UNIVERSITÀ DEGLI STUDI DI TRIESTE

DOTTORATO DI RICERCA

IN

INGEGNERIA DELL'INFORMAZIONE

X CICLO

Tesi di Dottorato di Ricerca

in

Elaborazione di Segnali ed Immagini

ADAPTIVE AND NONLINEAR SIGNAL PROCESSING

Dottorando:
Alberto CARINI

Relatore:
Chiar.mo Prof. Ing. Giovanni L. SICURANZA

Coordinatore:
Chiar.mo Prof. Ing. Giuseppe O. LONGO

To Prof. Giovanni L. Sicuranza, who has followed me from the early years of the Laurea thesis, teaching me the love for knowledge and research.

To Prof. V. John Mathews, who has taught me to do research rigorously and methodically.

To Prof. Enzo Mumolo, whose friendship, help, availability and experience never failed.

To my parents and my sister who have supported and stimulated my passion for research.

Acknowledgements

The Ph.D. candidate wish to thank the University of Utah and Prof. V. John Mathews for having hosted him for several months in Salt Lake City, U.S.A.; Prof. Giovanni L. Sicuranza, Prof. Enzo Mumolo and Prof. V. John Mathews for their help in reviewing this dissertation; the Dipartimento di Elettronica Elettrotecnica e Informatica and the Image Processing Laboratory of the University of Trieste for their support during the Ph.D. thesis; the reviewers of the various papers in which the subjects of this dissertation were first presented; the many researchers who helped the candidate with interesting discussions.

Contents

1 Introduction **1**

Appendices 6

1.A Discrete Volterra Filters 6

2 Fast Square-Root RLS Adaptive Filtering Algorithms **10**

2.1 Introduction 10

2.2 Review of RLS Adaptive Filtering 12

2.3 Preliminary Results 14

2.3.1 Some Useful Relations 14

2.3.2 Factorizations of the Autocorrelation Matrix . . . 16

2.3.3 Review of Givens Rotations 24

2.4 Fast Square-Root RLS Algorithms 28

2.4.1 Fast RLS Based upon Square-Root Factorization 28

2.4.2 Fast RLS Based upon UD-Factorization 34

2.4.3 Efficient Factorization Algorithms 38

2.5 Computer Simulations 41

2.6 Final Remarks and Conclusions 47

3 V-vector Algebra and its Application to Volterra Adaptive Filtering **48**

3.1 Introduction 48

3.2 The V-vector Algebra 50

3.3 V-vectors for Volterra and Linear Multichannel Filters . 61

3.4 A Reformulation of the Lee-Mathews' Adaptive Fast RLS Algorithm 64

3.5 A Volterra Givens Rotation Based Fast SQR RLS Filter 68

3.6 Conclusions 79

Appendices 80

3.A The Fundamental Operations between V-Matrices 80

4 Sufficient Stability Bounds for Slowly-Varying Direct-Form Recursive Linear Filters and Their Applications in Adaptive IIR Filters **81**
4.1 Introduction 81
4.2 Sufficient Conditions for the Stability of Slowly-Varying Direct-Form Recursive Systems 83
4.2.1 The Lyapunov Candidate Sequence 85
4.3 Stabilized Output Error Adaptive IIR Filters 88
4.4 Experimental Results 89
4.5 Concluding Remarks 95

5 Sufficient Stability Conditions for Discrete-Time Recursive Polynomial Filters **96**
5.1 Introduction 96
5.2 Sufficient Stability Conditions for Recursive Quadratic Filters 98
5.3 Sufficient Stability Conditions for General Order Recursive Volterra Filters 102
5.4 Final Remarks and Conclusion 108

6 The Inverse of Certain Nonlinear Systems **109**
6.1 Introduction 109
6.2 The Inverse of Certain Nonlinear Systems 110
6.3 pth Order Inverses 112
6.4 An Experimental Result 115
6.5 Concluding Remarks 117

