\theta f}(k+1) \begin{bmatrix} \mathbf{Q}_\theta(k) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$
   $\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)$ = last $(N+2) \times (N+2)$ elements of $\mathbf{Q}_\theta^{(N+2)}(k+1)$
   $\mathbf{c}(k+1) = \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}$
   Obtaining $\mathbf{Q}'_{\theta b}(k+1)$:
   $\begin{bmatrix} b \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k+1)\mathbf{c}(k+1)$
   Obtaining $\mathbf{f}(k+1)$:
   $\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}$
   Obtaining $\mathbf{Q}_\theta(k+1)$:
   $\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k+1) \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}$
   Joint Process Estimation:
   $\begin{bmatrix} e_{q_1}(k+1) \\ \mathbf{d}_{q_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k) \end{bmatrix}$
   $e(k+1) = e_{q_1}(k+1)/\gamma(k+1)$      % a priori error
   $\varepsilon(k+1) = e_{q_1}(k+1)\gamma(k+1)$      % a posteriori error
}
```

4.2.2 The FQR_PRI_F algorithm

Expressing $\mathbf{a}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k)]^{-\mathrm{T}}\mathbf{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the matrices in (4.9) and pre-multiplying the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ originated from the backward prediction problem by $\mathbf{Q}'_{\theta b}(k)\mathbf{Q}'^{\mathrm{T}}_{\theta b}(k)$ yields

$$\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}. \tag{4.15}$$

Given $\mathbf{a}(k+1)$, the angles of $\mathbf{Q}_\theta(k+1)$ are found through the following relation obtained by post-multiplying $\mathbf{Q}_\theta^{\mathrm{T}}(k+1)$ by the pinning vector.

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix} \tag{4.16}$$

By noting that the angles of $\mathbf{Q}'_{\theta_b}(k+1)$ can be updated with the same procedure used in the FQR_POS_F algorithm, we already have all the necessary equations of the fast QR-RLS algorithm as presented in Table 4.3. The detailed description of this algorithm is also found in Appendix 1.

Table 4.3 The FQRD_PRI_F equations.

FQR_PRI_F

```
for each $k$
{  Obtaining $e_f(k+1)$:
    $\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$
   $e_f(k+1) = e_{fq_1}(k+1)/\gamma(k)$
    Obtaining $\mathbf{a}(k+1)$:
    $\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}$
    Obtaining $\mathbf{Q}_{\theta f}(k+1)$:
   $\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$
   $cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$
   $sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$
    Obtaining $\mathbf{c}(k+1)$:
   $\mathbf{Q}_\theta^{(N+2)}(k+1) = \mathbf{Q}_{\theta f}(k+1) \begin{bmatrix} \mathbf{Q}_\theta(k) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}$
   $\hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)$ = last $(N+2) \times (N+2)$ elements of $\mathbf{Q}_\theta^{(N+2)}(k+1)$
   $\mathbf{c}(k+1) = \hat{\mathbf{Q}}_\theta^{(N+2)}(k+1)\mathbf{Q}'_{\theta b}(k) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}$
    Obtaining $\mathbf{Q}'_{\theta b}(k+1)$:
    $\begin{bmatrix} b \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}'_{\theta b}{}^{\mathrm{T}}(k+1)\mathbf{c}(k+1)$
    Obtaining $\mathbf{Q}_\theta(k+1)$:
    $\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix}$
    Joint Process Estimation:
    $\begin{bmatrix} e_{q_1}(k+1) \\ \mathbf{d}_{q_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k) \end{bmatrix}$
   $e(k+1) = e_{q_1}(k+1)/\gamma(k+1)$      % a priori error
   $\varepsilon(k+1) = e_{q_1}(k+1)\gamma(k+1)$      % a posteriori error
}
```

4.3 Lower Triangularization Algorithms (Updating Backward Prediction Errors)

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

Before presenting the derivations, we remark that the fast QRD-RLS 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 [5, 11].

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

$$\mathbf{U}^{(N+2)}(k) = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \| \mathbf{e}_b(k) \| \\ \mathbf{U}(k) & \mathbf{d}_{bq_2}(k) \end{bmatrix}. \tag{4.17}$$

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

$$\mathbf{U}^{(N+2)}(k) = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq_2}(k) & \mathbf{U}(k-1) \\ \| \mathbf{e}_f(k) \| & \mathbf{0}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{R}(k) \\ \| \mathbf{e}_f^{(0)}(k) \| & \mathbf{z}^{\mathrm{T}}(k) \end{bmatrix}, \tag{4.18}$$

where $[\mathbf{R}^{\mathrm{T}}(k)\ \mathbf{z}(k)]^{\mathrm{T}}$ represents the last $(N+1)$ columns of $\mathbf{U}^{(N+2)}(k)$. By considering the fact that (4.2), (4.3), (4.5), and (4.6) hold, $\| \mathbf{e}_f(k) \|$ can be recursively computed using (4.5).

Taking the inverse of (4.17) and (4.18), we have the following results:

$$\begin{aligned} [\mathbf{U}^{(N+2)}(k)]^{-1} &= \begin{bmatrix} \frac{-\mathbf{U}^{-1}(k)\mathbf{d}_{bq_2}(k)}{\|\mathbf{e}_b(k)\|} & \mathbf{U}^{-1}(k) \\ \frac{1}{\|\mathbf{e}_b(k)\|} & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{-\mathbf{z}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)}{\|\mathbf{e}_f^{(0)}(k)\|} & \frac{1}{\|\mathbf{e}_f^{(0)}(k)\|} \\ \mathbf{R}^{-1}(k) & \mathbf{0} \end{bmatrix}. \end{aligned} \tag{4.19}$$

With the results obtained from (4.19), we can once more express vectors $\mathbf{f}^{(N+2)}(k+1)$ and $\mathbf{a}^{(N+2)}(k+1)$ in terms of the submatrices of $\left[\mathbf{U}^{(N+2)}(k+1)\right]^{-1}$. If we update $\mathbf{f}(k)$, the resulting algorithm is the FQR_POS_B whereas, by updating $\mathbf{a}(k)$, one generates the FQR_PRI_B algorithm.

4.3.1 The FQR_POS_B algorithm

Expressing $\mathbf{f}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k+1)]^{-\mathrm{T}}\mathbf{x}^{(N+2)}(k+1)$ in terms of the matrix in (4.19) originating from the the forward prediction problem, and pre-multiplying the resulting expression by $\mathbf{Q}'_{\theta f}(k+1)\mathbf{Q}'^{\mathrm{T}}_{\theta f}(k+1)$ yields

$$\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta_f}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}. \tag{4.20}$$

For this particular algorithm, we provide extra implementation details which are similar to the other fast QRD-RLS algorithms. We start by pointing out that matrix $\mathbf{Q}'_{\theta_f}(k)$ in (4.20) corresponds to the set of rotations used in (4.18) to eliminate $\mathbf{d}_{fq_2}(k)$ over $||\mathbf{e}_f(k)||$ such that the resulting matrix $\mathbf{U}^{(N+2)}(k)$ is lower triangular.

Therefore, making explicit the structure of $\mathbf{Q}'_{\theta_f}(k)$, the part of (4.18) given by the product $\mathbf{Q}'_{\theta_f}(k)\left[\mathbf{d}^{\mathrm{T}}_{fq_2}(k) \;\; \| \mathbf{e}_f(k) \|\right]^{\mathrm{T}}$ can be expressed as

$$\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}_f^{(0)}(k) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{d}_{fq_2}(k) \\ \| \mathbf{e}_f(k) \| \end{bmatrix} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_N}(k) & -\sin\theta'_{f_N}(k) \\ \mathbf{0}^{\mathrm{T}} & \sin\theta'_{f_N}(k) & \cos\theta'_{f_N}(k) \end{bmatrix} \cdots$$
$$\cdots \underbrace{\begin{bmatrix} \cos\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & -\sin\theta'_{f_0}(k) \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ \sin\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_0}(k) \end{bmatrix} \begin{bmatrix} d_{fq2_1}(k) \\ \vdots \\ d_{fq2_{N+1}}(k) \\ \| \mathbf{e}_f^{(N+1)}(k) \| \end{bmatrix}}_{\begin{bmatrix} 0 \\ \vdots \\ d_{fq2_{N+1}}(k) \\ \| \mathbf{e}_f^{(N)}(k) \| \end{bmatrix}}, \tag{4.21}$$

where, for this first multiplication, the quantity $\cos\theta'_{f_0}(k)d_{fq2_1}(k) - \sin\theta'_{f_0}(k) \parallel \mathbf{e}_f^{(N+1)}(k) \parallel$ was made zero and the quantity $\sin\theta'_{f_0}(k)d_{fq2_1}(k) + \cos\theta'_{f_0}(k) \parallel \mathbf{e}_f^{(N+1)}(k) \parallel$ corresponds to $\parallel \mathbf{e}_f^{(N)}(k) \parallel$.

In the ith multiplication, for i ranging from 1 to $N+1$, we have:

$$\begin{cases} \cos\theta'_{f_{i-1}}(k)d_{fq2_i}(k) = \sin\theta'_{f_{i-1}}(k) \parallel \mathbf{e}_f^{(N+2-i)}(k) \parallel \\ \parallel \mathbf{e}_f^{(N+1-i)}(k) \parallel = \sin\theta'_{f_{i-1}}(k)d_{fq2_i}(k) + \cos\theta'_{f_{i-1}}(k) \parallel \mathbf{e}_f^{(N+2-i)}(k) \parallel \end{cases} \tag{4.22}$$

The two expressions in (4.22) are represented graphically in Figure 4.1. From this figure, one clearly observes that $\parallel \mathbf{e}_f^{(N+1-i)}(k) \parallel$ can also be computed as follows.

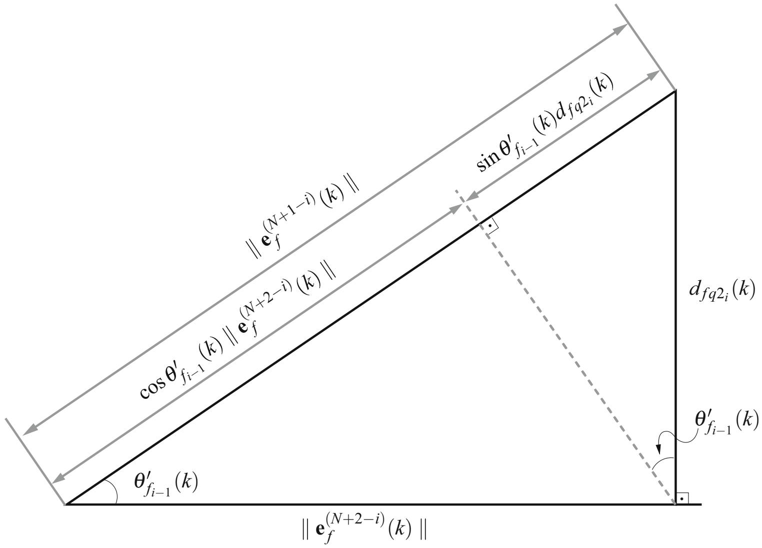

$\| \mathbf{e}_f^{(N+1-i)}(k) \|$
$\sin\theta'_{f_{i-1}}(k) d_{fq2_i}(k)$
$\cos\theta'_{f_{i-1}}(k) \| \mathbf{e}_f^{(N+2-i)}(k) \|$
$d_{fq2_i}(k)$
$\theta'_{f_{i-1}}(k)$
$\theta'_{f_{i-1}}(k)$
$\| \mathbf{e}_f^{(N+2-i)}(k) \|$

Fig. 4.1 Multiplication of $\mathbf{Q}'_{\theta f_i}(k)$: computing $\cos\theta'_{f_{i-1}}(k)$ and $\sin\theta'_{f_{i-1}}(k)$.

$$\| \mathbf{e}_f^{(N+1-i)}(k) \| = \sqrt{d_{fq2_i}^2(k) + \| \mathbf{e}_f^{(N+2-i)}(k) \|^2} \tag{4.23}$$

After computing $\| \mathbf{e}_f^{(N+1-i)}(k) \|$, the sine and the cosine are given below.

$$\begin{cases} \cos\theta'_{f_{i-1}}(k) = \frac{\|\mathbf{e}_f^{(N+2-i)}(k)\|}{\|\mathbf{e}_f^{(N+1-i)}(k)\|} \\ \sin\theta'_{f_{i-1}}(k) = \frac{d_{fq2_i}(k+1)}{\|\mathbf{e}_f^{(N+1-i)}(k)\|} \end{cases} \tag{4.24}$$

In the derivation of (4.20), it can be observed that the last element of $\mathbf{f}(k+1)$ is $\frac{x(k+1)}{\|\mathbf{e}_f^{(0)}(k+1)\|}$. The term $\frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|}$ can be calculated as $\gamma(k)\sin\theta_f(k+1)$ where $\sin\theta_f(k+1) = \frac{e_{fq_1}(k+1)}{\|\mathbf{e}_f(k+1)\|}$ is the sine of the angle of rotation matrix $\mathbf{Q}_f(k+1)$.

Using or not this information, the prior knowledge of the last element of $\mathbf{f}(k+1)$, leads to two distinct versions of the FQR_POS_B algorithm. The first version [6] uses this information as following discussed. Multiplying (4.20) by $\mathbf{Q'}_{\theta f}^{\mathrm{T}}$, we obtain the relation (4.25) with $\frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|}$ being represented by $f_0(k+1)$, $\frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|}$ by $f_{N+2}(K)$, and the ith element of $\mathbf{f}(k)$ by $f_i(k)$, for i ranging from 1 to $N+1$.

$$\begin{bmatrix} \mathbf{f}(k) \\ f_{N+2}(k) \end{bmatrix} = \begin{bmatrix} f_1(k) \\ \vdots \\ f_{N+1}(k) \\ f_{N+2}(k) \end{bmatrix} = \mathbf{Q}'^{\mathrm{T}}_{\theta f}(k) \begin{bmatrix} f_0(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & \sin\theta'_{f_0}(k) \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ -\sin\theta'_{f_0}(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_0}(k) \end{bmatrix} \cdots \mathbf{Q}'^{\mathrm{T}}_{\theta f_{N+1-i}}(k+1) \cdots$$
$$\cdots \underbrace{\begin{bmatrix} \mathbf{I}_N & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & \cos\theta'_{f_N}(k) & \sin\theta'_{f_N}(k) \\ \mathbf{0}^{\mathrm{T}} & -\sin\theta'_{f_N}(k) & \cos\theta'_{f_N}(k) \end{bmatrix} \begin{bmatrix} f_0(k+1) \\ \vdots \\ f_{N-1}(k+1) \\ f_N(k+1) \\ f_{N+1}(k+1) \end{bmatrix}}_{\begin{bmatrix} f_0(K+1) \\ \vdots \\ f_{N-1}(k+1) \\ f_{N+1}(k) \\ aux_1 \end{bmatrix}}, \tag{4.25}$$

In the last multiplication above, the results (as shown below the *underbrace*) denoted by $f_{N+1}(k)$ and aux_1 are such that, for the ith multiplication (with i from 1 to $N+1$), we have:

$$f_{N+2-i}(k) = \cos\theta'_{f_{N+1-i}}(k+1) f_{N+1-i}(k+1) - \sin\theta'_{f_{N+1-i}}(k+1) aux_{i-1} \tag{4.26}$$
$$aux_i = -\sin\theta'_{f_{N+1-i}}(k+1) f_{N+1-i}(k+1) + \cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1} \tag{4.27}$$

where $aux_0 = f_{N+1}(k+1)$ is the last element of $\mathbf{f}(k+1)$ which is known *a priori*.

From (4.26), we can easily obtain $f_{N+1-i}(k+1)$ as in Equation (4.28) and with this value compute aux_i in (4.27) such that $f_{N+2}(k) = aux_{N+1}$ (which is actually not used afterwards).

$$f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - \sin\theta'_{f_{N+1-i}}(k+1) aux_{i-1}}{\cos\theta'_{f_{N+1-i}}(k+1)} \tag{4.28}$$

In the second version of this algorithm [5], we compute first $\frac{e_f(k+1)}{\|\mathbf{e}_f(k+1)\|} = \frac{\gamma(k) e_{fq_1}(k+1)}{\|\mathbf{e}_f(k+1)\|}$ and then (4.20) is used in a straightforward manner.

Once we have $\mathbf{f}(k+1)$, we find $\mathbf{Q}_\theta(k+1)$ with the same relation used in the upper triangularization algorithms, (4.13). Moreover, the joint process estimation is carried out in the same manner as in the forward prediction-based algorithms.

Table 4.4 The FQR_POS_B algorithm.

```
FQR_POS_B
for each $k$
{  Obtaining $\mathbf{d}_{fq_2}(k+1)$:
   $\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x^*(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$
   Obtaining $\| \mathbf{e}_f(k+1) \|$:
   $\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$
   Obtaining $\mathbf{Q}'_{\theta f}(k+1)$:
   $\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}_f^{(0)}(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$
   Obtaining $\mathbf{f}(k+1)$:
   $\begin{bmatrix} \frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} \\ \mathbf{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{f}(k) \\ \frac{\varepsilon_f(k+1)}{\|\mathbf{e}_f(k+1)\|} \end{bmatrix}$
   Obtaining $\mathbf{Q}_\theta(k+1)$:
   $\begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{\mathrm{T}}(k+1) \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}(k+1) \end{bmatrix}$
   Joint Process Estimation:
   $\begin{bmatrix} e_{q_1}(k+1) \\ \mathbf{d}_{q_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d^*(k+1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k) \end{bmatrix}$
   $e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$      % *a priori* error
   $\varepsilon(k+1) = e^*_{q_1}(k+1)\gamma(k+1)$      % *a posteriori* error
}
```

As a result, we have now available all the required expressions composing the FQR_POS_B algorithm as presented in Table 4.4. The detailed descriptions of two different versions of this algorithm is found in Appendix 2.

4.3.2 The FQR_PRI_B algorithm

This last algorithm of this family is obtained by expressing vector $\mathbf{a}^{(N+2)}(k+1) = [\mathbf{U}^{(N+2)}(k)]^{-\mathrm{T}}\mathbf{x}^{(N+2)}(k+1)/\sqrt{\lambda}$ in terms of the expression for $[\mathbf{U}^{(N+2)}(k)]^{-1}$ in (4.19) originated from the forward prediction problem. Next, by pre-multiplying $\mathbf{a}^{(N+2)}(k+1)$ by $\mathbf{Q}'_{\theta f}(k)\mathbf{Q}'^{\mathrm{T}}_{\theta f}(k)$, the following updating equation results.

$$\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix} \tag{4.29}$$

It is again important to mention that the last element of $\mathbf{a}(k+1)$ in (4.29) is already known to be equal to $\frac{x(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f^{(0)}(k)\|}$. This fact leads to two different versions of the same algorithm: the same approach used in (4.20) for vector $\mathbf{f}(k)$ can be employed here for vector $\mathbf{a}(k)$.

Table 4.5 The FQR_PRI_B algorithm.

FQR_PRI_B

```
for each $k$
{ Obtaining $\mathbf{d}_{fq_2}(k+1)$:
  $\begin{bmatrix} e_{fq_1}(k+1) \\ \mathbf{d}_{fq_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} x^*(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq_2}(k) \end{bmatrix}$
  Obtaining $\mathbf{a}(k+1)$:
  $\begin{bmatrix} \frac{e_b(k+1)}{\sqrt{\lambda}\|\mathbf{e}_b(k)\|} \\ \mathbf{a}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}(k) \\ \frac{e_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|} \end{bmatrix}$
  Obtaining $\| \mathbf{e}_f(k+1) \|$:
  $\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$
  Obtaining $\mathbf{Q}'_{\theta f}(k+1)$:
  $\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}_f^{(0)}(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$
  Obtaining $\mathbf{Q}_\theta(k+1)$:
  $\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}(k+1) \end{bmatrix}$
  Joint Process Estimation:
  $\begin{bmatrix} e_{q_1}(k+1) \\ \mathbf{d}_{q_2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d^*(k+1) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k) \end{bmatrix}$
  $e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$      % a priori error
  $\varepsilon(k+1) = e^*_{q_1}(k+1)\gamma(k+1)$      % a posteriori error
}
```

Once more, if we have $\mathbf{a}(k+1)$, we can find $\mathbf{Q}_\theta(k+1)$ using (4.16) and the joint process estimation is carried out as seen in the previous chapter, i.e., updating $\mathbf{d}_{q_2}(k)$ from $d(k)$ and $\mathbf{d}_{q_2}(k-1)$ as well as computing the error (*a priori* or *a posteriori*) from the *rotated* error $e_{q_1}(k)$. The FQR_PRI_B equations are presented in Table 4.5. The detailed descriptions of the two versions of this algorithm is found in Appendix 2.

In terms of computational complexity, Table 4.6 shows the comparisons among the four fast QRD algorithms according to their detailed pseudo-codes in Appendices 1 and 2. Note that $p = N + 1$ is the number of coefficients.

Table 4.6 Comparison of computational complexity.

ALGORITHM	ADD	MULT.	DIV.	SQRT
FQR_POS_F [3]	10p+3	26p+10	3p+2	2p+1
FQR_PRI_F [7]	10p+3	26p+11	4p+4	2p+1
FQR_POS_B (VERSION 1) [6]	8p+1	19p+4	4p+1	2p+1
FQR_POS_B (VERSION 2) [5]	8p+1	20p+5	3p+1	2p+1
FQR_PRI_B (VERSION 1) [4]	8p-1	19p+2	5p+1	2p+1
FQR_PRI_B (VERSION 2) [8]	8p+1	20p+6	4p+2	2p+1

4.4 The Order Recursive Versions of the Fast QRD Algorithms

The fast QRD-RLS algorithms employing lower triangularization of the input data matrix are known as "hybrid QR-lattice least squares algorithms". It is clear from previous sections 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 section is the presentation of the order recursive (or 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 [13]. These order recursive versions can then be implemented with a modular structure, which utilizes a unique type of lattice stage for each algorithm.

Before their derivation, in order to help their understanding, let us specify in Table 4.7 the meaning of each variable used in both algorithms. It is worth men-

Table 4.7 Summary of variables used in FQR_POS_B and FQR_PRI_B algorithms.

$\mathbf{d}_{fq}(k)$:	rotated forward desired vector
$\mathbf{d}_{fq2}(k)$:	last $N+1$ elements of $\mathbf{d}_{fq}(k)$
$\mathbf{e}_f(k)$:	forward error vector
$\Vert \mathbf{e}_f(k) \Vert$:	norm of $\mathbf{e}_f(k)$
$\mathbf{e}_{fq}(k)$:	rotated $\mathbf{e}_f(k)$
$e_{fq_1}(k)$:	first element of $\mathbf{e}_{fq}(k)$
$\mathbf{Q}_\theta(k)$:	Givens matrix (updates the Cholesky factor)
$x(k)$:	input signal
λ :	forgetting factor
$\mathbf{Q}'_{\theta f}(k+1)$:	Givens matrix that annihilates $\mathbf{d}_{fq2}(k+1)$ in (4.18)
$\Vert \mathbf{e}_f^{(0)}(k) \Vert$:	norm of $\mathbf{e}_f(k)$ in a zero coefficient case
$\mathbf{f}(k)$:	*a posteriori* normalized errors
$\mathbf{a}(k)$:	*a priori* normalized errors
$\mathbf{e}_b(k)$:	backward error vector
$\Vert \mathbf{e}_b(k) \Vert$:	norm of $\mathbf{e}_b(k)$
$\varepsilon_f(k)$:	*a posteriori* forward prediction error
$e_f(k)$:	*a priori* forward prediction error
$\varepsilon_b(k)$:	*a posteriori* backward prediction error
$e_b(k)$:	*a priori* backward prediction error
$\gamma(k)$:	product of cosines of the angles of $\mathbf{Q}_\theta(k)$
$\mathbf{e}_q(k)$:	rotated error vector
$e_{q_1}(k)$:	first element of $\mathbf{e}_q(k)$
$\mathbf{d}_q(k)$:	rotated desired vector
$\mathbf{d}_{q_2}(k)$:	last $N+1$ elements of $\mathbf{d}_q(k)$
$d(k)$:	desired signal
$e(k)$:	*a priori* output error

tioning here that a variable with no superscript implies in an N-th order quantity or, equivalently, is related to an $N+1$ coefficients filtering. Let us take as illustration the norm of the forward energy: $\| \mathbf{e}_f(k) \| = \| \mathbf{e}_f^{(N+1)}(k) \|$.

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 [5, 8] to develop these algorithms originally and the implications are well explained in those two references. As pointed out in [5], within this framework was the solution to the parameter identification problem first addressed using fast QR algorithms. The work of [14] 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 the normalized errors present in the orthogonal matrix $\mathbf{Q}_\theta(k)$ 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 order recursive version. Starting from (4.6), we rewrite this equation evaluated at $k+1$, with an explicit form of $\mathbf{Q}_\theta(k)$ in terms of a product of $N+1$ Givens rotations $\mathbf{Q}_{\theta_i}(k)$ and with $e_{fq_1}^{(0)}(k+1) = x(k+1)$; we suggest the reader to check, in the previous chapter, the structure of $\mathbf{Q}_{\theta_i}(k)$ for the lower triangularization case.

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

$$\cdots \begin{bmatrix} \cos\theta_0(k) & \mathbf{0}^{\mathrm{T}} & -\sin\theta_0(k) \\ \mathbf{0} & \mathbf{I}_N & \mathbf{0} \\ \sin\theta_0(k) & \mathbf{0}^{\mathrm{T}} & \cos\theta_0(k) \end{bmatrix} \begin{bmatrix} e_{fq_1}^{(0)}(k+1) \\ \lambda^{1/2} d_{fq2_1}(k) \\ \vdots \\ \lambda^{1/2} d_{fq2_{N+1}}(k) \end{bmatrix} \tag{4.30}$$

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

$$\begin{bmatrix} \cos\theta_0(k) e_{fq_1}^{(0)}(k+1) - \sin\theta_0 \lambda^{1/2} d_{fq2_{N+1}}(k) \\ \lambda^{1/2} d_{fq2_1}(k) \\ \vdots \\ \lambda^{1/2} d_{fq2_N}(k) \\ \sin\theta_0(k) e_{fq_1}^{(0)}(k+1) + \cos\theta_0(k) \lambda^{1/2} d_{fq2_{N+1}}(k) \end{bmatrix}. \tag{4.31}$$

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:

$$\begin{aligned} e^{(i)}_{fq_1}(k+1) &= \cos\theta_{i-1}(k)e^{(i-1)}_{fq_1}(k+1) \\ &\quad -\sin\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \end{aligned} \tag{4.32}$$

$$\begin{aligned} d_{fq2_{N+2-i}}(k+1) &= \sin\theta_{i-1}(k)e^{(i-1)}_{fq_1}(k+1) \\ &\quad +\cos\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k) \end{aligned} \tag{4.33}$$

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

If we use a similar procedure with the equation $\begin{bmatrix} \mathbf{0} \\ \| \mathbf{e}^{(0)}_f(k+1) \| \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{d}_{fq_2}(k+1) \\ \| \mathbf{e}_f(k+1) \| \end{bmatrix}$ which is part of (4.18) used in the FQR_POS_B algorithm, we will find

$$\cos\theta'_{f_{i-1}}(k+1) = \frac{\| \mathbf{e}^{(i)}_f(k+1) \|}{\| \mathbf{e}^{(i-1)}_f(k+1) \|}, \text{ and} \tag{4.34}$$

$$\sin\theta'_{f_{i-1}}(k+1) = \frac{d_{fq2_{N+2-i}}(k+1)}{\| \mathbf{e}^{(i-1)}_f(k+1) \|}. \tag{4.35}$$

In the last equation, i varies from 1 to $N+1$ and the updating of the forward error energy is performed by the following generalization of (4.5)

$$\| \mathbf{e}^{(i)}_f(k+1) \| = \sqrt{\lambda \| \mathbf{e}^{(i)}_f(k) \|^2 + [e^{(i)}_{fq_1}(k+1)]^2}. \tag{4.36}$$

All other equations are combined in a single loop by computing partial results from the partial results of the previous equations. The resulting algorithm is described in detail in Appendix 3 and, although not identical, is similar to the one presented in [15]. A stage of its lattice structure is depicted in Figure 4.2, where the rotation and angle processors can be easily understood from the algorithmic description.

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 [4]. The algorithm is shown in Appendix 3 and Figure 4.3 depicts one stage of the lattice structure for this algorithm.

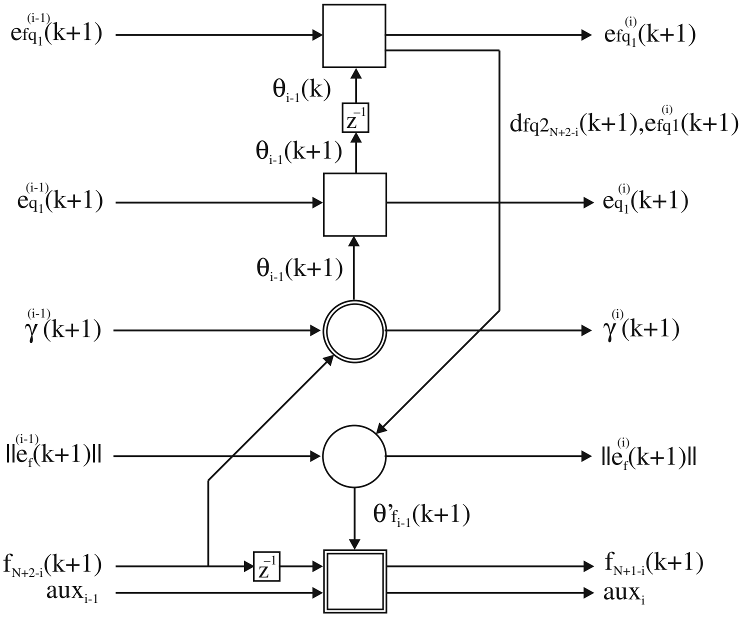


Fig. 4.2 One stage of the FQR_POS_B lattice structure.

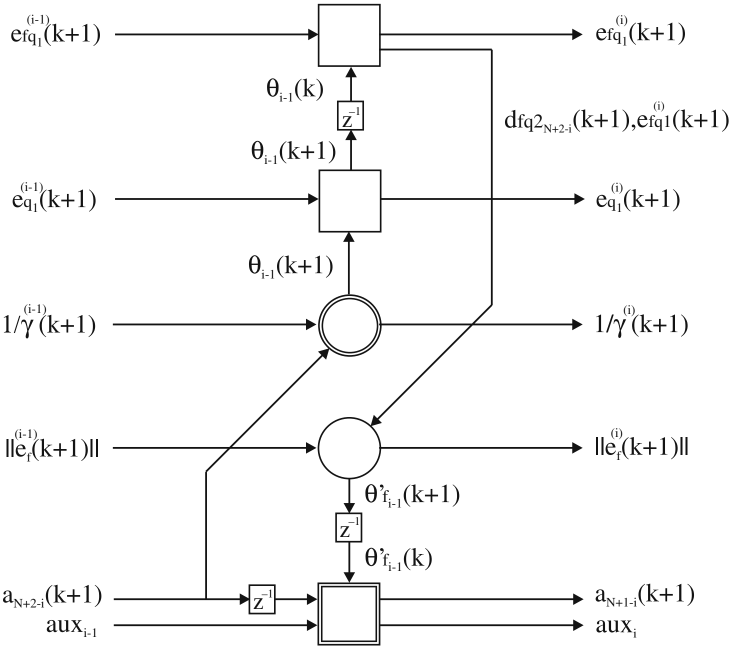


Fig. 4.3 One stage of the FQR_PRI_B lattice structure.

This section presented the order recursive or lattice versions of the fast QRD-RLS algorithms that update *a posteriori* and *a priori* backward errors. Results from the Gram–Schmidt orthogonalization can be used to conjecture that only the fast QR algorithms using lower triangularization, i.e., those updating backward prediction errors, would have their lattice versions easily implementable.

It can be observed from simulation results (not shown here) that the performance of the order recursive versions, in finite-precision implementations, is comparable to that of the original algorithms. The lattice versions have basically the same complexity as their original algorithms (FQR_POS_B and FQR_PRI_B) and lower complexity than the fast QR lattice algorithms previously proposed in [16].

4.5 Conclusion

This chapter presented the so-called fast versions of the QRD-RLS family of algorithms. In here, fast QRD-RLS algorithms using *a priori* and *a posteriori* forward and backward errors (based on upper or lower triangularization of the input data matrix) were derived. A comprehensive framework was used to classify the four fast QRD-RLS algorithms and their presentation were based on simple matrix equations, while detailed algorithmic descriptions were provided in appendices.

For those with a suitable application at hands and willing to select a fast QRD-RLS algorithm, as a finger rule, we stress the importance of those based on the updating of backward prediction errors (lower triangular algorithms as per the notation adopted in this chapter). Although they all might have similar computational complexity, the FQR_PRI_B and the FQR_POS_B algorithms present the desired feature of proven stability. The fast QRD-RLS algorithms can be used whenever the input signal vector consists of a delay line; in case where we have distinct signals each consisting of a delay line, multichannel versions are appropriate; this topic is addressed in Chapter 6.

Finally, it is worth mentioning that fast QRD-RLS algorithms do not provide the filter coefficients (transversal form) explicitly; they are however embedded in the internal variables. In applications where these weights are desirable, weight extraction techniques can be used as described in Chapter 11.

Appendix 1 - Pseudo-Code for the Fast QRD-RLS Algorithms Based on Forward Prediction Errors

1. The FQR_POS_F algorithm:

FQR_POS_F [3]

```
Initialization:
$\| \mathbf{e}_f(k) \| = \delta =$ small positive value;        $\mathbf{d}_{fq2}(k) = \mathbf{d}_{q2}(k) =$ zeros$(N+1,1)$;
$cos\theta(k) = cos\theta'_b(k) =$ ones$(N+1,1)$;               $sin\theta(k) = sin\theta'_b(k) =$ zeros$(N+1,1)$;
$\gamma(k) = 1$;                                                $\mathbf{f}(k) =$ zeros$(N+1,1)$;
for $k = 1, 2, \ldots$
{ $e^{(0)}_{fq_1}(k+1) = x(k+1)$;
  for $i = 1 : N+1$
  { $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) - sin\theta_{i-1}(k) \lambda^{1/2} d_{fq2_i}(k)$;
    $d_{fq2_i}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) + cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_i}(k)$;
  }
  $e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;
  $\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$; $cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$;
  $sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$;    $\mathbf{c}(k+1) = [1;$ zeros$(N+1,1)]$;
  for $i = 1 : N+1$
  { $c_{N+3-i}(k+1) = -sin\theta'_{b_{N+1-i}}(k) c_1$;      $c_1(k+1) = cos\theta'_{b_{N+1-i}}(k) c_1$;
  }
  $\mathbf{c}_{aux} = [0;\ c(k+1)]$;
  for $i = 1 : N+1$
  { $oldvalue = c_{aux_{i+1}}$;
    $c_{aux_{i+1}} = sin\theta_{i-1}(k) c_{aux_1} + cos\theta_{i-1}(k) c_{aux_{i+1}}$;
    $c_{aux_1} = cos\theta_{i-1}(k) c_{aux_1} - sin\theta_{i-1}(k) oldvalue$;
  }
  $oldvalue = c_{aux_1}$;
  $c_{aux_1} = cos\theta_f(k+1) c_{aux_1} - sin\theta_f(k+1) c_{aux_{N+3}}$;
  $c_{aux_{N+3}} = sin\theta_f(k+1) oldvalue + cos\theta_f(k+1) c_{aux_{N+3}}$;
  $\mathbf{c}(k+1) = \mathbf{c}_{aux}(2 : N+3)$;
  for $i = 1 : N+1$
  { $oldvalue = c_1$;                                $c_1 = \sqrt{c_1^2 + c_{i+1}^2}$;
    $cos\theta'_{b_{i-1}}(k+1) = oldvalue / c_1$;        $sin\theta'_{b_{i-1}}(k+1) = -c_{i+1} / c_1$;
  }
  $\mathbf{f}^{(N+2)}(k+1) = [\mathbf{f}(k);\ sin\theta_f(k+1)\gamma(k)]$;    $aux_0 = f^{(N+2)}_1(k+1)$;
  for $i = 1 : N+1$
  { $aux_i = cos\theta'_{b_{i-1}}(k+1) aux_{i-1} - sin\theta'_{b_{i-1}}(k+1) f^{(N+2)}_{i+1}(k+1)$;
    $f_i(k+1) = sin\theta'_{b_{i-1}}(k+1) aux_{i-1} + cos\theta'_{b_{i-1}}(k+1) f^{(N+2)}_{i+1}(k+1)$;
  }
  $\gamma^{(0)}(k+1) = 1$;
  for $i = 1 : N+1$
  { $sin\theta_{i-1}(k+1) = f_i(k+1) / \gamma^{(i-1)}(k+1)$;
    $cos\theta_{i-1}(k+1) = \sqrt{1 - sin^2\theta_{i-1}(k+1)}$;
    $\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1) \gamma^{(i-1)}(k+1)$;
  }
  $\gamma(k+1) = \gamma^{(N+1)}(k+1)$;                   $e^{(0)}_{q_1}(k+1) = d(k+1)$;
  for $i = 1 : N+1$
  { $e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) - sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_i}(k)$;
    $d_{q2_i}(k+1) = sin\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_i}(k)$;
  }
  $e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1)$;                  $e(k+1) = e_{q_1}(k+1) / \gamma(k+1)$;
}
```

2. The FQR_PRI_F algorithm:

FQR_PRI_F [7]

```
Initialization:
$\| \mathbf{e}_f(k) \| = \delta =$ small positive value;          $\mathbf{d}_{fq2}(k) = \mathbf{d}_{q2}(k) =$ zeros$(N+1,1)$;
$cos\theta(k) = cos\theta'_b(k) =$ ones$(N+1,1)$;                  $sin\theta(k) = sin\theta'_b(k) =$ zeros$(N+1,1)$;
$1/\gamma(k) = 1$;                                                 $\mathbf{a}(k) =$ zeros$(N+1,1)$;
for $k = 1, 2, \ldots$
{ $e^{(0)}_{fq_1}(k+1) = x(k+1)$;
  for $i = 1 : N+1$
  { $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) - sin\theta_{i-1}(k) \lambda^{1/2} d_{fq2_i}(k)$;
    $d_{fq2_i}(k+1) = sin\theta_{i-1}(k) e^{(i-1)}_{fq_1}(k) + cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_i}(k)$;
  }
  $e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;                          $e_f(k+1) = e_{fq_1}(k+1)[1/\gamma(k)]$;
  $\mathbf{a}^{(N+2)}(k+1) = [\mathbf{a}(k);\ \frac{e_f(k+1)}{\lambda^{1/2} \|\mathbf{e}_f(k)\|}]$;     $aux_0 = a^{(N+2)}_1(k+1)$;
  for $i = 1 : N+1$
  { $aux_i = cos\theta'_{b_{i-1}}(k) aux_{i-1} - sin\theta'_{b_{i-1}}(k) a^{(N+2)}_{i+1}(k+1)$;
    $a_i(k+1) = sin\theta'_{b_{i-1}}(k) aux_{i-1} + cos\theta'_{b_{i-1}}(k) a^{(N+2)}_{i+1}(k+1)$;
  }
  $\| \mathbf{e}_f(k+1) \| = \sqrt{e^2_{fq_1}(k+1) + \lambda \| \mathbf{e}_f(k) \|^2}$;
  $cos\theta_f(k+1) = \lambda^{1/2} \| \mathbf{e}_f(k) \| / \| \mathbf{e}_f(k+1) \|$;
  $sin\theta_f(k+1) = e_{fq_1}(k+1) / \| \mathbf{e}_f(k+1) \|$
  $\mathbf{c}(k+1) = [1;$ zeros$(N+1,1)]$;
  for $i = 1 : N+1$
  { $c_{N+3-i}(k+1) = -sin\theta'_{b_{N+1-i}}(k) c_1$;
    $c_1(k+1) = cos\theta'_{b_{N+1-i}}(k) c_1$;
  }
  $\mathbf{c}_{aux} = [0;\ c(k+1)]$;
  for $i = 1 : N+1$
  { $oldvalue = c_{aux_{i+1}}$;
    $c_{aux_{i+1}} = sin\theta_{i-1}(k) c_{aux_1} + cos\theta_{i-1}(k) c_{aux_{i+1}}$;
    $c_{aux_1} = cos\theta_{i-1}(k) c_{aux_1} - sin\theta_{i-1}(k) oldvalue$;
  }
  $oldvalue = c_{aux_1}$;
  $c_{aux_1} = cos\theta_f(k+1) c_{aux_1} - sin\theta_f(k+1) c_{aux_{N+3}}$;
  $c_{aux_{N+3}} = sin\theta_f(k+1) oldvalue + cos\theta_f(k+1) c_{aux_{N+3}}$;
  $\mathbf{c}(k+1) = c_{aux}(2 : N+3)$;
  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 $i = 1 : N+1$
  { $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + a_i^2(k+1)}$;
    $cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
    $sin\theta_{i-1}(k+1) = \frac{a_i(k+1)}{1/\gamma^{(i)}(k+1)}$;
  }
  $\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$;                         $e^{(0)}_{q_1}(k+1) = d(k+1)$;
  for $i = 1 : N+1$
  { $e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) - sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_i}(k)$;
    $d_{q2_i}(k+1) = sin\theta_{i-1}(k+1) e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_i}(k)$;
  }
  $e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1)$;                            $e(k+1) = e_{q_1}(k+1)/\gamma(k+1)$;
}
```

Appendix 2 - Pseudo-Code for the Fast QRD-RLS Algorithms Based on Backward Prediction Errors

1. The FQR_POS_B algorithm:

The first version of this algorithm is based on the fact that the last element of $\mathbf{f}(k+1)$, $f_{N+1}(k+1) = \frac{x(k+1)}{\|\mathbf{e}_f^{(0)}(k+1)\|}$, is known in advance.

FQR_POS_B - Version 1 [6]

```
Initialization:
$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$;          $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\theta(k) = \text{ones}(N+1,1)$;                 $sin\theta(k) = \text{zeros}(N+1,1)$;
$\| \mathbf{e}_f(k) \| = \delta = $ small positive value;   $\mathbf{f}(k) = \text{zeros}(N+1,1)$;
for $k = 1,2,\ldots$
{ $e_{fq_1}^{(0)}(k+1) = x^*(k+1)$;
  for $i = 1:N+1$
  { $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
    $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
  }
  $e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;          $\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;
  $\| \mathbf{e}_f^{(N+1)}(k+1) \| = \| \mathbf{e}_f(k+1) \|$;
  for $i = 1:N+1$
  { $\| \mathbf{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \mathbf{e}_f^{(N+2-i)}(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;
    $cos\theta'_{f_i}(k+1) = \| \mathbf{e}_f^{(N+2-i)}(k+1) \| / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;
    $sin\theta'_{f_i}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;
  }
  $aux_0 = x(k+1) / \| \mathbf{e}_f^{(0)}(k+1) \|$;          $f_{N+1}(k+1) = aux_0$;
  for $i = 1:N$
  { $f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - sin\theta'^*_{f_{N+1-i}}(k+1) aux_{i-1}}{cos\theta'_{f_{N+1-i}}(k+1)}$;
    $aux_i = -sin\theta'_{f_{N+1-i}}(k+1) f_{N+1-i}(k) + cos\theta'_{f_{N+1-i}}(k+1) aux_{i-1}$;
  }
  $\gamma^{(0)}(k+1) = 1$;
  for $i = 1:N+1$
  { $sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1) / \gamma^{(i-1)}(k+1)$;
    $cos\theta_{i-1}(k+1) = \sqrt{1 - |sin\theta_{i-1}(k+1)|^2}$;
    $\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1) \gamma^{(i-1)}(k+1)$;
  }
  $\gamma(k+1) = \gamma^{(N+1)}(k+1)$;          $e_{q_1}^{(0)}(k+1) = d^*(k+1)$;
  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 $\mathbf{f}(k+1)$ according to (4.20) and requires the calculation of $\frac{e_f(k+1)}{\|\mathbf{e}_f(k+1)\|}$.

FQR_POS_B - Version 2 [5]

```
Initialization:
$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1);$          $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1);$
$cos\theta(k) = \text{ones}(N+1,1);$                  $sin\theta(k) = \text{zeros}(N+1,1);$
$\| \mathbf{e}_f(k) \| = \delta = $ small positive value;   $\mathbf{f}(k) = \text{zeros}(N+1,1);\ \ \gamma(k) = 1$
for $k = 1,2,\ldots$
{ $e^{(0)}_{fq_1}(k+1) = x^*(k+1);$
  for $i = 1 : N+1$
  { $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k)e^{(i-1)}_{fq_1}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k);$
    $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k)e^{(i-1)}_{fq_1}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k);$
  }
  $e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1);$
  $\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2};$
  $\| \mathbf{e}^{(N+1)}_f(k+1) \| = \| \mathbf{e}_f(k+1) \|;$
  for $i = 1 : N+1$
  { $\| \mathbf{e}^{(N+1-i)}_f(k+1) \| = \sqrt{\| \mathbf{e}^{(N+2-i)}_f(k+1) \|^2 + |d_{fq2_i}(k+1)|^2};$
    $cos\theta'_{f_{N+1-i}}(k+1) = \| \mathbf{e}^{(N+2-i)}_f(k+1) \| \,/\, \| \mathbf{e}^{(N+1-i)}_f(k+1) \|;$
    $sin\theta'_{f_{N+1-i}}(k+1) = d^*_{fq2_i}(k+1)/ \| \mathbf{e}^{(N+1-i)}_f(k+1) \|;$
  }
  $aux_0 = \frac{\gamma(k)e^*_{fq_1}(k+1)}{\|\mathbf{e}_f(k+1)\|};$
  for $i = 1 : N+1$
  { $f_{i-1}(k+1) = cos\theta'_{f_{N+1-i}}(k+1)f_i(k) - sin\theta'^*_{f_{N+1-i}}(k+1)aux_{i-1};$
    $aux_i = sin\theta'_{f_{N+1-i}}(k+1)f_i(k) + cos\theta'_{f_{N+1-i}}(k+1)aux_{i-1};$
  }
  $\frac{\varepsilon_b(k+1)}{\|\mathbf{e}_b(k+1)\|} = f_0(k+1);$
  $f_{N+1}(k+1) = aux_{N+1};$
  $\gamma^{(0)}(k+1) = 1;$
  for $i = 1 : N+1$
  { $sin\theta_{i-1}(k+1) = f_{N+2-i}(k+1)/\gamma^{(i-1)}(k+1);$
    $cos\theta_{i-1}(k+1) = \sqrt{1 - |sin\theta_{i-1}(k+1)|^2};$
    $\gamma^{(i)}(k+1) = cos\theta_{i-1}(k+1)\gamma^{(i-1)}(k+1);$
  }
  $\gamma(k+1) = \gamma^{(N+1)}(k+1);$
  $e^{(0)}_{q_1}(k+1) = d^*(k+1);$
  for $i = 1 : N+1$
  { $e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1)e^{(i-1)}_{q_1}(k+1) - sin^*\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_{N+2-i}}(k);$
    $d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1)e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_{N+2-i}}(k);$
  }
  $e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1);$          $e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1);$
}
```

2. The FQR_PRI_B algorithm:

The first version of this algorithm is based on the fact that the last element of $\mathbf{a}(k+1)$ (or $a_{N+1}(k+1) = \frac{x(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f^{(0)}(k)\|}$) is known in advance.

FQR_PRI_B - Version 1 [4]

```
Initialization:
$\| \mathbf{e}_f^{(0)}(k) \| = \| \mathbf{e}_f(k) \| = \delta =$ small positive value;      $\mathbf{a}(k) = \text{zeros}(N+1,1)$;
$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$;                                      $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\theta(k) = \text{ones}(N+1,1)$;                                       $cos\theta'_f(k) = \text{ones}(N+1,1)$;
$sin\theta(k) = \text{zeros}(N+1,1)$;                                      $sin\theta'_f(k) = \text{zeros}(N+1,1)$;
for $k = 1,2,\ldots$
{ $e_{fq_1}^{(0)}(k+1) = x^*(k+1)$;
  for $i = 1 : N+1$
  { $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
    $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
  }
  $e_{fq_1}(k+1) = e_{fq_1}^{(N+1)}(k+1)$;
  $aux_0 = \frac{x^*(k+1)}{\lambda^{1/2}\|\mathbf{e}_f^{(0)}(k)\|}$;                              $a_{N+1}(k+1) = aux_0$;
  for $i = 1 : N$
  { $a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k) aux_{i-1}}{cos\theta'_{f_{i-1}}(k)}$;
    $aux_i = -sin\theta'^*_{f_{i-1}}(k) a_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k) aux_{i-1}$;
  }
  $\| \mathbf{e}_f^{(N+1)}(k+1) \| = \| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;
  for $i = 1 : N+1$
  { $\| \mathbf{e}_f^{(N+1-i)}(k+1) \| = \sqrt{\| \mathbf{e}_f^{(N+2-i)}(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;
    $cos\theta'_{f_{N+1-i}}(k+1) = \| \mathbf{e}_f^{(N+2-i)}(k+1) \| / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;
    $sin\theta'_{f_{N+1-i}}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}_f^{(N+1-i)}(k+1) \|$;
  }
  $1/\gamma^{(0)}(k+1) = 1$;
  for $i = 1 : N+1$
  { $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + |a_{N+2-i}(k+1)|^2}$
    $cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
    $sin\theta_{i-1}(k+1) = \frac{a^*_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;
  }
  $\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$;                        $e_{q_1}^{(0)}(k+1) = d^*(k+1)$;
  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 $\mathbf{a}(k+1)$ according to (4.29) and requires the calculation of $\frac{e'_f(k+1)}{\sqrt{\lambda}\|\mathbf{e}_f(k)\|}$.

FQR_PRI_B - Version 2 [8]

```
Initialization:
$\| \mathbf{e}_f(k) \| = \delta =$ small positive value;        $\mathbf{a}(k) = \text{zeros}(N+1,1)$;  $\gamma(k) = 1$;
$\mathbf{d}_{fq2}(k) = \text{zeros}(N+1,1)$;                    $\mathbf{d}_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\theta(k) = \text{ones}(N+1,1)$;                            $cos\theta'_f(k) = \text{ones}(N+1,1)$;
$sin\theta(k) = \text{zeros}(N+1,1)$;                           $sin\theta'_f(k) = \text{zeros}(N+1,1)$;
for $k = 1,2,\ldots$
{ $e^{(0)}_{fq_1}(k+1) = x^*(k+1)$;
  for $i = 1 : N+1$
  { $e^{(i)}_{fq_1}(k+1) = cos\theta_{i-1}(k)e^{(i-1)}_{fq_1}(k+1) - sin^*\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k)$;
    $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k)e^{(i-1)}_{fq_1}(k+1) + cos\theta_{i-1}(k)\lambda^{1/2}d_{fq2_{N+2-i}}(k)$;
  }
  $e_{fq_1}(k+1) = e^{(N+1)}_{fq_1}(k+1)$;
  $aux_0 = \frac{e_{fq_1}(k+1)}{\gamma(k)\lambda^{1/2}\|\mathbf{e}_f(k)\|}$;
  for $i = 1 : N+1$
  { $a_{i-1}(k+1) = cos\theta'_{f_{N+1-i}}(k)a_i(k) - sin\theta'_{f_{N+1-i}}(k)aux_{i-1}$;
    $aux_i = sin^*\theta'_{f_{N+1-i}}(k)a_i(k) + cos\theta'_{f_{N+1-i}}(k)aux_{i-1}$;
  }
  $\frac{e_b(k+1)}{\lambda^{1/2}\|\mathbf{e}_b(k)\|} = a_0(k+1)$;
  $a_{N+1}(k+1) = aux_{N+1}$;
  $\| \mathbf{e}_f(k+1) \| = \sqrt{|e_{fq_1}(k+1)|^2 + \lambda \| \mathbf{e}_f(k) \|^2}$;
  $\| \mathbf{e}^{(N+1)}_f(k+1) \| = \| \mathbf{e}_f(k+1) \|$;
  for $i = 1 : N+1$
  { $\| \mathbf{e}^{(N+1-i)}_f(k+1) \| = \sqrt{\| \mathbf{e}^{(N+2-i)}_f(k+1) \|^2 + |d_{fq2_i}(k+1)|^2}$;
    $cos\theta'_{f_{N+1-i}}(k+1) = \| \mathbf{e}^{(N+2-i)}_f(k+1) \| \, / \, \| \mathbf{e}^{(N+1-i)}_f(k+1) \|$;
    $sin\theta'_{f_{N+1-i}}(k+1) = d^*_{fq2_i}(k+1) / \| \mathbf{e}^{(N+1-i)}_f(k+1) \|$;
  }
  $1/\gamma^{(0)}(k+1) = 1$;
  for $i = 1 : N+1$
  { $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + |a_{N+2-i}(k+1)|^2}$;
    $cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
    $sin\theta_{i-1}(k+1) = \frac{a^*_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;
  }
  $\gamma(k+1) = 1/[1/\gamma^{(N+1)}(k+1)]$;                    $e^{(0)}_{q_1}(k+1) = d^*(k+1)$;
  for $i = 1 : N+1$
  { $e^{(i)}_{q_1}(k+1) = cos\theta_{i-1}(k+1)e^{(i-1)}_{q_1}(k+1) - sin^*\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_{N+2-i}}(k)$;
    $d_{q2_{N+2-i}}(k+1) = sin\theta_{i-1}(k+1)e^{(i-1)}_{q_1}(k+1) + cos\theta_{i-1}(k+1)\lambda^{1/2}d_{q2_{N+2-i}}(k)$;
  }
  $e_{q_1}(k+1) = e^{(N+1)}_{q_1}(k+1)$;                        $e(k+1) = e^*_{q_1}(k+1)/\gamma(k+1)$;
}
```

Appendix 3 - Pseudo-Code for the Order Recursive FQRD-RLS Algorithms

1. Order recursive version of the FQR_POS_B algorithm:

LATTICE FQR_POS_B

```
Soft-constrained initialization:
$\varepsilon$ = small positive value;
for $i = 0 : N+1$
{ $\| e_f^{(i)}(k) \| = \varepsilon$;
}
$d_{fq2}(k) = \text{zeros}(N+1,1)$;                $d_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\theta(k) = \text{ones}(N+1,1)$;                $sin\theta(k) = \text{zeros}(N+1,1)$;
$f(k) = \text{zeros}(N+1,1)$;
for each $k$
{  $e_{fq_1}^{(0)}(k+1) = x(k+1)$;
  $\| e_f^{(0)}(k+1) \| = \sqrt{[e_{fq_1}^{(0)}(k+1)]^2 + \lambda \| e_f^{(0)}(k) \|^2}$;
  $aux_0 = \frac{x(k+1)}{\|e_f^{(0)}(k+1)\|}$;
  $f_{N+1}(k+1) = aux_0$;
  $\gamma^{(0)}(k+1) = 1$;
  $e_{q_1}^{(0)}(k+1) = d(k+1)$;
  for $i = 1 : N+1$
  { $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) +$
                $cos\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
    $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) -$
            $sin\theta_{i-1}(k)\lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
    $\| e_f^{(i)}(k+1) \| = \sqrt{[e_{fq_1}^{(i)}(k+1)]^2 + \lambda \| e_f^{(i)}(k) \|^2}$;
    $cos\theta'_{f_{i-1}}(k+1) = \| e_f^{(i)}(k+1) \| / \| e_f^{(i-1)}(k+1) \|$;
    $sin\theta'_{f_{i-1}}(k+1) = d_{fq2_{N+2-i}}(k+1) / \| e_f^{(i-1)}(k+1) \|$;
    $f_{N+1-i}(k+1) = \frac{f_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k+1) aux_{i-1}}{cos\theta'_{f_{i-1}}(k+1)}$;
    $aux_i = -sin\theta'_{f_{i-1}}(k+1) f_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k+1) aux_{i-1}$;
    $\gamma^{(i)}(k+1) = \sqrt{[\gamma^{(i-1)}(k+1)]^2 - [f_{N+2-i}(k+1)]^2}$;
    $cos\theta_{i-1}(k+1) = \frac{\gamma^{(i)}(k+1)}{\gamma^{(i-1)}(k+1)}$;
    $sin\theta_{i-1}(k+1) = \frac{f_{N+2-i}(k+1)}{\gamma^{(i-1)}(k+1)}$;
    $dq2_{N+2-i}(k+1) = sin\theta_{i-1}(k+1) e_{q1}^{(i-1)}(k+1) +$
                $cos\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;
    $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q1}^{(i-1)}(k+1) -$
            $sin\theta_{i-1}(k+1)\lambda^{1/2} d_{q2_{N+2-i}}(k)$;
  }
  $e(k+1) = e_{q1}^{(N+1)}(k+1)/\gamma^{(N+1)}(k+1)$;
}
```

2. Order recursive version of the FQR_PRI_B algorithm:

```
LATTICE FQR_PRI_B
Soft-constrained initialization:
$\varepsilon$ = small positive value;
for $i = 0 : N+1$
{ $\| e_f^{(i)}(k) \| = \varepsilon$;
}
$d_{fq2}(k) = \text{zeros}(N+1,1)$;                $d_{q2}(k) = \text{zeros}(N+1,1)$;
$cos\theta(k) = \text{ones}(N+1,1)$;               $cos\theta'_f(k) = \text{ones}(N+1,1)$;
$sin\theta(k) = \text{zeros}(N+1,1)$;              $sin\theta'_f(k) = \text{zeros}(N+1,1)$;
$a(k) = \text{zeros}(N+1,1)$;
for each $k$
{  $aux_0 = \frac{x(k+1)}{\sqrt{\lambda}\|e_f^{(0)}(k)\|}$;
   $a_{N+1}(k+1) = aux_0$;
   $e_{fq_1}^{(0)}(k+1) = x(k+1)$;
   $\| e_f^{(0)}(k+1) \| = \sqrt{[e_{fq_1}^{(0)}(k+1)]^2 + \lambda \| e_f^{(0)}(k) \|^2}$;
   $1/\gamma^{(0)}(k+1) = 1$;
   $e_{q_1}^{(0)}(k+1) = d(k+1)$;
   for $i = 1 : N+1$
   {  $a_{N+1-i}(k+1) = \frac{a_{N+2-i}(k) - sin\theta'_{f_{i-1}}(k) aux_{i-1}}{cos\theta'_{f_{i-1}}(k)}$;
      $aux_i = -sin\theta'_{f_{i-1}}(k) a_{N+1-i}(k+1) + cos\theta'_{f_{i-1}}(k) aux_{i-1}$;
      $d_{fq2_{N+2-i}}(k+1) = sin\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) +$
                  $cos\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
      $e_{fq_1}^{(i)}(k+1) = cos\theta_{i-1}(k) e_{fq_1}^{(i-1)}(k+1) -$
               $sin\theta_{i-1}(k) \lambda^{1/2} d_{fq2_{N+2-i}}(k)$;
      $\| e_f^{(i)}(k+1) \| = \sqrt{[e_{fq_1}^{(i)}(k+1)]^2 + \lambda \| e_f^{(i)}(k) \|^2}$;
      $cos\theta'_{f_{i-1}}(k+1) = \| e_f^{(i)}(k+1) \| \,/\, \| e_f^{(i-1)}(k+1) \|$;
      $sin\theta'_{f_{i-1}}(k+1) = d_{fq2_{N+2-i}}(k+1) / \| e_f^{(i-1)}(k+1) \|$;
      $1/\gamma^{(i)}(k+1) = \sqrt{[1/\gamma^{(i-1)}(k+1)]^2 + [a_{N+2-i}(k+1)]^2}$;
      $cos\theta_{i-1}(k+1) = \frac{1/\gamma^{(i-1)}(k+1)}{1/\gamma^{(i)}(k+1)}$;
      $sin\theta_{i-1}(k+1) = \frac{a_{N+2-i}(k+1)}{1/\gamma^{(i)}(k+1)}$;
      $dq2_{N+2-i}(k+1) = sin\theta_{i-1}(k+1) e_{q1}^{(i-1)}(k+1) +$
                  $cos\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;
      $e_{q_1}^{(i)}(k+1) = cos\theta_{i-1}(k+1) e_{q_1}^{(i-1)}(k+1) -$
               $sin\theta_{i-1}(k+1) \lambda^{1/2} d_{q2_{N+2-i}}(k)$;
   }
   $e(k+1) = e_{q_1}^{(N+1)}(k+1)[1/\gamma^{(N+1)}(k+1)]$;
}
```

References

1. S. Haykin, Adaptive Filter Theory. 2nd edition Prentice-Hall, Englewood Cliffs, NJ, USA (1991)
2. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
3. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-38, no. 4, pp. 631–653 (April 1990)
4. A. A. Rontogiannis and S. Theodoridis, New fast inverse QR least squares adaptive algorithms. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 1412–1415 (May 1995)
5. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. SP-39, no. 4, pp. 879–891 (April 1991)
6. J. A. Apolinário Jr., M. G. Siqueira, and P. S. R. Diniz, On fast QR algorithms based on backward prediction errors: New results and comparisons. First Balkan Conference on Signal Processing, Communications, Circuits, and Systems, Istanbul, Turkey (June 2000)
7. J. A. Apolinário Jr. and P. S. R. Diniz, A new fast QR algorithm based on a priori errors. IEEE Signal Processing Letters, vol. 4, no. 11, pp. 307–309 (November 1997)
8. M. D. Miranda and M. Gerken, A hybrid QR-lattice least squares algorithm using a priori errors. 38th Midwest Symposium on Circuits and Systems, MWSCAS'95, Rio de Janeiro, Brazil, pp. 983–986 (August 1995)
9. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares adaptive filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. SP-41, no. 1, pp. 20–30 (January 1993)
10. P. G. Park and T. Kailath, A lattice algorithm dual to the extended inverse QR algorithm. Signal Processing, vol. 47, no. 2, pp. 115–133 (November 1995)
11. M. D. Miranda and M. Gerken, A hybrid least squares QR-lattice algorithm using a priori errors. IEEE Transactions on Signal Processing, vol. 45, no. 12, pp. 2900–2911 (December 1997)
12. M. G. Bellanger, The FLS-QR algorithm for adaptive filtering. Signal Processing, vol. 17, no. 4, pp. 291–304 (August 1989)
13. J. A. Apolinário Jr., *New algorithms of adaptive filtering: LMS with data-reusing and fast RLS based on QR decomposition*. DSc thesis, UFRJ - Federal University of Rio de Janeiro, Rio de Janeiro, Brazil (1998)
14. Maria D. Miranda, *Sobre algoritmos dos mínimos quadrados rápidos, recursivos na ordem, que utilizam triangularização ortogonal* (in Portuguese). PhD thesis, USP - University of São Paulo, São Paulo, Brazil (1996)
15. M. Terré and M. Bellanger, A systolic QRD-based algorithm for adaptive filtering and its implementation. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'94, Adelaide, Australia, pp. 373–376 (April 1994)
16. I. K. Proudler, J. G. McWhirter, and T. J. Shepard, Computationally efficient QR decomposition approach to least squares adaptive filtering. IEE Proceedings-F, vol. 138, no. 4, pp. 341–353 (August 1991)

Chapter 5
QRD Least-Squares Lattice Algorithms

Jenq-Tay Yuan

Abstract This chapter presents a full derivation of the square-root-free (SRF) QR-decomposition-based least-squares lattice (QRD-LSL) algorithms in complex form, based on linear interpolation (or two-sided prediction) theory as a generalization of linear prediction theory. The conventionally adopted QRD-LSL prediction algorithm can be derived directly from the QRD-LSL interpolation algorithm and then extended to solve the joint process estimation problem. The QRD-LSL interpolation algorithm that produces interpolation errors (residuals) of various orders may have potential implications for some signal processing and communication problems. Interestingly, the QRD-LSL interpolation algorithm can also be used to calculate the Kalman gain vector to implement the widely known recursive least-squares (RLS) algorithm in a transversal structure to generate the least-squares filter weights at each time step. Therefore, linear interpolation theory may provide a bridge between lattice filters and transversal filters. The chapter is organized as follows. Section 5.1 presents the fundamentals of QRD-LSL algorithms. The LSL interpolator and the LSL predictor are briefly presented in Section 5.2. Section 5.3 presents the SRF Givens rotation with feedback mechanism that is employed to develop the SRF QRD-LSL algorithms. In Section 5.4, the SRF QRD-LSL interpolation algorithm is derived, and then reduced to the SRF QRD-LSL prediction algorithm, which is then extended to develop the SRF joint process estimation. The RLS algorithm in the transversal structure based on the SRF QRD-LSL interpolation algorithm is presented in Section 5.5 followed by some simulation results in Section 5.6. Section 5.7 draws conclusions.

Jenq-Tay Yuan
Fu Jen Catholic University, Taipei, Taiwan – R.O.C.
e-mail: yuan@ee.fju.edu.tw

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_5, © Springer Science+Business Media, LLC 2009

5.1 Fundamentals of QRD-LSL Algorithms

An Nth-*stage lattice filter*, displayed in Figure 5.1, automatically generates all N of the outputs that would be provided by N separate transversal filters of length $1,2,\ldots,N$ [1–6]. In Figure 5.1, $\varepsilon_{f,m}(k)$ and $\varepsilon_{b,m}(k)$ are the forward and backward prediction errors of order m; $\pi_{f,1}(k),\ldots,\pi_{f,N}(k)$ and $\pi_{b,1}(k),\ldots,\pi_{b,N}(k)$ are the forward reflection coefficients and backward reflection coefficients, respectively, which can be obtained by minimizing the sum of weighted prediction error squares at each stage. Optimum higher order lattice filters can be constructed from lower order ones by simply adding more lattice stages, leaving the original stages unchanged. This property is called the *order-recursive property* (or decoupling property) and follows from the fact that lattice filters essentially perform the Gram–Schmidt orthogonalization recursively such that each stage as effectively as possible decorrelates (or orthogonalizes) the inputs that enter it. This order-recursive property of a lattice filter along with its good numerical property is also shared by the QR-decomposition (QRD)-based algorithms. Accordingly, a combination of both the QRD-based algorithm and the least-squares lattice (LSL) filter can be reasonably expected to be a powerful algorithm, known as the *QR-decomposition-based least-squares lattice* (QRD-LSL) *algorithm*, for solving the adaptive least-squares (LS) filtering problem when the corresponding filter weights are not required. The order-recursive property of the QRD-LSL algorithms allows a variable-length filter to be designed, since it permits dynamic assignment, and rapid automatic determination of the most effective filter length. Consequently, in many LS applications, such as acoustic echo cancelation [7], in which only the LS errors of various orders are required to reduce

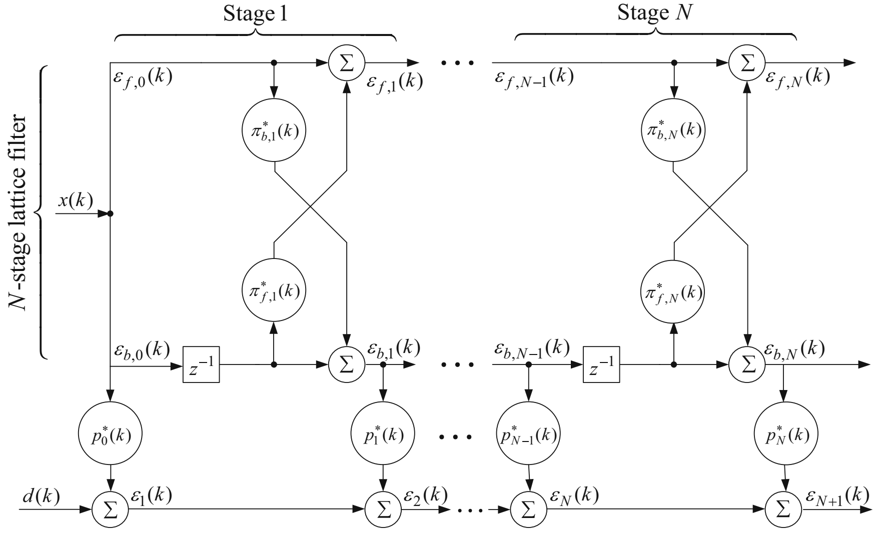


Fig. 5.1 Joint process estimation based on an Nth-*stage* lattice filter.

effectively near-end speech distortion and improve noise robustness without explicitly computing the corresponding filter weights, the QRD-LSL prediction algorithm along with the joint process estimation may play an important role. Even when the corresponding filter weights are required in applications such as system identification, the QRD-LSL interpolation algorithm that is developed in this chapter can still be applied to implement efficiently the well-known *recursive least-squares* (RLS) *algorithm*.

A recursive fast QRD-based LS filtering algorithm (or known as the QRD-based fast Kalman algorithm) of $\mathscr{O}[N]$ complexity was developed by Cioffi [8], where N is the number of taps in the adaptive filter. Although this algorithm is of a fixed-order transversal filter type and thus lacks the flexibility of order recursiveness, it presents for the first time the idea of a fast QRD-based algorithm for RLS estimation. Proudler et al. [9] developed a QRD-LSL prediction algorithm of $\mathscr{O}[N]$ complexity recursively in time and order. They then extended this QRD-LSL prediction algorithm to solve the joint process estimation problem. A similar fast LS lattice algorithm based on Givens rotations was developed independently by Ling [10]. Regalia and Bellanger [11] developed a hybrid QR-lattice algorithm that is closely related to the algorithms of Cioffi [8], Proudler et al. [9], and Ling [10], but they emphasized the relationship between fast QR algorithms and lattice filters. Other related works can also be found in Rontogiannis and Theodoridis [12], who proposed a unified approach for deriving fast QRD algorithms. The QRD-LSL algorithm is now well-known to provide many desirable features such as computational efficiency, fast rate of convergence and fast tracking capability, robustness against and insensitivity to round-off noise, and suitability for very large scale integration (VLSI) implementations. According to the simulations conducted by Yang and Bohme [13], the joint process estimation based on the QRD-LSL prediction algorithm is numerically more stable than the fixed order fast QR algorithms such as those developed by Cioffi [8] and Regalia and Bellanger [11]. The unstable behaviors exhibited by some fast Kalman algorithms may further justify the use of the order-recursive QRD-LSL algorithms.

Almost all of the QRD-LSL algorithms discussed in the literature are designed to solve the linear prediction problem recursively in time and order. This chapter addresses the QRD-LSL problem differently, and presents a complete derivation of the square-root-free (SRF) QRD-LSL algorithms from the perspective of linear interpolation. The derivation of the QRD-LSL interpolation algorithm is motivated by three main considerations. First, the QRD-LSL interpolation algorithm has potential implications for signal processing and communication problems, including data compression, coding and restoration of speech and images, and narrowband interference suppression in spread spectrum communications [14–18]. Secondly, the QRD-LSL prediction algorithm is in effect a special case of the QRD-LSL interpolation algorithm. As will be demonstrated in Section 5.4, the widely known QRD-LSL prediction algorithm can be developed directly from the QRD-LSL interpolation algorithm, because linear interpolation theory is a generalization of linear prediction theory and the former provides a broader interpretation and a more thorough understanding of the latter. Studies [19–22] have demonstrated that linear interpolation may substantially outperform linear prediction in terms of minimum mean squared error, because interpolation makes better use of the correlation between the nearest neighboring samples than does prediction. Thirdly, linear interpolation turns out to form a bridge between an order-recursive lattice filter that generates filter output only and the RLS algorithm, which also generates the adaptive filter weights. Skidmore and Proudler [23] first realized the crucial fact that the Kalman gain vector that is used to compute the RLS algorithm can be calculated as a particular set of normalized LS interpolation errors (residuals). The SRF QRD-LSL interpolation algorithm that produces interpolation errors of various orders can thus be adopted to implement the RLS algorithm. Although the resulting RLS algorithm of $\mathscr{O}[N\log_2 N]$ complexity may exhibit linear error growth with time in a limited-precision environment, it may offer a favorable compromise between some computationally efficient fast transversal filter algorithms of $\mathscr{O}[N]$ complexity, which may exhibit exponential error growth with time in a limited-precision environment, and some numerically stable algorithms of $\mathscr{O}[N^2]$ complexity, which may not be computationally feasible for some real-time applications.

5.2 LSL Interpolator and LSL Predictor

Linear interpolation estimates an unknown data sample based on a weighted sum of surrounding data samples. In one-dimensional signal processing, (p,f)th-*order* linear interpolation is the linear estimation of present input data samples $x(i)$ from its p past and f future neighboring data samples with the pre-windowing condition on the data (i.e., $x(i)=0$, for $i\le -1$), which is

$$\widehat{x}_{p,f}(i) = -\sum_{\substack{n=-p\\ n\neq 0}}^{f} b^*_{(p,f),n}(k-f)x(i+n), \qquad -f\le i\le k-f, \tag{5.1}$$

where $b_{(p,f),n}(k-f)$ is the interpolation coefficient at time k. The (p,f)th-*order* interpolation error at each time unit can thus be written as

$$\varepsilon^I_{p,f}(i) = x(i)-\widehat{x}_{p,f}(i) = \mathbf{b}^{\mathrm{H}}_{p,f}(k-f)\mathbf{x}_{N+1}(i+f), \qquad -f\le i\le k-f, \tag{5.2}$$

where $\mathbf{b}^{\mathrm{H}}_{p,f}(k-f) = [b^*_{(p,f),f}(k-f),\ldots,b^*_{(p,f),1}(k-f),1,b^*_{(p,f),-1}(k-f),\ldots,b^*_{(p,f),-p}(k-f)]$ is the interpolation coefficient vector at time k and the $(N+1)\times 1$ input vector is given as $\mathbf{x}_{N+1}(i) = [x(i),x(i-1),\ldots,x(i-N)]^{\mathrm{T}}$ in which $N=p+f$ is assumed implicitly. Notably, the notations used in this chapter are somewhat different from those used in the four previous chapters. In this chapter, we refer to any interpolation filter that operates on the present data sample as well as p past and f future data samples to produce the (p,f)th-*order* interpolation errors at its output as a (p,f)th-*order interpolator*. If the most recent data sample used is $x(k)$, then (5.2) can be written in a matrix form as

$$\boldsymbol{\varepsilon}^{I}_{p,f}(k-f) = \mathbf{X}_{N+1}(k)\mathbf{b}_{p,f}(k-f), \tag{5.3}$$

where $\boldsymbol{\varepsilon}^{I}_{p,f}(k-f) = \left[\, \varepsilon^{I^*}_{p,f}(-f) \;\ldots\; \varepsilon^{I^*}_{p,f}(-1) \;\varepsilon^{I^*}_{p,f}(0)\; \varepsilon^{I^*}_{p,f}(1) \;\ldots\; \varepsilon^{I^*}_{p,f}(k-f) \,\right]^{\mathrm{T}}$, $\mathbf{b}_{p,f}(k-f) = \left[\, b_{(p,f),f}(k-f) \;\ldots\; b_{(p,f),1}(k-f) \;1\; b_{(p,f),-1}(k-f) \;\ldots\; b_{(p,f),-p}(k-f) \,\right]^{\mathrm{T}}$, and the $(k+1)\times(N+1)$ matrix $\mathbf{X}_{N+1}(k)$ can be written as

$$\left[\; \underbrace{\begin{matrix} x^*(0) & 0 & \cdots & 0 \\ x^*(1) & x^*(0) & \ddots & \vdots \\ x^*(2) & x^*(1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \vdots & \vdots & \ddots & x^*(0) \\ \vdots & \vdots & \ddots & x^*(1) \\ \vdots & \vdots & \ddots & x^*(2) \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x^*(k) & x^*(k-1) & \cdots & x^*(k-f+1) \end{matrix}}_{f \text{ future data samples}} \quad \underbrace{\begin{matrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ x^*(0) \\ x^*(1) \\ \vdots \\ \vdots \\ \vdots \\ x^*(k-f) \end{matrix}}_{\substack{\text{data sample} \\ \text{to be estimated}}} \quad \underbrace{\begin{matrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 0 & \ddots & \vdots \\ x^*(0) & \ddots & \vdots \\ \vdots & \ddots & 0 \\ \vdots & \ddots & x^*(0) \\ \vdots & \ddots & \vdots \\ x^*(k-f-1) & \cdots & x^*(k-N) \end{matrix}}_{\substack{p \text{ past} \\ \text{data samples}}} \;\right].$$

5.2.1 LSL interpolator

The LS solution of the interpolation coefficients can be determined by minimizing the sum of interpolation error squares $\xi(k) = \sum_{i=-f}^{k-f} |\varepsilon^{I}_{p,f}(i)|^2 = \left[\boldsymbol{\varepsilon}^{I}_{p,f}(k-f)\right]^{\mathrm{H}} \boldsymbol{\varepsilon}^{I}_{p,f}(k-f) = \left[\mathbf{X}_{N+1}(k)\mathbf{b}_{p,f}(k-f)\right]^{\mathrm{H}} \left[\mathbf{X}_{N+1}(k)\mathbf{b}_{p,f}(k-f)\right] = \mathbf{b}^{\mathrm{H}}_{p,f}(k-f)\mathbf{R}(k)\mathbf{b}_{p,f}(k-f)$ subject to the constraint $\mathbf{h}^{\mathrm{H}}\mathbf{b}_{p,f}(k-f) = 1$, where $\mathbf{R}(k) = \mathbf{X}^{\mathrm{H}}_{N+1}(k)\mathbf{X}_{N+1}(k)$ is the $(N+1)\times(N+1)$ time-average correlation matrix of the input data samples $x(i)$ and $\mathbf{h}^{\mathrm{H}} = \left[\,\mathbf{0}^{\mathrm{T}}_{f} \;1\; \mathbf{0}^{\mathrm{T}}_{p}\,\right]$. Using the method of Lagrange, the interpolation coefficient vector can be computed to be $\mathbf{b}_{p,f}(k-f) = \left(\mathbf{h}^{\mathrm{T}}\mathbf{R}^{-1}(k)\mathbf{h}\right)^{-1}\mathbf{R}^{-1}(k)\mathbf{h}$ and the minimum sum of interpolation error squares of order (p,f) can be computed to be $I_{p,f}(k-f) = \xi_{min}(k) = \left(\mathbf{h}^{\mathrm{T}}\mathbf{R}^{-1}(k)\mathbf{h}\right)^{-1}$. The resulting augmented asymmetric interpolation normal equations can thus be written as

$$\mathbf{R}(k)\mathbf{b}_{p,f}(k-f) = \mathbf{i}_{p,f}(k-f), \tag{5.4}$$

where $\mathbf{i}_{p,f}(k-f) = [\,\mathbf{0}^{\mathrm{T}}_{f}\; I_{p,f}(k-f)\; \mathbf{0}^{\mathrm{T}}_{p}\,]^{\mathrm{T}}$ is the $(N+1)\times 1$ vector in which $\mathbf{0}_f$ and $\mathbf{0}_p$ are column vectors of f and p zeros, respectively. A computationally efficient

LSL interpolator developed in [22] requiring only $\mathscr{O}[N]$ operations via "intermediate predictions" can be employed to compute the exact order-updated interpolation errors.

The LSL interpolator computes the order-recursive interpolation errors as an additional *past* data sample [i.e., $(p,f) \rightarrow (p+1,f)$] and an additional *future* data sample [i.e., $(p,f) \rightarrow (p,f+1)$] are taken into account, respectively, as follows:

$$\varepsilon^I_{p+1,f}(k-f) = \varepsilon^I_{p,f}(k-f) - k^*_{p+1,f}(k)\varepsilon_{b,N+1}(k,k-f), \tag{5.5}$$

$$\varepsilon^I_{p,f+1}(k-f-1) = \varepsilon^I_{p,f}(k-f-1) - k^*_{p,f+1}(k)\varepsilon_{f,N+1}(k,k-f-1), \tag{5.6}$$

where both $k_{p+1,f}(k)$ and $k_{p,f+1}(k)$ are the complex-valued coefficients of the interpolator;

$$\varepsilon_{f,N+1}(i,i-f-1) = x(i) + \sum_{\substack{n=1 \\ n\neq f+1}}^{N+1} a^*_{N+1,n}(k)x(i-n), \quad 0 \leq i \leq k \tag{5.7}$$

is referred to as the $(N+1)$th-*order* intermediate forward prediction (IFP) error, as it is the prediction error of $x(i)$ based on a weighted linear combination of its $(N+1)$ previous data samples [i.e., $x(i-1),\ldots,x(i-f),x(i-f-2),\ldots,x(i-N-1)$] without considering present data sample $x(i-f-1)$, where $a_{N+1,n}(k), n = 1,2,\ldots,f,f+2,\ldots,N+1$ are $(N+1)$th-*order* IFP coefficients. Similarly, the $(N+1)$th-*order* intermediate backward prediction (IBP) error is defined as the prediction error of $x(i-N-1)$ based on a weighted linear combination of $x(i-N),x(i-N+1),\ldots,x(i-f-1),x(i-f+1),\ldots,x(i)$ without considering the data sample $x(i-f)$:

$$\varepsilon_{b,N+1}(i,i-f) = x(i-N-1) + \sum_{\substack{n=1 \\ n\neq p+1}}^{N+1} c^*_{N+1,n}(k)x(i+n-N-1), \;\; 0 \leq i \leq k, \tag{5.8}$$

where $c_{N+1,n}(k)$, $n = 1,2,\ldots,p,p+2,\ldots,N+1$ are $(N+1)$th-*order* intermediate backward prediction coefficients. It is worth pointing out that two special cases arise when $f=0$ and when $p=N$, and, as a result, the Nth-*order* IFP error, $\varepsilon_{f,N}(k,k-f)$, is reduced to $\varepsilon_{f,N}(k)$ and $\varepsilon_{f,N-1}(k)$, which are the conventional forward prediction (FP) errors of order N and order $(N-1)$, respectively. Similarly, when $p=0$ and $f=N$, the Nth-*order* intermediate backward prediction error, $\varepsilon_{b,N}(k,k-f)$, is reduced to $\varepsilon_{b,N-1}(k-1)$ and $\varepsilon_{b,N}(k)$, which are the conventional backward prediction (BP) errors of order $(N-1)$ and order N, respectively.

5.2.2 Orthogonal bases for LSL interpolator

To construct an LSL interpolator of order (p, f), Equations (5.5) and (5.6) must be applied p and f times, respectively. However, any sequencing between these two equations is permissible. Consequently, an LSL interpolator of order (p, f) has $C_N^p = C_N^f = \frac{N!}{p!f!}$ permissible realizations. For instance, to construct a (2,2)th-*order* interpolator, a total of $C_4^2 = 6$ permissible realizations may be identified by the sequences BFBF, FBFB, BBFF, FFBB, FBBF, and BFFB of intermediate backward (B) and intermediate forward (F) prediction errors that are used in (5.5) and (5.6), respectively. Six possible *order-recursive* realizations for a $(2,2)$th-*order* LSL interpolator can thus be constructed by employing the six *orthogonal bases*. As an example, the orthogonal basis identified by BFBF sequence may be verified as follows. The sequence BFBF reveals that, to estimate the data sample $x(k-2)$, the data sample immediately prior to $x(k-2)$(i.e., $x(k-3)$) is considered first corresponding to a B followed by the consideration of the data sample immediately future to $x(k-2)$ (i.e., $x(k-1)$) corresponding to an F. The above two operations are then followed by the consideration of one additional past data sample, $x(k-4)$, corresponding to a B, and one additional future data sample, $x(k)$, corresponding to an F. The intermediate prediction error basis, which is an orthogonal basis set, can therefore be generated by

$$\begin{aligned}
&\left[\, \varepsilon_{b,1}(k-2,k-2)\ \varepsilon_{f,2}(k-1,k-2)\ \varepsilon_{b,3}(k-1,k-2)\ \varepsilon_{f,4}(k,k-2)\ \varepsilon_{2,2}^{I}(k-2)\,\right]^{\mathrm{T}} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a_{2,2}^{*}(k) & 1 & 0 & 0 & 0 \\ c_{3,1}^{*}(k) & c_{3,3}^{*}(k) & 1 & 0 & 0 \\ a_{4,3}^{*}(k) & a_{4,1}^{*}(k) & a_{4,4}^{*}(k) & 1 & 0 \\ b_{(2,2),-1}^{*}(k-2) & b_{(2,2),1}^{*}(k-2) & b_{(2,2),-2}^{*}(k-2) & b_{(2,2),2}^{*}(k-2) & 1 \end{bmatrix} \begin{bmatrix} x(k-3) \\ x(k-1) \\ x(k-4) \\ x(k) \\ x(k-2) \end{bmatrix},
\end{aligned} \tag{5.9}$$

The above transformation is known as the Gram–Schmidt orthogonalization procedure. The other five orthogonal basis vectors identified by the sequences FBFB, BBFF, FFBB, FBBF, and BFFB, can be similarly verified. Each of the orthogonal bases identified by the six sequences can be used to construct an *order-recursive* realization for a $(2,2)$th-*order* LSL interpolator.

Both IFP and IBP errors must be computed before the *order-updated* interpolation errors in (5.5) and (5.6) can be computed. They can be computed using the conventional BP and FP errors, as follows.

$$\varepsilon_{b,N+1}(k,k-f) = \varepsilon_{b,N+1}(k) + l_{b,N+1}^{*}(k)\varepsilon_{p,f}^{I}(k-f), \tag{5.10}$$

$$\varepsilon_{f,N+1}(k,k-f-1) = \varepsilon_{f,N+1}(k) + l_{f,N+1}^{*}(k)\varepsilon_{p,f}^{I}(k-f-1), \tag{5.11}$$

where $l_{b,N+1}(k)$ and $l_{f,N+1}(k)$ are complex-valued coefficients; $\varepsilon_{b,N+1}(k)$ and $\varepsilon_{f,N+1}(k)$, which are conventional BP and FP errors, respectively, are directly accessible from an $(N+1)$th-*order LSL predictor* that can be embedded into an LSL interpolator.

Notably, both $\varepsilon^{I}_{p,f}(k-f)$ and $\varepsilon^{I}_{p,f}(k-f-1)$ are already computed from the previous interpolation lattice stage of the LSL interpolator. Equations (5.5), (5.6), (5.10), and (5.11) together with the well-known LSL predictor, constitute an *order-recursive LSL interpolator*.

5.2.3 LSL predictor

The well-known exact decoupling property of the LSL predictor can be readily demonstrated to be a special case of the orthogonal basis of the LSL interpolator corresponding to sequence BB...B. By setting $(p,f) = (N,0)$ (i.e., an Nth-*order* LSL predictor), there is one unique orthogonal basis set for use in an LSL interpolator of order $(N,0)$ (since $C_N^0 = 1$), which is $\left[\varepsilon_{b,1}(k,k), \varepsilon_{b,2}(k,k), \ldots, \varepsilon_{b,N}(k,k), \varepsilon^{I}_{N,0}(k)\right]$. This orthogonal basis set is clearly equivalent to $[\varepsilon_{b,0}(k-1), \varepsilon_{b,1}(k-1), \ldots, \varepsilon_{b,N-1}(k-1), \varepsilon_{f,N}(k)]$, which, in turn, corresponds to $[\varepsilon_{b,0}(k), \varepsilon_{b,1}(k), \ldots, \varepsilon_{b,N-1}(k)]$ that consists of a sequence of N uncorrelated backward prediction errors at all instants of time and forms a unique orthogonal basis set [5].

The widely known LSL predictor can also be derived directly from the LSL interpolator by setting $(p,f) = (N,0)$ and $(p,f) = (0,N)$ in (5.10) and (5.11), respectively, and yields

$$\varepsilon_{b,N+1}(k) = \varepsilon_{b,N}(k-1) - l^{*}_{b,N+1}(k)\varepsilon_{f,N}(k) \tag{5.12}$$

$$\varepsilon_{f,N+1}(k) = \varepsilon_{f,N}(k) - l^{*}_{f,N+1}(k)\varepsilon_{b,N}(k-1). \tag{5.13}$$

Notably, we have used the fact that $\varepsilon_{b,N+1}(k,k) = \varepsilon_{b,N}(k-1)$, $\varepsilon^{I}_{N,0}(k) = \varepsilon_{f,N}(k)$, $\varepsilon_{f,N+1}(k,k-N-1) = \varepsilon_{f,N}(k)$, and $\varepsilon^{I}_{0,N}(k-N-1) = \varepsilon_{b,N}(k-1)$. Interestingly, (5.12) and (5.13) can also be reduced directly from (5.6) and (5.5) by setting $(p,f) = (0,N)$ and $(p,f) = (N,0)$, respectively.

5.3 SRF Givens Rotation with Feedback Mechanism

Consider first a Givens rotation matrix given by $\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix}$, where the two parameters of the Givens rotation are the real cosine parameter c and the complex sine parameter s, such that $c^2 + |s|^2 = 1$. A Givens rotation used to zero out the element at the $(2,1)$ location is an elementary transformation of the form

$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_p \\ \beta_1 & \beta_2 & \dots & \beta_p \end{bmatrix} = \begin{bmatrix} \alpha_1' & \alpha_2' & \dots & \alpha_p' \\ 0 & \beta_2' & \dots & \beta_p' \end{bmatrix}, \tag{5.14}$$

where α_1 and α_1' are defined to be real and non-negative, whereas $\alpha_2 \dots \alpha_p$, $\alpha_2' \dots \alpha_p'$, $\beta_1 \dots \beta_p$, $\beta_2' \dots \beta_p'$ are all complex. Substituting $s = c\beta_1/\alpha_1$ into $c^2 + |s|^2 = 1$ yields $c = \alpha_1/\alpha_1'$, where $\alpha_1' \triangleq \sqrt{\alpha_1^2 + |\beta_1|^2}$. Accordingly,

$$c = \frac{\alpha_1}{\alpha_1'} \text{ and } s = \frac{\beta_1}{\alpha_1'}, \tag{5.15}$$

$$\left(\alpha_1'\right)^2 = \alpha_1^2 + |\beta_1|^2, \tag{5.16}$$

$$\left(\alpha_i'\right)^* = c\alpha_i^* + s\beta_i^*, \quad i = 2, \dots, p, \tag{5.17}$$

$$\left(\beta_i'\right)^* = c\beta_i^* - s^*\alpha_i^*, \quad i = 2, \dots, p. \tag{5.18}$$

Equations (5.15), (5.16), (5.17), and (5.18) summarize the square-root (SR) Givens rotation of a complex version since it requires a SR operation in the generic formulation. The SR operation may occupy a large area in a VLSI chip and many cycles may be required to complete such computations; consequently, the operation is slow. Proudler et al. [24] demonstrated that a finite-precision implementation of an SRF lattice algorithm achieved better numerical results than that of the conventional SR Givens rotation. Hsieh et al. [25] thus proposed a systematic way of generating a unified SRF Givens rotation to avoid the SR operation. In the remaining part of this section, the SRF Givens rotation developed in [25] is generalized to a complex form and extended to include a feedback mechanism, which is known to have a stabilizing effect when errors are made in the QRD-based RLS estimation, because of finite-precision effects [26, 27]. The generalized SRF Givens rotation with a feedback mechanism is then applied to develop the SRF QRD-LSL algorithms that are presented in Section 5.4.

By taking out a scaling factor from each row of the matrices on both sides of (5.14), the two rows, before and after the Givens rotation, are denoted, respectively, by

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_p \\ \beta_1 & \beta_2 & \dots & \beta_p \end{bmatrix} = \begin{bmatrix} \sqrt{k_a} & 0 \\ 0 & \sqrt{k_b} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \dots & a_p \\ b_1 & b_2 & \dots & b_p \end{bmatrix} \tag{5.19}$$

and

$$\begin{bmatrix} \alpha_1' \; \alpha_2' \; \ldots \; \alpha_p' \\ 0 \;\; \beta_2' \; \ldots \; \beta_p' \end{bmatrix} = \begin{bmatrix} \sqrt{k_a'} & 0 \\ 0 & \sqrt{k_b'} \end{bmatrix} \begin{bmatrix} a_1' \; a_2' \; \ldots \; a_p' \\ 0 \;\; b_2' \; \ldots \; b_p' \end{bmatrix}, \tag{5.20}$$

where k_a, k_b, k_a', and k_b' are the scaling factors resulting in SRF operations, and α_i' and β_i' are the updated α_i and β_i when β_1 is zeroed out. Replacing $\alpha_i = \sqrt{k_a}a_i$, $\alpha_i' = \sqrt{k_a'}a_i'$, $\beta_i = \sqrt{k_b}b_i$, $i = 1, \ldots, p$ and $\beta_i' = \sqrt{k_b'}b_i'$, $i = 2, \ldots, p$, in (5.15), (5.16), (5.17), and (5.18) leads to $c = \sqrt{k_a}a_1/\alpha_1'$, $s = \sqrt{k_b}b_1/\alpha_1'$, $a_1' = \alpha_1'/\sqrt{k_a'}$, $a_i' = k_a a_1 a_i + k_b b_1^* b_i/\sqrt{k_a'}\alpha_1'$, $i = 2, \ldots, p$, and $b_i' = \frac{\sqrt{k_a k_b}[a_1 b_i - b_1 a_i]}{\sqrt{k_b'}\alpha_1'}, i = 2, \ldots, p$, where $\alpha_1' = \sqrt{k_a a_1^2 + k_b|b_1|^2}$. Clearly, if $k_a' = (\alpha_1')^2/\mu^2$ and $k_b' = k_a k_b/\nu^2 (\alpha_1')^2 = k_a k_b/\mu^2\nu^2 k_a'$ are chosen, then the computation of a_1', a_i', and b_i' can avoid SR operation; μ and ν are parameters to be determined later. Substituting the chosen k_a' and k_b' into c, s, a_1', a_i', and, b_i' shown above yields $c = a_1/\mu \cdot \sqrt{k_a/k_a'}$, $s = b_1/\mu \cdot \sqrt{k_b/k_a'}$, $a_1' = \mu$, $a_i' = (k_a a_1 a_i + k_b b_1^* b_i)/\mu k_a', i = 2, \ldots, p$, and, $b_i' = \nu[a_1 b_i - b_1 a_i], i = 2, \ldots, p$. Evidently, the SR operations in a_1', a_i', and b_i' are eliminated, regardless of the values μ and ν.

Throughout this section, $\mu = 1$ (or $a_1' = 1$), $\nu = 1$, and $a_1 = 1$ were set such that the SRF results would be consistent with those of Gentleman [28] and McWhirter [29]. Accordingly, we have

$$k_a' = k_a + k_b|b_1|^2, \tag{5.21}$$

$$k_b' = \frac{k_a k_b}{k_a'} = \bar{c} \cdot k_b, \tag{5.22}$$

$$a_i' = \bar{c}a_i + \bar{s}^* b_i, \quad i = 2, \ldots, p, \tag{5.23}$$

$$b_i' = b_i - b_1 a_i, \quad i = 2, \ldots, p, \tag{5.24}$$

where

$$\bar{c} = \frac{k_a}{k_a'} \text{ and } \bar{s} = b_1 \cdot \frac{k_b}{k_a'} \tag{5.25}$$

are the defined generalized rotational parameters. Since $k_a = \alpha_1^2$ and $|\beta_1|^2$ corresponds to $\beta_1 \cdot \beta_1^* = k_b|b_1|^2$, we have $k_a' = k_a + k_b|b_1|^2 = \alpha_1^2 + |\beta_1|^2 = (\alpha_1')^2$. Therefore,

$$\bar{c} = \frac{k_a}{k_a'} = \frac{k_a}{(\alpha_1')^2} = c^2 \tag{5.26}$$

and $s = \sqrt{\frac{k_b}{k_a'}} \cdot b_1 = \sqrt{b_1 \cdot \bar{s}}$. The SRF Givens rotation with feedback mechanism can be derived by substituting (5.24) into (5.23):

$$a_i' = a_i(\bar{c} + \bar{s}^* b_1) + \bar{s}^* b_i' = a_i + \bar{s}^* b_i', \tag{5.27}$$

which is obtained using $\bar{c}+\bar{s}^*b_1 = \frac{k_a}{k'_a} + \frac{k_b b_1^*}{k'_a} b_1 = 1$. For notational convenience, taking the complex conjugate of both sides of (5.24) and (5.27) yields

$$\left(b'_i\right)^* = b_i^* - b_1^* a_i^*, \quad i = 2, \ldots, p, \tag{5.28}$$

$$\left(a'_i\right)^* = a_i^* + \bar{s} \cdot \left(b'_i\right)^*, \quad i = 2, \ldots, p. \tag{5.29}$$

Equations (5.21), (5.22), (5.25), (5.28), and (5.29) summarize the complex version of the *SRF Givens rotation with a feedback mechanism.*

5.4 SRF QRD-LSL Algorithms

This section develops the SRF QRD-LSL interpolation algorithm and the SRF QRD-LSL prediction algorithm. The latter algorithm is then extended to develop the SRF joint process estimation that utilizes information from the prediction lattice filter to generate the LS filtering estimate. More specifically, the joint process estimation problem displayed in Figure 5.1 is the optimal LS estimation of a process $d(k)$, called the desired response, from a related process $x(k)$, called the observations [2, 5].

The SRF QRD-LSL interpolation algorithm comprises six blocks: (a) FP block and BP block (summarized in Table 5.1); (b) IFP block and interpolation [Int(F)] block (summarized in Table 5.2); (c) IBP block and interpolation [Int(P)] block (summarized in Table 5.2) [30]. Overall, the SRF QRD-LSL interpolation algorithm has $\mathscr{O}[N]$ computational complexity per iteration without any SR operation. Notably, both FP and BP blocks, which constitute the SRF QRD-LSL prediction algorithm, are portions of the SRF QRD-LSL interpolation algorithm and must be used to compute both the conventional forward and backward prediction errors of order m [i.e., $e_{f,m}(k)$ and $e_{b,m}(k)$].

The SRF QRD-LSL interpolation algorithm performs adaptive filtering recursively in order and time. As an additional “future” stage is increased [i.e., $(p, f) \rightarrow (p, f+1)$], then both the IFP block and the Int(F) block, which are the QRD implementations of (5.11) and (5.6), respectively, must be used to compute the *a priori* interpolation error, $e^I_{p,f+1}(k-f-1)$. As an additional “past” stage is increased [i.e., $(p, f) \rightarrow (p+1, f)$], then both the IBP block and the Int(P) block, which are the QRD implementations of (5.10) and (5.5), respectively, must be used to compute the *a priori* interpolation error, $e^I_{p+1,f}(k-f)$. Throughout the chapter, the terms “ε” and “e” represent the *a posteriori* and *a priori* versions of estimation errors, respectively, whereas the term “$\bar{\varepsilon}$” represents the “*angle-normalized*” estimation error.

Table 5.1 SRF QRD-LSL algorithm and joint process estimation.

SRF QRD-LSL prediction and filtering
Initialization:
$\overline{\pi}_{f,m}(-1)=\overline{\pi}_{b,m}(-1)=e_{b,m-1}(-1)=0$, for order $m=1,2,\ldots,N$,
$\overline{p}_{m-1}(-1)=0$, for order $m=1,2,\ldots,N+1$,
$B_m(-2)=B_m(-1)=F_m(-1)=\delta$, for order $m=0,1\ldots,N$,
For $k=0,1,2\ldots$, set $e_{f,0}(k)=e_{b,0}(k)=x(k)$, $e_0(k)=d(k)$
For $k=-1,0,1,2\ldots$, set $\gamma_0(k)=1$
Prediction: For time $k=0,1,\ldots$, and prediction order $m=1,2,\ldots,N$.
FP block:
$B_{m-1}(k-1)=\lambda B_{m-1}(k-2)+\gamma_{m-1}(k-1)\lvert e_{b,m-1}(k-1)\rvert^2$
$\bar{c}_{b,m-1}(k-1)=\frac{\lambda B_{m-1}(k-2)}{B_{m-1}(k-1)}$
$\bar{s}_{b,m-1}(k-1)=\gamma_{m-1}(k-1)\frac{e^*_{b,m-1}(k-1)}{B_{m-1}(k-1)}$
$e_{f,m}(k)=e_{f,m-1}(k)-e_{b,m-1}(k-1)\overline{\pi}^*_{f,m}(k-1)$
$\overline{\pi}^*_{f,m}(k)=\overline{\pi}^*_{f,m}(k-1)+\bar{s}_{b,m-1}(k-1)e_{f,m}(k)$
$\gamma_m(k-1)=\bar{c}_{b,m-1}(k-1)\gamma_{m-1}(k-1)$
BP block:
$F_{m-1}(k)=\lambda F_{m-1}(k-1)+\gamma_{m-1}(k-1)\lvert e_{f,m-1}(k)\rvert^2$
$\bar{s}_{f,m-1}(k)=\gamma_{m-1}(k-1)\frac{e^*_{f,m-1}(k)}{F_{m-1}(k)}$
$e_{b,m}(k)=e_{b,m-1}(k-1)-e_{f,m-1}(k)\overline{\pi}^*_{b,m}(k-1)$
$\overline{\pi}^*_{b,m}(k)=\overline{\pi}^*_{b,m}(k-1)+\bar{s}_{f,m-1}(k)e_{b,m}(k)$
Joint process estimation: For time $k=0,1,\ldots$, and $m=1,2,\ldots,N+1$.
$B_{m-1}(k)=\lambda B_{m-1}(k-1)+\gamma_{m-1}(k)\lvert e_{b,m-1}(k)\rvert^2$
$\bar{c}_{b,m-1}(k)=\frac{\lambda B_{m-1}(k-1)}{B_{m-1}(k)}$
$\bar{s}_{b,m-1}(k)=\gamma_{m-1}(k)\frac{e^*_{b,m-1}(k)}{B_{m-1}(k)}$
$e_m(k)=e_{m-1}(k)-e_{b,m-1}(k)\overline{p}^*_{m-1}(k-1)$
$\overline{p}^*_{m-1}(k)=\overline{p}^*_{m-1}(k-1)+\bar{s}_{b,m-1}(k)e_m(k)$
$\gamma_m(k)=\bar{c}_{b,m-1}(k)\gamma_{m-1}(k)$

5.4.1 QRD based on interpolation

Before deriving the SRF QRD-LSL interpolation algorithm, we briefly describe a modified QR-decomposition for interpolation. It can be shown that a $(k+1)\times(k+1)$ orthogonal matrix $\overline{\mathbf{Q}}(k)$ can always be constructed from one of the C_N^f orthogonal basis sets described in Section 5.2 such that it applies a generalized orthogonal triangularization to $\mathbf{X}_{N+1}(k)$ in (5.3)

$$\overline{\mathbf{Q}}(k)\mathbf{X}_{N+1}(k)=\begin{bmatrix}\mathbf{Q}_{p,f}(k)\\ \mathbf{S}(k)\end{bmatrix}\mathbf{X}_{N+1}(k)=\begin{bmatrix}\mathbf{R}_{p,f}(k)\\ \mathbf{O}_{(k-N)\times(N+1)}\end{bmatrix}, \tag{5.30}$$

Table 5.2 Summary of SRF QRD-LSL interpolation algorithm.

SRF QRD-LSL interpolation algorithm