7 Equalization and Linearization of Nonlinear Systems **119**
7.1 Introduction 119
7.2 Ideal equalization and linearization 122
7.2.1 The ideal pre- and post-linearizers 126
7.3 pth Order Equalizers and Linearizers 128
7.4 Equalization and Linearization of Recursive Polynomial Systems 136
7.5 An application in Linearization of Loudspeakers 139
7.6 Concluding Remarks 143

8 Conclusions **145**
8.1 Summary 145
8.2 Suggestions for Future Research 146

Bibliography **148**

Chapter 1

Introduction

The development of Digital Signal Processing during the last thirty years is comparable only with the development of high-speed computers, microelectronics and fabrication technologies for integrated circuits. Indeed, the rapid development of Digital Signal Processing has parallelled in time the development of all those electronic technologies that have made physically realizable the algorithms of signal treatment. At the same time, the necessity for implementing more and more complex and powerful signal processing algorithms has been one of the most important *stimuli* for the development of those technologies.

The concept of signal is extremely general. A signal is any physical quantity that is a function of one or more independent variables such as time, distance, temperature or pressure. A signal is the voice, the music, a single image or a video, is the pression, temperature or the cardiac rhythm of a body, are the electrical impulses of brain, is the level of a tide or the flux of a ocean current, is a seismic waveform or the spectrum of a star, is the oscillation or vibration of a ship, a car, an airplane or any other engine, is the electromagnetic waveform by which data are transferred both on a wired or wireless transmission system, is the magnetic or optical or mechanical pattern by which data are storaged in any physical medium, is the N-tupla of electric signals that comes from an array of antennas or microphones, is the text of a book or of an handwritten sentence, etc. All these signals are the object of interest of Digital Signal Processing. Signal processing algorithms operate on these signals in order to extract information, to enhance some characteristics of the signal, to equalize the signal by compensating for both linear and nonlinear distortions, to compress data by removing redundancy, to regenerate the signal after its acquisition, to cite but a few applications.

For several years linear filters have played a crucial role in the de-

velopment of digital signal processing techniques. The success of linear filters is due to their inherent simplicity. Design, analysis and implementations of these filters are relatively straightforward in many applications. However, there are several situations in which linear filters perform poorly. In fact many real world systems present nonlinear characteristics. In order to cope with nonlinear systems different nonlinear filters have been developed. The most important nonlinear filters families are homomorphic filters, morphological filters, nonlinear mean filters, order statistics filters and polynomial (or Volterra) filters. The class of polynomial filters in particular has focused the attention of researchers for its simplicity. Volterra filter input-output relationship derives from the truncation of the discrete Volterra series. A brief review of the Volterra filter theory is presented in Appendix 1.A. Several researchers have used the Volterra series representation of nonlinear systems in order to cope with the nonlinearities that are present in digital satellite links [10, 11], in high density magnetic recordings [16], in voiceband data transmission systems [13], in high density optical systems [1], in biological phenomena [63, 74], in semiconductor devices [64, 99, 100], in image processing [112], in drift oscillations in random seas [71], in human voice [96] and in loudspeakers [49, 53, 69].

One of the most active research areas of Signal Processing is that of Adaptive Filtering. Adaptive filters have the remarkable ability of optimizing their own performances through recursive modification of internal parameters. This characteristic is particularly advantageous when the application environment in which they operate is not accurately known *a priori* by the designer. Moreover, this self adjustment capability allows the adaptive filters to operate optimally also in presence of signals with time varying statistics. As a matter of fact adaptive filters have found applications in areas as diverse as voiceband data modems, antenna arrays, radar, sonar, digital satellite transmission, mobile telephony, speech compression, voice echo cancellation and spectral estimation, to name but a few. In Figure 1.1 the typical structure of an adaptive filter is represented. The adaptive filter coefficients are adapted at each time in order to minimize a cost function of the error $e(n)$ between the output of the adaptive filter and a desired signal $d(n)$.