```
Initialization:
    $\overline{\Delta}_{f,m}(-1) = \overline{\Delta}_{b,m}(-1) = 0$, for order $m = 1, 2, \ldots, N+1$,
    $\overline{\rho}_{p,f}(-1) = 0$, for all $p$ and $f$,
    $I_{p,f}(k) = \delta$, for $k \leq -1$ and all $p$ and $f$,
    $B_m(-1,k) = F_m(-1,k) = \delta$, for $k \leq -1$ and order $m = 1, 2, \ldots, N+1$
    $e^I_{p,f}(k) = 0$ for $k \leq -1$ and all $p$ and $f$, and $\gamma_{0,0}(k) = 1$ for $k \leq -1$,
    For $k = 0, 1, 2, \ldots$, set $e^I_{0,0}(k) = x(k)$.
For time $k = 0, 1, \ldots$, starting with $p = f = 0$.
As $(p,f) \rightarrow (p, f+1)$:
  IFP block:
    $I_{p,f}(k-f-1) = \lambda I_{p,f}(k-f-2) + \gamma_{p,f}(k-f-1)|e^I_{p,f}(k-f-1)|^2$
    $\overline{c}_{I,N}(k-1) = \frac{\lambda I_{p,f}(k-f-2)}{I_{p,f}(k-f-1)}$ (needed only for deriving QRD-LSL prediction)
    $\overline{s}_{I,N}(k-1) = \gamma_{p,f}(k-f-1)\frac{e^{I*}_{p,f}(k-f-1)}{I_{p,f}(k-f-1)}$
    $e_{f,N+1}(k,k-f-1) = e_{f,N+1}(k) + e^I_{p,f}(k-f-1)\overline{\Delta}^*_{f,N+1}(k-1)$
    $\overline{\Delta}^*_{f,N+1}(k) = \overline{\Delta}^*_{f,N+1}(k-1) + \overline{s}_{I,N}(k-1)e_{f,N+1}(k)$
  Int(F) block:
    $F_{N+1}(k,k-f-1) = \lambda F_{N+1}(k-1,k-f-2)$
                        $+\gamma_{p,f}(k-f-1)|e_{f,N+1}(k,k-f-1)|^2$
    $\overline{c}'_{f,N+1}(k) = \frac{\lambda F_{N+1}(k-1,k-f-2)}{F_{N+1}(k,k-f-1)}$
    $\overline{s}'_{f,N+1}(k) = \gamma_{p,f}(k-f-1)\frac{e^*_{f,N+1}(k,k-f-1)}{F_{N+1}(k,k-f-1)}$
    $e^I_{p,f+1}(k-f-1) = e^I_{p,f}(k-f-1) - e_{f,N+1}(k,k-f-1)\overline{\rho}^*_{p,f+1}(k-1)$
    $\overline{\rho}^*_{p,f+1}(k) = \overline{\rho}^*_{p,f+1}(k-1) + \overline{s}'_{f,N+1}(k)e^I_{p,f+1}(k-f-1)$
    $\gamma_{p,f+1}(k-f-1) = \overline{c}'_{f,N+1}(k)\gamma_{p,f}(k-f-1)$
As $(p,f) \rightarrow (p+1, f)$:
  IBP block:
    $I_{p,f}(k-f) = \lambda I_{p,f}(k-f-1) + \gamma_{p,f}(k-f)|e^I_{p,f}(k-f)|^2$
    $\overline{s}_{I,N}(k) = \gamma_{p,f}(k-f)\frac{e^{I*}_{p,f}(k-f)}{I_{p,f}(k-f)}$
    $e_{b,N+1}(k,k-f) = e_{b,N+1}(k) + e^I_{p,f}(k-f)\overline{\Delta}^*_{b,N+1}(k-1)$
    $\overline{\Delta}^*_{b,N+1}(k) = \overline{\Delta}^*_{b,N+1}(k-1) + \overline{s}_{I,N}(k)e_{b,N+1}(k)$
  Int(P) block:
    $B_{N+1}(k,k-f) = \lambda B_{N+1}(k-1,k-f-1) + \gamma_{p,f}(k-f)|e_{b,N+1}(k,k-f)|^2$
    $\overline{c}'_{b,N+1}(k) = \frac{\lambda B_{N+1}(k-1,k-f-1)}{B_{N+1}(k,k-f)}$
    $\overline{s}'_{b,N+1}(k) = \gamma_{p,f}(k-f)\frac{e^*_{b,N+1}(k,k-f)}{B_{N+1}(k,k-f)}$
    $e^I_{p+1,f}(k-f) = e^I_{p,f}(k-f) - e_{b,N+1}(k,k-f)\overline{\rho}^*_{p+1,f}(k-1)$
    $\overline{\rho}^*_{p+1,f}(k) = \overline{\rho}^*_{p+1,f}(k-1) + \overline{s}'_{b,N+1}(k)e^I_{p+1,f}(k-f)$
    $\gamma_{p+1,f}(k-f) = \overline{c}'_{b,N+1}(k)\gamma_{p,f}(k-f)$
```

where $\mathbf{Q}_{p,f}(k)$ contains the first $(N+1)$ rows of $\overline{\mathbf{Q}}(k)$, whereas $\mathbf{S}(k)$ contains the remaining rows; $\mathbf{O}_{(k-N)\times(N+1)}$ is a null matrix of order $(k-N)\times(N+1)$. Since C_N^f possible sequences can be used, the $(N+1)\times(N+1)$ matrix $\mathbf{R}_{p,f}(k)$ in (5.30) can display C_N^f different forms, all of which contain one $f \times f$ lower triangular matrix (upper-left submatrix of $\mathbf{R}_{p,f}(k)$) and one $p \times p$ upper triangular matrix (lower-right submatrix of $\mathbf{R}_{p,f}(k)$) with zero elements filling the $(f+1)$st row, except for the $(f+1, f+1)$st element. More specifically, our results indicate that $\mathbf{R}_{p,f}(k)$ using the BFBFBF... sequence can be shown to be

$$\mathbf{R}_{p,f}(k) = \left[\begin{array}{ccccccccc}
\gamma_{F,1} & 0 & \dots & 0 & \times & \times & \dots & \dots & \times \\
\times & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \gamma_{F,2} & 0 & \vdots & \vdots & \ddots & \ddots & \vdots \\
\times & \dots & \dots & \gamma_{F,3} & \times & \times & \dots & \dots & \times \\
0 & \dots & \dots & 0 & I_{p,f}^{\frac{1}{2}}(k-f) & 0 & \dots & \dots & 0 \\
\times & \cdots & \cdots & \times & \times & \gamma_{B,1} & \times & \dots & \times \\
\vdots & \ddots & \ddots & \vdots & \vdots & 0 & \gamma_{B,2} & \ddots & \vdots \\
\vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \ddots & \times \\
\times & \dots & \dots & \times & \times & 0 & \dots & 0 & \gamma_{B,3}
\end{array}\right] \tag{5.31}$$

in which $\gamma_{F,1} = F_N^{\frac{1}{2}}(k,k-f)$, $\gamma_{F,2} = F_4^{\frac{1}{2}}(k-f+2,k-f)$, $\gamma_{F,3} = F_2^{\frac{1}{2}}(k-f+1,k-f)$, $\gamma_{B,1} = B_1^{\frac{1}{2}}(k-f,k-f)$, $\gamma_{B,2} = B_3^{\frac{1}{2}}(k-f+1,k-f)$, $\gamma_{B,3} = B_{N-1}^{\frac{1}{2}}(k-1,k-f)$, where $F_m^{\frac{1}{2}}(k-j,k-f)$ and $B_m^{\frac{1}{2}}(k-j,k-f)$ are the square roots of the minimum sum of mth-*order* intermediate forward and backward prediction error squares, respectively, whereas $I_{p,f}^{\frac{1}{2}}(k-f)$ is the square root of the minimum sum of (p,f)th-*order* interpolation error square; the symbol × denotes an element whose value is not of direct interest. We refer to the result in (5.30) as the *modified QR-decomposition for interpolation* and refer to the form in $\mathbf{R}_{p,f}(k)$ of (5.31) as the *standard lower/upper (LU) triangular form for* a (p,f)th-*order interpolator* based on the QR-decomposition. For example, a special case of (5.31) identified by sequence BFBF for a $(2,2)$th-*order* LSL interpolator can be expressed as

$$\mathbf{R}_{2,2}(k) = \left[\begin{array}{ccccc}
F_4^{\frac{1}{2}}(k,k-2) & 0 & \times & \times & \times \\
\times & F_2^{\frac{1}{2}}(k-1,k-2) & \times & \times & \times \\
0 & 0 & I_{2,2}^{\frac{1}{2}}(k-2) & 0 & 0 \\
\times & \times & \times & B_1^{\frac{1}{2}}(k-2,k-2) & \times \\
\times & \times & \times & 0 & B_3^{\frac{1}{2}}(k-1,k-2)
\end{array}\right] \tag{5.32}$$

Moreover, setting $(p,f) = (N,0)$, the lower-right $p \times p$ upper triangular submatrix of $\mathbf{R}_{p,f}(k)$ in (5.31) yields the following well-known upper triangular matrix after an unitary matrix is used to develop the QRD-LSL prediction algorithm.

$$\mathbf{R}_{N,0}(k-1)=\begin{bmatrix} B_0^{\frac{1}{2}}(k-1) & \times & \times & \dots & \times \\ & B_1^{\frac{1}{2}}(k-1) & \times & \dots & \times \\ & & B_2^{\frac{1}{2}}(k-1) & \dots & \times \\ & \mathbf{O} & & \ddots & \vdots \\ & & & & B_{N-1}^{\frac{1}{2}}(k-1) \end{bmatrix} \tag{5.33}$$

5.4.2 SRF QRD-LSL interpolation algorithm

The SRF QRD-LSL interpolation algorithm is derived according to the following seven stages. In each stage, either a single Givens rotation or a sequence of Givens rotations is applied. In the derivation, matrices are represented in uppercase boldface type and column vectors in lowercase boldface type, whereas scalars appear in plain text type. The dimensions of matrices and vectors appear as subscripts. For example, $\mathbf{A}_{m\times k}$ and $\mathbf{p}_m$ represent a $m\times k$ matrix and a $m\times 1$ column vector, respectively.

1. We first write (5.3), at time $k-2$, as $\boldsymbol{\varepsilon}^I_{p,f}(k-f-2)=\mathbf{X}_{N+1}(k-2)\mathbf{b}_{p,f}(k-f-2)$ and pre-multiply both sides of the equation by $\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)$, where $\boldsymbol{\Lambda}(k-2)=diag[\lambda^{k-2},\lambda^{k-3},\dots,1]$ is the $(k-1)\times(k-1)$ exponential weighting matrix in which $0\ll\lambda\leq 1$ is the forgetting factor. Next, we apply a sequence of N Givens rotations that define the $(k-1)\times(k-1)$ orthogonal matrix $\overline{\mathbf{Q}}(k-2)=\begin{bmatrix}\mathbf{Q}_{p,f}(k-2)\\ \mathbf{S}(k-2)\end{bmatrix}$ such that matrix $\mathbf{Q}_{p,f}(k-2)$ transforms data matrix $\mathbf{X}_{N+1}(k-2)$ into matrix

$$\mathbf{R}_{p,f}(k-2)=\begin{bmatrix}\mathbf{L}_{f\times f}(k-2) & \mathbf{p}_f(k-2) & \mathbf{B}_{f\times p}(k-2)\\ \mathbf{0}_f^{\mathrm{T}} & I_{p,f}^{\frac{1}{2}}(k-f-2) & \mathbf{0}_p^{\mathrm{T}}\\ \mathbf{A}_{p\times f}(k-2) & \mathbf{p}_p(k-2) & \mathbf{U}_{p\times p}(k-2)\end{bmatrix}, \tag{5.34}$$

which is in the standard LU triangular form for a (p,f)th-*order* interpolator as shown in (5.31) with $x(k-2)$ being the most recent data sample used. Notably, $\mathbf{L}_{f\times f}(k-2)$ is the $f\times f$ lower triangular matrix and $\mathbf{U}_{p\times p}(k-2)$ is the $p\times p$ upper triangular matrix. We may thus write

$$\begin{aligned}&\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)\overline{\mathbf{Q}}(k-2)\boldsymbol{\varepsilon}^I_{p,f}(k-f-2)\\ &\qquad=\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)\begin{bmatrix}\mathbf{R}_{p,f}(k-2)\\ \mathbf{O}_{(k-N-2)\times(N+1)}\end{bmatrix}\mathbf{b}_{p,f}(k-f-2).\end{aligned} \tag{5.35}$$

2. We may apply the transformations produced by $\begin{bmatrix}1 & \\ & \overline{\mathbf{Q}}(k-2)\end{bmatrix}$ and $\overline{\mathbf{Q}}(k-2)$ to the data vectors $\mathbf{x}_k(k-1)=[x^*(0),x^*(1),\dots,x^*(k-1)]^{\mathrm{T}}$ and $\mathbf{x}_{k-1}(k-N-3)=[0,\dots,0,x^*(0),\dots,x^*(k-N-3)]^{\mathrm{T}}$, respectively. The two data vectors containing the next future and next past data samples for consideration, respectively, will serve the purpose of deriving the order-updated recursions for the interpolation error in the later stages. We may thus write

$$\begin{aligned}&\begin{bmatrix}1 & \\ & \overline{\mathbf{Q}}(k-2)\end{bmatrix}\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\mathbf{x}_k(k-1)\\ &=[\lambda^{\frac{k-1}{2}}x^*(0),\mathbf{f}_f^{\mathrm{T}}(k-1),\Delta_{f,N+1}(k-1),\mathbf{f}_p^{\mathrm{T}}(k-1),\mathbf{f}_{k-N-2}^{\mathrm{T}}(k-1)]^{\mathrm{T}}\end{aligned} \tag{5.36}$$

and

$$\begin{aligned}&\overline{\mathbf{Q}}(k-2)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)\mathbf{x}_{k-1}(k-N-3)\\ &=[\mathbf{b}_f^{\mathrm{T}}(k-2),\Delta_{b,N+1}(k-2),\mathbf{b}_p^{\mathrm{T}}(k-2),\mathbf{b}_{k-N-2}^{\mathrm{T}}(k-2)]^{\mathrm{T}},\end{aligned} \tag{5.37}$$

where $\Delta_{f,N+1}(k-1)$ and $\Delta_{b,N+1}(k-2)$ are auxiliary parameters which will be used later to obtain the intermediate prediction errors. All vectors appearing on the right-hand-side of (5.36) and (5.37) are defined merely for convenience of presentation; their elements are not of direct interest. By appending both transformed data vectors obtained in (5.36) and (5.37), respectively, as the leftmost column and rightmost column of the matrix defined by $\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)$ $\begin{bmatrix}\mathbf{R}_{p,f}(k-2)\\ \mathbf{O}_{(k-N-2)\times(N+1)}\end{bmatrix}$ in (5.35) [see the box in (5.38)], together with the new data sample vector for time k at the bottom row, we obtain the following expanded matrix $\mathbf{D}(k)$, which can be written as

$$\left[\begin{array}{c|ccc|c}
\lambda^{\frac{k}{2}}x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0\\ \cline{2-4}
\lambda^{\frac{1}{2}}\mathbf{f}_f(k-1) & \lambda^{\frac{1}{2}}\mathbf{L}_{f\times f}(k-2) & \lambda^{\frac{1}{2}}\mathbf{p}_f(k-2) & \lambda^{\frac{1}{2}}\mathbf{B}_{f\times p}(k-2) & \lambda^{\frac{1}{2}}\mathbf{b}_f(k-2)\\
\lambda^{\frac{1}{2}}\Delta_{f,N+1}(k-1) & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}}I_{p,f}^{\frac{1}{2}}(k-f-2) & \mathbf{0}_p^{\mathrm{T}} & \lambda^{\frac{1}{2}}\Delta_{b,N+1}(k-2)\\
\lambda^{\frac{1}{2}}\mathbf{f}_p(k-1) & \lambda^{\frac{1}{2}}\mathbf{A}_{p\times f}(k-2) & \lambda^{\frac{1}{2}}\mathbf{p}_p(k-2) & \lambda^{\frac{1}{2}}\mathbf{U}_{p\times p}(k-2) & \lambda^{\frac{1}{2}}\mathbf{b}_p(k-2)\\
\lambda^{\frac{1}{2}}\mathbf{f}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}}\mathbf{b}_{k-N-2}(k-2)\\ \cline{2-4}
x^*(k) & \ldots & x^*(k-f-1) & \ldots & x^*(k-N-2)
\end{array}\right]. \tag{5.38}$$

3. Next, we apply a sequence of Givens rotations to annihilate all elements in the bottom row of the matrix $\mathbf{D}(k)$ except for the $(k+1,1)$th, $(k+1,f+2)$th, and $(k+1,N+3)$th elements. These rotations include an appropriate combination of a sequence of f Givens rotations proceeding leftwards from the $(k+1,f+1)$th element to the $(k+1,2)$th element and a sequence of p Givens rotations proceeding rightwards from the $(k+1,f+3)$th element to the $(k+1,N+2)$th element with the order in which the elements are annihilated in accordance with the sequencing chosen (e.g., BFBFBF...) to preserve the standard LU triangular form for interpolator in the transformed matrix. Note that any sequencing between F and B is permissible. Accordingly, there are C_N^f possible sequences. For example, if the sequence BFBFBF... is chosen, then the elements at the bottom row of $\mathbf{D}(k)$ in the following order: $(k+1,f+3)$, $(k+1,f+1)$, $(k+1,f+4)$, $(k+1,f)$, $(k+1,f+5)$, $(k+1,f-1)$, ..., $(k+1,N+2)$, $(k+1,2)$ will be annihilated successively. Such a sequence of N Givens rotations defines the $(k+1)\times(k+1)$ orthogonal matrix $\mathbf{L}(k)$ that transforms the matrix $\mathbf{D}(k)$ to the matrix $\mathbf{E}(k)$ as follows:

$$\mathbf{L}(k)\mathbf{D}(k)=\mathbf{E}(k), \tag{5.39}$$

where $\mathbf{E}(k)$ can be written as

$$\begin{bmatrix} \lambda^{\frac{k}{2}} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ \lambda^{\frac{1}{2}} \Delta_{f,N+1}(k-1) & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} I_{p,f}^{\frac{1}{2}}(k-f-2) & \mathbf{0}_p^{\mathrm{T}} & \lambda^{\frac{1}{2}} \Delta_{b,N+1}(k-2) \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \lambda^{\frac{1}{2}} \mathbf{f}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}_{k-N-2}(k-2) \\ \overline{\varepsilon}_{f,N+1}^*(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & \overline{\varepsilon}^* & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}_{b,N+1}^*(k-1,k-f-1) \end{bmatrix}.$$

Since $x^*(k-f-1)$ at the bottom row of $\mathbf{D}(k)$ is the present data sample to be estimated by its p past and f future neighboring data samples, it was not annihilated by $\mathbf{L}(k)$ in the above transformation. Consequently, a non-zero quantity, $\overline{\varepsilon}^*$, was generated at the bottom row of $\mathbf{E}(k)$. By using the fact that orthogonal rotations are norm preserving, one can show that $\overline{\varepsilon}^*$ is actually the complex conjugate of the (p,f)th-*order* angle-normalized interpolation error, $\overline{\varepsilon}_{p,f}^{I^*}(k-f-1)$, with $x(k-1)$ being the most recent data sample used. Notably, due to the non-zero quantity $\overline{\varepsilon}^*$ at the bottom row of $\mathbf{E}(k)$, the $(k+1,1)$th and $(k+1,N+3)$th elements of $\mathbf{E}(k)$ in (5.39) are $\overline{\varepsilon}_{f,N+1}^*(k,k-f-1)$ and $\overline{\varepsilon}_{b,N+1}^*(k-1,k-f-1)$, which are the angle-normalized intermediate forward and backward prediction errors of order $N+1$, respectively.

4. Both the conventional angle-normalized FP error, $\overline{\varepsilon}_{f,N+1}(k)$, and the angle-normalized backward prediction error, $\overline{\varepsilon}_{b,N+1}(k-1)$, appear if the $(k+1,f+2)$th element of $\mathbf{E}(k)$ is annihilated. This can be accomplished by applying a single Givens rotation to $\mathbf{E}(k)$. We may thus write

$$\mathbf{J}_{I,N}(k-1)\mathbf{E}(k) = \mathbf{F}(k), \tag{5.40}$$

where $\mathbf{J}_{I,N}(k-1) = \begin{bmatrix} \mathbf{I}_{f+1} & & & \\ & c_{I,N}(k-1) & & s_{I,N}^*(k-1) \\ & & \mathbf{I}_{k-f-2} & \\ & -s_{I,N}(k-1) & & c_{I,N}(k-1) \end{bmatrix}$ and $\mathbf{F}(k)$ can be written as

$$\begin{bmatrix} \lambda^{\frac{k}{2}} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ \Delta_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & I_{p,f}^{\frac{1}{2}}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & \Delta_{b,N+1}(k-1) \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \lambda^{\frac{1}{2}} \mathbf{f}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}_{k-N-2}(k-2) \\ \overline{\varepsilon}_{f,N+1}^*(k) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}_{b,N+1}^*(k-1) \end{bmatrix}$$

5. The annihilation in (5.40) has the effect of computing both the intermediate forward and backward prediction errors from the conventional forward and backward prediction errors. By taking out a scaling factor [i.e., the square root term of each row of $\mathbf{E}(k)$ and $\mathbf{F}(k)$] from each row of matrices $\mathbf{E}(k)$ and $\mathbf{F}(k)$ [31], the rows before and after the Givens rotation in (5.40) is denoted, respectively, by

$$
\begin{aligned}
\mathbf{E}(k) &= \mathbf{E}_1(k)\mathbf{E}_2(k) \\
&= diag\left[\sqrt{\lambda^k}, \ldots, \sqrt{\lambda I_{p,f}(k-f-2)}, \ldots, \sqrt{\gamma_{p,f}(k-f-1)}\right]
\end{aligned}
$$

$$
\begin{bmatrix}
x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\
\bar{\mathbf{f}}_f(k) & \overline{\mathbf{L}}_{f\times f}(k-1) & \bar{\mathbf{p}}_f(k-1) & \overline{\mathbf{B}}_{f\times p}(k-1) & \bar{\mathbf{b}}_f(k-1) \\
\overline{\Delta}_{f,N+1}(k-1) & \mathbf{0}_f^{\mathrm{T}} & 1 & \mathbf{0}_p^{\mathrm{T}} & \overline{\Delta}_{b,N+1}(k-2) \\
\bar{\mathbf{f}}_p(k) & \overline{\mathbf{A}}_{p\times f}(k-1) & \bar{\mathbf{p}}_p(k-1) & \overline{\mathbf{U}}_{p\times p}(k-1) & \bar{\mathbf{b}}_p(k-1) \\
\bar{\mathbf{f}}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \bar{\mathbf{b}}_{k-N-2}(k-2) \\
e^*_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & e^{I^*}_{p,f}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+1}(k-1,k-f-1)
\end{bmatrix}, \qquad (5.41)
$$

and

$$
\begin{aligned}
\mathbf{F}(k) &= \mathbf{F}_1(k)\mathbf{F}_2(k) \\
&= diag\left[\sqrt{\lambda^k}, \ldots, \sqrt{I_{p,f}(k-f-1)}, \ldots, \sqrt{\gamma'_{p,f}(k-f-1)}\right]
\end{aligned}
$$

$$
\begin{bmatrix}
x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\
\bar{\mathbf{f}}_f(k) & \overline{\mathbf{L}}_{f\times f}(k-1) & \bar{\mathbf{p}}_f(k-1) & \overline{\mathbf{B}}_{f\times p}(k-1) & \bar{\mathbf{b}}_f(k-1) \\
\overline{\Delta}_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & 1 & \mathbf{0}_p^{\mathrm{T}} & \overline{\Delta}_{b,N+1}(k-1) \\
\bar{\mathbf{f}}_p(k) & \overline{\mathbf{A}}_{p\times f}(k-1) & \bar{\mathbf{p}}_p(k-1) & \overline{\mathbf{U}}_{p\times p}(k-1) & \bar{\mathbf{b}}_p(k-1) \\
\bar{\mathbf{f}}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \bar{\mathbf{b}}_{k-N-2}(k-2) \\
e^*_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+1}(k-1)
\end{bmatrix}. \qquad (5.42)
$$

Some elements of the matrices in (5.41) and (5.42) and the corresponding elements of the matrices in (5.19) and (5.20), respectively, can be related as follows: $k_a = \lambda I_{p,f}(k-f-2)$, $k_b = \gamma_{p,f}(k-f-1)$, $a_1 = 1$, $a_2 = \overline{\Delta}_{f,N+1}(k-1)$, $a_3 = \overline{\Delta}_{b,N+1}(k-2)$, $b_1 = e^{I^*}_{p,f}(k-f-1)$, $b_2 = e^*_{f,N+1}(k,k-f-1)$, $b_3 = e^*_{b,N+1}(k-1,k-f-1)$, and $k'_a = I_{p,f}(k-f-1)$, $k'_b = \gamma'_{p,f}(k-f-1)$, $a'_1 = 1$, $a'_2 = \overline{\Delta}_{f,N+1}(k)$, $a'_3 = \overline{\Delta}_{b,N+1}(k-1)$, $b'_1 = 0$, $b'_2 = e^*_{f,N+1}(k)$, $b'_3 = e^*_{b,N+1}(k-1)$. Notably, "e" represents the *a priori* estimation error. Substituting k_a, k'_a, k_b, b_1, a_2, a'_2, b_2, b'_2, a_3, a'_3, b_3, and b'_3 into (5.21), (5.25), (5.28), and (5.29) (by letting $i = 2$ and $i = 3$) yields

$$I_{p,f}(k-f-1) = \lambda I_{p,f}(k-f-2) + \gamma_{p,f}(k-f-1)|e^I_{p,f}(k-f-1)|^2, \quad (5.43)$$

$$\bar{s}_{I,N}(k-1) = \gamma_{p,f}(k-f-1)\frac{e^{I^*}_{p,f}(k-f-1)}{I_{p,f}(k-f-1)}, \quad (5.44)$$

$$e_{f,N+1}(k) = e_{f,N+1}(k,k-f-1) - e^I_{p,f}(k-f-1)\overline{\Delta}^*_{f,N+1}(k-1), \quad (5.45)$$

$$\overline{\Delta}^*_{f,N+1}(k) = \overline{\Delta}^*_{f,N+1}(k-1) + \bar{s}_{I,N}(k-1)e_{f,N+1}(k), \quad (5.46)$$

$$e_{b,N+1}(k) = e_{b,N+1}(k,k-f) - e^I_{p,f}(k-f)\overline{\Delta}^*_{b,N+1}(k-1), \quad (5.47)$$

$$\overline{\Delta}^*_{b,N+1}(k) = \overline{\Delta}^*_{b,N+1}(k-1) + \bar{s}_{I,N}(k)e_{b,N+1}(k). \quad (5.48)$$

Note that the *a priori* IFP error $e_{f,N+1}(k,k-f-1)$ in (5.45) is still unknown whereas the *a priori* FP error $e_{f,N+1}(k)$ in (5.45) has already been computed by using the SRF QRD-LSL prediction algorithm (see Table 5.1), which will be derived in Section 5.4.3. In order to compute the *a priori* IFP error, we "reverse" (5.45) in formulation such that, given the *a priori* FP error, the *a priori* IFP error can be computed as

$$e_{f,N+1}(k,k-f-1) = e_{f,N+1}(k) + e^{I}_{p,f}(k-f-1)\overline{\Delta}^{*}_{f,N+1}(k-1). \tag{5.49}$$

Similarly, (5.47) can be reformulated as

$$e_{b,N+1}(k,k-f) = e_{b,N+1}(k) + e^{I}_{p,f}(k-f)\overline{\Delta}^{*}_{b,N+1}(k-1), \tag{5.50}$$

which also corresponds to the "reversed" formulation. Notably, (5.49) and (5.50) correspond to (5.11) and (5.10), respectively, and these computed intermediate prediction errors are then used to compute the order-updated interpolation errors in the following stages, as $(p,f) \rightarrow (p,f+1)$ and as $(p,f) \rightarrow (p+1,f)$.

6. To obtain the order-updated interpolation error of order $(p,f+1)$ from that of the current order (p,f) as an additional *future* data sample is used, we proceed with by transforming the matrix $\mathbf{E}(k)$ into the standard LU triangular form for a $(p,f+1)$st-*order* interpolator. This transformation can be achieved by initially applying an $(k+1)\times(k+1)$ orthogonal matrix $\mathbf{P}(k)$ to $\mathbf{E}(k)$. The matrix $\mathbf{P}(k)$ represents the combined transformation produced by a sequence of $(k-N-2)$ Givens rotations, which has the effect of annihilating all the $(k-N-2)$ elements of vector $\lambda^{\frac{1}{2}}\mathbf{f}_{k-N-2}(k-1)$ in the first column of matrix $\mathbf{E}(k)$. We may thus write

$$\mathbf{P}(k)\mathbf{E}(k) = \mathbf{G}(k), \tag{5.51}$$

where $\mathbf{G}(k)$ can be written as

$$\begin{bmatrix} \lambda^{\frac{1}{2}} F^{\frac{1}{2}}_{N+1}(k-1) & \mathbf{0}^{\mathrm{T}}_{f} & 0 & \mathbf{0}^{\mathrm{T}}_{p} & \times \\ \mathbf{f}_{f}(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_{f}(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_{f}(k-1) \\ \lambda^{\frac{1}{2}}\Delta_{f,N+1}(k-1) & \mathbf{0}^{\mathrm{T}}_{f} & \lambda^{\frac{1}{2}} I^{\frac{1}{2}}_{p,f}(k-f-2) & \mathbf{0}^{\mathrm{T}}_{p} & \lambda^{\frac{1}{2}}\Delta_{b,N+1}(k-2) \\ \mathbf{f}_{p}(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_{p}(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_{p}(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}}\mathbf{b}'_{k-N-2}(k-2) \\ \overline{\varepsilon}^{*}_{f,N+1}(k,k-f-1) & \mathbf{0}^{\mathrm{T}}_{f} & \overline{\varepsilon}^{I*}_{p,f}(k-f-1) & \mathbf{0}^{\mathrm{T}}_{p} & \overline{\varepsilon}^{*}_{b,N+1}(k-1,k-f-1) \end{bmatrix}$$

where $\lambda F_{N+1}(k-1) = \lambda^{k}|x(0)|^{2} + \|\lambda^{\frac{1}{2}}\mathbf{f}_{k-N-2}(k-1)\|^{2}$ is the minimum weighted sum of the FP error square. This transformation is followed by rotating the element $\lambda^{\frac{1}{2}}\Delta_{f,N+1}(k-1)$ in the first column of matrix $\mathbf{G}(k)$ into the $(1,f+2)$th element of the matrix such that the resulting matrix $\mathbf{C}(k)$ in (5.52) will be in standard LU triangular form that can be used to compute the order-updated interpolation error. We may thus write

$$\begin{bmatrix} c_{\Delta}(k-1) & & s_{\Delta}^{*}(k-1) & \\ & \mathbf{I}_f & & \\ -s_{\Delta}(k-1) & & c_{\Delta}(k-1) & \\ & & & \mathbf{I}_{k-f-1} \end{bmatrix} \mathbf{G}(k) = \mathbf{C}(k), \tag{5.52}$$

where $\mathbf{C}(k)$ can be written as

$$\begin{bmatrix} \gamma_F & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} \rho_{p,f+1}(k-1) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \gamma_I & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}'_{k-N-2}(k-2) \\ \overline{\varepsilon}^{*}_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & \overline{\varepsilon}^{I*}_{p,f}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}^{*}_{b,N+1}(k-1,k-f-1) \end{bmatrix}$$

in which $\gamma_I = \lambda^{\frac{1}{2}} I_{p,f+1}^{\frac{1}{2}}(k-f-2)$; the $(1,1)$th element of $\mathbf{C}(k)$, $\gamma_F = \lambda^{\frac{1}{2}} F_{N+1}^{\frac{1}{2}}$ $(k-1,k-f-2)$, is the square root of the minimum weighted sum of the IFP error square and $\rho_{p,f+1}(k-1)$ is the interpolation auxiliary parameter that will be used in stage 7. Notably, the $(f+2,f+2)$th element of $\mathbf{C}(k)$ becomes $\lambda^{\frac{1}{2}} I_{p,f+1}^{\frac{1}{2}}(k-f-2)$. Clearly, the upper-left $(N+2)\times(N+2)$ submatrix of $\mathbf{C}(k)$ denotes the standard LU triangular form for a $(p,f+1)$st order interpolator and is used in the next stage to compute the order-updated interpolation error of order $(p,f+1)$.

7. We are now positioned to obtain the order-updated recursion $(p,f) \to (p,f+1)$ for the interpolation error as an additional future data sample is used by applying one single Givens rotation to $\mathbf{C}(k)$ so as to annihilate $\overline{\varepsilon}^{*}_{f,N+1}(k,k-f-1)$ at the bottom row of $\mathbf{C}(k)$. We thus obtain

$$\mathbf{J}'_{F,N+1}(k)\mathbf{C}(k) = \mathbf{J}(k), \tag{5.53}$$

where $\mathbf{J}'_{F,N+1}(k) = \begin{bmatrix} c'_{f,N+1}(k) & & s'^{*}_{f,N+1}(k) \\ & \mathbf{I}_{k-1} & \\ -s'_{f,N+1}(k) & & c'_{f,N+1}(k) \end{bmatrix}$ and $\mathbf{J}(k)$ is

$$\begin{bmatrix} F_{N+1}^{\frac{1}{2}}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & \rho_{p,f+1}(k) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} I_{p,f+1}^{\frac{1}{2}}(k-f-2) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}'_{k-N-2}(k-2) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \overline{\varepsilon}^{I*}_{p,f+1}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}^{*}_{b,N+2}(k,k-f-1) \end{bmatrix}.$$

Similar to the procedure in (5.41) and (5.42), scalar factors in the first row and the bottom row in matrices $\mathbf{C}(k)$ and $\mathbf{J}(k)$ can be taken out:

$$\begin{aligned}\mathbf{C}(k) &= \mathbf{C}_1(k)\mathbf{C}_2(k)\\ &= diag\left[\sqrt{\lambda F_{N+1}(k-1,k-f-2)},\ldots,\sqrt{\gamma_{p,f}(k-f-1)}\right]\end{aligned}$$
$$\begin{bmatrix} 1 & \mathbf{0}_f^{\mathrm{T}} & \overline{\rho}_{p,f+1}(k-1) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \bar{\mathbf{f}}_f(k) & \overline{\mathbf{L}}_{f\times f}(k-1) & \overline{\mathbf{p}}_f(k-1) & \overline{\mathbf{B}}_{f\times p}(k-1) & \overline{\mathbf{b}}_f(k-1) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \times & \mathbf{0}_p^{\mathrm{T}} & \times \\ \bar{\mathbf{f}}_p(k) & \overline{\mathbf{A}}_{p\times f}(k-1) & \overline{\mathbf{p}}_p(k-1) & \overline{\mathbf{U}}_{p\times p}(k-1) & \overline{\mathbf{b}}_p(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \overline{\mathbf{b}}'_{k-N-2}(k-2) \\ e^*_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & e^{I*}_{p,f}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+1}(k-1,k-f-1) \end{bmatrix}, \tag{5.54}$$

and

$$\begin{aligned}\mathbf{J}(k) &= \mathbf{J}_1(k)\mathbf{J}_2(k)\\ &= diag\left[\sqrt{F_{N+1}(k,k-f-1)},\ldots,\sqrt{\gamma_{p,f+1}(k-f-1)}\right]\end{aligned}$$
$$\begin{bmatrix} 1 & \mathbf{0}_f^{\mathrm{T}} & \overline{\rho}_{p,f+1}(k) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \bar{\mathbf{f}}_f(k) & \overline{\mathbf{L}}_{f\times f}(k-1) & \overline{\mathbf{p}}_f(k-1) & \overline{\mathbf{B}}_{f\times p}(k-1) & \overline{\mathbf{b}}_f(k-1) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \times & \mathbf{0}_p^{\mathrm{T}} & \times \\ \bar{\mathbf{f}}_p(k) & \overline{\mathbf{A}}_{p\times f}(k-1) & \overline{\mathbf{p}}_p(k-1) & \overline{\mathbf{U}}_{p\times p}(k-1) & \overline{\mathbf{b}}_p(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \overline{\mathbf{b}}'_{k-N-2}(k-2) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & e^{I*}_{p,f+1}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+2}(k,k-f-1) \end{bmatrix}. \tag{5.55}$$

Some elements of matrices $\mathbf{C}_1(k)$ and $\mathbf{C}_2(k)$ in (5.54) and matrices $\mathbf{J}_1(k)$ and $\mathbf{J}_2(k)$ in (5.55) and the corresponding elements of the matrices in (5.19) and (5.20), respectively, can be related as follows: $k_a = \lambda F_{N+1}(k-1,k-f-2)$, $k_b = \gamma_{p,f}(k-f-1)$, $a_1 = 1$, $a_2 = \overline{\rho}_{p,f+1}(k-1)$, $b_1 = e^*_{f,N+1}(k,k-f-1)$, $b_2 = e^{I*}_{p,f}(k-f-1)$, and $k'_a = F_{N+1}(k,k-f-1)$, $k'_b = \gamma_{p,f+1}(k-f-1)$, $a'_1 = 1$, $a'_2 = \overline{\rho}_{p,f+1}(k)$, $b'_1 = 0$, $b'_2 = e^{I*}_{p,f+1}(k-f-1)$. Substituting k_a, k'_a, k_b, b_1, a_2, a'_2, b_2, and b'_2 into (5.21), (5.22), (5.25), (5.28), and (5.29) (by letting $i=2$) yields

$$\begin{aligned}F_{N+1}(k,k-f-1) = \lambda F_{N+1}(k-1,k-f-2)&\\ +\gamma_{p,f}(k-f-1)|e_{f,N+1}(k,k-f-1)|^2&, \end{aligned}\tag{5.56}$$

$$\overline{c}'_{f,N+1}(k) = \frac{\lambda F_{N+1}(k-1,k-f-2)}{F_{N+1}(k,k-f-1)}, \tag{5.57}$$

$$\overline{s}'_{f,N+1}(k) = \gamma_{p,f}(k-f-1)\frac{e^*_{f,N+1}(k,k-f-1)}{F_{N+1}(k,k-f-1)}, \tag{5.58}$$

$$e^I_{p,f+1}(k-f-1) = e^I_{p,f}(k-f-1) - e_{f,N+1}(k,k-f-1)\overline{\rho}^*_{p,f+1}(k-1), \tag{5.59}$$

$$\overline{\rho}^*_{p,f+1}(k) = \overline{\rho}^*_{p,f+1}(k-1) + \overline{s}'_{f,N+1}(k)e^I_{p,f+1}(k-f-1), \tag{5.60}$$

$$\gamma_{p,f+1}(k-f-1) = \overline{c}'_{f,N+1}(k)\gamma_{p,f}(k-f-1). \tag{5.61}$$

Equations (5.43), (5.44), (5.49), (5.46) (summarized in the IFP block of Table 5.2) and (5.56), (5.57), (5.58), (5.59), (5.60), and (5.61) (summarized in the Int(F) block of Table 5.2) constitute the *SRF QRD-LSL interpolation algorithm* as $(p,f) \rightarrow (p,f+1)$. However, $e^I_{p,f+1}(k-f-1)$ computed in (5.59) is actually the *a priori* interpolation error of order $(p,f+1)$. The *a posteriori* interpolation error of order $(p,f+1)$ can then be computed by

$$\varepsilon^I_{p,f+1}(k-f-1) = \gamma_{p,f+1}(k-f-1)e^I_{p,f+1}(k-f-1). \tag{5.62}$$

The derivation of the order-updated interpolation error of order $(p,f) \rightarrow (p+1,f)$ can be similarly obtained.

5.4.3 SRF QRD-LSL prediction algorithm and SRF joint process estimation

The widely known SRF QRD-LSL prediction algorithm, which consists of both forward prediction (FP) block and backward prediction (BP) block summarized in Table 5.1, is actually a special case of the SRF QRD-LSL interpolation algorithm. The SRF QRD-LSL prediction algorithm in FP block and BP block can be directly derived by setting $(p,f) = (0,N)$ in the IFP block and $(p,f) = (N,0)$ in the IBP block, respectively. In deriving the SRF QRD-LSL prediction algorithm, the following results have been used. $e^I_{0,N}(k-N-1) = e_{b,N}(k-1)$, $e_{f,N+1}(k,k-N-1) = e_{f,N}(k)$, $I_{0,N}(k-N-1) = B_N(k-1)$, $e^I_{N,0}(k) = e_{f,N}(k)$, $e_{b,N+1}(k,k) = e_{b,N}(k-1)$, $I_{N,0}(k) = F_N(k)$. The FP block and the BP block can also be reduced directly from Int(P) block and Int(F) block by setting $(p,f) = (N,0)$ and $(p,f) = (0,N)$, respectively.

The SRF QRD-LSL prediction algorithm provides the mathematical foundation for the joint process estimation displayed in Figure 5.1 and is used as a sub-system to solve the joint process estimation problem where two optimal estimations are performed jointly. The two optimal estimations are (a) the forward reflection coefficients $\overline{\pi}_{f,m}(k)$ in the FP block and the backward reflection coefficients $\overline{\pi}_{b,m}(k)$ in the BP block both of which characterize a multistage lattice predictor with input $x(k)$ in the LS sense; (b) the regression coefficients $\overline{p}_m(k)$ that characterize a LS estimator of $d(k)$ to be developed below.

The SRF joint process estimation is developed by first considering a special case of (5.30) by setting $(p,f) = (m,0)$, which transforms the data matrix $\mathbf{X}_m(k-1)$

into the following upper triangular form that clearly is related to the QRD-LSL prediction, with $x(k-1)$ being the most recent data sample used,

$$\overline{\mathbf{Q}}(k-1)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\mathbf{X}_m(k-1) = \begin{bmatrix} \mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) \\ \mathbf{O}_{(k-m)\times(m-1)} & \mathbf{0} \end{bmatrix}, \tag{5.63}$$

where $\mathbf{R}_{m-1,0}(k-1)$ is a $(m-1)\times(m-1)$ upper triangular matrix as shown in (5.33) and $\mathbf{p}_{b,m-1}(k-1)$ is a $(m-1)\times 1$ vector whose elements are not of direct interest. The same $k\times k$ unitary matrix $\overline{\mathbf{Q}}(k-1)$ is also applied to a weighted *desired signal vector* to obtain

$$\overline{\mathbf{Q}}(k-1)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\mathbf{d}(k-1) = \begin{bmatrix} \mathbf{p}(k-1) \\ p_{m-1}(k-1) \\ \mathbf{v}(k-1) \end{bmatrix}, \tag{5.64}$$

where $\mathbf{d}^{\mathrm{T}}(k-1) = [d^*(0), d^*(1), \ldots, d^*(k-1)]$, $\mathbf{p}^{\mathrm{T}}(k-1) = [p_0(k-1), p_1(k-1), \ldots, p_{m-2}(k-1)]$, and $\mathbf{v}(k-1)$ is a vector containing the remaining $(k-m)$ elements. Subtracting (5.64) from (5.63), the latter of which has been first post-multiplied by a $m\times 1$ tap weight vector $\mathbf{w}(k-1)$, yields the following transformed estimation error vector

$$\begin{aligned}&\overline{\mathbf{Q}}(k-1)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\boldsymbol{\varepsilon}(k-1)\\ &\quad= \begin{bmatrix} \mathbf{p}(k-1) \\ p_{m-1}(k-1) \\ \mathbf{v}(k-1) \end{bmatrix} - \begin{bmatrix} \mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) \\ \mathbf{O}_{(k-m)\times(m-1)} & \mathbf{0} \end{bmatrix}\mathbf{w}(k-1),\end{aligned} \tag{5.65}$$

where $\boldsymbol{\varepsilon}(k-1) = [\varepsilon^*(0), \varepsilon^*(1), \ldots, \varepsilon^*(k-1)]^{\mathrm{T}} = \mathbf{d}(k-1) - \mathbf{X}_m(k-1)\cdot\mathbf{w}(k-1)$ is the error vector whose element $\varepsilon^*(k-1)$ is the error in estimating $d^*(k-1)$ of the desired response from a linear combination of $x^*(k-m), \ldots, x^*(k-1)$ when using the tap weight vector $\mathbf{w}(k-1)$ at time $(k-1)$.

The optimum tap weight vector, if desired, can be obtained from $\begin{bmatrix} \mathbf{p}(k-1) \\ p_{m-1}(k-1) \end{bmatrix}$ $= \begin{bmatrix} \mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) \end{bmatrix}\mathbf{w}_o(k-1)$ using back-substitution and this choice of $\mathbf{w}_o(k-1)$ gives a minimized error vector as

$$\min_{\mathbf{w}_o(k-1)} \|\boldsymbol{\varepsilon}(k-1)\| = \|\mathbf{v}(k-1)\|. \tag{5.66}$$

Given new observation $x^*(k)$ along with the new desired response $d^*(k)$ received at time k and by appending the transformed desired data vector in (5.64) as the rightmost column of (5.63) followed by a sequence of $(m-1)$ Givens rotations proceeding rightwards from the $(k+1,1)$th element to the $(k+1,m-1)$th element that are used to annihilate the bottom row vector $\mathbf{x}_{m-1}^{\mathrm{H}}(k) = [x^*(k), x^*(k-1), \ldots, x^*(k-m+2)]$ yields

$$\mathbf{T}_{m-1}(k)\begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) & \lambda^{\frac{1}{2}}\mathbf{p}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) & \lambda^{\frac{1}{2}}p_{m-1}(k-1) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\mathbf{v}(k-1) \\ \mathbf{x}_{m-1}^{\mathrm{H}}(k) & x^*(k-m+1) & d^*(k) \end{bmatrix}$$
$$= \underbrace{\begin{bmatrix} \mathbf{R}_{m-1,0}(k) & \mathbf{p}_{b,m-1}(k) & \mathbf{p}(k) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) & \lambda^{\frac{1}{2}}p_{m-1}(k-1) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\mathbf{v}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \overline{\varepsilon}_{b,m-1}^{*}(k) & \overline{\varepsilon}_{m-1}^{*}(k) \end{bmatrix}}_{\mathbf{L}(k)}, \tag{5.67}$$

where the sequence of Givens rotations used in $\mathbf{T}_{m-1}(k) = J_0(k)\cdot J_1(k)\cdot\ldots\cdot J_{m-2}(k)$ turns out to be the same cosine and sine parameters used in the FP block. As a result of this transformation, both the $(m-1)$th-*order* angle-normalized backward prediction error $\overline{\varepsilon}_{b,m-1}^{*}(k)$ and the $(m-1)$th-*order* angle-normalized joint process estimation error $\overline{\varepsilon}_{m-1}^{*}(k)$ are produced.

The order-updated angle-normalized joint process estimation error $\overline{\varepsilon}_{m}^{*}(k)$ can be obtained by applying the following Givens rotation to matrix $\mathbf{L}(k)$ to annihilate the $(k+1,m)$th element and yields

$$\begin{bmatrix} \mathbf{I}_{m-1} & & & \\ & c_{b,m-1}(k) & & s_{b,m-1}^{*}(k) \\ & & \mathbf{I}_{k-m} & \\ & -s_{b,m-1}(k) & & c_{b,m-1}(k) \end{bmatrix}\mathbf{L}(k) = \underbrace{\begin{bmatrix} \mathbf{R}_{m-1,0}(k) & \mathbf{p}_{b,m-1}(k) & \mathbf{p}(k) \\ \mathbf{0}^{\mathrm{T}} & B_{m-1}^{\frac{1}{2}}(k) & p_{m-1}(k) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\mathbf{v}(k-1) \\ \mathbf{0}^{\mathrm{T}} & 0 & \overline{\varepsilon}_{m}^{*}(k) \end{bmatrix}}_{\mathbf{H}(k)}. \tag{5.68}$$

By taking out a scaling factor from each row of matrices $\mathbf{L}(k)$ and $\mathbf{H}(k)$, the rows before and after the Givens rotation in (5.68) is denoted, respectively, by

$$\mathbf{L}(k) = \mathbf{L}_1(k)\mathbf{L}_2(k) = diag\left[\ldots,\sqrt{\lambda B_{m-1}(k-1)},\ldots,\sqrt{\gamma_{m-1}(k)}\right]$$
$$\begin{bmatrix} \overline{\mathbf{R}}_{m-1,0}(k) & \overline{\mathbf{p}}_{b,m-1}(k) & \overline{\mathbf{p}}(k) \\ \mathbf{0}^{\mathrm{T}} & 1 & \overline{p}_{m-1}(k-1) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\overline{\mathbf{v}}(k-1) \\ \mathbf{0}^{\mathrm{T}} & e_{b,m-1}^{*}(k) & e_{m-1}^{*}(k) \end{bmatrix}, \tag{5.69}$$

and

$$\mathbf{H}(k) = \mathbf{H}_1(k)\mathbf{H}_2(k) = diag\left[\ldots,\sqrt{B_{m-1}(k)},\ldots,\sqrt{\gamma_{m}(k)}\right]$$
$$\begin{bmatrix} \overline{\mathbf{R}}_{m-1,0}(k) & \overline{\mathbf{p}}_{b,m-1}(k) & \overline{\mathbf{p}}(k) \\ \mathbf{0}^{\mathrm{T}} & 1 & \overline{p}_{m-1}(k) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\overline{\mathbf{v}}(k-1) \\ \mathbf{0}^{\mathrm{T}} & 0 & e_{m}^{*}(k) \end{bmatrix}. \tag{5.70}$$

Some elements of the matrices in (5.69) and (5.70) and the corresponding elements of the matrices in (5.19) and (5.20), respectively, can be related as follows: $k_a = \lambda B_{m-1}(k-1)$, $k_b = \gamma_{m-1}(k)$, $a_1 = 1$, $a_2 = \overline{p}_{m-1}(k-1)$, $b_1 = e_{b,m-1}^{*}(k)$, $b_2 = e_{m-1}^{*}(k)$, $k_a' = B_{m-1}(k)$, $k_b' = \gamma_m(k)$, $a_1' = 1$, $a_2' = \overline{p}_{m-1}(k)$, $b_1' = 0$, and

$b'_2 = e^*_m(k)$. Substituting k_a, k'_a, k_b, b_1, a_2, a'_2, b_2, and b'_2 into (5.22), (5.25), (5.28), and (5.29) (by letting $i = 2$) yields the following SRF joint process estimation, which is also summarized in Table 5.1.

SRF joint process estimation:

$$e_m(k) = e_{m-1}(k) - e_{b,m-1}(k)\overline{p}^*_{m-1}(k-1), \tag{5.71}$$

$$\overline{p}^*_{m-1}(k) = \overline{p}^*_{m-1}(k-1) + \overline{s}_{b,m-1}(k)e_m(k), \tag{5.72}$$

where $\overline{c}_{b,m-1}(k) = \frac{\lambda B_{m-1}(k-1)}{B_{m-1}(k)}$ and $\overline{s}_{b,m-1}(k) = \gamma_{m-1}(k)\frac{e^*_{b,m-1}(k)}{B_{m-1}(k)}$ in which $\gamma_m(k) = \gamma_{m-1}(k)\overline{c}_{b,m-1}(k)$. The *a posteriori* estimation error can be computed by $\varepsilon_m(k) = \gamma_m(k)e_m(k)$.

5.5 SRF (QRD-LSL)-Based RLS Algorithm

For many applications such as system identification, adaptive equalization, and active noise control, wherein the filter weights of the corresponding LS algorithm are required. The SRF QRD-LSL interpolation algorithm can be applied to implement the RLS algorithm in the transversal structure that generates the desired weight vector and the resulting algorithm is referred to as the SRF (QRD-LSL)-based RLS algorithm. The $(N+1)$st-*order* RLS algorithm with the tap weight vector at time k, $\mathbf{w}_{N+1}(k)$, can be calculated recursively in time using $\mathbf{w}_{N+1}(k) = \mathbf{w}_{N+1}(k-1) + \mathbf{k}_{N+1}(k)e^*(k)$ (see Table 2.5 in Chapter 2), where $\mathbf{k}_{N+1}(k) = \mathbf{R}^{-1}(k)\cdot\mathbf{x}_{N+1}(k)$ is the Kalman gain vector.

The Kalman gain vector can also be calculated as a particular set of normalized least-squares interpolation errors [23]:

$$\mathbf{k}^{\mathrm{T}}_{N+1}(k) = \left[\frac{\varepsilon^I_{N,0}(k)}{I_{N,0}(k)}, \ldots, \frac{\varepsilon^I_{p+1,f-1}(k-f+1)}{I_{p+1,f-1}(k-f+1)}, \frac{\varepsilon^I_{p,f}(k-f)}{I_{p,f}(k-f)}, \right.$$
$$\left. \frac{\varepsilon^I_{p-1,f+1}(k-f-1)}{I_{p-1,f+1}(k-f-1)}, \ldots, \frac{\varepsilon^I_{0,N}(k-N)}{I_{0,N}(k-N)}\right]. \tag{5.73}$$

Equation (5.73) can be proved as follows. From (5.4), we have $\mathbf{b}_{p,f}(k-f) = \mathbf{R}^{-1}(k)\mathbf{i}_{p,f}(k-f) = I_{p,f}(k-f)\cdot\mathbf{R}^{-1}(k)\left[\mathbf{0}^{\mathrm{T}}_f \; 1 \; \mathbf{0}^{\mathrm{T}}_p\right]^{\mathrm{T}}$. Taking the Hermitian on both sides of the above equation and realizing that the correlation matrix is Hermitian [i.e., $\mathbf{R}^{\mathrm{H}}(k) = \mathbf{R}(k)$], we have

$$\left[\mathbf{0}_f^{\mathrm{T}}\ 1\ \mathbf{0}_p^{\mathrm{T}}\right]\cdot\mathbf{R}^{-1}(k)=\frac{1}{I_{p,f}(n-f)}\cdot\mathbf{b}_{p,f}^{\mathrm{H}}(n-f). \tag{5.74}$$

We then post-multiply both sides of the (5.74) by $\mathbf{x}_{N+1}(k)$. Using $\mathbf{k}_{N+1}(k)=\mathbf{R}^{-1}(k)\cdot\mathbf{x}_{N+1}(k)$ and (5.2) [by setting $i=k-f$ in (5.2)] yields the $(f+1)$st element of the Kalman gain vector.

$$\left[\mathbf{k}_{N+1}(k)\right]_{f+1}=\left[\mathbf{0}_f^{\mathrm{T}}\ 1\ \mathbf{0}_p^{\mathrm{T}}\right]\cdot\mathbf{R}^{-1}(k)\cdot\mathbf{x}_{N+1}(k) \tag{5.75}$$

$$=\frac{1}{I_{p,f}(k-f)}\cdot\mathbf{b}_{p,f}^{\mathrm{H}}(k-f)\cdot\mathbf{x}_{N+1}(k)=\frac{\varepsilon_{p,f}^{I}(k-f)}{I_{p,f}(k-f)} \tag{5.76}$$

By adjusting the values of p and f with $p+f=N$, all the elements of the Kalman gain vector in (5.73) can be calculated. Equation (5.74) reveals the connection between linear interpolation and the inverse of the time-average correlation matrix that is required to obtain the Kalman gain vector.

The Kalman gain vector is the most decorrelated version of the normalized input vector, because each element of the Kalman gain vector in (5.73) is the optimum two-sided prediction residual of each corresponding element of the input vector $\mathbf{x}_{N+1}(k)$. Accordingly, the Kalman gain vector corresponds to the least redundant version of the input vector. Therefore, the elements of $\mathbf{k}_{N+1}(k)$ may be responsible for the fast convergence of the RLS algorithm. The SRF QRD-LSL interpolation algorithm can be used to calculate $\mathbf{k}_{N+1}(k)$ in (5.73) in an *order-recursive* manner through a divide and conquer method [23]; the resulting SRF (QRD-LSL)-based RLS algorithm requires $\mathscr{O}[N\log_2 N]$ operations for a transversal filter of order N.

5.6 Simulations

All simulations were run on a PC, in a floating-point environment, with 10-tap weights. The effective number of mantissa bits in the floating-point representation is reduced to observe the finite-precision effects by truncating the mantissa at a predefined position without affecting the exponent. Three algorithms were applied to perform adaptive prediction of an autoregressive (AR) process that closely follows the one presented in [13], and their numerical robustness were evaluated in this scenario. The observation is $x(k)=x_{AR}(k)+w_1(k)$, where $x_{AR}(k)=-\sum_{i=1}^{10}a_i x_{AR}(k-i)+w_2(k)$ is an AR process of order 10 with an eigenvalue spread approximately 1020. Both $w_1(k)$ and $w_2(k)$ are independent white noise processes with zero mean. The variance of $w_2(k)$ is set to cause $x_{AR}(k)$ to have a variance of 0 dB, whereas the variance of $w_1(k)$ is chosen such that the signal-to-noise ratio is 30 dB. The desired response is therefore $x(k)$ while the observations are $[x(k-1),x(k-2),\ldots,x(k-10)]$.

Figure 5.2 compares three learning curves obtained by ensemble-averaging the prediction error squares over 200 independent experimental trials. The three learning

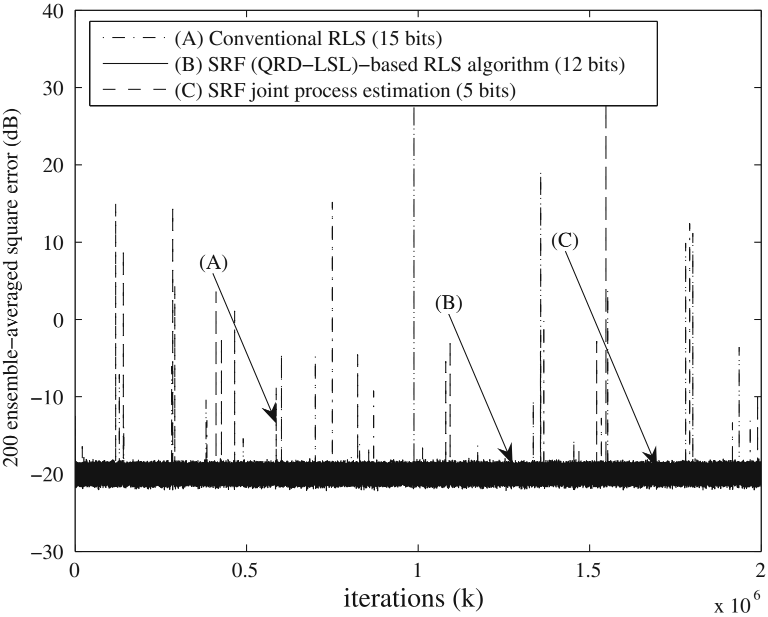


Fig. 5.2 Learning curves of three algorithms in a finite-precision environment.

curves are the results of using a conventional RLS algorithm of $\mathscr{O}[N^2]$ complexity in Table 2.5 (in Chapter 2), the SRF (QRD-LSL)-based RLS algorithm of $\mathscr{O}[N\log_2 N]$ complexity, and the joint process estimation of $\mathscr{O}[N]$ complexity in Table 5.1 without computing the optimal tap weight vector, all with a forgetting factor of $\lambda = 0.996$ and a regularization parameter $\delta = 0.004$. Our simulations demonstrated that when exact arithmetic is used, all three algorithms yielded exactly the same result, but they exhibited various unstable behaviors with finite-precision computation. While the conventional RLS algorithm became unacceptable for less than 16 mantissa bits, the SRF (QRD-LSL)-based RLS algorithm with 12 mantissa bits ran successfully for two million iterations (at which point the experiment was terminated). Although the mean square error produced by the joint process estimation based on the QRD-LSL prediction algorithm is a little higher than that of the SRF (QRD-LSL)-based RLS algorithm, it still produces useful results with only five mantissa bits. This is because the joint process section is subordinate to the prediction section, and the numerical stability of the joint process estimation thus depends on the SRF QRD-LSL prediction algorithm. It can be shown that the computations of the conventional forward and backward prediction errors in the SRF QRD-LSL prediction algorithm involve the error feedback mechanism described in Section 5.3, and therefore their error growth in a finite-precision environment is always bounded. For example, the computation of the forward prediction error summarized in the FP block in Table 5.1 is rewritten as

$$e_{f,m}(k) = e_{f,m-1}(k) - e_{b,m-1}(k-1)\overline{\pi}^{*}_{f,m}(k-1), \tag{5.77}$$

$$\overline{\pi}^{*}_{f,m}(k) = \overline{\pi}^{*}_{f,m}(k-1) + \overline{s}_{b,m-1}(k-1)e_{f,m}(k). \tag{5.78}$$

A unique feature of the error feedback mechanism is that the *a priori* prediction error $e_{f,m}(k)$ computed in (5.77) is fed back to time-update the forward reflection coefficient $\overline{\pi}^{*}_{f,m}(k)$ whose cumulative error is bounded for all time in a finite-precision environment. However, as mentioned in Section 5.4, the computations of the intermediate forward and backward prediction errors from the conventional forward and backward prediction errors in the two equation pairs using (5.49), (5.46) and (5.50), (5.48) in the QRD-LSL interpolation algorithm, correspond to the reversed formulations. These reversed formulations no longer involve an error feedback mechanism, potentially causing numerical instability in the QRD-LSL interpolation algorithm in a finite-precision environment. It has been shown by Skidmore and Proudler [23] that the error growth in this reversed formulation is *linear* with time in nature in a finite-precision environment (i.e., the errors accumulate but are not amplified by each iteration). This linear error growth may have resulted in the divergence of the SRF (QRD-LSL)-based RLS algorithm before two million iterations are completed when less than 12 mantissa bits is used. However, this linear error growth should be contrasted to the exponential error growth with time exhibited by the fast transversal filter (FTF) algorithm [32] that may much more rapidly diverge from the correct solution.

5.7 Conclusion

This chapter develops the SRF QRD-LSL interpolation algorithm, which is then reduced to the SRF QRD-LSL prediction algorithm, which is in turn extended to develop the SRF joint process estimation. As described in [32, 33], no exact-RLS algorithm, whether in lattice or transversal filter, can always be stable, because of limited-precision instabilities. The RLS algorithm based on the SRF QRD-LSL interpolation algorithm is no exception, and it may diverge in a finite-precision environment due to error accumulation. Simulations indicated that the SRF (QRD-LSL)-based RLS algorithm using only eight mantissa bits, started to diverge from the correct solution around $k = 2 \times 10^5$ in the computer experiment conducted in Section 5.6, because of the two equation pairs, (5.49), (5.46) and (5.50), (5.48), computed in the QRD-LSL interpolation algorithm. In contrast, the computations of the forward and backward prediction errors in the SRF QRD-LSL prediction algorithm involve an error feedback mechanism, and therefore the corresponding error growth in a finite-precision environment is always bounded. Consequently, the joint process estimation based on the SRF QRD-LSL prediction algorithm still exhibits well-conditioned behaviors with only five mantissa bits. However, the joint process estimation does not generate the filter weights, which are required in some applications.

Skidmore and Proudler [23] employed exactly the same concept as demonstrated by the two equation pairs, (5.49), (5.46) and (5.50), (5.48), to implement an SRF QR-RLS algorithm, referred to as the KaGE RLS algorithm. Although the KaGE RLS algorithm, like the SRF (QRD-LSL)-based RLS algorithm, may exhibit linear error growth with time in a finite-precision environment owing to the computation of the two equation pairs described above, simulation results presented in [23] reveal that the KaGE RLS algorithm can run reliably for many millions of iterations using single precision arithmetic and is inherently much more stable than the stabilized FTF algorithm proposed by Maouche and Slock [34]. Therefore, as suggested in [23], to generate the transversal filter weights, the KaGE algorithm as well as the SRF (QRD-LSL)-based RLS algorithm presented in this chapter, both employing interpolation residuals and both of $\mathscr{O}[N\log_2 N]$ complexity, may offer a favorable compromise between the computationally efficient FTF algorithms of $\mathscr{O}[N]$ complexity and the stable algorithms of $\mathscr{O}[N^2]$ complexity, such as the Inverse QRD-RLS algorithm proposed by Alexander and Ghirnikar [35]. The Inverse QRD-RLS algorithm may not be computationally feasible for some real-time implementations of long adaptive filters.

References

1. D. T. Lee, M. Morf, and B. Friedlander, Recursive least squares ladder estimation algorithms. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-29, no. 3, pp. 627–641 (June 1981)
2. L. J. Griffiths, An adaptive lattice structure for noise-cancelling applications. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'78, Tulsa, USA, pp. 873–90 (April 1978)
3. J. Makhoul, A class of all-zero lattice digital filters: Properties and applications. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-26, pp. 304–314 (August 1978)
4. P. Strobach, Linear Prediction Theory – A Mathematical Basis for Adaptive Systems. Springer-Verlag, Berlin, Germany (1990)
5. S. Haykin, Adaptive Filter Theory. 4th edition Prentice-Hall, Englewood Cliffs, NJ, USA (2002)
6. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley, NJ, USA (2003)
7. F. Capman, J. Boudy, and P. Lockwood, Acoustic echo cancellation using a fast QR-RLS algorithm and multirate schemes. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 969–972 (May 1995)
8. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-38, no. 4, pp. 631–653 (April 1990)
9. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, QRD-based lattice filter algorithms. SPIE Conference on Advanced Algorithms and Architectures for Signal Processing, San Diego, USA, vol. 1152, pp. 56–67 (August 1989)
10. F. Ling, Givens rotation based least squares lattice and related algorithms. IEEE Transactions on Signal Processing, vol. 39, no. 7, pp. 1541–1551 (July 1991)
11. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 879–891 (April 1991)

12. A. A. Rontogiannis and S. Theodoridis, New fast QR decomposition least squares adaptive algorithms. IEEE Transactions on Signal Processing, vol. 46, no. 8, pp. 2113–2121 (August 1998)
13. B. Yang and J. F. Böhme, Rotation-based RLS algorithms: unified derivations, numerical properties, and parallel implementations. IEEE Transactions on Signal Processing, vol. 40, no. 5, pp. 1151–1167 (May 1992)
14. A. K. Jain, Image coding via a nearest neighbors image model. IEEE Transactions on Communications, vol. COMM-23, no. 3, pp. 318–331 (March 1975)
15. E. A. Gifford, B. R. Hunt, and M. W. Marcellin, Image coding using adaptive recursive interpolative DPCM. IEEE Transactions on Image Processing, vol. 4, no. 8, pp. 1061–1069 (August 1995)
16. G. Yang, H. Leich, and R. Boite, Voiced speech coding at very low bit rates based on forward–backward waveform prediction. IEEE Transactions on Speech and Audio Processing, vol. 3, no. 1, pp. 40–47 (January 1995)
17. E. Masry, Closed-form analytical results for the rejection of narrow-band interference in PN spread spectrum systems-Part II: linear interpolation filters. IEEE Transactions on Communications, vol. COMM-33, no. 1, pp. 10–19 (January 1985)
18. J.-T. Yuan and J.-N. Lee, Narrowband interference rejection in DS/CDMA systems using adaptive (QRD-LSL)-based nonlinear ACM interpolators. IEEE Transactions on Vehicular Technology, vol. 52, no. 2, pp. 374–379 (March 2003)
19. S. Kay, Some results in linear interpolation theory. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-31, no. 3, pp. 746–749 (June 1983)
20. B. Picinobono and J. M. Kerilis, Some properties of prediction and interpolation errors. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-36, no. 4, pp. 525–531 (April 1988)
21. C. K. Coursey and J. A. Stuller, Linear interpolation lattice. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 965–967 (April 1991)
22. J.-T. Yuan, QR-decomposition-based least-squares lattice interpolators. IEEE Transactions on Signal Processing, vol. 48, no. 1, pp. 70–79 (January 2000)
23. I. D. Skidmore and I. K. Proudler, The KaGE RLS algorithm. IEEE Transactions on Signal Processing, vol. 51, no. 12, pp. 3094–3104 (December 2003)
24. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, The QRD-based least squares lattice algorithm: Some computer simulations using finite wordlengths. IEEE International Symposium on Circuits and Systems, ISCAS'90, New Orleans, USA, pp. 258–261 (May 1990)
25. S. F. Hsieh, K. J. R. Liu, and K. Yao, A unified square-root-free approach for QRD-based recursive least squares estimation. IEEE Transactions on Signal Processing, vol. 41, no. 3, pp. 1405–1409 (March 1993)
26. F. Ling, D. Manolakis, and J. G. Proakis, Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-34, no. 4, pp. 837–845 (August 1986)
27. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, Computationally efficient QR-decomposition approach to least squares adaptive filtering. IEE Proceedings-F, vol. 138, no. 4, pp. 341–353 (August 1991)
28. W. M. Gentleman, Least squares computations by Givens transformations without square roots. IMA Journal of Applied Mathematics, vol. 12. pp. 329–336 (December 1973)
29. J. G. McWhirter, Recursive least-squares minimization using a systolic array. SPIE Real-Time Signal Processing VI, vol. 431, pp. 105–112 (January 1983)
30. J.-T. Yuan, C.-A. Chiang, and C.-H. Wu, A square-root-free QRD-LSL interpolation algorithm. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'2008, Las Vegas, USA, pp. 3813–3816 (April 2008)
31. C.-A. Chiang, A recursive least-squares (RLS) algorithm based on interpolation lattice recursive structure. M.S. thesis, Adviser: J.-T. Yuan, Fu Jen Catholic University, Taipei, Taiwan, R.O.C. (April 2006)

32. J. M. Cioffi and T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32, no. 2, pp. 304–337 (April 1984)
33. S. Ljung, Fast algorithms for integral equations and least-squares identification problems. Ph.D. thesis, Linkoping University, Sweden (1983)
34. K. Maouche and D. T. M. Slock, Fast subsampled-updating stabilized fast transversal filter (FSU SFTF) RLS algorithm for adaptive filtering. IEEE Transactions on Signal Processing, vol. 48, no. 8, pp. 2248–2257 (August 2000)
35. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)

Chapter 6
Multichannel Fast QRD-RLS Algorithms

António L. L. Ramos and Stefan Werner

Abstract When considering multichannel adaptive implementations, it is often possible to directly apply standard single-channel algorithms to the multichannel problem, e.g., the numerically stable and fast converging QR decomposition recursive least-square (QRD-RLS) algorithm. Even though such a solution would provide fast convergence, it may be computationally too complex due to a large number of coefficients. In order to obtain a computationally efficient solution, RLS-type algorithms specially tailored for the multichannel setup are a good option. This chapter introduces various multichannel fast QRD-RLS (MC-FQRD-RLS) algorithms that can be seen as extensions of the basic single-channel FQRD-RLS algorithms to the case of a multichannel input vector, where it can be assumed that each channel has a time-shift structure. We provide, in a general framework, a comprehensive and up-to-date discussion of the MC-FQRD-RLS algorithms, addressing issues such as derivation, implementation, and comparison in terms of computational complexity.

6.1 Introduction

Multichannel signal processing can be found in various applications such as color image processing, multispectral remote sensing imagery, biomedicine, channel equalization, stereophonic echo cancelation, multidimensional signal processing, Volterra-type non-linear system identification, and speech enhancement [1, 2]. When choosing among the adaptive algorithms that can cope with multichannel signals, the choice is more than often based on stability, convergence speed, and

António L. L. Ramos
Buskerud University College, Kongsberg – Norway
e-mail: `antonio.ramos@hibu.no`

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

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_6, © Springer Science+Business Media, LLC 2009

computational complexity. The standard QR decomposition recursive least-squares (QRD-RLS) algorithm stands out as potential good option because of its well-known fast convergence property and excellent numerical behavior. However, its $\mathscr{O}[P^2]$ computational complexity makes its use prohibitive when higher order filters are required.

The FQRD-RLS algorithms, in general, offer the same fast converging feature as the standard QRD-RLS algorithm, while attaining a lower computational complexity, which is achieved by exploiting the underlying time-shift structure of the input signal vector. Historically, the first solutions were presented for the case where the input signal is just a "tapped-delay" line, i.e., a single-channel signal, and multichannel fast QRD-RLS (MC-FQRD-RLS) algorithms arise as a natural extensions of basic FQRD-RLS algorithms making them useful also in multichannel applications. The MC-FQRD-RLS algorithms can be classified into three distinct ways, according to [3]: (1) which error vector is being updated (*a priori* or *a posteriori*); (2) the type of prediction used (forward or backward),[1] and; (3) the approach taken for the derivation (sequential- or block-type). The first two are inherited from the single-channel case, whereas the last one, specific for multichannel algorithms, determines how new multichannel input samples are processed. These three concepts are combined in Table 6.1 for the case of MC-FQRD-RLS algorithms based on backward prediction errors, which are the ones addressed in this work. The structure parameter introduced in this table simply denotes the way a particular algorithm is implemented.

Table 6.1 Classification of the MCFQRD-RLS algorithms.

Error type	Approach and order		Structure	References	Algorithm
MCFQR POS_B	**BLOCK-CHANNEL**	*Equal Order*	Lattice	[4]	**1**
			Transversal	[4–6]	**2**
		Multiple Order	Lattice	—	**3**
			Transversal	[7, 8]	**4**
	SEQUENTIAL-CHANNEL	*Equal Order*	Lattice	Implicit in [9]	**5**
			Transversal	Suggested in [5]	**6**
		Multiple Order	Lattice	[9]	**7**
			Transversal	[8, 10]	**8**
MCFQR PRI_B	**BLOCK-CHANNEL**	*Equal Order*	Lattice	[11]	**9**
			Transversal	[4, 6, 11]	**10**
		Multiple Order	Lattice	—	**11**
			Transversal	[7]	**12**
	SEQUENTIAL-CHANNEL	*Equal Order*	Lattice	Implicit in [11]	**13**
			Transversal	Implicit in [11]	**14**
		Multiple Order	Lattice	[11]	**15**
			Transversal	[11]	**16**

[1] This chapter does not consider FQRD-RLS algorithms based on the updating of the forward error vector. As pointed out in Chapter 4, and still holding for the multichannel case, these algorithms are reported to be unstable, in contrast with their backward error vector updating-based counterparts.

Depending on the approach taken for the derivation, the $\mathscr{O}[P^2]$ computational complexity of the standard QRD-RLS implementation can be reduced to $\mathscr{O}[MP]$ and $\mathscr{O}[M^2P]$, for sequential- and block-channel processing algorithms, respectively; with P being the total number of filter coefficients and M the number of channels. Although the computational cost of block-channel algorithms is slightly higher than sequential-channel algorithms, they are more suitable for parallel processing implementations.

After reformulating the problem for the multichannel case, most of the equations are turned into matrix form, with some involving matrix inversion operations. Beside being potential sources of numerical instability, the computational burden is also greatly increased contrasting with desirable low-complexity feature of single-channel FQRD-RLS algorithms that rely on scalar operations only. Nevertheless, many good solutions to these problems exist and are addressed in the following.

The reminder of this chapter is organized as follows. In Section 6.2, we introduce the standard MC-FQRD-RLS problem formulation along with a new definition of the input vector that allows for a more general setup where the various channels may have different orders. The sequential- and block-type algorithms are discussed in Sections 6.3 and 6.4, respectively, while order-recursive implementations are addressed in Section 6.5. An application example and computational complexity issues are presented in Sections 6.6.1 and 6.6.2, respectively. Finally, closing remarks are given in Section 6.7.

6.2 Problem Formulation

The MC-FQRD-RLS problem for channels of equal orders was addressed in early 90's, in [5], where the sequential approach was introduced to avoid complicated matrix operations, and in [12] where both sequential and block approaches were addressed. Later, multiple-order block-multichannel algorithms comprising scalar operations only were introduced in [3, 8, 11, 13].

For the multichannel problem, the weighted least-squares (WLS) objective function, introduced in Chapter 2, is given as

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i}[d(i) - \mathbf{x}_P^{\mathrm{T}}(i)\mathbf{w}_P(k)]^2 = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k), \tag{6.1}$$

where $\mathbf{e}(k)$ is the weighted error vector defined as

$$\begin{aligned}\mathbf{e}(k) &= \begin{bmatrix} d(k) \\ \lambda^{1/2}d(k-1) \\ \vdots \\ \lambda^{k/2}d(0) \end{bmatrix} - \begin{bmatrix} \mathbf{x}_P^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}_P^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}_P^{\mathrm{T}}(0) \end{bmatrix} \mathbf{w}_P(k) \\ &= \mathbf{d}(k) - \mathbf{X}_P(k)\mathbf{w}_P(k), \end{aligned} \tag{6.2}$$

and

$$\mathbf{x}_P^{\mathrm{T}}(k) = \begin{bmatrix} \mathbf{x}_k^{\mathrm{T}} & \mathbf{x}_{k-1}^{\mathrm{T}} & \cdots & \mathbf{x}_{k-N+1}^{\mathrm{T}} \end{bmatrix}, \tag{6.3}$$

where $\mathbf{x}_k^{\mathrm{T}} = [x_1(k) \quad x_2(k) \quad \cdots \quad x_M(k)]$ is the input signal vector at instant k. As illustrated in Figure 6.1, N is the number of filter coefficients per channel, M is the number of input channels, and $\mathbf{w}_P(k)$ is the $P \times 1$ coefficient vector at time instant k, $P = MN$ being the total number of elements for the case of channels with equal orders.

The good numerical behavior of the QR-decomposition-based RLS algorithms is due to the fact that they make use of the square root $\mathbf{U}_P(k)$ of the input-data autocorrelation matrix $\mathbf{X}_P^T(k)\mathbf{X}_P(k)$. The lower triangular matrix $\mathbf{U}_P(k)$ is also known as the Cholesky factor of $\mathbf{X}_P^T(k)\mathbf{X}_P(k)$ and can be obtained by applying a set of Givens rotations $\mathbf{Q}_P(k)$ onto $\mathbf{X}_P(k)$. Hence, pre-multiplying both sides of (6.2) by unitary matrix $\mathbf{Q}_P(k)$ does not alter its norm yielding

$$\mathbf{e}_q(k) = \mathbf{Q}_P(k)\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_{q1}(k) \\ \mathbf{e}_{q2}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{U}_P(k) \end{bmatrix} \mathbf{w}_P(k). \tag{6.4}$$

Again, minimizing $\|\mathbf{e}_q(k)\|^2$ is equivalent to minimizing the cost function of (6.1). In other words, Equation (6.1) is minimized by choosing $\mathbf{w}_P(k)$ in (6.4) such that $\mathbf{d}_{q2}(k) - \mathbf{U}_P(k)\mathbf{w}_P(k)$ equals zero, i.e.,

$$\mathbf{w}_P(k) = \mathbf{U}_P^{-1}(k)\mathbf{d}_{q2}(k). \tag{6.5}$$

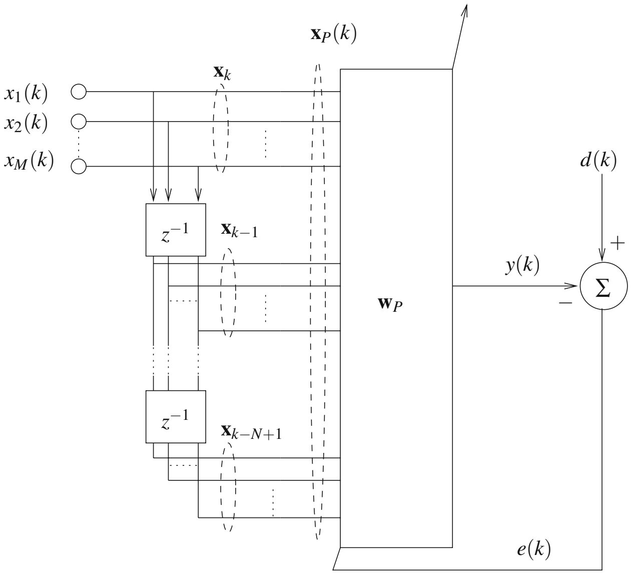


Fig. 6.1 Multichannel adaptive filter: case of equal order.

In many applications, we may come across the need of dealing with different channel orders which is referred to as the multiple-order case. In such a scenario, the elements of the input vector are arranged in a slightly different manner than for the equal-order case treated above. Below we explain how to build the input vector such that we can cope with both equal- and multiple-order channels.

6.2.1 Redefining the input vector

Let us define $N_1, N_2, \ldots, N_M$ as the number of *taps* in each of the M channels *tapped delay lines*. Thus, the total number of taps in the input vector is $P = \sum_{r=1}^{M} N_r$. Without loss of generality, we will assume $N_1 \geq N_2 \geq \cdots \geq N_M$.

Figure 6.2 shows an example of a multichannel scenario with $M = 3$ channels of unequal orders, where $N_1 = 4, N_2 = 3, N_3 = 2$, i.e., $P = 4+3+2 = 9$. The following approach to construct the input vector, $\mathbf{x}_P(k)$, was considered in [11] first channel are chosen to be the leading elements of $\mathbf{x}_P(k)$, followed by N_2–N_3 *pairs of samples* from the first and second channels, followed by N_3–N_4 *triples of samples* of the first three channels and so on till the $N_M - N_{M+1}$ *M-tuples of samples* of all channels. It is assumed that $N_{M+1} = 0$.

Alternatively, consider the $N_1 \times M$ matrix $\tilde{\mathbf{X}}(k)$ whose ith row contains the N_i input data samples of channel i, i.e.,

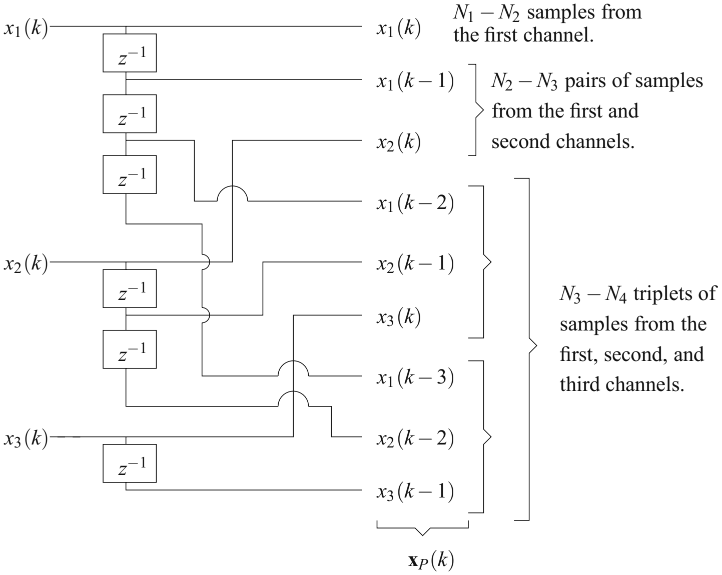

$x_1(k)$
z^{-1}
z^{-1}
z^{-1}
$x_2(k)$
z^{-1}
z^{-1}
$x_3(k)$
z^{-1}
$x_1(k)$
$N_1 - N_2$ samples from the first channel.
$x_1(k-1)$
$x_2(k)$
$N_2 - N_3$ pairs of samples from the first and second channels.
$x_1(k-2)$
$x_2(k-1)$
$x_3(k)$
$x_1(k-3)$
$x_2(k-2)$
$x_3(k-1)$
$N_3 - N_4$ triplets of samples from the first, second, and third channels.
$\mathbf{x}_P(k)$

Fig. 6.2 Example of input vector defined as in [11].

$$\tilde{\mathbf{X}}(k) = \begin{bmatrix} x_1(k) & x_1(k-1) & x_1(k-2) & \ldots & x_1(k-N_1+1) \\ \mathbf{0}_{1\times(N_1-N_2)} & x_2(k) & x_2(k-1) & \ldots & x_2(k-N_2+1) \\ \vdots & & \vdots & & \vdots \\ \mathbf{0}_{1\times(N_1-N_M)} & & x_M(k) & \ldots & x_M(k-N_M+1) \end{bmatrix} \tag{6.6}$$

where the zero-vectors appearing to the left in each row are of proper size to maintain the dimension of $\tilde{\mathbf{X}}(k)$ (if $N_1 = N_2 = \cdots = N_M$, the zeros will disappear). The multichannel input vector $\mathbf{x}_P(k)$ is obtained by simply stacking the columns of matrix $\tilde{\mathbf{X}}(k)$ and excluding the zeros that were inserted in (6.6). We see from (6.6) that the last M samples of vector $\mathbf{x}_P(k)$ are $\{x_l(k-N_i)\}_{i=1}^{M}$. As a result, updating the input vector from one time instant to the next, i.e., from $\mathbf{x}_P(k)$ to $\mathbf{x}_P(k+1)$, becomes particularly simple. This is because we know that, by construction, the last M samples of $\mathbf{x}_P(k)$ are to be removed when shifting in the new input samples $\{x_i(k+1)\}_{i=1}^{M}$.

The procedure detailed above gives rise to two distinct ways of obtaining the expanded input vector, $\mathbf{x}_{P+M}(k+1)$: (1) The new samples from each channel are shifted one-by-one and processed recursively, from the first to the last channel and; (2) All new samples from the different channels are shifted in together and processed simultaneously.

The first approach leads to sequential-type multichannel algorithms and the second one results in block-type multichannel algorithms. Before presenting the different algorithms, we take a closer look at the sequential- and block-type input vectors.

6.2.2 Input vector for sequential-type multichannel algorithms

For the sequential-channel case, the extended input vector, $\mathbf{x}_{P+M}(k+1)$, is constructed from $\mathbf{x}_P(k)$ in M successive steps as

$$\mathbf{x}_{P+1}^{\mathrm{T}}(k+1) = \begin{bmatrix} x_1(k+1) & \mathbf{x}_P^{\mathrm{T}}(k) \end{bmatrix}, \tag{6.7}$$

$$\mathbf{x}_{P+i}^{\mathrm{T}}(k+1) = \begin{bmatrix} x_i(k+1) & \mathbf{x}_{P+i-1^{\mathrm{T}}(k+1)} \end{bmatrix} \mathbf{P}_i, \tag{6.8}$$

where $\mathbf{P}_i$ is a permutation matrix which takes the most recent sample $x_i(k+1)$ of the ith channel to position p_i and left shifts the first $p_i - 1$ elements of $\mathbf{x}_{P+i-1}^{\mathrm{T}}(k+1)$, where

$$p_i = \sum_{r=1}^{i-1} r(N_r - N_{r+1}) + i, \quad i = 1,\, 2,\, \ldots,\, M. \tag{6.9}$$

After processing all M channels, the first P elements of the updated extended input vector constitute the input vector of the next iteration, i.e., $\mathbf{x}_{P+M}^{\mathrm{T}}(k+1) = [\mathbf{x}_P^{\mathrm{T}}(k+1) \quad x_1(k-N_1+1) \quad \cdots \quad x_M(k-N_M+1)]$.

6.2.3 Input vector for block-type multichannel algorithms

For the case of block-channel multichannel algorithms, the expanded input vector, $\mathbf{x}_{P+M}(k+1)$, is given by

$$\begin{aligned}\mathbf{x}^{\mathrm{T}}_{P+M}(k+1) &= \left[x_1(k+1) \quad x_2(k+1) \quad \cdots \quad x_M(k+1) \quad \mathbf{x}^{\mathrm{T}}_P(k)\right]\mathbf{P}\\ &= \left[\mathbf{x}^{\mathrm{T}}_{k+1} \quad \mathbf{x}^{\mathrm{T}}_P(k)\right]\mathbf{P},\end{aligned} \tag{6.10}$$

where $\mathbf{P} = \mathbf{P}_M\mathbf{P}_{M-1}\cdots\mathbf{P}_1$ is a product of M permutation matrices that moves the most recent sample of the ith channel (for $i = 1,2,\ldots,M$) to position p_i (given by Equation (6.9)) in vector $\mathbf{x}_{P+M}(k+1)$.

After the above process is terminated, we have $\mathbf{x}^{\mathrm{T}}_{P+M}(k+1) = [\mathbf{x}^{\mathrm{T}}_P(k+1) \quad x_1(k-N_1+1)\cdots \quad x_M(k-N_M+1)]$, such that the first P elements of $\mathbf{x}^{\mathrm{T}}_{P+M}(k+1)$ provide the input vector for the next iteration. In order to illustrate the role of the permutation matrix $\mathbf{P}$, let us return to the example depicted in Figure 6.2. In this example, the expanded input vector $\mathbf{x}_{P+M}(k+1)$ is obtained by inserting the new samples in positions $p_1 = 1$, $p_2 = 3$, and $p_3 = 6$, respectively, i.e.,

$$\mathbf{P}^{\mathrm{T}}\left[\begin{array}{c} x_1(k+1)\\ x_2(k+1)\\ x_3(k+1)\\ \hline \mathbf{x}_P(k)\end{array}\right] = \mathbf{P}^{\mathrm{T}}\left[\begin{array}{c} x_1(k+1)\\ x_2(k+1)\\ x_3(k+1)\\ \hline x_1(k)\\ x_1(k-1)\\ x_2(k)\\ x_1(k-2)\\ x_2(k-1)\\ x_3(k)\\ x_1(k-3)\\ x_2(k-2)\\ x_3(k-1)\end{array}\right] = \left[\begin{array}{c} x_1(k+1)\\ x_1(k)\\ x_2(k+1)\\ x_1(k-1)\\ x_2(k)\\ x_3(k+1)\\ x_1(k-2)\\ x_2(k-1)\\ x_3(k)\\ \hline x_1(k-3)\\ x_2(k-2)\\ x_3(k-1)\end{array}\right] = \left[\begin{array}{c} \mathbf{x}_P(k+1)\\ \hline x_1(k-3)\\ x_2(k-2)\\ x_3(k-1)\end{array}\right]. \tag{6.11}$$

6.3 Sequential-Type MC-FQRD-RLS Algorithms

In this section, we consider algorithms that process the channels sequentially. In the following, we shall derive the *a priori* [11] and the *a posteriori* [3] versions of sequential MC-FQRD-RLS algorithms based on updating backward error vectors.

Due to close similarities with the single-channel case, basic concepts on how to solve the *backward* and *forward* prediction problems are omitted. Indeed, the sequential processing of multichannel signals corresponds to solving the single-channel algorithm M times, with M being the number of channels. Moreover, from the forward prediction problem, the extended input data matrices used in sequential-channel algorithms are defined as

$$\mathbf{X}_{P+i}(k) = \begin{bmatrix} \mathbf{x}_{P+i}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{x}_{P+i}^{\mathrm{T}}(k-1) \\ \vdots \\ \lambda^{k/2}\mathbf{x}_{P+i}^{\mathrm{T}}(0) \end{bmatrix}, \; i = 1,2,\ldots,M, \tag{6.12}$$

where vector $\mathbf{x}_{P+i}(k)$ is the extended input vector defined in Equation (6.8).

6.3.1 Triangularization of the information matrix

Equation (6.12) suggests that the updating of the information matrix is performed in M forward steps for each iteration.

First step ($i = 1$)

$\mathbf{X}_{P+1}(k)$ can be defined as

$$\mathbf{X}_{P+1}(k) = \begin{bmatrix} x_1(k) & \\ \lambda^{1/2}x_1(k-1) & \mathbf{X}_P(k-1) \\ \vdots & \\ \lambda^{k/2}x_1(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} = \left[\mathbf{d}_f^{(1)}(k) \left| \begin{matrix} \mathbf{X}_P(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{matrix} \right. \right], \tag{6.13}$$

where $\mathbf{d}_{f1}^{(1)}(k) = [x_1(k) \quad \lambda^{1/2}x_1(k-1) \quad \cdots \quad \lambda^{k/2}x_1(0)]$.

Let $\mathbf{Q}_P^{(1)}(k)$ be the orthogonal matrix associated with the Cholesky factor $\mathbf{U}_P$ $(k-1)$ of matrix $\mathbf{X}_P^{\mathrm{T}}(k-1)\mathbf{X}_P(k-1)$. Then, from (6.13), we can write

$$\begin{bmatrix} \mathbf{Q}_P^{(1)}(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{1\times 1} \end{bmatrix} \left[\mathbf{d}_f^{(1)}(k) \left| \begin{matrix} \mathbf{X}_P(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{matrix} \right. \right] = \begin{bmatrix} \mathbf{e}_{fq1}^{(1)}(k) & \mathbf{0} \\ \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ \lambda^{k/2}x_1(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{6.14}$$

To complete the triangularization process of $\mathbf{X}_{P+1}(k)$ leading to $\mathbf{U}_{P+1}(k)$, we pre-multiply (6.14) by two other Givens rotation matrices as follows:

$$\begin{aligned} \begin{bmatrix} \mathbf{0} \\ \mathbf{U}_{P+1}(k) \end{bmatrix} &= \mathbf{Q}_f'^{(1)}(k)\mathbf{Q}_f^{(1)}(k) \begin{bmatrix} \mathbf{e}_{fq1}^{(1)}(k) & \mathbf{0} \\ \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ \lambda^{k/2}x_1(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \\ &= \mathbf{Q}_f'^{(1)}(k) \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ e_{fP}^{(1)}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \end{aligned} \tag{6.15}$$

In the previous equation, $\mathbf{Q}_f{}^{(1)}(k)$ is the orthogonal matrix zeroing $\mathbf{e}_{fq1}^{(1)}(k)$ generating $e_{fP}^{(1)}(k)$. Matrix $\mathbf{Q}'_f{}^{(1)}(k)$ completes the triangularization process by zeroing $\mathbf{d}_{fq2}^{(1)}(k)$ from (6.15) in a top down procedure against $e_{fP}^{(1)}(k)$. Removing the resulting null section in the upper part of (6.15) gives

$$\mathbf{U}_{P+1}(k) = \mathbf{Q}'_{\theta f}{}^{(1)}(k) \begin{bmatrix} \mathbf{d}_{fq2}^{(1)}(k) & \mathbf{U}_P(k-1) \\ e_{fP}^{(1)}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{6.16}$$

From (6.16), we get the following relation that is useful for the updating of $\mathbf{a}_P(k)$ and $\mathbf{f}_P(k)$, the *a priori* and the *a posteriori* error vectors, respectively.

$$[\mathbf{U}_{P+1}(k+1)]^{-1} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \frac{1}{e_{fP}^{(1)}(k+1)} \\ \mathbf{U}_P^{-1}(k) & -\frac{1}{e_{fP}^{(1)}(k+1)}\mathbf{U}_P^{-1}(k)\mathbf{d}_{fq2}^{(1)}(k+1) \end{bmatrix} \left[\mathbf{Q}'_{\theta f}{}^{(1)}(k+1)\right]^{\mathrm{T}} \tag{6.17}$$

Also from (6.16), we see that $\mathbf{Q}'_{\theta f}{}^{(1)}(k)$ is the Givens rotation matrix responsible for zeroing $\mathbf{d}_{fq2}^{(1)}(k)$ against $e_{fP}^{(1)}(k)$, i.e.,

$$\begin{bmatrix} \mathbf{0} \\ e_{f0}^{(1)}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \mathbf{d}_{fq2}^{(1)}(k+1) \\ e_{fP}^{(1)}(k+1) \end{bmatrix}. \tag{6.18}$$

The updating of $\mathbf{d}_{fq2}^{(1)}(k)$ is performed according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(1)}(k+1) \\ \mathbf{d}_{fq2}^{(1)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta_P}^{(0)}(k) \begin{bmatrix} x_1(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq2}^{(1)}(k) \end{bmatrix}, \tag{6.19}$$

where $\mathbf{Q}_{\theta_P}^{(0)}(k) = \mathbf{Q}_{\theta_P}^{(M)}(k-1)$, i.e., the values obtained after processing the Mth channel on last iteration. For $1 < i \leq M$, $\mathbf{d}_{fq2}^{(i)}(k)$ is updated according to

$$\begin{bmatrix} \tilde{e}_{fq1}^{(i)}(k+1) \\ \mathbf{d}_{fq2}^{(i)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta_{P+i-1}}^{(i-1)}(k+1) \begin{bmatrix} x_i(k+1) \\ \lambda^{1/2}\mathbf{d}_{fq2}^{(i)}(k) \end{bmatrix}. \tag{6.20}$$

Following steps ($i > 1$)

The input information matrix $\mathbf{X}_{P+i}(k)$ is related to $\mathbf{X}_{P+i-1}(k)$ according to

$$\mathbf{X}_{P+i}(k) = \begin{bmatrix} x_i(k) & \\ \lambda^{1/2} x_i(k-1) & \\ \vdots & \mathbf{X}_{P+i-1}(k) \\ \lambda^{k/2} x_i(0) & \end{bmatrix} \mathbf{P}_i. \tag{6.21}$$

As in the first step, matrix $\mathbf{X}_{P+i}(k)$ must be triangularized to obtain $\mathbf{U}_{P+i}(k)$ (Cholesky factor of $\mathbf{X}^{\mathrm{T}}_{P+i}(k)\mathbf{X}_{P+i}(k)$). This process is detailed in the following. Let $\mathbf{Q}_{\theta_{P+i-1}}(k)$ denotes the orthogonal matrix associated with the QR decomposition of $\mathbf{X}_{P+i-1}(k)$. From (6.21), we can write

$$\mathbf{Q}_f^{(i)}(k) \begin{bmatrix} \mathbf{Q}_{P+i-1}(k) & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_{P+i}(k) \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} =$$
$$\mathbf{Q}_f^{(i)}(k) \begin{bmatrix} \mathbf{e}^{(i)}_{fq1_{P+i-1}}(k) & \mathbf{0} \\ \mathbf{d}^{(i)}_{fq2}(k) & \mathbf{U}_{P+i-1}(k) \\ \lambda^{k/2}\mathbf{x}_i(0) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{P}_i = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{d}^{(i)}_{fq2}(k) & \mathbf{U}_{P+i-1}(k) \\ e^{(i)}_{f_{P+i-1}}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{P}_i. \tag{6.22}$$

Equation (6.22) is obtained by annihilating $\mathbf{e}^{(i)}_{fq1_{P+i-1}}(k)$ into the first element of the last row of the matrix using an appropriate orthogonal matrix, $\mathbf{Q}_f^{(i)}(k)$, and thereafter removing the resulting null section.

The existence of the permutation matrix $\mathbf{P}_i$ in (6.22) prevents us from directly annihilating $\mathbf{d}^{(i)}_{fq2}(k)$ into $e^{(i)}_{f_{P+i-1}}(k)$ to complete the triangularization of matrix $\mathbf{X}_{P+i}(k)$ (i.e., generating $\mathbf{U}_{P+i}(k)$). Figure 6.3 illustrates the application of Givens rotations under these circumstances. The process is summarized as follows. The permutation factor, $\mathbf{P}_i$, right shifts $\mathbf{d}^{(i)}_{fq2}(k)$ to the ith position as shown in the first part of the figure. Then, the set of $P+i-p_i$ Givens rotation matrices, ${\mathbf{Q}'_{\theta f}}^{(i)}$, nullifies the first $P+i-p_i$ elements of $\mathbf{d}^{(i)}_{fq2}(k)$ by rotating them against $e^{(i)}_{f_{P+i-1}}(k)$ in a top down procedure. The desired triangular structure is finally reached using another permu-

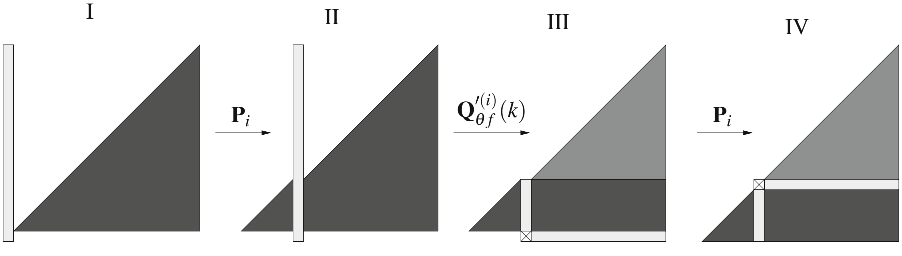


Fig. 6.3 Obtaining the lower triangular factor $\mathbf{U}_{P+i}(k)$. The lighter color tone on top of parts III and IV represents the matrix elements that have been rotated against the bottom line by the set of Givens rotations in $\mathbf{Q}'^{(i)}_{\theta f}(k)$.

tation factor that moves the last row of the matrix to the $P - p_i + 1$ position, after downshifting the previous $P - p_i$ rows. This permutation factor coincides with $\mathbf{P}_i$.

Remark 1. The lower triangular matrix $\mathbf{U}_{P+i}(k)$, obtained as described above must be positive definite. That is guaranteed if its diagonal elements and $e_{f_{P+i-1}}^{(i)}(k)$ are positive. Recalling that $e_{f_{P+i-1}}^{(i)}(k)$ is the absolute value of the forward error, $\mathbf{U}_{P+i}(k)$ will be positive definite if it is initialized properly.

The procedure above can be written in a more compact form as

$$\mathbf{U}_{P+i}(k) = \mathbf{P}_i {\mathbf{Q}'_{\theta f}}^{(i)}(k) \begin{bmatrix} \mathbf{d}_{fq2}^{(i)}(k) & \mathbf{U}_{P+i-1}(k) \\ e_{f_{P+i-1}}^{(i)}(k) & \mathbf{0}^{\mathrm{T}} \end{bmatrix} \mathbf{P}_i. \tag{6.23}$$

From (6.23), the following relation can be derived

$$\begin{aligned} [\mathbf{U}_{P+i}(k+1)]^{-1} &= \mathbf{P}_i^{\mathrm{T}} \\ \times & \begin{bmatrix} \mathbf{0}^{\mathrm{T}} & \frac{1}{e_{f_{P+i-1}}^{(i)}(k+1)} \\ \mathbf{U}_{P+i-1}^{-1}(k+1) & -\frac{\mathbf{U}_{P+i-1}^{-1}(k+1)\mathbf{d}_{fq2}^{(i)}(k+1)}{e_{f_{P+i-1}}^{(i)}(k+1)} \end{bmatrix} \times {\mathbf{Q}'^{\mathrm{T}}_{\theta f}}^{(i)}(k+1)\mathbf{P}_i^{\mathrm{T}}. \end{aligned} \tag{6.24}$$

From (6.23), we realize that ${\mathbf{Q}'_{\theta f}}^{(i)}(k)$ is the Givens rotation matrix responsible for zeroing $\mathbf{d}_{fq2}^{(i)}(k)$ against $e_{fP}^{(i)}(k)$, which yields

$$\begin{bmatrix} \mathbf{0} \\ e_{f0}^{(i)}(k+1) \end{bmatrix} = {\mathbf{Q}'_{\theta f}}^{(i)}(k+1) \begin{bmatrix} \mathbf{d}_{fq2}^{(i)}(k+1) \\ e_{fP}^{(i)}(k+1) \end{bmatrix}. \tag{6.25}$$

6.3.2 A priori and A posteriori versions

The *a priori* and the *a posteriori* versions of the sequential-channel algorithm are based on updating expanded vectors $\mathbf{a}_{P+i}(k+1)$ or $\mathbf{f}_{P+i}(k+1)$ given by

$$\mathbf{a}_{P+i}(k+1) = \lambda^{-1/2}\mathbf{U}_{P+i}^{-T}(k+1)\mathbf{x}_{P+i}(k+1) = \begin{bmatrix} \mathbf{a}^{(i)}(k+1) \\ \mathbf{a}_P(k+1) \end{bmatrix}, \quad \text{and} \tag{6.26}$$

$$\mathbf{f}_{P+i}(k+1) = \mathbf{U}_{P+i}^{-T}(k+1)\mathbf{x}_{P+i}(k+1) = \begin{bmatrix} \mathbf{f}^{(i)}(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}, \tag{6.27}$$

for $i = 1, 2, \ldots, M$.

In (6.26) and (6.27), $\mathbf{a}^{(i)}(k+1)$ and $\mathbf{f}^{(i)}(k+1)$ are vectors containing the first i elements of $\mathbf{a}_{P+i}(k+1)$ $\mathbf{f}_{P+i}(k+1)$, respectively. Recall that for $i=1$, $\mathbf{U}_{P+i}^{-1}(k+1)$ in (6.26) and (6.27) equals $\mathbf{U}_{P+1}^{-1}(k)$ as defined in (6.17).

The updating of $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$ is accomplished in M forward steps at each instant k:

$$\begin{array}{l}\mathbf{a}_P(k) \rightarrow \mathbf{a}_{P+1}(k+1) \rightarrow \;\cdots\; \rightarrow \mathbf{a}_{P+M}(k+1),\\ \mathbf{f}_P(k) \rightarrow \mathbf{f}_{P+1}(k+1) \rightarrow \;\cdots\; \rightarrow \mathbf{f}_{P+M}(k+1).\end{array}$$

From (6.24), (6.8), and (6.27), we get the following recursive expression for $\mathbf{f}_{P+i}(k+1)$:

$$\mathbf{f}_{P+i}(k+1) = \mathbf{P}_i\mathbf{Q}'{}_{\theta f}{}^{(i)}(k+1)\begin{bmatrix}\mathbf{f}_{P+i-1}(k+1)\\ p_{P+i-1}^{(i)}(k+1)\end{bmatrix}, \tag{6.28}$$

where

$$p_{P+i-1}^{(i)}(k+1) = \frac{e_{P+i-1}^{(i)}(k+1)}{|e_{f_{P+i-1}}^{(i)}(k+1)|}, \quad \text{for } i = 1,2,\ldots,M. \tag{6.29}$$

The scalar quantity $e_{P+i-1}^{(i)}(k+1)$ is the *a posteriori* forward prediction error for the ith channel, and $|e_{f_{P+i-1}}^{(i)}(k+1)|$ is given by

$$|e_{f_{P+i-1}}^{(i)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_{P+i-1}}^{(i)}(k)|\right)^2 + |e_{fq1_{P+i-1}}^{(i)}(k+1)|^2}. \tag{6.30}$$

For $i=1$, rather than (6.8), we would use (6.7) in obtaining (6.28), which simply means that $\mathbf{P}_1 = \mathbf{I}$, i.e,

$$\mathbf{f}_{P+1}(k+1) = \mathbf{Q}'_{\theta f}{}^{(1)}(k+1)\begin{bmatrix}\mathbf{f}_P(k)\\ p^{(1)}(k+1)\end{bmatrix}, \tag{6.31}$$

with $e_P{}^{(1)}(k+1)$ denoting the *a posteriori* forward prediction error of the first channel, and $|e_{f_P}^{(1)}(k+1)|$ is given by

$$|e_{f_P}^{(1)}(k+1)| = \sqrt{\left(\lambda^{1/2}|e_{f_P}^{(1)}(k)|\right)^2 + |(e_{fq1_P}^{(i)}(k+1)|^2}. \tag{6.32}$$

Similarly, using (6.24), (6.8), and (6.26), we get the following recursive expression for $\mathbf{a}_{P+i}(k+1)$:

$$\mathbf{a}_{P+i}(k+1) = \lambda^{-1/2}\mathbf{P}_i\mathbf{Q}'{}_{\theta f}{}^{(i)}(k+1)\begin{bmatrix}\mathbf{a}_{P+i-1}(k+1)\\ r_{P+i-1}^{(i)}(k+1)\end{bmatrix}, \tag{6.33}$$

where

$$r^{(i)}_{P+i-1}(k+1) = \frac{e'^{(i)}_{P+i-1}(k+1)}{\gamma^{(i-1)}\sqrt{\lambda}|e^{(i)}_{f_{P+i-1}}(k)|}, \quad \text{for } i = 1, 2, \ldots, M, \tag{6.34}$$

and scalar quantity $e^{(i)}_{P+i-1}(k)$ is the *a priori* forward prediction error for the ith channel. Again, for $i = 1$, we use (6.7) in obtaining (6.33) instead of (6.8), yielding

$$\mathbf{a}_{P+1}(k+1) = \lambda^{-1/2}\mathbf{Q}'_{\theta f}{}^{(1)}(k+1) \begin{bmatrix} \mathbf{a}_P(k) \\ r^{(1)}(k+1) \end{bmatrix}. \tag{6.35}$$

Remark 2. Examining (6.28) and recalling the definitions of $\mathbf{Q}'_{\theta f}{}^{(i)}(k+1)$ and $\mathbf{P}_i$, we realize that the last $p_i - 1$ elements of $\mathbf{f}_{P+i}(k+1)$ and $\mathbf{f}_{P+i-1}(k+1)$ are identical. Indeed, Givens rotations in $\mathbf{Q}'_{\theta f}{}^{(i)}(k+1)$ are such that they act on a smaller portion of $\mathbf{f}_{P+i}(k+1)$ at each i; then $\mathbf{P}_i$ shifts down the unchanged elements which remain unchanged for next i. This fact reduces the computational cost. The same observation holds for vectors $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{a}_{P+i-1}(k+1)$.

For the algorithm based on updating the *a posteriori* backward errors, the Givens rotations matrices $\mathbf{Q}_{\theta_{P+i}}(k+1)$ needed in the next forward steps are obtained from

$$\mathbf{Q}^{(i)}_{\theta_{P+i}}(k+1) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma^{(i)}_{P+i}(k+1) \\ \mathbf{f}_{P+i}(k+1) \end{bmatrix}, \quad \text{for } i \geq 1. \tag{6.36}$$

For the *a priori* case, the equivalent relations are

$$\begin{bmatrix} 1/\gamma^{(i)}_{P+i}(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}^{(i)}_{\theta_{P+i}}(k+1) \begin{bmatrix} 1 \\ \mathbf{a}_{P+i}(k+1) \end{bmatrix}, \quad \text{for } i \geq 1. \tag{6.37}$$

After the Mth channel is processed, the joint process estimation is performed according to

$$\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}^{(0)}_{\theta}(k+1) \begin{bmatrix} e_{q1}(k) \\ \mathbf{d}_{q2}(k) \end{bmatrix}. \tag{6.38}$$

The *a posteriori* and the *a priori* multiple order sequential algorithms are summarized in Tables 6.2 and 6.3, respectively.

6.3.3 Alternative implementations

As observed before, when all channels are of equal order, matrix $\mathbf{P}_i$ in (6.23) degenerates to identity and, after carrying out the multiplication by $\mathbf{Q}'_{\theta f}{}^{(i)}(k)$ on the right side of that equation, $\mathbf{U}_{P+i}(k)$ can be partitioned as follows.

Table 6.2 Algorithm number **8** of Table 6.1 [10].

The multiple order sequential-type MC-FQRD_POS_B

```
Initializations:
$\mathbf{d}_{fq2}^{(i)} = zeros(P,1)$;   $\mathbf{f}_j^{(M)}(0) = \mathbf{0}$;   $\mathbf{d}_{q2} = \mathbf{0}$;   $\gamma_P^{(0)}(0) = 1$;
$e_{f_P}^{(i)}(0) = \mu$; $i = 1,2,\cdots,M$,  all cosines = 1,   and all sines = 0.
for $k = 1,2,\cdots$
{ $\gamma_0^{(1)} = 1$;   $e_{q1}^{(0)}(k+1) = d(k+1)$;
  for $i = 1:M$,
  { $e_{fq1_0}^{(i)}(k+1) = x_i(k+1)$;
    for $j = 1:P$, % Obtaining $e_{fq1}^{(i)}(k+1)$ and $\mathbf{d}_{fq2}^{(i)}(k+1)$:
    {$e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2}\sin\left[\theta_j^{(i-1)}(k)\right] \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k)$;
      $\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1) = \lambda^{1/2}\cos\left[\theta_j^{(i-1)}(k)\right] \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k) - \sin^*\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1)$;
    }
    $\|e_{f_P}^{(i)}(k+1)\| = \sqrt{\left(\lambda^{1/2}\|e_{f_P}^{(i)}(k)\|\right)^2 + \|e_{fq1_P}^{(i)}(k+1)\|^2}$;
    for $j = P:-1:p_i$, % Obtaining ${\mathbf{Q}'_{\theta f}}^{(i)}(k+1)$:
    {$\|e_{f_{j-1}}^{(i)}(k+1)\| = \sqrt{\|e_{f_j}^{(i)}(k+1)\|^2 + \|\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1)\|^2}$;
      $\cos{\theta'_f}_j^{(i)}(k+1) = \|e_{f_j}^{(i)}(k+1)\| / \|e_{f_{j-1}}^{(i)}(k+1)\|$;
      $\sin{\theta'_f}_j^{(i)}(k+1) = \left[\cos{\theta'_f}_j^{(i)}(k+1)\ \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1) / e_{f_j}^{(i)}(k+1)\right]^*$;
    }
    $p_P^{(i)}(k+1) = \gamma_P^{(i-1)}(k)\left[e_{fq1_P}^{(i)}(k+1)\right]^* / \|e_{f_P}^{(i)}(k+1)\|$;
    for $j = P:-1:p_i$, % Obtaining $\mathbf{f}^{(i)}(k+1)$:
    {$\mathbf{f}_{P-j+1}^{(i)}(k+1) = \cos{\theta'_f}_j^{(i)}(k+1)\mathbf{f}_{P-j+2}^{(i-1)}(k+1) - \left[\sin{\theta'_f}_j^{(i)}(k+1)\right]^* p_j^{(i)}(k+1)$;
      $p_{j-1}^{(i)}(k+1) = \sin{\theta'_f}_j^{(i)}(k+1)\mathbf{f}_{P-j+2}^{(i-1)}(k+1) + \cos{\theta'_f}_j^{(i)}(k+1)p_j^{(i)}(k+1)$;
    }
    $\mathbf{f}_{P+1-p_i+1}^{(i)}(k+1) = p_{p_i-1}^{(i)}(k+1)$;
    for $j = p_i:P$, % Obtaining $\mathbf{Q}_\theta^{(i)}(k)$:
    {$\sin\theta_j^{(i)}(k) = -\left[\mathbf{f}_{P-j+2}^{(i)}(k+1)\right]^* / \gamma_{j-1}^{(i)}(k)$;

      $\cos\theta_j^{(i)}(k) = \sqrt{1 - \|\sin\theta_j^{(i)}(k)\|^2}$;
      $\gamma_j^{(i)}(k) = \cos\theta_j^{(i)}(k)\gamma_{j-1}^{(i)}(k)$;
    }
  } for $i$
  for $j = 1:P$, % Joint process estimation:
  {$e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1)e_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k)$;
    $\mathbf{d}_{q2}^{(P-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k) - \left[\sin\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1)$;
  }
  $\mathbf{e}(k+1) = \left[e_{q1}^{(P)}(k+1)\right]^* / \gamma_P^{(0)}(k)$;
} for $k$
```

Obs.: $\theta_j^{(M)}(k) = \theta_j^{(0)}(k+1)$ and $\mathbf{f}_{P-j+2}^{(M)}(k) = \mathbf{f}_{P-j+2}^{(0)}(k+1)$. The *asterisk* $(*)$ denotes complex conjugation.

$$\mathbf{U}_{P+i}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ e_{f0}^{(i)}(k+1) & \mathbf{C} \end{bmatrix}, \quad \text{for } i = 1,2,\ldots,M. \tag{6.39}$$

Taking the inverse of (6.39) yields

$$[\mathbf{U}_{P+i}(k+1)]^{-1} = \begin{bmatrix} -\left[e_{f0}^{(i)}(k+1)\right]^{-1}\mathbf{C}\mathbf{B}^{-1} & \left[e_{f0}^{(i)}(k+1)\right]^{-1} \\ \mathbf{B}^{-1} & \mathbf{0} \end{bmatrix}. \tag{6.40}$$

Table 6.3 Algorithm number **16** of Table 6.1 [11].

The multiple order sequential-type MC-FQRD_PRI_B

```
Initializations:
$\mathbf{d}_{fq2}^{(i)} = zeros(P,1)$;   $\mathbf{a}_j^{(M)}(0) = \mathbf{0}$;   $\mathbf{d}_{q2} = \mathbf{0}$;   $\gamma_P^{(0)}(0) = 1$;
$e_{f_P}^{(i)}(0) = \mu$; $i = 1,2,\cdots,M$,  all cosines = 1,   and all sines = 0.
for $k = 1,2,\cdots$
{ $\gamma_0^{(1)} = 1$;   $e_{q1}^{(0)}(k+1) = d(k+1)$;
  for $i = 1:M$,
  { $e_{fq1_0}^{(i)}(k+1) = x_i(k+1)$;
    for $j = 1:P$, % Obtaining $e_{fq1}^{(i)}(k+1)$ and $\mathbf{d}_{fq2}^{(i)}(k+1)$:
    {$e_{fq1_j}^{(i)}(k+1) = \cos\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1) + \lambda^{1/2} \sin\left[\theta_j^{(i-1)}(k)\right] \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k)$;
      $\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1) = \lambda^{1/2} \cos\left[\theta_j^{(i-1)}(k)\right] \mathbf{d}_{fq2_{P-j+1}}^{(i)}(k) - \sin^*\left[\theta_j^{(i-1)}(k)\right] e_{fq1_{j-1}}^{(i)}(k+1)$;
    }
    $r_P^{(i)}(k+1) = \lambda^{-1/2}[e_{fq1_j}^{(i)}(k+1)]^* / \left[\gamma_P^{(i-1)}(k)\|e_{f_P}^{(i)}(k)\|\right]$;
    for $j = P:-1:p_i$, % Obtaining $\mathbf{a}^{(i)}(k+1)$:
    {$\mathbf{a}_{P-j+1}^{(i)}(k+1) = \cos\theta'^{(i)}_{f\,j}(k)\mathbf{a}_{P-j+2}^{(i-1)}(k+1) - \left[\sin\theta'^{(i)}_{f\,j}(k)\right]^* r_j^{(i)}(k+1)$;
      $r_{j-1}^{(i)}(k+1) = \sin\theta'^{(i)}_{f\,j}(k)\mathbf{a}_{P-j+2}^{(i-1)}(k+1) + \cos\theta'^{(i)}_{f\,j}(k) r_j^{(i)}(k+1)$;
    }
    $\mathbf{a}_{P+1-p_i+1}^{(i)}(k+1) = r_{p_i-1}^{(i)}(k+1)$;
    $\|e_{f_P}^{(i)}(k+1)\| = \sqrt{\left(\lambda^{1/2}\|e_{f_P}^{(i)}(k)\|\right)^2 + \|e_{fq1_P}^{(i)}(k+1)\|^2}$;
    for $j = P:-1:p_i$, % Obtaining $\mathbf{Q}'^{(i)}_{\theta f}(k+1)$:
    {$\|e_{f_{j-1}}^{(i)}(k+1)\| = \sqrt{\|e_{f_j}^{(i)}(k+1)\|^2 + \|\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1)\|^2}$;
      $\cos\theta'^{(i)}_{f\,j}(k+1) = \|e_{f_j}^{(i)}(k+1)\| / \|e_{f_{j-1}}^{(i)}(k+1)\|$;
      $\sin\theta'^{(i)}_{f\,j}(k+1) = \left[\cos\theta'^{(i)}_{f\,j}(k+1)\;\mathbf{d}_{fq2_{P-j+1}}^{(i)}(k+1) / \|e_{f_j}^{(i)}(k+1)\|\right]^*$;
    }
    for $j = p_i:P$, % Obtaining $\mathbf{Q}_\theta^{(i)}(k)$:
    {$\gamma_j^{(i)}(k) = 1/\sqrt{(1/\gamma_{j-1}^{(i)}(k))^2 + (\mathbf{a}_{P-j+2}^{(i)}(k+1))^2}$
      $\cos\theta_j^{(i)}(k) = \gamma_j^{(i)}(k)/\gamma_{j-1}^{(i)}(k)$;
      $\sin\theta_j^{(i)}(k) = \left[\mathbf{a}_{P-j+2}^{(i)}(k+1)\cos\theta_j^{(i)}(k)/\gamma_{j-1}^{(i)}(k)\right]^*$;
    }
  } for $i$
  for $j = 1:P$, % Joint process estimation:
  {$e_{q1}^{(j)}(k+1) = \cos\theta_j^{(0)}(k+1) e_{q1}^{(j-1)}(k+1) + \lambda^{1/2}\sin\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k)$;
    $\mathbf{d}_{q2}^{(P-j+1)}(k+1) = \lambda^{1/2}\cos\theta_j^{(0)}(k+1)\mathbf{d}_{q2}^{(P-j+1)}(k) - \left[\sin\theta_j^{(0)}(k+1)\right]^* e_{q1}^{(j-1)}(k+1)$;
  }
  $\mathbf{e}(k+1) = \left[e_{q1}^{(P)}(k+1)\right]^* / \gamma_P^{(0)}(k+1)$;
} for $k$
```

Obs.: $\theta_j^{(M)}(k) = \theta_j^{(0)}(k+1)$ and $\mathbf{a}_{P-j+2}^{(M)}(k) = \mathbf{a}_{P-j+2}^{(0)}(k+1)$. The *asterisk* $(*)$ denotes complex conjugation.

Using (6.40), and recalling (6.26), (6.27), and the definition of the input vector given in (6.8), we can now write vectors $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$, respectively, as

$$\mathbf{a}_{P+i}(k+1) = \begin{bmatrix} * \\ x_i(k+1)/\left[\sqrt{\lambda}e_{f0}^{(i)}(k)\right] \end{bmatrix}, \text{ and} \tag{6.41}$$

$$\mathbf{f}_{P+i}(k+1) = \begin{bmatrix} * \\ x_i(k+1)/e_{f0}^{(i)}(k). \end{bmatrix} \tag{6.42}$$

As we can see from (6.41) and (6.42), the last element of $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$ are known at each iteration i (for $i = 1,2,\ldots,M$) prior to the updating process. That observation is the key step leading to two alternative implementations of these algorithms for the special case when the channels are of the same order. Thus, the recursive updating of vectors $\mathbf{a}_{P+i}(k+1)$ and $\mathbf{f}_{P+i}(k+1)$ are performed now based on this assumption.

6.4 Block-Type MC-FQRD-RLS Algorithms

This section discusses a general framework for block-type multichannel algorithms using the extended input signal vector $\mathbf{x}_{P+M}(k+1)$ defined in Section 6.2.3. These algorithms, despite exhibiting a higher computational burden as compared to the sequential-type ones, have some attractive features, e.g., suitability for parallel implementation.

To begin with, in Section 6.4.1 we shall revisit the backward and forward prediction problems applied to a block-multichannel scenario from where some fundamental equations are derived. The *a priori* and the *a posteriori* versions are discussed in Section 6.4.2 followed by a brief overview of some alternative implementations in Section 6.4.3.

6.4.1 The backward and forward prediction problems

Using the definition of the input vector for the multiple order channels case as given by (6.10), define the input data matrix $\mathbf{X}_{P+M}(k+1)$ as follows.

$$\mathbf{X}_{P+M}(k+1) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}_{P+M}(k+1) \\ \lambda^{1/2}\mathbf{x}^{\mathrm{T}}_{P+M}(k) \\ \vdots \\ \lambda^{(k+1)/2}\mathbf{x}^{\mathrm{T}}_{P+M}(0) \end{bmatrix} \tag{6.43}$$

The above matrix can be partitioned into two distinct ways, depending onto the prediction problem, **backward** or **forward**, to be solved.

Define the **backward prediction** error matrix for block processing of equal order channels as

$$\begin{aligned} \mathbf{E}^{b}(k+1) &= \mathbf{D}_{b}(k+1) - \mathbf{X}_{P}(k+1)\mathbf{W}_{b}(k+1) \\ &= \begin{bmatrix} \mathbf{X}_{P}(k+1) & \mathbf{D}_{b}(k+1) \end{bmatrix} \begin{bmatrix} -\mathbf{W}_{b}(k+1) \\ \mathbf{I} \end{bmatrix} \end{aligned} \tag{6.44}$$

where $\mathbf{D}_{b}(k+1)$ and $\mathbf{W}_{b}(k+1)$ are the respective desired response and coefficient vector matrices of the backward prediction problem.

Using the relation in (6.44), and assuming that the post-multiplication by permutation matrix $\mathbf{P}$ is already carried out, $\mathbf{X}_{P+M}(k+1)$ can be partitioned as

$$\mathbf{X}_{P+M}(k+1) = \left[\mathbf{X}_P(k+1)\ \mathbf{D}_b(k+1)\right]. \tag{6.45}$$

The process of obtaining lower triangular matrix $\mathbf{U}_{P+M}(k+1)$ from $\mathbf{X}_{P+M}(k+1)$ is performed as follows:

$$\begin{aligned}\mathbf{Q}_b(k+1)\mathbf{Q}(k)\mathbf{X}_{P+M}(k+1) &= \mathbf{Q}_b(k+1)\begin{bmatrix}\mathbf{0} & \mathbf{E}_{bq1}(k+1)\\ \mathbf{U}_P(k+1) & \mathbf{D}_{bq2}(k+1)\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{E}_b(k+1)\\ \mathbf{U}_P(k+1) & \mathbf{D}_{bq2}(k+1)\end{bmatrix},\end{aligned} \tag{6.46}$$

where $\mathbf{Q}(k)$ contains $\mathbf{Q}_P(k+1)$ as a submatrix which triangulates $\mathbf{X}_P(k+1)$, generating $\mathbf{U}_P(k+1)$. Matrix $\mathbf{Q}_b(k+1)$ is responsible for generating the lower triangular matrix $\mathbf{E}_b(k+1)$ from $\mathbf{E}_{bq1}(k+1)$.

By removing the ever-increasing null section in (6.46), $\mathbf{U}_{P+M}(k+1)$ can be finally written as follows:

$$\mathbf{U}_{P+M}(k+1) = \begin{bmatrix}\mathbf{0} & \mathbf{E}_b(k+1)\\ \mathbf{U}_P(k+1) & \mathbf{D}_{bq2}(k+1)\end{bmatrix}. \tag{6.47}$$

The inverse of $\mathbf{U}_{P+M}(k+1)$ as given by (6.47) will be useful in further steps and is defined as

$$[\mathbf{U}_{P+M}(k+1)]^{-1} = \begin{bmatrix}-\mathbf{U}_P^{-1}(k+1)\mathbf{D}_{bq2}(k+1)\mathbf{E}_b^{-1}(k+1) & \mathbf{U}_P^{-1}(k+1)\\ \mathbf{E}_b^{-1}(k+1) & \mathbf{0}\end{bmatrix}. \tag{6.48}$$

Now, define **forward prediction** error matrix $\mathbf{E}^f(k+1)$ as

$$\begin{aligned}\mathbf{E}^f(k+1) &= \mathbf{D}_f(k+1) - \begin{bmatrix}\mathbf{X}_P(k)\\ \mathbf{0}\end{bmatrix}\mathbf{W}_f(k+1) = \begin{bmatrix}\mathbf{D}_f(k+1) & \begin{matrix}\mathbf{X}_P(k)\\ \mathbf{0}^{\mathrm{T}}\end{matrix}\end{bmatrix}\\ &\times\begin{bmatrix}\mathbf{I}\\ \mathbf{W}_f(k+1)\end{bmatrix} = \mathbf{X}_{P+M}(k+1)\begin{bmatrix}\mathbf{I}\\ \mathbf{W}_f(k+1)\end{bmatrix},\end{aligned} \tag{6.49}$$

where $\mathbf{D}_f(k+1)$ and $\mathbf{W}_f(k+1)$ are the desired response and the coefficient vector matrix of the backward prediction problem, respectively. A modified input data matrix $\bar{\mathbf{X}}_{P+M}(k+1)$ incorporating permutation matrix $\mathbf{P}$ is defined as follows.[2]

[2] Note that $\bar{\mathbf{X}}_{P+M}(k+1)$ is formed by adding $M-1$ rows of zeros to $\mathbf{X}_{P+M}(k+1)$ such that $\mathbf{U}_{P+M}(k+1)$ has the correct dimension in (6.55), i.e., $(P+M)\times(P+M)$.

$$\bar{\mathbf{X}}_{P+M}(k+1) = \begin{bmatrix} \mathbf{x}_{P+M}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{x}_{P+M}^{\mathrm{T}}(k) \\ \vdots \\ \lambda^{(k+1)/2}\mathbf{x}_{P+M}^{\mathrm{T}}(0) \\ \mathbf{0}_{(M-1)\times(P+M)} \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{D}_f(k+1) & \begin{array}{c}\mathbf{X}_P(k) \\ \mathbf{0}^{\mathrm{T}}\end{array} \\ \hline \multicolumn{2}{c}{\mathbf{0}_{(M-1)\times(P+M)}} \end{array}\right] \mathbf{P} \tag{6.50}$$

In order to triangulate $\bar{\mathbf{X}}_{P+M}(k+1)$ in (6.50) and obtain $\mathbf{U}_{P+M}(k+1)$, three sets of Givens rotation matrices $\mathbf{Q}(k)$, $\mathbf{Q}_f(k+1)$, and $\mathbf{Q}'_f(k+1)$ are needed [4, 5, 11]. The role of each matrix in the triangulation process is illustrated in the following equation.

$$\mathbf{Q}'_f(k+1)\mathbf{Q}_f(k+1)\mathbf{Q}(k)\bar{\mathbf{X}}_{P+M}(k+1) = \mathbf{Q}'_f(k+1)\mathbf{Q}_f(k+1) \\ \times \begin{bmatrix} \mathbf{E}_{fq1}(k+1) & \mathbf{0} \\ \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \lambda^{(k+1)/2}\mathbf{x}_0^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} \\ \multicolumn{2}{c}{\mathbf{0}_{(M-1)\times(P+M)}} \end{bmatrix} \mathbf{P} = \mathbf{Q}'_f(k+1) \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \mathbf{E}_f(k+1) & \mathbf{0} \end{bmatrix} \mathbf{P} \tag{6.51}$$

In (6.51), $\mathbf{Q}(k)$ contains $\mathbf{Q}_P(k)$ as a sub-matrix which triangulates $\mathbf{X}_P(k)$, generating $\mathbf{U}_P(k)$. Matrix $\mathbf{Q}_f(k+1)$ is responsible for the zeroing of matrix $\mathbf{E}_{fq1}(k+1)$. Note that, when working with fixed-order (or fixed-dimension, as opposed to the ever-increasing dimension of $\mathbf{Q}_P(k)$), this is equivalent to annihilating $\mathbf{e}_{fq1}^{\mathrm{T}}(k+1)$, the first row of $\mathbf{E}_{fq1}(k+1)$, against the diagonal of $\lambda^{1/2}\mathbf{E}_f(k)$, generating $\mathbf{E}_f(k+1)$, as shown in (6.54).

Removing the ever-increasing null section in (6.51) and using the fixed-order matrix $\mathbf{Q}'_{\theta f}(k+1)$ embedded in $\mathbf{Q}'_f(k+1)$, we obtain

$$\bar{\mathbf{U}}_{P+M}(k+1) = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \mathbf{E}_f(k+1) & \mathbf{0} \end{bmatrix} \mathbf{P}. \tag{6.52}$$

Also starting from (6.51) and by using the fixed-order matrices $\mathbf{Q}_\theta(k)$ embedded in $\mathbf{Q}_P(k)$ and $\overline{\mathbf{Q}}_f(k+1)$ embedded in $\mathbf{Q}_f(k+1)$, we obtain after some algebraic manipulations the following equations:

$$\begin{bmatrix} \mathbf{e}_{fq1}^{\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} \mathbf{x}_{k+1}^{\mathrm{T}} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}, \tag{6.53}$$

where $\mathbf{x}_{k+1}^{\mathrm{T}} = [x_1(k+1) \quad x_2(k+1)\cdots x_M(k+1)]$ is the *forward reference signal* and $\mathbf{e}_{fq1}^{\mathrm{T}}(k+1)$ is the *rotated forward error*, and

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{E}_f(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} \mathbf{e}_{fq1}^{\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{E}_f(k) \end{bmatrix}. \tag{6.54}$$

Let $\mathbf{E}_x^f(k+1) = [\mathbf{E}^f(k+1)]^{\mathrm{T}}\mathbf{E}^f(k+1)$ be the *forward prediction error covariance* matrix, where $\mathbf{E}^f(k+1)$ is the forward prediction error matrix defined in (6.49). Using $\bar{\mathbf{U}}_{P+M}(k+1)$ as defined in (6.52) instead of $\bar{\mathbf{X}}_{P+M}(k+1)$, it is straightforward to show that $\mathbf{E}_x^f(k+1) = \mathbf{E}_f^{\mathrm{T}}(k+1)\mathbf{E}_f(k+1)$. Thus, $M \times M$ lower triangular matrix $\mathbf{E}_f(k+1)$ in (6.52) and (6.54) can be interpreted as the Cholesky factor of the *forward prediction error covariance* matrix.

Note that the permutation matrix $\mathbf{P}$ in (6.52) prevents a direct annihilation of the first M columns – corresponding to matrix $\mathbf{D}_{fq2}(k+1) = [\mathbf{d}_{fq2}^{(1)}(k+1)\ \mathbf{d}_{fq2}^{(2)}(k+1)\ \cdots\ \mathbf{d}_{fq2}^{(M)}(k+1)]$ – against the anti-diagonal of $\mathbf{E}_f(k+1)$ using the set of Givens rotations $\mathbf{Q}'_{\theta f}(k+1) = \mathbf{Q}'_{\theta f}{}^{(N)}(k+1)\cdots\mathbf{Q}'_{\theta f}{}^{(2)}(k+1)\mathbf{Q}'_{\theta f}{}^{(1)}(k+1)$. From (6.52) it can be seen that this permutation factor, $\mathbf{P} = \mathbf{P}_M\mathbf{P}_{M-1}\cdots\mathbf{P}_1$, will right-shift the first M columns to position p_i, for $i = M$ to 1, in this order. Thus, only the first $P+i-p_i$ elements of each $\mathbf{d}_{fq2}^{(i)}(k+1)$ will be rotated against the anti-diagonal of $\mathbf{E}_f(k+1)$ using the set of Givens rotations in $\mathbf{Q}'_{\theta f}(k+1)$. It is straightforward to see that, when the position $p_i = i$, the corresponding permutation factor $\mathbf{P}_i$ degenerates to an identity matrix. If this is true for all M channels, this formulation leads to the equal-order algorithms of [4–6, 11].

The process explained above is illustrated in Figure 6.4 (parts I–III) for a three-channel case with the first two channels having equal length, i.e., $p_1 = 1$ and $p_2 = 2$; consequently, $\mathbf{P}_1 = \mathbf{P}_2 = \mathbf{I}$. Part I of this figure shows the initial state as in (6.52) but with reduced dimension, and the operations involving matrices $\mathbf{Q}'_{\theta f}(k+1)$ and $\mathbf{P}$ are illustrated in parts II and III, respectively. As we can see, the resulting matrix $\bar{\mathbf{U}}_{P+M}(k+1)$ in (6.52) does not have the desired lower triangular shape, as depicted in part III of this figure. Hence, another permutation factor, $\overline{\mathbf{P}}$, is needed for up-shifting the $(P+M-i+1)$th row to the $(P+M-p_i+1)$th position (see Figure 6.4 – parts III–IV) leading to

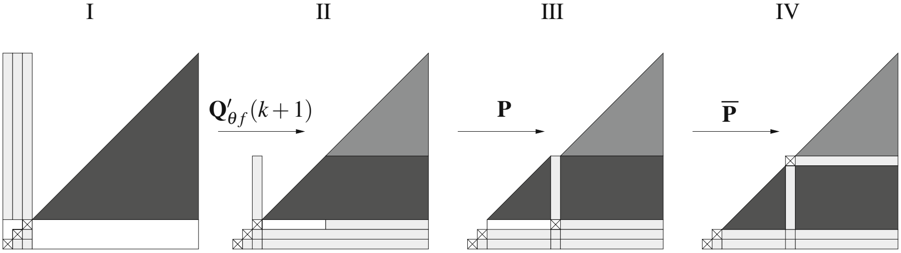


Fig. 6.4 Obtaining the lower triangular $\mathbf{U}_{P+M}(k+1)$. The lighter color tone on top of parts II–IV denotes the matrix elements that have been rotated against the third line from the bottom by the third (and last) set of Givens rotations embedded in $\mathbf{Q}'_{\theta f}(k+1)$.

$$\mathbf{U}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix} \mathbf{D}_{fq2}(k+1) & \mathbf{U}_P(k) \\ \mathbf{E}_f(k+1) & \mathbf{0} \end{bmatrix}\mathbf{P}, \tag{6.55}$$

where permutation matrix $\overline{\mathbf{P}} = \mathbf{P}_1\mathbf{P}_2\cdots\mathbf{P}_M$.

From (6.55), it is possible to obtain

$$\begin{aligned} [\mathbf{U}_{P+M}(k+1)]^{-1} &= \mathbf{P}^{\mathrm{T}} \\ &\times \begin{bmatrix} \mathbf{0} & \mathbf{E}_f^{-1}(k+1) \\ \mathbf{U}_P^{-1}(k) & -\mathbf{U}_P^{-1}(k)\mathbf{D}_{fq2}(k+1)\mathbf{E}_f^{-1}(k+1) \end{bmatrix} \mathbf{Q}'^{\mathrm{T}}_{\theta f}(k+1)\overline{\mathbf{P}}^{\mathrm{T}}, \end{aligned} \tag{6.56}$$

which will be used in the next section to derive the *a priori* and the *a posteriori* versions of the algorithm. Also from (6.55), we can write

$$\begin{bmatrix} \mathbf{0} \\ * \\ \mathbf{E}_f^0(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \mathbf{E}_f(k+1) \end{bmatrix}, \tag{6.57}$$

where $\mathbf{E}_f^0(k+1)$ is the Cholesky factor of the zero-order error covariance matrix.[3] The *asterisk* $*$ denotes possible non-zero elements according to the process explained above.

6.4.2 A priori and A posteriori versions

If matrix $\mathbf{U}_{P+M}(k)$ is used to denote the Cholesky factor of $\mathbf{X}^{\mathrm{T}}_{P+M}(k)\mathbf{X}_{P+M}(k)$, we can define the *a priori* and *a posteriori* backward error vectors, $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$, as follows:

$$\mathbf{a}_{P+M}(k+1) = \lambda^{-1/2}\mathbf{U}^{-T}_{P+M}(k)\mathbf{x}_{P+M}(k+1), \text{ and} \tag{6.58}$$

$$\mathbf{f}_{P+M}(k+1) = \mathbf{U}^{-T}_{P+M}(k+1)\mathbf{x}_{P+M}(k+1). \tag{6.59}$$

Vectors $\mathbf{a}_P(k+1)$ and $\mathbf{f}_P(k+1)$ are contained within matrix $\mathbf{Q}_\theta(k+1)$ [5, 11].

From (6.10), (6.56), and (6.58), we can write

$$\mathbf{a}_{P+M}(k+1) = \overline{\mathbf{P}}\lambda^{-1/2}\mathbf{Q}'_{\theta f}(k)\begin{bmatrix} \mathbf{a}_P(k) \\ \mathbf{r}(k+1) \end{bmatrix}, \tag{6.60}$$

[3] The term is coined due to the fact that, in the single-channel case, the corresponding scalar $\|\mathbf{e}_f^{(0)}(k+1)\| = \sum_{i=0}^{k+1}\lambda^{(k+1-i)}x^2(i)$ is the norm of the zero-order forward prediction error which is an estimate (albeit biased) of the input variance. Also note that, for zero-order prediction, the forward prediction error vector equals its backward counterpart, and $\|\mathbf{e}_f^{(0)}(k)\| = \|\mathbf{e}_b^{(0)}(k)\|$.

where

$$\begin{aligned}\mathbf{r}(k+1) &= \lambda^{-1/2}\mathbf{E}_f^{-T}(k)\left[\mathbf{x}_{k+1} - \mathbf{W}_f^T(k)\mathbf{x}_P(k)\right] \\ &= \lambda^{-1/2}\mathbf{E}_f^{-T}(k)\mathbf{e}'_f(k+1),\end{aligned} \tag{6.61}$$

with $\mathbf{e}'_f(k+1)$ being the *a priori* forward error vector. Thus, $\mathbf{r}(k+1)$ can be interpreted as the Nth-order normalized *a priori* forward error vector. Matrix, $\mathbf{W}_f(k)$ given by

$$\mathbf{W}_f(k) = \mathbf{U}_P^{-1}(k-1)\mathbf{D}_{fq2}(k), \tag{6.62}$$

contains the coefficient vectors of the forward prediction problem. The matrix inversion operation in (6.61) can be avoided using the solution in [11], i.e.,

$$\begin{bmatrix} * \\ \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1)\begin{bmatrix} 1/\gamma(k) \\ -\mathbf{r}(k+1) \end{bmatrix}. \tag{6.63}$$

Combining (6.48), (6.58), and the definition of the input vector in (6.10), $\mathbf{a}_{P+M}(k+1)$ can be expressed as

$$\mathbf{a}_{P+M}(k+1) = \begin{bmatrix} \mathbf{a}^{(N)}(k+1) \\ \mathbf{a}_P(k+1) \end{bmatrix}, \tag{6.64}$$

where the $M \times 1$ element vector of $\mathbf{a}_{P+M}(k+1)$, $\mathbf{a}^{(N)}(k+1)$, is given by

$$\begin{aligned}\mathbf{a}^{(N)}(k+1) &= \lambda^{-1/2}\mathbf{E}_b^{-T}(k)\left[\mathbf{x}_{k-N+1} - \mathbf{W}_b^T(k)\mathbf{x}_P(k+1)\right] \\ &= \lambda^{-1/2}\mathbf{E}_b^{-T}(k)\mathbf{e}'_b(k+1),\end{aligned} \tag{6.65}$$

with $\mathbf{e}'_b(k+1)$ being the Nth-order *a priori* backward error vector and matrix $\mathbf{W}_b(k)$ contains the coefficient vectors of the backward prediction problem. From the last equation, $\mathbf{a}^{(N)}(k+1)$ can be thought as the Nth-order normalized *a priori* backward error vector. [4]

Using similar procedure, combining Equations (6.10), (6.56), and (6.59), yields

$$\mathbf{f}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix} \mathbf{f}_P(k) \\ \mathbf{p}(k+1) \end{bmatrix}, \tag{6.66}$$

where

$$\begin{aligned}\mathbf{p}(k+1) &= \mathbf{E}_f^{-T}(k+1)\left[\mathbf{x}_{k+1} - \mathbf{W}_f^T(k+1)\mathbf{x}_P(k)\right] \\ &= \mathbf{E}_f^{-T}(k+1)\mathbf{e}_f(k+1),\end{aligned} \tag{6.67}$$

[4] As shall be seen in Section 6.5, this argument can be taken further to demonstrate that $\mathbf{a}_{P+M}(k+1)$ is actually a nesting of vectors $\mathbf{a}^{(j)}(k+1)$ of size $M \times 1$, for $j = 0, 1, .., N$. Similar observation holds for vector $\mathbf{f}_{P+M}(k+1)$.

with $\mathbf{e}_f(k+1)$ being the *a posteriori* forward error vector. Therefore, $\mathbf{p}(k+1)$ is interpreted as the Nth-order normalized *a posteriori* forward error vector. Matrix $\mathbf{W}_f(k+1)$ contains the coefficient vectors of the forward prediction problem.

Now, from (6.48), (6.59), and the definition of the input vector in (6.10), $\mathbf{f}_{P+M}(k+1)$ can be partitioned as

$$\mathbf{f}_{P+M}(k+1) = \begin{bmatrix} \mathbf{f}^{(N)}(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}, \tag{6.68}$$

where vector $\mathbf{f}^{(N)}(k+1)$ is given by

$$\begin{aligned} \mathbf{f}^{(N)}(k+1) &= \mathbf{E}_b^{-T}(k+1)\left[\mathbf{x}_{k-N+1} - \mathbf{W}_b^T(k+1)\mathbf{x}_P(k+1)\right] \\ &= \mathbf{E}_b^{-T}(k+1)\mathbf{e}_b(k+1), \end{aligned} \tag{6.69}$$

with $\mathbf{e}_b(k+1)$ being the Nth-order *a posteriori* backward error vector and matrix $\mathbf{W}_b(k+1)$ contains the coefficient vectors of the backward prediction problem. Hence, $\mathbf{f}^{(N)}(k+1)$ can be regarded as the Nth-order normalized *a posteriori* backward error vector.

To solve for $\mathbf{p}(k+1)$ avoiding the matrix inversion in (6.67), we can use

$$\overline{\mathbf{Q}}_f(k+1)\begin{bmatrix} \gamma(k) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix}. \tag{6.70}$$

Proof. From (6.54), it is clear that $\mathbf{E}_f(k+1)$ is the Cholesky factor of

$$[\tilde{\mathbf{e}}_{fq1} \quad \lambda^{1/2}\mathbf{E}_f^{\mathrm{T}}(k)][\tilde{\mathbf{e}}_{fq1} \quad \lambda^{1/2}\mathbf{E}_f^{\mathrm{T}}(k)]^{\mathrm{T}}. \tag{6.71}$$

Hence, (6.54) can be written in a product form as follows [14]:

$$\mathbf{E}_f^{\mathrm{T}}(k+1)\mathbf{E}_f(k+1) = \tilde{\mathbf{e}}_{fq1}(k+1)\tilde{\mathbf{e}}_{fq1}^{\mathrm{T}}(k+1) + \lambda\mathbf{E}_f^{\mathrm{T}}(k)\mathbf{E}_f(k). \tag{6.72}$$

Pre-multiply and post-multiply (6.72) by $\mathbf{E}_f^{-T}(k+1)\gamma^2(k)$ and $\mathbf{E}_f^{-1}(k+1)$, respectively. After some algebraic manipulations, we have

$$\gamma^2(k)\mathbf{I} = \mathbf{p}(k+1)\mathbf{p}^{\mathrm{T}}(k+1) + \Psi \tag{6.73}$$

where $\Psi = \lambda\gamma^2(k)\mathbf{E}_f^{-T}(k+1)\mathbf{E}_f^T(k)\mathbf{E}_f(k)\mathbf{E}_f^{-1}(k+1)$.

Finally, after pre-multiplying and post-multiplying (6.73) by $\mathbf{p}^{\mathrm{T}}(k+1)$ and $\mathbf{p}(k+1)$, respectively, and dividing the result by $\mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1)$, we obtain

$$\begin{aligned} \gamma^2(k) &= \mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1) + \frac{\mathbf{p}^{\mathrm{T}}(k+1)\Psi\mathbf{p}(k+1)}{\mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1)} \\ &= \mathbf{p}^{\mathrm{T}}(k+1)\mathbf{p}(k+1) + *^2. \end{aligned} \tag{6.74}$$

The expression in (6.74) can be regarded as a Cholesky product. Hence, it can be factored as

$$\begin{bmatrix} \gamma(k) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q} \begin{bmatrix} * \\ \mathbf{p}(k+1) \end{bmatrix}, \tag{6.75}$$

where $\mathbf{Q}$ is an orthogonal matrix.

If we recall our starting point in (6.54), we can see that $\mathbf{Q}$ is related to $\overline{\mathbf{Q}}_f(k+1)$. Moreover, from the knowledge of the internal structure of $\overline{\mathbf{Q}}_f(k+1)$, we can conclude that $\mathbf{Q} = \overline{\mathbf{Q}}_f^{\mathrm{T}}(k+1)$ satisfies (6.75) leading to (6.70). Vector $\mathbf{p}(k+1)$ can be easily obtained from (6.70) because $\gamma(k)$ and $\overline{\mathbf{Q}}_f(k+1)$ are known quantities. The reader can use similar arguments in order to prove (6.63).

The rotation angles in matrix $\mathbf{Q}_\theta(k)$ are obtained using

$$\mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma(k+1) \\ \mathbf{f}_P(k+1) \end{bmatrix}, \tag{6.76}$$

for the *a posteriori* case, and

$$\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_P(k+1) \end{bmatrix}, \tag{6.77}$$

for the *a priori* case.

Finally, the joint process estimation is performed as

$$\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_\theta(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}, \tag{6.78}$$

and the *a priori error* is given by [5, 11]

$$e(k+1) = e_{q1}(k+1)/\gamma(k+1). \tag{6.79}$$

The *a posteriori* and *a priori* block-MC-FQRD-RLS algorithms are summarized in Tables 6.4 and 6.5, respectively. In these tables, the common equations are in gray in order to highlight the difference between both algorithms.

6.4.3 Alternative implementations

Similar to their sequential-channel processing counterparts, alternative implementations for the block-channel algorithms are available when the M channels are of equal order, i.e., $N_i = N$ and $P = MN$. In this particular case, the last M elements of vectors $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ are known prior to their updating through Equations (6.58) and (6.59), respectively [6].

Table 6.4 Equations of the *a posteriori* block-MCFQRD based on the update of backward prediction errors [5–7].

```
MCFQRD_POS_B
For each $k$, do
{ 1. Obtaining $\mathbf{D}_{fq2}(k+1)$ and $\mathbf{e}_{fq1}(k+1)$
    $\begin{bmatrix}\mathbf{e}^{\mathrm{T}}_{fq1}(k+1)\\ \mathbf{D}_{fq2}(k+1)\end{bmatrix} = \mathbf{Q}_{\theta}(k)\begin{bmatrix}\mathbf{x}^{\mathrm{T}}_{k+1}\\ \lambda^{1/2}\mathbf{D}_{fq2}(k)\end{bmatrix}$    (6.53)
  2. Obtaining $\mathbf{E}_f(k+1)$ and $\overline{\mathbf{Q}}_f(k+1)$
    $\begin{bmatrix}\mathbf{0}^{\mathrm{T}}\\ \mathbf{E}_f(k+1)\end{bmatrix} = \overline{\mathbf{Q}}_f(k+1)\begin{bmatrix}\mathbf{e}^{\mathrm{T}}_{fq1}(k+1)\\ \lambda^{1/2}\mathbf{E}_f(k)\end{bmatrix}$    (6.54)
  3. Obtaining $\mathbf{p}(k+1)$
    $\begin{bmatrix}*\\ \mathbf{p}(k+1)\end{bmatrix} = \overline{\mathbf{Q}}_f(k+1)\begin{bmatrix}\gamma(k)\\ \mathbf{0}\end{bmatrix}$    implements (6.67)
  4. Obtaining $\mathbf{Q}'_{\theta f}(k+1)$
    $\begin{bmatrix}\mathbf{0}\\ *\\ \mathbf{E}^{0}_{f}(k+1)\end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix}\mathbf{D}_{fq2}(k+1)\\ \mathbf{E}_f(k+1)\end{bmatrix}$    (6.57)
  5. Obtaining $\mathbf{f}_P(k+1)$
    $\mathbf{f}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k+1)\begin{bmatrix}\mathbf{f}_P(k)\\ \mathbf{p}(k+1)\end{bmatrix}$    (6.66)
  6. Obtaining $\mathbf{Q}_{\theta}(k+1)$ and $\gamma(k+1)$
    $\mathbf{Q}_{\theta}(k+1)\begin{bmatrix}1\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\gamma(k+1)\\ \mathbf{f}_P(k+1)\end{bmatrix}$    (6.76)
  7. Joint estimation
    $\begin{bmatrix}e_{q1}(k+1)\\ \mathbf{d}_{q2}(k+1)\end{bmatrix} = \mathbf{Q}_{\theta}(k+1)\begin{bmatrix}d(k+1)\\ \lambda^{1/2}\mathbf{d}_{q2}(k)\end{bmatrix}$    (6.78)
  8. Obtaining the a priori error
    $e(k+1) = e_{q1}(k+1)/\gamma(k+1)$    (6.79)
}
```

Assuming $\overline{\mathbf{P}} = \mathbf{P} = \mathbf{I}$, after the multiplication by $\mathbf{Q}'_{\theta f}(k+1)$ is carried out in (6.55), matrix $\mathbf{U}_{P+M}(k+1)$ can be partitioned as

$$\mathbf{U}_{P+M}(k+1) = \begin{bmatrix}\mathbf{0} & \mathbf{B}\\ \mathbf{E}^{0}_{f}(k+1) & \mathbf{C}\end{bmatrix} \tag{6.80}$$

and by taking its inverse, yields

$$[\mathbf{U}_{P+M}(k+1)]^{-1} = \begin{bmatrix}-\left[\mathbf{E}^{0}_{f}(k+1)\right]^{-1}\mathbf{C}\mathbf{B}^{-1} & \left[\mathbf{E}^{0}_{f}(k+1)\right]^{-1}\\ \mathbf{B}^{-1} & \mathbf{0}\end{bmatrix}. \tag{6.81}$$

If we now use (6.81) together with (6.58), (6.59), and (6.10), vectors $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ can be written, respectively, as

$$\mathbf{a}_{P+M}(k+1) = \begin{bmatrix}*\\ \lambda^{-1/2}\left[\mathbf{E}^{0}_{f}(k)\right]^{-T}\mathbf{x}_{k+1}\end{bmatrix}, \text{ and} \tag{6.82}$$

Table 6.5 Equations of the *a priori* block-MCFQRD based on the update of backward prediction errors [6, 7, 11].

```
MCFQRD_PRI_B
For each $k$, do
{  1. Obtaining $\mathbf{D}_{fq2}(k+1)$ and $\mathbf{e}_{fq1}(k+1)$
      $\begin{bmatrix} \mathbf{e}^{\mathrm{T}}_{fq1}(k+1) \\ \mathbf{D}_{fq2}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}_{k+1} \\ \lambda^{1/2}\mathbf{D}_{fq2}(k) \end{bmatrix}$        (6.53)
   2. Obtaining $\mathbf{E}_f(k+1)$ and $\overline{\mathbf{Q}}_f(k+1)$
      $\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{E}_f(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} \mathbf{e}^{\mathrm{T}}_{fq1}(k+1) \\ \lambda^{1/2}\mathbf{E}_f(k) \end{bmatrix}$        (6.54)
   3. Obtaining $\mathbf{r}(k+1)$
      $\begin{bmatrix} * \\ \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f(k+1) \begin{bmatrix} 1/\gamma(k) \\ -\mathbf{r}(k+1) \end{bmatrix}$        implements (6.61)
   4. Obtaining $\mathbf{a}_P(k+1)$
      $\mathbf{a}_{P+M}(k+1) = \overline{\mathbf{P}}\mathbf{Q}'_{\theta f}(k) \begin{bmatrix} \mathbf{a}_P(k) \\ \mathbf{r}(k+1) \end{bmatrix}$        (6.60)
   5. Obtaining $\mathbf{Q}'_{\theta f}(k+1)$
      $\begin{bmatrix} \mathbf{0} \\ * \\ \mathbf{E}^0_f(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}(k+1) \\ \mathbf{E}_f(k+1) \end{bmatrix}$        (6.57)
   6. Obtaining $\mathbf{Q}_{\theta}(k+1)$ and $\gamma(k+1)$
      $\begin{bmatrix} 1/\gamma(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_{\theta}(k+1) \begin{bmatrix} 1 \\ -\mathbf{a}_P(k+1) \end{bmatrix}$        (6.77)
   7. Joint estimation
      $\begin{bmatrix} e_{q1}(k+1) \\ \mathbf{d}_{q2}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}(k+1) \begin{bmatrix} d(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}(k) \end{bmatrix}$        (6.78)
   8. Obtaining the a priori error
      $e(k+1) = e_{q1}(k+1)/\gamma(k+1)$        (6.79)
}
```

$$\mathbf{f}_{P+M}(k+1) = \begin{bmatrix} * \\ \left[\mathbf{E}^0_f(k+1)\right]^{-T} \mathbf{x}_{k+1} \end{bmatrix}. \tag{6.83}$$

From (6.82) and (6.83), we can see that the last M elements of $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ are known quantities. The alternative implementations of these algorithms arise when the recursive updating of these vectors is performed based on this *a priori* knowledge.

6.5 Order-Recursive MC-FQRD-RLS Algorithms

Block multichannel algorithms are more suitable for order-recursive implementations and parallel processing as compared to their sequential-channel counterparts. Therefore, only the formers shall be addressed in this section. Sequential-channel processing algorithms can also be implemented order-recursively up to a certain degree [9, 11], however, adding order-recursiveness to the already existing channel recursive nature, leads to more complicated structures.

For sake of simplicity, the special case of all M channels having equal orders, i.e., $N_i = N$, is considered. We shall start by revisiting the definitions of Cholesky factor of the information matrix, $\mathbf{U}_{P+M}(k+1)$, given by Equations (6.47) and (6.55), obtained from solving the backward and the forward prediction problems, respectively, and reproduced below assuming channels of equal orders.

$$\mathbf{U}_{P+M}(k+1) = \mathbf{U}_{N+1}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{E}_b^N(k+1) \\ \mathbf{U}_N(k+1) & \mathbf{D}_{bq2}^N(k+1) \end{bmatrix} \tag{6.84}$$

$$= \mathbf{Q}'_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}^N(k+1) & \mathbf{U}_N(k) \\ \mathbf{E}_f^N(k+1) & \mathbf{0} \end{bmatrix} \tag{6.85}$$

The superscript N added to variables $\mathbf{E}_b(k+1)$, $\mathbf{D}_{bq2}$, $\mathbf{D}_{fq2}(k+1)$, and $\mathbf{E}_f(k+1)$ emphasizes that these quantities are related to the Nth-order prediction problem.[5] The last two equations can be written in a generalized form as

$$\mathbf{U}_{j+1}(k+1) = \begin{bmatrix} \mathbf{0} & \mathbf{E}_b^{(j)}(k+1) \\ \mathbf{U}_j(k+1) & \mathbf{D}_{bq2}^{(j)}(k+1) \end{bmatrix} \tag{6.86}$$

$$= \mathbf{Q}'^{(j)}_{\theta f}(k+1) \begin{bmatrix} \mathbf{D}_{fq2}^{(j)}(k+1) & \mathbf{U}_j(k) \\ \mathbf{E}_f^{(j)}(k+1) & \mathbf{0} \end{bmatrix}, \tag{6.87}$$

for $j = 0, 1, \ldots, N$. This property is the key to derive the order-recursive versions of the algorithms. Indeed, the information provided by (6.86) and (6.87) justifies the generalization of Equations (6.65) and (6.69) from Section 6.4.2, respectively, as

$$\mathbf{a}^{(j)}(k+1) = \lambda^{-1/2}[\mathbf{E}_b^{(j)}(k)]^{-T}\mathbf{e}'^{(j)}_b(k+1) \tag{6.88}$$

and

$$\mathbf{f}^{(j)}(k+1) = [\mathbf{E}_b^{(j)}(k+1)]^{-T}\mathbf{e}_b^{(j)}(k+1) \tag{6.89}$$

where $\mathbf{e}'^{(j)}_b(k+1)$ and $\mathbf{e}_b^{(j)}(k+1)$ are, respectively, the jth-order *a priori* and *a posteriori* backward error vectors, for $j = 0, 1, \ldots, N$. Therefore, vectors $\mathbf{a}_{P+M}(k+1)$ and $\mathbf{f}_{P+M}(k+1)$ can be regarded as a nesting of $N+1$ subvectors of size $M \times 1$, i.e.,

$$\mathbf{a}_{N+1}(k+1) = \begin{bmatrix} \mathbf{a}^{(N)}(k+1) \\ \mathbf{a}^{(N-1)}(k+1) \\ \vdots \\ \mathbf{a}^{(0)}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{(N)}(k+1) \\ \mathbf{a}_N(k+1) \end{bmatrix} \tag{6.90}$$

[5] Note that the subscripts $_{P+M}$ and $_{N+1}$ are interchangeable and $_{N+1}$ is used whenever the order of the prediction problems needs to be highlighted, whereas $_{P+M}$ emphasizes vectors or matrices dimensions.

and

$$\mathbf{f}_{N+1}(k+1) = \begin{bmatrix} \mathbf{f}^{(N)}(k+1) \\ \mathbf{f}^{(N-1)}(k+1) \\ \vdots \\ \mathbf{f}^{(0)}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{(N)}(k+1) \\ \mathbf{f}_N(k+1) \end{bmatrix}. \tag{6.91}$$

Recalling that $\mathbf{Q}'_{\theta f}(k)$ and $\mathbf{Q}'_{\theta f}(k+1)$ are used to update $\mathbf{a}_{N+1}(k)$ and $\mathbf{f}_{N+1}(k)$, respectively, we can finally rewrite Equations (6.60) and (6.66) into an order-recursive form, i.e., for $j = 1, 2, \ldots, N$, as

$$\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_j(k+1) \end{bmatrix}, \tag{6.92}$$

where $\mathbf{r}_j(k+1)$ is the jth-order normalized *a priori* forward error vector, and

$$\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_j(k+1) \end{bmatrix} \tag{6.93}$$

where $\mathbf{p}_j(k+1)$ is the jth-order normalized *a posteriori* forward error vector.

Equation (6.93) implements the order-recursive counterpart of step 5 (Table 6.4) of the *a posteriori* version of the MC-FQRD-RLS algorithm, whereas (6.92) stands for corresponding order-recursive implementation of step 4 (Table 6.5) of the *a priori* version.

Now we shall see how to compute the set of Givens rotations in $\mathbf{Q}'_{\theta f}{}^{(j)}(k+1)$. Specifically, we shall find a way to carry out steps 4 and 5 of algorithms in Tables 6.4 and 6.5, respectively, in an order-recursive manner. We begin with noting that previous arguments support the partitioning of matrix $\mathbf{D}_{fq2}(k+1)$ into N $M \times M$-blocks as follows:

$$\mathbf{D}_{fq2}(k+1) = \begin{bmatrix} \mathbf{D}^{(1)}_{fq2}(k+1) \\ \vdots \\ \mathbf{D}^{(N)}_{fq2}(k+1) \end{bmatrix}. \tag{6.94}$$

Now, recalling (6.87), we realize that (6.57) can be rewritten as

$$\begin{bmatrix} \mathbf{0}_{M(N-j+1)\times M} \\ \mathbf{0}_{M(j-1)\times M} \\ \mathbf{E}^{(j-1)}_{f}(k+1) \end{bmatrix} = \mathbf{Q}'_{\theta f}{}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(j-1)\times M} \\ \mathbf{D}^{(j)}_{fq2}(k+1) \\ \mathbf{0}_{M(N-j)\times M} \\ \mathbf{E}^{(j)}_{f}(k+1) \end{bmatrix} \tag{6.95}$$

for $j = 1, 2, \ldots, N$. Moreover, the inherent order-recursiveness of previous equation leads us to conclude that matrix $\mathbf{Q}'_{\theta f}(k+1)$ in (6.57) can be regarded as a product of the form:

$$\mathbf{Q}'_{\theta f}(k+1) = {\mathbf{Q}'_{\theta f}}^{(N)}(k+1){\mathbf{Q}'_{\theta f}}^{(N-1)}(k+1) \cdots {\mathbf{Q}'_{\theta f}}^{(1)}(k+1), \tag{6.96}$$

where each ${\mathbf{Q}'_{\theta f}}^{(j)}(k+1)$, for $j = 1, 2, \ldots, N$, is a product itself of M^2 elementary Givens rotation matrices.

As for step 6 (Tables 6.4 and 6.5), it is straightforward to see that the rotation angles $\mathbf{Q}_{\theta}^{(j)}(k+1)$ are now obtained through

Table 6.6 Algorithm number **1** of Table 6.1 [4].

Lattice block-MCFQRD_POS_B

```
Initializations:
$\mathbf{f}_P(0) = 0;\quad \mathbf{D}_{fq2}(0) = 0;\quad \gamma_0(0) = 1;\quad \mathbf{d}_{q2}(0) = 0;\quad \mathbf{E}_f^{j}(0) = \mu\mathbf{I},$
$\mu = small\ \ number$,   all $cosines = 1$,   and all $sines = 0$;
For each $k$, do
{ $\tilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) = \mathbf{x}_{k+1}^{\mathrm{T}}$;
   Obtaining $\mathbf{E}_f^{(0)}(k+1)$ and $\mathbf{p}_0(k+1)$:
   $\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(0)}(k+1) & \mathbf{p}_0(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f^{(0)}(k+1) \begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) & \gamma_0(k) \\ \lambda^{1/2}\mathbf{E}_f^{(0)}(k) & \mathbf{0} \end{bmatrix};$
   $\mathbf{f}^{(N+1)}(k+1) = \mathbf{p}_0(k+1);\quad \gamma_0(k+1) = 1;$
   $\mathbf{e}_{q1}(k+1) = d(k+1);$
   for $j = 1:N$
   { 1. Obtaining $\mathbf{D}_{fq2}^{(j)}(k+1)$ and $\mathbf{e}_{fq1}^{(j)}(k+1)$:
      $\begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}^{(j)}(k) \begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{(j-1)\,\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{D}_{fq2}^{(j)}(k) \end{bmatrix};$
     2. Obtaining $\mathbf{E}_f^{(j)}(k+1)$ and $\mathbf{p}_j(k+1)$:
      $\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(j)}(k+1) & \mathbf{p}_j(k+1) \end{bmatrix} = \overline{\mathbf{Q}}_f^{(j)}(k+1) \begin{bmatrix} \tilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) & \gamma_j(k) \\ \lambda^{1/2}\mathbf{E}_f^{(j)}(k) & \mathbf{0} \end{bmatrix};$
     3. Obtaining ${\mathbf{Q}'_{\theta f}}^{(j)}(k+1)$:
      $\begin{bmatrix} \mathbf{0}_{M(N-j+1)\times M} \\ \mathbf{0}_{M(j-1)\times M} \\ \mathbf{E}_f^{(j-1)}(k+1) \end{bmatrix} = {\mathbf{Q}'_{\theta f}}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(j-1)\times M} \\ \mathbf{D}_{fq2}^{(j)}(k+1) \\ \mathbf{0}_{M(N-j)\times M} \\ \mathbf{E}_f^{(j)}(k+1) \end{bmatrix};$
     4. Obtaining $\mathbf{f}^{(j)}(k+1)$:
      $\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_{j-1}(k+1) \end{bmatrix} = {\mathbf{Q}'_{\theta f}}^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{f}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{p}_j(k+1) \end{bmatrix};$
     5. Obtaining $\mathbf{Q}_{\theta}^{(j)}(k+1)$ and $\gamma_j(k+1)$:
      $\mathbf{Q}_{\theta}^{(j)}(k+1) \begin{bmatrix} \gamma_{j-1}(k+1) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_j(k+1) \\ \mathbf{f}^{(j-1)}(k+1) \end{bmatrix};$
     6. Joint estimation:
      $\begin{bmatrix} \mathbf{e}_{q1}^{(j)}(k+1) \\ \mathbf{d}_{q2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta}^{(j)}(k+1) \begin{bmatrix} \mathbf{e}_{q1}^{(j-1)}(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}^{(j)}(k) \end{bmatrix};$
     7. Obtaining the a priori error:
      $e_j(k+1) = e_{q1}^{(j)}(k+1)/\gamma_j(k+1);$
   } % for $j$
} % for $k$
```

$$\begin{bmatrix} 1/\gamma_j(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} 1/\gamma_{j-1}(k+1) \\ -\mathbf{a}^{(j-1)}(k+1) \end{bmatrix} \tag{6.97}$$

for the *a priori* algorithm, and

$$\mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} \gamma_{j-1}(k+1) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \gamma_j(k+1) \\ \mathbf{f}^{(j-1)}(k+1) \end{bmatrix} \tag{6.98}$$

for the *a posteriori* case. The joint estimation (step 7) is performed according to

$$\begin{bmatrix} \mathbf{e}_{q1}^{(j)}(k+1) \\ \mathbf{d}_{q2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} \mathbf{e}_{q1}^{(j-1)}(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}^{(j)}(k) \end{bmatrix}. \tag{6.99}$$

Table 6.7 Algorithm number **9** of Table 6.1 [11].