This is the scenario where the research work of this thesis takes place. The Ph.D. candidate has made research on two main fields of Digital Signal Processing: on linear and nonlinear adaptive filtering and on the emerging problem of nonlinear channel equalization or linearization by means of both recursive and non recursive polynomial filters. The research work was started by the development of some novel fast and nu-

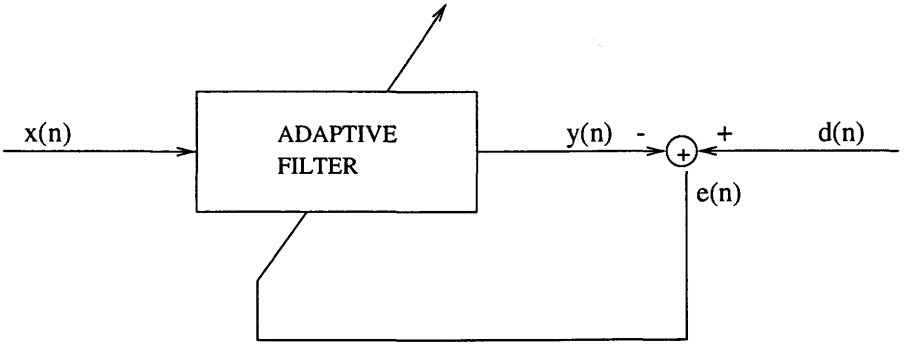


Figure 1.1: The adaptive filter.

merically stable square root RLS algorithms for linear filters. A very common and fast converging approach to adaptive filtering is the Recursive Least Square (RLS) technique. In this technique at each time the following exponentially windowed cost function is minimized

$$J_n = \sum_{k=0}^{n} \lambda^{n-k}[d(k) - d_n(k)]^2 \tag{1.1}$$

where λ is an exponential weight called "forgetting factor" that controls the rate of tracking time varying signals, $d(k)$ is the desired adaptive filter output and $d_n(k)$ is the adaptive filter output at time k. In particular in the case of linear adaptive filters we have

$$d_n(k) = \mathbf{w}_n^T \mathbf{x}_k \tag{1.2}$$

where $\mathbf{x}_k$ is the input data vector and $\mathbf{w}_n$ is the optimal vector coefficient at time n. The classical solution of the RLS problem requires an order N^2 of multiplications (where N is the linear filter memory length) and suffers long term numerical instability in a limited precision environment. Several algorithms have appeared in literature in order to solve the RLS problem in a fast manner (*i.e.* with computational complexity of order N). Many of these algorithms suffer severe numerical stability problems [29, 29, 41, 60, 134]. The most successful fast and numerically stable classes of algorithms are the Stabilized Fast Transversal Filter [135], the Lattice RLS [60] and Fast Lattice QR algorithms [110, 111, 115, 121]. In particular, the latter class has focused the interest of researchers in recent years for its numerical robustness. The student has developed some novel fast and numerically stable RLS algorithms based on two different square root factorizations of the inverse autocorrelation matrix. The novel algorithms belong to the class of Fast Lattice QR algorithms,

but the derivation of the algorithms is algebraic, rather than geometrical as in most of Fast QR algorithms.

Successively, the student has involved himself in extending these algorithms to the general class of Volterra filters. In particular, a novel algebra called V-vector Algebra has been developed. This is a formal basis that allows the development of Volterra adaptive filter algorithms as an extension of linear adaptive techniques. Particularly, the vectors of linear algebra are here substituted by a novel entity, the V-vector, which can be viewed as a non rectangular matrix. Despite V-vector algebra was initially derived in order to develop Volterra adaptive filters, it can be applied also for the development of multichannel linear adaptive filters with channels of different memory lengths.