```
Lattice block-MCFQR_PRI_B
Initializations:
$\mathbf{a}_P(0) = 0$;  $\mathbf{D}_{fq2}(0) = 0$;  $\gamma_0(0) = 1$;  $\mathbf{d}_{q2}(0) = 0$;  $\mathbf{E}_f^j(0) = \mu\mathbf{I}$,
$\mu = small\ \ number$,  all $cosines = 1$,  and all $sines = 0$;
For each $k$, do
{  $\widetilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) = \mathbf{x}_{k+1}^{\mathrm{T}}$;
   Obtaining $\mathbf{E}_f^{(0)}(k+1)$ and $\mathbf{r}_0(k+1)$:
   $\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(0)}(k+1) & \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f^{(0)}(k+1) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(0)\,T}(k+1) & 1/\gamma_0(k) \\ \lambda^{1/2}\mathbf{E}_f^{(0)}(k) & -\mathbf{r}_0(k+1) \end{bmatrix}$;
   $\mathbf{a}^{(0)}(k+1) = \mathbf{r}_0(k+1)$;  $\gamma_0(k+1) = 1$;
   $\mathbf{e}_{q1}(k+1) = d(k+1)$;
   for $j = 1 : N$
   {  1. Obtaining $\mathbf{D}_{fq2}^{(j)}(k+1)$ and $\mathbf{e}_{fq1}^{(j)}(k+1)$:
        $\begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) \\ \mathbf{D}_{fq2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j-1)\,\mathrm{T}}(k+1) \\ \lambda^{1/2}\mathbf{D}_{fq2}^{(j)}(k) \end{bmatrix}$;
      2. Obtaining $\mathbf{E}_f^{(j)}(k+1)$ and $\mathbf{p}_j(k+1)$:
        $\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & * \\ \mathbf{E}_f^{(j)}(k+1) & \mathbf{0} \end{bmatrix} = \overline{\mathbf{Q}}_f^{(j)}(k+1) \begin{bmatrix} \widetilde{\mathbf{e}}_{fq1}^{(j)\,\mathrm{T}}(k+1) & 1/\gamma_j(k) \\ \lambda^{1/2}\mathbf{E}_f^{(j)}(k) & -\mathbf{r}_j(k+1) \end{bmatrix}$;
      3. Obtaining $\mathbf{a}^{(j)}(k+1)$:
        $\begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j)}(k+1) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_{j-1}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta f}'^{(j)}(k) \begin{bmatrix} \mathbf{0}_{M(N-j)} \\ \mathbf{a}^{(j-1)}(k) \\ \mathbf{0}_{M(j-1)} \\ \mathbf{r}_j(k+1) \end{bmatrix}$;
      4. Obtaining $\mathbf{Q}_{\theta f}'^{(j)}(k+1)$:
        $\begin{bmatrix} \mathbf{0}_{M(N-j+1)\times M} \\ \mathbf{0}_{M(j-1)\times M} \\ \mathbf{E}_f^{(j-1)}(k+1) \end{bmatrix} = \mathbf{Q}_{\theta f}'^{(j)}(k+1) \begin{bmatrix} \mathbf{0}_{M(j-1)\times M} \\ \mathbf{D}_{fq2}^{(j)}(k+1) \\ \mathbf{0}_{M(N-j)\times M} \\ \mathbf{E}_f^{(j)}(k+1) \end{bmatrix}$;
      5. Obtaining $\mathbf{Q}_\theta^{(j)}(k+1)$ and $\gamma_j(k+1)$:
        $\begin{bmatrix} 1/\gamma_j(k+1) \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} 1/\gamma_{j-1}(k+1) \\ -\mathbf{a}^{(j-1)}(k+1) \end{bmatrix}$;
      6. Joint estimation:
        $\begin{bmatrix} \mathbf{e}_{q1}^{(j)}(k+1) \\ \mathbf{d}_{q2}^{(j)}(k+1) \end{bmatrix} = \mathbf{Q}_\theta^{(j)}(k+1) \begin{bmatrix} \mathbf{e}_{q1}^{(j-1)}(k+1) \\ \lambda^{1/2}\mathbf{d}_{q2}^{(j)}(k) \end{bmatrix}$;
      7. Obtaining the a priori error:
        $e_j(k+1) = e_{q1}^{(j)}(k+1)/\gamma_j(k+1)$;
   } % for $j$
} % for $k$
```

Finally, in order to adjust the equations of steps 1–3 of the algorithms in Tables 6.4 and 6.5 to this formulation, it suffices to observe that they can be split up into blocks that will be processed in an order-recursive way. The resulting lattice (or order-recursive) versions of the block-type MC-FQRD-RLS algorithms based on *a posteriori* and *a priori* backward prediction errors are summarized in Tables 6.6 and 6.7, respectively.

6.6 Application Example and Computational Complexity Issues

In this section, the effectiveness of the algorithms addressed in this work is illustrated in a non-linear filtering problem. In addition, we provide a brief discussion on computational complexity issues.

6.6.1 Application example

The computer experiment consists of a non-linear system identification. The plant is a simple truncated second-order Volterra system [2] which can be summarized as

$$d(k) = \sum_{n_1=0}^{L-1} w_{n_1}(k)x(k-n_1) + \sum_{n_1=0}^{L-1}\sum_{n_2=0}^{L-1} w_{n_1,n_2}(k)x(k-n_1)x(k-n_2) + \rho(k). \quad (6.100)$$

Equation (6.100) can be easily reformulated as a multichannel problem with $M = L+1$ channels, where the most recent sample of the ith channel is

$$x_i(k) = \begin{cases} x(k), & i = 1, \\ x(k)x(k-i+2), & i = 2,\ldots,L+1, \end{cases}$$

and the ith channel order is

$$N_i = \begin{cases} L, & i = 1,2, \\ L-i+2, & i = 3,\ldots,L+1. \end{cases}$$

In the experiment, we have used $L = 4$ and the resulting multichannel system is depicted in Figure 6.5. The forgetting factor was set to $\lambda = 0.98$, and the power of the observation noise $\rho(k)$ was chosen such that the signal-to-noise-ratio (SNR) is 60 dB. The learning curves of the multichannel FQRD-RLS algorithms are compared to the normalized least-mean-squares[6] (NLMS) [15–17] and the result is plotted in Figure 6.6 for an average of 100 independent runs. The trade-

[6] The updating of the coefficient vector for the NLMS algorithm was performed according to

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \frac{\mu}{\sigma + \|\mathbf{x}(k)\|^2}\mathbf{x}(x)e^*(k),$$

where $\mathbf{x}(k)$ is the input signal vector, σ is a small constant, and parameter μ was set equal to 1.

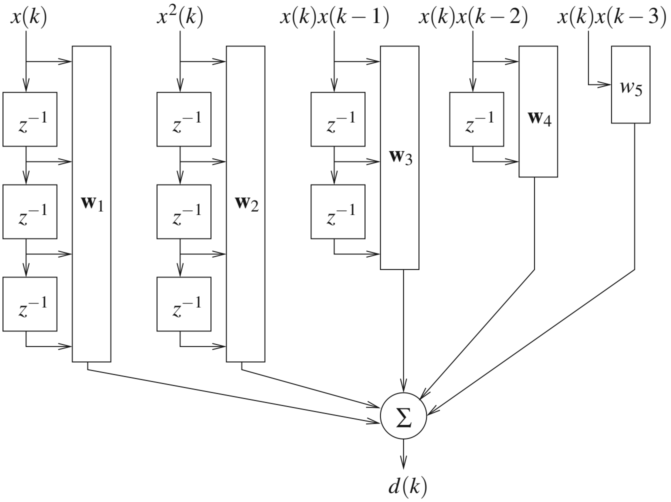


Fig. 6.5 Multichannel set-up for a truncated second order Volterra system, $L = 4$.

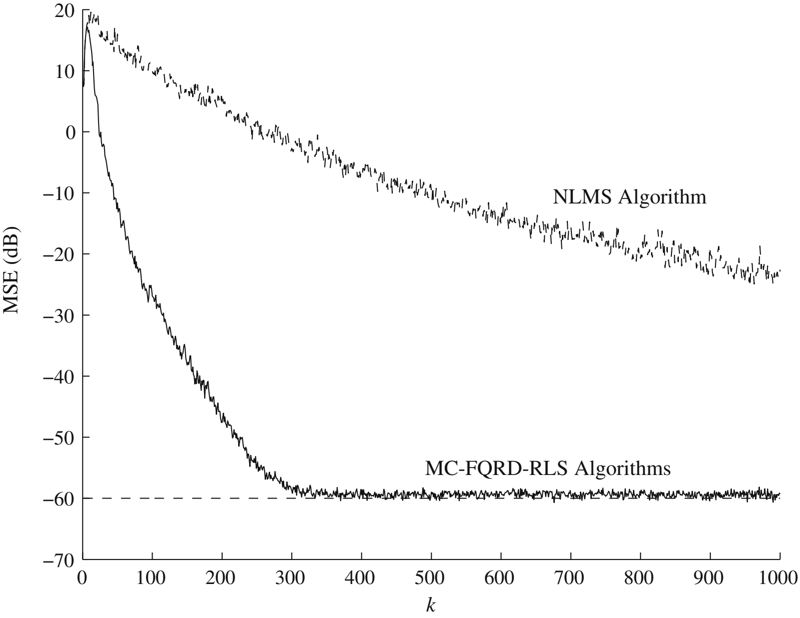


Fig. 6.6 Learning curves of the non-linear system identification experiment.

off between computational complexity and speed of convergence is also illustrated in that figure.

6.6.2 Computational complexity issues

The computational complexity of the MC-FQRD-RLS algorithms, in terms of multiplication, divisions, and square roots per input sample, is summarized in Table 6.8, according to the classification introduced earlier in Table 6.1. Generally speaking, when the same structure and approach are considered, algorithms based on the *a posteriori* backward prediction errors updating have lower computational burden when compared to their *a priori* counterparts.

The suitability of the lattice structure for real-time implementations comes at the cost of a slight increase in the computational burden of the algorithms. On the other hand, sequential-type algorithms $\mathscr{O}[MP]$ computational complexity is lower by one order as compared to the $\mathscr{O}[M^2P]$ block-type multichannel algorithms computational complexity. It is also evident that the MC-FQRD-RLS algorithms outperform the conventional QRD-RLS and inverse QRD-RLS algorithms, $\mathscr{O}[P^2]$, in terms of computational costs, while maintaining the good numerical stability features. The computational advantage of block-type MC-FQRD-RLS algorithms is increasing for larger number of coefficients per channel, N, ($P = MN$ for channels of equal orders).

Table 6.8 Computational complexity of MC-FQRD-RLS algorithms, according to Table 6.1.

Algorithm	Multiplications	Divisions	Squared roots
Algs. **2, 4** [5–7], summarized in Table 6.4	$4NM^2+11NM+$ $5M^2+6M+7N-$ $(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$	$2NM+2M+N-$ $2M\sum_{i=1}^{M}(p_i-i)$	$2NM+M+N-$ $2M\sum_{i=1}^{M}(p_i-i)$
Algs. **10, 12** [6, 7, 11], summarized in Table 6.5	$4NM^2+11NM+$ $5M^2+6M+9N-$ $(4M^2+6M)\sum_{i=1}^{M}(p_i-i)$	$2NM+3M+2N-$ $2M\sum_{i=1}^{M}(p_i-i)+2$	$2NM+M+N-$ $2M\sum_{i=1}^{M}(p_i-i)$
Alg. **1** [4], summarized in Table 6.6	$4M^3N+17M^2N+$ $12MN+5M^2+5M$	$2M^2N+3MN+2M$	$M^2N+2MN+M$
Alg. **9** [11] summarized in Table 6.7	$4M^3N+17M^2N+$ $14MN+5M^2+6M$	$2M^2N+5MN+3M$	$M^2N+2MN+M$
Algs. **6, 8** [10], summarized in Table 6.2	$14NM+13M+$ $5N-9\sum_{i=1}^{M}p_i$	$3NM+4M-$ $3\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$
Algs. **14, 16** [11] summarized in Table 6.3	$15NM+14M+$ $5N-10\sum_{i=1}^{M}p_i$	$4NM+5M-$ $4\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$
Algs. **5, 7** [9]	$14NM+13M+$ $5N-9\sum_{i=1}^{M}p_i$	$4NM+5M-$ $4\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$
Algs. **13, 15** [11]	$15NM+14M+$ $5N-10\sum_{i=1}^{M}p_i$	$5NM+6M-$ $5\sum_{i=1}^{M}p_i$	$2NM+3M-$ $2\sum_{i=1}^{M}p_i$

6.7 Conclusion

The MC-FQRD-RLS algorithms exploit the time-shift structure in each channel to reduce the computational complexity of the conventional QRD-RLS algorithm which is of order $\mathscr{O}[P^2]$, P being the number of coefficients. This chapter introduced various MC-FQRD-RLS algorithms based on the updating of the backward prediction error vector. Channels are allowed to have arbitrary orders, which enables us to deal with more general multichannel systems, e.g., Volterra systems. The algorithms presented in this chapter were derived using two distinct approaches: (1) *sequential approach* where channels are processed individually in a sequential manner, and (2) *block approach* that jointly processes all channels. Considering the case of M channels, the computational complexities associated with block and sequential algorithms are $\mathscr{O}[MP]$ and $\mathscr{O}[M^2P]$, respectively. That is, taking a sequential approach will render the lowest complexity. The main advantages of the block algorithms are that they favor parallel processing implementations and can easily be turned into an order-recursive form. To clarify the differences among the many versions available in the literature, we provided a classification of these algorithms and their associated computational complexities.

Acknowledgements This work was partially funded by the Academy of Finland, Smart and Novel Radios (SMARAD) Center of Excellence and by the Buskerud University College (HIBU), Department of Technology (ATEK), Norway.

References

1. N. Kalouptsidis and S. Theodoridis, Adaptive Systems Identification and Signal Processing Algorithms. Prentice-Hall, Upper Saddle River, NJ, USA (1993)
2. V. J. Mathews and G. L. Sicuranza, Polynomial Signal Processing. Wiley-Intercience: John Wiley & Sons, New York, NY, USA (2000)
3. A. L. L. Ramos, J. A. Apolinário Jr., and S. Werner, Multichannel fast QRD-LS adaptive filtering: Block-channel and sequential-channel algorithms based on updating backward prediction errors. Signal Processing, vol. 87, no. 7, pp. 1782–1798 (July 2007)
4. A. L. L. Ramos and J. A. Apolinário Jr., A lattice version of the multichannel FQRD algorithm based on *a posteriori* backward errors. 11th International Conference on Telecommunications, ICT'2004 (LNCS3124), Fortaleza, Brazil, vol. 1, pp. 488–497 (August 2004)
5. M. G. Bellanger and P. A. Regalia, The FLS-QR algorithm for adaptive filtering: the case of multichannel signals. Signal Processing (EURASIP), vol. 22, no. 2, pp. 115–126 (February 1991)
6. C. A. Medina S., J. A. Apolinário Jr., and M. G. Siqueira, A unified framework for multichannel fast QRD-LS adaptive filters based on backward prediction errors. 45th Midwest Symposium on Circuits and Systems, MWSCAS'2002, vol. 3, pp. 668–671, Tulsa, USA (August 2002)
7. A. L. L. Ramos, J. A. Apolinário Jr., and S. Werner, A general approach to the derivation of block multichannel fast QRD-RLS algorithms. European Signal Processing Conference, EUSIPCO'2005, Antalya, Turkey, vol. 1, pp. 1–4 (September 2005)

8. M. A. Syed and V. J. Mathews, QR-decomposition based algorithms for adaptive Volterra filtering. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 40, no. 6, pp. 372–382 (June 1993)
9. A. L. L. Ramos, J. A. Apolinário Jr., and M. G. Siqueira, A new order recursive multiple order multichannel fast QRD algorithm. 38th Midwest Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, vol. 1, pp. 965–969 (November 2004)
10. A. L. L. Ramos and J. A. Apolinário Jr., A new multiple order multichannel fast QRD algorithm and its application to non-linear system identification. XXI Simpósio Brasileiro de Telecomunicações, SBT'2004, Belém, Brazil, vol. 1, pp. 1–4 (Setember 2004)
11. A. A. Rontogiannis and S. Theodoridis, Multichannel fast QRD-LS adaptive filtering: new technique and algorithms. IEEE Transactions on Signal Processing, vol. 46, no. 11, pp. 2862–2876 (November 1998)
12. M. A. Syed, QRD-based fast multichannel adaptive algorithms. International Conference on Acoustics, Speech, and Signal Processing, ICASSSP'91, Toronto, Canada, vol. 3, no. 6, pp. 1837–1840 (April 1991)
13. M. Harteneck, J. G. McWhirter, I. K. Proudler, and R. W. Stewart, Algorithmically engineered fast multichannel adaptive filter based QR-RLS. IEE Proceedings Vision, Image and Signal Processing, vol. 146, no. 1, pp. 7–13 (February 1999)
14. G. H. Golub and C. F. Van Loan, Matrix Computations. 3rd edition The Johns Hopkins University Press, Baltimore, MD, USA (1996)
15. P. S. R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation. 3rd edition Springer, New York, NY, USA (2008)
16. S. Haykin, Adaptive Filter Theory. 4th Edition Prentice-Hall, Englewood-Cliffs, NJ, USA (2002)
17. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley & Sons, Hoboken, NJ, USA (2003)

Chapter 7
Householder-Based RLS Algorithms

Athanasios A. Rontogiannis and Sergios Theodoridis

Abstract This chapter presents recursive least-squares (RLS) algorithms, which are based on numerically robust Householder transformations. Section 7.1 introduces the conventional orthogonal Householder transform and provides its geometrical interpretation. The hyperbolic Householder and the row (orthogonal and hyperbolic) Householder transforms are also briefly described. In Section 7.2, the Householder RLS (HRLS) algorithm is presented, its relation with the other known square-root RLS algorithms is discussed, and two applications, where the HRLS algorithm has been successfully applied, are presented. By considering a block-by-block update approach to the RLS problem, the block exact Householder QRD-RLS algorithm is developed in Section 7.3, which constitutes a generalization of the conventional QRD-RLS algorithm. Section 7.4 presents an inverse QRD-RLS algorithm, which employs block updating via the application of appropriate row orthogonal Householder transformations. Finally, a sliding window block RLS algorithm, which comprises a pair of row Householder transforms, is introduced in Section 7.5.

7.1 Householder Transforms

The Householder transform is an orthogonal matrix transform, named after Alston S. Householder, who has discovered it in the late 1950s [1]. It defines a norm-preserving transformation on a vector as the reflection of the vector with respect to a properly selected hyperplane. More specifically, let **x** and **y** be $M \times 1$ vectors. The projection of **x** onto **y** is then defined as follows:

Athanasios A. Rontogiannis
National Observatory of Athens, Athens – Greece
e-mail: tronto@space.noa.gr

Sergios Theodoridis
University of Athens, Athens – Greece
e-mail: stheodor@di.uoa.gr

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_7, © Springer Science+Business Media, LLC 2009

$$P_{\mathbf{y}}(\mathbf{x}) = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{y}}{||\mathbf{y}||^2}\mathbf{y} \tag{7.1}$$

By denoting with $P_{\mathbf{y}}^{\perp}(\mathbf{x})$ the projection of $\mathbf{x}$ onto the hyperplane, which is perpendicular to $\mathbf{y}$, we may write

$$\mathbf{x} = P_{\mathbf{y}}^{\perp}(\mathbf{x}) + P_{\mathbf{y}}(\mathbf{x}). \tag{7.2}$$

The Householder transform of $\mathbf{x}$ with respect to $\mathbf{y}$ is then given by [2]

$$T_{\mathbf{y}}(\mathbf{x}) = P_{\mathbf{y}}^{\perp}(\mathbf{x}) - P_{\mathbf{y}}(\mathbf{x}) \tag{7.3}$$

or, from (7.1) and (7.2),

$$T_{\mathbf{y}}(\mathbf{x}) = \left(\mathbf{I} - 2\frac{\mathbf{y}\mathbf{y}^{\mathrm{T}}}{||\mathbf{y}||^2}\right)\mathbf{x}. \tag{7.4}$$

The matrix

$$\mathbf{H} = \mathbf{I} - \beta\mathbf{y}\mathbf{y}^{\mathrm{T}} \tag{7.5}$$

where $\beta = 2/||\mathbf{y}||^2$ is easily seen to be orthogonal and symmetric. It is called Householder matrix and consists a rank-1 update to the identity matrix. The Householder transform is illustrated graphically in Figure 7.1. It can be seen from Figure 7.1 that multiplication of $\mathbf{x}$ with the Householder matrix $\mathbf{H}$ is equivalent to reflecting $\mathbf{x}$ with respect to the hyperplane, which is perpendicular to $\mathbf{y}$.

In many signal processing applications, it turns out to be convenient to transform an initial set of data to a sparse one, which is equivalent to the initial dataset in terms of a certain type of invariance. Mobilizing the Householder transformation in (7.5) and a suitable choice of $\mathbf{y}$ in terms of $\mathbf{x}$, it is possible to compress all the energy of $\mathbf{x}$ in just one element of $T_{\mathbf{y}}(\mathbf{x})$. Indeed, it can be easily verified from (7.4) that, by

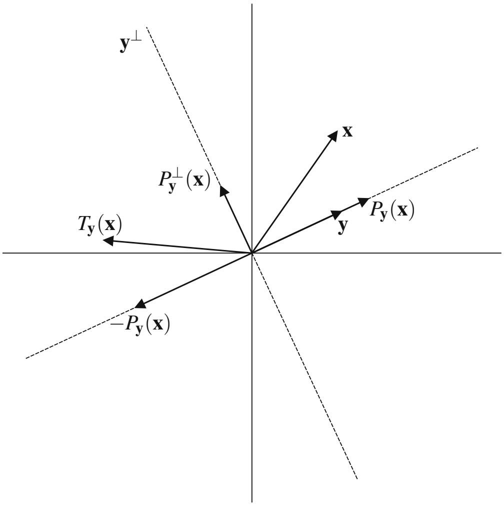

$\mathbf{y}^{\perp}$
$\mathbf{x}$
$P_{\mathbf{y}}^{\perp}(\mathbf{x})$
$P_{\mathbf{y}}(\mathbf{x})$
$\mathbf{y}$
$T_{\mathbf{y}}(\mathbf{x})$
$-P_{\mathbf{y}}(\mathbf{x})$

Fig. 7.1 Geometrical representation of the orthogonal Householder transform.

setting

$$\mathbf{y} = \mathbf{x} + ||\mathbf{x}||\mathbf{a}_i, \tag{7.6}$$

then

$$T_{\mathbf{y}}(\mathbf{x}) = \mathbf{H}\mathbf{x} = -||\mathbf{x}||\mathbf{a}_i, \tag{7.7}$$

where $\mathbf{a}_i$ is the ith column of the $M \times M$ identity matrix.[1] That is, a Householder matrix can be designed so that to annul a block of elements by reflecting a vector onto a coordinate axis. This procedure is illustrated in Figure 7.2.

The Householder transformation is a numerically robust approach to produce sparsity in data. Compared to the Givens rotations approach, which is used to zero one vector element at a time, multiple vector elements are zeroed with the application of one Householder reflection. Thus, an $M \times M$ Householder reflection can be considered equivalent to a succession of $M-1$ Givens rotations, requiring less computations.

Due to its special form, a Householder matrix need not be computed explicitly. For instance, multiplication of $\mathbf{H}$ with an $M \times M$ matrix $\mathbf{A}$ is expressed as

$$\mathbf{H}\mathbf{A} = \mathbf{A} - 2\frac{\mathbf{y}}{||\mathbf{y}||^2}(\mathbf{y}^{\mathrm{T}}\mathbf{A}) \tag{7.8}$$

and thus a matrix-by-matrix multiplication is replaced by matrix-by-vector and vector-by-vector operations.

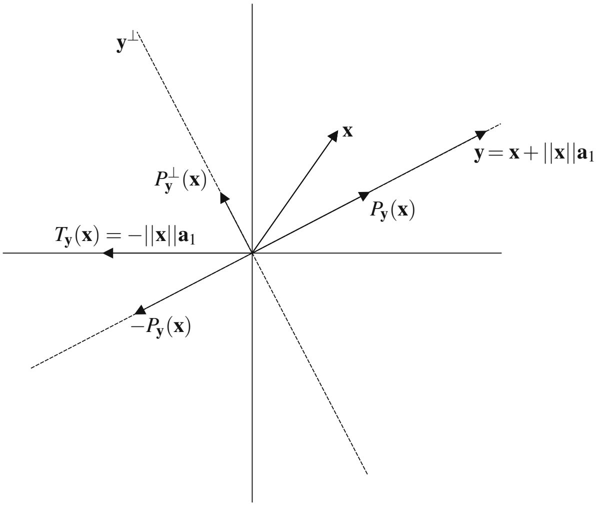

$\mathbf{y}^{\perp}$
$\mathbf{x}$
$\mathbf{y} = \mathbf{x} + ||\mathbf{x}||\mathbf{a}_1$
$P^{\perp}_{\mathbf{y}}(\mathbf{x})$
$P_{\mathbf{y}}(\mathbf{x})$
$T_{\mathbf{y}}(\mathbf{x}) = -||\mathbf{x}||\mathbf{a}_1$
$-P_{\mathbf{y}}(\mathbf{x})$

Fig. 7.2 The Householder transform that zeroes entries of vector $\mathbf{x}$.

[1] One could also choose $\mathbf{y} = \mathbf{x} - ||\mathbf{x}||\mathbf{a}_i$, which results in $\mathbf{H}\mathbf{x} = ||\mathbf{x}||\mathbf{a}_i$.

7.1.1 Hyperbolic Householder transforms

A slightly different matrix that can be used to introduce zeros in a vector or a column of a matrix is the hyperbolic Householder transform [3]. Let $\boldsymbol{\Phi}$ be an $M \times M$ diagonal matrix with entries $+1$ and -1. Then, the hyperbolic norm of a vector $\mathbf{x}$ is defined as follows:

$$\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x} = \sum_{i=1}^{M} \phi_i |x_i|^2 \tag{7.9}$$

where we assumed that $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x} > 0$, ϕ_i is the ith diagonal element of $\boldsymbol{\Phi}$ and x_i the ith element of $\mathbf{x}$.

An $M \times M$ matrix $\mathbf{Q}$ satisfying

$$\mathbf{Q}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{Q} = \boldsymbol{\Phi}, \tag{7.10}$$

is called hypernormal matrix with respect to $\boldsymbol{\Phi}$ and has the property to preserve the hyperbolic norm of a vector, i.e., if $\mathbf{z} = \mathbf{Q}^{\mathrm{T}}\mathbf{x}$ then $\mathbf{z}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{z} = \mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}$.

A matrix $\mathbf{H}$ defined as

$$\mathbf{H} = \boldsymbol{\Phi} - \beta \mathbf{y}\mathbf{y}^{\mathrm{T}}, \tag{7.11}$$

where $\beta = 2/(\mathbf{y}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{y})$, is called hyperbolic Householder transform. It is not difficult to verify that $\mathbf{H}$ is symmetric and hypernormal with respect to $\boldsymbol{\Phi}$. Similarly to the orthogonal Householder transforms, $\mathbf{y}$ can be properly selected so that application of $\mathbf{H}$ in (7.11) annihilates all but one entries of a vector. Indeed, by setting

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{x} + \left(\frac{x_i}{|x_i|}\sqrt{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}}\right)\mathbf{a}_i \tag{7.12}$$

and assuming that $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x} > 0$ and $\phi_i = 1$, then

$$\mathbf{H}\mathbf{x} = -\left(\frac{x_i}{|x_i|}\sqrt{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}}\right)\mathbf{a}_i \tag{7.13}$$

and all hyperbolic energy of $\mathbf{x}$, i.e., $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}$, is compressed to its ith entry.

7.1.2 Row Householder transforms

The Householder transforms discussed so far are designed to introduce zeros in a *column* of a matrix. As shown in [4], Householder matrices (either orthogonal or hyperbolic) can also be constructed to zero one *row* of a matrix. Let $\mathbf{A}$ be an $(M+1) \times M$ matrix expressed as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{C} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix} \tag{7.14}$$

where the $M \times M$ matrix $\mathbf{C}$ is assumed to be invertible. Suppose we wish to eliminate the last row $\mathbf{b}^{\mathrm{T}}$ of $\mathbf{A}$ by applying an $(M+1) \times (M+1)$ orthogonal Householder transformation, as the one given in (7.5), i.e.,

$$\mathbf{H}\begin{bmatrix}\mathbf{C}\\ \mathbf{b}^{\mathrm{T}}\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{C}}\\ \mathbf{0}^{\mathrm{T}}\end{bmatrix}. \tag{7.15}$$

It has been proven in [4] that for an $\mathbf{H}$ to satisfy the last equation, its defining $(M+1) \times 1$ vector $\mathbf{y}$ must be constructed as

$$\mathbf{y} = \frac{1}{||\mathbf{b}||}\begin{bmatrix}\mathbf{C}^{-\mathrm{T}}\mathbf{b}\\ \\ -1-\sqrt{1+(\mathbf{C}^{-\mathrm{T}}\mathbf{b})^{\mathrm{T}}(\mathbf{C}^{-\mathrm{T}}\mathbf{b})}\end{bmatrix}. \tag{7.16}$$

Note that finding $\mathbf{y}$ requires the computation of the inverse of $\mathbf{C}$.

By following a similar analysis, it can be shown [4] that a hyperbolic Householder matrix, defined as in (7.11), can be constructed to eliminate $\mathbf{b}^{\mathrm{T}}$. Its defining vector will be now given by

$$\mathbf{y} = \frac{1}{||\mathbf{b}||}\begin{bmatrix}\mathbf{C}^{-\mathrm{T}}\mathbf{b}\\ \\ -1-\sqrt{1+\phi_{M+1}(\mathbf{C}^{-\mathrm{T}}\mathbf{b})^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}(\mathbf{C}^{-\mathrm{T}}\mathbf{b})}\end{bmatrix}, \tag{7.17}$$

where ϕ_{M+1} is the last diagonal element of the $(M+1) \times (M+1)$ matrix $\boldsymbol{\Phi}$ and $\tilde{\boldsymbol{\Phi}}$ its upper left $M \times M$ diagonal block.

Remarks

In the previous analysis, matrices and vectors with real entries have been assumed. In case of complex entries:

- The analysis of Section 7.1 remains under the condition that the ith element of vector $\mathbf{x}$ is real. The orthogonal Householder transform is now given by

$$\mathbf{H} = \mathbf{I} - \frac{2}{||\mathbf{y}||^2}\mathbf{y}\mathbf{y}^{\mathrm{H}} \tag{7.18}$$

- The hyperbolic Householder transform is expressed as

$$\mathbf{H} = \boldsymbol{\Phi} - \frac{2}{\mathbf{y}^{\mathrm{H}}\boldsymbol{\Phi}\mathbf{y}}\mathbf{y}\mathbf{y}^{\mathrm{H}}, \tag{7.19}$$

  while Equations (7.12) and (7.13) still hold with the hyperbolic energy now defined as $\mathbf{x}^{\mathrm{H}}\boldsymbol{\Phi}\mathbf{x}$.
- The analysis of Section 7.1.2 still holds by replacing simple transposition with conjugate transposition in Equations (7.14), (7.15), (7.16), and (7.17), and constructing Householder matrices from (7.18) and (7.19).

7.2 The Householder RLS (HRLS) Algorithm

In Chapter 3, the conventional and the inverse QR-decomposition recursive least-squares (QRD-RLS) algorithms have been presented. The main characteristic of these algorithms is that the triangular Cholesky factor of the input data correlation matrix (or its inverse) is updated in each iteration based on a sequence of Givens rotations. It is well known that a symmetric positive definite matrix has an infinite number of square-roots. These can be expressed as the product of an arbitrary orthogonal matrix with the respective Cholesky factor. In this section, we present an RLS algorithm, which updates in time an arbitrary square-root of the input data correlation matrix and provides naturally the LS weight vector. It will be shown that such an update is performed by applying a properly constructed data dependent Householder matrix.

Let $\mathbf{S}(k)$ be an arbitrary (not necessarily triangular) $(N+1)\times(N+1)$ square-root factor of the data correlation matrix $\mathbf{R}(k)$ at time k. Then the following relation holds:

$$\mathbf{R}(k) = \mathbf{S}^{\mathrm{T}}(k)\mathbf{S}(k) \tag{7.20}$$

Based on the new data vector $\mathbf{x}(k)$ and the square-root factor of the previous time instant, we define the following vector:[2]

$$\mathbf{u}(k) = \frac{\mathbf{S}^{-\mathrm{T}}(k-1)\mathbf{x}(k)}{\sqrt{\lambda}} \tag{7.21}$$

where λ is the forgetting factor of the exponentially weighted LS cost function. Let now $\mathbf{P}(k)$ be an $(N+2)\times(N+2)$ orthogonal matrix, which performs the following transformation:

$$\mathbf{P}(k)\begin{bmatrix} 1 \\ \mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} -\delta(k) \\ \mathbf{0} \end{bmatrix}, \tag{7.22}$$

where $\delta(k)$ is a positive scalar. According to (7.22), matrix $\mathbf{P}(k)$ zeros $\mathbf{u}(k)$, which coincides with the last $N+1$ elements of the vector on the left-hand-side of the equation. Due to the orthogonality of $\mathbf{P}(k)$ and (7.21), the positive scalar $\delta(k)$ is given by

$$\delta(k) = \sqrt{1+||\mathbf{u}(k)||^2} = \sqrt{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}. \tag{7.23}$$

It is most interesting that the orthogonal matrix $\mathbf{P}(k)$, which satisfies (7.22), also updates in time the inverse square-root factor of $\mathbf{R}(k-1)$. Indeed, by formulating the equation

$$\mathbf{P}(k)\begin{bmatrix} \mathbf{0}^{\mathrm{T}} & 1 \\ \lambda^{-1/2}\mathbf{S}^{-\mathrm{T}}(k-1) & \mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{z}^{\mathrm{T}}(k) & -\delta(k) \\ \mathbf{Y}(k) & \mathbf{0} \end{bmatrix} \tag{7.24}$$

[2] This vector is reminiscent of the so-called Kalman gain vector, which appears in the conventional RLS algorithm [5]. The Kalman gain vector is obtained by replacing $\mathbf{S}^{-\mathrm{T}}(k-1)$ in (7.21) by $\mathbf{R}^{-\mathrm{T}}(k-1)$.

and multiplying each side of (7.24) by its transpose and equating parts, $\mathbf{z}(k)$ and $\mathbf{Y}(k)$ are obtained from the following set of expressions:

$$\mathbf{z}(k) = -\frac{\mathbf{S}^{-1}(k-1)\mathbf{u}(k)}{\sqrt{\lambda}\delta(k)} = -\frac{\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}{\lambda\delta(k)} \tag{7.25}$$

and

$$\mathbf{Y}^{\mathrm{T}}(k)\mathbf{Y}(k) = \lambda^{-1}\mathbf{R}^{-1}(k-1) - \mathbf{z}(k)\mathbf{z}^{\mathrm{T}}(k). \tag{7.26}$$

Note that, according to (7.25), $\mathbf{z}(k)$ is a scaled version of the Kalman gain vector [5], which can be used to update the LS weight vector. Moreover, let us consider the well-known LS input correlation matrix update equation

$$\mathbf{R}(k) = \lambda\mathbf{R}(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k). \tag{7.27}$$

Application of the matrix inversion lemma in (7.27) leads to the expression on the right-hand-side of (7.26). Thus, $\mathbf{R}^{-1}(k) = \mathbf{Y}^{\mathrm{T}}(k)\mathbf{Y}(k)$, verifying that

$$\mathbf{Y}(k) = \mathbf{S}^{-\mathrm{T}}(k). \tag{7.28}$$

Since Equation (7.22) represents an annihilation of a block of elements in a vector, the natural choice for $\mathbf{P}(k)$ is a Householder matrix. According to the analysis of Section 7.1, a Householder matrix $\mathbf{P}(k)$ satisfying (7.22) is expressed as follows:

$$\mathbf{P}(k) = \mathbf{I} - \beta(k)\mathbf{y}(k)\mathbf{y}^{\mathrm{T}}(k), \tag{7.29}$$

where

$$\mathbf{y}(k) = \begin{bmatrix} 1+\delta(k) \\ \mathbf{u}(k) \end{bmatrix}, \tag{7.30}$$

and

$$\beta(k) = \frac{1}{\delta(k)(1+\delta(k))}. \tag{7.31}$$

The Householder RLS (HRLS) algorithm is completed with the formulas required for the computation of the *a priori* LS error and the update of the LS weight vector, respectively, i.e.,

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k), \text{ and} \tag{7.32}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{e(k)}{\delta(k)}\mathbf{z}(k). \tag{7.33}$$

The basic steps of the HRLS algorithm are summarized in Table 7.1, where the structure of the orthogonal matrix $\mathbf{P}(k)$ has not been taken into consideration. However, by exploiting the special form of the employed Householder transformation, $\mathbf{P}(k)$ need not be explicitly computed. More specifically, from (7.28) to (7.31) and (7.24), the following explicit update equation for $\mathbf{S}^{-1}(k-1)$ is obtained:

Table 7.1 The basic steps of the HRLS algorithm.

```
HRLS
Initialize $0 \ll \lambda < 1, \mathbf{w}(0) = \mathbf{0}, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{S}^{-1}(0) = \sqrt{\varepsilon}\mathbf{I}$
for each $k$
{ Computing $\mathbf{u}(k)$:
  $\mathbf{u}(k) = \lambda^{-1/2}\mathbf{S}^{-\mathrm{H}}(k-1)\mathbf{x}(k)$
  Obtaining $\mathbf{P}(k)$ and $\delta(k)$:
  $\mathbf{P}(k)\begin{bmatrix} 1 \\ \mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} -\delta(k) \\ \mathbf{0} \end{bmatrix}$
  Updating $\mathbf{S}^{-1}(k-1)$:
  $\mathbf{P}(k)\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{S}^{-\mathrm{H}}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{z}^{\mathrm{H}}(k) \\ \mathbf{S}^{-\mathrm{H}}(k) \end{bmatrix}$
  Obtaining $e(k)$ and $\mathbf{w}(k)$:
  $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
  $\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{e^*(k)}{\delta(k)}\mathbf{z}(k)$
}
```

$$\mathbf{S}^{-1}(k) = \lambda^{-1/2}\mathbf{S}^{-1}(k-1) + \frac{\mathbf{z}(k)\mathbf{u}^{\mathrm{T}}(k)}{1+\delta(k)}. \tag{7.34}$$

Thus, we are led to an analytical form of the HRLS algorithm, as illustrated in Table 7.2.

The HRLS algorithm was originally introduced in [6] and [7] and later rediscovered independently in [8]. The main advantage of the algorithm is its numerically robust behavior in a finite-precision environment. This is clearly shown in Figure 7.3, where the mean squared error of the RLS and HRLS algorithms is depicted for a numerically unstable situation. More specifically, we have considered a system of order $N = 8$ and an input signal is generated as follows:

$$x(k) = \cos(0.05\pi k) + \sqrt{2}\cos(0.3\pi k) + \eta(k) \tag{7.35}$$

Table 7.2 Analytical form of the HRLS algorithm.

```
HRLS
Initialize $0 \ll \lambda < 1, \mathbf{w}(0) = \mathbf{0}, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{S}^{-1}(0) = \sqrt{\varepsilon}\mathbf{I}$
for each $k$
{ $\mathbf{u}(k) = \lambda^{-1/2}\mathbf{S}^{-\mathrm{H}}(k-1)\mathbf{x}(k)$
  $\delta(k) = \sqrt{1 + ||\mathbf{u}(k)||^2}$
  $\mathbf{z}(k) = -\frac{\lambda^{-1/2}\mathbf{S}^{-1}(k-1)\mathbf{u}(k)}{\delta(k)}$
  $\mathbf{S}^{-1}(k) = \lambda^{-1/2}\mathbf{S}^{-1}(k-1) + \frac{\mathbf{z}(k)\mathbf{u}^{\mathrm{H}}(k)}{1+\delta(k)}$
  $e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
  $\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{e^*(k)}{\delta(k)}\mathbf{z}(k)$
}
```

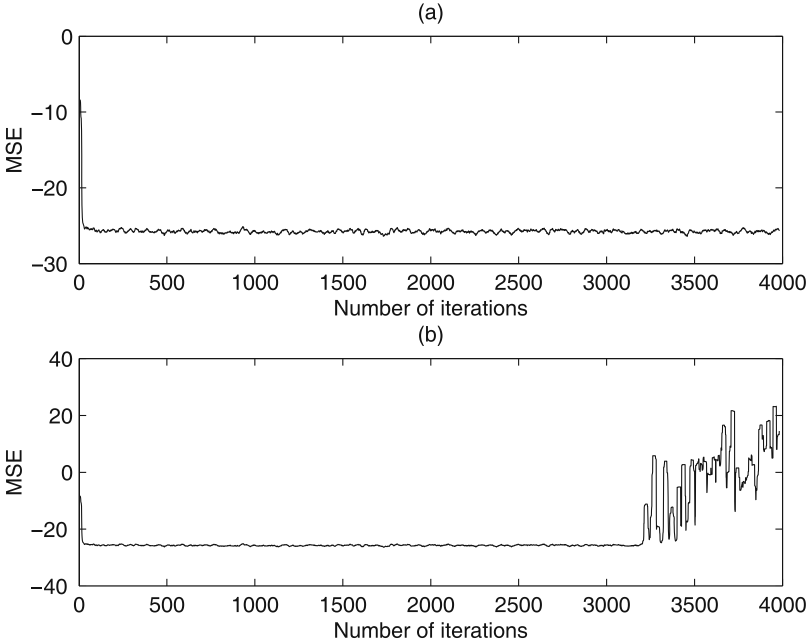


Fig. 7.3 (**a**) Mean squared error of HRLS, (**b**) Mean squared error of RLS.

where $\eta(k)$ is zero-mean Gaussian random noise with variance equal to 10^{-10}. Note that the 8×8 autocorrelation matrix of the input signal is nearly singular. A forgetting factor $\lambda = 0.99$ has been used in both schemes. As observed from Figure 7.3, RLS diverges after a number of iterations, while HRLS retains a numerically robust behavior.

Concerning the computational complexity, the HRLS algorithm requires slightly more arithmetic operations compared to the other square-root RLS schemes (conventional and inverse QRD-RLS algorithms). However, as indicated in the analysis performed in [9], the HRLS is the fastest numerically robust RLS algorithm in terms of MATLAB execution time, irrespective of the problem size N. This is due to the fact that the HRLS algorithm includes simple matrix-by-vector and vector-by-vector operations, which are best suited for implementation in the MATLAB environment.

It should be emphasized that the HRLS algorithm is closely related to the inverse QRD-RLS algorithm. In both algorithms, an orthogonal transformation is constructed by zeroing a vector quantity and then this transformation is applied for the update of an inverse square-root factor of the input data correlation matrix.

As shown in Chapter 3, the orthogonal transformation used in the inverse QRD-RLS algorithm can also be applied for the update of the square-root factor itself, leading to the QRD-RLS algorithm. It is easily verified that the same holds for the HRLS algorithm. More specifically, as proven in [6], the Householder matrix $\mathbf{P}(k)$ satisfies the following equations:

$$\mathbf{P}(k)\begin{bmatrix} -\mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{S}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}}(k) \\ \mathbf{S}(k) \end{bmatrix}, \text{ and} \tag{7.36}$$

$$\mathbf{P}(k)\begin{bmatrix} -d(k) \\ \lambda^{1/2}\hat{\mathbf{d}}_{q2}(k-1) \end{bmatrix} = \begin{bmatrix} e_{q1}(k) \\ \hat{\mathbf{d}}_{q2}(k) \end{bmatrix}. \tag{7.37}$$

where $e_{q1}(k)$ is the rotated estimation error defined in Chapter 3 and $\hat{\mathbf{d}}_{q2}(k)$ is an orthogonal transformation of the rotated desired signal vector satisfying $\mathbf{S}(k)\mathbf{w}(k) = \hat{\mathbf{d}}_{q2}(k)$. In the QRD-RLS algorithm, the generation of the orthogonal transformation and the update of the triangular Cholesky factor are performed jointly, without involving the inverse Cholesky factor. Similarly, from (7.36), except for the update of $\mathbf{S}(k-1)$, matrix $\mathbf{P}(k)$ could also be produced as an orthogonal row Householder reflection, which zeroes row $-\mathbf{x}^{\mathrm{T}}(k)$ with respect to the matrix $\lambda^{1/2}\mathbf{S}(k-1)$. However, the analysis of Section 7.1.2 has shown that, in order to construct such a matrix, the inverse of the square-root factor $\mathbf{S}(k-1)$ is also required (see Equation (7.16)). Thus, it seems that a Householder-based RLS algorithm equivalent to the QRD-RLS cannot be derived.

In the analysis presented in [10], it is shown that several variants of the RLS family are closely related to algorithms developed for the Kalman filtering problem. Under this framework, the QRD-RLS and the inverse QRD-RLS schemes are akin to the so-called information and square-root covariance filters, respectively [11, 12]. In a similar way, the HRLS algorithm is related to Potter's square-root covariance filter, which was the first square-root Kalman filter implementation, developed in the early 1960s [13]. It is noteworthy that Potter's square-root filter was the variant of the Kalman filter, which, due to its exceptional numerical properties, has been utilized in the navigation software of the Apollo system [14].

7.2.1 Applications

In the following, we briefly review two specific applications, namely adaptive pre-whitening and equalization in wideband code division multiple access (WCDMA) downlink, whereby the HRLS algorithm has been successfully utilized as a component of a larger system.

7.2.1.1 Adaptive pre-whitening

In several applications, such as blind source separation (BSS) or independent component analysis (ICA), it is desirable that the input signal measurements are whitened prior to applying any specific method. This procedure is known as pre-whitening [15]. Let us assume that the signal vector $\mathbf{x}(k)$ is wide sense stationary with zero mean and autocorrelation matrix $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)]$, where $E[\cdot]$ denotes statistical expectation. Then, to pre-whiten $\mathbf{x}(k)$ we need to determine a matrix transformation $\mathbf{T}$ such that, if

$$\mathbf{v}(k) = \mathbf{T}\mathbf{x}(k), \tag{7.38}$$

then

$$E[\mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(k)] = \mathbf{T}\mathbf{R}\mathbf{T}^{\mathrm{T}} = \mathbf{I}. \tag{7.39}$$

Apparently, (7.39) is satisfied if $\mathbf{T}$ is selected as the transpose of a square-root of $\mathbf{R}^{-1}$. Note that, under time-varying conditions, the whitening transformation $\mathbf{T}$ must be properly updated in time. Based on these two observations, a reasonable choice for $\mathbf{T}$ is

$$\mathbf{T} = \mathbf{S}^{-\mathrm{T}}(k-1), \tag{7.40}$$

i.e.,

$$\mathbf{v}(k) = \mathbf{S}^{-\mathrm{T}}(k-1)\mathbf{x}(k). \tag{7.41}$$

This transformation maintains the whitening property with respect to the deterministic autocorrelation matrix for every k, i.e.,

$$\mathbf{S}^{-\mathrm{T}}(k-1)\mathbf{R}(k-1)\mathbf{S}(k-1) = \mathbf{I}. \tag{7.42}$$

From (7.41) the transformed data vector is now $\mathbf{v}(k) = \sqrt{\lambda}\mathbf{u}(k)$, with $\mathbf{u}(k)$ defined in (7.21). Assuming $\mathbf{x}(k)$ to be wide sense stationary, $\mathbf{v}(k)$ enjoys the following property [8]:

$$\lim_{k\to\infty} E[\mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(k)] \approx (1-\lambda)\mathbf{I}. \tag{7.43}$$

The adaptive whitening procedure is summarized by the first four steps of the algorithm in Table 7.2. This procedure is numerically robust, as the update of the whitening transformation stems from the application of a Householder reflection.

7.2.1.2 Equalization in WCDMA downlink

Wideband code division multiple access (WCDMA) has been adopted in several modern telecommunication standards such as the Universal Mobile Telecommunication Systems (UMTS) standard. Due to strong interference from other users, a critical task in such systems is the design of an equalizer in the downlink. Most often, the conventional RAKE receiver is used, which is, however, interference limited. Another solution is to employ a linear minimum mean squares error equalizer, which provides the estimate of the kth symbol of the mth user as [16]

$$\hat{y}_m(k) = \mathbf{c}_m^{\mathrm{T}}(k)\boldsymbol{\Theta}^{\mathrm{T}}(\sigma_y^2\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}} + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{x}(k), \tag{7.44}$$

where $\mathbf{x}(k)$ is the vector of the received samples, $\boldsymbol{\Theta}$ is the system channel matrix, $\mathbf{c}_m^{\mathrm{T}}(k)$ is the spreading sequence of user k for symbol m, and $\sigma_y^2, \sigma_\eta^2$ are the variances of the transmitted sequence and the receiver noise, respectively. From (7.44) we can identify that the equalizer consists of two parts; the conventional RAKE receiver

$\mathbf{c}_m^{\mathrm{T}}(k)\boldsymbol{\Theta}^{\mathrm{T}}$ and a preceding filter $(\sigma_y^2\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}}+\sigma_\eta^2\mathbf{I})^{-1}$. It can be easily shown that this prefilter coincides with the inverse autocorrelation matrix $\mathbf{R}^{-1}$ of the received sequence $\mathbf{x}(k)$. By considering a Toeplitz approximation of $\mathbf{R}$, the prefilter coefficients are given by the elements of the middle row of $\mathbf{R}^{-1}$ [16]. If $\mathbf{s}_d^{\mathrm{T}}$ stands for the middle row of the square-root factor $\mathbf{S}^{-1}$ of $\mathbf{R}^{-1}$, the prefilter coefficients will be given by the elements of the following vector:

$$\mathbf{v}=\mathbf{S}^{-1}\mathbf{s}_d. \tag{7.45}$$

Therefore, explicit computation of the inverse autocorrelation matrix is not necessary. Instead, its square-root factor only need to be computed. Moreover, in an adaptive equalization setup, the prefilter coefficients can be updated by applying the HRLS algorithm, which provides, for every time instant the inverse square-root factor $\mathbf{S}^{-1}(k)$ [16], as shown in Table 7.2.

7.3 The Householder Block Exact QRD-RLS Algorithm

In a *block* RLS algorithm, the LS weight vector is updated in a data block-by-block basis, instead of the sample-by-sample updating as it is the case for the conventional RLS algorithms. In [17], a block QRD-RLS algorithm has been developed, where an exponentially block weighting was utilized. In the following, we slightly modify this algorithm, so that the LS weight vector, obtained at each block iteration, coincides with its sample-by-sample counterpart as this is computed by the conventional QRD-RLS algorithm. Such an algorithm is called block exact QRD-RLS algorithm and belongs to a more general class of such algorithms [18].

Let us assume a block length equal to Q. The LS error vector corresponding to $k+1$ blocks can be expressed as follows:

$$\mathbf{e}(k)=\begin{bmatrix}\mathbf{e}_k\\ \mathbf{e}_{k-1}\\ \vdots\\ \mathbf{e}_0\end{bmatrix}=\mathbf{d}(k)-\mathbf{X}(k)\mathbf{w}(k). \tag{7.46}$$

Let us define for $i=0,\ldots,k$ the $Q\times(N+1)$ data block $\mathbf{X}_i^{\mathrm{T}}$ and the $Q\times 1$ desired response block $\mathbf{d}_i$ as follows:

$$\mathbf{X}_i^{\mathrm{T}}=\begin{bmatrix}\mathbf{x}^{\mathrm{T}}((i+1)Q-1)\\ \vdots\\ \mathbf{x}^{\mathrm{T}}(iQ+1)\\ \mathbf{x}^{\mathrm{T}}(iQ)\end{bmatrix}=\begin{bmatrix}\mathbf{x}_{0,i}\ \mathbf{x}_{1,i}\ \cdots\ \mathbf{x}_{N,i}\end{bmatrix} \tag{7.47}$$

and

$$\mathbf{d}_i = \begin{bmatrix} d((i+1)Q-1) \\ \vdots \\ d(iQ+1) \\ d(iQ) \end{bmatrix}. \tag{7.48}$$

Then, the input data matrix $\mathbf{X}(k)$ and the desired response vector $\mathbf{d}(k)$ in (7.46) are expressed as follows:

$$\mathbf{X}(k) = \begin{bmatrix} \boldsymbol{\Lambda}\mathbf{X}_k^{\mathrm{T}} \\ \lambda^{Q/2}\boldsymbol{\Lambda}\mathbf{X}_{k-1}^{\mathrm{T}} \\ \vdots \\ \lambda^{kQ/2}\boldsymbol{\Lambda}\mathbf{X}_0^{\mathrm{T}} \end{bmatrix} \tag{7.49}$$

and

$$\mathbf{d}(k) = \begin{bmatrix} \boldsymbol{\Lambda}\mathbf{d}_k \\ \lambda^{Q/2}\boldsymbol{\Lambda}\mathbf{d}_{k-1} \\ \vdots \\ \lambda^{kQ/2}\boldsymbol{\Lambda}\mathbf{d}_0 \end{bmatrix}, \tag{7.50}$$

where $\boldsymbol{\Lambda}$ is a $Q \times Q$ weighting matrix given by

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1 & \cdots & 0 & 0 \\ 0 & \lambda^{1/2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda^{(Q-1)/2} \end{bmatrix}. \tag{7.51}$$

It is not difficult to verify from the above definitions that minimization of the norm of $\mathbf{e}(k)$ given in (7.46) with respect to $\mathbf{w}(k)$, provides the exact exponentially weighted LS solution for time instant $(k+1)Q-1$, i.e., $\mathbf{w}(k)$ minimizes the following cost function:

$$J(\mathbf{w}(k)) = \sum_{j=0}^{(k+1)Q-1} \lambda^{(k+1)Q-1-j}[d(j) - \mathbf{x}^{\mathrm{T}}(j)\mathbf{w}(k)]^2. \tag{7.52}$$

To see how we can obtain this solution adaptively in a block-by-block basis, we observe that $\mathbf{X}(k)$ from (7.49) can be rewritten as

$$\mathbf{X}(k) = \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{X}(k-1) \end{bmatrix}, \tag{7.53}$$

where $\tilde{\mathbf{X}}_k^{\mathrm{T}} = \boldsymbol{\Lambda}\mathbf{X}_k^{\mathrm{T}}$. From the last equation, the deterministic data correlation matrix at time $(k+1)Q-1$ can be expressed as follows:

$$\mathbf{R}(k) = \lambda^Q \mathbf{R}(k-1) + \tilde{\mathbf{X}}_k \tilde{\mathbf{X}}_k^{\mathrm{T}}, \tag{7.54}$$

where $\mathbf{R}(k-1) = \mathbf{X}^{\mathrm{T}}(k-1)\mathbf{X}(k-1)$ is the data correlation matrix at time $kQ-1$. By expressing $\mathbf{R}(k)$ and $\mathbf{R}(k-1)$ by the respective Cholesky factorizations, (7.54) is written as

$$\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \lambda^Q \mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1) + \tilde{\mathbf{X}}_k \tilde{\mathbf{X}}_k^{\mathrm{T}}, \tag{7.55}$$

where $\mathbf{U}(k), \mathbf{U}(k-1)$ are assumed to be $(N+1)\times(N+1)$ upper triangular matrices. From (7.55), a block time update of the Cholesky factor of the data correlation matrix can be realized according to the following relation:

$$\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}, \tag{7.56}$$

where $\tilde{\mathbf{P}}(k)$ is an orthogonal matrix. As suggested in (7.56), $\tilde{\mathbf{P}}(k)$ must be properly selected to zero the $Q\times(N+1)$ block $\tilde{\mathbf{X}}_k^{\mathrm{T}}$ with respect to the triangular matrix $\lambda^{Q/2}\mathbf{U}(k-1)$. Moreover, by following similar analysis as in Chapter 3 for the conventional QRD-RLS algorithm, it can be easily shown that $\tilde{\mathbf{P}}(k)$ block updates in time a rotated desired $(N+1)\times 1$ signal vector, $\mathbf{d}_{q_2}(k)$, as in

$$\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{d}}_k \\ \lambda^{Q/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{e}}_k \\ \mathbf{d}_{q_2}(k) \end{bmatrix}, \tag{7.57}$$

where $\tilde{\mathbf{d}}_k = \boldsymbol{\Lambda}\mathbf{d}_k$ and $\tilde{\mathbf{e}}_k$ being a $Q\times 1$ rotated error block. The exact LS weight vector at time $(k+1)Q-1$ is then given by the following triangular system of equations:

$$\mathbf{U}(k)\mathbf{w}(k) = \mathbf{d}_{q_2}(k), \tag{7.58}$$

which can be easily solved using back-substitution.

As mentioned before, matrix $\tilde{\mathbf{P}}(k)$ must be designed to eliminate the block $\tilde{\mathbf{X}}_k^{\mathrm{T}} = [\tilde{\mathbf{x}}_{0,k}, \tilde{\mathbf{x}}_{1,k}, \cdots, \tilde{\mathbf{x}}_{N,k}]$, where $\tilde{\mathbf{x}}_{n,k} = \boldsymbol{\Lambda}\mathbf{x}_{n,k}$, while retaining the triangular structure of the Cholesky factor. By inspecting (7.56), it is easily shown that $\tilde{\mathbf{P}}(k)$ can be expressed as a product of $N+1$ orthogonal $(N+Q+1)\times(N+Q+1)$ Householder matrices as follows:

$$\tilde{\mathbf{P}}(k) = \tilde{\mathbf{P}}_N(k)\tilde{\mathbf{P}}_{N-1}(k)\cdots\tilde{\mathbf{P}}_0(k). \tag{7.59}$$

Matrix $\tilde{\mathbf{P}}_n(k)$, $n = 0,1,\ldots,N$, zeroes the $(n+1)$-th column of the input data block with respect to the corresponding diagonal element of the upper triangular factor, i.e.,

$$\tilde{\mathbf{P}}_n(k) \begin{bmatrix} \mathbf{0} \cdots \tilde{\mathbf{x}}_{n,k}^{(n)} & \tilde{\mathbf{x}}_{n+1,k}^{(n)} & \cdots \tilde{\mathbf{x}}_{N,k}^{(n)} \\ & \mathbf{U}^{(n)}(k-1) & \end{bmatrix} = \begin{bmatrix} \mathbf{0} \cdots \mathbf{0} & \tilde{\mathbf{x}}_{n+1,k}^{(n+1)} & \cdots \tilde{\mathbf{x}}_{N,k}^{(n+1)} \\ & \mathbf{U}^{(n+1)}(k-1) & \end{bmatrix}, \tag{7.60}$$

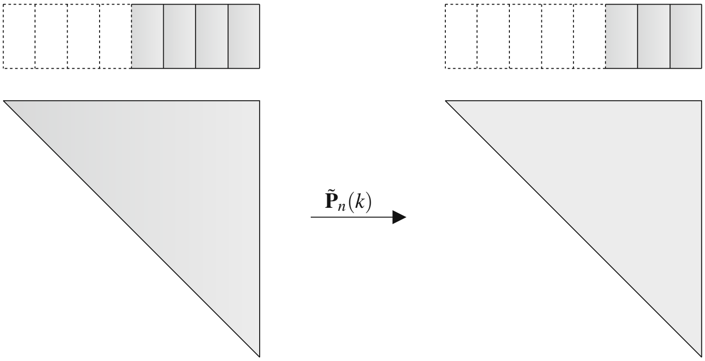


Fig. 7.4 Application of Householder transformations in the block exact QRD-RLS algorithm.

which is also illustrated in Figure 7.4. In (7.60), $\mathbf{U}^{(0)}(k-1) = \lambda^{Q/2}\mathbf{U}(k-1)$, $\mathbf{U}^{(N)}(k-1) = \mathbf{U}(k)$, and $\tilde{\mathbf{x}}_{j,k}^{(n)}, j = n,\ldots,N$ is the $(j+1)$-th column of the data block after the application of matrices $\tilde{\mathbf{P}}_0(k),\ldots,\tilde{\mathbf{P}}_{n-1}(k)$. To be more specific, $\tilde{\mathbf{P}}_n(k)$ is an orthogonal Householder matrix, whose defining vector, according to (7.6), is given by[3]

$$\mathbf{y}_n(k) = \begin{bmatrix} \tilde{\mathbf{x}}_{n,k}^{(n)} \\ \mathbf{0}_n \\ u_{n+1,n+1} + \rho_{n+1} \\ \mathbf{0}_{N-n} \end{bmatrix}, \tag{7.61}$$

where $u_{n+1,n+1}$ is the $(n+1)$-th diagonal element of $\mathbf{U}^{(n)}(k-1)$ and $\rho_{n+1}^2 = u_{n+1,n+1}^2 + ||\tilde{\mathbf{x}}_{n,k}^{(n)}||^2$.

The block exact Householder QRD-RLS algorithm is summarized in Table 7.3. It must be noticed that block updating using Householder transforms results in remarkable reduction in computational complexity compared to sample-by-sample updating using Givens rotations. More specifically, the Householder-based approach requires $QN^2 + \mathscr{O}[QN]$ arithmetic operations, whereas the Givens rotations approach needs $2QN^2 + \mathscr{O}[QN]$ operations, that is, the computational complexity is almost halved.

Moreover, as described in [17], a slightly modified version of the block Householder QRD-RLS algorithm can be implemented in a systolic array architecture, providing a throughput rate similar to that of the Givens rotations method.

[3] To be precise, due to the definition of the Householder transform in (7.5), (7.6), and (7.7), the resulting $\mathbf{U}(k)$ is the Cholesky factor of $\mathbf{R}(k)$ multiplied by -1. This, however, does not affect the analysis of the algorithm.

Table 7.3 The basic steps of the block exact Householder QRD-RLS algorithm.

```
Block exact Householder QRD-RLS
Initialize $0 \ll \lambda < 1, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{U}(0) = 1/\sqrt{\varepsilon}\mathbf{I}$
for each $k$
{ Computing $\tilde{\mathbf{P}}(k)$ and updating $\mathbf{U}(k-1)$:
  $\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{H}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}$
  Updating $\mathbf{d}_{q_2}(k-1)$:
  $\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{d}}_k^* \\ \lambda^{Q/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{e}}_k \\ \mathbf{d}_{q_2}(k) \end{bmatrix}$
  Obtaining $\mathbf{w}(k)$:
  $\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q_2}(k)$
}
```

7.4 The Householder Block Exact Inverse QRD-RLS Algorithm

In the block exact QRD-RLS algorithm, the orthogonal matrix $\tilde{\mathbf{P}}(k)$ and the updated Cholesky factor $\mathbf{U}(k)$ are jointly obtained from (7.56). Then, the inverse of $\mathbf{U}(k)$ need to be computed in order to extract the LS weight vector according to (7.58). Clearly, it would be more convenient to be able to work directly with the inverse of the Cholesky factor. It is well known that if $\tilde{\mathbf{P}}(k)$ is orthogonal and satisfies (7.56), then

$$\tilde{\mathbf{P}}(k) \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \lambda^{-Q/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{E}^{\mathrm{T}}(k) \\ \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix}, \tag{7.62}$$

where $\mathbf{E}(k)$ is an $(N+1) \times Q$ matrix. To see this, let us note that we can express an identity matrix as follows:

$$\begin{aligned} \mathbf{I} &= \begin{bmatrix} \mathbf{O}_{(N+1)\times Q} & \lambda^{-Q/2}\mathbf{U}^{-1}(k-1) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{O}_{(N+1)\times Q} & \lambda^{-Q/2}\mathbf{U}^{-1}(k-1) \end{bmatrix} \tilde{\mathbf{P}}^{\mathrm{T}}(k)\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}(k) \ \mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix} \end{aligned} \tag{7.63}$$

and thus $\mathbf{W} = \mathbf{U}^{-1}(k)$. From (7.62), the inverse Cholesky factor can be updated. However, in order to compute $\tilde{\mathbf{P}}(k)$ we have to resort to (7.56), which requires the Cholesky factor itself. To avoid using $\mathbf{U}(k-1)$ explicitly, the following lemma, which has been derived in [4], can be employed.

Lemma 1. *Let* $\mathbf{G}(k) = -\lambda^{-Q/2}\mathbf{U}^{-T}(k-1)\tilde{\mathbf{X}}_k$ *and let* $\hat{\mathbf{P}}(k)$ *be an orthogonal matrix such that*

$$\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{I}_Q\\ \mathbf{G}(k)\end{bmatrix}=\begin{bmatrix}\boldsymbol{\Delta}(k)\\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}, \tag{7.64}$$

where $\mathbf{I}_Q$ *is the* $Q\times Q$ *identity matrix and* $\boldsymbol{\Delta}(k)$ *is a* $Q\times Q$ *matrix. Then*

$$\hat{\mathbf{P}}(k)\begin{bmatrix}\tilde{\mathbf{X}}_k^T\\ \lambda^{Q/2}\mathbf{U}(k-1)\end{bmatrix}=\begin{bmatrix}\mathbf{O}_{Q\times(N+1)}\\ \mathbf{V}\end{bmatrix}. \tag{7.65}$$

If $\mathbf{V}$ *is upper triangular, then* $\mathbf{V}=\mathbf{U}(k)$ *and*

$$\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)}\\ \lambda^{-Q/2}\mathbf{U}^{-T}(k-1)\end{bmatrix}=\begin{bmatrix}\mathbf{E}^T(k)\\ \mathbf{U}^{-T}(k)\end{bmatrix}, \tag{7.66}$$

where $\mathbf{E}(k)=\lambda^{-Q/2}\mathbf{U}^{-1}(k-1)\mathbf{G}(k)\boldsymbol{\Delta}^{-1}(k)$.

The crucial task now becomes to determine the orthogonal matrix in (7.64) so that the triangular structures are retained in (7.66). As a generalization of the procedure for $Q=1$ (inverse QRD-RLS algorithm of Chapter 3), $\hat{\mathbf{P}}(k)$ can be constructed as the product of $N+1$ orthogonal row Householder reflections [4], i.e,

$$\hat{\mathbf{P}}(k)=\hat{\mathbf{P}}_N(k)\hat{\mathbf{P}}_{N-1}(k)\cdots\hat{\mathbf{P}}_0(k). \tag{7.67}$$

The row Householder matrix $\hat{\mathbf{P}}_n(k), n=0,1,\ldots,N$, zeros the $(n+1)$-th row of $\mathbf{G}(k)$ as follows:

$$\hat{\mathbf{P}}_n(k)\begin{bmatrix}\boldsymbol{\Delta}_n(k)\\ \mathbf{0}^{\mathrm{T}}\\ \vdots\\ \mathbf{g}_n^{\mathrm{T}}(k)\\ \mathbf{g}_{n+1}^{\mathrm{T}}(k)\\ \vdots\\ \mathbf{g}_N^{\mathrm{T}}(k)\end{bmatrix}=\begin{bmatrix}\boldsymbol{\Delta}_{n+1}(k)\\ \mathbf{0}^{\mathrm{T}}\\ \vdots\\ \mathbf{0}^{\mathrm{T}}\\ \mathbf{g}_{n+1}^{\mathrm{T}}(k)\\ \vdots\\ \mathbf{g}_N^{\mathrm{T}}(k)\end{bmatrix} \tag{7.68}$$

where $\boldsymbol{\Delta}_0(k)=\mathbf{I}_Q$, $\boldsymbol{\Delta}_N(k)=\boldsymbol{\Delta}(k)$, and $\mathbf{g}_n^{\mathrm{T}}(k)$ is the $(n+1)$-th row of $\mathbf{G}(k)$ for $n=0,1,\ldots,N$. This procedure is also graphically illustrated in Figure 7.5. From the analysis on row Householder matrices and (7.68), the defining vector of the row Householder matrix $\hat{\mathbf{P}}_n(k)$ is given by

$$\mathbf{y}_n(k)=\frac{1}{||\mathbf{g}_n(k)||}\begin{bmatrix}\boldsymbol{\Delta}_n^{-\mathrm{T}}(k)\mathbf{g}_n(k)\\ \mathbf{0}_n\\ -1-\sqrt{1+(\boldsymbol{\Delta}_n^{-\mathrm{T}}(k)\mathbf{g}_n(k))^{\mathrm{T}}\boldsymbol{\Delta}_n^{-\mathrm{T}}(k)\mathbf{g}_n(k)}\\ \mathbf{0}_{N-n}\end{bmatrix}. \tag{7.69}$$

Recall that $\hat{\mathbf{P}}_n(k)$ need not be explicitly constructed, but, instead, $\mathbf{y}_n(k)$ is computed from (7.69) and is properly used in (7.68). Thus, the orthogonal matrix $\hat{\mathbf{P}}(k)$ can be constructed from (7.67), (7.68), and (7.69) and then applied in (7.66) in order to

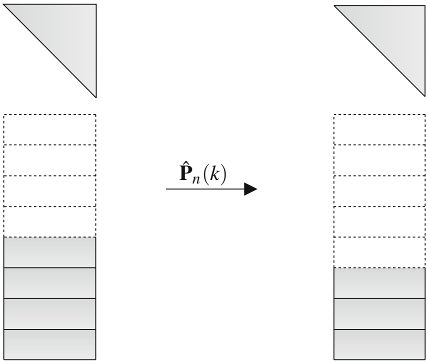


Fig. 7.5 Application of row Householder transformations in the block exact inverse QRD-RLS algorithm.

update directly the inverse Cholesky factor. Furthermore, the LS weight vector can be efficiently updated using the following block recursive formula [4]:

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \mathbf{E}(k)\boldsymbol{\Delta}^{-\mathrm{T}}(k)\left[\tilde{\mathbf{d}}_k - \tilde{\mathbf{X}}_k^{\mathrm{T}}\mathbf{w}(k-1)\right], \tag{7.70}$$

where $\mathbf{w}(k)$ refers to the exponentially weighted LS solution at time $(k+1)Q-1$.

The Householder block exact inverse QRD-RLS algorithm is summarized in Table 7.4.

As mentioned before, in the second step of the algorithm, inversion of the $Q \times Q$ matrices $\boldsymbol{\Delta}_n(k)$ is required. Note, however, that in most applications that employ a block-type recursive scheme, the block length Q is taken to be much smaller than the filter size N [18]. Therefore, the overall burden in complexity, which is due to these inverse matrix calculations, is not considerable.

Table 7.4 The basic steps of the block exact Householder inverse QRD-RLS algorithm.

Block exact Householder inverse QRD-RLS

```
Initialize $0 \ll \lambda < 1, \mathbf{w}(0) = \mathbf{0}, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{U}^{-1}(0) = \sqrt{\varepsilon}\mathbf{I}$
for each $k$
{ Computing $\mathbf{G}(k)$:
  $\mathbf{G}(k) = -\lambda^{-Q/2}\mathbf{U}^{-\mathrm{H}}(k-1)\tilde{\mathbf{X}}_k$
  Computing $\hat{\mathbf{P}}(k)$ and $\boldsymbol{\Delta}(k)$:
  $\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{I}_Q \\ \mathbf{G}(k)\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Delta}(k) \\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}$
  Updating $\mathbf{U}^{-1}(k-1)$:
  $\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \lambda^{-Q/2}\mathbf{U}^{-\mathrm{H}}(k-1)\end{bmatrix} = \begin{bmatrix}\mathbf{E}^{\mathrm{H}}(k) \\ \mathbf{U}^{-\mathrm{H}}(k)\end{bmatrix}$
  Computing $\mathbf{w}(k)$:
  $\mathbf{w}(k) = \mathbf{w}(k-1) - \mathbf{E}(k)\boldsymbol{\Delta}^{-\mathrm{H}}(k)\left[\tilde{\mathbf{d}}_k^* - \tilde{\mathbf{X}}_k^{\mathrm{H}}\mathbf{w}(k-1)\right]$
}
```

7.5 Sliding Window (SW) Householder Block Implementation

In the previous analysis, an exponentially weighted LS cost function has been considered, which is more frequently used in practice. In a time-varying environment, an alternative popular approach is to employ a *sliding* window (SW) on the data. To reduce complexity, data can be organized in blocks of size Q, and the LS weight vector can be updated on a block-by-block basis. In a block SW formulation, the LS error vector, whose norm is to be minimized, is given by

$$\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_k \\ \mathbf{e}_{k-1} \\ \vdots \\ \mathbf{e}_{k-L+1} \end{bmatrix} = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k), \tag{7.71}$$

where L is the window size,

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{X}_k^{\mathrm{T}} \\ \mathbf{X}_{k-1}^{\mathrm{T}} \\ \vdots \\ \mathbf{X}_{k-L+1}^{\mathrm{T}} \end{bmatrix} \tag{7.72}$$

and

$$\mathbf{d}(k) = \begin{bmatrix} \mathbf{d}_k \\ \mathbf{d}_{k-1} \\ \vdots \\ \mathbf{d}_{k-L+1} \end{bmatrix}. \tag{7.73}$$

The $Q \times (N+1)$ input data matrices $\mathbf{X}_i^{\mathrm{T}}$ and the $Q \times 1$ desired response vectors $\mathbf{d}_i$, $i = k-L+1, \ldots, k$, are given by (7.47) and (7.48), respectively. Let us now define the following augmented quantities:

$$\bar{\mathbf{X}}(k) = \begin{bmatrix} \mathbf{X}(k) \\ \mathbf{X}_{k-L}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}_k^{\mathrm{T}} \\ \mathbf{X}(k-1) \end{bmatrix}, \text{ and} \tag{7.74}$$

$$\bar{\mathbf{d}}(k) = \begin{bmatrix} \mathbf{d}(k) \\ \mathbf{d}_{k-L} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_k \\ \mathbf{d}(k-1) \end{bmatrix}. \tag{7.75}$$

The LS problem can be solved recursively via a two-step procedure. This procedure comprises an update step and a downdate step, as described in the following representation:

$$\mathbf{w}(k-1) \longrightarrow \bar{\mathbf{w}}(k) \longrightarrow \mathbf{w}(k), \tag{7.76}$$

where $\bar{\mathbf{w}}(k)$ is the solution of the LS problem, which results by substituting in (7.71) $\mathbf{X}(k)$ and $\mathbf{d}(k)$ by $\bar{\mathbf{X}}(k)$ and $\bar{\mathbf{d}}(k)$, respectively. The update step can be implemented directly by using one of the algorithms described in Sections 7.3 and 7.4. For the downdate step, the following relation between the involved data correlation matrices holds:

$$\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) = \bar{\mathbf{X}}^{\mathrm{T}}(k)\bar{\mathbf{X}}(k) - \mathbf{X}_{k-L}\mathbf{X}_{k-L}^{\mathrm{T}} \tag{7.77}$$

or

$$\mathbf{R}(k) = \bar{\mathbf{R}}(k) - \mathbf{X}_{k-L}\mathbf{X}_{k-L}^{\mathrm{T}}. \tag{7.78}$$

By expressing the correlation matrices in terms of their Cholesky factors, the last equation is rewritten as follows:

$$\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \bar{\mathbf{U}}^{\mathrm{T}}(k)\bar{\mathbf{U}}(k) - \mathbf{X}_{k-L}\mathbf{X}_{k-L}^{\mathrm{T}}. \tag{7.79}$$

In [3], it is shown that there exists a hypernormal matrix $\mathbf{H}(k)$ with respect to the signature

$$\boldsymbol{\Phi} = \begin{bmatrix} -\mathbf{I}_Q & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{N+1} \end{bmatrix} \tag{7.80}$$

such that

$$\mathbf{H}(k)\begin{bmatrix} \mathbf{X}_{k-L}^{\mathrm{T}} \\ \bar{\mathbf{U}}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}. \tag{7.81}$$

Matrix $\mathbf{H}(k)$ can be constructed as the product of $N+1$ hyperbolic Householder matrices, which annihilate the columns of $\mathbf{X}_{k-L}^{\mathrm{T}}$ with respect to the diagonal elements of $\bar{\mathbf{U}}(k)$. Note that $\bar{\mathbf{U}}(k)$ in (7.81) has been obtained from the initial update step, and thus (7.81) provides the Cholesky factor of the SW RLS problem at time k. However, in order to compute the required LS solution $\mathbf{w}(k)$, we have to resort to the inverse Cholesky factor. It is easily shown that $\mathbf{H}(k)$ can also be used to compute $\mathbf{U}^{-1}(k)$ from $\bar{\mathbf{U}}^{-1}(k)$ as in [4]

$$\mathbf{H}(k)\begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \bar{\mathbf{U}}^{-T}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{\mathrm{T}}(k) \\ \mathbf{U}^{-T}(k) \end{bmatrix}, \tag{7.82}$$

where $\mathbf{F}(k)$ is an $(N+1)\times Q$ matrix. To compute $\mathbf{H}(k)$, a result similar to that of Lemma 1 can be employed [4]. More specifically, by defining the vector $\bar{\mathbf{G}}(k) = -\bar{\mathbf{U}}^{-\mathrm{T}}(k)\mathbf{X}_{k-L}$, a hypernormal matrix $\mathbf{H}(k)$ such that

$$\mathbf{H}(k)\begin{bmatrix} \mathbf{I}_Q \\ \bar{\mathbf{G}}(k) \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\Delta}}(k) \\ \mathbf{O}_{(N+1)\times Q} \end{bmatrix}, \tag{7.83}$$

where $\bar{\boldsymbol{\Delta}}(k)$ is a $Q\times Q$ matrix, also satisfies (7.82). To retain the triangular structure of the matrices in (7.82), $\mathbf{H}(k)$ is constructed as a sequence of $N+1$ row hyperbolic Householder reflections $\mathbf{H}_n(k)$, $n = 0,1,\ldots,N$, with respect to the signature $\boldsymbol{\Phi}$ given in (7.80). $\mathbf{H}_n(k)$ annihilates the $(n+1)$-th row of $\bar{\mathbf{G}}(k)$ with respect to the upper $Q\times Q$ block of the matrix on the left-hand-side of (7.83). Thus, from the analysis on row Householder matrices and (7.80), its defining vector can be written as

Table 7.5 The basic steps of the sliding window (SW) block HRLS algorithm.

```
SW block Householder RLS
Run the update step $L$ times to obtain $\mathbf{w}(L)$ and $\mathbf{U}^{-1}(L)$
for each $k > L$
{ Update Step
    Computing $\mathbf{G}(k)$:
    $\mathbf{G}(k) = -\mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{X}_k$
    Computing $\mathbf{Q}(k)$:
    $\mathbf{Q}(k)\begin{bmatrix}\mathbf{I}_Q \\ \mathbf{G}(k)\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Delta}(k) \\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}$
    Updating $\mathbf{U}^{-1}(k-1)$:
    $\mathbf{Q}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}^{-\mathrm{H}}(k-1)\end{bmatrix} = \begin{bmatrix}\mathbf{E}^{\mathrm{H}}(k) \\ \bar{\mathbf{U}}^{-\mathrm{H}}(k)\end{bmatrix}$
    Computing $\bar{\mathbf{w}}(k)$:
    $\bar{\mathbf{w}}(k) = \mathbf{w}(k-1) - \mathbf{E}(k)\boldsymbol{\Delta}^{-\mathrm{H}}(k)\left[\mathbf{d}_k^* - \mathbf{X}_k^{\mathrm{H}}\mathbf{w}(k-1)\right]$
  Downdate Step
    Computing $\bar{\mathbf{G}}(k)$:
    $\bar{\mathbf{G}}(k) = -\bar{\mathbf{U}}^{-\mathrm{H}}(k)\mathbf{X}_{k-L}$
    Computing $\mathbf{H}(k)$:
    $\mathbf{H}(k)\begin{bmatrix}\mathbf{I}_Q \\ \bar{\mathbf{G}}(k)\end{bmatrix} = \begin{bmatrix}\bar{\boldsymbol{\Delta}}(k) \\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}$
    Downdating $\bar{\mathbf{U}}^{-1}(k)$:
    $\mathbf{H}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \bar{\mathbf{U}}^{-\mathrm{H}}(k)\end{bmatrix} = \begin{bmatrix}\mathbf{F}^{\mathrm{H}}(k) \\ \mathbf{U}^{-\mathrm{H}}(k)\end{bmatrix}$
  Computing $\mathbf{w}(k)$:
    $\mathbf{w}(k) = \bar{\mathbf{w}}(k) - \mathbf{F}(k)\bar{\boldsymbol{\Delta}}^{-\mathrm{H}}(k)\left[\mathbf{d}_{k-L}^* - \mathbf{X}_{k-L}^{\mathrm{H}}\bar{\mathbf{w}}(k)\right]$
}
```

$$\bar{\mathbf{y}}_n(k) = \frac{1}{||\bar{\mathbf{g}}_n(k)||}\begin{bmatrix} \bar{\boldsymbol{\Delta}}_n^{-\mathrm{T}}(k)\bar{\mathbf{g}}_n(k) \\ \mathbf{0}_n \\ -1-\sqrt{1-(\bar{\boldsymbol{\Delta}}_n^{-\mathrm{T}}(k)\bar{\mathbf{g}}_n(k))^{\mathrm{T}}\bar{\boldsymbol{\Delta}}_n^{-\mathrm{T}}(k)\bar{\mathbf{g}}_n(k)} \\ \mathbf{0}_{N-n} \end{bmatrix}, \tag{7.84}$$

where $\bar{\boldsymbol{\Delta}}_0(k) = \mathbf{I}_Q$, $\bar{\boldsymbol{\Delta}}_N(k) = \bar{\boldsymbol{\Delta}}(k)$, and $\bar{\mathbf{g}}_n^{\mathrm{T}}(k)$ is the $(n+1)$-th row of $\bar{\mathbf{G}}(k)$ for $n = 0, 1, \ldots, N$. Furthermore, the LS weight vector is computed according to the following recursive formula [4]:

$$\mathbf{w}(k) = \bar{\mathbf{w}}(k) + \mathbf{F}(k)\bar{\boldsymbol{\Delta}}^{-\mathrm{T}}(k)(\mathbf{d}_{k-L} - \mathbf{X}_{k-L}^{\mathrm{T}}\bar{\mathbf{w}}(k)). \tag{7.85}$$

The SW block HRLS algorithm is summarized in Table 7.5. The algorithm consists of an update step, implemented with the block inverse QRD-RLS algorithm using a row orthogonal Householder matrix $\mathbf{Q}(k)$, and a downdate step as described in the previous analysis. Note that in the initialization phase the update step is executed L times, before the algorithm switches to the two-step procedure.

7.6 Conclusion

The aim of the chapter was to present various RLS algorithms, whose recursion procedure is based on Householder transforms. Householder transforms are known to possess exceptional numerical properties, and thus the resulting RLS schemes exhibit numerical robustness in finite-precision environments. The HRLS algorithm was first presented, which updates in time an arbitrary square-root of the data correlation matrix using an orthogonal Householder transformation. The algorithm is particularly attractive in several applications, due to its low-computational complexity and numerical robustness. In the sequel, three other Householder-based RLS algorithms have been described. The common characteristic of these schemes is that the weight vector is updated on a block-by-block basis, which calls directly for the application of Householder reflections instead of Givens rotations. The first two algorithms can be considered as generalizations of the conventional sample-by-sample QRD-RLS and inverse QRD-RLS algorithms, respectively. The block exact QRD-RLS scheme provides the per block update of the data correlation matrix Cholesky factor, with the application of a sequence of orthogonal Householder matrices. The block exact inverse QRD-RLS algorithm block updates the inverse Cholesky factor through a sequence of row orthogonal Householder matrices. The third algorithm was a SW block RLS scheme, which also manipulates the inverse Cholesky factor. The algorithm provides the respective LS weight vector by utilizing two Householder transformations, namely a row orthogonal and a row hyperbolic, in order to update and downdate the inverse Cholesky factor, respectively.

References

1. A. O. Householder, The Theory of Matrices in Numerical Analysis. Dover Publications Inc., New York, NY, USA (1964)
2. A. O. Steinhardt, Householder transforms in signal processing. IEEE Signal Processing Magazine, vol. 5, pp. 4–12 (July 1988)
3. C. M. Rader and A. O. Steinhardt, Hyperbolic Householder transformations. IEEE Transactions on Acoustics Speech and Signal Processing, vol. 34, no. 6, pp. 1859–1602 (December 1986)
4. A. W. Bojanczyk, G. G. Nagy and R. J. Plemmons, Block RLS using row Householder reflections. Linear Algebra and its Applications, vol. 188–189, pp. 31–61 (1993)
5. S. Haykin, Adaptive Filter Theory. 4th edition, Prentice-Hall Inc., Upper Saddle River, NJ, USA (2002)
6. A. A. Rontogiannis and S. Theodoridis, On inverse factorization adaptive least squares algorithms. Signal Processing (Elsevier), vol. 52, pp. 35–47 (July 1996)
7. A. A. Rontogiannis and S. Theodoridis, An adaptive LS algorithm based on orthogonal Householder transformations. IEEE International Conference on Electronics, Circuits and Systems, ICECS'96, Rhodes, Greece, pp. 860–863 (October 1996)
8. S. C. Douglas, Numerically robust $O(N^2)$ RLS algorithms using least-squares prewhitening. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '2000, Istanbul, Turkey (June 2000)
9. S. C. Douglas and R. Losada, Adaptive filters in Matlab: from novice to expert. IEEE Signal Processing Education Workshop, Pine Mountain, USA, pp. 168–173 (October 2002)

10. A. H. Sayed and T. Kailath, A state-space approach to adaptive RLS filtering. IEEE Signal Processing Magazine, vol. 11, pp. 18–60 (July 1994)
11. G. J. Bierman, Factorization Methods for Discrete Sequential Estimation. Academic Press, New York, NY, USA (1977)
12. G. J. Bierman and C. L. Thronton, Numerical comparison of Kalman filter algorithms: orbit determination case study. Automatica, vol. 13, pp. 13–25 (January 1977)
13. R. Batin, Astronautical Guidance. McGraw-Hill Book Company, New York, NY, USA (1964)
14. L. A. McGee and S. F. Schmidt, Discovery of the Kalman filter as a practical tool for aerospace and industry. NASA Technical Memorandum 86847, USA (November 1985)
15. S. C. Douglas, Blind source separation and independent component analysis: a crossroads of tools and ideas. Fourth International Symposium on Independent Component Analysis and Blind Source Separation, Napa, Japan (April 2003)
16. K. Hooli, M. Latva-aho and M. Juntti, Performance evaluation of adaptive chip-level channel equalizers in WCDMA downlink. IEEE International Conference on Communications, ICC'2001, Helsinki, Finland (June 2001)
17. K. R. Liu, S.-F. Hsieh and K. Yao, Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation. IEEE Transactions on Signal Processing, vol. 40, no. 4, pp. 946–958 (April 1998)
18. G.-O. Glentis, K. Berberidis and S. Theodoridis, Efficient least squares adaptive algorithms for FIR transversal filtering. IEEE Signal Processing Magazine, vol. 16, pp. 13–41 (July 1999)

Chapter 8
Numerical Stability Properties

Phillip Regalia and Richard Le Borne

Abstract Designers of algorithms must not only solve the problem of interest, but do so using methods which are robust under perturbations in the data as well as the intermediate parameters of the method. More generally, it is often the case that the actual problem of interest is too complicated to solve directly; simplifying assumptions are necessary. At each stage, from problem identification, to the setup of the problem to be solved using some method, to the ultimate algorithm to be implemented in code, perturbations and their effects must be anticipated and analyzed. Stability is the property that assesses the level of robustness to perturbations that is required before the computed solution given by an algorithm can be used with confidence. The origin of the perturbation can vary, as pointed out above. What is important, however, is to have analysis that supports the premise that a small change in the problem results in a small change to the solution.

8.1 Introduction

The use of the QR-decomposition (vs. QR-algorithm) depends greatly on its reputation for providing consistently usable results. When an algorithm's reputation suffers, whether deserved or not, users often seek an alternative. It is therefore important, if not essential, to first establish the criteria in which a reliable solution can be guaranteed. For example, the reputation of Gaussian elimination suffered greatly in the 1940s because of its inability to always provide a usable solution. It was not until an analysis performed by J. H. Wilkinson [1], that introduced the

Phillip Regalia
Catholic University of America, Washington, DC – USA
e-mail: regalia@cua.edu

Richard Le Borne
Tennessee Technological University, Cookeville, TN – USA
e-mail: rleborne@tntech.edu

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_8, © Springer Science+Business Media, LLC 2009

relationship between perturbations and the conditioning of the problem, before confidence in Gaussian elimination could be re-established. In this chapter, we will focus our attention on stability and what it means in the context of the development process that begins with the clear articulation of the problem and ends with the computer implementation of the algorithm.

8.2 Preliminaries

The usefulness of a computed solution is a statement assessing its numerical accuracy. When the estimate of the desired solution is a vector, required is a means for assessing its accuracy. The vector norm assigns a single number to a vector and this property is very useful for assessing the error in a given quantity. There are different vector norms, but the most often used are considered equivalent in $\mathbb{R}^N$ in that one norm is within a constant factor of another. For example, the Euclidean norm, or two-norm, is defined as the square-root of the sum of squares of vector elements. For the vector of filter weights, $\mathbf{w} = [w_1, w_2, \ldots, w_N]^{\mathrm{T}}$, its Euclidean norm, $||\mathbf{w}||_2$ is given by

$$\begin{aligned} ||\mathbf{w}||_2 &= \sqrt{w_1^2 + w_2^2 + \cdots + w_N^2} && (8.1)\\ &= \sqrt{\sum_{i=1}^{N} w_i^2} && (8.2) \end{aligned}$$

For matrices, the norm again is used to associate a single number to it but its definition is chosen to be compatible with vector norms. For example, the matrix norm associated with the Euclidean norm is called the spectral norm. For $\mathbf{X} \in \mathbb{R}^{N\times M}$, its spectral norm, $||\mathbf{X}||_2$, is defined as

$$\begin{aligned} ||\mathbf{X}||_2 &= \max_{||\mathbf{w}||_2=1} ||\mathbf{X}\mathbf{w}||_2 && (8.3)\\ &= \sqrt{\lambda_{max}(\mathbf{X}^{\mathrm{T}}\mathbf{X})} && (8.4)\\ &= \sigma_{max}(\mathbf{X}), && (8.5) \end{aligned}$$

where $\lambda_{max}(\mathbf{X}^{\mathrm{T}}\mathbf{X})$ is the maximum eigenvalue of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\sigma_{max}(\mathbf{X})$ is the largest singular value of the data matrix $\mathbf{X}$. The singular value of a matrix will next be defined. For a full discussion on vector and matrix norms a good source is [2, Chapter 2].

Suppose at time index k, we are given the data matrix $\mathbf{X}(k) \in \mathbb{R}^{(k+1)\times(N+1)}$ of rank r. Then there are unitary matrices $\mathbf{U}(k) \in \mathbb{R}^{(k+1)\times(k+1)}$ and $\mathbf{V}(k) \in \mathbb{R}^{(N+1)\times(N+1)}$ where $\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \mathbf{I} \in \mathbb{R}^{(k+1)\times(k+1)}$ and $\mathbf{V}^{\mathrm{T}}(k)\mathbf{V}(k) = \mathbf{I} \in \mathbb{R}^{(N+1)\times(N+1)}$ and a diagonal matrix $\boldsymbol{\Sigma}(k) \in \mathbb{R}^{(k+1)\times(N+1)}$ where $\boldsymbol{\Sigma}(k) = \mathrm{diag}(\sigma_1(k), \sigma_2(k), \ldots, \sigma_r(k), 0, \ldots, 0) = \mathrm{diag}(\boldsymbol{\Sigma}_+(k), 0, \ldots, 0)$ with $\sigma_1(k) \geq \sigma_2(k) \geq \ldots \geq \sigma_r > 0$. Then the singular value decomposition is given by:

Definition 1 (Singular Value Decomposition).

$$\mathbf{X}(k) = \mathbf{U}(k)\boldsymbol{\Sigma}(k)\mathbf{V}^{\mathrm{T}}(k) \tag{8.6}$$

$$= \mathbf{U}(k)\begin{bmatrix}\boldsymbol{\Sigma}_{+}(k) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{V}^{\mathrm{T}}(k) \tag{8.7}$$

Here, $\sigma_i(k),\ i=1,\ldots,r$ are the singular values of $\mathbf{X}(k)$ and for

$$\begin{aligned}\mathbf{U}(k) &= [\mathbf{u}_1(k),\ldots,\mathbf{u}_{k+1}(k)] &(8.8)\\ \mathbf{V}(k) &= [\mathbf{v}_1(k),\ldots,\mathbf{v}_{N+1}(k)], &(8.9)\end{aligned}$$

we have that the $\mathbf{u}_i(k),\ i=1,\ldots,k+1$ and $\mathbf{v}_j(k),\ j=1,\ldots,N+1$ are, respectively, the left and right singular vectors associated with the singular values:

$$\begin{aligned}\mathbf{X}^{\mathrm{T}}(k)\mathbf{u}_i(k) &= \sigma_i\mathbf{v}_i(k), \quad i=1,\ldots,r &(8.10)\\ \mathbf{X}^{\mathrm{T}}(k)\mathbf{u}_i(k) &= 0, \quad i=r+1,\ldots,k+1, &(8.11)\end{aligned}$$

and,

$$\begin{aligned}\mathbf{X}(k)\mathbf{v}_i(k) &= \sigma_i\mathbf{u}_i(k), \quad i=1,\ldots,r &(8.12)\\ \mathbf{X}(k)\mathbf{v}_i(k) &= 0, \quad i=r+1,\ldots,N+1 &(8.13)\end{aligned}$$

It also holds that the square of the ith right singular value, $\sigma_i(k)$, $i=1, \ldots, N+1$ is equal to the ith eigenvalue of the correlation matrix, $\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)$, that is, $\lambda_i(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)) = \sigma_i^2$, and $\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\mathbf{v}_i(k) = \lambda_i\mathbf{v}(k) = \sigma_i^2\mathbf{v}_i(k)$, $i=1, \ldots, N+1$.

The pseudo-inverse, or generalized inverse, $\mathbf{X}^{\dagger}(k)$ can then be defined using the singular value decomposition:

Definition 2 (Pseudo-Inverse).

$$\underbrace{\mathbf{X}^{\dagger}(k)}_{(N+1)\times(k+1)} = \mathbf{V}(k)\underbrace{\begin{bmatrix}\boldsymbol{\Sigma}_{+}^{-1}(k) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{(N+1)\times(k+1)}\mathbf{U}^{\mathrm{T}}(k) \tag{8.14}$$

Consider the least-squares filtering problem which seeks to find the weight vector $\mathbf{w}(k)$ such that

$$\mathbf{X}(k)\,\mathbf{w}(k) = \mathbf{d}(k) \tag{8.15}$$

with $\mathbf{w}(k)$ satisfying

$$\min_{\mathbf{w}} \|\mathbf{d}(k) - \mathbf{X}(k)\,\mathbf{w}\|_2. \tag{8.16}$$

It is well known [2, Chapter 3] that $\mathbf{w}(k)$ solves (8.15) if $\mathbf{w}(k) = \mathbf{X}^{\dagger}(k)\mathbf{d}(k) + (\mathbf{I} - \mathbf{X}^{\dagger}(k)\mathbf{X}(k))\mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^{(N+1)\times 1}$ is arbitrary. For (8.16), $\mathbf{z}$ must obviously be the zero vector so that

$$\mathbf{w}(k) = \mathbf{X}^{\dagger}(k)\,\mathbf{d}(k). \tag{8.17}$$

Note that for the case that $\mathbf{X}(k)$ has full rank, i.e., $\text{rank}(\mathbf{X}(k)) = N+1$, then $\mathbf{X}^{\dagger}(k)\mathbf{X}(k) = \mathbf{I}^{(N+1)\times(N+1)}$ and the solution is unique and can be given in terms of the normal equations:

$$\mathbf{X}^{\mathrm{T}}(k)\,\mathbf{X}(k)\,\mathbf{w}(k) = \mathbf{X}^{\mathrm{T}}(k)\,\mathbf{d}(k) \tag{8.18}$$

$$\mathbf{w}(k) = \left(\mathbf{X}^{\mathrm{T}}(k)\,\mathbf{X}(k)\right)^{-1}\mathbf{X}^{\mathrm{T}}(k)\,\mathbf{d}(k) \tag{8.19}$$

$$\mathbf{w}(k) = \mathbf{X}^{\dagger}(k)\,\mathbf{d}(k). \tag{8.20}$$

Returning to our interest in whether the computed results from a recursive least-squares method are usable, we turn to the cause for perturbations in a result that, in theory, should not affect the convergence properties. The amount of deviation in a computed quantity from its exact value, be it the filter weights or the filter *a posteriori* residuals, can be affected from two sources: the nature of the problem and the method chosen to solve it. For recursive least-squares, we are interested in computing a sequence of least-squares solutions $\mathbf{w}(k)$, $k = N, N+1, \ldots$, where, at time index k, we have the overdetermined system of equations $\mathbf{X}(k)\,\mathbf{w}(k) \approx \mathbf{d}(k)$ in which we are interested in determining either $\mathbf{w}(k)$, the least squares filter weights, or the minimal-valued filter residuals, $\boldsymbol{\varepsilon}(k) = \mathbf{d}(k) - \mathbf{X}(k)\,\mathbf{w}(k)$. Before the stability of a method can be assessed, however, the sensitivity of the problem to changes in the data must be studied. Clear terminology is needed for this distinction.

8.2.1 Conditioning, forward stability, and backward stability

Depending on our purposes, we may not be interested in the determination of the least-squares solution $\mathbf{w}(k)$ directly but only the least-squares residuals, $\varepsilon(k)$. This choice can, and in the case of recursive least-squares does, have an impact regarding the sensitivity of the problem to perturbations in the input data. In general, this sensitivity inherent to the problem is regarded as the *conditioning* of the problem and is the first step in assessing the quality of a method.

Definition 3. A problem is well-conditioned if small changes in the data invoke only small changes in the solution. Otherwise, the problem is considered to be ill-conditioned.

For example, for linear square systems of equations in which the number of unknowns equals the number of equations, $\mathbf{Ax} = \mathbf{b}$, for $\mathbf{A} \in \mathbb{R}^{n\times n}$ of full rank, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{n\times 1}$, the sensitivity of the solution, $\mathbf{x}$, to changes in the elements in $\mathbf{A}$ is measured through the condition number, $\kappa_2(\mathbf{A})$, of the matrix and is defined by

$$\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \, \|\mathbf{A}^{-1}\|_2. \tag{8.21}$$

This quantity gives a measurement to the proximity of $\mathbf{A}$ to a singular matrix. When $\mathbf{A}$ is nearly singular, a small change in its entries could have a very profound change in its solution $\mathbf{x}$, regardless of the method chosen to solve the problem!

Suppose that because of inaccuracies whose origin is purposely left vague, instead of determining $\mathbf{w}(k)$, we have computed $\hat{\mathbf{w}}(k)$. Whether it is acceptable to use $\hat{\mathbf{w}}(k)$ in place of $\mathbf{w}(k)$ is often not answered through direct measurement of the absolute or relative error, $\|\mathbf{w}(k) - \hat{\mathbf{w}}(k)\|_2$ or $\|\mathbf{w}(k) - \hat{\mathbf{w}}(k)\|_2 / \|\mathbf{w}(k)\|_2$, $\|\mathbf{w}(k)\|_2 \neq 0$, respectively (since we would simply use $\mathbf{w}(k)$!). However, through error analysis techniques it is sometimes possible to bound this quantity by parameters that are computable. When this is the case, the analysis bounding the absolute or relative error is termed a *forward* or *direct* error analysis. If the bounded quantity is small enough over general operating conditions, the computed solution $\hat{\mathbf{w}}(k)$ is deemed usable and the method employed to solve the problem is considered *forward stable*.

Definition 4. A method for solving $\mathbf{Xw} = \mathbf{d}$ is forward stable if it has a small forward error. That is, the computed result $\hat{\mathbf{w}}$ satisfies the condition that $\frac{\|\mathbf{w}-\hat{\mathbf{w}}\|_2}{\|\mathbf{w}\|_2}$ (with $\|\mathbf{w}\|_2 \neq 0$) is small.

Often, a forward analysis is too difficult to achieve useful results. When this is the case, an alternative approach for the error analysis, termed a *backward* error analysis, may be considered. For this, the computed solution $\hat{\mathbf{w}}(k)$ is interpreted as the exact solution to some other problem. Considering this new problem to be a perturbation of the original problem defines the perspective for the analysis. When the perturbed problem is near enough to the original problem, the computed solution is considered to be *backward* stable. When there are many possible problems in which $\hat{\mathbf{w}}(k)$ is the exact solution, the smallest perturbation from the original problem is chosen.

Definition 5. A method for solving a problem is backward stable if its backward error is small. That is, its computed solution is the exact solution to a slightly perturbed problem.

The analysis to measure the effects of finite-precision is often an application of the results found from the stability analysis of the method; the perturbations are defined to model the effects of computer arithmetic and finite-precision representation. At this level, the implementation of the method could include variations for handling the storage and numerical computations. The focus, then, would be to tailor the implementation of the method to exploit the capabilities and avoid the handicaps given by a computer representation environment.

To summarize the above, the attention is usually on the forward error since this translates directly to the usefulness of the computed solution. When a direct analysis to bound the forward error is not possible or too difficult, a useful means for interpreting and connecting the concepts of conditioning and forward and backward errors, when defined in a consistent way is [3, Chapter 1, p. 10],

$$\text{Forward Error} \lesssim (\text{Condition Number})\times(\text{Backward Error}).$$

For a well-conditioned problem, a small backward error implies a small forward error. But an ill-conditioned problem could lead to a misinterpretation of a small backward error since here there could still be a large forward error. A method is forward or backward stable if it produces a small forward or backward error, respectively. But a backward stable method does not necessarily produce a usable computed solution if the problem itself is sensitive to perturbations.

8.3 The Conditioning of the Least-Squares Problem

Signal processing problems, in particular least-squares problems, are recursively updated in time as new measurement data is received. On a more abstract setting, this can be formulated as a non-linear mapping that produces a sequence of vectors that (hopefully) approximate with increasing accuracy some desired solution. Determining the stability of this mapping, i.e., the study performed on the mapping to determine the degree of continuity under the effect of perturbations, is the initial goal of an analysis. Without additional specifics regarding continuity, the interpretation of a stability analysis may be left to the reader; an unnecessary consequence that should be avoided. To this end, it is often the case when analyzing the effects of perturbations to make the following distinction: Conditioning refers to the problem that is to be solved, while stability refers to either the method or its implementation as an algorithm. This means that it is possible to get a bad computed solution because of the nature of the problem or because of the manner chosen to solve it. An analysis assessing only the method/algorithm will be incomplete without an analysis of the problem.

8.3.1 The conditioning of the least-squares problem

Before we formally state the least-squares problem, we introduce the following terminology. The following requires distinct notation to represent exact values from those which have been affected by perturbations. We will denote a perturbation by the δ symbol and the perturbed quantity by inserting a ˜ above the symbol. For example, the presence of perturbations in the data matrix and the desired output will be denoted and defined by $\tilde{\mathbf{X}}(k) = \mathbf{X}(k) + \delta\mathbf{X}(k)$ and $\tilde{\mathbf{d}}(k) = \mathbf{d}(k) + \delta\mathbf{d}(k)$, respectively.

We now present two theorems from Golub and Wilkinson [4] that bound the effect of perturbations in $\mathbf{X}(k)$ and $\mathbf{d}(k)$ on the solutions to the least squares problem (8.15) and (8.16). For $\mathbf{X}(k)$ having full rank, we define its condition number κ_2 as

$$\kappa_2\big(\mathbf{X}(k)\big) \triangleq \frac{\sigma_1(k)}{\sigma_{N+1}(k)} \tag{8.22}$$

$$= \|\mathbf{X}(k)\|_2 \left\| \left[\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\right]^{-1} \mathbf{X}^{\mathrm{T}}(k) \right\|_2 . \tag{8.23}$$

From the singular value decomposition it follows that,

$$\kappa_2(\mathbf{X}(k))^2 \triangleq \|\mathbf{X}(k)\|_2^2 \left\| \left[\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\right]^{-1} \right\|_2 . \tag{8.24}$$

Theorem 1. *Let* $\mathbf{w}(k)$, $\boldsymbol{\varepsilon}(k)$, $\tilde{\mathbf{w}}(k)$, *and* $\tilde{\boldsymbol{\varepsilon}}(k)$ *satisfy*

$$\|\mathbf{X}(k)\,\mathbf{w}(k) - \mathbf{d}(k)\|_2 = \min_{\mathbf{w}} \|\boldsymbol{\varepsilon}(k)\|_2 \tag{8.25}$$

$$\|(\mathbf{X}(k) + \delta\mathbf{X}(k))\,\tilde{\mathbf{w}}(k) - (\mathbf{d}(k) + \delta\mathbf{d}(k))\|_2 = \min_{\tilde{\mathbf{w}}} \|\tilde{\boldsymbol{\varepsilon}}(k)\|_2 \tag{8.26}$$

where

$$\boldsymbol{\varepsilon}(k) = \mathbf{d}(k) - \mathbf{X}(k)\,\mathbf{w}(k), \textit{ and}$$
$$\tilde{\boldsymbol{\varepsilon}}(k) = \left[\mathbf{d} + \tilde{\mathbf{d}}(k)\right] - \left[\mathbf{X}(k) + \delta\mathbf{X}(k)\right]\tilde{\mathbf{w}}(k).$$

If $\delta_{\max}$ *is given by*

$$\delta_{\max} = \max\left\{ \frac{\|\delta\mathbf{X}(k)\|_2}{\|\mathbf{X}(k)\|_2}, \frac{\|\delta\mathbf{d}(k)\|_2}{\|\mathbf{d}(k)\|_2} \right\} < \frac{\sigma_{N+1}(k)}{\sigma_1(k)} \tag{8.27}$$

and $\sin(\theta)$ *by*

$$\sin(\theta) = \frac{\rho_{\mathbf{w}(k)}}{\|\mathbf{d}(k)\|_2} \neq 1, \tag{8.28}$$

where $\rho_{\mathbf{w}(k)} \triangleq \|\mathbf{X}(k)\,\mathbf{w}(k) - \mathbf{d}(k)\|_2$ is the minimal least squares residual, then

$$\frac{\|\tilde{\mathbf{w}}(k) - \mathbf{w}(k)\|_2}{\|\mathbf{w}(k)\|_2} \leq \delta_{max} \left\{ \frac{2\kappa_2(\mathbf{X}(k)}{\cos(\theta)} + \tan(\theta)\kappa_2(\mathbf{X}(k))^2 \right\} + \mathscr{O}(\delta_{\max}^2) \quad (8.29)$$

and

$$\frac{\|\tilde{\boldsymbol{\varepsilon}}(k) - \boldsymbol{\varepsilon}(k)\|_2}{\|\mathbf{d}(k)\|_2} \leq \delta_{\max}\,(1 + 2\kappa_2(\mathbf{X}(k)))\min(1, k-N) + \mathscr{O}(\delta_{\max}^2). \quad (8.30)$$

Theorem 1 tells us that the sensitivity of the least squares filter residuals, $\varepsilon(k)$, are proportional to the conditioning, $\kappa_2(\mathbf{X}(k))$. Comparatively, the sensitivity of the filter weights to perturbations are proportional to the square of the conditioning, $\kappa_2^2(\mathbf{X}(k))$. This result pertains to the nature of the problem being solved and is independent of the method employed to solve it.

Under the conditions of Theorem 1 with the data matrix $\mathbf{X}(k)$ having full rank and no assumed perturbations in $\mathbf{d}(k)$, the conditioning of the least-squares problem, $\kappa_{LS}(\mathbf{X}(k), \mathbf{d}(k))$ is defined as [5, Chapter 1],

Definition 6.

$$\kappa_{LS}(\mathbf{X}(k), \mathbf{d}(k)) = \kappa_2(\mathbf{X}(k)) \left(1 + \kappa_2(\mathbf{X}(k)) \frac{\|\boldsymbol{\varepsilon}(k)\|_2}{\|\mathbf{X}(k)\|_2 \, \|\mathbf{w}(k)\|_2} \right). \quad (8.31)$$

From (8.31) it is seen that the conditioning of the least-squares problem, that is, the sensitivity of the least-squares problem to perturbations in the data matrix, depends on the *a posteriori* residual and thus on the right-hand-side vector $\mathbf{d}(k)$.

8.3.2 Consistency, stability, and convergence

The recursive least-squares problem produces a sequence of solutions, $\mathbf{x}(k)$, $k = N+1, \ldots,$ and it is because of this that the issue of convergence is of interest. Specifically, even though in a stationary environment the recursive least-squares solution converges, perturbations may significantly alter these theoretical properties to the extent that the computed solution may not converge.

Suppose the computed solution, $\hat{\mathbf{w}}(k)$ at time index k, is a very good approximation to the true least-squares solution $\mathbf{w}(k)$. Since the next computed solution,

$\hat{\mathbf{w}}(k+1)$, involves $\hat{\mathbf{w}}(k)$, it is natural to question the usefulness for our definition of a stable method. After a large number of iterations, the accumulative effect from each approximate solution could have disastrous consequences regarding the notion of convergence.

To this end, there is a need for additional conditions on the method before the sequence of computed solutions can be guaranteed to converge to a good approximation of the desired solution, be it the filter weights (the least-squares solution) or the filter *a posteriori* error (residual). In a recursive environment, the update parameters are typically non-linearly interlaced. Abstractly, let the parameters used in the method's update scheme be denoted as a non-linear mapping $\{f : A \longrightarrow B\}$ where A denotes the set of input data and parameters at the current state and B denotes the computed solution as well as the updated parameters to be used in the next recursion. The mapping f is strongly associated to the least-squares problem, and as such inherits any restrictions such as matrix structure (symmetry, close-to-Toeplitz, positive definitiveness, etc.). Any perturbations to this mapping must not interfere with this association. We will formalize this, but first we need to define what we mean by an equicontinuous mapping.

Definition 7. The set $\{B\}$ is *admissible* if for all $\mathbf{a} \in A$ $\phi(\mathbf{a}) = \mathbf{b} \in B$ and $\mathbf{a}, \mathbf{b}$ are associated to a least-squares problem.

Definition 8. The mapping $\{f : A \longrightarrow B\}$ is *continuous at* $\mathbf{a}$ if for all $\zeta > 0$ there exists an $\eta > 0$, such that $\| \phi(\hat{\mathbf{a}}) - \phi(\mathbf{a}) \| \leq \zeta$ whenever $\| \hat{\mathbf{a}} - \mathbf{a} \| \leq \eta$.

If we are interested in the convergence properties of all members f_δ in a neighborhood of f then we need the notion of consistency in addition to continuity.

Definition 9. The mapping f is *consistent* if for some $\eta > 0$, and all $\hat{\mathbf{a}} = \mathbf{a} + \delta\mathbf{a}$, $\| \delta\mathbf{a} \| < \eta$, it holds that $\phi(\hat{\mathbf{a}}) = \hat{\mathbf{b}} \in B$, B admissible.

Definition 10. $\Phi_\delta = f_\delta^{-1}$ is called an *approximation to the inverse mapping* f^{-1} if and only if f_δ is consistent with respect to the mapping f.

Theorem 2. *For ϕ_δ consistent with respect to f, a sufficient condition for convergence is the equicontinuity of Φ_δ.*

The proof of Theorem 2 can be found in ([6] p. 10).

8.4 The Recursive QR Least-Squares Methods

The QR-decomposition relies on a process that will replace selected non-zero entries of a matrix or vector with zeros. When this process is performed using orthogonal matrices, desirable properties concerning numerical stability result. We review first a direct stability analysis for the full QR decomposition adaptive filtering analysis, and then observe that the sufficient conditions for stable behavior reduce to a form of backward consistency. We then examine fast least-squares algorithms based on the QR decomposition. Although a direct stability analysis is considerably more complicated (if not intractable), backward consistency conditions can be obtained in a simple form, and relate to convergence via Theorem 2.

8.4.1 Full QR decomposition adaptive algorithm

For the full QR decomposition adaptive filtering algorithm, the time recursions absorb a new input vector $\mathbf{x}(k)$ and reference sample $d(k)$ at each time instant. The $N+1$ elements of $\mathbf{x}(k)$, however, need not derive from a tapped delay line, and indeed might derive from $N+1$ separate channels or sensor outputs, for example.

The basic update recursion for the triangular array appears as:

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \mathbf{U}(k) \end{bmatrix} = \mathbf{Q}_{\theta}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{U}(k-1) \end{bmatrix}, \qquad \mathbf{U}(-1) = \mathbf{U}_{-1}, \tag{8.32}$$

in which $\mathbf{U}_{-1}$ is the initial condition on the triangular array (typically the zero array or a small multiple of the identity), and the orthogonal matrix $\mathbf{Q}_{\theta}(k)$ is chosen to null the entries of the top row of the array. In practice, roundoff errors will also contaminate the updated triangular array; if we denote by $\tilde{\mathbf{U}}(k)$ the finite-precision representation of the triangular array, the finite-precision counterpart of the basic update recursion may be written as

$$\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \tilde{\mathbf{U}}(k) \end{bmatrix} = \mathbf{Q}_{\tilde{\theta}}(k) \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\tilde{\mathbf{U}}(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \delta\mathbf{U}(k) \end{bmatrix}, \qquad \tilde{\mathbf{U}}(-1) = \mathbf{U}_{-1}, \tag{8.33}$$

in which $\mathbf{Q}_{\tilde{\theta}}(k)$ is the product of rotations determined from the (finite-precision) triangularization of $\tilde{\mathbf{U}}(k)$, and the second term on the right-hand-side accounts for the difference between the first term, were it calculated in exact arithmetic, and the actual stored result on the left-hand side. We assume also that the initial condition $\mathbf{U}_{-1}$ admits an exact representation in finite-precision.

We examine first a direct stability analysis and then illustrate the connection with backward consistency concepts.

By squaring up either side of (8.32), we recover the familiar recursion

$$\begin{aligned}\mathbf{R}(k) &= [\mathbf{0}\;\; \mathbf{U}^{\mathrm{T}}(k)]\begin{bmatrix}\mathbf{0}^{\mathrm{T}}\\ \mathbf{U}(k)\end{bmatrix}\\ &= [\mathbf{x}(k)\;\; \lambda^{1/2}\mathbf{U}^{\mathrm{T}}(k-1)]\underbrace{\mathbf{Q}_{\theta}^{\mathrm{T}}(k)\,\mathbf{Q}_{\theta}(k)}_{\mathbf{I}}\begin{bmatrix}\mathbf{x}^{\mathrm{T}}(k)\\ \lambda^{1/2}\mathbf{U}(k-1)\end{bmatrix}\\ &= \lambda\,\mathbf{U}^{\mathrm{T}}(k-1)\,\mathbf{U}(k-1)+\mathbf{x}(k)\,\mathbf{x}^{\mathrm{T}}(k)\\ &= \lambda\,\mathbf{R}(k-1)+\mathbf{x}(k)\,\mathbf{x}^{\mathrm{T}}(k), \qquad \mathbf{R}(-1)=\mathbf{U}_{-1}^{\mathrm{T}}\mathbf{U}_{-1}.\end{aligned} \tag{8.34}$$

A similar squaring-up operation applied to (8.33) gives

$$\tilde{\mathbf{R}}(k)=\lambda\,\tilde{\mathbf{R}}(k-1)+\mathbf{x}(k)\,\mathbf{x}^{\mathrm{T}}(k)+\delta\mathbf{R}(k), \qquad \tilde{\mathbf{R}}_{-1}=\mathbf{U}_{-1}^{\mathrm{T}}\mathbf{U}_{-1}, \tag{8.35}$$

in which

$$\begin{aligned}\delta\mathbf{R}(k) &= [\mathbf{x}\;\; \lambda^{1/2}\tilde{\mathbf{U}}^{\mathrm{T}}(k-1)]\,\mathbf{Q}_{\tilde{\theta}}^{\mathrm{T}}(k)\begin{bmatrix}\mathbf{0}^{\mathrm{T}}\\ \delta\mathbf{U}(k)\end{bmatrix}\\ &\quad+[\mathbf{0}\;\; \delta\mathbf{U}^{\mathrm{T}}(k)]\,\mathbf{Q}_{\tilde{\theta}}(k)\begin{bmatrix}\mathbf{x}^{\mathrm{T}}(k)\\ \lambda^{1/2}\tilde{\mathbf{U}}(k-1)\end{bmatrix}\\ &\quad+[\delta\mathbf{U}(k)]^{\mathrm{T}}\delta\mathbf{U}(k)\end{aligned} \tag{8.36}$$

accounts for the roundoff errors injected at iteration k, once expressed in the covariance domain. The difference $\tilde{\mathbf{R}}(k)-\mathbf{R}(k)$ thus adheres to the recursion

$$[\tilde{\mathbf{R}}(k)-\mathbf{R}(k)]=\lambda\,[\tilde{\mathbf{R}}(k-1)-\mathbf{R}(k-1)]+\delta\mathbf{R}(k), \qquad \tilde{\mathbf{R}}(-1)-\mathbf{R}(-1)=\mathbf{0}. \tag{8.37}$$

This recursion admits the explicit solution

$$[\tilde{\mathbf{R}}(k)-\mathbf{R}(k)]=\sum_{n=0}^{k}\lambda^{k-n}\,\delta\mathbf{R}(n). \tag{8.38}$$

Now, if the finite-precision errors are bounded at each iteration, meaning that $\|\delta\mathbf{R}(n)\|\leq B<\infty$ for some constant B, where $\|\cdot\|$ denotes any valid matrix norm, then the difference $\tilde{\mathbf{R}}(k)-\mathbf{R}(k)$ is easily bounded as follows:

$$\begin{aligned}\|\tilde{\mathbf{R}}(k)-\mathbf{R}(k)\| &= \left\|\sum_{n=0}^{k}\lambda^{n-k}\,\delta\mathbf{R}(n)\right\| \\ &\le \sum_{n=0}^{k}\lambda^{n-k}\,\|\delta\mathbf{R}(n)\| \\ &\le B\sum_{n=0}^{k}\lambda^{n-k} \\ &\le B\,\frac{1}{1-\lambda}, \qquad \text{for all } k, \end{aligned} \tag{8.39}$$

in which the final inequality is valid for $0<\lambda<1$. We observe, as expected, that a smaller value of B (corresponding to higher precision in the calculations) results in $\tilde{\mathbf{R}}(k)$ tracking its exact-arithmetic counterpart $\mathbf{R}(k)$ more closely. Conversely, choosing λ closer to one results in a less favorable distance bound between $\tilde{\mathbf{R}}(k)$ and $\mathbf{R}(k)$. This is because values of λ closer to one induce a greater effective memory of the algorithm, so that a given arithmetic error $\delta\mathbf{R}(n)$ will linger more prominently through successive iterations.

So what does this bound imply about the difference $\tilde{\mathbf{U}}(k)-\mathbf{U}(k)$? To answer this, we note that $\mathbf{U}(k)$ [respectively, $\tilde{\mathbf{U}}(k)$] is a Cholesky factor of $\mathbf{R}(k)$ [respectively, $\tilde{\mathbf{R}}(k)$]; the Cholesky factor becomes unique once we specify whether it is upper or lower triangular, with positive elements along the diagonal.[1] We note also that the Cholesky factor is a continuous function of a positive definite matrix argument (e.g., [7]). Thus, if $\mathbf{R}(k)$ and $\tilde{\mathbf{R}}(k)$ remain positive definite (to be addressed shortly), there exists a constant c such that the uniform bound $\|\tilde{\mathbf{R}}(k)-\mathbf{R}(k)\|\le B/(1-\lambda)$ for all k implies that

$$\|\tilde{\mathbf{U}}(k)-\mathbf{U}(k)\| \le \frac{cB}{1-\lambda}, \quad \text{for all } k. \tag{8.40}$$

Now, positive definiteness of $\mathbf{R}(k)$ and $\tilde{\mathbf{R}}(k)$ (which are never explicitly formed), reduces to a full rank condition on $\mathbf{U}(k)$ and $\tilde{\mathbf{U}}(k)$. In view of the triangular structure, this reduces, in turn, to either matrix having non-zero elements in each diagonal position, which is rather easily checked.

To summarize, the following three conditions:

1. Bounded arithmetic errors: $\|\delta\mathbf{U}(n)\|\le cB$ for all n [giving the numerator in (8.40)];
2. Forgetting factor strictly less than one: $0<\lambda<1$;
3. Full rank data: all diagonal elements of $\tilde{\mathbf{U}}(k)$ remain positive;

[1] One can also define a Cholesky factor with respect to the anti-diagonal, as in Chapter 3. We revert to the more conventional approach of triangular arrays with respect to the main diagonal in this chapter, so that the various statements to follow are more consistent with the cited references. The conclusions concerning stability carry over to algorithms based on an anti-diagonal Cholesky factorization as well, although the notations to describe intermediate quantities would change somewhat.

are sufficient to ensure that the finite-precision representation $\tilde{\mathbf{U}}(k)$ remains within a bounded distance from its exact arithmetic counterpart $\mathbf{U}(k)$ as k increases. The first two conditions are under the designer's control; the third condition is data dependent, and is usually captured as a *persistence of excitation* constraint. The formal definition, in the present context, takes the following form: Let $\sigma_1(k)$ and $\sigma_{N+1}(k)$ be the largest and smallest singular values, respectively, of the data matrix $\mathbf{X}(k)$. The data which build $\mathbf{X}(k)$ are persistently exciting of order $N+1$ if these extremal singular values are uniformly bounded, meaning that there exists positive constants κ_a and κ_b such that

$$\sigma_1(k) \leq \kappa_a < \infty, \quad \text{and} \quad \sigma_{N+1}(k) \geq \kappa_b > 0 \qquad \text{for all } k > k_0 \tag{8.41}$$

where k_0 is some starting time. These inequalities imply that the condition number of $\mathbf{X}(k)$ is bounded:

$$\frac{\sigma_1(k)}{\sigma_{N+1}(k)} \leq \frac{\kappa_a}{\kappa_b} < \infty, \qquad \text{for all } k > k_0 \tag{8.42}$$

and thus that the least-squares problem remains well-posed. Note that the ratio κ_a/κ_b can be shown equal to the quantity $\kappa_2(\mathbf{X}(k))$ from (8.23).

Fortunately, the persistence of excitation condition is easily checked in the orthogonal triangularization procedure:

Result 1 *If the input data are bounded, then persistent excitation holds if and only if there exists a constant $c > 0$ for which*

$$\cos\theta_i(l) \geq c, \qquad \textit{for all } i \textit{ and } l.$$

For the verification, we note that the triangular matrix $\mathbf{U}(\cdot)$ becomes rank deficient if and only if at least one of its diagonal elements vanishes. Thus let $\mathbf{U}_{ii}(l)$ denote the ith diagonal element of the Cholesky factor at any time l. The formula for the rotation angles which achieve the triangularization at an arbitrary time instant l is

$$\cos\theta_i(l) = \frac{\lambda^{1/2}\mathbf{U}_{ii}(l-1)}{\mathbf{U}_{ii}(l)}, \qquad i = 0, 1, \ldots, N-1, \tag{8.43}$$

which shows that $\cos\theta_i(l) = 0$ for some i if and only if $\mathbf{U}_{ii}(l-1) = 0$, i.e., if and only if $\mathbf{U}(l-1)$ is rank deficient. Thus if $\mathbf{U}(l-1)$ has full rank for all l, then $\cos\theta_i(l) > 0$ for all i and l, which is to say that $\cos\theta_i(l) = 0$ for some i and l if and only if the smallest singular value $\sigma_{N+1}(l-1) = 0$. By continuity arguments, bounding the smallest singular value $\sigma_{N+1}(l-1)$ away from zero for all l must likewise bound $\cos\theta_i(l)$ away from zero for all i and l, and vice versa.

To treat the joint-process portion, recall that the basic structure takes the form of the orthogonal filter

$$\begin{bmatrix} e_{q_1}(k) \\ \mathbf{d}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_\theta(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix}. \tag{8.44}$$

In a practical implementation that includes roundoff errors, the recursion instead takes the form

$$\begin{bmatrix} \tilde{e}_{q_1}(k) \\ \tilde{\mathbf{d}}_{q_2}(k) \end{bmatrix} = \mathbf{Q}_{\tilde{\theta}}(k) \begin{bmatrix} d(k) \\ \lambda^{1/2}\tilde{\mathbf{d}}_{q_2}(k-1) \end{bmatrix} + \begin{bmatrix} \delta e(k) \\ \delta\mathbf{d}_{q_2}(k) \end{bmatrix}, \tag{8.45}$$

in which $\mathbf{Q}_{\tilde{\theta}}(k)$ is the product of rotations determined from the (finite-precision) triangularization of $\tilde{\mathbf{U}}(k)$. Observe that both $\tilde{\mathbf{d}}_{q_2}(k)$ and $\tilde{\mathbf{d}}_{q_2}(k-1)$ occur in this equation, indicative of a feedback loop involving the state variables $\tilde{\mathbf{d}}_{q_2}(\cdot)$. As such, roundoff errors accumulated in $\tilde{\mathbf{d}}_{q_2}(k)$ will propagate in time. Similar to the development above, provided $\lambda < 1$, the feedback loop may be shown exponentially stable, inducing bounded error growth provided the injected roundoff error (modeled by the term $\delta\mathbf{d}_{q_2}(k)$ above) is bounded at each time instant.

For the deeper question of whether the computed $\tilde{\mathbf{d}}_{q_2}(k)$ has relevance to the underlying least-squares problem, return to the unperturbed system (8.44) and partition the transition matrix $\mathbf{Q}_\theta(k)$ as

$$\mathbf{Q}_\theta(k) = \begin{bmatrix} \gamma(k) & \mathbf{g}^{\mathrm{T}}(k) \\ \mathbf{f}(k) & \mathbf{E}(k) \end{bmatrix}, \tag{8.46}$$

in which $\mathbf{E}(k)$ is $(N+1)\times(N+1)$, $\mathbf{f}(k)$ is $(N+1)\times 1$, $\mathbf{g}(k)$ is $(N+1)\times 1$, and $\gamma(k)$ is a scalar. The state equation may then be solved "backwards in time" as

$$\begin{aligned} \mathbf{d}_{q_2}(k) = {} & \lambda^{(l+1)/2}\,\boldsymbol{\Phi}(k,k-l)\,\mathbf{d}_{q_2}(k-l) \\ & + \underbrace{\begin{bmatrix}\mathbf{f}(k) & \mathbf{E}(k)\mathbf{f}(k-1) & \boldsymbol{\Phi}(k,k-1)\mathbf{f}(k-2) & \cdots & \boldsymbol{\Phi}(k,k-l+1)\mathbf{f}(k-l)\end{bmatrix}}_{\mathscr{C}(k,k-l)} \\ & \times \begin{bmatrix} d(k) \\ \lambda^{1/2}d(k-1) \\ \lambda\, d(k-2) \\ \vdots \\ \lambda^{l/2}d(k-l) \end{bmatrix}, \end{aligned} \tag{8.47}$$

in which

$$\boldsymbol{\Phi}(k,k-l) \triangleq \begin{cases} \mathbf{E}(k)\mathbf{E}(k-1)\cdots\mathbf{E}(k-l+1), & l \geq 1; \\ \mathbf{I}, & l = 0; \end{cases} \tag{8.48}$$

is the *state transition matrix* [8] from time $k-l$ to k, and $\mathscr{C}(k,k-l)$ is the *controllability matrix* [8] for the system, over the same time window. The system is said to be *uniformly controllable* [9] provided there exists a window length L, and constants

$a > 0$ and $b < \infty$, for which the Gramian of the controllability matrix $\mathscr{C}(k,k-L)$ is bounded and of full rank:

$$a\mathbf{I} \leq \mathscr{C}(k,k-L)\mathscr{C}^{\mathrm{T}}(k,k-L) \leq b\mathbf{I}, \qquad \text{for all } k \geq k_0, \tag{8.49}$$

where k_0 is some starting time. [Observe that $\sqrt{a}$ and $\sqrt{b}$ bound the extremal singular values of $\mathscr{C}(k,k-L)$.] Since the controllability matrix $\mathscr{C}(k,k-L)$ has dimensions $(N+1) \times (L+1)$, clearly we must have $L \geq N$ if the full rank condition is to hold.

The relevance of this condition is that, when satisfied, an arbitrary configuration for $\tilde{\mathbf{d}}_{q_2}(k)$ can be reached by an appropriate choice of the (exponentially weighted) reference vector

$$[d(k), \lambda^{1/2}d(k-1), \ldots, \lambda^{L/2}d(k-L)]^{\mathrm{T}}. \tag{8.50}$$

As such, even with numerical errors accumulated in $\tilde{\mathbf{d}}_{q_2}(k)$, the values so obtained may be considered the *exact* state produced by some reference vector, as required of admissibility defined in Section 8.3.2.

Now, since the matrix $\mathbf{Q}_{\tilde{\theta}}(k)$ is orthogonal for each k, one may show (e.g., [10])

$$\boldsymbol{\Phi}(k,k-L)\,\boldsymbol{\Phi}^{\mathrm{T}}(k,k-L) + \mathscr{C}(k,k-L)\mathscr{C}^{\mathrm{T}}(k,k-L) = \mathbf{I}, \qquad \text{for all } k \text{ and } L, \tag{8.51}$$

so that

$$\begin{aligned}\mathscr{C}(k,k-L)\mathscr{C}^{\mathrm{T}}(k,k-L) &= \mathbf{I} - \boldsymbol{\Phi}(k,k-L)\,\boldsymbol{\Phi}^{\mathrm{T}}(k,k-L)\\ &\leq \mathbf{I}\end{aligned} \tag{8.52}$$

providing automatic satisfaction of the upper bound from (8.49) using $b = 1$.

For the lower bound, we claim:

Result 2 *With $L = N$, the joint-process section is uniformly controllable provided there exists a constant $c > 0$ for which*

$$\cos\theta_i(l) \geq c > 0, \qquad \textit{for all } i \textit{ and } l \geq k_0. \tag{8.53}$$

The proof amounts to calculating the square matrix $\mathscr{C}(k,k-N)$ and observing that it becomes rank deficient if and only if $\cos\theta_i(l) = 0$ for any order index i and any time index $k-L \leq l \leq k$. By continuity arguments, bounding $\cos\theta_i(l)$ away from zero must also bound the smallest singular value of $\mathscr{C}(k,k-L)$ away from zero, proving existence of a constant a fulfilling the lower bound from (8.49). In view of result 1 above, we see that persistence of excitation is sufficient to ensure backward consistency of the full QR algorithm, and that this in turn suffices for bounded error growth.

8.5 Fast QR Algorithms

Perhaps the earliest fast QR decomposition adaptive filtering algorithm (where "fast" means having computational complexity that scales linearly with the filter order N) was devised by Cioffi [11]. Somewhat more coherent developments of two varieties of such fast algorithms were obtained soon thereafter by Proudler et al. [12, 13]: The first featured order recursions in both ascending and descending order and was later rederived in [14], while the second exhibited all order recursions in ascending order, and is better known as the QRD lattice algorithm [15]. A traditional lattice-based derivation of this latter algorithm is found also in Ling [16].

The time recursion of the prediction section of a fast least-squares algorithm is a dynamic system of the form

$$\boldsymbol{\xi}(k) = f\big[\boldsymbol{\xi}(k-1), x(k)\big], \qquad \boldsymbol{\xi}(-1) = \boldsymbol{\xi}_{-1}, \tag{8.54}$$

in which $\boldsymbol{\xi}(\cdot)$ is the state vector (collecting all the quantities that must be stored at each iteration to propagate the solution in time), the map $f[\cdot,\cdot]$ accounts for the update equations, and $\boldsymbol{\xi}_{-1}$ is the initial condition on the state vector. In the full QR algorithm reviewed above, the state is simply $\boldsymbol{\xi}(k) = \mathbf{U}(k)$, and the map $f[\cdot,\cdot]$ performs the orthogonal triangularization to update $\mathbf{U}(k-1)$ to $\mathbf{U}(k)$. Fast least-squares algorithms exploit the shift structure of the data matrix $\mathbf{X}(k)$, allowing a more compact representation of the state and reducing the number of operations in the update equations $f[\cdot,\cdot]$ to a quantity linear in N. Unlike the full QR algorithm, however, the update equations cannot readily be rewritten as a linear recurrence relation, thus complicating considerably any attempt at a direct error analysis.

Backward consistency concepts, on the other hand, are still useful in assessing error propagation properties in such fast least-squares algorithms [17–19]. Introduce the set of reachable states in exact arithmetic, i.e., the set of state vector configurations that may be reached as the input sequence $x(0)$, $x(1)$, $\ldots$, $x(k)$ varies over $\mathbb{R}^k$, and the initial condition $\boldsymbol{\xi}_{-1}$ varies over all "valid" initial conditions. The set of valid initial conditions is best thought of as the set of state vector orientations $\boldsymbol{\xi}(-1)$ that can be deposited by some past input sequence $x(-1)$, $x(-2)$, $x(-3)$, $\ldots$, that may potentially extend infinitely into the past.

Now, in finite-precision, the actual prediction section behaves as

$$\tilde{\boldsymbol{\xi}}(k) = f\big[\tilde{\boldsymbol{\xi}}(k-1), x(k)\big] + \delta\boldsymbol{\xi}(k), \tag{8.55}$$

in which $\delta\boldsymbol{\xi}(k)$ accounts for the roundoff errors injected in the state vector at time k. Provided the computed state vector $\tilde{\boldsymbol{\xi}}(k)$ remains within the set of reachable states, it is indistinguishable from the exact state produced by a different input sequence $\tilde{x}(0)$, $\tilde{x}(1)$, $\ldots$, $\tilde{x}(k)$ using possibly a different (but valid) initial condition $\tilde{\boldsymbol{\xi}}_{-1}$. If we now drive the perturbed system and its exact arithmetic counterpart (8.54) with the same future sequence $x(k+1)$, $x(k+2)$, $\ldots$, and allow both systems to evolve *without further arithmetic errors*, the perturbed trajectory $\tilde{\boldsymbol{\xi}}(\cdot)$ will return to the true trajectory $\boldsymbol{\xi}(\cdot)$ provided $\lambda < 1$, and the future data is persistently exciting. This is because, in the absence of arithmetic errors, either system is but a rewriting of the full QR algorithm using a different initial condition $\mathbf{U}(k)$ or $\tilde{\mathbf{U}}(k)$ at time k, and fed with the same input sequence from time k forward. With $\lambda < 1$, the algorithm forgets its initial condition as time evolves. This basic argument shows that, subject to consistency of the state vector, a least-squares algorithms (fast or full) will enjoy stable error propagation [17, 18], in which the error propagation experiment assumes no further arithmetic errors after a perturbation is injected. (This stable error propagation was first observed in [20] from a direct analysis, although without connections to backward consistency concepts). As such, explosive error growth must be preceded by an inconsistent value arising numerically in the state vector, a condition therefore to be avoided if at all possible. We should note that the error propagation experiment described here assumes a single perturbation followed by exact arithmetic calculations. In practice, roundoff errors are injected at each time-step, requiring due attention to error accumulation (as we did for the full QR algorithm above). Although stable propagation of a single error is a necessary condition for bounded error accumulation, it need not be sufficient. Nonetheless, provided the error propagation properties are exponentially stable, which generically holds provided $\lambda < 1$ and the input data are persistently exciting, bounded error growth may be expected to hold, at least for sufficiently fine numerical resolution [21]. Stronger results, in the form of Theorem 2 from Section 8.3.2, are also applicable here ([6] p. 10).

We first review the data structure applicable to fast least-squares algorithms and the data consistency properties which stem from these. For notational simplicity, we first set $\lambda = 1$ and denote the resulting data matrix as $\mathbf{X}_1(k)$ (where the subscript 1 emphasizes that $\lambda = 1$); this assumes a pre-windowed Hankel structure:

$$\mathbf{X}_1(k) = \begin{bmatrix} x(k) & x(k-1) & \cdots & x(k-N) \\ x(k-1) & x(k-2) & \cdots & x(k-N-1) \\ \vdots & \iddots & \iddots & \vdots \\ x(N) & \cdots & x(1) & x(0) \\ \vdots & \iddots & x(0) & 0 \\ x(1) & \iddots & \iddots & \vdots \\ x(0) & 0 & \cdots & 0 \end{bmatrix}. \tag{8.56}$$

If we introduce the correlation lags

$$r_n = \sum_{i=0}^{k-n} x(i)\,x(i+n), \qquad n = 0, 1, \ldots, N, \tag{8.57}$$

and rename the most recent input samples as

$$q_1 = x(k), \quad q_2 = x(k-1), \quad \ldots \quad q_N = x(k-N+1), \tag{8.58}$$

then for any k, the correlation matrix takes the structured form

$$\begin{aligned}\mathbf{R}_1(k) &= \mathbf{X}_1^{\mathrm{T}}(k)\,\mathbf{X}_1(k)\\ &= \begin{bmatrix} r_0 & r_1 & \cdots & r_N \\ r_1 & r_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_1 \\ r_N & \cdots & r_1 & r_0 \end{bmatrix} - \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & .\cdot{}^{\cdot} & 0 & q_1 \\ 0 & .\cdot{}^{\cdot} & .\cdot{}^{\cdot} & \vdots \\ 0 & q_1 & \cdots & q_N \end{bmatrix}^2 .\end{aligned} \tag{8.59}$$

This is a symmetric Toeplitz matrix minus the square of a triangular Hankel matrix.

For the case $\lambda < 1$, let

$$\boldsymbol{\Lambda}(k) = \mathrm{diag}(1, \lambda^{1/2}, \lambda, \ldots, \lambda^{k/2}) \tag{8.60}$$

so that the data matrix may be written as

$$\mathbf{X}(k) = \boldsymbol{\Lambda}(k)\,\mathbf{X}_1(k), \tag{8.61}$$

in terms of the (unweighted) data matrix $\mathbf{X}_1(k)$ from (8.56). Let now

$$\mathbf{L} = \mathrm{diag}(1, \lambda^{1/2}, \ldots, \lambda^{N/2}). \tag{8.62}$$

Because $\mathbf{X}_1(k)$ is a Hankel matrix, we may observe the identity

$$\boldsymbol{\Lambda}(k)\,\mathbf{X}_1(k) = \overline{\mathbf{X}}(k)\,\mathbf{L}^{-1} \tag{8.63}$$

in which $\overline{\mathbf{X}}(k)$ is a Hankel matrix akin to (8.56) but built from the exponentially weighted sequence

$$\bar{x}(k-n) = \lambda^{n/2}\,x(k-n), \qquad n = 0, 1, \ldots, k. \tag{8.64}$$

As such, a transformed correlation matrix becomes

$$\mathbf{R}_1(k) \triangleq \mathbf{L}\,\mathbf{R}(k)\,\mathbf{L} = \mathbf{L}\,\mathbf{X}^{\mathrm{T}}(k)\,\mathbf{X}(k)\,\mathbf{L} = \overline{\mathbf{X}}^{\mathrm{T}}(k)\,\overline{\mathbf{X}}(k) \tag{8.65}$$

which is the Gramian of a pre-windowed Hankel matrix $\overline{\mathbf{X}}(k)$. It may thus be written as a structured matrix as in (8.59), in which r_n and q_n are now defined from the (exponentially weighted) sequence $\bar{x}(n)$ from (8.64). This "trick" will allow us to examine the data consistency independently of the value of λ. We should emphasize that the error accumulation effects, however, will still vary with λ, as illustrated in the analysis of the full QR algorithm above.

The basic data consistency question for fast algorithms may thus be summarized as follows:

- Given a correlation matrix $\mathbf{R}_1(k) = \mathbf{L}\,\mathbf{R}(k)\,\mathbf{L}$, under what conditions can we find a pre-windowed Hankel matrix, call it $\overline{\mathbf{X}}(n)$, for which $\mathbf{R}(k) = \overline{\mathbf{X}}^{\mathrm{T}}(k)\,\overline{\mathbf{X}}(k)$?
- Given the stored variables at time k in the prediction section of a fast least-squares algorithm, under what conditions can they be considered the exact values obtained from some input sequence?

For the first query, clearly $\mathbf{R}_1(k)$ must be positive semi-definite, and assume the structured form illustrated in (8.59) above. Is this sufficient as well? The following result was first obtained in [22] in the context of digital filter design, and developed in greater detail in [23–27]: [2]

Result 3 *The covariance matrix* $\mathbf{R}_1(k) = \mathbf{L}\mathbf{X}^T(k)\,\mathbf{X}(k)\,\mathbf{L}$ *may be factored as*

$$\mathbf{R}_1(k) = \overline{\mathbf{X}}^T\overline{\mathbf{X}}, \tag{8.66}$$

with $\overline{\mathbf{X}}$ *a pre-windowed Hankel matrix, provided* $\mathbf{R}_1(k)$ *is positive definite and assumes the "Toeplitz minus squared Hankel" structure illustrated in (8.59).*

In fact, when $\mathbf{R}_1(k)$ is positive definite, there are infinitely many pre-windowed Hankel matrices $\overline{\mathbf{X}}$ fulfilling the factorization; they may all be parameterized via inverse scattering constructions [26, 27]. We should emphasize that the number of rows in any such factor $\overline{\mathbf{X}}$ is not easily controlled in the constructive procedure behind this result [27]; if we constrain the number of rows in $\overline{\mathbf{X}}$, the factorization problem is considerably more difficult. In practice, the number of rows of $\overline{\mathbf{X}}$ is not of immediate concern to the error analysis.

We review next the constructive procedure behind past input reconstruction, adapted from [27], and then examine consistency issues in fast QR adaptive filters.

8.5.1 Past input reconstruction

Here we review the procedure for factoring a structured covariance matrix as in (8.59) into a Hankel data matrix $\overline{\mathbf{X}}$. Let $\mathbf{Z}$ be the shift matrix with ones on the subdiagonal and zeros elsewhere. The displacement structure [28] of the covariance matrix from (8.59) becomes

[2] The original claim from [22] was that such a factorization will exist even when $\mathbf{R}_1(k)$ is positive semi-definite, and singular. This is not true in general [26, 27], unless a certain supplementary condition is satisfied as well. We shall sidestep this technical difficulty by focusing on the positive definite case in what follows.

$$\begin{aligned}\mathbf{R}_1 - \mathbf{Z}\mathbf{R}_1\mathbf{Z}^{\mathrm{T}} &= \begin{bmatrix} r_0 & r_1 & \cdots & r_N \\ r_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r_N & 0 & \cdots & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ q_1 \\ \vdots \\ q_N \end{bmatrix} [\cdot]^{\mathrm{T}} \\ &= \begin{bmatrix} \sqrt{r_0} \\ r_1/\sqrt{r_0} \\ \vdots \\ r_N/\sqrt{r_0} \end{bmatrix} [\cdot]^{\mathrm{T}} - \begin{bmatrix} 0 \\ r_1/\sqrt{r_0} \\ \vdots \\ r_N/\sqrt{r_0} \end{bmatrix} [\cdot]^{\mathrm{T}} - \begin{bmatrix} 0 \\ q_1 \\ \vdots \\ q_N \end{bmatrix} [\cdot]^{\mathrm{T}}, \end{aligned} \tag{8.67}$$

where "[·]" means "repeat the previous vector". From the three "generator vectors" [29] so exposed, create the $3 \times (N+1)$ array,

$$\mathbf{G} = \begin{bmatrix} \sqrt{r_0} & r_1/\sqrt{r_0} & r_2/\sqrt{r_0} & \cdots & r_N/\sqrt{r_0} \\ 0 & q_1 & q_2 & \cdots & q_N \\ 0 & r_1/\sqrt{r_0} & r_2/\sqrt{r_0} & \cdots & r_N/\sqrt{r_0} \end{bmatrix}, \tag{8.68}$$

and iterate the following procedure:

1. Shift the first row of the array one position to the right:

$$\mathbf{G} \xrightarrow{\mathbf{Z}} \begin{bmatrix} 0 & \sqrt{r_0} & r_1/\sqrt{r_0} & \cdots & r_{N-1}/\sqrt{r_0} \\ 0 & q_1 & q_2 & \cdots & q_N \\ 0 & r_1/\sqrt{r_0} & r_2/\sqrt{r_0} & \cdots & r_N/\sqrt{r_0} \end{bmatrix}. \tag{8.69}$$

2. Choose a hyperbolic rotation to annihilate the second element of the first non-zero column. In the first pass, this appears as

$$\begin{aligned}&\begin{bmatrix} 1/\cos\theta_0 & \sin\theta_0/\cos\theta_0 & 0 \\ \sin\theta_0/\cos\theta_0 & 1/\cos\theta_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & \sqrt{r_0} & r_1/\sqrt{r_0} & \cdots & r_{N-1}/\sqrt{r_0} \\ 0 & q_1 & q_2 & \cdots & q_N \\ 0 & r_1/\sqrt{r_0} & r_2/\sqrt{r_0} & \cdots & r_N/\sqrt{r_0} \end{bmatrix} \\ &\qquad = \begin{bmatrix} 0 & y_1 & \times & \cdots & \times \\ 0 & 0 & \times & \cdots & \times \\ 0 & r_1/\sqrt{r_0} & r_2/\sqrt{r_0} & \cdots & r_N/\sqrt{r_0} \end{bmatrix}, \end{aligned} \tag{8.70}$$

in which $y_1 = \sqrt{r_0 - q_1^2}$ and $\sin\theta_0 = -q_1/\sqrt{r_0}$.

3. Choose a hyperbolic rotation to annihilate the third element of the first non-zero column. In the first pass, this appears as

$$\begin{bmatrix} 1/\cos\phi_0 & 0 & \sin\phi_0/\cos\phi_0 \\ 0 & 1 & 0 \\ \sin\phi_0/\cos\phi_0 & 0 & 1/\cos\phi_0 \end{bmatrix} \times \begin{bmatrix} 0 & y_1 & \times & \cdots & \times \\ 0 & 0 & \times & \cdots & \times \\ 0 & r_1/\sqrt{r_0} & r_2/\sqrt{r_0} & \cdots & r_N/\sqrt{r_0} \end{bmatrix}$$
$$= \begin{bmatrix} 0 & y_2 & \times & \cdots & \times \\ 0 & 0 & \times & \cdots & \times \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}, \qquad (8.71)$$

in which $y_2 = \sqrt{y_1^2 - (r_1^2/r_0)}$ and $\sin\phi_0 = -(r_1/\sqrt{r_0})/y_1$.

4. Replace **G** with the resulting array from (8.71), and iterative the above procedure a further $N-1$ times to eliminate all the elements of the second and third rows.

The above procedure will successfully terminate, and yield angles satisfying

$$|\sin\theta_n| < 1 \quad \text{and} \quad |\sin\phi_n| < 1, \qquad \text{for all } n, \qquad (8.72)$$

if and only if the matrix $\mathbf{R}_1$ is positive definite [29].

A flow graph of the basic array operations appears as Figure 8.1, in which "z" denotes a right shift operation. Imagine now changing the flow direction of the lower two branches of the flow graph, to obtain Figure 8.2. In doing so, each hyperbolic rotation is converted to a planar (or orthogonal) rotation. We may now terminate the right-hand side of the figure by a lossless load $\mathbf{S}_L(z)$, where lossless here means:

- $\mathbf{S}_L(z)$ is analytic in $|z| < 1$, thus admitting a convergent series expansion

$$\mathbf{S}_L(z) = \sum_{n=0}^{\infty} \mathbf{S}_n z^n, \qquad |z| < 1. \qquad (8.73)$$

If z^{-1} is the unit delay operator from digital filter design, then z is the (anti-causal) unit advance operator, and $\mathbf{S}_L(z)$ may be understood as a stable and anti-causal transfer function, with $\{\mathbf{S}_n\}$ its anti-causal impulse response.

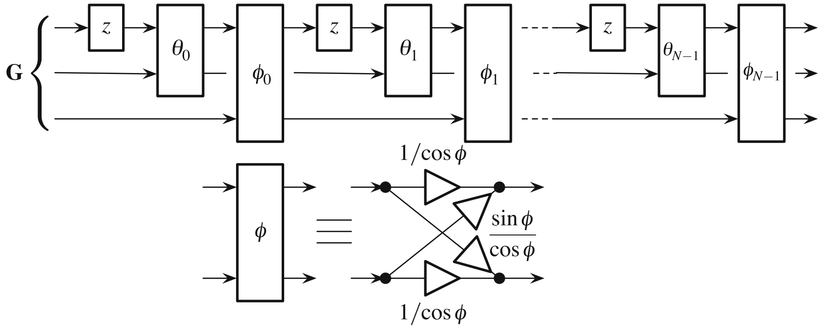


Fig. 8.1 Illustrating the successive annihilation operations applied to the rows of the **G** array.

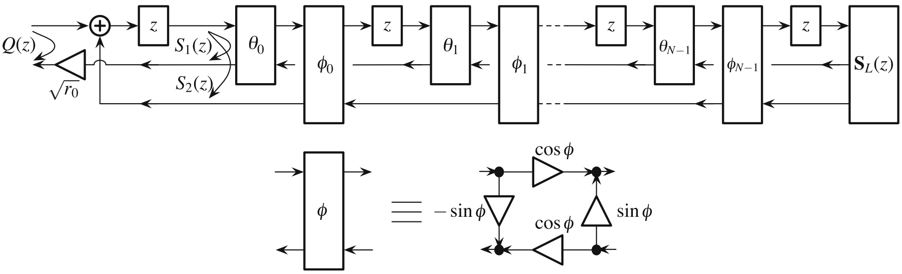


Fig. 8.2 Orthogonal filter, obtained by reversing the flow direction of the lower two branches.

- Upon partitioning $\mathbf{S}_L(z) = \begin{bmatrix} S_{L,1}(z) \\ S_{L,2}(z) \end{bmatrix}$, the radial limits along the unit circle are power complementary:

$$|S_{L,1}(e^{j\omega})|^2 + |S_{L,2}(e^{j\omega})|^2 = 1, \qquad \text{for all } \omega. \tag{8.74}$$

The flow graph with reversed directions on the lower two branches, and incorporating the load $\mathbf{S}_L(z)$, is sketched in Figure 8.2; the curved arrows indicate partial transfer functions $S_1(z)$ and $S_2(z)$ which result at the left-hand-side of the figure. Provided the resulting $S_2(z)$ satisfies the supplementary constraint

$$1 - zS_2(z) \neq 0, \qquad \text{for all } |z| = 1, \tag{8.75}$$

we may close the left-hand side of Figure 8.2 to obtain a stable and anti-causal transfer function

$$Q(z) = \sqrt{r_0}\,\frac{zS_1(z)}{1 - zS_2(z)} = \sum_{n=1}^{\infty} q_n z^n, \tag{8.76}$$

with the following property:

Property 1. Let $\bar{x}(k-n+1) = q_n$, $n = 1, 2, 3, \ldots$. The Hankel matrix $\overline{\mathbf{X}}(k)$ built from this sequence is a factor of the covariance matrix $\mathbf{R}_1$:

$$\mathbf{R}_1 = \overline{\mathbf{X}}^{\mathrm{T}}(k)\overline{\mathbf{X}}(k). \tag{8.77}$$

All such factors may be generated in this way, as $\mathbf{S}_L(z)$ varies over the set of lossless transfer functions.

A proof behind this construct may be found in [27]. We should note that the impulse response $\{q_n\}$ will, in general, have infinite duration. The simplest choice for the right-hand-side load $\mathbf{S}_L(z)$ is a constant unit norm vector of the form $\mathbf{S}_L(z) = \begin{bmatrix} \cos\phi_N \\ \sin\phi_N \end{bmatrix}$, where ϕ_N is any convenient value.

This result shows, thus, that there exists a one-to-one correspondence between the set of positive definite structured matrices of the form (8.59), and the set of rotation angles and scale factor fulfilling the inequalities

$$r_0 > 0 \quad \text{and} \quad \cos\theta_i > 0, \quad \cos\phi_i > 0, \qquad i = 0, 1, \ldots, N-1. \tag{8.78}$$

Note that we end up with $2N+1$ parameters, as expected, because the structured matrix $\mathbf{R}_1$ [cf. (8.59)] is specified by $2N+1$ values, namely r_0, …, r_N and q_1, …, q_N. We confirm next that $2N+1$ is likewise the minimum state vector dimension of a fast least-squares prediction algorithm.

8.5.2 Reachable states in fast least-squares algorithms

We first consider the fast least-squares algorithm from [12, 14], which is a slightly simpler variant of the original fast QR algorithm from [11], and summarized in Table 8.1. A flow graph of the algorithm appears as Figure 8.3; rotations with a zero at one output represent angle solving steps (steps 2, 4 or 5 in the table), with the angles then copied to the data rotation steps (steps 3 and 6 in the table).

We observe that the state of the algorithm is the collection of $2N+1$ stored variables

$$x_{f,0}(k), \ldots, x_{f,N-1}(k), \sqrt{E_{f,N}(k)}, \varepsilon_{b,0}(k), \ldots, \varepsilon_{b,N-1}(k) \tag{8.79}$$

since all other variables are calculated from these. Thus, the minimum state vector dimension can be no larger than $2N+1$. The minimum state vector dimension can be no smaller, either, since there are $2N+1$ parameters involved in reconstructing past inputs by the procedure of Section 8.5.1, which is to say that $2N+1$ is indeed the minimum state vector dimension of a fast least-squares algorithm.

Suppose we consider now the experiment in which the input sequence $\{x(k)\}$ is allowed to vary over all sequences for which the structured matrix $\mathbf{R}_1(k)$ from (8.59) remains positive definite, and let us keep track of all state orientations that may be reached in the algorithm of Table 8.1 when using exact arithmetic. This defines the set of reachable states for the algorithm, and is characterized by the inequalities [30]

$$\sqrt{E_{f,N}(k)} > 0, \qquad \sum_{i=0}^{N-1} \varepsilon_{b,i}^2(k) < 1. \tag{8.80}$$

These inequalities are necessary and sufficient for the rotation angles determined in steps 2 and 5 of Table 8.1 to satisfy

$$|\sin\theta_i(k)| < 1, \qquad |\sin\phi_i(k)| < 1, \qquad \text{for all } i. \tag{8.81}$$

For any such state, therefore, a valid input sequence may be deduced, based on the following property:

Table 8.1 Minimal fast QR decomposition least-squares prediction algorithm.

FQR_POS_B – Version 2[a]

Available at time k: $x_{f,i}(k-1)$, $i=0,1,\ldots,N-1$;
$\sqrt{E_{f,N}(k-1)}$;
$\varepsilon_{b,i}(k-1)$, $i=0,1,\ldots,N-1$;

New datum: $x(k)$;

1. Obtain conversion factor $\gamma_{N-1}(k)$:

$$\gamma_{N-1}(k)=\sqrt{1-\sum_{i=0}^{N-1}\varepsilon_{b,i}^{2}(k-1)}\,;$$

2. Solve for $\theta_i(k)$ angles:

$$\begin{bmatrix}\gamma_{i-1}(k)\\ 0\end{bmatrix}=\begin{bmatrix}\cos\theta_i(k) & \sin\theta_i(k)\\ -\sin\theta_i(k) & \cos\theta_i(k)\end{bmatrix}\begin{bmatrix}\gamma_i(k)\\ \varepsilon_{b,i}(k-1)\end{bmatrix},\qquad i=N-1,\ldots,1,0;$$

3. Update forward prediction variables. With $e_{f,0}(k)=x(k)$, run

$$\begin{bmatrix}x_{f,i}(k)\\ e_{f,i+1}(k)\end{bmatrix}=\begin{bmatrix}\cos\theta_i(k) & \sin\theta_i(k)\\ -\sin\theta_i(k) & \cos\theta_i(k)\end{bmatrix}\begin{bmatrix}\sqrt{\lambda}\,x_{f,i}(k-1)\\ e_{f,i}(k)\end{bmatrix},\qquad i=0,1,\ldots,N-1;$$

4. Update square-root forward prediction energy:

$$\begin{bmatrix}\sqrt{E_{f,N}(k)}\\ 0\end{bmatrix}=\begin{bmatrix}\cos\theta_N(k) & \sin\theta_N(k)\\ -\sin\theta_N(k) & \cos\theta_N(k)\end{bmatrix}\begin{bmatrix}\sqrt{\lambda\,E_{f,N}(k-1)}\\ e_{f,N}(k)\end{bmatrix};$$

5. Solve for $\phi_i(k)$ angles:

$$\begin{bmatrix}\sqrt{E_{f,i}(k)}\\ 0\end{bmatrix}=\begin{bmatrix}\cos\phi_i(k) & \sin\phi_i(k)\\ -\sin\phi_i(k) & \cos\phi_i(k)\end{bmatrix}\begin{bmatrix}\sqrt{E_{f,i+1}(k)}\\ x_{f,i}(k)\end{bmatrix},\qquad i=N-1,\ldots,1,0;$$

6. Update backward prediction errors: with $e_{f,N}(k)=\gamma_{N-1}(k-1)\,\sin\theta_N(k-1)$, run

$$\begin{bmatrix}\varepsilon_{f,i}(k)\\ \varepsilon_{b,i+1}(k)\end{bmatrix}=\begin{bmatrix}\cos\phi_i(k) & \sin\phi_i(k)\\ -\sin\phi_i(k) & \cos\phi_i(k)\end{bmatrix}\begin{bmatrix}\varepsilon_{f,i+1}(k)\\ \varepsilon_{b,i}(k-1)\end{bmatrix},\qquad i=N-1,\ldots,1,0;$$

At the end of the loop, set $\varepsilon_{b,0}(k)=\varepsilon_{f,0}(k)$.

[a] This algorithm corresponds to the one introduced in [14] and named FQR_POS_B–Version 2 in Chapter 4.

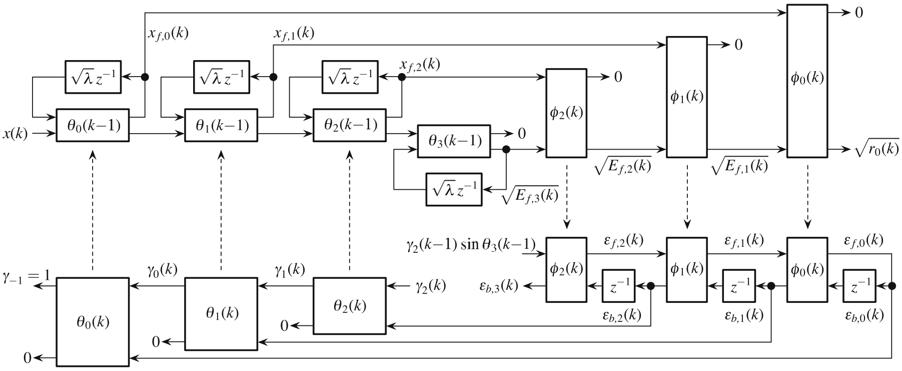


Fig. 8.3 Flow graph of minimal fast QR decomposition least-squares algorithm.

Property 2. Let $\theta_0(k)$, …, $\theta_{N-1}(k)$ and $\phi_0(k)$, …, $\phi_{N-1}(k)$ be the angles computed in exact arithmetic from the algorithm of Table 8.1, for a given input sequence $\{x(\cdot)\}$. Let $\mathbf{R}_1(k)$ be the structured matrix from (8.59) built using this same input sequence. The angles $\theta_0(k), \ldots, \theta_{N-1}(k)$ and $\phi_0(k), \ldots,$ $\phi_{N-1}(k)$ are precisely those which achieve the annihilation of the generator vectors in the algorithm of Section 8.5.1.

The proof of this property involves more advanced notions of displacement generators [31], and may be found in [27, Appendix A]. The upshot is that, as long as the computed state variables in the algorithm satisfy the inequalities (8.80), they may be considered the exact values associated with a perturbed input sequence: From the computed state variables (or the rotation angles which annihilate them in steps 2 and 5 of Table 8.1), a valid past input sequence that would give rise to the computed values may be placed in evidence from the anti-causal filter of Figure 8.2. Backward consistency is thus straightforward to ensure in this version of the algorithm.

8.5.3 QR decomposition lattice algorithm

A noted oddity of the previous fast algorithm is that the order recursions run in both ascending and descending order, which can impede pipelined implementations. An alternate fast QR decomposition algorithm, obtained by Proudler et al. [12, 13], and Ling [16], runs all recursions in ascending order, and consists of the cascade of basic sections illustrated in Figure 8.4; the resulting algorithm is summarized in Table 8.2.

We observe that the total storage is $5N$ variables in the state vector, which is greater than the minimal number $2N+1$. Thus, some redundancy necessarily exists

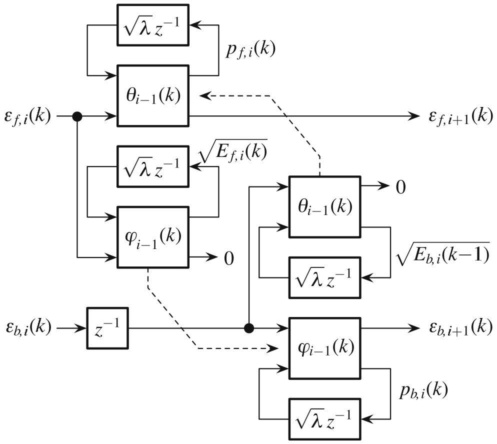


Fig. 8.4 Basic section to be cascaded to obtain the QRD-lattice algorithm.

among the elements of the state vector. For example, one may show [15] that, in exact arithmetic,

$$p_{f,i}(k) = \frac{\sum_{j=0}^{k} \varepsilon_{f,i}(j)\,\varepsilon_{b,i}(j-1)}{\sqrt{E_{b,i}(k-1)}}, \text{ and} \tag{8.82}$$

$$p_{b,i}(k) = \frac{\sum_{j=0}^{k} \varepsilon_{f,i}(j)\,\varepsilon_{b,i}(j-1)}{\sqrt{E_{f,i}(k)}}. \tag{8.83}$$

From this follows easily the equality

$$p_{f,i}(k)\sqrt{E_{b,i}(k-1)} - p_{b,i}(k)\sqrt{E_{f,i}(k)} = 0, \qquad \text{for each } k \text{ and } i, \tag{8.84}$$

inducing one constraint on four of the state variables from any section, so that one variable may be deduced given the remaining three. In the same manner, redundancy between sections may be deduced from the equality

$$\frac{\sqrt{E_{f,i+1}(k)}}{\sqrt{E_{f,i}(k)}} - \frac{\sqrt{E_{b,i+1}(k)}}{\sqrt{E_{b,i}(k-1)}} = 0, \qquad \text{for each } k \text{ and } i, \tag{8.85}$$

since either ratio relates to the reflection coefficient from section i (e.g., [14, 29]). The values calculated in finite precision, however, will not generally satisfy these constraints, which is to say that consistency is not inherited by the QRD-lattice algorithm (a generic problem with non-minimal realizations [17]). For this algorithm, however, the error accumulation properties will nonetheless remain bounded provided $\lambda < 1$ and the input sequence $\{x(k)\}$ is persistently exciting. This is because the algorithm consists of rotations, and variables reinjected through the feedback loop are all attenuated by $\sqrt{\lambda}$; the proof of bounded error growth mimics that of Section 8.4.1 for the full QR adaptive filtering algorithm. Alternatively, a minimal

Table 8.2 QR decomposition lattice least-squares prediction algorithm.

FQRD–Lattice [12, 13, 16]

Available at time k: $p_{f,i}(k-1)$, $i = 0,1,\ldots,N-1$; (forward prediction variables)
$p_{b,i}(k-1)$, $i = 0,1,\ldots,N-1$; (backward prediction variables)
$\sqrt{E_{f,i}(k-1)}$, $i = 0,1,\ldots,N-1$; $\sqrt{\text{forward prediction energies}}$
$\sqrt{E_{b,i}(k-1)}$, $i = 0,1,\ldots,N-1$; $\sqrt{\text{backward prediction energies}}$
$\varepsilon_{b,i}(k-1)$, $i = 0,1,\ldots,N-1$; (backward prediction residuals)

New datum: $\varepsilon_{f,0}(k) = \varepsilon_{b,0}(k) = x(k)$;

For $i = 0, 1, \ldots, N-1$, do:

1. Solve for $\theta_i(k)$ angle:

$$\begin{bmatrix} \sqrt{E_{b,i}(k-1)} \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta_i(k) & \sin\theta_i(k) \\ -\sin\theta_i(k) & \cos\theta_i(k) \end{bmatrix} \begin{bmatrix} \sqrt{\lambda\, E_{b,i}(k-2)} \\ \varepsilon_{b,i}(k-1) \end{bmatrix};$$

2. Update forward prediction variables:

$$\begin{bmatrix} p_{f,i}(k) \\ \varepsilon_{f,i+1}(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_i(k) & \sin\theta_i(k) \\ -\sin\theta_i(k) & \cos\theta_i(k) \end{bmatrix} \begin{bmatrix} \sqrt{\lambda}\, p_{f,i}(k-1) \\ \varepsilon_{f,i}(k) \end{bmatrix};$$

3. Solve for $\varphi_i(k)$ angle:

$$\begin{bmatrix} \sqrt{E_{f,i}(k)} \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\varphi_i(k) & \sin\varphi_i(k) \\ -\sin\varphi_i(k) & \cos\varphi_i(k) \end{bmatrix} \begin{bmatrix} \sqrt{\lambda\, E_{f,i}(k-1)} \\ \varepsilon_{f,i}(k) \end{bmatrix};$$

4. Update backward prediction variables:

$$\begin{bmatrix} p_{b,i}(k) \\ \varepsilon_{b,i+1}(k) \end{bmatrix} = \begin{bmatrix} \cos\varphi_i(k) & \sin\varphi_i(k) \\ -\sin\varphi_i(k) & \cos\varphi_i(k) \end{bmatrix} \begin{bmatrix} \sqrt{\lambda}\, p_{b,i}(k-1) \\ \varepsilon_{b,i}(k-1) \end{bmatrix};$$

variant of the rotation-based lattice algorithm may also be developed [32], using principles of spherical trigonometry. The resulting algorithm inherits the backward consistency property of other minimal algorithms, with the resulting stable error propagation.

8.6 Conclusion

We have reviewed how standard concepts of backward consistency and conditioning intervene in the analysis of recursive least-squares algorithms, and focusing in particular how these concepts play out in QR decomposition recursive least-squares adaptive filtering algorithms. The notion of persistence of excitation on the input sequence proves equivalent to that of a condition number remaining uniformly bounded in time.

The full QR decomposition update equations admit a reasonably direct analysis showing bounded error accumulation, owing to the orthogonal nature of the calculations combined with past data being attenuated via the forgetting factor λ. The conditions for such bounded error growth may also be explained via backward consistency: If numerical errors are indistinguishable from perturbation on the past input data, then the effects of such numerical errors must be forgotten at the same rate as the past input data. This concept proves convenient when studying fast least-squares algorithms, which tend not to admit tractable error analyses through other methods. Two fast QR decomposition algorithms were highlighted. The first is minimal in its storage requirements, and admits simple checks for backward consistency. The rotation angles of that algorithm were shown to have a direct connection to past input reconstruction procedure, which serves to validate the simple description of the set of reachable states. The QR decomposition lattice algorithm was also reviewed, as it offers a modular, pipelineable structure. Although the algorithm is not minimal and thus does not offer backward consistency in general, bounded error growth can be shown in a manner similar to that for the full QR decomposition algorithm. This represents a noted improvement over some other fast least-squares algorithms, notably certain fast transversal filters, which can exhibit explosive error growth once the backward consistency conditions are violated [15, 17, 18].

References

1. J. H. Wilkinson, Error analysis of direct methods of matrix inversion. Journal of the Association for Computing Machinery (JACM), vol. 8, no. 3, pp. 281–330 (July 1961)
2. G. W. Stewart and J. Sun, Matrix Perturbation Theory. Academic Press, San Diego, CA, USA (1990)
3. N. J. Higham, Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1996)
4. G. H. Golub and J. H. Wilkinson, Note on the iterative refinement of least squares solution. Numerische Mathematik, vol. 9, pp. 139–148 (1966)
5. Å. Björck, Numerical Methods for Least Squares Problems. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1996)
6. F. Chaitin-Chatelin and V. Frayssé, Lectures on Finite Precision Computations. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1996)
7. J. H. Wilkinson, A priori error analysis of algebraic processes. Proceedings of the International Congress of Mathematicians, Izdat. MIE, pp. 629–639, Moscow, USSR (1968)
8. T. Kailath, Linear Systems. Prentice-Hall, Englewood Cliffs, NJ, USA (1980)
9. R. R. Bitmead and B. D. O. Anderson, Lyapunov techniques for the exponential stability of linear difference equations with random coefficients. IEEE Transactions on Automatic Control, vol. 25, no. 4, pp. 782–787 (August 1980)
10. P. Dewilde and A.-J. van der Veen, Time-Varying Systems and Computations. Kluwer Academic Publishers, Boston, MA, USA (1998)
11. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, no. 4, pp. 631–653 (April 1990)

12. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, Fast QRD-based algorithms for least squares linear prediction. Mathematics in Signal Processing II (J. G. McWhirter, ed.), Institute of Mathematics and its Applications (IMA) Conference Series, Clarendon Press, no. 26, pp. 465–488, Oxford, UK (September 1990)
13. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, Computationally efficient QR decomposition approach to least squares adaptive filtering. IEE Proceedings-Part F, vol. 138, pp. 341–353 (August 1991)
14. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 879–891 (April 1991)
15. S. Haykin, Adaptive Filter Theory. 3rd edition Prentice-Hall, Upper Saddle River, NJ, USA (1996)
16. F. Ling, Givens rotation based least-squares lattice and related algorithms. IEEE Transactions on Signal Processing, vol. 39, no, 7, pp. 1541–1551 (July 1991)
17. P. A. Regalia, Numerical stability issues in fast least-squares adaptation algorithms. Optical Engineering, vol. 31, pp. 1144–1152 (June 1992)
18. D. T. M. Slock, The backward consistency concept and round-off error propagation dynamics in recursive least-squares algorithms. Optical Engineering, vol. 31, pp. 1153–1169 (June 1992)
19. J. R. Bunch, R. C. Le Borne, and I. K. Proudler, A conceptual framework for consistency, conditioning, and stability issues in signal processing. IEEE Transactions on Signal Processing, vol. 49, no. 9, pp. 1971–1981 (September 2001)
20. S. Ljung and L. Ljung, Error propagation properties of recursive least-squares adaptation algorithms. Automatica, vol. 21, no. 2, pp. 157–167 (March 1985)
21. A. P. Liavas and P. A. Regalia, On the numerical stability and accuracy of the conventional recursive least-squares algorithm. IEEE Transactions on Signal Processing, vol. 47, no. 1, pp. 88–96 (January 1999)
22. C. T. Mullis and R. A. Roberts, The use of second-order information in the approximation of discrete-time linear systems. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-24, no. 3, pp. 226–238 (June 1976)
23. Y. Inouye, Approximation of multivariable linear systems with impulse response and autocorrelation sequences. Automatica, vol. 19, no. 2, pp. 265–277 (May 1983)
24. A. M. King, U. B. Desai, and R. E. Skelton, A generalized approach to q-Markov covariance equivalent realizations for discrete systems. Automatica, vol. 24, no. 4, pp. 507–515 (July 1988)
25. P. A. Regalia, M. Mboup, and M. Ashari, A class of first- and second-order interpolation problems in model reduction. Archiv für Elektronik und Ubertragungstechnik, vol. 49, no. 5/6, pp. 332–343 (September 1995)
26. D. Alpay, V. Bolotnikov, and Ph. Loubaton, On tangential H_2 interpolation with second order norm constraints. Integral Equations and Operator Theory (Birkhäuser Basel), vol. 24, no. 2, pp. 156–178 (June 1996)
27. P. A. Regalia, Past input reconstruction in fast least-squares algorithms. IEEE Transactions on Signal Processing, vol. 45, no. 9, pp. 2231–2240 (September 1997)
28. T. Kailath and A. H. Sayed, Displacement structure: Theory and applications. SIAM Review, vol. 37, no. 3, pp. 297–386 (September 1995)
29. H. Lev-Ari and T. Kailath, Lattice filter parametrization and modeling of nonstationary processes. IEEE Transactions on Information Theory, vol. IT-30, no. 1, pp. 2–16 (January 1984)
30. P. A. Regalia, Numerical stability properties of a QR-based fast least squares algorithm. IEEE Transactions on Signal Processing, vol. 41, no. 6, pp. 2096–109 (June 1993)
31. J. Chun, T. Kailath, and H. Lev-Ari, Fast parallel algorithms for QR and triangular factorization. SIAM Journal on Scientific and Statistical Computing, vol. 8, no. 6, pp. 899–913 (November 1987)
32. F. Desbouvries and P. A. Regalia, A minimal rotation-based FRLS lattice algorithm. IEEE Transactions on Signal Processing, vol. 45, no. 5, pp. 1371–1374 (May 1997)

Chapter 9
Finite and Infinite-Precision Properties of QRD-RLS Algorithms

Paulo S. R. Diniz and Marcio G. Siqueira

Abstract This chapter analyzes the finite and infinite-precision properties of QR-decomposition recursive least-squares (QRD-RLS) algorithms with emphasis on the conventional QRD-RLS and fast QRD-lattice (FQRD-lattice) formulations. The analysis encompasses deriving mean squared values of internal variables in steady-state and also the mean squared error of the deviations of the same variables assuming fixed-point arithmetic. In particular, analytical expressions for the excess of mean squared error and for the variance of the deviation in the tap coefficients of the QRD-RLS algorithm are derived, and the analysis is extended to the error signal of the FQRD-lattice algorithm. All the analytical results are confirmed to be accurate through computer simulations. Conclusions follow.

9.1 Introduction

The implementation of the QR-decomposition recursive least-squares (QRD-RLS) algorithm requires the knowledge of the dynamic range of its internal variables in order to determine their wordlengths. For the systolic array implementation of the QRD-RLS algorithm, the steady-state values of the cosines and sines of the Givens rotations and the bounds for the dynamic range of the processing cells contents are known [1]. An attractive feature of the QRD-RLS algorithm is the bounded input/bounded output stability, as proven in [1, 2].

In this chapter, we present a complete quantitative analysis of the dynamic range of all internal quantities of the QRD-RLS and fast QRD (FQRD)-lattice algorithms.

Paulo S. R. Diniz
Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro – Brazil
e-mail: diniz@lps.ufrj.br

Marcio G. Siqueira
Cisco Systems, Inc., San Jose, CA – USA
e-mail: mgs@cisco.com

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_9,

First, stability conditions for the QRD-RLS algorithm are derived by examining the quantization error propagation in all recursive equations of the algorithm assuming fixed-point arithmetic. In addition, the mean squared value of the errors accumulated in all variables of the algorithm are derived and accurate expressions for the excess of mean squared error and the mean squared value of the error in the filter coefficients are proposed for the conventional QRD-RLS algorithm. The results are later extended to the FQRD-lattice algorithm. Simulations follow finite-precision analysis. The chapter is concluded with a discussion on the derived and simulated results.

9.2 Precision Analysis of the QR-Decomposition RLS Algorithm

RLS algorithms update the adaptive filter coefficients in order to minimize the following objective (cost) function

$$\xi(k) = \sum_{i=0}^{k} \lambda^{k-i} |\bar{e}(i)|^2 = \sum_{i=0}^{k} [d(i) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(i)]^2, \tag{9.1}$$

where $\mathbf{x}(k) = [x(k)\, x(k-1) \cdots x(k-N)]^{\mathrm{T}}$ is the input signal vector, $\mathbf{w}(k) = [w_0(k)\, w_1(k) \cdots w_N(k)]^{\mathrm{T}}$ is the coefficient vector at instant k, $\bar{e}(i)$ is the output error at instant i (computed with $\mathbf{w}(k)$), and λ is the forgetting factor. The input signal $x(k)$ is considered a Gaussianly distributed white random variable with zero mean and variance σ_x^2. However, the analysis is extended for non-white inputs.

The error vector, the reference signal vector, and the input signal information matrix can be defined, respectively, as

$$\mathbf{e}^{\mathrm{T}}(k) \triangleq [\bar{e}(k)\, \lambda^{1/2}\bar{e}(k-1) \cdots \lambda^{k/2}\bar{e}(0)], \tag{9.2}$$

$$\mathbf{d}^{\mathrm{T}}(k) \triangleq [d(k)\, \lambda^{1/2}d(k-1) \cdots \lambda^{k/2}d(0)], \text{ and} \tag{9.3}$$

$$\begin{aligned}\mathbf{X}^{\mathrm{T}}(k) &\triangleq [\mathbf{x}(k)\, \lambda^{1/2}\mathbf{x}(k-1) \cdots \lambda^{k/2}\mathbf{x}(k)] \\ &= \begin{bmatrix} x(k) & \lambda^{1/2}x(k-1) & \cdots & \lambda^{k/2}x(0) \\ x(k-1) & \lambda^{1/2}x(k-2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x(k-N) & \lambda^{1/2}x(k-N-1) & \cdots & 0 \end{bmatrix},\end{aligned} \tag{9.4}$$

so that the objective function can be rewritten as $\xi(k) = \mathbf{e}^{\mathrm{T}}(k)\mathbf{e}(k)$, where $\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k)$.

The input data matrix $\mathbf{X}(k)$ can be triangularized through Givens rotations. The resulting triangularized matrix $\mathbf{U}(k)$, assuming the case of upper triangularization, may be given by

$$\mathbf{U}(k) \overset{\Delta}{=} \begin{bmatrix} u_{0,0}(k) & u_{0,1}(k) & \cdots & u_{0,N}(k) \\ 0 & u_{1,1}(k) & \cdots & u_{1,N}(k) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{N,N}(k) \end{bmatrix}. \tag{9.5}$$

When the orthogonal transformation denoted by $\mathbf{Q}(k)$ is applied to the error vector, it follows that:

$$\begin{aligned} \mathbf{Q}(k)\mathbf{e}(k) &= \mathbf{Q}(k)\,\mathbf{d}(k) - \mathbf{Q}(k)\mathbf{X}(k)\mathbf{w}(k) = \hat{\mathbf{d}}(k) - \hat{\mathbf{X}}(k)\,\mathbf{w}(k) \\ &= \begin{bmatrix} \hat{\mathbf{e}}_1(k) \\ \hat{\mathbf{e}}_2(k) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{d}}_1(k) \\ \hat{\mathbf{d}}_2(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{k-N,N+1} \\ \mathbf{U}(k) \end{bmatrix} \mathbf{w}(k). \end{aligned} \tag{9.6}$$

The QRD-RLS algorithm shown in Table 9.1 originates from the formulation above, see [3–5] for details. The operator Q[$\cdot$] denotes quantization and should be ignored in the discussions of the current section. The notation f_Q denotes finite-precision version of f.[1]

In Equations (9.7) and (9.8), $a_{i,i}(k)$ (not explicitly shown) represents an intermediate quantity that appears in the position $(1,i)$ of the matrix $\mathbf{Q}'_{i-1}(k)\cdots\mathbf{Q}'_0(k)\mathbf{X}^p(k)$ (see (9.11)), that will be eliminated by the Givens rotation $\mathbf{Q}'_i(k)$. Quantities $b_i(k)$ represent intermediate values of the first element of $\mathbf{Q}'_{i-1}(k)\cdots\mathbf{Q}'_0(k)\mathbf{d}_2^p(k)$.

9.2.1 Infinite-precision analysis

Let's now derive the mean squared value of several quantities related to the QRD-RLS algorithm, namely $\cos\theta_i(k)$, $\sin\theta_i(k)$, $u_{i,j}(k)$, $a_{i,j}(k)$, $\hat{d}_{2,i}(k)$, $b_i(k)$, and $e_q(k)$. These results are required to analyze the QRD algorithm implemented with finite precision.

It is worth mentioning that the analysis is valid for averages taken as k is large. Although the label k is redundant in most expressions it can be useful for transient analysis.

9.2.1.1 Mean squared values of sines and cosines

From the implementation of (9.7), (9.8), (9.10), and (9.11), with infinite-precision arithmetic, it is can be shown that

$$u_{i,i}(k) = \sqrt{\lambda u_{i,i}^2(k-1) + a_{i,i}^2(k)}; \tag{9.18}$$

therefore,

[1] In practice, vectors and matrices with growing dimensions in Equations (9.7), (9.8), (9.9), (9.10), (9.11), (9.12), (9.13), (9.14), (9.15), (9.16), and (9.17) should be replaced by fixed dimension ones. This notation was chosen to clarify the presentation.

Table 9.1 Conventional QR-decomposition RLS algorithm.

QRD-RLS

Matrix Formulation: For $i = 0, \ldots, N$, do

$$\cos_Q \theta_i(k) = \mathrm{Q}\left[z \frac{\lambda^{1/2} u_{i,i;Q}(k-1)}{\mathrm{Q}\left[\sqrt{\mathrm{Q}[\lambda u^2_{i,i;Q}(k-1) + a^2_{i,i;Q}(k)]}\right]} \right] \tag{9.7}$$

$$\sin_Q \theta_i(k) = \mathrm{Q}\left[\frac{a_{i,i;Q}(k)}{\mathrm{Q}\left[\sqrt{\mathrm{Q}[\lambda u^2_{i,i;Q}(k-1) + a^2_{i,i;Q}(k)]}\right]} \right] \tag{9.8}$$

$$\mathbf{Q}'_{i;Q}(k) = \begin{bmatrix} \cos_Q \theta_i(k) & & -\sin_Q \theta_i(k) & \\ & \mathbf{I}_{k-N+i-1} & & \\ \sin_Q \theta_i(k) & & \cos_Q \theta_i(k) & \\ & & & \mathbf{I}_{N-i} \end{bmatrix} \tag{9.9}$$

$$\mathbf{X}^p_Q(k) = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(k) \\ \mathbf{0}_{k-N-1,N+1} \\ \lambda^{1/2}\mathbf{U}_Q(k-1) \end{bmatrix} \tag{9.10}$$

$$\begin{aligned} \hat{\mathbf{X}}_Q(k) &= \mathrm{Q}[\mathbf{Q}'_{N;Q}(k)\mathrm{Q}[\mathbf{Q}'_{N-1;Q}(k) \cdots \mathrm{Q}[\mathbf{Q}'_{0;Q}(k)\mathbf{X}^p_Q(k)] \cdots]] \\ &= \mathrm{Q}[\tilde{\mathbf{Q}}_Q(k)\mathbf{X}^p_Q(k)] \\ &= \begin{bmatrix} \mathbf{0}_{k-N,N+1} \\ \mathbf{U}_Q(k) \end{bmatrix} \end{aligned} \tag{9.11}$$

$$\mathbf{d}^p_Q(k) = \begin{bmatrix} d^*(k) \\ \lambda^{1/2}\hat{\mathbf{d}}_{1;Q}(k-1) \\ \lambda^{1/2}\hat{\mathbf{d}}_{2;Q}(k-1) \end{bmatrix} \tag{9.12}$$

$$\begin{aligned} \hat{\mathbf{d}}_Q(k) &= \mathrm{Q}[\mathbf{Q}'_{N;Q}(k)\mathrm{Q}[\mathbf{Q}'_{N-1;Q}(k) \cdots \mathrm{Q}[\mathbf{Q}'_{0;Q}(k)\mathbf{d}^p_Q(k)] \cdots]] \\ &= \mathrm{Q}[\tilde{\mathbf{Q}}(k)\mathbf{d}^p_Q(k)] \\ &= \begin{bmatrix} e_{q;Q}(k) \\ \hat{\mathbf{d}}_{1;Q}(k) \\ \hat{\mathbf{d}}_{2;Q}(k) \end{bmatrix} \end{aligned} \tag{9.13}$$

Back-substitution: For $j = 0, \ldots, N$ do

$$f_{j;Q}(k) = \mathrm{Q}\left[\sum_{i=j+1}^{N} w_{i;Q}(k) u_{j,i;Q}(k) \right] \tag{9.14}$$

$$w_{j;Q}(k) = \mathrm{Q}\left[\frac{d_{2,j;Q}(k) - f_{j;Q}(k)}{u_{j,j;Q}(k)} \right] \tag{9.15}$$

Error calculation: Use one of the equations

$$e_Q(k) = \mathrm{Q}[e_{q;Q}(k)\mathrm{Q}[\cos_Q \theta_0(k) \cdots \mathrm{Q}[\cos_Q \theta_N(k) \cos_Q \theta_{N-1}(k)] \cdots]]] \tag{9.16}$$

$$e_Q(k) = d(k) - \mathrm{Q}[\mathbf{w}^{\mathrm{T}}_Q(k)\mathbf{x}(k)] \tag{9.17}$$

$$\cos\theta_i(k) = \frac{\lambda^{1/2}u_{i,i}(k-1)}{u_{i,i}(k)}, \text{ and} \tag{9.19}$$

$$\sin\theta_i(k) = \frac{a_{i,i}(k)}{u_{i,i}(k)}. \tag{9.20}$$

The mean squared value of $\cos\theta_i(k)$ is then given by

$$E\left\{\cos^2\theta_i(k)\right\} = E\left\{\lambda\frac{\sum_{j=0}^{k-1}\lambda^{k-1-j}a_{i,i}^2(j)}{\sum_{j=0}^{k}\lambda^{k-j}a_{i,i}^2(j)}\right\} \approx \lambda. \tag{9.21}$$

whereas the mean squared value of $\sin\theta_i(k)$ can be calculated by using the fundamental trigonometric identity

$$E\{\sin^2\theta_i(k)\} \approx 1-\lambda, \tag{9.22}$$

for $k \longrightarrow \infty$.

9.2.1.2 Mean squared value of $u_{i,j}(k)$

The derivations of the mean squared values of the elements of $\mathbf{U}(k)$ are somewhat involved, especially when the order of the adaptive filter is high. Under the assumption that the input signal samples are uncorrelated, it is possible to deduce the desired formulas [6]:

$$E\{u_{i,i}^2(k)\} \approx \frac{\sigma_x^2}{1-\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^i, \text{ and} \tag{9.23}$$

$$E\{u_{i,j}^2(k)\} \approx \frac{\sigma_x^2}{1+\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^i, \tag{9.24}$$

where the last equation is valid for $j > i$, $i = 0, 1, \ldots, N$.

9.2.1.3 Mean squared value of $a_{i,j}(k)$

From (9.18), it is possible to verify that

$$E[u_{i,i}^2(k)] = \frac{E[a_{i,i}^2(k)]}{1-\lambda}. \tag{9.25}$$

From (9.25) and (9.23), one can show that

$$E\{a_{i,i}^2(k)\} = \sigma_x^2\left[\frac{2\lambda}{1+\lambda}\right]^i, \tag{9.26}$$

for $i = 0, \ldots, N$. Since $a_{i,j}(k)$ for $i \neq j$ and $a_{i,i}(k)$ are result of similar dynamic equations, it is possible to show that

$$E\{a_{i,j}^2(k)\} = \sigma_x^2 \left[\frac{2\lambda}{1+\lambda}\right]^i, \tag{9.27}$$

for $i = 0, \ldots, N$ and $j > i$.

9.2.1.4 Mean squared value of $\hat{\mathbf{d}}_{2,i}(k)$

The adaptive filter coefficients are calculated using (9.6), that is

$$\hat{d}_{2,i}(k) = \sum_{j=i}^{N} u_{i,j}(k) w_j(k). \tag{9.28}$$

If $u_{i,i}(k)$ is considered the only element in the ith row of $\mathbf{U}(k)$ with non-zero mean, and that the elements in a given row are uncorrelated, the following expression approximation is valid.

$$E\{\hat{d}_{2,i}^2(k)\} \approx \sum_{j=i}^{N} E\{u_{i,j}^2(k)\} E\{w_j^2(k)\} \tag{9.29}$$

In addition, considering that the mean of $w_i(k)$ is much larger than the variance, so that the mean squared value can be replaced by the squared mean, the equation becomes

$$E\{\hat{d}_{2,i}^2(k)\} \approx \left[\frac{2\lambda}{1+\lambda}\right]^i \left[\frac{\sigma_x^2}{1-\lambda} w_{o,i}^2 + \frac{\sigma_x^2}{1+\lambda} \sum_{j=i+1}^{N} w_{o,j}^2\right], \tag{9.30}$$

where

$$w_{o,i}^2 = E\{w_i^2(k)\}. \tag{9.31}$$

9.2.1.5 Mean squared value of $b_i(k)$

The intermediate values of the first element of $\hat{\mathbf{d}}(k)$ during the application of the Givens rotations, denoted by $b_i(k)$ for $i = 0, 1 \ldots, N$, are given by

$$b_{i+1}(k) = -\lambda^{1/2} \hat{d}_{2,i}(k-1) \sin\theta_i(k) + b_i(k) \cos\theta_i(k). \tag{9.32}$$

Since $E\{\cos\theta_i(k) \sin\theta_i(k)\}$ is relatively small, it is possible to infer that

$$\begin{aligned} E\{b_{i+1}^2(k)\} &= \lambda E\{\hat{d}_{2,i}^2(k-1)\} E\{\sin^2\theta_i(k)\} + E\{b_i^2(k)\} E\{\cos^2\theta_i(k)\} \\ &= \lambda(1-\lambda) E\{\hat{d}_{2,i}^2(k-1)\} + \lambda E\{b_i^2(k)\} \\ &= \sum_{j=1}^{i+1} \lambda^{i-j+2} (1-\lambda) E\{\hat{d}_{2,j}^2(k)\} + \lambda^{i+1} E\{d^2(k)\}. \end{aligned} \tag{9.33}$$

9.2.1.6 Mean squared value of $e_q(k)$

From (9.21) and (9.16), and by assuming that the mean values of the cosines are much larger than their variance [7], it is possible to verify that

$$E\{e_q^2(k)\} \approx \frac{E\{e^2(k)\}}{\lambda^{N+1}}. \tag{9.34}$$

If the QRD-RLS algorithm is applied in a sufficient order identification problem, where the desired signal can be modeled by a moving average process with a measurement noise with variance σ_r^2. After convergence, it is expected that

$$E\{e_q^2(k)\} \approx \frac{\sigma_r^2}{\lambda^{N+1}}. \tag{9.35}$$

9.2.1.7 Dynamic range

The internal variables of the QRD-RLS algorithm are the elements of $\mathbf{U}(k)$, of $\hat{\mathbf{d}}_2(k)$, and the sines and cosines. Let's assume that all variables are represented in fixed-point arithmetic in the range -1 to $+1$, in order to derive the conditions on the input signal variance to ensure that overflow does not occur frequently in internal variables of the algorithm.

The off-diagonal elements of $\mathbf{U}(k)$ have zero mean and mean squared values much smaller than the diagonal elements. The diagonal elements usually have larger mean squared values as λ approaches 1; as such, some strategy to control the overflow must be devised. Considering that for λ close to one, the mean of $u_{i,i}(k)$ is large as compared to its standard deviation, one can calculate its mean squared value through its squared mean. From (9.23), it follows that

$$E\{u_{i,i}(k)\} \approx \frac{\sigma_x}{\sqrt{1-\lambda}} \left[\frac{2\lambda}{1+\lambda}\right]^{i/2}. \tag{9.36}$$

As $u_{0,0}(k)$ has the largest energy, and if the maximum value for $u_{0,0}(k)$ is 1, satisfying the condition

$$\frac{\sigma_x}{\sqrt{1-\lambda}} < 1 \tag{9.37}$$

is sufficient to avoid frequent internal overflow.

The values of the entries of $\hat{\mathbf{d}}_2(k)$ should also be kept in the range -1 and $+1$. For any k,

$$E\{\hat{d}_{2,i}(k)\} = \sum_{j=i}^{N} E\{u_{i,j}(k)w_j(k)\}. \tag{9.38}$$

Assuming that the mean of $u_{i,j}(k)$ is zero for $i \neq j$, and that the standard deviations of $u_{i,i}(k)$ and $w_i(k)$ are small compared to their respective mean, it is possible to verify that

$$\begin{aligned} E\{\hat{d}_{2,i}(k)\} = E\{u_{i,i}(k)w_i(k)\} &\approx E\{u_{i,i}(k)\}E\{w_i(k)\} \\ &= \frac{\sigma_x}{\sqrt{1-\lambda}}\left[\frac{2\lambda}{1+\lambda}\right]^{i/2} w_{o,i}. \end{aligned} \tag{9.39}$$

The most stringent case is $i = 0$, so that frequent overflows can be avoided if, the following inequality is satisfied:

$$\sigma_x^2\, w_{o,i}^2 < 1-\lambda. \tag{9.40}$$

This inequality requires the mean squared value of the taps $w_{o,i}$, that accounts for the relative power of the reference signal. Although the values of $w_{o,i}$ are not known in advance, a rough estimate of $\sigma_x^2\, w_{o,i}^2$ can be obtained through the power of the reference signal [8].

9.2.2 Stability analysis

In this section, the fixed-point quantization errors are first modeled, and the recursive equations describing the total error in each quantity of the QRD-RLS algorithm are derived. For that, we discuss the conditions to guarantee the stability of the algorithm.

For the analysis results presented here, we assume that the input signal has been properly scaled in order to avoid overflow. Two's complement arithmetic is used for numeric representation. It is taken for granted that no overflow occurs so that additions and subtractions do not introduce quantization errors.

The multiplication, division, and square-root operations introduce, respectively, quantization errors described by

$$\eta_M(a,b) \triangleq ab - \mathrm{Q}[ab], \tag{9.41}$$

$$\eta_D(a,b) \triangleq a/b - \mathrm{Q}[a/b], \tag{9.42}$$

$$\eta_S(a) \triangleq \sqrt{a} - \mathrm{Q}[\sqrt{a}], \tag{9.43}$$

where a and b are scalars. For inner product with quantization after addition, the errors are denoted as

$$\eta_M[(a_1,b_1);\cdots;(a_i,b_i)] \triangleq \sum_{j=1}^{i} a_j b_j - \mathrm{Q}\left[\sum_{j=1}^{i} a_j b_j\right]. \tag{9.44}$$

The quantization errors for matrix–vector and matrix–matrix products are modeled as

$$\eta_M(\mathbf{A},\mathbf{b}) \overset{\Delta}{=} \mathbf{A}\mathbf{b} - \mathrm{Q}[\mathbf{A}\mathbf{b}] \tag{9.45}$$

and

$$\mathbf{N}_M(\mathbf{A},\mathbf{B}) \overset{\Delta}{=} \mathbf{A}\mathbf{B} - \mathrm{Q}[\mathbf{A}\mathbf{B}], \tag{9.46}$$

respectively.

Instantaneous quantizations are performed by rounding, for any type of arithmetic. The quantization error has zero mean and variance $2^{-2B}/12$, where B is the number of bits excluding the sign.

The overall quantization error in each quantity is defined as the difference between its value in infinite-precision implementation and its value in finite-precision implementation, that is

$$\Delta a(k) \overset{\Delta}{=} a(k) - a_Q(k). \tag{9.47}$$

Matrix $\hat{\mathbf{X}}(k)$ is defined as

$$\begin{aligned}\hat{\mathbf{X}}(k) &= \tilde{\mathbf{Q}}(k)\mathbf{X}^p(k) = [\tilde{\mathbf{Q}}_Q(k) + \Delta\tilde{\mathbf{Q}}(k)][\mathbf{X}_Q^p(k) + \Delta\mathbf{X}^p(k)]\\ &= \tilde{\mathbf{Q}}_Q(k)\mathbf{X}_Q^p(k) + \tilde{\mathbf{Q}}_Q(k)\Delta\mathbf{X}^p(k) + \Delta\tilde{\mathbf{Q}}(k)\mathbf{X}_Q^p(k)\\ &\quad + \Delta\tilde{\mathbf{Q}}(k)\Delta\mathbf{X}^p(k).\end{aligned} \tag{9.48}$$

From (9.11) and (9.47), it follows that

$$\begin{aligned}\tilde{\mathbf{Q}}_Q(k)\mathbf{X}_Q^p(k) &= \mathrm{Q}[\tilde{\mathbf{Q}}_Q(k)\mathbf{X}_Q^p(k)] + \mathbf{N}_M[\tilde{\mathbf{Q}}_Q(k),\mathbf{X}_Q^p(k)]\\ &= \hat{\mathbf{X}}_Q(k) + \mathbf{N}_M[\tilde{\mathbf{Q}}_Q(k),\mathbf{X}_Q^p(k)].\end{aligned} \tag{9.49}$$

It can then be shown that

$$\begin{aligned}\Delta\hat{\mathbf{X}}(k) = \hat{\mathbf{X}}(k) - \hat{\mathbf{X}}_Q(k) &= \tilde{\mathbf{Q}}_Q(k)\Delta\mathbf{X}^p(k) + \Delta\tilde{\mathbf{Q}}(k)\Delta\mathbf{X}^p(k)\\ &\quad + \Delta\tilde{\mathbf{Q}}(k)\mathbf{X}_Q^p(k) + \mathbf{N}_M[\tilde{\mathbf{Q}}_Q(k),\mathbf{X}_Q^p(k)].\end{aligned} \tag{9.50}$$

Using (9.50), (9.10), the definition of (9.47) and considering that $\mathbf{U}(k) = \mathbf{U}_Q(k) + \Delta\mathbf{U}(k)$, we find that

$$\begin{aligned}\begin{bmatrix}\mathbf{0}_{k-N,N+1}\\ \Delta\mathbf{U}(k)\end{bmatrix} &= \tilde{\mathbf{Q}}_Q(k)\begin{bmatrix}\mathbf{0}_{k-N,N+1}\\ \lambda^{1/2}\Delta\mathbf{U}(k-1)\end{bmatrix}\\ &\quad + \Delta\tilde{\mathbf{Q}}(k)\begin{bmatrix}\mathbf{x}^{\mathrm{T}}(k)\\ \mathbf{0}_{k-N-1,N+1}\\ \lambda^{1/2}\mathbf{U}(k-1)\end{bmatrix} + \mathbf{N}_M[\tilde{\mathbf{Q}}_Q(k),\mathbf{X}_Q^p(k)].\end{aligned} \tag{9.51}$$

The above equation represents the dynamics of the error in the input signal matrix after triangularization. The convergence in average of $\mathbf{U}(k)$ can be guaranteed if the following inequality is satisfied

$$\lambda^{1/2} \parallel \tilde{\mathbf{Q}}_Q(k) \parallel_2 \leq 1, \tag{9.52}$$

where the two norm of a matrix is defined here as the square root of the largest eigenvalue. Hence,

$$\parallel \tilde{\mathbf{Q}}_Q(k) \parallel_2 = MAX_i \sqrt{\cos_Q^2 \theta_i(k) + \sin_Q^2 \theta_i(k)}. \tag{9.53}$$

Then, the stability condition can be rewritten as follows:

$$\lambda < \frac{1}{MAX_i \left[\cos_Q^2 \theta_i(k) + \sin_Q^2 \theta_i(k)\right]}. \tag{9.54}$$

By assuming instantaneous errors term non-zero in (9.51), we can show that

$$\begin{aligned} \parallel E\{\Delta\mathbf{U}(k)\} \parallel \leq \lambda^{1/2} \parallel E\{\tilde{\mathbf{Q}}_Q(k)\} \parallel \parallel E\{\Delta\mathbf{U}(k-1)\} \parallel \\ + \parallel E\{\Delta\tilde{\mathbf{Q}}(k)\} \parallel \parallel E\{\mathbf{X}(k)\} \parallel . \end{aligned} \tag{9.55}$$

Notice that (9.54) is also sufficient to guarantee that (9.55) is stable. For $\lambda = 1$ and $E\{\|\tilde{\mathbf{Q}}_Q(k)\|\} = 1$, the norm of $E[\Delta\mathbf{U}(k)]$ increases indefinitely, if the input signal is non-zero.

9.2.3 Error propagation analysis in steady-state

In this subsection, we derive the error propagation. Analytical expressions for the mean squared value of the errors in the prediction error and in the tap coefficients are obtained.

9.2.3.1 Mean squared value of $\Delta a_{i,j}(k)$

During the triangularization process, the intermediate value that the jth element of the first row of $\mathbf{X}^p(k)$ assumes in the ith Givens rotation is denoted as $a_{i,j}(k)$. These quantities are given by

$$a_{i+1,j}(k) = a_{i,j}(k)\cos\theta_i(k) - \lambda^{1/2} u_{i,j}(k-1)\sin\theta_i(k), \tag{9.56}$$

where $a_{0,j}(k) \overset{\Delta}{=} x(k-j)$. The equation above can be solved recursively as

$$\begin{aligned} a_{i,j}(k) = {} & x(k-j) \prod_{m=0}^{i-1} [\cos_Q \theta_m(k) + \Delta \cos \theta_m(k)] \\ & -\lambda^{1/2} \sum_{m=0}^{i-1} [u_{m,j;Q}(k-1) + \Delta u_{i,j}(k-1)][\sin_Q \theta_m(k) + \Delta \sin \theta_m(k)] \cdot \\ & \prod_{n=m+1}^{i-1} [\cos_Q \theta_n(k) + \Delta \cos \theta_n(k)], \end{aligned} \tag{9.57}$$

Note that in the last equality $a_{i,j}(k)$ is expressed as a function of the quantities in the finite-precision implementation and their respective errors. By neglecting all second-order and higher-order error terms, it follows that

$$\begin{aligned} a_{i,j}(k) \approx {} & x(k-j) \prod_{m=0}^{i-1} \cos_Q \theta_m(k) \\ & -\lambda^{1/2} \sum_{m=0}^{i-1} u_{m,j;Q}(k-1) \sin_Q \theta_m(k) \prod_{n=m+1}^{i-1} \cos_Q \theta_n(k) \\ & +x(k-j) \sum_{n=0}^{i-1} \Delta \cos \theta_n(k) \prod_{\substack{m=0 \\ m \neq n}}^{i-1} \cos_Q \theta_m(k) \\ & -\lambda^{1/2} \sum_{m=0}^{i-1} [u_{m,j;Q}(k-1) \Delta \sin \theta_m(k) \\ & +\Delta u_{m,j}(k-1) \sin_Q \theta_m(k)] \prod_{n=m+1}^{i-1} \cos_Q \theta_n(k) \\ & -\lambda^{1/2} \sum_{m=0}^{i-1} u_{m,j;Q}(k-1) \sin_Q \theta_m(k) \cdot \\ & \left\{ \sum_{n=m+1}^{i-1} \Delta \cos \theta_n(k) \prod_{\substack{q=0 \\ q \neq n}}^{i-1} \cos_Q \theta_q(k) \right\}. \end{aligned} \tag{9.58}$$

In finite-precision case, $a_{i,j;Q}(k)$ is given by

$$\begin{aligned} a_{i,j;Q}(k) = {} & x(k-j) \prod_{m=0}^{i-1} \cos_Q \theta_m(k) \\ & -\lambda^{1/2} \sum_{m=0}^{i-1} u_{m,j;Q}(k-1) \sin_Q \theta_m(k) \prod_{n=m+1}^{i-1} \cos_Q \theta_n(k) \\ & -\sum_{m=0}^{i-1} \eta_M^{aij}(k) \prod_{n=m+1}^{i-1} \cos_Q \theta_n(k), \end{aligned} \tag{9.59}$$

The quantities $\eta_M^{aij}(k) = \eta_M[(a_{i,j}(k), \cos\theta_i(k)); (\lambda^{1/2}u_{i,j}(k-1), \sin\theta_i(k))]$, for $m = 0,\ldots,i$, represent quantization noises generated by the products.
From (9.57) and (9.59), it follows that

$$\begin{aligned}
\Delta a_{i,j}(k) = x(k-j) & \sum_{n=0}^{i-1} \Delta\cos_Q\theta_n(k) \prod_{\substack{m=0\\ m\neq n}}^{i-1} \cos_Q\theta_m(k) \\
& -\lambda^{1/2} \sum_{m=0}^{i-1} [u_{m,j;Q}(k-1)\Delta\sin\theta_m(k) + \Delta u_{m,j}(k-1)\sin_Q\theta_m(k)] \cdot \\
& \prod_{n=m+1}^{i-1} \cos_Q\theta_n(k)\lambda^{1/2} \sum_{m=0}^{i-1} u_{m,j;Q}(k-1)\sin_Q\theta_m(k) \cdot \\
& \left\{ \sum_{n=m+1}^{i-1} \Delta\cos\theta_n(k) \prod_{\substack{q=m+1\\ n\neq q}}^{i-1} \cos_Q\theta_q(k) \right\} \\
& + \sum_{m=0}^{i-1} \eta_m^{aij}(k) \prod_{n=m+1}^{i-1} \cos_Q\theta_n(k). \qquad (9.60)
\end{aligned}$$

We assume now that $x(k)$, $\Delta\cos\theta_i(k)$, $\Delta\sin\theta_i(k)$, and $\eta_i^{aij}(k)$ are all zero mean with comparatively small cross-correlations. We also assume that $E\{u_{i,j;Q}^2(k-1)\}$ and $E\{[\Delta u_{i,j}(k-1)]^2\}$ can be replaced by $E\{u_{i,j;Q}^2(k)\}$ and $E\{[\Delta u_{i,j}(k)]^2\}$, respectively, by considering them stationary. Another assumption is that the mean squared value of quantities in finite and infinite-precision coincide. Therefore, using the assumptions (9.22) and (9.21), the resulting expression for the mean squared value of $\Delta a_{i,j}(k)$ is given by

$$\begin{aligned}
E\{[\Delta a_{i,j}(k)]^2\} = \sigma_x^2\lambda^{i-1} & \sum_{n=0}^{i-1} E\{[\Delta\cos\theta_n(k)]^2\} \\
& + \sum_{m=0}^{i-1} \lambda^{i-m} \left\{ E\{u_{m,j}^2(k)\} E\{[\Delta\sin\theta_m(k)]^2\} + E\{[\Delta u_{m,j}(k)]^2\}(1-\lambda) \right\} \\
& + \sum_{m=0}^{i-1} \left\{ E\{u_{m,j}^2(k)\}(1-\lambda) \sum_{n=m+1}^{i-1} E\{[\Delta\cos\theta_n(k)]^2\}\lambda^{i-m-1} \right\} \\
& + \frac{\lambda^i - 1}{\lambda - 1}\sigma_n^2, \qquad (9.61)
\end{aligned}$$

where σ_n^2 is the variance of $\eta_M^{aij}(k)$.

9.2.3.2 Mean squared value of $\Delta b_i(k)$

The values $b_i(k)$ correspond to the first element of the intermediate vectors resulting from the application of Givens rotations to vector $\hat{\mathbf{d}}_2(k)$. The form of deriving $\Delta b_i(k)$ is similar to $\Delta a_{i,j}(k)$ and the result is

$$\begin{aligned}
\Delta b_i(k) \approx d(k) \sum_{m=0}^{i-1} \Delta \cos\theta_m(k) \prod_{\substack{j=0\\ j\neq m}}^{i-1} \cos\theta_j(k) \\
& -\lambda^{1/2} \sum_{j=0}^{i-1} [\hat{d}_{2,j;Q}(k-1)\Delta \sin\theta_j(k) + \Delta \hat{d}_{2,j}(k-1)\sin\theta_j(k)] \cdot \\
& \prod_{m=j+1}^{i-1} \cos_Q \theta_m(k) - \lambda^{1/2} \sum_{j=0}^{i-1} \hat{d}_{2,j;Q}(k-1)\sin_Q \theta_j(k) \cdot \\
& \left\{ \sum_{p=j+1}^{i-1} \Delta \cos\theta_p(k) \prod_{\substack{m=j+1\\ m\neq p}}^{i-1} \cos_Q \theta_m(k) \right\} \\
& + \sum_{j=0}^{i-1} \eta_j^{bi}(k) \prod_{m=j+1}^{i-1} \cos_Q \theta_m(k),
\end{aligned} \tag{9.62}$$

where $\eta_j^{bi}(k) = \eta_M[(b_{i-1}(k), \cos\theta_{i-1}(k)); (\lambda^{1/2}\hat{d}_{2,i-1}(k-1), \sin\theta_{i-1}(k))]$. Using the assumption that $\Delta \cos\theta_i(k)$, $\Delta \sin\theta_i(k)$, and $\eta_j^{bi}(k)$ are all zero mean and have small cross-correlation with each other and also assuming that $E\{\hat{d}^2_{2,j;Q}(k-1)\} = E\{\hat{d}^2_{2,j;Q}(k)\}$ and $E\{[\Delta\hat{d}_{2,j;Q}(k-1)]^2\} = E\{[\Delta\hat{d}_{2,j;Q}(k)]^2\}$, it can be shown that

$$\begin{aligned}
E\{[\Delta b_i(k)]^2\} = \sigma_x^2 \|\mathbf{w}_o(k)\|^2 \lambda^{i-1} \sum_{m=0}^{i-1} E\{[\Delta \cos\theta_m(k)]^2\} \\
& + \sum_{j=0}^{i-1} \lambda^{i-j} \{E\{\hat{d}^2_{2,j}(k)\} E\{[\Delta \sin\theta_j(k)]^2\} + E\{[\Delta\hat{d}_{2,j}(k)]^2\}(1-\lambda)\} \\
& + \sum_{j=0}^{i-1} E\{\hat{d}^2_{2,j}(k)\}(1-\lambda) \sum_{p=j+1}^{i-1} E\{[\Delta \cos\theta_p(k)]^2\}\lambda^{i-j-1} \\
& + \frac{\lambda^i - 1}{\lambda - 1}\sigma_n^2,
\end{aligned} \tag{9.63}$$

where σ_n^2 is the variance of $\eta_j^{bi}(k)$, whereas $E\{d^2(k)\}$ was approximated by $\sigma_x^2\|\mathbf{w}_o(k)\|^2$ by neglecting the effects of the measurement noise in the desired signal. Vector $\mathbf{w}_o(k)$ represents the tap coefficients of the unknown model.

9.2.3.3 Mean squared value of $\Delta u_{i,i}(k)$

The value of $\Delta u_{i,i}(k)$ can be derived from (9.18) as follows:

$$\begin{aligned}\Delta u_{i,i}(k) = & \sqrt{\lambda u_{i,i}^2(k-1)+a_{i,i}^2(k)} \\ & -\sqrt{\lambda u_{i,i;Q}^2(k-1)+a_{i,i;Q}^2(k)-\eta_M[\lambda u_{i,i;Q}^2(k-1),a_{i,i;Q}^2(k)]} \\ & +\eta_S[\lambda u_{i,i;Q}^2(k-1)+a_{i,i}^2(k)]\end{aligned} \tag{9.64}$$

Considering that the equation above has a square-root operation, the following approximation can be used

$$\sqrt{r}-\sqrt{r-\Delta r}\approx\frac{\Delta r}{2\sqrt{r}}; \tag{9.65}$$

so that

$$\begin{aligned}E\{[\Delta u_{i,i}(k)]^2\} \approx & \lambda^2 E\{[\Delta u_{i,i}(k)]^2\}+(1-\lambda)E\{[\Delta a_{i,i}(k)]^2\} \\ & +\frac{\sigma_n^2}{4\sigma_x^2}(1-\lambda)\left[\frac{1+\lambda}{2\lambda}\right]^i+\sigma_n^2.\end{aligned} \tag{9.66}$$

9.2.3.4 Mean square value of $\Delta\sin\theta_i(k)$

For a division operation, the following approximation is valid for small Δr

$$\frac{1}{r+\Delta r}\approx\frac{1}{r}\left[1-\frac{\Delta r}{r}\right]. \tag{9.67}$$

In (9.20), by replacing $a_{i,i}(k)$ and $u_{i,i}(k)$, respectively, by $a_{i,i;Q}(k)+\Delta a_{i,i}(k)$ and $u_{i,i;Q}(k)+\Delta u_{i,i}(k)$, and using the approximation above, it is possible to show that

$$\Delta\sin\theta_i(k)=\frac{\Delta a_{i,i}(k)}{u_{i,i}(k)}+\frac{a_{i,i}(k)}{u_{i,i}^2(k)}\Delta u_{i,i}(k)+\eta_D(k), \tag{9.68}$$

where $\eta_D(k)$ represents $\eta_D[a_{i,i;Q}(k),u_{i,i;Q}(k)]$. Now, considering that the instantaneous and accumulated errors are zero mean and with relatively small cross-correlations, and using the averaging principle, it can be demonstrated that

$$\begin{aligned}E\{[\Delta\sin\theta_i(k)]^2\} \approx & \frac{E\{[\Delta a_{i,i}(k)]^2\}\{(1-\lambda)^3+(1-\lambda)\}}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^i \\ & +\frac{E\{[\Delta u_{i,i}(k)]^2\}\lambda^2(1-\lambda)^2}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^i \\ & +\frac{(1-\lambda)^3\sigma_n^2}{4\sigma_x^4}\left[\frac{1+\lambda}{2\lambda}\right]^{2i} \\ & +\frac{\sigma_n^2(1-\lambda)^2}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^i+\sigma_n^2.\end{aligned} \tag{9.69}$$

9.2.3.5 Mean squared value of $\Delta \cos\theta_i(k)$

The cosines of the Givens rotations in the infinite and finite-precision implementations of the QRD-RLS algorithm are respectively expressed by

$$\cos\theta_i(k) = \frac{\lambda^{1/2}u_{i,i;Q}(k-1)}{u_{i,i;Q}(k)} + \frac{\lambda^{1/2}\Delta u_{i,i}(k-1)}{u_{i,i;Q}(k)} \\ - \frac{\lambda^{1/2}u_{i,i;Q}(k-1)}{u^2_{i,i;Q}(k)}\Delta u_{i,i}(k) \tag{9.70}$$

and

$$\cos_Q\theta_i(k) = \frac{\lambda^{1/2}u_{i,i;Q}(k-1)}{u_{i,i;Q}(k)} - \eta_D(k), \tag{9.71}$$

where $\eta_D(k)$ represents $\eta_D[\lambda^{1/2}u_{i,i;Q}(k-1), u_{i,i;Q}(k)]$. With these equations, one can show that

$$\Delta\cos\theta_i(k) = \frac{\lambda^{1/2}\Delta u_{i,i}(k-1)}{u_{i,i}(k)} - \frac{\lambda^{1/2}u_{i,i}(k-1)}{u^2_{i,i}(k)}\Delta u_{i,i}(k) + \eta_D(k). \tag{9.72}$$

Thus, if we take the squared value of (9.72) and apply the expected value operation to the resulting equation, we obtain

$$E\{[\Delta\cos\theta_i(k)]^2\} \approx \lambda\frac{E\{[\Delta u_{i,i}(k-1)]^2\}}{E\{u^2_{i,i}(k)\}} + \lambda\frac{E\{[\Delta u_{i,i}(k)]^2\}}{E\{u^2_{i,i}(k)\}} \\ -2\lambda\frac{E\{\Delta u_{i,i}(k-1)\Delta u_{i,i}(k)\}}{E\{u^2_{i,i}(k)\}} + E\{\eta^2_D(k)\}. \tag{9.73}$$

It should be noted that $\Delta u_{i,i}(k-1)$ and $\Delta u_{i,i}(k)$ are not uncorrelated and that the mean of their product can be calculated as

$$\begin{aligned} E\{\Delta u_{i,i}(k-1)\Delta u_{i,i}(k)\} &\approx E\{\lambda\frac{u_{i,i}(k-1)}{u_{i,i}(k)}[\Delta u_{i,i}(k-1)]^2\} \\ &\approx \lambda E\{[\Delta u_{i,i}(k-1)]^2\} \\ &= \lambda E\{[\Delta u_{i,i}(k)]^2\}. \end{aligned} \tag{9.74}$$

From (9.73), (9.74), and (9.66) we get

$$\begin{aligned}E\{[\Delta\cos\theta_i(k)]^2\} \approx{}& \frac{\lambda(1-\lambda)^3E\{[\Delta u_{i,i}(k)]^2\}}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^i \\ &+\frac{\lambda(1-\lambda)^2E\{[\Delta a_{i,i}(k)]^2\}}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^i+\sigma_n^2 \\ &+\frac{\lambda(1-\lambda)^2\sigma_n^2}{4\sigma_x^4}\left[\frac{1+\lambda}{2\lambda}\right]^{2i}+\frac{\lambda(1-\lambda)\sigma_n^2}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^i. \qquad (9.75)\end{aligned}$$

9.2.3.6 Mean squared value of $\Delta u_{i,j}(k)$

For the infinite-precision implementation of the QRD-RLS algorithm, the elements of the triangularized matrix $\mathbf{U}(k)$ are calculated by

$$\begin{aligned}u_{i,j}(k) &= \lambda^{1/2}u_{i,j}(k-1)\cos\theta_i(k)+a_{i,j}(k)\sin\theta_i(k) \qquad (9.76)\\ &= [\lambda^{1/2}u_{i,j;Q}(k-1)+\lambda^{1/2}\Delta u_{i,j}(k-1)][\cos_Q\theta_i(k)+\Delta\cos\theta_i(k)]\\ &\quad+[a_{i,j;Q}(k)+\Delta a_{i,j}(k)][\sin_Q\theta_i(k)+\Delta\sin\theta_i(k)]\\ &= \lambda^{1/2}u_{i,j;Q}(k-1)\cos_Q\theta_i(k)+a_{i,j;Q}(k)\sin_Q\theta_i(k)\\ &\quad+\lambda^{1/2}\Delta u_{i,j}(k-1)\cos_Q\theta_i(k)\\ &\quad+\lambda^{1/2}u_{i,j;Q}(k-1)\Delta\cos\theta_i(k)\\ &\quad+a_{i,j;Q}(k)\Delta\sin\theta_i(k)+\Delta a_{i,j}(k)\sin_Q\theta_i(k). \qquad (9.77)\end{aligned}$$

In finite-precision implementation the elements of $\mathbf{U}_Q(k)$ are given by

$$\begin{aligned}u_{i,j;Q}(k) &= \lambda^{1/2}u_{i,j;Q}(k-1)\cos_Q\theta_i(k)\\ &\quad+a_{i,j;Q}(k)\sin_Q\theta_i(k)-\eta_M(k), \qquad (9.78)\end{aligned}$$

where $\eta_M(k)$ represents $\eta_M[(\lambda^{1/2}u_{i,j;Q}(k-1),\cos_Q\theta_i(k));(a_{i,j;Q}(k),\sin_Q\theta_i(k))]$. Subtracting (9.77) from (9.78), and replacing the quantities in finite precision by their infinite precision counterpart, we obtain

$$\begin{aligned}\Delta u_{i,j}(k) &= \lambda^{1/2}\Delta u_{i,j}(k-1)\cos\theta_i(k)+\lambda^{1/2}u_{i,j}(k-1)\Delta\cos\theta_i(k)\\ &\quad+a_{i,j}(k)\Delta\sin\theta_i(k)+\Delta a_{i,j}(k)\sin\theta_i(k)+\eta_M(k). \qquad (9.79)\end{aligned}$$

Assuming that $\eta_M(k)$ as well as the accumulated errors are zero mean with relatively small cross-correlations, then

$$\begin{aligned}E\{[\Delta u_{i,j}(k)]^2\} \approx{}& \lambda E\{[\Delta u_{i,j}(k-1)]^2\}E\{\cos^2\theta_i(k)\}\\ &+\lambda E\{u_{i,j}^2(k-1)\}E\{[\Delta\cos\theta_i(k)]^2\}\\ &+E\{a_{i,j}^2(k)\}E\{[\Delta\sin\theta_i(k)]^2\}\\ &+E\{[\Delta a_{i,j}(k)]^2\}E\{\sin^2\theta_i(k)\}+E\{\eta_M^2(k)\}. \qquad (9.80)\end{aligned}$$

Using (9.22) and (9.21), in the steady-state we get

$$\begin{aligned} E\{[\Delta u_{i,j}(k)]^2\} \approx{}& \frac{\lambda E\{u_{i,j}^2(k-1)\}E\{[\Delta \cos\theta_i(k)]^2\}}{1-\lambda^2} \\ &+\frac{E\{a_{i,j}^2(k)\}E\{[\Delta \sin\theta_i(k)]^2\}}{1-\lambda^2} \\ &+\frac{E\{[\Delta a_{i,j}(k)]^2\}}{1+\lambda}+\frac{E\{\eta_M^2(k)\}}{1-\lambda^2}. \end{aligned} \tag{9.81}$$

The equation above can be simplified into two different ways. For $i \neq j$, we can apply (9.23) and (9.27) resulting in

$$\begin{aligned} E\{[\Delta u_{i,j}(k)]^2\} \approx{}& \frac{\lambda \sigma_x^2 E\{[\Delta \cos\theta_i(k)]^2\}}{(1-\lambda)(1-\lambda^2)}\left[\frac{2\lambda}{1+\lambda}\right]^i \\ &+\frac{\sigma_x^2 E\{[\Delta \sin\theta_i(k)]^2\}}{1-\lambda^2}\left[\frac{2\lambda}{1+\lambda}\right]^i \\ &+\frac{E\{[\Delta a_{i,j}(k)]^2\}}{1+\lambda}+\frac{\sigma_n^2}{1-\lambda^2}, \end{aligned} \tag{9.82}$$

where σ_n^2 here is the variance of $\eta_M(k)$. For $i = j$, we have to substitute (9.69), (9.75), (9.23), and (9.24) in (9.81) in order to derive, after some manipulation

$$\begin{aligned} E\{[\Delta u_{i,i}(k)]^2\} \approx{}& \frac{2\lambda^2-2\lambda+3}{2\lambda^2-\lambda+1}E\{[\Delta a_{i,i}(k)]^2\} \\ &+\frac{3\lambda^2-4\lambda+2}{2\lambda^2-\lambda+1}\frac{\sigma_n^2}{4\sigma_x^2}\left[\frac{2\lambda}{1+\lambda}\right]^i \\ &+\frac{(2\lambda^2-2\lambda+2)\sigma_n^2}{(2\lambda^2-\lambda+1)(1-\lambda)} \\ &+\frac{\sigma_n^2\sigma_x^2}{(1-\lambda)^2(2\lambda^2-\lambda+1)}\left[\frac{1+\lambda}{2\lambda}\right]^i. \end{aligned} \tag{9.83}$$

9.2.3.7 Mean squared value of $\Delta\hat{d}_{2,i}(k)$

The elements of vector $\hat{\mathbf{d}}_2(k)$ are resultant of the application of $N+1$ Givens rotations to $\lambda^{1/2}\mathbf{d}_2(k-1)$, that is

$$\hat{d}_{2,i}(k) = \lambda^{1/2}\hat{d}_{2,i}(k-1)\cos\theta_i(k) + b_i(k)\sin\theta_i(k). \tag{9.84}$$

This equation is similar to (9.76); as a consequence, by following the same steps to derive (9.79), we can show that

$$
\begin{aligned}
E\{[\Delta \hat{d}_{2,i}(k)]^2\} \approx\ & \frac{\lambda E\{\hat{d}_{2,i}^2(k)\}E\{[\Delta \cos\theta_i(k)]^2\}}{1-\lambda^2} \\
& + \frac{E\{b_i^2(k)\}E\{[\Delta \sin\theta_i(k)]^2\}}{1-\lambda^2} \\
& + \frac{E\{[\Delta b_i(k)]^2\}}{1+\lambda} + \frac{\sigma_n^2}{1-\lambda^2}. \qquad (9.85)
\end{aligned}
$$

9.2.3.8 Mean squared value of $\Delta e(k)$

The error signal in the infinite- and finite-precision implementations are given by

$$
e(k) = e_q(k)\cos\theta_N(k)\cdots\cos\theta_0(k) \qquad (9.86)
$$

and

$$
e_Q(k) = \mathrm{Q}[e_{q;Q}(k)\mathrm{Q}[\cos_Q\theta_0(k)\cdots\mathrm{Q}[\cos_Q\theta_N(k)\cos_Q\theta_{N-1}(k)]\cdots]]], \qquad (9.87)
$$

respectively.
From (9.47), the *a posteriori* error signal $e(k)$ can be expressed as

$$
\begin{aligned}
e(k) &= [e_{q;Q}(k) + \Delta e_q(k)][\cos\theta_N(k) + \Delta\cos\theta_N(k)]\cdots \\
&\quad\ [\cos_Q\theta_1(k) + \Delta\cos\theta_1(k)][\cos_Q\theta_0(k) + \Delta\cos\theta_0(k)] \\
&= e_{q;Q}(k)\cos_Q\theta_N(k)\cdots\cos_Q\theta_0(k) + \Delta e_q(k)\cos\theta_N(k)\cdots\cos\theta_0(k) \\
&\quad + e_{q,Q}(k)\left[\sum_{i=0}^{N}\Delta\cos_Q\theta_i(k)\prod_{\substack{j=0\\ i\neq j}}^{N}\cos\theta_j(k)\right] \\
&\quad + \Delta e_{q;Q}(k)\cos_Q\theta_N(k)\cdots\cos_Q\theta_0(k), \qquad (9.88)
\end{aligned}
$$

where, in the last expression, the error terms of the second and higher order were ignored.

The application of (9.41) to the multiplication operations of (9.16) yields

$$
\begin{aligned}
e_Q(k) \approx\ & e_{q;Q}(k)\cos_Q\theta_N(k)\cos_Q\theta_{N-1}(k)\cdots\cos_Q\theta_1(k)\cos_Q\theta_0(k) \\
& - e_{q;Q}(k)\left[\sum_{i=0}^{N}\prod_{j=i+1}^{N}\cos_Q\theta_j(k)\eta_i^e(k)\right] - \eta_{N+1}^e(k). \qquad (9.89)
\end{aligned}
$$

By replacing (9.88) and (9.89) in the definition (9.47), $\Delta e(k)$ results in

$$\begin{aligned}\Delta e(k) \approx\ & e_{q;Q}(k)\left[\sum_{i=0}^{N}\Delta\cos\theta_i(k)\prod_{\substack{j=0\\ i\neq j}}^{N}\cos\theta_j(k)\right]\\ &+\Delta e_{q;Q}(k)\cos_Q\theta_0(k)\cdots\cos_Q\theta_N(k)\\ &+e_{q;Q}(k)\left[\sum_{i=0}^{N}\prod_{j=i+1}^{N}\cos_Q\theta_j(k)\eta_i^e(k)\right]-\eta_{N+1}^e(k).\end{aligned}\tag{9.90}$$

We assume that $\Delta e_{q;Q}(k)$, $e_q(k)$, $\Delta\cos\theta_i(k)$, and $\eta_i^e(k)$, for $i=0,\ldots,N+1$, are all zero mean with relatively small cross-correlation between each other. Also, the variance $\eta_i^e(k)$ is considered to be σ_n^2 (that is the variance of the quantization noise), and $E\{\cos^2\theta_i(k)\}\approx\lambda$.

With the above assumptions, one can show that the expected value of the accumulated quantization error in the *a posteriori* error signal is given by

$$\begin{aligned}E\{[\Delta e(k)]^2\} \approx\ & E\{e_{q;Q}^2(k)\}\left[\sum_{i=0}^{N}E\{[\Delta\cos\theta_i(k)]^2\}\lambda^N\right]\\ &+E\{[\Delta e_q(k)]^2\}\lambda^{N+1}+E\{e_{q;Q}^2(k)\}\sum_{i=0}^{N}\lambda^{N-i}\sigma_n^2+\sigma_n^2.\end{aligned}\tag{9.91}$$

In the first term of the right-hand-side of the equation above, we can substitute $E\{e_q^2(k)\}$ as suggested in (9.35). In the third term, (9.35) should also be applied. In the second term, if it is noted that $e_q(k)$ is the first element of $\mathbf{d}_2(k)$, from (9.63), we can determine $E\{[\Delta e_q(k)]^2\}$ by setting $i=N+1$, i.e.,

$$\begin{aligned}E\{[\Delta e_q(k)]^2\} \approx\ & \sigma_x^2\|\mathbf{w}_o(k)\|^2\lambda^N\sum_{m=0}^{N}E\{[\Delta\cos\theta_m(k)]^2\}\\ &+\sum_{j=0}^{N}\lambda^{N+1-j}\left[E\{\hat{d}_{2,j}^2(k)\}E\{[\Delta\sin\theta_j(k)]^2\}+E\{[\Delta\hat{d}_{2,j}(k)]^2\}(1-\lambda)\right]\\ &+\sum_{j=0}^{N}E\{\hat{d}_{2,j}^2(k)\}(1-\lambda)\sum_{p=j+1}^{N}E\{[\Delta\cos\theta_p(k)]^2\}\lambda^{N-j}\\ &+\frac{\lambda^{N+1}-1}{\lambda-1}\sigma_n^2.\end{aligned}\tag{9.92}$$

9.2.3.9 Mean squared value of $\Delta w_i(k)$

The tap coefficients of the adaptive filter in the QRD-RLS algorithm are calculated through the back-substitution algorithm, as illustrated in (9.14) and (9.15). After some manipulations, it can be shown that

$$\begin{aligned}\Delta w_i(k) = & \frac{\hat{d}_{2,i}(k) - \sum_{j=i+1}^{N} w_i(k) u_{i,j}(k)}{u_{i,i}(k)} \\ & - \frac{\hat{d}_{2,i;Q}(k) - \sum_{j=i+1}^{N} w_{i;Q}(k) u_{i,j;Q}(k) + \eta_M(k)}{u_{i,i;Q}(k)} + \eta_D(k),\end{aligned} \tag{9.93}$$

where in the above equation we have

$$\eta_M(k) \triangleq \eta_M \left[\sum_{j=i+1}^{N} w_{i;Q}(k) u_{i,j;Q}(k) \right] \tag{9.94}$$

and

$$\eta_D(k) \triangleq \eta_D \left[\left(\hat{d}_{2,i;Q}(k) - \sum_{j=i+1}^{N} w_{i;Q}(k) u_{i,j;Q}(k) + \eta_8(k) \right), u_{i,i;Q}(k) \right]. \tag{9.95}$$

From the expression above and using the approximation in (9.67), we obtain

$$\begin{aligned}\Delta w_i(k) \approx & \frac{\Delta \hat{d}_{2,i}(k) - \sum_{j=i+1}^{N} [w_{j;Q}(k) \Delta u_{i,j}(k) + \Delta w_j(k) u_{i,j;Q}(k)]}{u_{i,i;Q}(k)} \\ & - \frac{w_{i;Q}(k) \Delta u_{i,i}(k)}{u_{i,i;Q}(k)} + \frac{\eta_M(k)}{u_{i,i;Q}(k)} + \eta_D(k),\end{aligned} \tag{9.96}$$

where, in the last expression, we replaced the finite-precision quantities by their infinite-precision counterparts. The introduced errors are of second order, and can therefore be neglected. Assuming that $\eta_M(k)$ and $\eta_D(k)$ are uncorrelated and zero mean; employing the averaging principle [9], it can be shown that

$$\begin{aligned}E\{[\Delta w_i(k)]^2\} \approx & \frac{E\{[\Delta \hat{d}_{2,i}(k)]^2\}}{E\{u_{i,i}^2(k)\}} + \frac{\sum_{j=i}^{N} E\{w_j^2(k)\} E\{[\Delta u_{i,j}(k)]^2\}}{E\{u_{i,i}^2(k)\}} \\ & + \frac{\sum_{j=i+1}^{N} E\{[\Delta w_j(k)]^2\} E\{u_{i,j}^2(k)\}}{E\{u_{i,i}^2(k)\}} \\ & + \frac{\sigma_n^2}{E\{u_{i,i}^2(k)\}} + \sigma_n^2.\end{aligned} \tag{9.97}$$

In order to calculate $E\{[\Delta w_i(k)]^2\}$, it is necessary to have $E\{[\Delta u_{i,j}(k)]^2\}$ and $E\{[\Delta d_{2,i}(k)]^2\}$, that in turn require the values of $E\{[\Delta a_{i,i}(k)]^2\}$, $E\{[\Delta b_i(k)]^2\}$, $E\{[\Delta u_{i,i}(k)]^2\}$, $E\{[\Delta \sin \theta_i(k)]^2\}$, and $E\{[\Delta \cos \theta_i(k)]^2\}$.

From (9.97), we can determine the mean of the squared norm of the deviation in the tap coefficients as follows:

$$E\{\|\Delta \mathbf{w}(k)\|^2\} = \sum_{i=0}^{N} E\{[\Delta w_i(k)]^2\} \tag{9.98}$$

9.2.4 Simulation results

The derived equations were verified through simulations using a system identification application where both input signal and measurement noise were pseudo-random sequences with normal distribution and zero-mean. Four different moving-averaging processes were utilized to emulate unknown systems with orders equal to 4, 6, 8, and 10, respectively. In all simulations, the QRD-RLS algorithm ran for 1500 iterations and the simulation results were obtained by averaging the results of 100 independent runs. Infinite-precision simulations were executed with 64 bits floating-point arithmetic. In the fixed-point implementation, the quantities were represented by numbers with magnitude less than unity. Frequent overflow was avoided by choosing the input signal variance appropriately.

The first experiment was aimed to verify the results for different moving average processes. The input signal variance was fixed at -30 dB, while the additional noise variance was -70 dB. The forgetting factor was $\lambda = 0.95$ and the wordlength was 15 bits. The measured and calculated results for $E\{\|\mathbf{w}(k)\|_2^2\}$ and $E\{[\Delta e(k)]^2\}$ are presented in Table 9.2, and it can be seen that simulated and calculated values are in close agreement.

Table 9.2 Fixed-point environment: simulations for distinct MA processes with 15 bits and forgetting factor $\lambda = 0.95$.

MA	$E\{\|\Delta\mathbf{w}(k)\|_2^2\}$ (dB)		$E\{[\Delta e(k)]^2\}$ (dB)	
process	Simulated	Calculated	Simulated	Calculated
MA1	−64.7	−64.3	−92.6	−92.2
MA2	−62.7	−62.2	−91.7	−91.3
MA3	−61.2	−60.7	−91.2	−90.8
MA4	−59.9	−59.2	−90.9	−90.5

The formulas (9.91) and (9.98) were tested for different values of λ, $\sigma_x^2 = -25$ dB and $\sigma_r^2 = -70$ dB. The wordlength again was 15 bits. The results are presented in Table 9.3. Again, we observe close agreement between the calculated and simulated values.

Table 9.3 Simulations for distinct λ.

Forgetting factor (λ)	$E\{\|\Delta\mathbf{w}(k)\|_2^2\}$ (dB)		$E\{[\Delta e(k)]^2\}$ (dB)	
	Simulated	Calculated	Simulated	Calculated
0.90	−68.2	−67.7	−93.7	−93.2
0.93	−69.0	−68.7	−93.0	−92.6
0.95	−69.7	−69.3	−92.5	−92.2
0.97	−70.3	−70.1	−92.1	−91.8
0.99	−70.9	−70.7	−91.6	−91.3

The theoretical results were also verified for different wordlengths with $\sigma_x^2 = -30$ dB, $\lambda = 0.95$ and $\sigma_r^2 = -70$ dB. Table 9.4 illustrates the results. As can be noted in the current and in all previous experiments, the obtained formulas are shown to model accurately the finite wordlength effects in the main quantities of the QRD-RLS algorithm.

Table 9.4 Simulations for distinct precisions.

Number of bits	$E\{\|\Delta\mathbf{w}(k)\|_2^2\}$ (dB)		$E\{[\Delta e(k)]^2\}$ (dB)	
	Simulated	Calculated	Simulated	Calculated
12	−46.5	−46.3	−74.1	−74.1
15	−68.2	−64.3	−91.7	−92.2
20	−94.8	−94.4	−122.3	−122.3
25	−124.9	−124.6	−152.8	−152.4
30	−154.0	−154.7	−182.8	−182.5

9.3 Precision Analysis of the Fast QRD-Lattice Algorithm

This section discusses the finite-precision analysis for the FQRD-lattice algorithm proposed by McWhirter [10]. The notation of this reference was followed. The C Language pseudo-code for the FQRD-lattice algorithm is shown in Table 9.5, which details all algorithmic steps labeled from step (S.1) through (S.18). It can be seen that this algorithm takes advantage of two operations named *rotor* and *cisor* for performing all internal computations.

Exploring the fact that only two basic operations are performed by this algorithm, a very regular structure can be derived as shown in Figure 9.1. This figure uses squares to represent *rotor* cells (that perform rotations) and circles to represent *cisor* cells (that perform cosine/sine calculations). The small cells in the last stage represent multipliers.

Table 9.5 C Language pseudo-code for the FQRD-lattice algorithm.

FQRD-Lattice RLS [10]