Another field of adaptive filtering that was studied by the candidate is the adaptive infinite impulse response (IIR) filtering. Adaptive IIR filters have been the subject of active research for several years. In spite of the large amount of work that has been done some open issues still remain. One of these issues is that of ensuring the stability of the time-varying IIR filter that results from the identification process. The candidate has contributed to this research area by developing a sufficient time-varying bound on the maximum variation of the coefficients of an exponentially stable time-varying direct-form homogeneous linear recursive filter. The stability bound is less conservative than all previously derived bounds for time-varying IIR systems. The bound is then applied to control the step size of output-error adaptive IIR filters to achieve bounded-input bounded-output stability of the adaptive filter.

The second part of the doctorate research work was devoted to the emerging problem of nonlinear equalization or linearization. This research started with the development of some theorems for the exact inversion and pth order inversion of a large class of nonlinear filters. This class includes most causal polynomial systems with finite order as well as many nonlinear filters with nonpolynomial input-output relationship. In particular, it was proved that the inverse of many Volterra filters is a recursive polynomial filter. Recursive polynomial filters are inherently unstable in the sense that it is always possible to find bounded input signals which drive the filter to instability. Indeed, the stability of many polynomial filters is input dependent: if the linear part of the filter is stable and the input signal is sufficiently small the filter is stable [21, 68, 75, 84, 86, 87, 88, 97, 98]. A first sufficient stability condition for recursive quadratic filters was derived by the student at the time of the Laurea thesis. During the Ph.D. studies, this condition has been extended to the most general case of recursive Volterra filters of any finite

order.

By exploiting the expression of the exact inverse of recursive or nonrecursive polynomial filters that was recently derived, a theory for the exact and pth order equalization or linearization of nonlinear systems with known recursive or nonrecursive polynomial input-output relationship was developed. The theory is an extension of the standard equalization technique for linear systems. Moreover, the proposed pth order linearizers/equalizers can be implemented by cascading modular and stable components. Thus, these filters can be easily realized using VLSI circuits. By using the above mentioned theory of pth order equalization/linearization a subband pth order prelinearizer for loudspeakers was designed and successfully tested in a synthetic environment.

The dissertation is organized as follows. In Chapter 2, the novel fast square-root RLS adaptive filtering algorithms for linear filters are presented. In Chapter 3 the extension by means of V-vector algebra of linear adaptive filters to Volterra and linear multichannel filters is discussed. The sufficient stability bounds for slowly-varying direct-form recursive linear filters and their applications in adaptive IIR filters are presented in Chapter 4. The stability of discrete time recursive polynomial filter is discussed in Chapter 5. In Chapter 6, the theorems we mentioned earlier about the inversion of a wide class of nonlinear systems are derived. Chapter 7 presents the theory for the equalization and linearization of nonlinear systems. Final conclusions and recommendations for future work are made in Chapter 8.

Appendices

1.A Discrete Volterra Filters

A single input single output discrete system can be defined as an operator that maps input sequences $x(n)$ of $\mathcal{R}$[1] or $\mathcal{C}$ into output sequences $y(n)$ of $\mathcal{R}$ or $\mathcal{C}$, where the discrete index n belongs to $\mathcal{Z}$ or $\mathcal{N}$, such that

$$y(n) = S[x(n)]. \tag{1.3}$$

It is well known that *time-invariant* (TI) systems are characterized by the property

$$y(n+m) = S[x(n+m)] \tag{1.4}$$

and that the output of a linear time-invariant system (LTI) is given by the convolutional sum

$$y(n) = \sum_{m=-\infty}^{+\infty} h(m)x(n-m), \tag{1.5}$$

where $h(m)$ is the impulse response of the discrete system and completely characterizes the LTI system.

A system is defined *causal* if the output at any given time does not depend on the future values of the input. An LTI system is causal if and only if

$$h(m) = 0 \quad \forall m < 0. \tag{1.6}$$

In a very similar manner, a discrete nonlinear time invariant system (NTI), with certain restrictions, can be represented with the following input-output relationship called Volterra series.

$$y(n) = h_0 + \sum_{m_1=-\infty}^{+\infty} h_1(m_1)x(n-m_1) +$$

[1] $\mathcal{N}, \mathcal{Z}, \mathcal{R}, \mathcal{C}$ represent the sets of natural, integer, real, complex numbers, respectively

$$\sum_{m_1=-\infty}^{+\infty} \sum_{m_2=-\infty}^{+\infty} h_2(m_1, m_2)x(n-m_1)x(n-m_2) + \ldots + \sum_{m_1=-\infty}^{+\infty} \ldots \sum_{m_p=-\infty}^{+\infty} h_p(m_1, \ldots, m_p)x(n-m_1)\ldots x(n-m_p) + \ldots \tag{1.7}$$

Another way of expressing the input-output relationship in (1.7) is

$$y(n) = H_0[x(n)] + H_1[x(n)] + H_2[x(n)] + \ldots + H_p[x(n)] + \ldots \tag{1.8}$$

in which

$$H_0[x(n)] = h_0 \tag{1.9}$$

and

$$H_p[x(n)] = \sum_{m_1=-\infty}^{+\infty} \ldots \sum_{m_p=-\infty}^{+\infty} h_p(m_1, \ldots, m_p)x(n-m_1)\ldots x(n-m_p). \tag{1.10}$$

Note the analogy with linear filters: the input-output relationship in (1.7) is a sum of convolutional series of different order and the term of order 1 is the input-output relation of a discrete linear filter. The terms $h_p(m_1, m_2, \ldots, m_p)$ are called Volterra kernels and fully characterize the nonlinear system response. In particular we can always assume these kernels symmetric, *i.e.* by considering any permutation $(i_1, i_2, \ldots, i_p)$ of the indexes $(m_1, m_2, \ldots, m_p)$ we can always assume

$$h_p(i_1, i_2, \ldots, i_p) = h_p(m_1, m_2, \ldots, m_p). \tag{1.11}$$

In fact, if the Volterra kernel $h_p(m_1, m_2, \ldots, m_p)$ is not symmetric we can compute the kernel

$$h'_p(m_1, m_2, \ldots, m_p) = \frac{1}{p!} \sum_{\substack{\text{all possible} \\ \text{permutations of} \\ (m_1, \ldots, m_p)}} h_p(n_1, n_2, \ldots, n_p). \tag{1.12}$$

By substitution in (1.10) it is trivial to prove that $h_p(m_1, \ldots, m_p)$ and $h'_p(m_1, \ldots, m_p)$ define the same operator.

A nonlinear time-invariant system is causal if and only if

$$h_p(m_1, m_2, \ldots, m_p) = 0 \quad \forall m_i < 0 \quad i = 1, \ldots, p \quad \text{and} \quad \forall p. \tag{1.13}$$

The causal operator $H_p[x(n)]$ can also be written as

$$H_p[x(n)] = \sum_{m_1=0}^{+\infty} \sum_{m_1=m_2}^{+\infty} \ldots \sum_{m_p=m_{p-1}}^{+\infty} h_p(m_1, \ldots, m_p)x(n-m_1)\ldots x(n-m_p) \tag{1.14}$$

where we have summed together all the terms $h_p(m_1, \ldots, m_p)$ that differ only for an index permutation.

The Volterra series is a power series with memory. This can be seen by changing the input by a gain factor a such that the new input is $ax(n)$. The output $y(n)$ is then

$$y(n) = \sum_{p=0}^{+\infty} H_p[ax(n)] = \sum_{p=0}^{+\infty} a^n H_p[x(n)]. \tag{1.15}$$

Indeed the Volterra series is considered as a Taylor series with memory. As a consequence of this power series character, convergence difficulties arise when nonlinear systems including saturating elements are modelled. In fact, the same problem exists also for the Taylor series representation of strongly nonlinear functions.