```
void rotor (double $x_{in}$, double $y_{in}$, double $x_{out}$, double $y_{out}$, double $c_{in}$, double $s_{in}$)
{
    $x_{out} = \mathrm{Q}[\lambda^{1/2} x_{in} c_{in} + y_{in} s_{in}]$;                                   (S.1)
    $y_{out} = \mathrm{Q}[-\lambda^{1/2} x_{in} s_{in} + y_{in} c_{in}]$;                                  (S.2)
}

void cisor (double $x_{in}$, double $y_{in}$, double $x_{out}$, double $b_{in}$, double $b_{out}$, double $c_{out}$,
double $s_{out}$)
{
    double aux;

    $x_{out} = \mathrm{Q}[\sqrt{\mathrm{Q}[\lambda x_{in}^2 + y_{in}^2]}]$;                                  (S.3)
    $c_{out} = \mathrm{Q}[\lambda^{1/2} x_{in} / x_{out}]$;                                       (S.4)
    $s_{out} = \mathrm{Q}[y_{in} / x_{out}]$;                                                (S.5)
    $b_{out} = \mathrm{Q}[b_{in} c_{out}]$;                                                  (S.6)
}

void FqrdLattice (double $x(k)$, double $d(k)$, double $e(k)$)
{
    int i;
    double aux;