Discrete Volterra filters are originated from the discrete Volterra series with a double truncation. They are obtained by limiting the memory of the Volterra series to a memory length N (time truncation) and by limiting the maximum order of the nonlinearity (order truncation). The input-output relationship of a Volterra filter of order p and memory length N is given by

$$\begin{aligned} y(n) \;=\; & h_0 + \sum_{m_1=0}^{N-1} h_1(m_1)x(n-m_1) + \\ & \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \ldots + \\ & \sum_{m_1=0}^{N-1} \cdots \sum_{m_p=0}^{N-1} h_p(m_1,\ldots,m_p)x(n-m_1)\ldots x(n-m_p) \end{aligned} \tag{1.16}$$

It is costume to drop the constant term h_0 from the expression (1.16). Thus, in the next chapters we will not consider anymore this term.

Volterra filters can be represented by means of multidimensional linear transforms. In fact, it is possible to consider for example the relation that defines the second order homogeneous Volterra filter

$$y(n) = \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2(m_1,m_2)x(n-m_1)x(n-m_2) \tag{1.17}$$

as a particular form of 2-D filtering

$$w(n_1,n_2) = \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2(m_1,m_2)v(n_1-m_1,n_2-m_2) \tag{1.18}$$

where $v(n_1 - m_1, n_2 - m_2) = x(n_1 - m_1)x(n_2 - m_2)$ and $n_1 = n_2 = n$, such that $y(n) = w(n, n)$. In order to characterize the quadratic kernel it is possible to use the 2-D z-transform of $h_2(m_1, m_2)$

$$H_2(z_1, z_2) = \sum_{m_1=0}^{N-1} \sum_{m_2=0}^{N-1} h_2(m_1, m_2) z_1^{-m_1} z_2^{-m_2} \tag{1.19}$$

or its Fourier transform. We can deal in a similar manner also with higher order homogeneous Volterra filters.

Another interesting interpretation of Volterra filters is that of considering the Volterra filter as a linear multichannel filter bank with channels of different memory length and input given by a product of samples. Let us consider for instance the second order homogeneous filter in (1.17). This is also given by

$$y(n) = \sum_{i=0}^{N-1} L_i[x(n)x(n-i)] \tag{1.20}$$

where the linear filter L_i is

$$L_i[u(n)] = \sum_{j=0}^{N-1-i} h_2(j, j+i)u(n-j). \tag{1.21}$$

This interpretation of Volterra filters has inspired the development of the V-vector Algebra presented in Chapter 3.

Chapter 2

Fast Square-Root RLS Adaptive Filtering Algorithms

2.1 Introduction

Many real-time signal processing problems, such as adaptive filtering and prediction as well as system identification, can be solved by means of recursive least squares (RLS) algorithms [12, 60]. However, sometimes RLS algorithms exhibit unacceptable numerical performances in limited precision environments. Numerical problems during recursion can be particularly experienced in 'fast' RLS algorithms. Fast RLS algorithms require a complexity that grows linearly with the filter order. A number of algorithms which overcome the numerical instability problem of fast RLS algorithms have appeared in the literature [89, 90, 110, 111, 115, 135]. Popular fast and stable RLS algorithms such as the numerically stable fast transversal filter (SFTF) [135] and the fast lattice QR decomposition algorithms [110, 111, 115, 121] reduce the computational complexity to $O(N)$ operations per time instant. Other stable algorithms, such as the QR decomposition based algorithms [4, 51] or the square-root Schur RLS adaptive filters [143, 144] require, for their fast implementation, a systolic array of processing elements.

Part of the content of this chapter was presented in
Alberto Carini, "A Novel Givens Rotation Based Fast SQR-RLS Algorithm," *Proc. of EUSIPCO-96, VIII European Signal Processing Conference*, Trieste, Italy, September 10-13 1996, pp. 1235-1238
Alberto Carini and Enzo Mumolo, "Fast square-root RLS Adaptive Filtering Algorithms," *Signal Processing*, Vol. 57, No. 3, Mar. 1997, pp. 233-250

In this chapter we present some novel fast and numerically stable RLS algorithms. These algorithms are based on the derivation of two different Cholesky or UDU^T square-root factorizations of the autocorrelation matrix $\mathbf{\Omega}_n$. Actually, in classical RLS algorithms, numerical instabilities arise because, due to the finite precision of processors and to error propagation, the autocorrelation matrix looses its properties, namely symmetry and positive definitiveness. In fast RLS algorithms, instabilities may appear because the Kalman Gain vector, directly derived from $\mathbf{\Omega}_n$, at a certain point cannot be associated to any positive definite autocorrelation matrix. In square-root algorithms, however, this problem is avoided by directly updating a square-root factor of $\mathbf{\Omega}_n$ or a quantity related to this factor. In this way, in fact, we implicitly impose the symmetry and positive definitiveness of the autocorrelation matrix. However, the square-root technique by itself is not sufficient to achieve the numerical stability of the algorithm, which depends also on the numerically robust computation of each parameter of the algorithm. Extensive experimentation has shown that the proposed algorithms exhibit excellent robustness in limited precision environments, and adaptive filtering with the mantissa rounded to 4 bits have been performed for over 4 million samples without any instability. Furthermore, a comparison with the SFTF adaptive algorithm shows a much better numerical behaviour in low precision environments.

The proposed algorithms are closely connected with the classical Fast QR and Lattice QR algorithms [4, 9, 31, 51, 60, 89, 90, 110, 111, 115, 121, 149]. Like these algorithms, they are based on the exploitation of a Cholesky or UDU^T factor of the autocorrelation matrix and this fact leads to the coincidence of the joint process part, which estimates the desired signal. Unlike Fast QR and Lattice filters, however, their derivation is not geometrical but rather algebraic and it is based on the identity between two Cholesky or UDU^T factorization forms of the autocorrelation matrix.

It is worth stressing that the proposed algorithms do not determine the filter coefficients of a transversal filter but the coefficients of a lattice realization. Thus, the algorithms can be applied to system identification as well as to adaptive filtering and prediction applications.

Finally, a direct dependency of the prediction error from the input signal makes these algorithms suitable for ADPCM applications in signal coding.

The outline of this chapter is the following. In Section 2.2 we review the RLS adaptive problem and we recall some quantities which are very important for the development of fast algorithms and which are also

used in our algorithms. In Section 2.3, different forms of upper triangular square-root factorizations of the autocorrelation matrix, which constitute the base of the proposed algorithms, are derived. Section 2.4 is devoted to the description of the fast, $O(N)$, RLS algorithms and in Section 2.5 some experimental results, including a comparison with the algorithm reported in [135], will be given. Finally, in Section 2.6, some concluding remarks are reported.

2.2 Review of RLS Adaptive Filtering

The output of a time-varying FIR filter of order N is given by

$$d_n(k) = \mathbf{w}_n^T \mathbf{x}_k, \tag{2.1}$$

where

$$\mathbf{w}_n^T = [\ w_0(n), w_1(n), \ldots, w_{N-1}(n)\ ] \tag{2.2}$$

is the impulse response at time instant n and

$$\mathbf{x}_k^T = [\ x(k), x(k-1), \ldots, x(k-N+1)\ ] \tag{2.3}$$

is the input vector at time instant k. Let $d(k)$ be the desired response signal. The objective is to compute the coefficient vector $\mathbf{w}_n$ in such a way that the filter output is as close as possible to the desired response signal. This leads to a minimization of the exponentially weighted cost function

$$J_n = \sum_{k=0}^{n} \lambda^{n-k} |d(k) - \mathbf{w}_n^T \mathbf{x}_k|^2 \tag{2.4}$$

at each time instant n, where λ is a forgetting factor that controls the speed of tracking time-varying signals. The solution of the minimization problem is given by