    $e_0^f(k) = e_0^b(k) = x(k)$;                                            (S.7)
    $e_0(k) = d(k)$;                                                         (S.8)
    $\alpha_0(k) = 1.0$;                                                     (S.9)
    for (i=1; i $\leq$ N+1; i++)
    {
        cisor ($\alpha_{i-1}^b(k-1), e_{i-1}^b(k), \alpha_{i-1}^b(k), \alpha_{i-1}(k), \alpha_i(k), c_i^f(k), s_i^f(k)$);      (S.10)
        rotor ($\beta_{i-1}^f(k-1), e_{i-1}^f(k), \beta_{i-1}^f(k), e_i^f(k), c_i^f(k-1), s_i^f(k-1)$);        (S.11)
        rotor ($\beta_{i-1}(k-1), e_{i-1}(k), \beta_{i-1}(k), e_i(k), c_i^f(k), s_i^f(k)$);              (S.12)
        $\varepsilon_i^f(k) = \mathrm{Q}[\alpha_i(k-1) e_i^f(k)]$;                                  (S.13)
        $\varepsilon_i(k) = \mathrm{Q}[\alpha_i(k) e_i(k)]$;                                        (S.14)
        cisor ($\alpha_{i-1}^f(k-1), e_{i-1}^f(k), \alpha_{i-1}^f(k), aux, aux, c_i^b(k), s_i^b(k)$);        (S.15)
        rotor ($\beta_{i-1}^b(k-2), e_{i-1}^b(k), \beta_{i-1}^b(k-1), e_i^b(k), c_i^b(k), s_i^b(k)$);       (S.16)
        $\varepsilon_i^b(k) = \mathrm{Q}[\alpha_i(k) e_i^b(k)]$;                                     (S.17)
    }
    $e(k) = \mathrm{Q}[\alpha_{N+1}(k) e_{N+1}(k)]$;                                      (S.18)
}
```

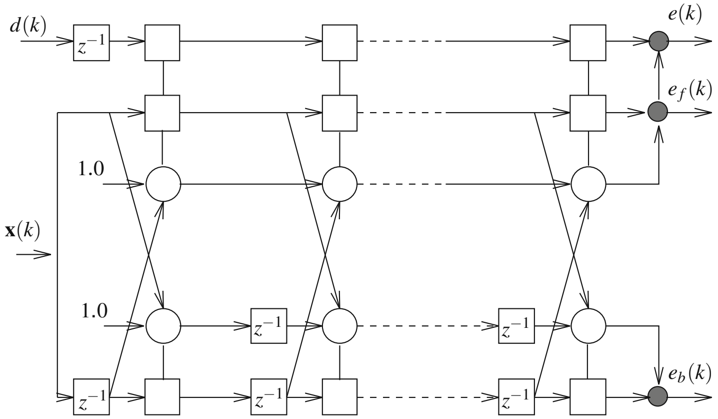


Fig. 9.1 Structure representing the FQRD-lattice algorithm.

9.3.1 Infinite-precision analysis

This section derives mean squared values of the internal variables in the FQRD-lattice algorithm. They are of key importance for the finite-precision analysis that will be performed in the next subsection.

9.3.1.1 Mean squared values of $c_i^f(k)$ and $s_i^f(k)$

Previous studies [11] have shown that the mean squared values of the forward recursion cosines and sines are

$$E\{[c_i^f(k)]^2\} = \lambda, \text{ and} \tag{9.99}$$

$$E\{[s_i^f(k)]^2\} = 1-\lambda. \tag{9.100}$$

Simulations for these variables in the QRD-RLS and for the FQRD-lattice algorithms indicate that these approximations are reasonable.

9.3.1.2 Mean squared values of $\beta_i^f(k)$, $e_i^f(k)$, and $\alpha_i^f(k)$

Step (S.11) of the FQRD-lattice algorithm implies that

$$\beta_i^f(k) = \lambda^{1/2} c_i^f(k-1)\beta_{i-1}^f(k-1) + s_i^f(k-1)e_{i-1}^f(k), \text{ and} \tag{9.101}$$

$$e_i^f(k) = -\lambda^{1/2} s_i^f(k-1)\beta_{i-1}^f(k-1) + c_i^f(k-1)e_{i-1}^f(k). \tag{9.102}$$

If it is supposed that the sines and cosines of the previous equations are uncorrelated with each other, and that the values of the sines are zero-mean, then it is possible to obtain the following relations:

$$\begin{aligned} E\{[\beta_i^f(k)]^2\} &= \lambda E\{[c_i^f(k-1)]^2\}E\{[\beta_{i-1}^f(k-1)]^2\} \\ &\quad + E\{[s_i^f(k-1)]^2\}E\{[e_{i-1}^f(k)]^2\}, \end{aligned} \tag{9.103}$$

$$\begin{aligned} E\{[e_i^f(k)]^2\} &= \lambda E\{[s_i^f(k)]^2\}E\{[\beta_{i-1}^f(k)]^2\} \\ &\quad + E\{[c_i^f(k)]^2\}E\{[e_{i-1}^f(k)]^2\}. \end{aligned} \tag{9.104}$$

Substituting relations (9.99) and (9.100) in (9.103), it is possible to show that

$$E\{[\beta_{i-1}^f(k)]^2\} = \frac{E\{[e_{i-1}^f(k)]^2\}}{1+\lambda}. \tag{9.105}$$

Substituting relations (9.99), (9.100), and (9.105) on (9.104), we obtain

$$E\{[e_i^f(k)]^2\} = \frac{2\lambda}{1+\lambda}E\{[e_{i-1}^f(k)]^2\}. \tag{9.106}$$

Since $e_0^f(k) = x(k)$, according to step (S.10), it follows that

$$E\{[e_i^f(k)]^2\} = \sigma_x^2\left[\frac{2\lambda}{1+\lambda}\right]^i. \tag{9.107}$$

Consequently, according to (9.105),

$$E\{[\beta_i^f(k)]^2\} = \frac{\sigma_x^2}{1+\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^i. \tag{9.108}$$

The recursion formula for $\alpha_{i-1}^f(k)$ is given by

$$\alpha_{i-1}^f(k) = \sqrt{\lambda[\alpha_{i-1}^b(k-1)]^2 + [e_{i-1}^f(k)]^2}, \tag{9.109}$$

according to step (S.15). Supposing that $\alpha_{i-1}^f(k)$ and $e_{i-1}^f(k)$ are stationary for $k \rightarrow \infty$, it follows that

$$E\{[\alpha_i^f(k)]^2\} = \frac{E\{[e_i^f(k)]^2\}}{1-\lambda}. \tag{9.110}$$

Using (9.107), the following expression results:

$$E\{[\alpha_i^f(k)]^2\} = \frac{\sigma_x^2}{1-\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^i. \tag{9.111}$$

9.3.1.3 Mean squared values of $c_i^b(k)$ and $s_i^b(k)$

According to the algorithm step (S.15), it follows that sines and cosines are calculated by

$$c_i^b(k) = \frac{\lambda^{1/2}\alpha_{i-1}^f(k-1)}{\alpha_{i-1}^f(k)}, \text{ and} \tag{9.112}$$

$$s_i^b(k) = \frac{e_{i-1}^f(k)}{\alpha_{i-1}^f(k)}. \tag{9.113}$$

Thus, the mean squared values of the backward recursion sines and cosines are

$$E\{[c_i^b(k)]^2\} = \frac{\lambda E\{[\alpha_{i-1}^f(k-1)]^2\}}{E\{[\alpha_{i-1}^f(k)]^2\}}, \text{ and} \tag{9.114}$$

$$E\{[s_i^b(k)]^2\} = \frac{E\{[e_{i-1}^f(k)]^2\}}{E\{[\alpha_{i-1}^f(k)]^2\}}. \tag{9.115}$$

In the above equations, the averaging principle [9] was used. Considering that $\alpha_{i-1}^f(k)$ is statistically stationary as $k \to \infty$ and using the fundamental trigonometric relation, it follows

$$E\{[c_i^b(k)]^2\} = \lambda, \tag{9.116}$$

$$E\{[s_i^b(k)]^2\} = 1-\lambda. \tag{9.117}$$

Surprisingly, these mean square values are the same as the ones for the forward recursion sines and cosines, and different from those in the fast QRD-RLS proposed by Bellanger [10, 12].

9.3.1.4 Mean squared values of $\beta_i^b(k)$, $e_i^b(k)$, and $\alpha_i^b(k)$

The relations for $\beta_i^b(k)$ and $e_i^b(k)$ derived from step (S.16) are totally analogous to the ones for $\beta_i^f(k)$ and $e_i^f(k)$ shown in (9.101) and (9.102). Considering that the mean squared values for cosines and sines are the same in the backward and forward rotations and using the same statistical independence assumptions, it is possible to obtain the following expressions:

$$E\{[e_i^b(k)]^2\} = \sigma_x^2 \left[\frac{2\lambda}{1+\lambda}\right]^i \tag{9.118}$$

$$E\{[\beta_i^b(k)]^2\} = \frac{\sigma_x^2}{1+\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^i \tag{9.119}$$

$$E\{[\alpha_i^b(k)]^2\} = \frac{\sigma_x^2}{1-\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^i . \tag{9.120}$$

9.3.1.5 Mean squared values of $\beta_i(k)$ and $e_i(k)$

Using properties of the triangularized input signal matrix [11], a very simple relationship for the mean square value of $\beta_i(k)$ can be derived. It is supposed that the reference input $d(k)$ is an MA process added with white Gaussian measurement noise $r(k)$ so that $d(k) = w^o(k) * x(k) + r(k)$. In this case, $w^o(k)$ is a sequence with the coefficients of the MA process with non-zero values for $k = 0, \ldots, N$.

$$E\{\beta_i^2(k)\} = \left[\frac{2\lambda}{1+\lambda}\right]^i \left[\frac{\sigma_x^2}{1-\lambda}[w_i^o]^2 + \frac{\sigma_x^2}{1+\lambda}\sum_{j=i+1}^{N}[w_j^o]^2\right] \tag{9.121}$$

Using the norm conservation property of Givens rotations, a relation between $E\{e_i^2(k)\}$ and $E\{\beta_i^2(k)\}$ can be derived as follows:

$$E\{e_i^2(k)\} = \sigma_x^2\|\mathbf{w}^o\|^2 + (\lambda-1)\sum_{j=0}^{i-1} E\{\beta_i^2(k)\}, \tag{9.122}$$

where $\mathbf{w}^o$ is a vector with $N+1$ entries with the sequence $w^o(k)$, $k = 0, \ldots, N$.

9.3.2 *Finite-precision analysis*

9.3.2.1 Mean squared value of $\Delta\alpha_i^b(k)$

According to step (S.10), the finite-precision version of $\alpha_i^b(k)$, denoted by $\alpha_{i;Q}^b(k)$, can be modeled as

$$\begin{aligned}\alpha_{i-1;Q}^b(k) = {} & \sqrt{\lambda[\alpha_{i-1;Q}^b(k-1)]^2 + [e_{i-1;Q}^f(k)]^2 + \eta_M(k)} \\ & + \eta_S(k),\end{aligned} \tag{9.123}$$

where $\eta_M(k)$ and $\eta_S(k)$ are instantaneous quantization errors due to multiplication and square-root operations. Considering only first order terms, it is possible to obtain

$$\begin{aligned}\Delta\alpha^b_{i-1}(k) = {} & \frac{\lambda\alpha^b_{i-1}(k-1)\Delta\alpha^b_{i-1}(k-1)+e^b_{i-1}(k)\Delta e^b_{i-1}(k)}{\alpha^b_{i-1}(k)} \\ & -\frac{\eta_M(k)}{\alpha^b_{i-1}(k)}+\eta_S(k). \end{aligned} \tag{9.124}$$

Squaring the above equation, supposing that the deviations and instantaneous quantization noises are all zero mean and uncorrelated with each other, and substituting Equations (9.118) and (9.120) it follows that

$$\begin{aligned}E\{[\Delta\alpha^b_{i-1}(k)]^2\} = {} & \frac{E[\Delta e^b_{i-1}(k)]^2}{1+\lambda}+\frac{\sigma_n^2}{4\sigma_x^2}\frac{1}{1+\lambda}\left[\frac{1+\lambda}{2\lambda}\right]^{i-1} \\ & +\frac{\sigma_n^2}{1-\lambda^2}. \end{aligned} \tag{9.125}$$

The averaging principle [9] was used on the derivation.

9.3.2.2 Mean squared values of $\Delta s^f_i(k)$ and $\Delta c^f_i(k)$

Using relations derived from step (S.15) and first-order approximations it is possible to write

$$E\{[\Delta s^f_i(k)]^2\} = E\left[\frac{\Delta e^b_{i-1}(k)}{\alpha^b_{i-1}(k)}-\frac{e^b_{i-1}(k)\Delta\alpha^b_{i-1}(k)}{[\alpha^b_{i-1}(k)]^2}+\eta_D(k)\right]^2. \tag{9.126}$$

Using only first-order terms, supposing that the deviations and quantization noise are all zero mean and uncorrelated with each other, and using the averaging principle [9], it is possible to derive

$$\begin{aligned}E\{[\Delta s^f_i(k)]^2\} = {} & \frac{E[\Delta e^b_{i-1}(k)]^2}{\sigma_x^2}(1-\lambda)\left[\frac{1+\lambda}{2\lambda}\right]^{i-1}+\sigma_n^2 \\ & +E\{[\Delta\alpha^b_{i-1}(k)]^2\}\sigma_x^2(1-\lambda)^2\left[\frac{1+\lambda}{2\lambda}\right]^{i-1}. \end{aligned} \tag{9.127}$$

The same methodology can be used to obtain the mean squared value of $\Delta c^f_i(k)$. The result is

$$E\{[\Delta c^f_i(k)]^2\} = \frac{2\lambda(1-\lambda^2)}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^{i-1}E\{[\Delta\alpha^b_{i-1}(k)]^2\}+\sigma_n^2. \tag{9.128}$$

9.3.2.3 Mean squared values of $\Delta\beta_i^f(k)$ and $\Delta e_i^f(k)$

The evolution of $\beta_i^f(k)$ is described by (9.103). If second-order errors are neglected, it is possible to write

$$\begin{aligned}\Delta\beta_{i-1}^f(k) = {}& \lambda^{1/2}c_i^f(k-1)\Delta\beta_{i-1}^f(k-1)\\ &+\lambda^{1/2}\Delta c_i^f(k-1)\beta_{i-1}^f(k-1)+\Delta s_i^f(k-1)e_{i-1}^f(k)\\ &+s_i^f(k-1)\Delta e_{i-1}^f(k)+\eta_M(k).\end{aligned}\tag{9.129}$$

Supposing that all the deviations and the instantaneous quantization noise are zero mean and uncorrelated with each other, it is possible to get

$$\begin{aligned}E\{[\Delta\beta_{i-1}^f(k)]^2\} = {}& \frac{\lambda\sigma_x^2}{(1+\lambda)(1-\lambda^2)}\left[\frac{2\lambda}{1+\lambda}\right]^{i-1}E\{[\Delta c_i^f(k)]^2\}\\ &+\frac{\sigma_n^2}{1-\lambda^2}+\frac{\sigma_x^2}{1-\lambda^2}\left[\frac{2\lambda}{1+\lambda}\right]^{i-1}E\{[\Delta s_i^f(k)]^2\}\\ &+E\{[\Delta s_i^f(k)]^2\}+\frac{E\{[\Delta e_{i-1}^f(k)]^2\}}{1+\lambda}.\end{aligned}\tag{9.130}$$

Using Equation (9.104) and following the same steps, it can be shown that

$$\begin{aligned}E\{[\Delta e_i^f(k)]^2\} = {}& \sigma_x^2\left[\frac{2\lambda}{1+\lambda}\right]^{i-1}E\{[\Delta c_i^f(k)]^2\}+\lambda E\{[\Delta e_{i-1}^f(k)]^2\}\\ &+E\{[\Delta s_i^f(k)]^2\}\sigma_x^2\frac{\lambda}{1+\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^{i-1}\\ &+\lambda(1-\lambda)E\{[\Delta s_i^f(k)]^2\}+\sigma_n^2.\end{aligned}\tag{9.131}$$

9.3.2.4 Mean squared values of $\Delta\varepsilon_i^f(k)$, $\Delta\alpha_i(k)$, and $\Delta\varepsilon_i(k)$

According to step (S.13), it is possible to write

$$\varepsilon_i^f(k) = \alpha_i(k-1)e_i^f(k).\tag{9.132}$$

Using the same methodology of previous derivations, it is possible to obtain

$$E\{[\Delta\varepsilon_i^f(k)]^2\} = \sigma_x^2\left[\frac{2\lambda}{1+\lambda}\right]^{i}E\{[\Delta\alpha_i(k)]^2\}+\lambda^i E\{[\Delta e_i^f(k)]^2\}+\sigma_n^2.\tag{9.133}$$

Step (S.10) implies that

$$\alpha_i(k) = c_i^f(k-1)\alpha_{i-1}(k).\tag{9.134}$$

The mean squared value for $\Delta\alpha_i(k)$ can be shown to be

$$E\{[\Delta\alpha_i(k)]^2\} = E\{[\Delta c_i^f(k)]^2\}\lambda^{i-1} + \lambda E\{[\Delta\alpha_{i-1}(k)]^2\} + \sigma_n^2. \quad (9.135)$$

It can be seen that $\varepsilon_i(k)$ is described by step (S.14). The mean squared value for $\Delta\varepsilon_i(k)$ can be calculated as

$$\begin{aligned} E\{[\Delta\varepsilon_i(k)]^2\} &= E\{[\Delta\alpha_i(k)]^2\}E\{e_i^2(k)\} + \sigma_n^2 \\ &\quad + \frac{\sigma_x^2}{1-\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^{i-1} E\{[\Delta e_i(k)]^2\}. \end{aligned} \quad (9.136)$$

9.3.2.5 Mean squared values of $\Delta\alpha_i^f(k)$, $\Delta s_i^b(k)$, $\Delta c_i^f(k)$, $\Delta\beta_i(k)$, and $\Delta e_i(k)$

According to Table 9.5, these quantities have dynamic relations that are very similar to their “dual” (forward or backward) counterparts. Using the same methodology described in the previous sections, the following relations are derived:

$$\begin{aligned} E\{[\Delta\alpha_{i-1}^f(k)]^2\} &= \frac{E[\Delta e_{i-1}^f(k)]^2}{1+\lambda} + \frac{\sigma_n^2}{4\sigma_x^2}\frac{1}{1+\lambda}\left[\frac{1+\lambda}{2\lambda}\right]^{i-1} \\ &\quad + \frac{\sigma_n^2}{1-\lambda^2} \end{aligned} \quad (9.137)$$

$$\begin{aligned} E\{[\Delta s_i^b(k)]^2\} &= \frac{E[\Delta e_{i-1}^f(k)]^2}{\sigma_x^2}(1-\lambda)\left[\frac{1+\lambda}{2\lambda}\right]^{i-1} + \sigma_n^2 \\ &\quad + E\{[\Delta\alpha_{i-1}^f(k)]^2\}\sigma_x^2(1-\lambda)^2\left[\frac{1+\lambda}{2\lambda}\right]^{i-1} \end{aligned} \quad (9.138)$$

$$E\{[\Delta c_i^f(k)]^2\} = \frac{2\lambda(1-\lambda^2)}{\sigma_x^2}\left[\frac{1+\lambda}{2\lambda}\right]^{i-1} E\{[\Delta\alpha_{i-1}^f(k)]^2\} + \sigma_n^2 \quad (9.139)$$

$$\begin{aligned} E\{[\Delta e_i^b(k)]^2\} &= \sigma_x^2\left[\frac{2\lambda}{1+\lambda}\right]^{i-1} E\{[\Delta c_i^b(k)]^2\} + \lambda E\{[\Delta e_{i-1}^b(k)]^2\} \\ &\quad + E\{[\Delta s_i^b(k)]^2\}\sigma_x^2\frac{\lambda}{1+\lambda}\left[\frac{2\lambda}{1+\lambda}\right]^{i-1} \\ &\quad + \lambda(1-\lambda)E\{[\Delta s_i^b(k)]^2\} + \sigma_n^2 \end{aligned} \quad (9.140)$$

The other two remaining values required to compute $E\{[\Delta e(k)]^2\}$ can be derived from step (S.12) and are shown below.

$$
\begin{aligned}
E\{[\Delta\beta_{i-1}(k)]^2\} = {} & \frac{\lambda}{1-\lambda^2}E\{[\Delta c_i^f(k)]^2 E\{\beta_{i-1}^2(k)\} \\
& +\frac{1}{1+\lambda}E\{[\Delta e_{i-1}(k)]^2\}+E\{e_{i-1}^2(k)\}+\frac{\sigma_n^2}{1-\lambda^2} \\
& +\frac{1}{1-\lambda^2}E\{[\Delta s_i^f(k)]^2\} \qquad (9.141)
\end{aligned}
$$

$$
\begin{aligned}
E\{[\Delta e_i(k)]^2\} = {} & \lambda E\{[\Delta e_{i-1}(k)]^2\}+E\{[\Delta c_i^f(k)]^2\}E\{e_{i-1}^2(k)\} \\
& +(1-\lambda)\lambda E\{[\Delta\beta_{i-1}(k)]^2\} \\
& +\lambda E\{[\Delta s_i^f(k)]^2\}E\{\beta_{i-1}^2(k)\}+\sigma_n^2 \qquad (9.142)
\end{aligned}
$$

9.3.2.6 Mean squared value of $\Delta e(k)$

Using step (S.18), the mean squared accumulated quantization error of the *a posteriori* error signal is given by

$$
E\{[\Delta e(k)]^2\}=\lambda^N E\{[\Delta e_{N+1}(k)]^2\}+E\{[\Delta\alpha_{N+1}(k)]^2\}\frac{\sigma_r^2}{\lambda^i}+\sigma_n^2. \quad (9.143)
$$

9.3.3 Simulation results

Intensive simulations were performed to verify the accuracy of derived relations in both infinite and finite-precision. Different values of λ, σ_x^2 and different number of bits were used. In the simulations, 2s complement rounding was used, the input was white Gaussian noise with $\sigma_x^2 = -30$ dB, $\lambda = 0.99$, the measurement error signal had variance $\sigma_r^2 = -70$ dB, and an MA process of order 2 was used. A total of 10,000 points were calculated in both finite-precision and infinite-precision and the last 9000 samples were averaged. The results of simulated and calculated results for $E\{[\Delta e(k)]^2\}$ are displayed in Table 9.6.

Simulations with different values of λ were also performed. On these simulations, 15 bits were used. The input signals were the same as in the previous simulations. The results are shown in Table 9.7.

Table 9.6 Simulation results of $E\{[\Delta e(k)]^2\}$ – different number of bits ($\lambda = 0.99$).

Number of Bits	Simulated (dB)	Calculated (dB)
10	−60.66	−59.94
15	−89.55	−90.05
20	−119.46	−120.15
30	−178.84	−180.36

Table 9.7 Simulation results of $E\{[\Delta e(k)]^2\}$ – different values of λ (15 bits).

λ	Simulated (dB)	Calculated (dB)
0.90	−87.75	−88.96
0.95	−88.67	−89.67
0.98	−89.40	−89.98
0.99	−89.63	−90.05

9.4 Conclusion

This chapter describes propagation models for the error generated by quantization in two important QRD-RLS algorithms, namely the conventional and the FQRD-lattice algorithms. These algorithms are among the sub-class of algorithms that are known to have stable behavior in finite-precision implementations. The approach presented consists of deriving the steady-state mean squared values of all internal variables of the algorithms as well as the mean squared values of their errors originating from quantization.

As a rule, the analytical expressions for the internal variables allow access to estimates of their dynamic range, which in turn should be employed in determining their required wordlengths. In addition, the expressions related to the quantization effects provide tools to estimate the loss in accuracy originating from error propagation in the internal variables. The derived expressions are all verified to be quite accurate through the simulations presented.

References

1. K. J. R. Liu, S.-F. Hsieh, K. Yao, and C.-T. Chiu, Dynamic range, stability, and fault-tolerant capability of finite-precision RLS systolic array based on Givens rotations. IEEE Transactions on Circuits and Systems, vol. 38, no. 6, pp. 625–636 (June 1991)
2. S. Leung and S. Haykin, Stability of recursive QRD-LS algorithms using finite-precision systolic array implementation. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 5, pp. 760–763 (May 1989)
3. S. Haykin, Adaptive Filter Theory. Prentice-Hall, Englewood Cliffs, NJ, USA (1991)
4. J. G. McWhirter, Recursive least-squares minimization using a systolic array. SPIE Real-Time Signal Processing VI, vol. 431, pp. 105–112 (January 1983)
5. W. H. Gentleman and H. T. Kung, Matrix triangularization by systolic arrays. SPIE Real-Time Signal Processing IV, vol. 298, pp. 19–26 (January 1981)
6. P. S. R. Diniz and M. G. Siqueira, Finite precision analysis of the QR-recursive least squares algorithm. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 42, pp. 334–348 (May 1995)
7. A. Papoulis, Probability, Random Variables, and Stochastic Processes. McGraw-Hill Book Company, New York, NY, USA (1965)
8. C. Caraiscos and B. Liu, A roundoff error analysis of the LMS adaptive algorithm. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-32, no, 1, pp. 34–41 (February 1984)

9. C. G. Samson and V. U. Reddy, Fixed point error analysis of the normalized ladder algorithm. IEEE Transactions on Audio, Speech, and Signal Processing, vol. ASSP-31, no. 5, pp. 1177–1191 (October 1983)
10. N. Kalouptsidis and S. Theodoridis, Adaptive System Identification and Signal Processing Algorithms, Prentice-Hall, Upper Saddle River, NJ, USA (1993)
11. M. G. Siqueira and P. S. R. Diniz, Infinite precision analysis of the QR-recursive least squares algorithm. IEEE International Symposium on Circuit and Systems, ISCAS'93, Chicago, USA, pp. 878–881 (May 1993)
12. M. G. Siqueira, P. S. R. Diniz, and A. Alwan, Infinite precision analysis of the fast QR decomposition RLS algorithm. IEEE International Symposium on Circuits and Systems, ISCAS'94, London, UK, vol. 2, pp. 293–296 (May–June 1994)

Chapter 10
On Pipelined Implementations of QRD-RLS Adaptive Filters

Jun Ma and Keshab K. Parhi

Abstract This chapter discusses the pipelined systolic implementations of QR-decomposition-based recursive least-squares (QRD-RLS) adaptive filters. The *annihilation-reording look-ahead* technique is presented as an attractive technique for pipelining of Givens rotation (or CO-ordinate Rotation DIgital Computer (CORDIC)) based adaptive filters. It is an *exact* look-ahead and is based on CORDIC arithmetic, which is known to be numerically stable. The conventional look-ahead is based on multiply–add arithmetic. The annihilation-reording look-ahead technique transforms an orthogonal *sequential* adaptive filtering algorithm into an equivalent orthogonal *concurrent* one by creating additional concurrency in the algorithm. Parallelism in the transformed algorithm is explored, and different implementation styles including *pipelining, block processing*, and *incremental block processing* are presented. Their complexity are also studied and compared. The annihilation-reording look-ahead is employed to develop *fine-grain* pipelined QRD-RLS adaptive filters. Both *implicit* and *explicit* weight extraction algorithms are considered. The proposed pipelined architectures can be operated at arbitrarily high sample rate without degrading the filter convergence behavior. Stability under finite-precision arithmetic are studied and proved for the proposed architectures. The complexity of the pipelined architectures are analyzed and compared.

Jun Ma
Shanghai Jiaotong University, Shanghai – China
e-mail: majun@ic.sjtu.edu.cn

Keshab K. Parhi
University of Minnesota, Minneapolis, MN – USA
e-mail: parhi@umn.edu

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_10, © Springer Science+Business Media, LLC 2009

10.1 QRD-RLS Systolic Architecture

Recursive least squares (RLS) [1] based adaptive filters have wide applications in channel equalization, voiceband modem, high-definition TV (HDTV), digital audio broadcast (DAB) system, beamforming, and speech and image processing. Historically, least mean squares (LMS) based adaptive filters are preferred in practical applications due to their simplicity and ease of implementation. A limitation of LMS algorithm is that it has a very slow convergence rate. The convergence of the LMS algorithm is also very sensitive to the eigenvalue spread of the correlation matrix of the input data. In applications such as HDTV equalizer and DAB system, there are only limited number of data samples available. LMS-based equalizer may not be able to reach convergence. The convergence of the RLS algorithm is an order of magnitude faster than that of the LMS algorithm, but its complexity is an order of magnitude higher. However, with rapid advances in scaled very large scale integration (VLSI) technologies, it is possible to implement RLS adaptive filters on single chips which will make them attractive due to their rapid convergence behavior.

QR decomposition-based RLS (QRD-RLS) algorithm [1], also referred as Givens rotation or COordinate Rotation DIgital Computer (CORDIC)-based RLS algorithm in this chapter, is the most promising RLS algorithm since it possess desirable properties for VLSI implementations such as regularity, good finite word-length behavior, and can be mapped onto CORDIC arithmetic-based processors [2–5]. The QRD-RLS algorithm can be summarized as follows. The notations used in this chapter are slightly different from those used in previous chapters, e.g. Chapters 2–4. For ease of reading, their correspondences are summarized in Table 10.1. At each sample time instance n, evaluate a residual (*a posteriori*) error:

$$e(n) = y(n) - \mathbf{u}^{\mathrm{T}}(n)\,\mathbf{w}(n), \tag{10.1}$$

where $\mathbf{u}(n)$ and $y(n)$ denote the p-element vector of signal samples and the reference signal at time instance n, respectively, and $\mathbf{w}(n)$ is the p-element vector of weights which minimize the quantity

$$\begin{aligned} \xi(n) &= \parallel \Lambda^{1/2}(n)\mathbf{e}(n) \parallel^2 \\ &= \parallel \Lambda^{1/2}(n)\,(\mathbf{y}(n) - A(n)\mathbf{w}(n)) \parallel^2, \end{aligned} \tag{10.2}$$

Table 10.1 Notation correspondences between Chapter 10 and Chapters 2–4.

Notations	Chapter 10	Chapters 2–4
Time index	n	k
Input signal	$u(n)$	$x(k)$
Input vector	$\mathbf{u}(n)$	$\mathbf{x}(k)$
Input matrix	$A(n)$	$\mathbf{X}(k)$
Reference signal	$y(n)$	$d(k)$
Cholesky factor	$R(n)$	$\mathbf{U}(k)$
Cholesky factor degree	$p \times p$	$(N+1)\times(N+1)$

where $\mathbf{y}(n) = [y(1),\ldots,y(n)]^{\mathrm{T}}$ denotes the sequence of all reference signal samples obtained up to time instance n, $A(n) = [\mathbf{u}(1), \mathbf{u}(2), \cdots, \mathbf{u}(n)]^{\mathrm{T}}$ is the input data matrix, and $\Lambda(n) = \mathrm{diag}[\lambda^{n-1}, \cdots, \lambda, 1]$ is the diagonal matrix of the forgetting factors. Here, we assume that all the data are real. The extension to the complex case does not seem to have any particular difficulties. The optimum weight vector $\mathbf{w}_{ls}$ of the QRD-RLS solution can be obtained by solving the following equation:

$$R(n)\,\mathbf{w}_{ls}(n) = \mathbf{p}(n), \tag{10.3}$$

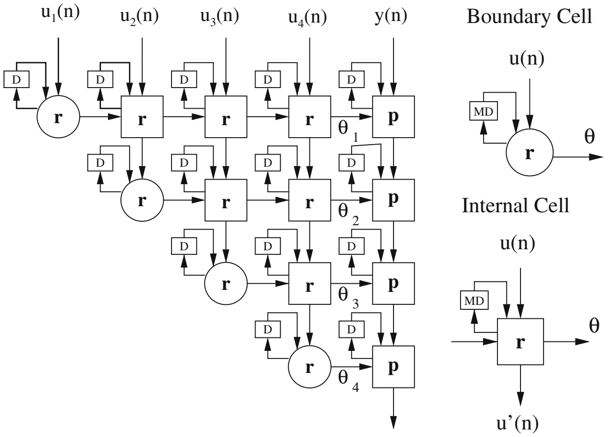
where $R(n)$ and $\mathbf{p}(n)$ are p-by-p matrix and p-by-1 vector, respectively, which are obtained by applying a QR decomposition to the weighted data matrix $\Lambda^{1/2}(n)A(n)$ and the weighted reference vector $\Lambda^{1/2}(n)\mathbf{y}(n)$, respectively. $R(n)$, which is usually referred to, in the literature [17], as the *Cholesky factor*, is chosen, in this chapter, to be an upper triangular matrix.

In practice, the QR decomposition is implemented in a recursive manner. With each incoming data sample set, a new row $\mathbf{u}^{\mathrm{T}}(n)$ is appended to the data matrix $A(n-1)$ to yield $A(n)$. An orthogonal transformation matrix $Q(n)$ is determined as products of p Givens rotation matrices to null the last row of $A(n)$. Thus the triangular matrix $R(n-1)$ gets updated to $R(n)$. The determined matrix $Q(n)$ is then used to update $\mathbf{p}(n-1)$ to $\mathbf{p}(n)$. The QR update procedure can be described by the following equation:

$$\begin{bmatrix} R(n) & \mathbf{p}(n) \\ \mathbf{0}_p^{\mathrm{T}} & \alpha(n) \end{bmatrix} = Q(n) \begin{bmatrix} \lambda^{1/2}R(n-1) & \lambda^{1/2}\mathbf{p}(n-1) \\ \mathbf{u}^{\mathrm{T}}(n) & y(n) \end{bmatrix}. \tag{10.4}$$

A systolic array-based signal flow graph (SFG) representation of the QR update procedure is shown in Figure 10.1. In this figure, cells with symbols r and p inside are the elements of the upper triangular matrix R and the vector $\mathbf{p}$, respectively, as shown in Equation (10.4). The recursive update relationship from time index $n-1$ to n, shown in Equation (10.4), is reflected by the delay element denoted by the small square cell with symbol D inside in Figure 10.1. The input data $u_i(n)$ at the top row of Figure 10.1 are the elements of input vector $\mathbf{u}(n)$ in Equation (10.4), and the reference data $y(n)$ in Figure 10.1 corresponds to the reference data $y(n)$ in Equation (10.4). In Figure 10.1, the circle and square cells denote Givens rotations or CORDIC operations with circle cells operating in *vectoring mode* and square cells operating in *rotating mode*. The functionality of the recursive update from time index $n-1$ to n is shown at the bottom of Figure 10.1. The c and s denote $\cos\theta$ and $\sin\theta$, respectively, which are chosen to annihilate $x_1(n)$. The determined rotation angle is then applied to rotate the second column vector which consists of $\lambda^{M/2}r_2(n-1)$ and $x_2(n)$, where $M=1$ in the case of Equation (10.4). The forgetting factor λ in Equation (10.4), for clarity purpose, is omitted in the circle and square cell in Figure 10.1.

In practice, there are two types of QRD-RLS algorithms. One is *implicit* weight extraction-based RLS algorithms, which are found useful in applications such as adaptive beamforming. In these algorithms, the residual error $e(n)$ is obtained without the explicit computation of the weight vector $\mathbf{w}(n)$. A popular implicit weight



$$\begin{bmatrix} r_1(n) & r_2(n) \\ 0 & x_2'(n) \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \lambda^{M/2} r_1(n-1) & \lambda^{M/2} r_2(n-1) \\ x_1(n) & x_2(n) \end{bmatrix}$$

Fig. 10.1 The systolic array-based signal flow graph representation of the QR update procedure.

extraction algorithm is due to McWhirter etc. [6]. The other is *explicit* weight extraction-based RLS algorithms, which are found useful in applications such as channel equalization. One such algorithm is due to Gentleman and Kung [7], which involves a triangular update part and a linear array for triangular back-solving. The linear array part does not make use of Givens rotations, and thus cannot be efficiently combined with the triangular update part. To overcome this problem, alternative QRD-RLS algorithms with *inverse updating* have been developed to achieve explicit weight extraction and also make use of Givens rotations. A typical structure is the double-triangular type adaptive inverse QR algorithm [8, 9]. This algorithm performs a QR update in an upper triangular matrix and an inverse QR update for weight extraction in a lower triangular matrix.

One of the important ways to design efficient RLS algorithms for high-speed/ low-power applications is through *pipelining* [10, 11] and *parallel processing* [12]. Both implicit and explicit weight extraction-based QRD-RLS algorithms can be easily pipelined at cell level (also referred as *coarse-grain pipelining*). However, the speed or sample rate of the algorithms is limited by the recursive operations in individual cells as shown in Figure 10.1. In many applications, such as image coding and beamforming, very high data rates would be required, and the sequential QRD-RLS algorithms may not be able to operate at such high data rate. In this chapter, we exploit the parallelism that exists in the QRD-RLS algorithm and consider pipelining at finer level such as bit or multi-bit level (also referred as *fine-grain pipelining*). Notice that apart from being used to increase speed, pipelining can also be used to reduce power dissipation in low to moderate speed applications [13].

To exploit the parallelism and increase the speed of the QRD-RLS, look-ahead techniques [14] or block processing techniques can be applied. The look-ahead techniques and the so called STAR rotation have been used in [15] to allow fine-grain pipelining with little hardware overhead. However, this is achieved at the cost of degradation of filtering performance due to the approximations in the algorithms. Block processing was used to speed up the QRD-RLS in [16], however with large hardware overhead. Both algorithms are based on multiply–add arithmetic. If one insists on not using multiply–add arithmetic for their implementation, there is no trivial extension of the look-ahead technique to the QRD-RLS algorithm.

There are other fast QRD-RLS algorithms, which are computationally more efficient than the original algorithm [17, 18]. Square-root free forms of QRD-RLS are presented in [17–22]. A unified approach to square-root QRD-RLS algorithm is presented in [19]. A low-complexity square-root free algorithm is developed in [20]. In [21], a scaled version of the fast Givens rotation [17] is developed that prevents overflow and underflow. A division as well as square-root free algorithm has been proposed in [23]. In [18], a fast QRD-RLS algorithm based on Givens rotations was introduced. However, all these fast algorithms suffer the same pipelining difficulty as the QRD-RLS algorithm, i.e., they cannot be pipelined at fine-grain level.

In this chapter, the *annihilation-reording look-ahead* technique [24] is presented to achieve fine-grain pipelining in QRD-RLS adaptive filters. It is an exact look-ahead and based on CORDIC arithmetic. One of the nice properties of this technique is that it can transform an *orthogonal sequential* recursive DSP algorithm to an equivalent *orthogonal concurrent* one by creating additional concurrency in the algorithm. The resulting transformed algorithm possesses pipelinability, stability (if the original one is stable), and good finite word-length behavior which are attractive for VLSI implementations.

The rest of the chapter are organized as follows. The annihilation-reording look-ahead technique is presented in Section 10.2. The derivation of pipelined CORDIC-based RLS adaptive filters using the proposed look-ahead technique is presented in Section 10.3. Section 10.4 draws conclusions and briefly discusses the adaptive beamforming application. Appendix provides the derivation and proof of some key formulas presented in the chapter.

10.2 The Annihilation-Reording Look-Ahead Technique

In this section, we introduce the annihilate-reording look-ahead technique as an exact look-ahead based on CORDIC arithmetic. Similar to the traditional look-ahead, it transforms a sequential recursive algorithm to an equivalent concurrent one by creating additional parallelism in the algorithm. It is based on CORDIC arithmetic and is suitable for pipelining of Givens rotation-based adaptive filtering algorithms. The annihilation-reording look-ahead technique can be derived from two aspects. One is from the *block processing* point of view, the other is from the *iteration* point of view. The former is practical in real applications, while the latter shows the connection with the traditional look-ahead technique. During our derivation, the forgetting factor λ is omitted for clarity purpose.

This section is organized as follows. The derivations of the annihilation-reording look-ahead through block processing and iteration are presented in Subsection 10.2.1 and Subsection 10.2.2, respectively. The relationship with the conventional multiply–add look-ahead is shown in Subsection 10.2.3. Subsection 10.2.4 explores the parallelism in the proposed look-ahead transformed algorithm. Different implementation styles are then presented in Subsection 10.2.5. Finally, a lemma for stability invariance is stated and proved in Subsection 10.2.6.

10.2.1 Look-ahead through block processing

In this subsection, we derive the annihilation-reording look-ahead transformation for Givens rotation-based algorithms via block-processing formulation.

> The annihilation-reording look-ahead technique can be summarized as the following two-step procedure.
>
> *1.* Formulate block updating form of the recursive operations with block size equal to the pipelining level M.
> *2.* Choose a sequence of Givens rotations to perform the updating in such a way that it first operates on the block data and then updates the recursive variables. The aim is to reduce the computational complexity of a block update inside the feedback loop to the same complexity as a single-step update.

Assume that a three-time speed up is desired for the QR update shown in Figure 10.1. Consider the block update form of the QR update procedure shown in Figure 10.2 with block size equal to the desired pipelining level 3. A sequence

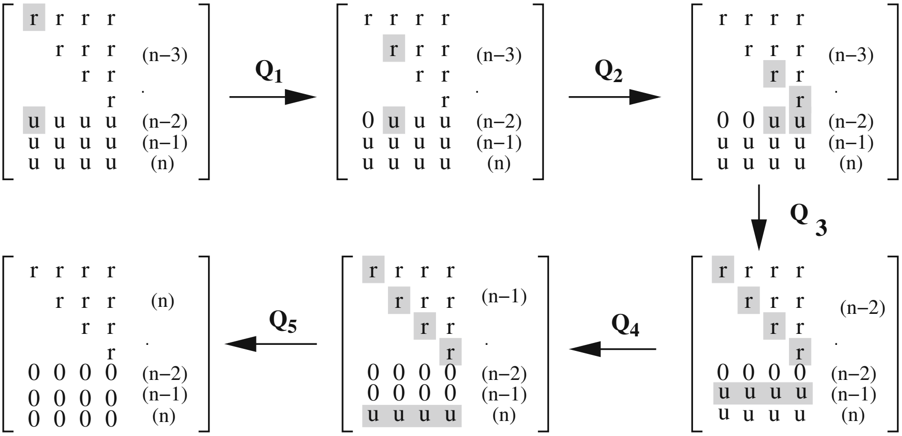


Fig. 10.2 QRD-RLS block update.

of Givens rotations is then chosen to annihilate the block data $\mathbf{u}(n-2), \mathbf{u}(n-1)$, and $\mathbf{u}(n)$, and update $R(n-3)$ to $R(n)$. In this figure, traditional sequential update operation is used. The sample data is annihilated in a row-by-row manner and the diagonal r elements are involved in each update. The SFG of a typical r element update is shown in Figure 10.3. It can be seen that the number of rotations inside the feedback loop increases linearly with the number of delay elements in the loop. Therefore, there is no net improvement in the sample or clock speed.

The annihilation-reording look-ahead technique is illustrated in Figure 10.4. In this figure, the sample data is annihilated in a column-by-column manner and the diagonal r elements are updated only at the last step. The SFG of a typical r element update is shown in Figure 10.5. It can been seen that, without increasing the loop computational complexity, the number of delay elements in the feedback loop is increased from one delay element to three delay elements. These three delay elements can then be redistributed around the loop using the *retiming* technique [25] to achieve fine-grain pipelining by three-level. The two CORDIC units outside the

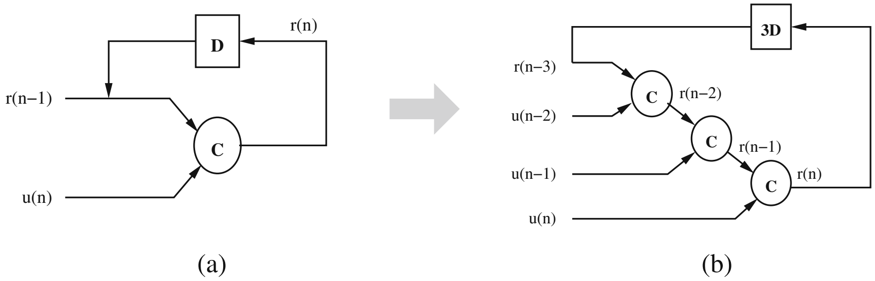


Fig. 10.3 (**a**) The sequential QR update procedure. (**b**) The block update procedure with block size 3.

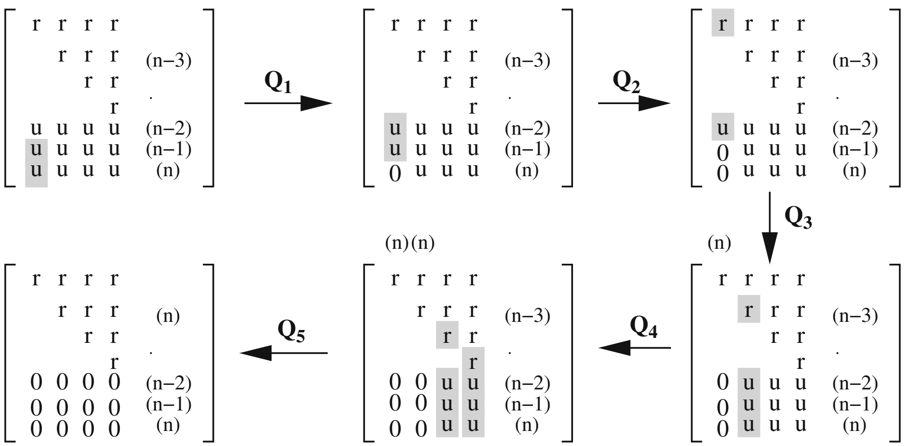


Fig. 10.4 QRD-RLS annihilation-reordering look-ahead.

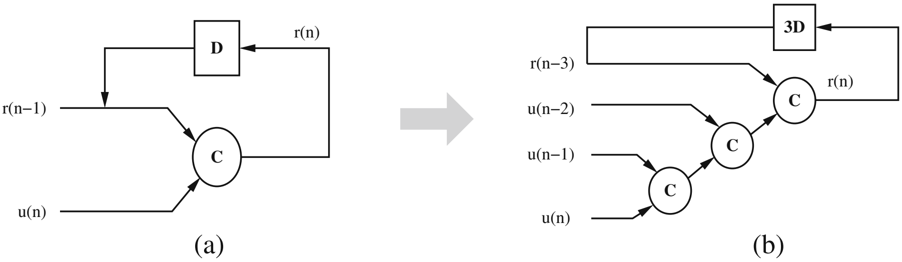


Fig. 10.5 (**a**) A sequential QR update procedure. (**b**) The three-level pipelined architecture using annihilation-reording look-ahead.

feedback loop are the computation overhead due to the look-ahead transformation. Since they are feed-forward, *cutset pipelining* [26] can be applied to speed them up. Furthermore, the overhead CORDIC units outside the loop can be arranged in a *tree* structure to explore the parallelism and reduce overall latency.

10.2.2 Look-ahead through iteration

Alternatively, the annihilation-reording look-ahead can be derived through matrix iterations. From Figure 10.1 and Equation (10.4), the basic QR recursion is given as follows:

$$\begin{bmatrix} r(n) \\ 0 \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} r(n-1) \\ u(n) \end{bmatrix}, \tag{10.5}$$

where $r(n)$ and $u(n)$ correspond to the boundary element and input data to the boundary element in Figure 10.1, respectively. A direct look-ahead by iterating Equation (10.5) two times can be performed by the following embedding procedure. Equation (10.5) can be rewritten as

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & s_1 \\ 0 & 1 & 0 \\ -s_1 & 0 & c_1 \end{bmatrix} \begin{bmatrix} r(n-1) \\ 0 \\ u(n) \end{bmatrix}. \tag{10.6}$$

From Equation (10.5), we also have

$$\begin{bmatrix} r(n-1) \\ 0 \\ u(n) \end{bmatrix} = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r(n-2) \\ u(n-1) \\ u(n) \end{bmatrix}. \tag{10.7}$$

Substituting Equation (10.7) into Equation (10.6) leads to

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & s_1 \\ 0 & 1 & 0 \\ -s_1 & 0 & c_1 \end{bmatrix} \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r(n-2) \\ u(n-1) \\ u(n) \end{bmatrix}. \tag{10.8}$$

This is the one-step iterated version of Equation (10.5). Iterating (10.8) once more leads to the following two-step iterated version of (10.5).

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 & 0 & 0 & s_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_1 & 0 & 0 & c_1 \end{bmatrix} \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ 0 & 1 & 0 & 0 \\ -s_2 & 0 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 & s_3 & 0 & 0 \\ -s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r(n-3) \\ u(n-2) \\ u(n-1) \\ u(n) \end{bmatrix} \tag{10.9}$$

The SFG of (10.9) is shown in Figure 10.3. Notice that this transformation does not help much, since all three CORDIC operations involve updating the r element. Although the feedback loop contains three delays, the computation time in the loop is also increased by a factor of three. Therefore, the overall sample rate remains unaltered.

In order to increase the sample rates, the following transformation is considered. Notice that, in (10.9), instead of vectoring the input vector in the order of $(1,2), (1,3)$, and $(1,4)$, we could apply the Givens matrix in a different order so that the input vector is annihilated in the order of $(3,4), (2,3)$, and $(1,2)$, where notation (i, j) represents a pair of row indexes of input matrix in (10.9). For example, $(3,4)$ denotes that the Givens matrix will operate on input vector $[u(n-1), u(n)]^{\mathrm{T}}$, According to this scheme, the input samples are pre-processed and the r elements are updated only at the last step. This leads to the following three-level annihilation-reording look-ahead transformation for CORDIC-based RLS adaptive filters.

$$\begin{bmatrix} r(n) \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1^{'} & s_1^{'} & 0 & 0 \\ -s_1^{'} & c_1^{'} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2^{'} & s_2^{'} & 0 \\ 0 & -s_2^{'} & c_2^{'} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_1^{'} & s_1^{'} \\ 0 & 0 & -s_1^{'} & c_1^{'} \end{bmatrix} \begin{bmatrix} r(n-3) \\ u(n-2) \\ u(n-1) \\ u(n) \end{bmatrix}$$

The SFG of the above transformation is shown in Figure 10.5, which is the same as the one derived from the block processing point of view. Therefore, without increasing the loop computation time, we increase the number of delays in the feedback loop from one delay element to three delay elements. These three delay elements can then be redistributed around the loop to achieve pipelining by three-level.

The above derivation is similar to the traditional multiply–add look-ahead [11] procedure in the sense that both perform look-ahead through iteration. However, it can be seen that the block processing derivation in Section 10.2.1 is more simple and practical in real applications.

10.2.3 Relationship with multiply–add look-ahead

It is worth mentioning here, for the first order case, that there exists strong similarity of the transformed flow graphs between the annihilation-reording look-ahead and the multiply–add look-ahead. Consider the first order IIR digital filter described by the following equation:

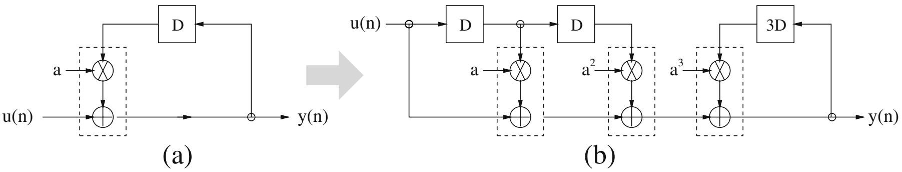
$$y(n) = ay(n-1) + u(n). \tag{10.10}$$

The SFG of (10.10) is shown in Figure 10.6(a). After applying the multiply–add look-ahead transformation with pipelining level 3, the resulting equation is given as

$$y(n) = a^3 y(n-3) + a^2 u(n-2) + au(n-1) + u(n). \tag{10.11}$$

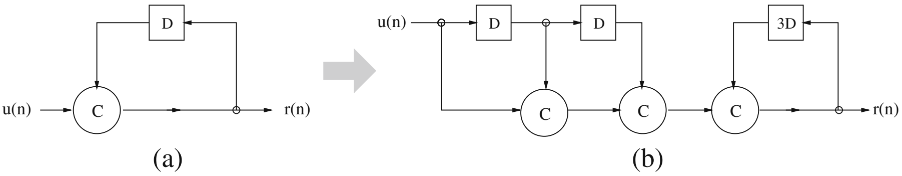
The SFG of (10.11) is shown in Figure 10.6(b). The filter sample rate can be increased by a factor of 3 after redistributing the three delay elements in the feedback loop using the *retiming* technique [25].

On the other hand, a redraw of Figure 10.5 can be carried out as in Figure 10.7. Comparing Figures 10.7 to Figure 10.6, it is seen that the two graphs are essentially the same except that the *multiply–add* units are replaced by the *CORDIC* units. Thus, both the annihilation-reording look-ahead and the multiply–add look-ahead explore the intra-iteration constraints in the recursive algorithms and create additional concurrency. The differences between the two techniques lie in that the multiply–add look-ahead is suitable for multiply–add arithmetic-based recursive digital filters where the filter coefficients are fixed. The annihilation-reording look-ahead is suitable for CORDIC arithmetic (or Givens rotation) based adaptive digital filters, where the filter coefficients are adaptive to the input data.



(a) (b)

Fig. 10.6 (a) A first-order IIR digital filter. (b) The three-level pipelined architecture using multiply–add look-ahead.



(a) (b)

Fig. 10.7 (a) A sequential QR update procedure. (b) The three-level pipelined architecture using annihilation-reording look-ahead.

10.2.4 Parallelism in annihilation-reording look-ahead

In this subsection, we show explicitly how the annihilation-reording look-ahead technique explore the parallelism in the recursive algorithm and create the additional concurrency.

Consider the sequential QR update procedure shown in Figure 10.5(a). Its dependence graph (DG) representation is shown in Figure 10.8. In this figure, the little circle denotes CORDIC operations. The arrows denote dependency between signals. The k direction is the time increasing direction. The arrows along the k direction represent the dependency between iterations. For example, $r(n+1)$ is dependent on $r(n)$, $r(n)$ is dependent on $r(n-1)$, and so on. As we mentioned in Section 10.1, it is this kind of dependency that limits the speed of the QR update and thus limits the sample rate. The annihilation-reording look-ahead actually breaks this dependency and transforms the original DG into an equivalent DG which consists of M *independent* sub-DGs, where M is the desired pipelining level. Since these M sub-DGs are independent, they can be executed in parallel. Therefore, the M independent sub-DGs are the additional concurrency created by look-ahead transformation. For $M = 3$, the three sub-DGs: *DG-I, DG-II*, and *DG-III* are shown in Figure 10.9. Figure 10.9 is the DG representation of Figure 10.5(b). From Figure 10.9, it is seen that the computation of $r(n)$ is *not* dependent on the $r(n-1)$ anymore, instead dependent on the r element three iterations back in time which is $r(n-3)$. Similarly, $r(n+1)$ is dependent on $r(n-2)$, and $r(n+2)$ depends on $r(n-1)$. Therefore, after look-ahead transformation, the dependency between consecutive iterations are broken down into three independent operation sequences with each sequence having a dependency between every three iterations. For each sub-DGs, the dependency which is perpendicular to the k direction does not cause problem, since *index transformation* [27] (which is equivalent to the *cut-set* pipelining for SFG) can be performed to reveal these dependency. Therefore, the three independent sequences, which consist of $3^2 = 9$ independent CORDIC operations in total, are created for one iteration. In general, for pipelining level M, M^2 independent CORDIC operations are created in the algorithm for one iteration. As M increases, infinite parallelism can be created in the algorithm, thus can achieve arbitrarily high sample rate.

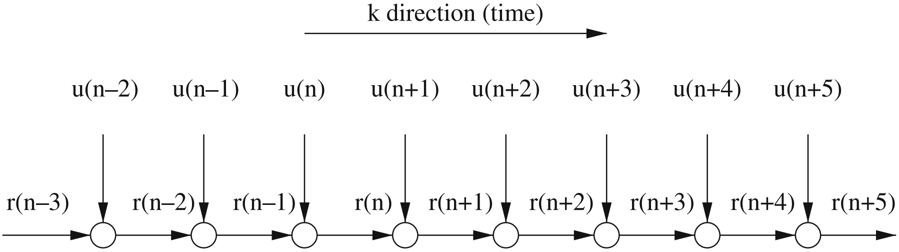


Fig. 10.8 The dependence graph of the sequential QR update procedure.

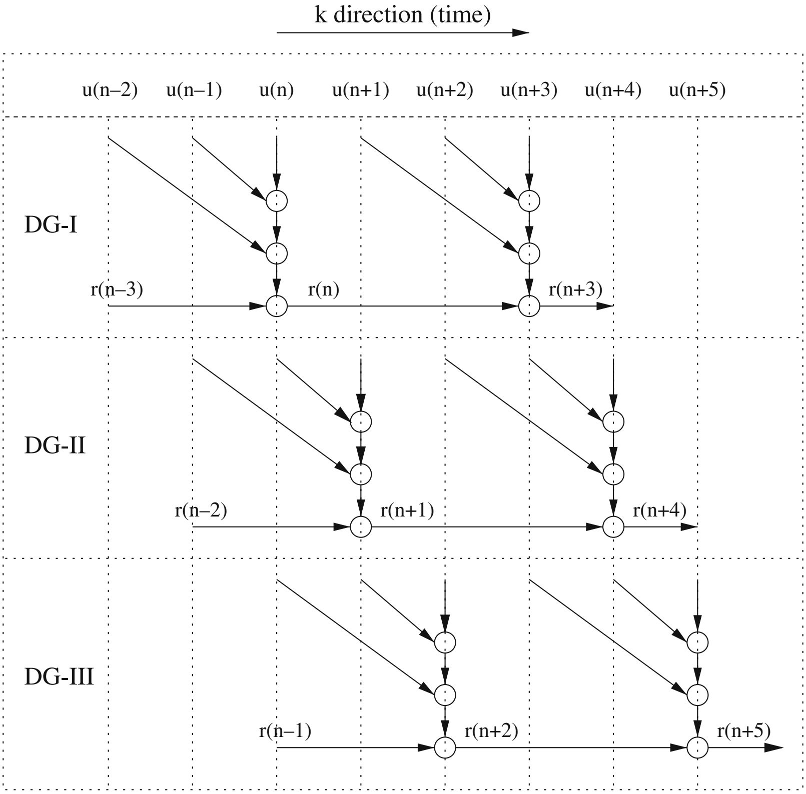


Fig. 10.9 The dependence graph of the pipelined QR update with pipelining level 3.

10.2.5 Pipelined and block processing implementations

In this subsection, we present three concurrent realizations of CORDIC-based RLS adaptive filters. They are *pipelining*, *block processing*, and *incremental block processing*.

10.2.5.1 Pipelined realization

Consider the three sub-DGs in Figure 10.9. If they are mapped along the *k* direction, we obtain the SFG representation. Since the three sub-DGs are independent, they can be mapped onto the same CORDIC operation resources and operated in a pipeline interleaving fashion [11]. This leads to the pipelined realization of CORDIC-based adaptive filters shown in Figure 10.10. In this figure, the input data samples are processed in the block manner through a tapped delay line as shown in Figure 10.11(a). Since consecutive block samples are shift-overlapped, thus all

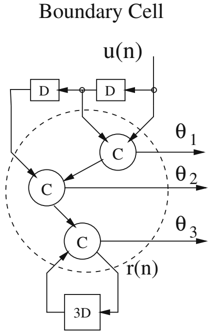


Fig. 10.10 Pipelined realization with pipelining level 3.

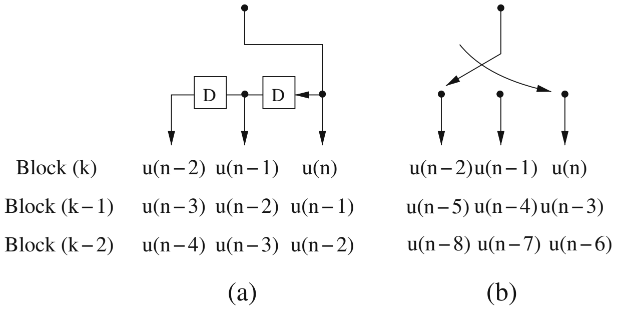


Fig. 10.11 Serial-to-parallel conversion for (**a**) Pipelining and (**b**) Block Processing.

filtering output can be obtained consecutively. The implementation complexity in terms of CORDIC units for pipelined realization is linear with respect to the pipelining level M which is $M = 3$ CORDIC units in Figure 10.10.

10.2.5.2 Block processing realization

In block processing, the three sub-DGs are mapped independently along the k direction to obtain the block processing realization shown in Figure 10.12. In block realizations, input samples are processed in the form of non-overlapping blocks to generate non-overlapping output samples. The block of multiple inputs is derived from the single serial input by using a serial-to-parallel converter at the input as shown in Figure 10.11(b), and the serial output is derived from the block of outputs by a parallel-to-serial converter at the output. The implementation complexity in terms of CORDIC units for block processing realization is quadratic with respect to the pipelining level M which is $M^2 = 3^2 = 9$ CORDIC units in Figure 10.12. To reduce complexity, incremental block processing technique can be used.

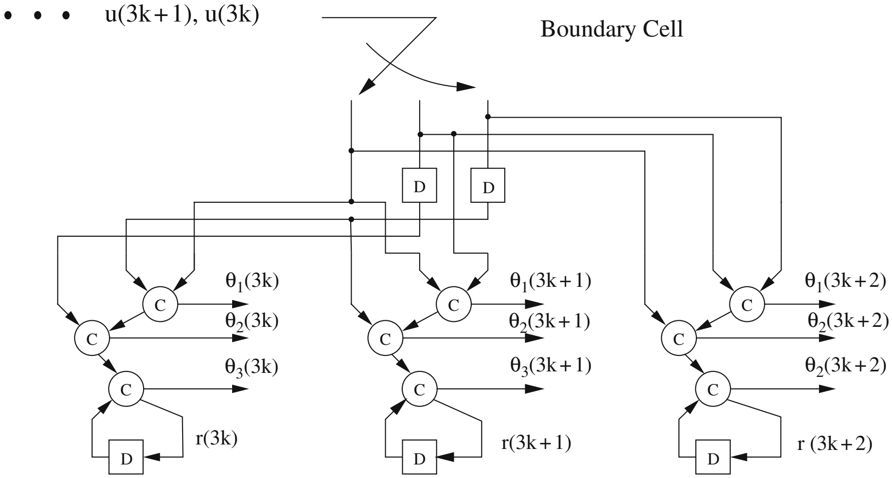


Fig. 10.12 Block processing realization with block size 3.

10.2.5.3 Incremental block processing realization

Consider the annihilation-reording look-ahead transformed DG shown in Figure 10.9. Instead of using $u(n+1), u(n), u(n-1)$, and $r(n-2)$ to obtain $r(n+1)$, $r(n+1)$ can be computed incrementally using $u(n+1)$ and $r(n)$ once $r(n)$ is available. Similarly, $r(n+2)$ can be computed incrementally using $u(n+2)$ and $r(n+1)$ once $r(n+1)$ is available. The DG of incremental block QR update is shown in Figure 10.13. Mapping the DG along the *k* direction gives us the SFG representation of the incremental block processing realization shown in Figure 10.14. The implementation complexity in terms of CORDIC units for incremental block processing is linear with respect to the pipelining level *M* which is $2M-1 = 2\times 3-1 = 5$ CORDIC units in Figure 10.14. Notice that the incremental computation parts do not contain feedback loops, thus cutset pipelining can be employed to speed them up.

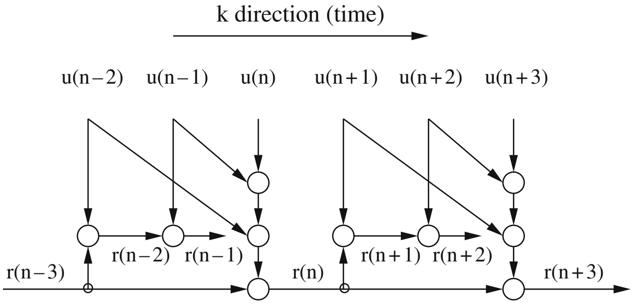


Fig. 10.13 The dependence graph of the incremental block QR update with block size 3.

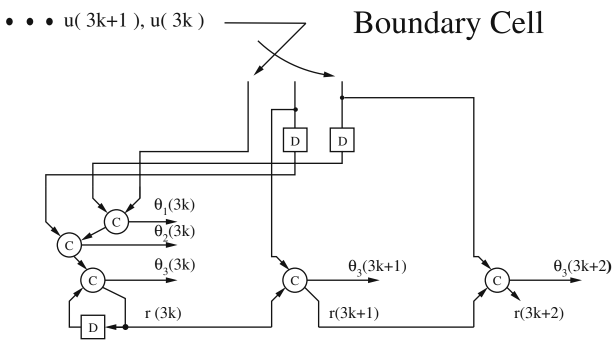


Fig. 10.14 Incremental block processing realization with block size 3.

Therefore, in terms of number of CORDIC units used in the implementation, pipelined realization is better than incremental block processing and block processing, and incremental block processing is better than block processing. In practice, the chosen of implementation styles depends on the target applications and available hardware resources.

10.2.6 Invariance of bounded input and bounded output

In this subsection, we show a property of the annihilation-reording look-ahead transformation. It will be useful in the proof of the stability of the pipelined QRD-RLS algorithm in Section 10.3.2.

Lemma 1. *Consider the compound CORDIC cell denoted by the dashed circle in Figure 10.10. Under finite-precision arithmetic, if each individual CORDIC cell is bounded input and bounded output (BIBO), then the compound CORDIC cell is also BIBO.*

Proof. Assume the pipelining level is M, from Figure 10.10, the look-ahead transformed compound CORDIC cell consists of cascade connections of M CORDIC units. Since each of them is BIBO under finite-precision arithmetic, therefore the compound cell is also BIBO.

10.3 Pipelined CORDIC-Based RLS Adaptive Filters

In this section, we apply the annihilation-reording look-ahead techniques to the CORDIC-based RLS adaptive filters and derive fine-grain pipelined topologies. We consider both algorithms with implicit weight extraction (conventional QRD-RLS) and explicit weight extraction (inverse QRD-RLS).

This section is organized as follows. The pipelined QRD-RLS with implicit weight extraction is presented in Section 10.3.1. Its stability under finite-precision arithmetic is studied and proved in Section 10.3.2. Finally, the pipelined adaptive inverse QR algorithm for explicit weight extraction is presented in Section 10.3.3.

10.3.1 Pipelined QRD-RLS with implicit weight extraction

Consider the QRD-RLS formulation given in Equations (10.1), (10.2), (10.3), and (10.4). After the triangular matrix $R(n)$ and the corresponding vector $\mathbf{p}(n)$ are generated, the optimum weight vector $\mathbf{w}(n)$ can be obtained by solving Equation (10.3). The residual error $e(n)$ is then computed as

$$e(n) = y(n) - \mathbf{u}^{\mathrm{T}}(n) R^{-1}(n)\, \mathbf{p}(n). \tag{10.12}$$

However, for some applications such as adaptive beamforming, this proves to be unnecessary. Since in these cases, the residual error $e(n)$ is usually the only variable interested, and it is not necessary to compute the weight vector $\mathbf{w}(n)$ explicitly. In [6], it is shown that the estimation error $e(n)$ may be obtained directly as the *product* of two variables, the angle-normalized residual $\alpha(n)$ and the likelihood factor $\gamma(n)$. $\alpha(n)$ and $\gamma(n)$ are obtained by applying the same orthogonal transformation matrix $Q(n)$ to the vector $[\mathbf{p}(n-1), y(n)]^{\mathrm{T}}$ and the pinning vector $\boldsymbol{\pi} = [0, \cdots, 0, 1]^{\mathrm{T}}$ [6]. Therefore, the adaptive RLS algorithm can be summarized as

$$\begin{bmatrix} R(n) & \mathbf{p}(n) & \mathbf{s}(n) \\ \mathbf{0}_p^{\mathrm{T}} & \alpha(n) & \gamma(n) \end{bmatrix} = Q(n) \begin{bmatrix} \lambda^{1/2} R(n-1) & \lambda^{1/2} \mathbf{p}(n-1) & \mathbf{0}_p \\ \mathbf{u}^{\mathrm{T}}(n) & y(n) & 1 \end{bmatrix}, \tag{10.13}$$

where $\mathbf{0}_p$ is the p-by-1 null vector. The SFG representation of the algorithm is shown in Figure 10.15, where problem size p is chosen to be 4. The circle and square cells in Figure 10.15 denote CORDIC operations which follow the same notations in Figure 10.1. The circle cell with letter G inside denotes a Gaussian rotation (or a linear CORDIC operation). Its functionality is shown in the figure. Notice that the converting factor cells which generate the likelihood factor γ does not contain recursive operations.

In Figure 10.15, the recursive operation in the cell limits the throughput of the input samples. To increase the sample rates, the annihilation-reording look-ahead technique is applied.

The recursive updating formula for the QRD-RLS with implicit weight extraction is given in Equation (10.13). Its block updating form with block size M is given as follows:

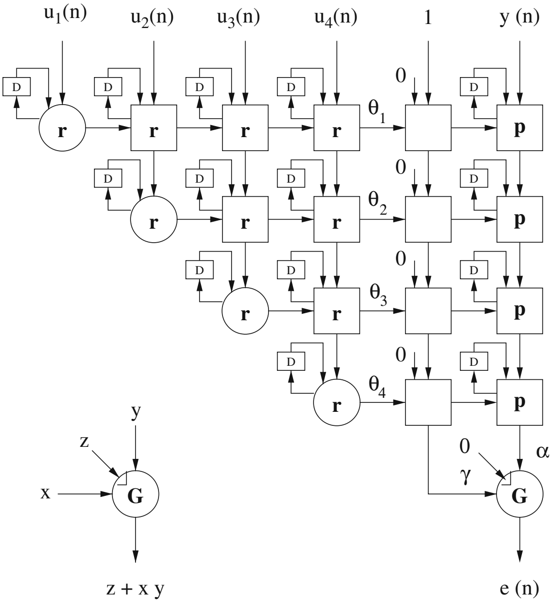


Fig. 10.15 Signal flow graph representation for RLS minimizations.

$$\begin{bmatrix} R(n) & \mathbf{p}(n) & \mathbf{s}(n) \\ O_{M\times p} & \boldsymbol{\alpha}(n) & \boldsymbol{\gamma}(n) \end{bmatrix} = Q(n) \begin{bmatrix} \lambda^{M/2}R(n-M) & \lambda^{M/2}\mathbf{p}(n-M) & \mathbf{0}_p \\ U_M(n) & \mathbf{y}_M(n) & \boldsymbol{\delta}_M \end{bmatrix}, \tag{10.14}$$

where $U_M(n)$ is an M-by-p matrix defined as

$$U_M(n) = [\mathbf{u}(n-M+1),\cdots,\mathbf{u}(n-1),\mathbf{u}(n)]^{\mathrm{T}},$$

and $\mathbf{y}_M(n)$ is an M-by-1 vector defined as

$$\mathbf{y}_M(n) = [y(n-M+1),\cdots,y(n-1),y(n)]^{\mathrm{T}}.$$

In (10.14), $O_{M\times p}$ and $\mathbf{0}_p$ denote M-by-p null matrix and p-by-1 null vector, respectively, $\boldsymbol{\alpha}(n)$ and $\boldsymbol{\gamma}(n)$ are M-by-1 vectors, and $\boldsymbol{\delta}_M$ is a M-by-1 constant vector defined as $\boldsymbol{\delta}_M = [0,\ldots,0,1]^{\mathrm{T}}$. The estimation error $e(n)$ can be calculated as the *inner product* of the angle-normalized residual vector $\boldsymbol{\alpha}(n)$ and the likelihood vector $\boldsymbol{\gamma}(n)$, i.e.,

$$e(n) = \boldsymbol{\alpha}^{\mathrm{T}}(n)\,\boldsymbol{\gamma}(n). \tag{10.15}$$

The proof is given in Appendix.

We now determine a sequence of Givens rotations, whose product form the orthogonal transformation matrix $Q(n)$ in (10.14), to annihilate the block input data matrix $U_M(n)$. The order of the Givens rotations is chosen such that the input data is pre-processed and block-data update is finished in the same complexity as a single-data update. This procedure was illustrated in detail in Figure 10.4. A three-level fine-grain pipelined QR update topology was shown in Figure 10.5. After applying the annihilation-reording look-ahead, the concurrent QRD-RLS algorithm can be realized using different implementation styles such as pipelining, block processing, and incremental block processing as discussed in Section 10.2.5. In the rest of the chapter, we only show the topologies for the pipelined realization. The other realizations can be derived similarly. A fine-grain pipelining implementation with pipelining level 3 of CORDIC-based QRD-RLS adaptive filter with implicit weight extraction is shown in Figure 10.16. In this figure, all cell notations follow the notations in Figure 10.15 except that they are compound versions. The internal structure of each compound cell is shown at the bottom part of Figure 10.16. Compared to Figure 10.15, the three-level pipelined architecture tripled the number of CORDIC units and communication bandwidth which is linear with respect to the pipelining level. Thus, in general, the total complexity is $\mathscr{O}[\frac{1}{2}Mp^2]$ CORDIC units per sample time, where p is the input sample size, and M is the pipelining level.

10.3.2 Stability analysis

It is generally recognized that the QR decomposition-based algorithms have good numerical properties, which means that they can perform with an acceptable manner in a short word-length environment. This is due to the fact that the algorithms consist of only orthogonal transformation which leads to inherent stability under finite-precision implementation. From Sections 10.2.1 and 10.2.2, we see that the annihilation-reording look-ahead transformation only involves orthogonal transformation and does not alternate the orthogonality of the algorithm. This implies that the pipelined algorithms also maintain the good numerical properties. Let's define the *stability* of the QRD-RLS algorithm in the sense of BIBO i.e., under finite-precision arithmetic, if the input signals are bounded, then the output residual error $e(n)$ is also bounded. We have the following result:

Theorem 1. *Under finite-precision arithmetic, given a pipelining level M, the M-level fine-grain pipelined CORDIC-based RLS adaptive filter algorithm with implicit weight extraction is stable.*

Proof. In [28], it is shown that for the sequential QRD-RLS algorithm shown in Figure 10.15, a CORDIC cell, operating with finite-precision arithmetic, constitutes a BIBO subsystem of the array. From Figure 10.16, the pipelined algorithm has the same architecture as the sequential one except that all CORDIC cells are compound versions. Therefore, by Lemma 1, a *compound* CORDIC cell, operating with

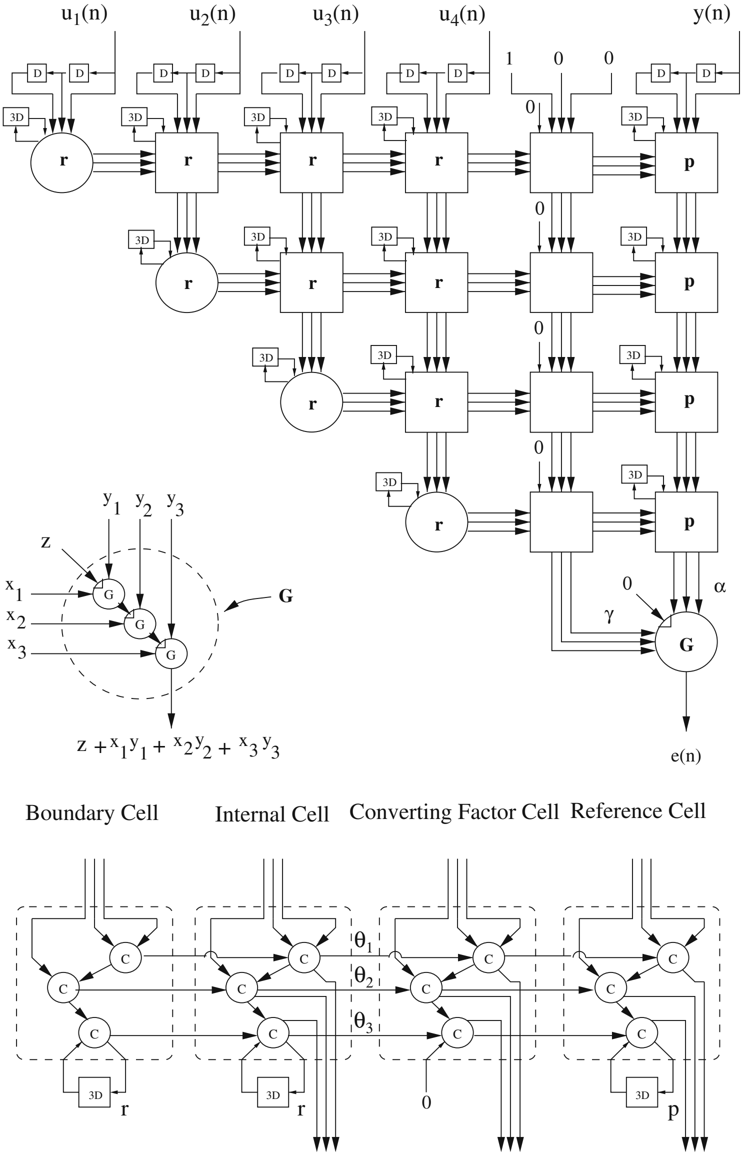


Fig. 10.16 A three-level fine-grain pipelined CORDIC-based implicit weight extraction QRD-RLS adaptive filter architecture.

finite-precision arithmetic, constitutes a BIBO subsystem of the array. Thus, if the desired response $y(n)$ and input samples $\mathbf{u}(n)$ in Figure 10.16 are bounded, the quantized value of the input of the final linear CORDIC cell is also bounded, which leads to the bounded residual error $e(n)$. This completes the proof of Theorem 1.

The stability of other CORDIC-based RLS adaptive filtering algorithms presented in this chapter can also be proved using the similar approach as in Theorem 1 and will not be repeated any further.

10.3.3 Pipelined QRD-RLS with explicit weight extraction

In applications such as channel equalization, RLS-based equalization algorithms such as, e.g., *decision-directed schemes* [29] and *orthogonalized constant modulus algorithms* [30], require the explicit availability of the filter weight vector $\mathbf{w}(n)$. The standard (Gentleman-Kung type) QRD-RLS update scheme involves two computational steps which cannot be efficiently combined on a pipelined array. To circumvent the difficulty, inverse updating-based algorithms are developed [30–32]. Here, we focus on a double-triangular type adaptive inverse QR algorithm [8].

Consider the least squares formulation given in (10.1), (10.2), (10.3), and (10.4). Define the $(p+1)$-by-$(p+1)$ upper triangular compound matrix $\tilde{R}(n)$ as

$$\tilde{R}(n) = \begin{bmatrix} \lambda^{1/2}R(n) & \lambda^{1/2}\mathbf{p}(n) \\ \mathbf{0}_p^{\mathrm{T}} & \gamma(n) \end{bmatrix},$$

where $\gamma(n)$ is a scalar factor, and $R(n), \mathbf{p}(n), \mathbf{0}_p^{\mathrm{T}}$ are defined as in Section 10.1. Using Equation (10.3), $\tilde{R}^{-1}$ is then given as

$$\begin{aligned} \tilde{R}^{-1}(n) &= \begin{bmatrix} \lambda^{-1/2}R^{-1}(n) & -R^{-1}(n)\mathbf{p}(n)/\gamma(n) \\ \mathbf{0}_p^{\mathrm{T}} & 1/\gamma(n) \end{bmatrix} \\ &= \begin{bmatrix} \lambda^{-1/2}R^{-1}(n) & -\mathbf{w}(n)/\gamma(n) \\ \mathbf{0}_p^{\mathrm{T}} & 1/\gamma(n) \end{bmatrix}. \end{aligned}$$

Notice that $\tilde{R}^{-1}$ remains upper triangular and the optimal weight vector $\mathbf{w}(n)$ is explicitly shown on the rightmost column of $\tilde{R}^{-1}$ except for a scaling factor $-1/\gamma$. Now, consider the QR update of the upper triangular compound matrix $\tilde{R}$. From (10.4), we have

$$\begin{bmatrix} \tilde{R}(n) \\ \mathbf{0}_{p+1}^{\mathrm{T}} \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}(n-1) \\ \tilde{\mathbf{u}}^{\mathrm{T}}(n) \end{bmatrix}, \tag{10.16}$$

where $\tilde{\mathbf{u}}^{\mathrm{T}}(n) = [\mathbf{u}^{\mathrm{T}}(n), y(n)]$, and $\tilde{Q}(n)$ is determined as products of $(p+1)$ Givens rotation matrices to null the input sample vector $\tilde{\mathbf{u}}^{\mathrm{T}}(n)$ and update matrix $\tilde{R}$. Extending the $(p+2)$-by-$(p+1)$ matrix on the right-hand-side of (10.16) to the $(p+2)$-by-$(p+2)$ square matrix by adding an extra column vector $[\mathbf{0}_{p+1}^{\mathrm{T}}, 1]^{\mathrm{T}}$ to its right leads to

$$\begin{bmatrix} \tilde{R}(n) & \mathbf{v}(n) \\ \mathbf{0}_{p+1}^{\mathrm{T}} & d(n) \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}(n-1) & \mathbf{0}_{p+1} \\ \tilde{\mathbf{u}}^{\mathrm{T}}(n) & 1 \end{bmatrix}, \tag{10.17}$$

where vector $\mathbf{v}(n)$ and scalar $d(n)$ correspond to the QR update of vector $\mathbf{0}_{p+1}$ and scalar 1. Inverting the matrix on both sides of Equation (10.17) (the matrix is non-singular since $\tilde{R}$ is non-singular) and noticing that $Q^{-1} = Q^{\mathrm{T}}$ lead to

$$\begin{bmatrix} \tilde{R}^{-1}(n) & \mathbf{v}'(n) \\ \mathbf{0}_{p+1}^{\mathrm{T}} & d'(n) \end{bmatrix} = \begin{bmatrix} \tilde{R}^{-1}(n-1) & \mathbf{0}_{p+1} \\ -\tilde{\mathbf{u}}^{\mathrm{T}}(n)\tilde{R}^{-1}(n-1) & 1 \end{bmatrix} \tilde{Q}^{\mathrm{T}}(n). \tag{10.18}$$

Taking the transposition on both sides of (10.18), we obtain

$$\begin{bmatrix} \tilde{R}^{-T}(n) & \mathbf{0}_{p+1} \\ \mathbf{v}'^{\mathrm{T}}(n) & d'(n) \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}^{-T}(n-1) & -\tilde{R}^{-T}(n-1)\tilde{\mathbf{u}}(n) \\ \mathbf{0}_{p+1}^{\mathrm{T}} & 1 \end{bmatrix}.$$

Thus, we have the following inverse updating formula

$$\begin{bmatrix} \tilde{R}^{-T}(n) \\ \mathbf{v}'^{\mathrm{T}}(n) \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}^{-T}(n-1) \\ \mathbf{0}_{p+1}^{\mathrm{T}} \end{bmatrix}. \tag{10.19}$$

Notice that the orthogonal transformation matrix $\tilde{Q}(n)$, which updates the upper triangular matrix $\tilde{R}$ in (10.16), also updates the lower triangular matrix $\tilde{R}^{-T}$ in (10.19). Thus, the double-triangular adaptive inverse QR algorithm can be summarized as follows:

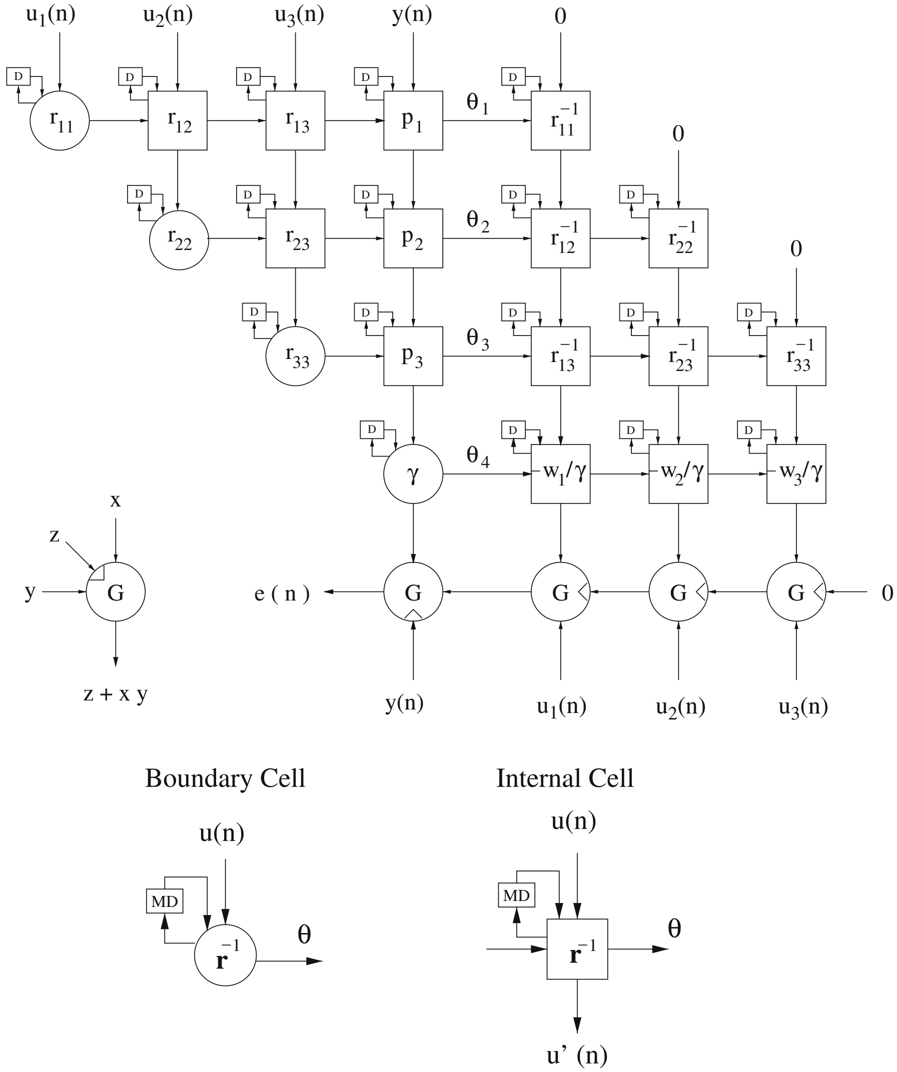
$$\begin{bmatrix} \tilde{R}(n) & \tilde{R}^{-T}(n) \\ \mathbf{0}_{p+1}^{\mathrm{T}} & \mathbf{v}'^{\mathrm{T}}(n) \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}(n-1) & \tilde{R}^{-T}(n-1) \\ \tilde{\mathbf{u}}^{\mathrm{T}}(n) & \mathbf{0}_{p+1}^{\mathrm{T}} \end{bmatrix}. \tag{10.20}$$

The important point lies in noticing that the scaled weight vector $-\mathbf{w}/\gamma$ *explicitly* sits on the bottom row of lower triangular matrix $\tilde{R}^{-T}$ or the rightmost column of upper triangular matrix $\tilde{R}^{-1}$ as shown before. Therefore, we could achieve parallel weight extraction by taking out the last row elements of $\tilde{R}^{-T}(n)$ and multiply them by the scaling factor $\gamma(n)$ sitting on the lower right corner of upper triangular matrix $\tilde{R}(n)$.

An efficient SFG representation of the CORDIC-based double-triangular adaptive inverse QR algorithm is shown in Figure 10.17. In this figure, the notation follows the ones in Figure 10.1. The operation for updating r^{-1} elements is shown at the bottom part of Figure 10.17. The residual error $e(n)$ is computed according to (10.1) as shown in the figure.

The element on the lower right corner of lower triangular matrix $\tilde{R}^{-T}$ contains value $1/\gamma(n)$ and is not shown in the figure. This algorithm has complexity $\mathscr{O}[p^2]$ Givens rotations per sample period, where p is the size of the array.

From Figure 10.17, we see that the double-triangular adaptive inverse QR algorithm can be easily pipelined at cell level after applying *cut-set* pipelining. The speed or sample rates of the algorithm's implementation is, however, limited by recursive operations in each individual cell as described algebraically by (10.20). We now apply the annihilation-reording look-ahead technique to derive concurrent



$$\begin{bmatrix} r_1^{-1}(n) & r_2^{-1}(n) \\ 0 & u_2'(n) \end{bmatrix} = \begin{bmatrix} c & s \\ -\,s & c \end{bmatrix} \begin{bmatrix} \lambda^{-M/2}\, r_1^{-1}(n\text{-}1) & \lambda^{-M/2}\, r_2^{-1}(n-1) \\ u_1(n) & u_2(n) \end{bmatrix}$$

Fig. 10.17 Signal flow graph representation of double-triangular type adaptive inverse QR algorithm.

adaptive inverse QR algorithm for high-speed CORDIC-based parallel RLS weight extraction.

The block updating form with block size M of the sequential updating Equation (10.20) is given as follows:

$$\begin{bmatrix} \tilde{R}(n) & \tilde{R}^{-T}(n) \\ 0_{M\times(p+1)} & V(n) \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}(n-M) & \tilde{R}^{-T}(n-M) \\ \tilde{U}_M^{\mathrm{T}}(n) & 0_{M\times(p+1)} \end{bmatrix}, \tag{10.21}$$

where $\tilde{U}_M^{\mathrm{T}}(n)$ is a M-by-$(p+1)$ matrix defined as

$$\tilde{U}_M^{\mathrm{T}}(n) = \left[\tilde{\mathbf{u}}(n-M+1) \cdots \tilde{\mathbf{u}}(n-1)\ \tilde{\mathbf{u}}(n)\right]^{\mathrm{T}},$$

$O_{M\times(p+1)}$ denotes a M-by-$(p+1)$ null matrix, and $V(n)$ is a M-by-$(p+1)$ matrix.

The derivation of (10.21) essentially follows the algebraic manipulation in (10.16), (10.17), (10.18), (10.19), and (10.20) provided that we start from the following block update equation

$$\begin{bmatrix} \tilde{R}(n) \\ 0_{M\times(p+1)} \end{bmatrix} = \tilde{Q}(n) \begin{bmatrix} \tilde{R}(n-M) \\ \tilde{U}_M(n) \end{bmatrix}. \tag{10.22}$$

Notice that the $\tilde{Q}(n)$ matrix in (10.22) is different from the $\tilde{Q}(n)$ in (10.16), though we use the same notation here.

Apply the annihilation-reording procedure described in Figure 10.4; we obtain the concurrent architecture shown in Figure 10.5. A complete three-level fine-grain pipelined topology for CORDIC-based QRD-RLS with explicit parallel weight extraction is shown in Figure 10.18. In this figure, all cell notations follow the notations in Figure 10.17 except that some of them are compound versions. The internal structure of each compound cell is shown at the bottom part of Figure 10.18. Compared to Figure 10.17, the number of CORDIC units and communication bandwidth are tripled which is linear with respect to the pipelining level. In general, the total complexity for pipelined realization of CORDIC-based QRD-RLS with explicit weight extraction is $\mathscr{O}[Mp^2]$, where M is the pipelining level and p is the size of input samples.

10.4 Conclusion

In this chapter, the annihilation-reording look-ahead technique is presented to achieve fine-grain pipelining for CORDIC-based RLS adaptive filtering algorithms. It is an exact look-ahead and based on CORDIC arithmetic. The look-ahead transformation can be derived from either the block processing or the iteration point of view, while the former is simpler and more practical and the latter shows the

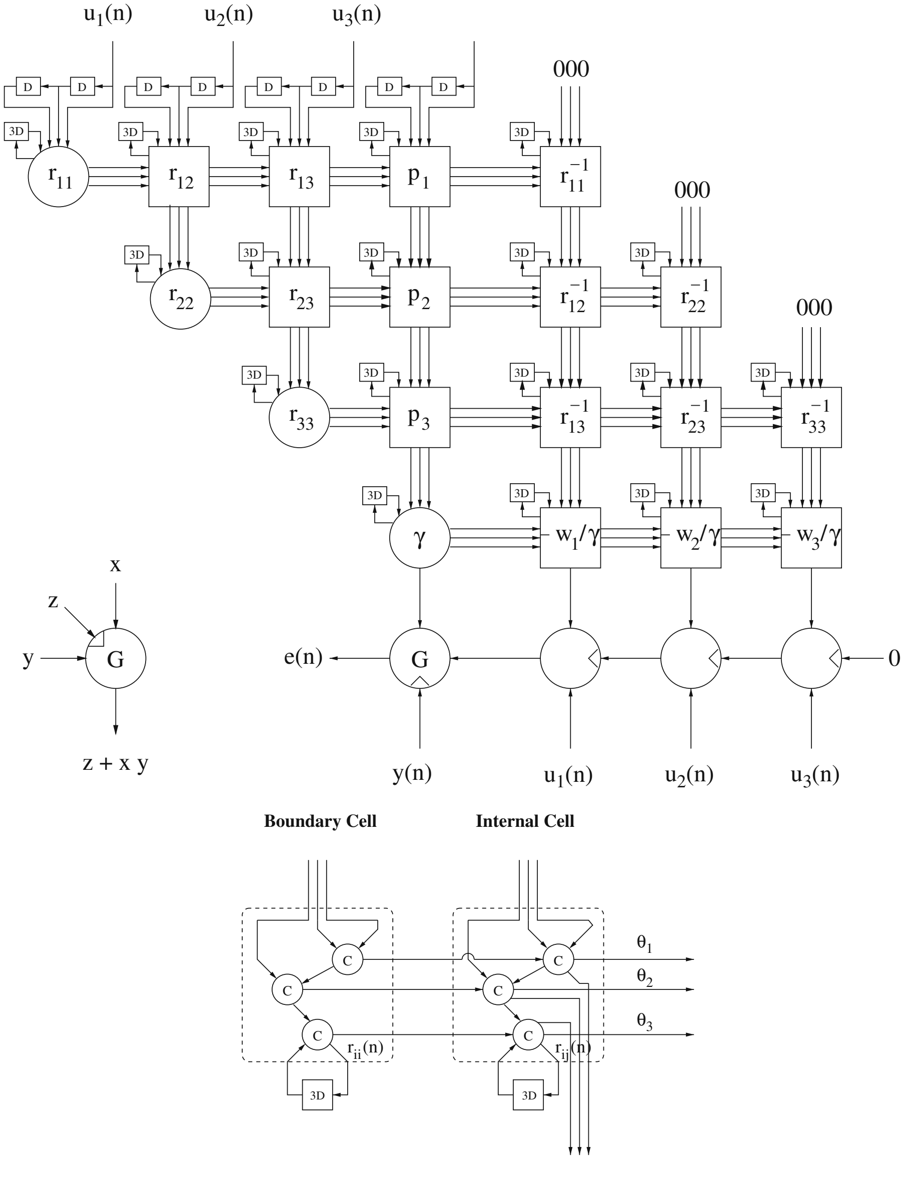


Fig. 10.18 A three–level fine-grain pipelined topology of CORDIC-based double-triangular adaptive inverse QR algorithm.

connection with the traditional multiply–add look-ahead technique. The exploration of the parallelism in the annihilation-reording look-ahead transformation leads to three implementation styles namely pipelining, block processing, and incremental block processing. The implementation complexity in terms of CORDIC units for pipelined realization is the least, and the one for the block processing is the most.

CORDIC-based RLS filters with implicit weight extraction are found useful in applications such as adaptive beamforming, and the ones with explicit weight extraction are found useful in applications such as channel equalization. The application of proposed look-ahead technique to these RLS filters lead to fine-grain pipelined topologies which can be operated at arbitrarily high sample rate. The pipelined algorithms maintain the orthogonality and the stability under finite-precision arithmetic.

QRD-RLS adaptive filters can also be used for adaptive beamforming applications, e.g., the linearly constrained minimum variance (LCMV) adaptive beamformer [33], which will be briefly discussed here. The details can be found in [24]. The LCMV adaptive beamforming is a constrained least squares minimization problem. Solving the constrained minimization problem directly leads to the QRD-MVDR beamforming realization [9]. An alternative is the unconstrained reformulation which leads to the QR decomposition-based generalized sidelobe canceler (GSC) realization [34, 35]. Both MVDR and GSC beamformer can be realized using CORDIC arithmetic. The application of the annihilation-reording look-ahead technique to these adaptive beamforming algorithms leads to fine-grain pipelined topologies [24]. Furthermore, they consist of only Givens rotations which can be mapped onto CORDIC arithmetic-based processors [5].

The implementation complexity in terms of CORDIC units for various RLS-based algorithms and implementation styles are shown in Table 10.2. The implementation complexity of QRD-MVDR and QRD-GSC is obtained from reference [24]. From the table, we see that the pipelining level M is a dimension variable in the complexity expressions for all algorithms and implementation styles. The pipelined and incremental block processing realizations require a linear increasing CORDIC units with an increasing factor of M for the pipelining and a factor of $2M-1$ for the incremental block processing. The complexity factor for the block processing is M^2 which is quadratic with respect to the pipelining level. The adaptive inverse QR algorithm requires approximately two times of CORDIC units as the QRD-RLS algorithm since an extra lower triangular matrix is needed to extract the weight vector. The MVDR topology outperforms GSC topology in terms of complexity by employing a constraint *post-processor* rather than a constraint *pre-processor* for the GSC realization.

Table 10.2 The implementation complexity in terms of CORDIC units for various RLS-based algorithms and implementation styles.

Implementation styles	QRD-RLS	Inverse QR	QRD-MVDR	QRD-GSC
Pipelining	$\frac{1}{2}Mp^2$	Mp^2	$M(\frac{1}{2}p^2+Kp)$	$MK(\frac{1}{2}p^2+p)$
Incremental block	$\frac{1}{2}(2M-1)p^2$	$(2M-1)p^2$	$(2M-1)(\frac{1}{2}p^2+Kp)$	$(2M-1)K(\frac{1}{2}p^2+p)$
Block processing	$\frac{1}{2}M^2p^2$	M^2p^2	$M^2(\frac{1}{2}p^2+Kp)$	$M^2K(\frac{1}{2}p^2+p)$

Appendix

In this appendix, we derive Equations (10.14) and (10.15). At time instance $(n-M)$, apply the QR decomposition to the weighted data matrix $\Lambda^{1/2}(n-M)A(n-M)$ and the reference vector $\mathbf{y}(n)$ as follows:

$$Q(n-M)\Lambda^{1/2}(n-M)\left[A(n-M)\ \mathbf{y}(n-M)\right] = \begin{bmatrix} R(n-M) & \mathbf{p}(n-M) \\ O & \mathbf{v}(n-M) \end{bmatrix}, \quad (10.23)$$

where $R(n-M)$ is p-by-p upper triangular matrix, $\mathbf{p}(n-M)$ and $\mathbf{v}(n-M)$ are p-by-1 and $(n-M-p)$-by-1 vectors, respectively. At time n, the new inputs $U_M(n)$ and $\mathbf{y}_M(n)$ become available processing, we have

$$\left[A(n)\ \mathbf{y}(n)\right] = \begin{bmatrix} A(n-M) & \mathbf{y}(n-M) \\ U_M^{\mathrm{T}}(n) & \mathbf{y}_M(n) \end{bmatrix}. \quad (10.24)$$

Define

$$\bar{\Lambda}^{1/2}(n) = \begin{bmatrix} \lambda^{M/2}\Lambda^{1/2}(n-M) & \\ & I_M \end{bmatrix}, \text{ and} \quad (10.25)$$

$$\bar{Q}(n-M) = \begin{bmatrix} Q(n-M) & \\ & I_M \end{bmatrix}. \quad (10.26)$$

Then

$$\bar{Q}(n-M)\bar{\Lambda}^{1/2}(n)\left[A(n)\ \mathbf{y}(n)\right] = \begin{bmatrix} \lambda^{M/2}R(n-M) & \lambda^{M/2}\mathbf{p}(n-M) \\ O & \lambda^{M/2}\mathbf{v}(n-M) \\ U_M^{\mathrm{T}}(n) & \mathbf{y}_M(n) \end{bmatrix}. \quad (10.27)$$

Notice that here we choose $\bar{\Lambda}^{1/2}(n)$ instead of $\Lambda^{1/2}(n)$. $\Lambda^{1/2}(n)$ differs from $\bar{\Lambda}^{1/2}(n)$ in replacing I_M by $\Lambda^{1/2}(M)$. Using $\Lambda^{1/2}(n)$ will lead to extra operations of the input data. Conversely, using $\bar{\Lambda}^{1/2}(n)$ will not and also not affect the algorithm's convergence behavior due to the existence of $\lambda^{M/2}$ in $\bar{\Lambda}^{1/2}(n)$.

Apply the orthogonal matrix $Q(n)$ which consists of a sequence of Givens rotations to annihilate the input data $U_M(n)$ using $\lambda^{M/2}R(n-M)$ in (10.27), we have

$$\begin{bmatrix} R(n) & \mathbf{p}(n) \\ O & \mathbf{v}(n) \\ O_{M\times N} & \boldsymbol{\alpha}(n) \end{bmatrix} = Q(n) \begin{bmatrix} \lambda^{M/2}R(n-M) & \lambda^{M/2}\mathbf{p}(n-M) \\ O & \lambda^{M/2}\mathbf{v}(n-M) \\ U_M(n) & \mathbf{y}_M(n) \end{bmatrix}, \quad (10.28)$$

which derives the first two columns in (10.14). Next we prove Equation (10.15) and justify the third column in (10.14). From (10.2), we have

$$\mathbf{e}(n-M) = \mathbf{y}(n-M) - A(n-M)\mathbf{w}(n-M). \quad (10.29)$$

By (10.23), we then have

$$Q(n-M)\Lambda^{1/2}(n-M)\mathbf{e}(n-M)=\begin{bmatrix}\mathbf{p}(n-M)\\ \mathbf{v}(n-M)\end{bmatrix}-\begin{bmatrix}R(n-M)\\ O\end{bmatrix}\mathbf{w}(n-M). \tag{10.30}$$

Let

$$\begin{aligned}\boldsymbol{\varepsilon}(n-M)&=Q^{1/2}(n-M)\Lambda^{1/2}(n-M)\mathbf{e}(n-M)\\ \bar{\mathbf{e}}_M(n)&=\mathbf{y}_M(n)-U_M^{\mathrm{T}}(n)\mathbf{w}(n-M)\\ \mathbf{e}_M(n)&=\mathbf{y}_M(n)-U_M^{\mathrm{T}}(n)\mathbf{w}(n),\end{aligned} \tag{10.31}$$

where $\boldsymbol{\varepsilon}(n-M),\bar{\mathbf{e}}_M(n)$, and $\mathbf{e}_M(n)$ are M-by-1 vectors. By (10.1), it is seen that the last element in $\mathbf{e}_M(n)$ is the desired residual error $e(n)$ at time instance n. From (10.30) and (10.31), we obtain

$$\begin{bmatrix}\lambda^{M/2}\boldsymbol{\varepsilon}(n-M)\\ \bar{e}_M(n)\end{bmatrix}=\begin{bmatrix}\lambda^{M/2}\mathbf{p}(n-M)\\ \lambda^{M/2}\mathbf{v}(n-M)\\ \mathbf{y}_M(n)\end{bmatrix}-\begin{bmatrix}\lambda^{M/2}R(n-M)\\ O\\ U_M^{\mathrm{T}}(n)\end{bmatrix}\mathbf{w}(n-M). \tag{10.32}$$

Apply the orthogonal matrix $Q(n)$ to both sides of (10.32) and use (10.28) resulting

$$Q(n)\begin{bmatrix}\lambda^{M/2}\boldsymbol{\varepsilon}(n-M)\\ \mathbf{e}_M(n)\end{bmatrix}=\begin{bmatrix}\mathbf{p}(n)\\ \mathbf{v}(n)\\ \boldsymbol{\alpha}(n)\end{bmatrix}-\begin{bmatrix}R(n)\\ O\\ O\end{bmatrix}\mathbf{w}(n). \tag{10.33}$$

Notice that, after annihilating the input block data $U_M^{\mathrm{T}}(n)$, the weight vector $\mathbf{w}(n-M)$ has been updated to $\mathbf{w}(n)$. Correspondingly, the residual error $\bar{\mathbf{e}}_M(n)$ becomes $\mathbf{e}_M(n)$ as defined in (10.31). Now, moving $Q(n)$ to the right-hand-side of (10.33), and noticing that $Q(n)$ is orthogonal, we obtain

$$\begin{bmatrix}\lambda^{M/2}\boldsymbol{\varepsilon}(n-M)\\ \mathbf{e}_M(n)\end{bmatrix}=Q^{\mathrm{T}}(n)\begin{bmatrix}\mathbf{p}(n)-R(n)\mathbf{w}(n)\\ \mathbf{v}(n)\\ \boldsymbol{\alpha}(n)\end{bmatrix}. \tag{10.34}$$

Since the optimum weight vector $\mathbf{w}(n)$ satisfies $\mathbf{p}(n)-R(n)\mathbf{w}(n)=\mathbf{0}_p$, (10.34) reduces to

$$\begin{bmatrix}\lambda^{M/2}\boldsymbol{\varepsilon}(n-M)\\ \mathbf{e}_M(n)\end{bmatrix}=Q^{\mathrm{T}}(n)\begin{bmatrix}\mathbf{0}_p\\ \mathbf{v}(n)\\ \boldsymbol{\alpha}(n)\end{bmatrix}. \tag{10.35}$$

Noticing that the last element of $\mathbf{e}_M(n)$ is $e(n)$, we have

$$\begin{aligned}e(n)&=\begin{bmatrix}\mathbf{0}_{n-M}^{\mathrm{T}} & \delta_M^{\mathrm{T}}\end{bmatrix}Q^{\mathrm{T}}(n)\begin{bmatrix}\mathbf{0}_p\\ \mathbf{v}(n)\\ \boldsymbol{\alpha}(n)\end{bmatrix}\\ &=\begin{bmatrix}\mathbf{0}_{n-M}^{\mathrm{T}} & \delta_M^{\mathrm{T}}\end{bmatrix}Q^{\mathrm{T}}(n)\begin{bmatrix}O_{(n-M)\times M}\\ I_M\end{bmatrix}\boldsymbol{\alpha}(n).\end{aligned} \tag{10.36}$$

The second equality is due to the fact that $Q^{\mathrm{T}}(n)$ only makes use of elements $\mathbf{0}_p$ and $\boldsymbol{\alpha}(n)$ in the vector $[\mathbf{0}_p^{\mathrm{T}},\mathbf{v}^{\mathrm{T}}(n),\boldsymbol{\alpha}^{\mathrm{T}}(n)]^{\mathrm{T}}$. Taking the transpose on both sides of (10.36), and noticing that $e(n)$ is a scalar, then

$$\begin{aligned} e(n) &= \boldsymbol{\alpha}^{\mathrm{T}}(n)\left[\, O_{M\times(n-M)}\ I_M \,\right]\left(Q(n)\begin{bmatrix} O_{n-M} \\ \delta_M \end{bmatrix}\right) \\ &= \boldsymbol{\alpha}^{\mathrm{T}}(n)\left[\, O_{M\times(n-M)}\ I_M \,\right]\begin{bmatrix} \mathbf{s}(n) \\ \boldsymbol{\gamma}(n) \end{bmatrix} \\ &= \boldsymbol{\alpha}^{\mathrm{T}}(n)\,\boldsymbol{\gamma}(n). \end{aligned} \tag{10.37}$$

The second equality justify the third column in (10.14). This completes the derivation of (10.14) and (10.15).

References

1. S. Haykin, Adaptive Filter Theory. 2nd edition Prentice-Hall, Englewood Cliffs, NJ, USA (1991)
2. J. E. Volder, The CORDIC trigonometric computing technique. IRE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334 (September 1959)
3. Y. H. Hu, Cordic-based VLSI architectures for digital signal processing. IEEE Signal Processing Magazine, no. 3, vol. 9, pp. 16–35 (July 1992)
4. G.J. Hekstra and E. F. Deprettere, Floating point CORDIC. 11th Symposium on Computer Arithmetic, Windsor, Canada, pp. 130–137 (June 1993)
5. E. Rijpkema, G. Hekstra, E. Deprettere, and J. Ma, A strategy for determining a Jacobi specific dataflow processor. IEEE International Conference on Application-Specific Systems, Architectures and Processors, Zurich, Switzerland, pp. 53–64 (July 1997)
6. J. G. McWhirter, Recursive least-squares minimization using a systolic array. Proc. SPIE: Real Time Signal Processing VI, vol. 431, pp. 105–112 (January 1983)
7. W. M. Gentleman and H. T. Kung, Matrix triangularization by systolic arrays. Proceedings SPIE: Real-Time Signal Processing IV, vol. 298, pp. 298–303 (January 1981)
8. T. J. Shepherd, J. G. McWhirter, and J. E.Hudson, Parallel weight extraction from a systolic adaptive beamformer. Mathematics in Signal Processing II (J.G. McWhirter ed.), pp. 775–790, Clarendon Press, Oxford (1990)
9. J. G. McWhirter and T. J. Shepherd, Systolic array processor for MVDR beamforming. IEE Proceedings, vol. 136, pp. 75–80 (April 1989)
10. K. K. Parhi, Algorithm transformation techniques for concurrent processors. Proceedings of the IEEE, vol. 77, pp. 1879–1895 (December 1989)
11. K. K. Parhi and D. G. Messerschmitt, Pipeline interleaving and parallelism in recursive digital filters–Part I: Pipelining using scattered look-ahead and decomposition. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, pp. 1118–1134 (July 1989)
12. K. K. Parhi and D. G. Messerschmitt, Pipeline interleaving and parallelism in recursive digital filters–Part II: Pipelined incremental block filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, pp. 1099–1117 (July 1989)
13. A. P. Chandrakasan, S. Sheng, and R. W. Broderson, Low-power CMOS digital design. IEEE Journal on Solid-State Circuits, vol. 27, pp. 473–484 (April 1992)
14. K. K. Parhi, VLSI Digital Signal Processing Systems, Design and Implementation. John Wiley & Sons, New York, NY, USA (1999)
15. K. J. Raghunath and K. K. Parhi, Pipelined RLS adaptive filtering using Scaled Tangent Rotations (STAR). IEEE Transactions on Signal Processing, vol. 40, pp. 2591–2604 (October 1996)
16. T. H. Y. Meng, E. A. Lee, and D. G. Messerschmitt, Least-squares computation at arbitrarily high speeds, International Conference on Acoustics, Speech, and Signal Processing, ICASSP'1987, Dallas, USA, pp. 1398–1401 (April 1987)

17. G. H. Golub and C. F. V. Loan, Matrix Computation. Johns Hopkins University Press, Baltimore, MD, USA (1989)
18. J. M. Cioffi, The fast adaptive ROTOR RLS algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, pp. 631–653 (April 1990)
19. S. F. Hsieh, K. J. R. Liu, and K. Yao, A unified square-root-free Givens rotation approach for QRD-based recursive least squares estimation. IEEE Transactions on Signal Processing, vol. 41, pp. 1405–1409 (March 1993)
20. S. Hammarling, A note on modifications to Givens plane rotation. IMA Journal of Applied Mathematics, vol. 13, no. 2, pp. 215–218 (1974)
21. J. L. Barlow and I. C. F. Ipsen, Scaled Givens rotations for solution of linear least-squares problems on systolic arrays. SIAM Journal on Scientific and Statistical Computing, vol. 13, no. 5, pp. 716–733 (September 1987)
22. J. Götze and U. Schwiegelshohn, A square-root and division free Givens rotation for solving least-squares problems on systolic arrays. SIAM Journal on Scientific and Statistical Computing, vol. 12, no. 4, pp. 800–807 (July 1991)
23. E. Franrzeskakis and K. J. R. Liu, A class of square-root and division free algorithms and architectures for QRD-based adaptive signal processing. IEEE Transactions on Signal Processing, vol. 42, pp. 2455–2469 (September 1994)
24. J. Ma, K. K. Parhi, and E. F. Deprettere, Annihilation reordering lookahead pipelined CORDIC based RLS adaptive filters and their application to adaptive beamforming. IEEE Transactions on Signal Processing, vol. 48, pp. 2414–2431 (August 2000)
25. C. E. Leiserson, F. Rose, and J. Saxe, Optimizing synchronous circuitry by retiming. 3rd Caltech Conference on VLSI, Pasadena, USA, pp. 87–116 (March 1983)
26. K. K. Parhi, High-level algorithm and architecture transformations for DSP synthesis. Journal of VLSI Signal Processing, vol. 9, pp. 121–143 January 1995
27. E. F. Deprettere, P. Held, and P. Wielage, Model and methods for regular array design. International Journal of High Speed Electronics; Special issue on Massively Parallel Computing–Part II, vol. 4(2), pp. 133–201 (1993)
28. H. Leung and S. Haykin, Stability of recursive QRD-LS algorithms using finite-precision systolic array implementation. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 5, pp. 760–763 (May 1989)
29. M. Moonen and E. F. Deprettere, A fully pipelined RLS-based array for channel equalization. Journal of VLSI Signal Processing, vol. 14, pp. 67–74 (October 1996)
30. R. Gooch and J. Lundell, The CM array: an adaptive beamformer for constant modulus signals. International Conference on Acoustics, Speech, and Signal Processing, ICASSP'1986, Tokyo, Japan, pp. 2523–2526 (April 1986)
31. C.-T. Pan and R. J. Plemmons, Least squares modifications with inverse factorizations: parallel implications. Journal of Computational and Applied Mathematics, vol. 27, pp. 109–127 (September 1989)
32. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. SP-41, no. 1, pp. 20–30 (January 1993)
33. O. L. Frost III, An algorithm for linearly constrained adaptive array processing. Proceedings of the IEEE, vol. 60, no. 8, pp. 926–935 (August 1972)
34. R. L. Hanson and C. L. Lawson, Extensions and applications of the Householder algorithm for solving linear least squares problems. AMS Mathematics of Computation, vol. 23, no. 108, pp. 787-812 (October 1969)
35. S. P. Applebaum and D. J. Chapman, Adaptive arrays with main beam constraints. IEEE Transactions on Antennas and Propagation, vol. AP-24, no. 5, pp. 650–662 (September 1976)

Chapter 11
Weight Extraction of Fast QRD-RLS Algorithms

Stefan Werner and Mohammed Mobien

Abstract The main limitation of fast QR-decomposition recursive least-squares (FQRD-RLS) algorithms is that they lack an explicit weight vector term. Furthermore, they do not directly provide the variables allowing for a straightforward computation of the weight vector as is the case with the conventional QRD-RLS algorithm, where a back-substitution procedure can be used to compute the coefficients. Therefore, the applications of the FQRD-RLS algorithms are limited to certain output-error-based applications (e.g., noise cancelation), or to applications that can provide a decision-feedback estimate of the training signal (e.g., equalizers operating in decision-directed mode). This chapter presents some observations that allow us to apply the FQRD-RLS algorithms in applications that traditionally have required explicit knowledge of the transversal weights. Section 11.1 reviews the basic concepts of QRD-RLS and the particular FQRD-RLS algorithm that is used in the development of the new applications. Section 11.2 describes how to identify the implicit FQRD-RLS transversal weights. This allows us to use the FQRD-RLS algorithm in a system identification setup. Section 11.3 applies the FQRD-RLS algorithm to burst-trained systems, where the weight vector is updated during a training phase and then kept fixed and used for output filtering. Section 11.4 applies the FQRD-RLS algorithm for single-channel active noise control, where a copy of the adaptive filter is required for filtering a different input sequence than that of the adaptive filter. A discussion on multichannel and lattice extensions is provided in Section 11.5. Finally, conclusions are drawn in Section 11.6.

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

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

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_11, © Springer Science+Business Media, LLC 2009

11.1 FQRD-RLS Preliminaries

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

11.1.1 QR decomposition algorithms

The RLS algorithm minimizes the following cost function:

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

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

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

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

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

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

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

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

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

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

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

and

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

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

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

The QRD-RLS algorithm is complete with the definition of the *a priori* error value $e(k)=e_{q1}(k)/\gamma(k)$, where $\gamma(k)$ is a scalar found in matrix $\mathbf{Q}_{\theta}(k)$, see (11.8).

11.1.2 FQR_POS_B algorithm

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

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11.2 System Identification with FQRD-RLS

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

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

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

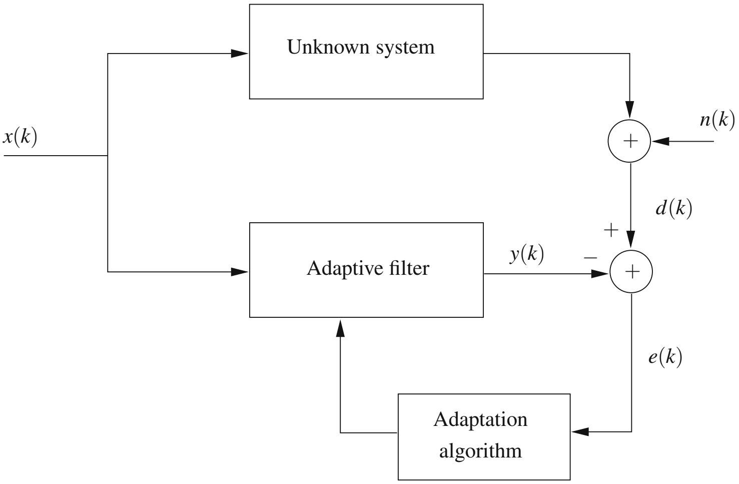


Fig. 11.1 Schematic diagram of a system-identification application.

11.2.1 Weight extraction in the FQRD-RLS algorithm

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

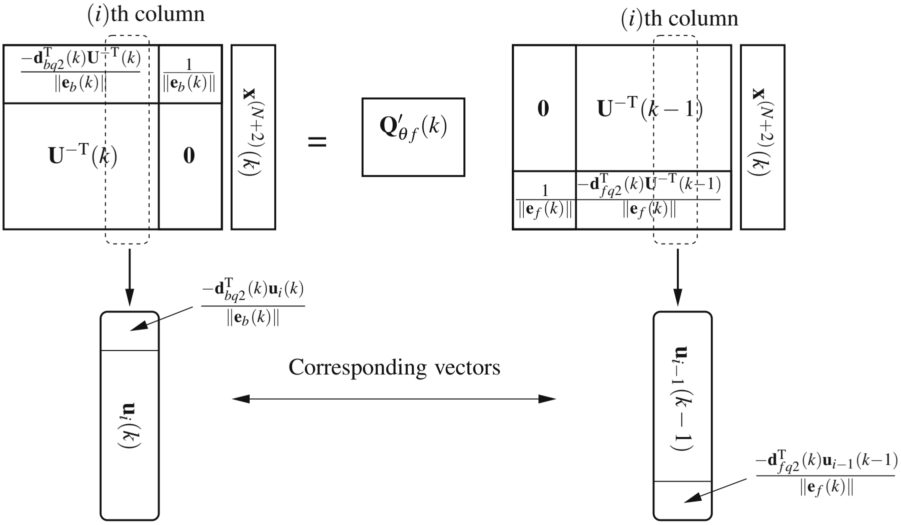

(i)th column
$\frac{-\mathbf{d}_{bq2}^{\mathrm{T}}(k)\mathbf{U}^{-\mathrm{T}}(k)}{\|\mathbf{e}_b(k)\|}$
$\frac{1}{\|\mathbf{e}_b(k)\|}$
$\mathbf{U}^{-\mathrm{T}}(k)$
$\mathbf{0}$
$\mathbf{x}^{(N+2)}(k)$
$=$
$\mathbf{Q}'_{\theta f}(k)$
(i)th column
$\mathbf{0}$
$\mathbf{U}^{-\mathrm{T}}(k-1)$
$\frac{1}{\|\mathbf{e}_f(k)\|}$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

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

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

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

where

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

The update is given by [5]

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

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

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

which gives the relation in (11.23).

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

11.2.2 Example

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

Table 11.1 FQRD-RLS WE algorithm.

```
Weight extraction

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

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

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

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

Table 11.2 Computational complexity of WE [8].

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

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

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

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

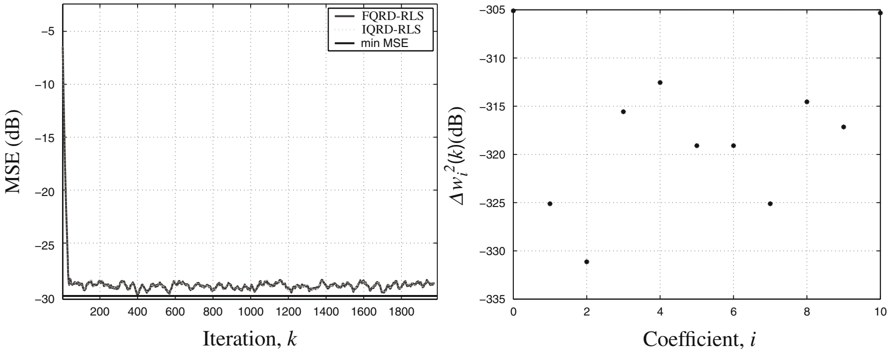


Fig. 11.3 Learning curves of the FQRD-RLS and the IQRD-RLS algorithms (left figure), squared difference between coefficient weights of the IQRD-RLS and the FQRD-RLS algorithms.

11.3 Burst-trained Equalizer with FQRD-RLS

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

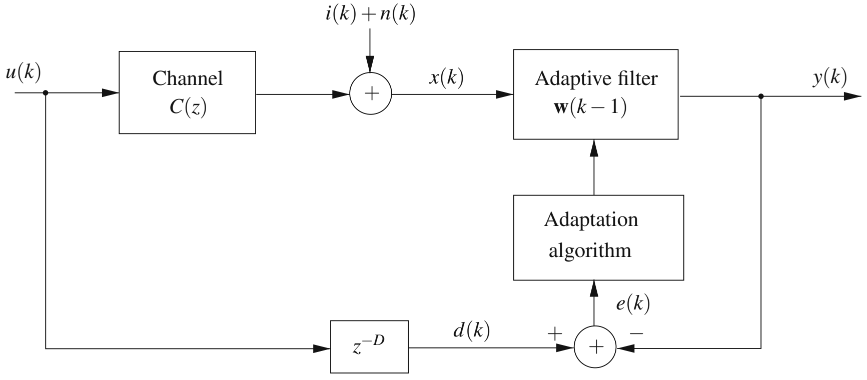


Fig. 11.4 Schematic diagram of an adaptive equalizer.

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

11.3.1 Problem description

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

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

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

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

11.3.2 Equivalent-output filtering

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

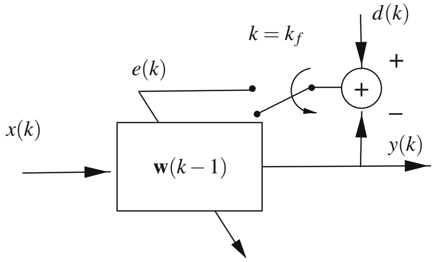


Fig. 11.5 Operation of a burst-trained equalizer. The adaptive filter coefficients are updated during *training mode* ($k \leq k_f$), and kept fixed and used for output filtering in *data mode* ($k > k_f$). Note that there is no weight update after $k > k_f$.

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

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

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

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

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

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

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

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

where

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

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

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

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

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

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

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

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

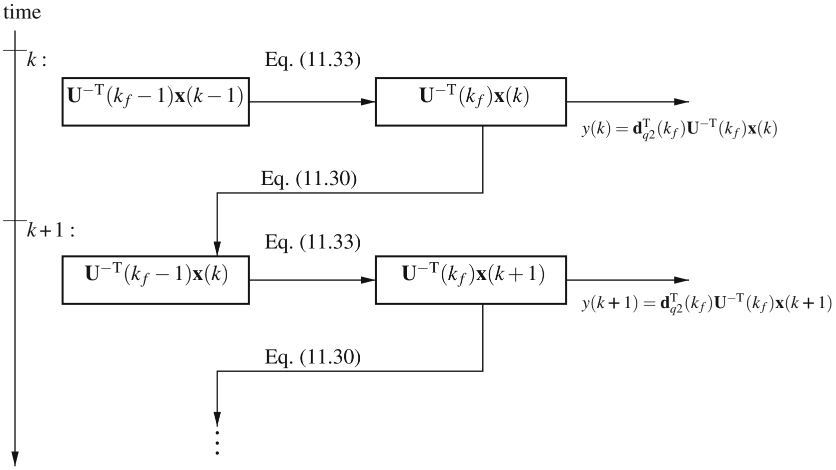

time
k :
Eq. (11.33)
$\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k-1)$
$\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$
$y(k) = \mathbf{d}^{\mathrm{T}}_{q2}(k_f)\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k)$
Eq. (11.30)
$k+1$:
Eq. (11.33)
$\mathbf{U}^{-\mathrm{T}}(k_f-1)\mathbf{x}(k)$
$\mathbf{U}^{-\mathrm{T}}(k_f)\mathbf{x}(k+1)$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

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

11.3.3 Equivalent-output filtering with explicit weight extraction

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

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

Equivalent-output filtering

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

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

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

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

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

with initial values

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

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

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

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

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

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

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

where $c(k) = x(k)\ \forall k > k_f$ and $c(k) = 0\ \forall k \leq k_f$. We see that (11.37) allows us to extract the weights in a distributed manner, i.e., one weight per each new incoming sample. Such "on-the-fly" extraction provides us with all the weights after $N+1$ iterations, and still produces the correct output $y_c(k)$ before all the weights are acquired (according to (11.37)). Invoking Lemmas 1 and 2 using a unit pulse in parallel with (11.37) will sequence out the weights $w_i(k_f)$ at the time instant they show up in (11.37).

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

11.3.4 Example

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

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

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

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

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

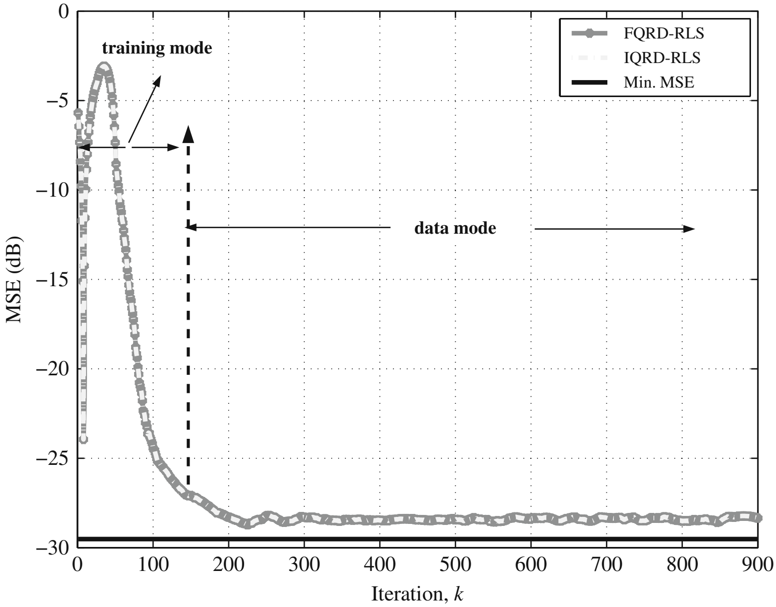


Fig. 11.7 Learning curves of the FQRD-RLS and the IQRD-RLS algorithms.

11.4 Active Noise Control and FQRD-RLS

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

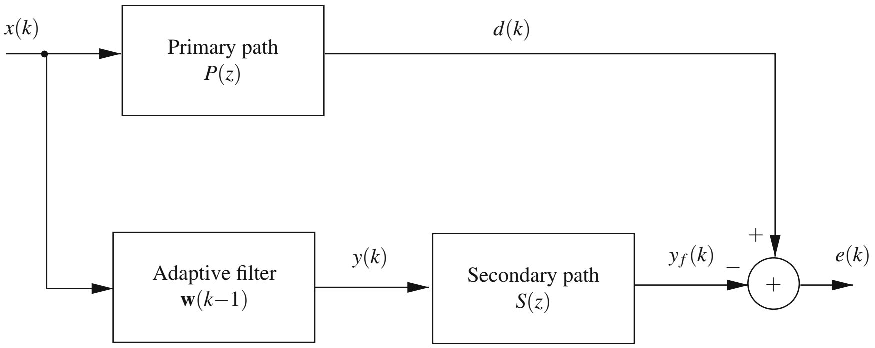


Fig. 11.8 Schematic diagram of an ANC system.

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

11.4.1 Filtered-x RLS

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

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

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

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

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

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

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

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

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

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

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

giving the following updating expression

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

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

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

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

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

11.4.2 Modified filtered-x FQRD-RLS

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

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

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

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

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

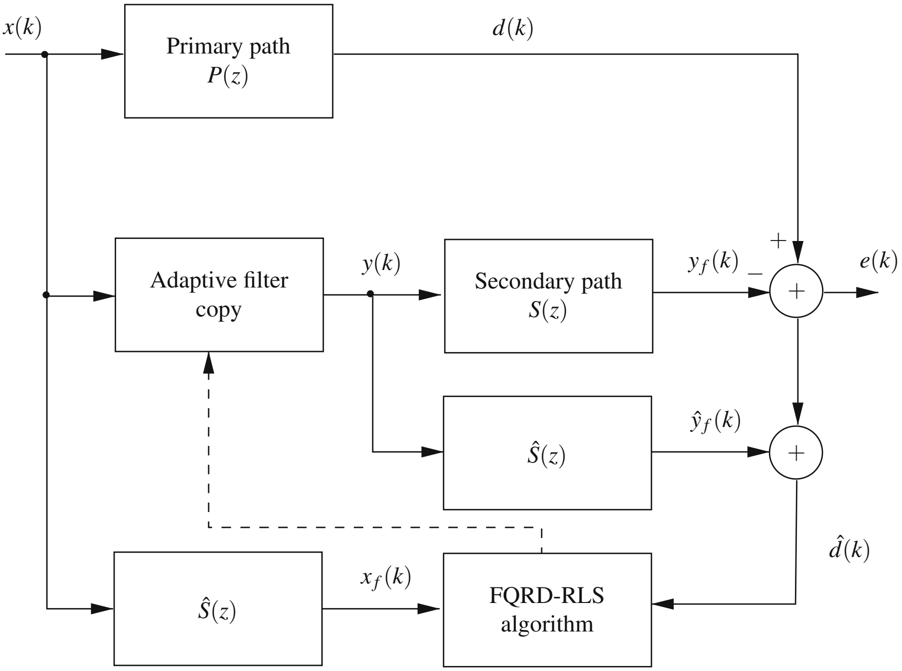


Fig. 11.9 FQRD-RLS in an ANC system.

rate, say once every K samples. This type of intermediate solution was considered in [11], where a QRD least-squares lattice (QRD-LSL) algorithm was employed in the lower branch of Figure 11.9. Obviously, such an approach will no longer yield identical results to an IQRD-RLS algorithm, and the convergence behavior will certainly be different.

Our goal here is to reproduce, at each iteration, the exact output $y(k)$ associated with the weight vector $\mathbf{w}(k-1)$ embedded in the FQRD-RLS algorithm. The total computational complexity for calculating $y(k)$ and updating $\mathbf{w}(k-1)$ will be $\mathscr{O}[N]$ per iteration. The resulting structure will yield exactly the same result as the modified filtered-x IQRD-RLS in algorithm [11], while reducing the computational complexity by an order of magnitude.

The solution is very similar to the problem dealt with in Section 11.3, where the weight vector in the FQRD-RLS algorithm was used for fixed filtering. The main difference here is that the weight vector in the lower branch is continuously updated by the FQRD-RLS algorithm.

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

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

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

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

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

where $*$ is a “don’t-care” variable.

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

Table 11.4 Modified filtered-x FQRD-RLS algorithm.

Output filtering in the upper branch of Figure 11.9

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

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

11.4.3 Example

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

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

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

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

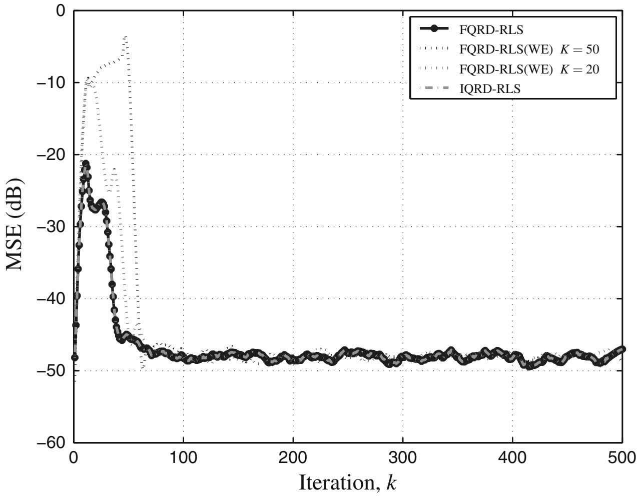


Fig. 11.10 Learning curves of the FQRD-RLS and IQRD-RLS algorithms implemented in the modified filtered-x structure. For comparison purposes, FQRD-RLS WE solutions are shown where the coefficients are extracted at a reduced rate, for every $K = 20$ (intermediary curve) or $K = 50$ (upper curve) iterations,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

11.5 Multichannel and Lattice Implementations

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

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

11.6 Conclusion

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

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

References

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

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

Chapter 12
On Linearly Constrained QRD-Based Algorithms

Shiunn-Jang Chern

Abstract The linearly constrained adaptive filtering (LCAF) technique has been extensively used in many engineering applications. In this chapter, we introduce the linearly constrained minimum variance (LCMV) filter, implemented using the linearly constrained recursive least squares (RLS) criterion, with the inverse QR decomposition (IQRD) approach. First, the direct form of recursively updating the constrained weight vector of LS solution based on the IQRD is developed, which is named as the LC-IQRD-RLS algorithm. With the IQRD approach, the parameters related to the *Kalman gain* are evaluated via *Givens* rotations and the LS weight vector can be computed without back-substitution. This algorithm is suitable to be implemented using systolic arrays with very large scale integration technology and DSP devices. For the sake of simplification, an alternative indirect approach, referred to as the generalized sidelobe canceler (GSC), is adopted for implementing the LCAF problem. The GSC structure essentially decomposes the adaptive weight vector into constrained and unconstrained components. The unconstrained component can then be freely adjusted to meet any criterion since the constrained component will always ensure that the constraint equations are satisfied. The indirect implementation could attain the same performance as that using the direct constrained approach and possesses better numerical properties. Via computer simulation, the merits of the LC-IQRD-RLS algorithms over the conventional LC-RLS algorithm and its modified version are verified.

12.1 Introduction

The linearly constrained (LC) adaptive filtering (LCAF) technique is known to have many applications in areas such as minimum variance spectral analysis, time delay estimation, and antenna array signal processing [1–5]. More recently, these

Shiunn-Jang Chern
National Sun Yat-Sen University, Kaohsiung City, Taiwan – R.O.C.
e-mail: chern@ee.nsysu.edu.tw

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_12, © Springer Science+Business Media, LLC 2009

constrained approaches have been applied to wireless communication systems for multiuser detection [6, 7]. Among the adaptive filtering algorithms, in most practical applications, the RLS algorithm has shown to offer better convergence rate and steady-state mean-squared error (MSE) over the least mean squares (LMS)-based algorithms. Unfortunately, the widespread acceptance of the RLS filter has been refrained due to numerical instability problems when it is implemented in limited-precision environments. Performance degradation is especially noticeable for the family of techniques collectively known as “fast” RLS filters [8–10]. To circumvent this problem, a well-known numerical stable RLS algorithm, which is called the QR-decomposition RLS (QRD-RLS) algorithm was proposed [8, 11–13]. It computes the QR decomposition (triangular orthogonalization) of the input data matrix using Givens rotation and then solves the LS weight vector by means of the back-substitution procedure.

The QRD-RLS algorithm can be implemented using the systolic array [14–17] with very large scale integration (VLSI) technology and DSP devices. However, in some practical applications, if the LS weight vector is desired in each iteration, back-substitution steps must be performed accordingly. Due to the fact that back-substitution is a costly operation to be performed in an array structure, the so-called inverse QRD-RLS (IQRD-RLS) algorithm proposed in [18] is preferred, for the LS weight vector can be computed without implementing back-substitution.

In this chapter, we first introduce the LC-RLS filtering algorithm based on an IQRD, where a direct form of recursively updating the constrained weight vector of LS solution is developed. The basic approach of the LC-IQRD-RLS algorithm is very similar to that discussed in [2]. That is, based on the Kalman gain of the conventional IQRD-RLS algorithm, the LC-IQRD-RLS algorithm can be developed where the unconstrained form of the weight vector and the *a priori* estimation error can be avoided. In the IQRD-RLS algorithm, the parameters related to the Kalman gain are evaluated using the *Givens rotations* (QR decomposition), which is quite different from the one discussed in [2] (using the fast RLS algorithm), yielding different development. Usually, the IQRD-RLS algorithm has better numerical accuracy than the “fast” RLS algorithm. Indeed, the numerical instability may cause the constraint drift problem [19] for the constrained approach based on the conventional fast RLS algorithm, named the constrained fast LS (CFLS) algorithm [2].

In this chapter, an alternative indirect approach, referred to as the generalized side-lobe canceler (GSC), is employed [6, 8, 20, 21] for various applications. The GSC structure essentially decomposes the adaptive weight vector into constrained and unconstrained components. The unconstrained component can then be freely adjusted to meet any criterion since the constrained component will always ensure that the constraint equations are satisfied. The GSC-based algorithm could attain the same performance as the direct constrained approach and possesses better numerical properties.

In [22], the authors have proved that the optimal linearly constrained solution derived with the direct and indirect (GSC structure) structures are equivalent for the conjugate gradient and LMS-based algorithms, if the blocking matrix satisfies the following condition, e.g., $\mathbf{B}^{\mathrm{H}}\mathbf{B} = \mathbf{I}$. Furthermore, in [21] a more general proof of the equivalency of direct and GSC structures for the LMS-based algorithms was given.

This chapter is organized as follows. First, in Section 12.2, we derive the optimal linearly constrained LS weight vector solution, based on the IQRD, and discuss the rationale behind it. After that, in Section 12.3, the LC-IQRD-RLS algorithm is developed and applied to the linearly constrained minimum variance (LCMV) filtering problems to achieve the desired performance. In Section 12.4, an alternative indirect approach using the GSC structure is developed. To document the merits of the proposed algorithm, in Section 12.5, two applications with computer simulations results are given to show the efficiency in terms of fast convergence and numerical stability of the LC-IQRD-RLS and the GSC-IQRD-RLS algorithms over the constrained fast RLS (CFLS) algorithm. Finally, we give a conclusion in Section 12.6 to summarize this chapter.

Notations and used symbols: Vectors and matrices are denoted by boldface lower and upper case letters. All vectors are column vectors. $(\cdot)^{-1}$, $(\cdot)^{*}$, $(\cdot)^{\mathrm{T}}$, and $(\cdot)^{\mathrm{H}}$ denote inverse, complex conjugate, transpose, and conjugate transpose, respectively. $\|\cdot\|$ denotes Frobenius norm and $\mathbf{I}_N$ is the $N \times N$ identity matrix. Null matrix or vector is denoted by $\mathbf{0}$ with corresponding dimension.

12.2 Optimal Linearly Constrained QRD-LS Filter

As an introduction, we consider the configuration of the LCAF as depicted in Figure 12.1.[1] Here, $\mathbf{x}(k) = [x(k), x(k-1), \ldots, x(k-N)]^{\mathrm{T}}$ denotes the vector of sampled input signals at the time index k and the weight vector is defined as $\mathbf{w}(k) = [w_0(k), w_1(k), \ldots, w_N(k)]^{\mathrm{T}}$. In this chapter, we focus on the LCMV filtering problem, which uses the blind optimization approach, with the exponentially weighted least-squares (LS) method, the cost function is defined as

$$J_{LS}(\mathbf{w}) = \sum_{i=0}^{k} \lambda^{k-i} |y(i)|^2 = \sum_{i=0}^{k} \lambda^{k-i} |\mathbf{w}^{\mathrm{H}}\mathbf{x}(i)|^2. \tag{12.1}$$

The LS solution of $\mathbf{w}(k)$ is obtained by minimizing (12.1) with respect to $\mathbf{w}$, subject to multiple constraints. In (12.1), the parameter λ $(0 \ll \lambda \leq 1)$ is the forgetting factor that controls the speed of convergence and tracking capability of the algorithm. For convenience, we rewrite (12.1) in a matrix form, i.e.,

$$J_{LS}(\mathbf{w}) = \|\boldsymbol{\Lambda}^{1/2}(k)\mathbf{y}(k)\|^2 = \|\boldsymbol{\Lambda}^{1/2}(k)\mathbf{X}(k)\mathbf{w}\|^2 \tag{12.2}$$

[1] Note that the same algorithms developed in this chapter can also be employed for the case of a constrained linear combiner, when the input signal vector does not correspond to a delay line.

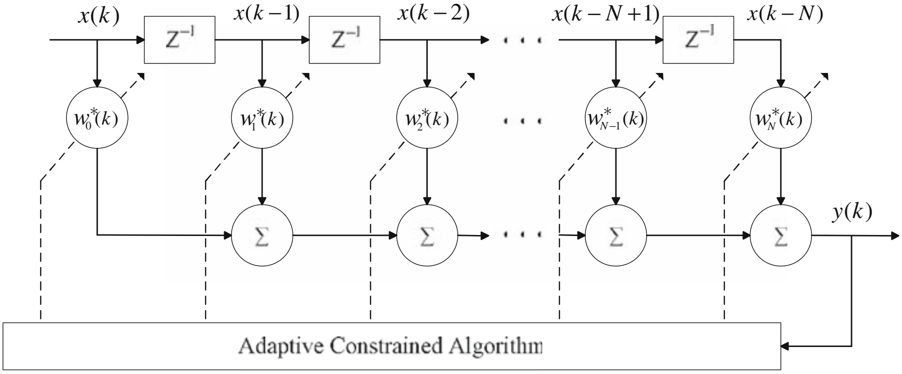


Fig. 12.1 The structure of a linearly constrained FIR filter.

where $\mathbf{y}(k) = [y(0), y(1), ...y(k)]^{\mathrm{T}}$ is denoted as the output vector. Also, $\boldsymbol{\Lambda}^{1/2}(k) = diag[\sqrt{\lambda^k}, \sqrt{\lambda^{k-1}}, \cdots, \sqrt{\lambda}, 1]$ is a diagonal matrix and $\mathbf{X}(k)$ is the $(k+1) \times (N+1)$ data matrix denoted by $\mathbf{X}(k) = [\mathbf{x}(0), \mathbf{x}(1), ..., \mathbf{x}(k)]^{\mathrm{H}}$. It can be noticed that the definition of the data matrix $\mathbf{X}(k)$ in this chapter is slightly different from the definition found in (2.14), Chapter 2; however, this difference in the way the input data matrix is defined results in equivalent algorithms. In the conventional QRD-RLS algorithm, an orthogonal matrix $\mathbf{Q}(k)$ of order $(k+1) \times (k+1)$, is used to perform the triangular orthogonalization of the data matrix $\boldsymbol{\Lambda}^{1/2}(k)\mathbf{X}(k)$ by means of *Givens rotations* [8, 12], that is,

$$\mathbf{Q}(k)\boldsymbol{\Lambda}^{1/2}(k)\mathbf{X}(k) = \begin{bmatrix} \mathbf{U}(k) \\ \mathbf{O} \end{bmatrix}, \tag{12.3}$$

where $\mathbf{U}(k)$ is an $(N+1) \times (N+1)$ upper triangular matrix, and $\mathbf{O}$ is a $(k-N) \times (N+1)$ null matrix.

We note that, in order to be consistent with the definition of the data matrix $\mathbf{X}(k)$, the definition of $\mathbf{U}(k)$ is also slightly different from that given in introductory chapters. Also, the *Givens rotation* is known to have the ability to be implemented in parallel and systolic structure. In consequence, (12.2) can be rewritten as

$$J_{LS}(\mathbf{w}) = \|\mathbf{U}(k)\mathbf{w}\|^2 . \tag{12.4}$$

For the linearly constrained optimization, (12.4) is minimized subjects to L linear constraints, e.g., $\mathbf{C}^{\mathrm{H}}\mathbf{w} = \mathbf{f}$, where the $(N+1) \times L$ constraint matrix is denoted as $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_L]$ and $\mathbf{f} = [f_1, f_2, ..., f_L]^{\mathrm{T}}$ is the L-element response column vector. Proceeding in a similar way as in [2], via Lagrange multipliers with the QR-decomposition approach, the linearly constrained LS solution is given by [1]

$$\mathbf{w}(k) = \left[\mathbf{U}^{\mathrm{H}}(k)\mathbf{U}(k)\right]^{-1}\mathbf{C}\left\{\mathbf{C}^{\mathrm{H}}\left[\mathbf{U}^{\mathrm{H}}(k)\mathbf{U}(k)\right]^{-1}\mathbf{C}\right\}^{-1}\mathbf{f}\,. \tag{12.5}$$

Based on the optimal constrained LS solution of (12.5), in next section we will develop the recursive form of (12.5), named the LC-IQRD-RLS algorithm.

12.3 The Adaptive LC-IQRD-RLS Filtering Algorithm

To derive the recursive form of (12.5) for using in a LCAF, as depicted in Figure 12.2, we define a new $(N+1)\times(N+1)$ matrix $\mathbf{S}(k)$, i.e.,

$$\mathbf{S}(k) = [\mathbf{U}^{\mathrm{H}}(k)\mathbf{U}(k)]^{-1}. \tag{12.6}$$

Its inverse can be easily shown to be equivalent to the following definition:

$$\begin{aligned}\mathbf{S}^{-1}(k) &= \mathbf{U}^{\mathrm{H}}(k)\mathbf{U}(k) \\ &= \mathbf{X}^{\mathrm{H}}(k)\boldsymbol{\Lambda}(k)\mathbf{X}(k) \\ &= \sum_{i=1}^{k}\lambda^{k-i}\mathbf{x}(i)\mathbf{x}^{\mathrm{H}}(i).\end{aligned} \tag{12.7}$$

For convenience, we define $\boldsymbol{\Gamma}(k) = \mathbf{S}(k)\mathbf{C}$ and $\boldsymbol{\Phi}(k) = \mathbf{C}^{\mathrm{H}}\boldsymbol{\Gamma}(k) = \mathbf{C}^{\mathrm{H}}\mathbf{S}(k)\mathbf{C}$; as a consequence, (12.5) can be expressed as

$$\mathbf{w}(k) = \boldsymbol{\Gamma}(k)\boldsymbol{\Phi}^{-1}(k)\mathbf{f}. \tag{12.8}$$

We may view (12.8) as the LCMV weight vector solution implemented by the LS approach with the IQRD. In what follows, based on the inverse *Cholesky factor* $\mathbf{U}^{-1}(k)$, the recursive form of (12.8) is developed. Also, some of the useful parameters and alternative recursive equations of $\boldsymbol{\Gamma}(k)$ and $\boldsymbol{\Phi}^{-1}(k)$, which are related to the inverse *Cholesky factor*, are derived. Recalling from [11], the upper triangular matrix of $\mathbf{U}(k)$ as defined in (12.3) can be written in a recursive form

$$\begin{bmatrix}\mathbf{U}(k)\\ \mathbf{0}^{\mathrm{T}}\end{bmatrix} = \mathbf{T}(k)\begin{bmatrix}\sqrt{\lambda}\mathbf{U}(k-1)\\ \mathbf{x}^{\mathrm{H}}(k)\end{bmatrix}, \tag{12.9}$$

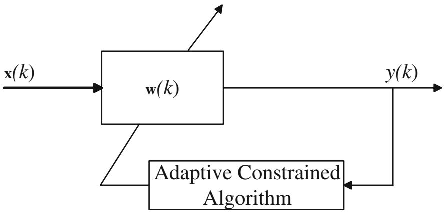


Fig. 12.2 The block diagram of the adaptive linearly constrained filter.

where $\mathbf{T}(k)$ is the $(N+2)\times(N+2)$ orthogonal matrix, which annihilates the Hermitian of the input vector, $\mathbf{x}^{\mathrm{H}}(k)$, by rotating it into $\sqrt{\lambda}\mathbf{U}(k-1)$. Thus, matrix $\mathbf{T}(k)$ can be formed as the product of $(N+1)$ *Givens rotations*. By multiplying both sides of (12.9) with their respective Hermitian on the left, it gives

$$\mathbf{U}^{\mathrm{H}}(k)\mathbf{U}(k) = \lambda\mathbf{U}^{\mathrm{H}}(k-1)\mathbf{U}(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{H}}(k). \tag{12.10}$$

By using the matrix inversion lemma in (12.10), and after some mathematical manipulation, we obtain

$$\mathbf{U}^{-1}(k)\mathbf{U}^{-\mathrm{H}}(k) = \frac{1}{\lambda}\mathbf{U}^{-1}(k-1)\mathbf{U}^{-\mathrm{H}}(k-1) - \mathbf{g}(k)\mathbf{g}^{\mathrm{H}}(k), \tag{12.11}$$

where the intermediate vector, $\mathbf{g}(k)$ is defined by

$$\mathbf{g}(k) = \frac{\mathbf{U}^{-1}(k-1)\mathbf{z}(k)}{\sqrt{\lambda}t(k)}. \tag{12.12}$$

The scalar variable $t(k)$ and intermediate vector $\mathbf{z}(k)$ of (12.12) are defined, respectively, by $t(k) = \sqrt{1+\mathbf{z}^{\mathrm{H}}(k)\mathbf{z}(k)}$ and

$$\mathbf{z}(k) = \frac{\mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{x}(k)}{\sqrt{\lambda}}. \tag{12.13}$$

Equation (12.11) implies the existence of an $(N+2)\times(N+2)$ orthogonal matrix $\mathbf{P}(k)$, which annihilates vector $\mathbf{z}(k)$, starting from the top, by rotating them into the element at the bottom of the augment vector $\left[\mathbf{z}^{\mathrm{T}}(k)\ 1\right]^{\mathrm{T}}$ [1], which is given by

$$\mathbf{P}(k)\begin{bmatrix}\mathbf{z}(k)\\ 1\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ t(k)\end{bmatrix}. \tag{12.14}$$

While updating the lower triangular matrix $\mathbf{U}^{-\mathrm{H}}(k)$ from $\mathbf{U}^{-\mathrm{H}}(k-1)$, with the rotation matrix $\mathbf{P}(k)$, we obtain the intermediate vector $\mathbf{g}^{\mathrm{H}}(k)$, i.e.,

$$\mathbf{P}(k)\begin{bmatrix}\lambda^{-1/2}\mathbf{U}^{-\mathrm{H}}(k-1)\\ \mathbf{0}^{\mathrm{T}}\end{bmatrix} = \begin{bmatrix}\mathbf{U}^{-\mathrm{H}}(k)\\ \mathbf{g}^{\mathrm{H}}(k)\end{bmatrix}. \tag{12.15}$$

We note that both $\mathbf{g}(k)$ and $\mathbf{z}(k)$ just described are computed using the same set of *Givens rotations*, when $\mathbf{U}^{-\mathrm{H}}(k)$ is updated from $\mathbf{U}^{-\mathrm{H}}(k-1)$. It is of interest to point out that vector $\mathbf{g}(k)$ scaled by $t(k)$, i.e.,

$$\mathbf{k}(k) = \frac{\mathbf{g}(k)}{t(k)}, \tag{12.16}$$

can be viewed as the *adaptation* or *Kalman gain* of the IQRD-RLS algorithm. Substituting (12.6) into (12.11), with definition (12.12) and (12.13), we can easily show that

$$\mathbf{S}(k) = \lambda^{-1}[\mathbf{S}(k-1) - \mathbf{k}(k)\mathbf{x}^{\mathrm{H}}(k)\mathbf{S}(k-1)], \tag{12.17}$$

with

$$\mathbf{k}(k) = \frac{\lambda^{-1}\mathbf{S}(k-1)\mathbf{x}(k)}{1+\lambda^{-1}\mathbf{x}^{\mathrm{H}}(k)\mathbf{S}(k-1)\mathbf{x}(k)}. \tag{12.18}$$

With the results of (12.17) and (12.18), we can prove that $\mathbf{k}(k) = \mathbf{S}(k)\mathbf{x}(k)$. Now, with the recursive equation of (12.17), the $(N+1)\times K$ matrix $\boldsymbol{\Gamma}(k) = \mathbf{S}(k)\mathbf{C}$ can be rewritten in a recursive form by doing the right multiplication on both sides of (12.17) by $\mathbf{C}$, i.e.

$$\begin{aligned}\boldsymbol{\Gamma}(k) &= \lambda^{-1}[\boldsymbol{\Gamma}(k-1) - \mathbf{k}(k)\mathbf{x}^{\mathrm{H}}(k)\boldsymbol{\Gamma}(k-1)] \\ &= \lambda^{-1}\boldsymbol{\Gamma}(k-1) - \mathbf{g}(k)\boldsymbol{\alpha}(k),\end{aligned} \tag{12.19}$$

where $\boldsymbol{\alpha}(k) = \mathbf{g}^{\mathrm{H}}(k)\mathbf{C}$. The recursive equation of $\boldsymbol{\Phi}(k) = \mathbf{C}^{\mathrm{H}}\boldsymbol{\Gamma}(k)$, previously defined when introducing (12.8), can be expressed as

$$\boldsymbol{\Phi}(k) = \lambda^{-1}[\boldsymbol{\Phi}(k-1) - \mathbf{k}(k)\mathbf{x}^{\mathrm{H}}(k)\boldsymbol{\Phi}(k-1)]. \tag{12.20}$$

Applying the matrix inversion lemma to (12.20), we have

$$\boldsymbol{\Phi}^{-1}(k) = \lambda[\mathbf{I} + \sqrt{\lambda}\mathbf{q}(k)\boldsymbol{\alpha}(k)]\boldsymbol{\Phi}^{-1}(k-1), \tag{12.21}$$

where $\mathbf{q}(k)$ is defined as

$$\mathbf{q}(k) = \frac{\sqrt{\lambda}\boldsymbol{\Phi}^{-1}(k-1)\boldsymbol{\alpha}^{\mathrm{H}}(k)}{1-\lambda\boldsymbol{\alpha}(k)\boldsymbol{\Phi}^{-1}(k-1)\boldsymbol{\alpha}^{\mathrm{H}}(k)}. \tag{12.22}$$

Based on (12.20) and (12.21), we can show that

$$\mathbf{q}(k) = \lambda^{-1/2}[\boldsymbol{\Phi}^{-1}(k)\boldsymbol{\alpha}(k)]. \tag{12.23}$$

Indeed, (12.23) is very useful for deriving the recursive form of (12.5). Finally, by applying the recursive equations defined in (12.19) and (12.21) to (12.8) and after simplification, we have the recursive implementation of (12.5), named the LC-IQRD-RLS algorithm

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \lambda[\mathbf{g}(k) - \sqrt{\lambda}\boldsymbol{\Gamma}(k)\mathbf{q}(k)]\boldsymbol{\alpha}(k)\boldsymbol{\Phi}^{-1}(k-1)\mathbf{f}. \tag{12.24}$$

With the definition of (12.12) and (12.13), (12.24) can be further simplified as

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \boldsymbol{\rho}(k)e(k), \tag{12.25}$$

where

$$\boldsymbol{\rho}(k) = \mathbf{k}(k) - \frac{\sqrt{\lambda}}{t(k)}\boldsymbol{\Gamma}(k)\mathbf{q}(k), \text{ and} \tag{12.26}$$

$$e(k) = \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k). \tag{12.27}$$

In (12.27), $e(k)$ can be viewed as the *a priori* output of the LCMV filter. This completes our derivation for the adaptive LC-IQRD-RLS algorithm, which is summarized in Table 12.1 for reference. For simplification, an alternative indirect approach, within the GSC framework, of the optimal linearly constrained LS solution, based on the IQRD-RLS algorithm, is developed in the next section.

Table 12.1 Summary of the adaptive LC-IQRD-RLS algorithm.

LC-IQRD-RLS

- Initialization (δ=small positive constant):

$$\begin{aligned} \mathbf{U}^{-1}(0) &= \delta^{-1}\mathbf{I} \\ \boldsymbol{\Gamma}(0) &= \mathbf{U}^{-1}(0)\mathbf{U}^{-\mathrm{H}}(0)\mathbf{C} \\ \boldsymbol{\Phi}^{-1}(0) &= [\mathbf{C}^{\mathrm{H}}\boldsymbol{\Gamma}(0)]^{-1} \\ \mathbf{w}(0) &= \boldsymbol{\Gamma}(0)[\mathbf{C}^{\mathrm{H}}\boldsymbol{\Gamma}(0)]^{-1}\mathbf{f} \end{aligned}$$

- For k=1,2,..., do

1. Compute the intermediate vector $\mathbf{z}(k)$:

$$\mathbf{z}(k) = \frac{\mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{x}(k)}{\sqrt{\lambda}}$$

2. Evaluate the rotations that define $\mathbf{P}(k)$ which annihilates vector $\mathbf{z}(k)$ and compute the scalar variable $t(k)$:

$$\mathbf{P}(k)\begin{bmatrix}\mathbf{z}(k) \\ 1\end{bmatrix} = \begin{bmatrix}\mathbf{0} \\ t(k)\end{bmatrix}$$

3. Update the lower triangular matrix $\mathbf{U}^{-\mathrm{H}}(k)$ and compute vectors $\mathbf{g}(k)$ and $\boldsymbol{\alpha}(k)$:

$$\mathbf{P}(k)\begin{bmatrix}\lambda^{-1/2}\mathbf{U}^{-\mathrm{H}}(k-1) \\ \mathbf{0}^{\mathrm{T}}\end{bmatrix} = \begin{bmatrix}\mathbf{U}^{-\mathrm{H}}(k) \\ \mathbf{g}^{\mathrm{H}}(k)\end{bmatrix}$$

$$\boldsymbol{\alpha}(k) = \mathbf{g}^{\mathrm{H}}(k)\mathbf{C}$$

4. Update the following equations and intermediate inverse matrix:

$$\begin{aligned} \boldsymbol{\Gamma}(k) &= \lambda^{-1}\boldsymbol{\Gamma}(k-1) - \mathbf{g}(k)\boldsymbol{\alpha}(k) \\ \mathbf{q}(k) &= \frac{\sqrt{\lambda}\boldsymbol{\Phi}^{-1}(k-1)\boldsymbol{\alpha}^{\mathrm{H}}(k)}{1-\lambda\boldsymbol{\alpha}(k)\boldsymbol{\Phi}^{-1}(k)\boldsymbol{\alpha}^{\mathrm{H}}(k)} \\ \boldsymbol{\Phi}^{-1}(k) &= \lambda\left[\mathbf{I}+\sqrt{\lambda}\mathbf{q}(k)\boldsymbol{\alpha}(k)\right]\boldsymbol{\Phi}^{-1}(k-1) \end{aligned}$$

5. Update the LS weight vector:

$$\begin{aligned} e(k) &= \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k) \\ \boldsymbol{\rho}(k) &= \mathbf{k}(k) - \frac{\sqrt{\lambda}}{t(k)}\boldsymbol{\Gamma}(k)\mathbf{q}(k) \\ \mathbf{w}(k) &= \mathbf{w}(k-1) - \boldsymbol{\rho}(k)e^{*}(k) \end{aligned}$$

12.4 The Adaptive GSC-IQRD-RLS Algorithm

The overall weight vector with the GSC [26] structure illustrated in Figure 12.3 is equivalent to that of Figure 12.2. With the GSC structure, an alternative indirect approach of the optimal constrained LS weight vector can be developed. First, the original weight vector of Figure 12.2 is decomposed into two parts, i.e.,

$$\mathbf{w}(k) = \mathbf{w}_c - \mathbf{B}\mathbf{w}_a(k)\ . \tag{12.28}$$

In (12.28), the weight vector of the upper path, $\mathbf{w}_c$, is referred to as *quiescent vector*, while the $(N+1)\times(N+1-L)$ matrix $\mathbf{B}$ of the lower path is the rank reduction or blocking matrix. Indeed, $\mathbf{B}$ could be any matrix, whose columns span the left null space of $\mathbf{C}$, e.g., $\mathbf{B}$ is full rank and satisfies

$$\mathbf{B}^{\mathrm{H}}\mathbf{C} = \mathbf{0} \quad \text{and} \quad \mathbf{C}^{\mathrm{H}}\mathbf{B} = \mathbf{0}\ . \tag{12.29}$$

Therefore, the columns of $\mathbf{B}$ form a basis for the null space of $\mathbf{C}^{\mathrm{H}}$ and $(N+1)\times(N+1-L)$ matrix $\mathbf{B}$ can be obtained from $\mathbf{C}$ by any orthogonalization procedures. On the other hand, the upper path vector $\mathbf{w}_c$ simply ensures that the constraint equations are satisfied. The overall system function with the GSC structure depicted in Figure 12.3, ideally, should be equivalent to the direct approach of Figure 12.2; therefore, substituting (12.28) into the definition of constraints, $\mathbf{C}^{\mathrm{H}}\mathbf{w}(k) = \mathbf{f}$, yields

$$\mathbf{C}^{\mathrm{H}}\mathbf{w}(k) = \mathbf{C}^{\mathrm{H}}\left[\mathbf{w}_c - \mathbf{B}\mathbf{w}_a(k)\right] = \mathbf{C}^{\mathrm{H}}\mathbf{w}_c - \mathbf{C}^{\mathrm{H}}\mathbf{B}\mathbf{w}_a(k) = \mathbf{f}\ . \tag{12.30}$$

From (12.30), it can be easily shown that

$$\mathbf{w}_c = \mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{C})^{-1}\mathbf{f} = \mathbf{F}\ . \tag{12.31}$$

Based on the above discussion, we learn that the general mathematical framework for the GSC structure relies on the unconstrained optimization. Clearly the $(N+1-L)\times 1$ weight vector, $\mathbf{w}_a(k)$, is unconstrained and can be freely adapted using any optimization criterion, while the overall weight vector will remain constrained.

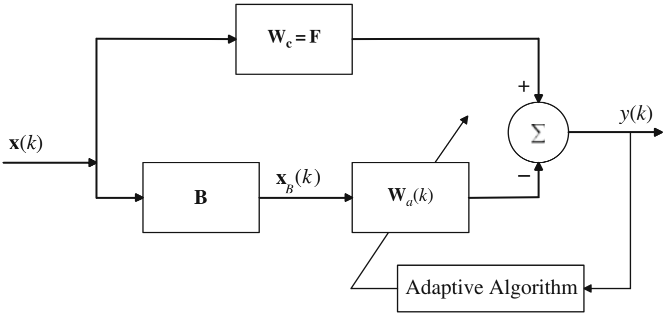


Fig. 12.3 The block diagram of the adaptive linearly constrained filter with the GSC structure.

With the GSC structure, the optimization problem becomes to choose the adaptive weight $\mathbf{w}_a(k)$ from any $\mathbf{w}_a$ in lower branch, to cancel jamming (interfering) signals in upper branch after computing $\mathbf{w}_c$ and $\mathbf{B}$. Under the condition described above, the cost function of (12.1) to be minimized, with the GSC structure, can be rewritten as

$$\begin{aligned} J_{LS}(\mathbf{w}_a) &= \sum_{i=0}^{k} \lambda^{k-i} |[\mathbf{w}_c - \mathbf{B}\mathbf{w}_a]^{\mathrm{H}} \mathbf{x}(i)|^2 \\ &= \sum_{i=0}^{k} \lambda^{k-i} |\mathbf{w}_c^{\mathrm{H}} \mathbf{x}(i) - [\mathbf{B}\mathbf{w}_a]^{\mathrm{H}} \mathbf{x}(i)|^2 \ . \end{aligned} \tag{12.32}$$

If we let $d(i)$ be the desired component of Figure 12.3, e.g., $d(i) = \mathbf{w}_c^{\mathrm{H}} \mathbf{x}(i)$, (12.32) can be represented as

$$\begin{aligned} J_{LS}(\mathbf{w}_a) &= \sum_{i=0}^{k} \lambda^{k-i} |[d(i) - \mathbf{w}_a^{\mathrm{H}} \mathbf{B}^{\mathrm{H}} \mathbf{x}(i)|^2 \\ &= \sum_{i=0}^{k} \lambda^{k-i} |e_B(i/k)|^2 \ = \|\boldsymbol{\Lambda}^{1/2}(k) \mathbf{e}_B(k/k)\|, \end{aligned} \tag{12.33}$$

where the error vector is designated by $\mathbf{e}_B(k/k) = [e_B(0/k), e_B(1/k), ..., e_B(k/k)]^{\mathrm{T}}$. Correspondingly, if we denote $\mathbf{d}(k) = [d(0), d(1), ..., d(k)]^{\mathrm{T}}$ as a desired signal vector, (12.33) becomes

$$J_{LS}(\mathbf{w}) = \|\boldsymbol{\Lambda}^{1/2}[\mathbf{d}^*(k) - \mathbf{X}_B(k)\mathbf{w}_a]\|^2. \tag{12.34}$$

Accordingly, the data matrix based on the structure of GSC is given by $\mathbf{X}_B(k) = [\mathbf{x}_B(0), \mathbf{x}_B(1), \cdots, \mathbf{x}_B(k)]^{\mathrm{H}}$, and $\mathbf{x}_B(i) = \mathbf{B}^{\mathrm{T}} \mathbf{x}(i)$. With the approach of the conventional QRD-RLS algorithm [8, 12, 18], an orthogonal matrix $\mathbf{Q}_B(k)$, is used to carry out the triangular orthogonalization of the data matrix, $\boldsymbol{\Lambda}^{1/2}(k)\mathbf{X}_B(k)$, via *Givens rotations*, that is,

$$\mathbf{Q}_B(k)\boldsymbol{\Lambda}^{1/2}(k)\mathbf{X}_B(k) = \begin{bmatrix} \mathbf{U}_B(k) \\ \mathbf{0} \end{bmatrix}, \tag{12.35}$$

where $\mathbf{U}_B(k)$ is an $(N+1-L) \times (N+1-L)$ upper triangular matrix, and $\mathbf{0}$ is a $(k-N+L) \times (N+1-L)$ null matrix. Similarly, to perform the orthogonal matrix, $\mathbf{Q}_B(k)$, on the weighted desired vector, $\boldsymbol{\Lambda}^{1/2}(k)\mathbf{d}(k)$, it gives

$$\mathbf{Q}_B(k)\boldsymbol{\Lambda}^{1/2}(k)\mathbf{d}^*(k) = \begin{bmatrix} \mathbf{z}_B(k) \\ \mathbf{v}_B(k) \end{bmatrix} , \tag{12.36}$$

where vectors $\mathbf{z}_B(k)$ and $\mathbf{v}_B(k)$ are with the dimensions of $(N+1-L) \times 1$ and $(k-N+L) \times 1$, respectively. Since orthogonal matrices are length preserving, using the results of (12.12) and (12.13), the cost function defined in (12.11) can be expressed as

$$J_{LS}(\mathbf{w}_a) = \left\| \begin{bmatrix} \mathbf{z}_B(k) - \mathbf{U}_B(k)\mathbf{w}_a \\ \mathbf{v}_B(k) \end{bmatrix} \right\|^2 . \tag{12.37}$$

It is straightforward to see that the norm in Equation (12.37) is minimized if the top partition of (12.37) is set to null. From (12.37), the optimum LS weight vector $\mathbf{w}_a(k)$ based on the QR decomposition can be obtained

$$\mathbf{U}_B(k)\mathbf{w}_a(k) = \mathbf{z}_B(k). \tag{12.38}$$

Similarly, the LS weight vector $\mathbf{w}_a(k)$ based on the IQRD can be represented by

$$\mathbf{w}_a(k) = \mathbf{U}_B^{-1}(k)\mathbf{z}_B(k). \tag{12.39}$$

As a consequence, the IQRD-RLS algorithm for updating the LS weight vector, $\mathbf{w}_a(k)$, is given by [18]

$$\mathbf{w}_a(k) = \mathbf{w}_a(k-1) + \frac{\mathbf{g}_B(k)}{t_B(k)} e_B^*(k), \tag{12.40}$$

with the *a priori* estimation error, $e_B(k)$ being defined by

$$e_B(k) = d(k) - \mathbf{w}_a^{\mathrm{H}}(k-1)\mathbf{x}_B(k). \tag{12.41}$$

The scalar variable $t_B(k)$ and the $(N+1)\times 1$ intermediate vector $\mathbf{g}_B(k)$ are defined as

$$t_B(k) = \sqrt{1 + \mathbf{z}_B^{\mathrm{H}}(k)\mathbf{z}_B(k)}, \tag{12.42}$$

and

$$\mathbf{g}_B(k) = \frac{\mathbf{U}_B^{-1}(k-1)\mathbf{z}_B(k)}{\sqrt{\lambda}\, t_B(k)}, \tag{12.43}$$

respectively, and the intermediate vector $\mathbf{z}_B(k)$, which provides the key to the parallelization of the IQRD-RLS approach, is designated by

$$\mathbf{z}_B(k) = \frac{\mathbf{U}_B^{-\mathrm{H}}(k-1)\mathbf{x}_B(k)}{\sqrt{\lambda}}. \tag{12.44}$$

It should be noted that both $\mathbf{g}_B(k)$ and $t_B(k)$ are evaluated entirely by rotation-based method, using *Givens rotations*, when $\mathbf{U}_B^{-1}(k)$ is updated from $\mathbf{U}_B^{-1}(k-1)$. From [18], it is known that there exists a rotation matrix $\mathbf{P}'(k)$ such that

$$\mathbf{P}'(k)\begin{bmatrix}\mathbf{z}_B(k)\\ 1\end{bmatrix} = \begin{bmatrix}\mathbf{o}\\ t_B(k)\end{bmatrix}, \tag{12.45}$$

where $\mathbf{P}'(k)$ successively annihilates the elements of the vector $\mathbf{z}_B(k)$, starting from the top, by rotating them into the element at the bottom of the augmented vector $[\mathbf{z}_B^{\mathrm{T}}(k), 1]^{\mathrm{T}}$. In consequence, we can evaluate $\mathbf{g}_B(k)$ as follows:

$$\mathbf{P}'(k)\begin{bmatrix}\lambda^{-1/2}\mathbf{U}_B^{-\mathrm{H}}(k-1)\\ \mathbf{0}^{\mathrm{T}}\end{bmatrix} = \begin{bmatrix}\mathbf{U}_B^{-\mathrm{H}}(k)\\ \mathbf{g}_B^{\mathrm{H}}(k)\end{bmatrix}. \tag{12.46}$$

It is of interest to point out that the scalar variable $t_B(k)$ and the intermediate vector $\mathbf{g}_B(k)$ are evaluated based on (12.45) and (12.46). Also, $\mathbf{g}_B(k)$ scaled by $t_B(k)$ can be viewed as the adaptation gain of the IQRD-RLS algorithm, which is defined as

$$\mathbf{k}_B(k) = \frac{\mathbf{g}_B(k)}{t_B(k)}. \tag{12.47}$$

Equation (12.40) tells us that the LS weight vector is updated by incrementing its old value by an amount equal to the *a priori* estimation error, $e_B(k/k-1)$ times the time-varying gain vector, $\mathbf{k}_B(k)$. Moreover, it can be shown that $\mathbf{k}_B(k) = \mathbf{S}_B(k)\mathbf{x}_B(k)$ with $\mathbf{S}_B(k) = \mathbf{U}_B^{-1}(k)\mathbf{U}_B^{-\mathrm{H}}(k)$. This completes the derivation of the GSC-IQRD-RLS algorithm which is summarized in Table 12.2. We note that, for this case where the

Table 12.2 Summary of the adaptive GSC-based IQRD-RLS algorithm.

GSC-IQRD-RLS

- Initialization δ=small positive constant:

$$\begin{aligned} \mathbf{U}_B^{-1}(0) &= \delta^{-1}\mathbf{I} \\ \mathbf{w}_c &= [\mathbf{C}(\mathbf{C}^{\mathrm{H}}\mathbf{C})^{-1}]\mathbf{f} \\ \mathbf{w}_a(0) &= \mathbf{0} \end{aligned}$$

- For k = 1, 2, ... , do

1. Compute the intermediate desired signal $d(k)$ and input vector $\mathbf{x}_B(k)$:

$$\begin{aligned} d(k) &= \mathbf{w}_c^{\mathrm{H}}\mathbf{x}(k) \\ \mathbf{x}_B(k) &= \mathbf{B}^{\mathrm{H}}\mathbf{x}(k) \end{aligned}$$

2. Compute the intermediate vector $\mathbf{z}_B(k)$:

$$\mathbf{z}_B(k) = \frac{\mathbf{U}_B^{-\mathrm{H}}(k-1)\mathbf{x}_B(k)}{\sqrt{\lambda}}$$

3. Evaluate the rotations that define $\mathbf{P}'(k)$ which annihilates vector $\mathbf{z}_B(k)$ and compute the scalar variable $t_B(k)$:

$$\mathbf{P}'(k)\begin{bmatrix} \mathbf{z}_B(k) \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ t_B(k) \end{bmatrix}$$

4. Update the lower triangular matrix $\mathbf{U}_B^{-\mathrm{H}}(k)$ and compute vector $\mathbf{g}_B(k)$:

$$\mathbf{P}'(k)\begin{bmatrix} \lambda^{-1/2}\mathbf{U}_B^{-\mathrm{H}}(k-1) \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_B^{-\mathrm{H}}(k) \\ \mathbf{g}_B^{\mathrm{H}}(k) \end{bmatrix}$$

5. Updating the LS weight vector:

$$\begin{aligned} e_B(k/k-1) &= d(k) - \mathbf{w}_a^{\mathrm{H}}(k-1)\mathbf{x}_B(k) \\ \mathbf{w}_a(k) &= \mathbf{w}_a(k-1) + \frac{\mathbf{g}_B(k)}{t_B(k)}e_B^*(k/k-1) \end{aligned}$$

adaptive part within the GSC structure is unconstrained, it is immediate to find a direct correspondence between the variables used here and those of the IQRD-RLS algorithm presented in Chapter 3.

12.5 Applications

Before proceeding further to conclude the advantages of the adaptive LC-IQRD-RLS and GSC-IQRD-RLS algorithms, it is instructive to develop an appreciation for the versatility of these important algorithms by applying them to LCMV filtering problems.

12.5.1 Application 1: Adaptive LCMV filtering for spectrum estimation

In the first application, we consider the spectral analysis which is very significant in many signal processing applications, such as speech signal processing and interference suppression. In this application, we would like to investigate the nulling capability of the LC-IQRD-RLS algorithm and compare it to the general constrained fast LS algorithm. As described in [2], with the general fast LS algorithm (see [2], Table 1), desired performance may not be satisfied due to round-off error during the adaptation processes. Common to adaptive filter parameter updating algorithm, a correction term, proportional to the quantity $\mathbf{f} - Cb\ f^{\mathrm{H}}\mathbf{w}(k)$, is an intuitively reasonable form to obtain a more robust modified version, referred to as the robust constrained FLS (RCFLS) algorithm. To do so, the frequency response of the LCMV filtering, for eliminating the undesired frequencies, is examined. We assume that input signal consists of three sinusoids buried in additive white noise, i.e.,

$$\mathbf{x}(k) = 10\sin(0.3k\pi) + 100\sin(0.66k\pi) + \sin(0.7k\pi) + b(k). \tag{12.48}$$

The corresponding normalized frequencies and amplitudes are set to be 0.15, 0.33, and 0.35, and 10, 100, and 1, respectively, and $b(k)$ denotes the additive white noise with zero-mean and variance, σ_b^2 such that the signal-to-noise ratio (SNR) is 40 dB.

Moreover, the filter is constrained to have unit response at two frequencies, viz., 0.1 and 0.25 (normalized digital frequencies). Each of the two unit response frequencies generates two-point constraints. In such case, $L = 4$ and $(N+1) = 11$ (weight coefficients), the constraint parameters are

$$\mathbf{C}^{\mathrm{T}} = \begin{bmatrix} 1 \ \cos(0.2\pi) & \cdots & \cos((N)0.2\pi) \\ 1 \ \cos(0.5\pi) & \cdots & \cos((N)0.5\pi) \\ 0 \ \sin(0.2\pi) & \cdots & \sin((N)0.2\pi) \\ 0 \ \sin(0.5\pi) & \cdots & \sin((N)0.5\pi) \end{bmatrix}_{4\times 11} \quad \text{and } \mathbf{f}^{\mathrm{T}} = \begin{bmatrix} 1 \ 1 \ 0 \ 0 \end{bmatrix}. \tag{12.49}$$

We know that, usually, it is more difficult to separate two signals closely in adjacent frequency band, especially when one has relatively larger power than the other; the signal with smaller power may be ignored and yields the wrong result.

From Figure 12.4, we observe that the learning curve of the RCFLS algorithm might be disturbed by the inversion of the correlation matrix (while computing the adaptation gain) which is inherently numerically unstable. The nulling capability for the undesired signal, the one with normalized frequency 0.35 and less power, is affected by the adjacent signal with frequency 0.33 and having relatively larger power. However, the LC-IQRD-RLS algorithm has faster convergence rate and better numerical stability than the RCFLS algorithm. For comparison, the nulling gains, in dB, for different frequency components using the LC-IQRD-RLS algorithm and the RCFLS algorithm, are listed in Table 12.3.

Next, let us consider the problem of constrained drift which may be defined as the squared norm of $\mathbf{C}^T\mathbf{w}(k) - \mathbf{f}$. Also, using the notation of functions found in Matlab®, the tolerance (Tol) of the numerical accuracy may be expressed as

$$Tol = max(size(A)) \times norm(A) \times eps,$$

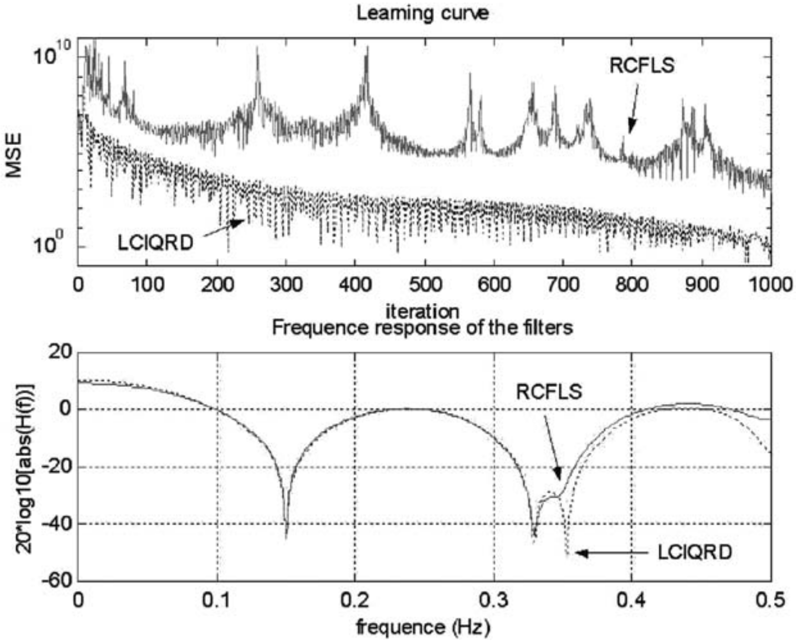


Fig. 12.4 Learning curve and frequency response of two algorithms after 1000 iterations with 100 independent runs.

Table 12.3 Comparison of nulling capability of RCFLS and LC-IQRD algorithms.

Iteration	Algorithms	Normalized frequency of the signals		
		0.15	0.33	0.35
1000	RCFLS	−44.05(dB)	−43.50(dB)	−28.35(dB)
1000	LCIQRD	−44.12(dB)	−44.04(dB)	−49.55(dB)
10000	RCFLS	−44.10(dB)	−45.63(dB)	−47.32(dB)
10000	LCIQRD	−44.99(dB)	−46.07(dB)	−54.46(dB)

where eps is the floating point relative accuracy and A is denoted as the correlation matrix. The smaller value of eps implies that a larger word-length is required to achieve a specific Tol; for instance, if we set $eps = 2^{-10}$, approximately, the word-length will be 10 bits. From the implementation point of view, eps can also be treated as the number of multiplication operator in the specific DSP device. Alternatively, Tol may be used to set a limit or the precision for our simulation environment. That is, for any singular value less than Tol will be treated as null or deriving a rounding error, during the computation procedure. In our simulation, the value of Tol is chosen to be 0.9453 and the corresponding value of eps to achieve the numerical accuracy of Tol is $eps = 2^{-7}$. In Figure 12.5, the results of the RCFLS algorithm, in terms of MSE, output power and the constrained drift, with the parameters described above are given. The jitters phenomenon of the RCFLS algorithm are found in

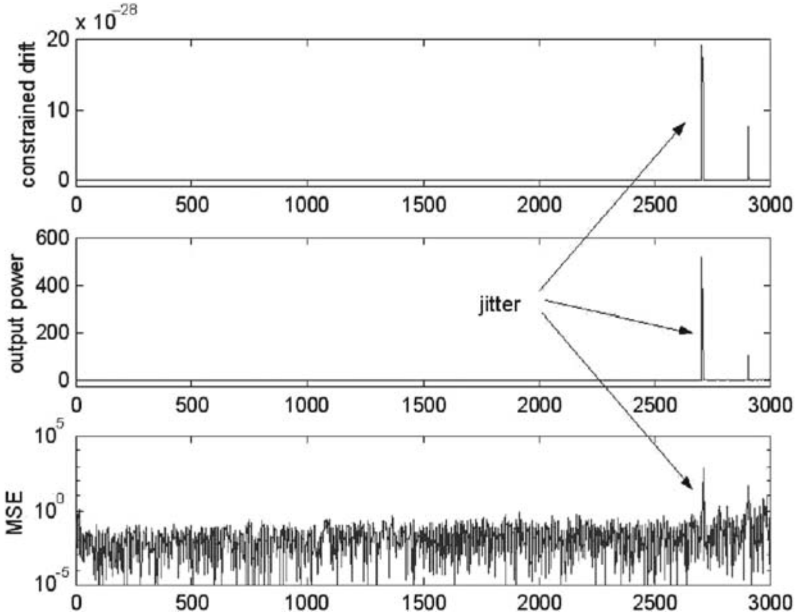


Fig. 12.5 Numerical properties of the RCFLS algorithm with $eps = 2^{-7}$.

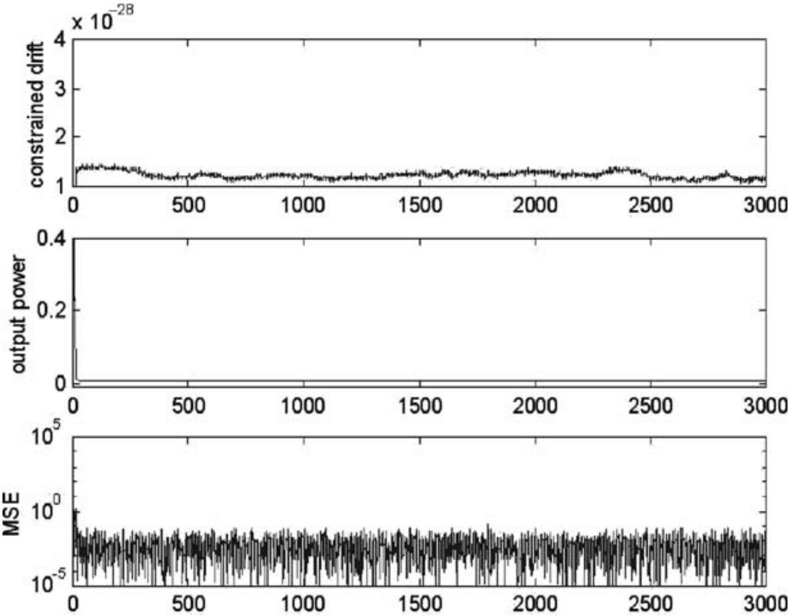


Fig. 12.6 Numerical properties of the LC-IQRD (LC-IQRD-RLS) algorithm with $eps = 2^{-7}$.

the learning curve (MSE), output power and the constrained drift, as depicted in Figure 12.5. However, as shown in Figure 12.6 with the same parameter as in Figure 12.5, the results of the LC-IQRD-RLS algorithm, in terms of MSE, output power and constrained drift, are much better than those of the RCFLS algorithm.

12.5.2 Application 2: Adaptive LCMV antenna array beamformer

There are two types of antenna array beamformers, viz., broadband array structure and narrowband array structure. In this application, the narrowband array beamformer structure is considered for interferences (or undesired signals) suppression. Basically, an array beamformer is a processor used in conjunction with an array of sensors to provide a versatile form of spatial filtering. Since the desired signal and the interference (or jammer) usually originate from different spatial locations, we are able to remove the jammer from the received signal by exploiting the spatial diversity at the receiver. The LCMV beamformer is known to be one of the most popular approaches for suppressing the undesired interference [16, 20, 23]. However, by using the adaptive array beamforming approach, the array system can automatically adjust its directional response to null the jammers, and thus enhances the reception of the desired signal.

The basic operation of the adaptive antenna array is usually described in terms of a receiving system steering a null, that is, a reduction in sensitivity in a certain

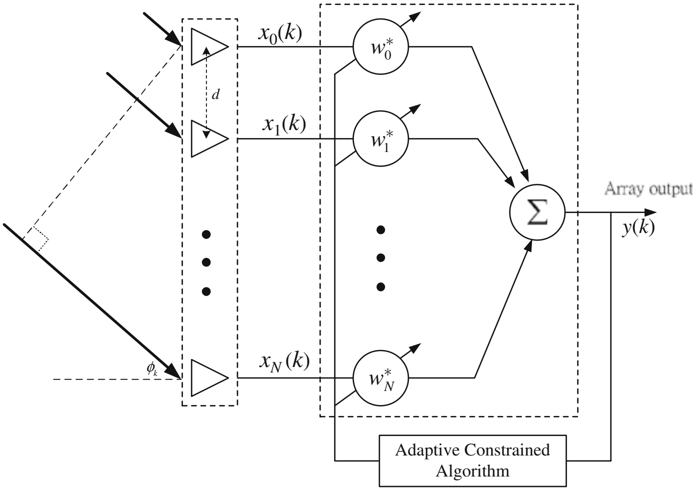


Fig. 12.7 Configuration of linearly constrained uniform spaced narrowband array beamformer.

position, towards a source of interference. It consists of a number of antenna elements coupled together via some form of amplitude control and phase shifting network to form a single output. The amplitude and phase control can be regarded as a set of complex weights, as illustrated in Figure 12.7. To start our derivation, let us consider a uniform linear array (ULA) and a wavefront, generated by a desired source of wavelength λ, propagating in an $N+1$ element array of sensors from a direction ϕ_k off the array boresight. Now, taking the first element in the array as the phase reference and with equal array spacing, d, the relative phase shift of the received signal at the nth element can be expressed as $\phi_{nk} = \frac{2\pi}{\lambda} d(k-1)\sin\phi_k$. Moreover, assuming that the spacing between the array elements is set to $\lambda/2$, the array response vector of this $(N+1)$-antenna ULA can be denoted by

$$\mathbf{a}(\phi_k) = \left[1,\, e^{-j\pi\sin(\phi_k)},\, \cdots,\, e^{-j(N)\pi\sin(\phi_k)}\right]^{\mathrm{T}}. \tag{12.50}$$

Thus, we choose ϕ_k toward the direction of arrival (DOA) of the desired source signal and suitably adjust the weights of adaptive array; the array will pass the desired source signal from direction ϕ_0 and steer nulls toward interference sources located at ϕ_k for $k \neq 0$. It can be shown that an $(N+1)$ element array has $(N+1)\times 1$ degrees of freedom giving up to $(N+1)\times 1$ independent pattern nulls. So it has better performance if the array has more antenna elements. We assume that the received signal in each sensor consists of a desired source signal buried in white Gaussian noise and three directional interferences (or jammers) incident at angles ϕ_1, ϕ_2, and ϕ_3, respectively. For convenience, the look direction of the desired source signal is chosen to be $\phi_1 = 0°$. In the constrained approach of beamforming algorithm, the use of adaptive array for suppressing interference, with a look-direction constraint, is highly dependent on the accuracy of the *a priori* information (DOA) to achieve the maximum signal-to-interference-plus-noise ratio (SINR). However, an error in the steering angle, termed *pointing error*, would cause the adaptive array to tend to null out the desired signal as if it were a jammer. To verify the observation described earlier in this application, the problem of adaptive beamformer with main-beam constrained, associated with the pointing error, is considered. The deviation angle $\Delta\phi$ is defined as the discrepancy that the constraints look-direction, ϕ_c, deviates from the true arrival direction of the desired signal ϕ_1, i.e., $\Delta\phi = \phi_1 - \phi_c$. We note that, if pointing error exists (the look direction and the main-beam constraint is deviated due to estimation error of arrival angle), one of the conventional approaches would be the derivatives constraint approach [24, 25]. In such a case we may adapt the derivative constraints (DC) into the beamformer where constraint matrix **C**, with L linear constraints, is given by

$$\mathbf{C} = \left[\, \mathbf{a}(\phi_c)\ \mathbf{a}^{(1)}(\phi_c)\ \cdots\ \mathbf{a}^{L-1}(\phi_c) \,\right], \tag{12.51}$$

where $\mathbf{a}^i(\phi_c) = \frac{\partial a(\phi)}{\partial \phi^i}|_{\phi=\phi_c}$ is defined as the ith derivative of the steering vector with respect to ϕ.

Moreover, there are two design approaches to obtain the response vector **f**. The first approach is to use the conventional beamformer response, i.e.,

$$\mathbf{f} = \frac{1}{M}[\mathbf{a}^{\mathrm{H}}(\phi_c)\mathbf{a}(\phi_c), \mathbf{a}^{\mathrm{H}}(\phi_c)\mathbf{a}^{(1)}(\phi_c), \cdots, \mathbf{a}^{\mathrm{H}}(\phi_c)\mathbf{a}^{(L-1)}(\phi_c)]^{\mathrm{T}} = \frac{1}{M}\left[\mathbf{a}^{\mathrm{H}}(\phi_c)\mathbf{C}\right]^{\mathrm{T}}. \tag{12.52}$$

This will make the beamformer force the lobe shape of the main beam. In the second scheme, we set the other derivatives to a zero response, i.e., $\mathbf{f} = [1, 0, \cdots, 0]^{\mathrm{T}}$, and this could make the beamformer achieve the main beam with a flat top shape. Since we can expect that the jammer power is, in general, much larger than the desired signal source, the SNR is set to 0 dB. In our simulations, we have used three jammers with different jammer power ratios (JPR), e.g., $JNR_1 = 10$ dB, $JNR_2 = 20$ dB, and $JNR_3 = 40$ dB, corresponding to incident angles $-30°$, $35°$, and $40°$. First, we consider the case of main-beam constraint (single constraint) only, and assumed that there is no pointing error, e.g., $\Delta\phi = \phi_1 - \phi_c = 0°$. The results, in terms of nulling capability, are given in Figure 12.8 for both LC-IQRD-RLS and GSC-IQRD-RLS algorithms, with the forgetting factor $\lambda = 0.98$. These results were evaluated after 200 snapshots and correspond to an average of 500 independent runs. From Figure 12.8, we observe that both algorithms have identical beampatterns. We have also compared the results obtained with constrained LS algorithms with those of the linearly constrained LMS (Frost's algorithm) [5] and its GSC counterpart [26]. As can be seen in Figure 12.8, the LC-IQRD-RLS algorithm outperforms, in terms of nulling capability shown in the beampatterns, the LC-LMS and the GSC-LMS algorithms. Moreover, as described in the literature for the case where $\mathbf{B}^{\mathrm{H}}\mathbf{B} = \mathbf{I}$, the LC-LMS algorithm is identical to the GSC-LMS algorithm [21]. It is worth noting that, for the LC-IQRD-RLS algorithm, the condition $\mathbf{B}^{\mathrm{H}}\mathbf{B} = \mathbf{I}$ is not required for achieving the equivalency with the GSC-IQRD-RLS algorithm.

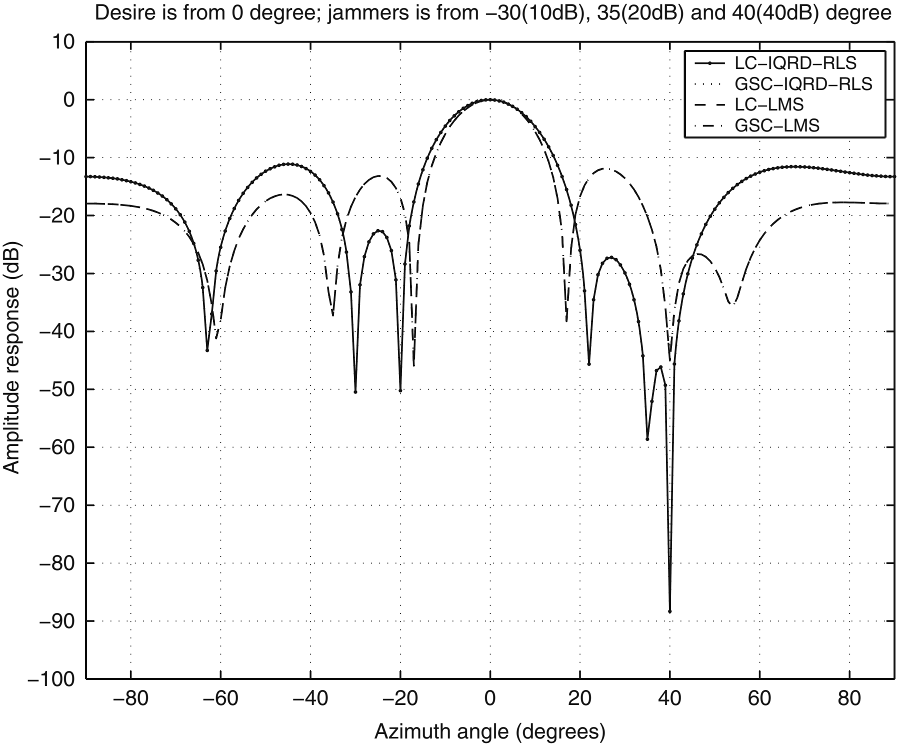


Fig. 12.8 Comparison of the GSC-IQRD-RLS algorithm with other algorithms, $\lambda = 0.98$ (without pointing error).

Next, we would like to see the effect due to pointing error, and to verify that the equivalency is also true for multiple constraint case. To do so, we consider the case that a pointing error, $\Delta\phi = \phi_1 - \phi_c = 3°$, occurs (also known as DOA mismatch). Under this circumstance, with and without using the derivative constraint for the main-beam, the performance of the same algorithms are investigated. We let the other parameters be the same as in the case without having pointing error. From Figure 12.9, we learn that, with the single constraint (without using the derivative constraint), the gain of main beam is attenuated due to mismatch of the true look direction and the main-beam constraint for the LC-IQRD-RLS and the GSC-IQRD-RLS algorithms. Although the use of the LC-LMS algorithm has less effect due to pointing error, nulling capability is still worse than the one with the LC-IQRD-RLS algorithm. But, with the use of derivative constraints for the main-beam constraint (multiple constraints, e.g., the main beam and its first order constraints), the effect due to the pointing error has been alleviated while keeping better nulling capability than the LC-LMS algorithm. In this case, the performance of the LC-IQRD-RLS algorithm is again identical to the GSC-IQRD-RLS algorithm. Usually, we could use higher order derivative constraint to achieve better result of combating DOA

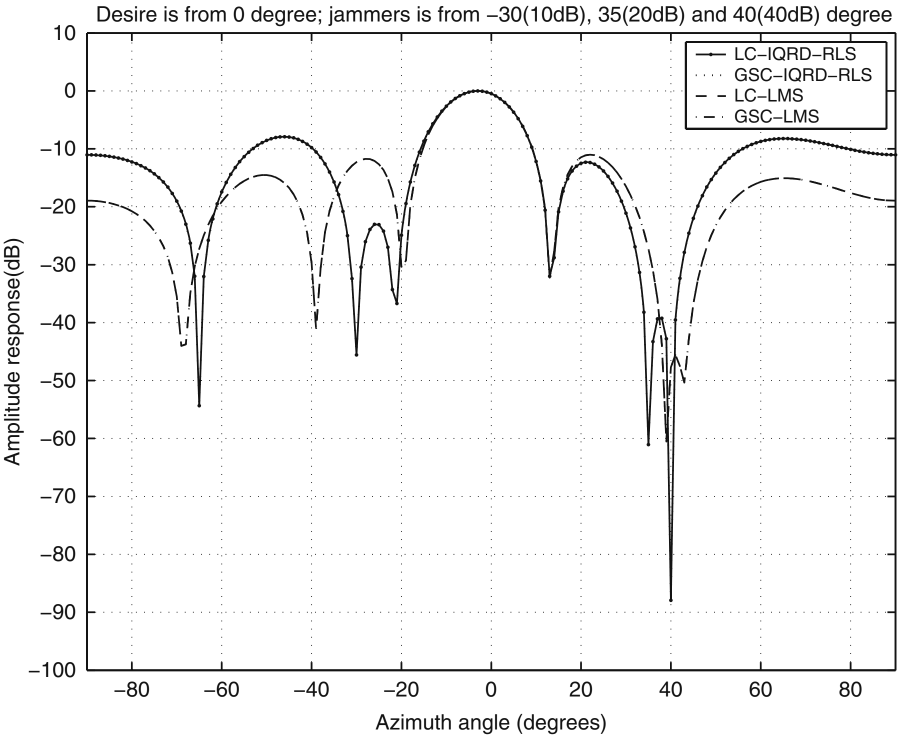


Fig. 12.9 Comparison of the GSC-IQRD-RLS algorithm with other algorithms, $\lambda = 0.98$ (pointing error with derivative constraint).

mismatch. For comparison, the results of nulling capability for Figures 12.8 and 12.9 are listed in Tables 12.4 and 12.5.

From Tables 12.4 and 12.5, we observe that the use of the first order main beam derivative constraint with the LC-LMS algorithm did not gain any benefit. Conversely, a significant gain improvement has been verified for the LC-IQRD-RLS algorithm.

From the results of the experiment carried out with this beamformer, we may conclude that, for the multiple constraint case, both the LC-IQRD-RLS and the GSC-IQRD-RLS algorithms did have the same performance. Although having a similar performance, the use of the GSC-IQRD-RLS algorithm has the advantage of requiring less computational complexity than the direct LC-IQRD-RLS algorithm.

Table 12.4 Comparison of nulling capability for various linearly constrained beamforming algorithms with single constraint (without pointing error).

	Desired signal	jammer 1	jammer 2	jammer 3
SNR	0 (dB)	10 (dB)	20 (dB)	40 (dB)
Azimuth algorithm	0^o	-30^o	35^o	40^o
LC-LMS algorithm	-0.445(dB)	-13.38(dB)	-27.44(dB)	-49.33(dB)
GSC-LMS algorithm	-0.445(dB)	-13.38(dB)	-27.44(dB)	-49.33(dB)
LC-IQRD-RLS algorithm	-4.169(dB)	-49.92(dB)	-58.08(dB)	-87.72(dB)
GSC-IQRD-RLS algorithm	-4.169(dB)	-49.92(dB)	-58.08(dB)	-87.72(dB)

Table 12.5 Comparison of nulling capability for various linearly constrained beamforming algorithms with multiples constraint (pointing error with derivative constraint).

	Desired signal	jammer 1	jammer 2	jammer 3
SNR	0 (dB)	10 (dB)	20 (dB)	40 (dB)
Azimuth algorithm	0^o	-30^o	35^o	40^o
LC-LMS algorithm	-0.4745(dB)	-12.3(dB)	-27.26(dB)	-47.7(dB)
GSC-LMS algorithm	-0.4745(dB)	-12.3(dB)	-27.26(dB)	-47.7(dB)
LC-IQRD-RLS algorithm	-0.4773(dB)	-45.58(dB)	-61(dB)	-87.95(dB)
GSC-IQRD-RLS algorithm	-0.4773(dB)	-45.58(dB)	-61(dB)	-87.95(dB)

12.6 Conclusion

In this chapter, we developed the direct and indirect IQRD-RLS-based constrained adaptive filtering algorithms, named the LC-IQRD-RLS and the GSC-IQRD-RLS algorithms, respectively, to implement the LCMV filter. The IQRD approach was chosen such that the constrained LS weight vector solution could be updated without using back-substitution, which is suitable to be implemented using a typical VLSI technology structure termed systolic array. To verify the merits of the LC-IQRD-RLS and the GSC-IQRD-RLS algorithms, we applied them to spectral analysis and smart antenna array beamforming problems. We have shown that, due to the numerical stability of evaluating the adaptation (or Kalman) gain via *Givens rotations*, the proposed LC-IQRD-RLS algorithm had less effect of constraint drift compared with the CFLS algorithm and its robust version [2]. Thus, we concluded that the LC-IQRD-RLS algorithm proposed in this chapter did perform better than the CFLS and RCFLS algorithms developed in [2], in terms of capability to null the undesired signal components output power as well as numerical efficiency in practical implementation.

References

1. S. J. Chern and C. Y. Chang, Adaptive linearly constrained inverse QRD-RLS beamformer for moving jammers suppression. IEEE Transactions on Antennas and Propagation, vol. 50, no. 8, pp. 1138–1150 (August 2002)

2. L. S. Resende, J. T. Romano, and M. G. Bellanger, A fast least-squares algorithm for linearly constrained adaptive filtering. IEEE Transactions on Signal Processing, vol. 44, no. 5, pp. 1168–1174 (May 1996)
3. D. H. Johnson and Dan E. Dudgeon, Array Signal Processing Concepts and Techniques. Prentice-Hall, Englewood Cliffs, NJ, USA (1993)
4. S. N. Lin and S. J. Chern, A new adaptive constrained LMS time delay estimation algorithm. Signal Processing (Elsevier), vol. 71, pp. 29–44 (November 1998)
5. O. L. Frost III, An algorithm for linearly constraint adaptive array processing. Proceedings of IEEE, vol. 60, no. 8, pp. 926–935 (August 1972)
6. S. J. Chern and C. Y. Chang, Adaptive MC-CDMA receiver with constrained constant modulus IQRD-RLS algorithm for MAI suppression. Signal Processing (Elsevier), vol. 83, no. 10, pp. 2209–2226 (October 2003)
7. J. B. Schodorf and D. W. Williams, Array processing techniques for multiuser detection. IEEE Transactions on Communications, vol. 45, no. 11, pp. 1375–1378 (November 1997)
8. S. Haykin, Adaptive Filter Theory. 3rd edition Prentice-Hall, Inc. Englewood Cliffs, NJ, USA (1996)
9. J. M. Cioffi and T. Kailath, Fast RLS transversal filters for adaptive filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, pp. 304–337 (June 1984)
10. J. M. Cioffi, Limited precision effects for adaptive filtering. IEEE Transactions on Circuits and Systems, vol. 34, pp. 821–833 (July 1987)
11. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, pp. 879–891 (April 1991)
12. G. H. Golub and C. F. Van Load, Matrix Computation. 3rd edition John Hopkins University Press, Baltimore, MD, USA (1996)
13. Z. S. Liu, QR Method of O(N) complexity in adaptive parameter estimation. IEEE Transactions on Signal Processing, vol. 43, no. 3, pp. 720–729 (March 1995)
14. J. M. Cioffi, High speed systolic implementation of fast QR adaptive filters. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'88, New York, pp. 1584–1587 (April 1988)
15. H. Leung and S. Haykin, Stability of recursive QRD-RLS algorithm using finite precision systolic array implementation. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, pp. 760–763 (May 1989)
16. M. Moonen, Systolic MVDR Beamforming with inverse updating. Proceedings IEE Pt F, vol. 140, no. 3, pp. 175–178 (March 1993)
17. C.-F. T. Tang, *Adaptive Array Systems Using QR-Based RLS and CRLS Techniques with Systolic Array Architectures*. Ph.D. thesis - Department of Electrical Engineering, University of Maryland, College Park, MD, USA (1991)
18. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)
19. D. T. M. Slock and T. Kailath, Numerical stable fast transversal filters for recursive least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 1, pp. 92–114 (January 1991)
20. M. Moonen and I. K. Proudler, MVDR beamforming and generalized sidelobe cancellation based on inverse updating with residual extraction. IEEE Transactions on Circuit and System II, vol. 47, no. 4, pp. 352–358 (April 2000)
21. B. R. Breed and J. Strauss, A short proof of the equivalence of LCMV and GSC beamforming. IEEE Signal Processing Letters, vol. 9, no. 6, pp. 168–169 (June 2002)
22. J. A. Apolinário Jr. and M. L. R. de Campos, The constrained conjugate gradient algorithm. IEEE Signal Processing Letters, vol. 7, no. 12, pp. 351–354 (December 2000)
23. R. A. Games, W. L. Estman, and M. J. Sousa, Fast algorithm and architecture for constrained adaptive side-lobe cancellation. IEEE Transactions on Antennas and Propagation, vol. 41, no. 5, pp. 683–686 (May 1993)

24. C. Y. Chang and S. J. Chern, Derivative constraint narrowband array beamformer with new IQML algorithm for wideband and coherent jammers suppression. IEICE Transactions on Communications, vol. E86-B, no. 2, pp. 829–837 (February 2003)
25. W. G. Najm, Constrained least squares in adaptive, imperfect arrays. IEEE Transactions on Antennas and Propagation, vol. 38, no. 11, pp. 1874–1878 (November 1990)
26. L. J. Griffiths and C. W. Jim, An alternative approach to linearly constrained adaptive beamforming. IEEE Transactions on Antennas and Propagation, vol. AP-30, no. 1, pp. 27–34 (January 1982)

Index