$$\mathbf{w}_n = \boldsymbol{\Omega}_n^{-1} \mathbf{p}_n, \tag{2.5}$$

where

$$\boldsymbol{\Omega}_n = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_k \mathbf{x}_k^T, \quad \text{and} \quad \mathbf{p}_n = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_k d(k) \tag{2.6}$$

are the autocorrelation and crosscorrelation matrices respectively. The problem is to develop a recursive version of (2.5) such that the number of operations per time instant is minimum and such that the recursion is numerically stable. For notational convenience, let us define $r_{k-1} = x(k-N)$ and $v_k = x(k)$. One can note that the input vector at time k, $\mathbf{x}_k$, can be formed discarding the r_{k-1} element contained in the vector $\mathbf{x}_{k-1}$ and adding the new v_k element.

The forward predictor $\mathbf{a}_n$ is defined as the filter which estimates v_n from $\mathbf{x}_{n-1}$. Similarly, the backward predictor $\mathbf{b}_n$ estimates r_{n-1} from $\mathbf{x}_n$. The forward and backward prediction errors are respectively given by

$$f_n(k) = v_k + \mathbf{a}_n^T \mathbf{x}_{k-1}, \tag{2.7}$$

$$b_n(k) = r_{k-1} + \mathbf{b}_n^T \mathbf{x}_k. \tag{2.8}$$

The Kalman gain vector $\mathbf{c}_n$ plays a fundamental role in the development of classical fast algorithms [60]. The definition of the Kalman gain vector is the following

$$\mathbf{c}_n = \boldsymbol{\Omega}_n^{-1} \mathbf{x}_n. \tag{2.9}$$

From (2.5), the gain vector can be viewed as a predictor which estimates the pinning sequence from $\mathbf{x}_k$ [60]. The corresponding prediction error, called likelihood variable, is reported in (2.10):

$$\gamma_n = 1 - \mathbf{c}_n^T \mathbf{x}_n. \tag{2.10}$$

The likelihood variable assumes a great importance in all Fast Transversal Filter algorithms. In fact it monitors the numerical stability of the algorithm itself. According to [60], γ_n is a real value bounded by zero and one, $0 \leq \gamma_n \leq 1$, and instability arises when γ_n exceeds these bounds due to finite precision of processors and to error propagation.

It is worth recalling that the forward and backward predictors can be recursively estimated as:

$$\mathbf{a}_n = \mathbf{a}_{n-1} - \mathbf{c}_{n-1} f_{n-1}(n), \tag{2.11}$$

$$\mathbf{b}_n = \mathbf{b}_{n-1} - \mathbf{c}_n b_{n-1}(n). \tag{2.12}$$

The terms v_k and r_{k-1} are used in the definition of the augmented (extended) input vector $\overline{\mathbf{x}}_k$ where

$$\overline{\mathbf{x}}_k = \begin{bmatrix} v_k \\ \mathbf{x}_{k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k \\ r_{k-1} \end{bmatrix}. \tag{2.13}$$

Finally, the autocorrelation of the forward and backward prediction errors are given by:

$$\alpha_n = \sum_{k=0}^{n} \lambda^{n-k} f_n^2(k); \qquad \beta_n = \sum_{k=0}^{n} \lambda^{n-k} b_n^2(k). \tag{2.14}$$

Fast RLS algorithms are usually based on the exploitation of the relationships between the quantities described above.

2.3 Preliminary Results

In this section we first derive some basic relations that involve the quantities defined in the previous paragraph. These relations are then used to develop different factorizations of the autocorrelation matrix. A brief review of the Givens rotations follows at the end of the section.

2.3.1 Some Useful Relations

Proposition 2.3.1 *Let $f_n(k)$, $b_n(k)$ and $\mathbf{x}_{k-1}$ be the forward, backward errors and input vector respectively, as defined in (2.7), (2.8) and (2.3). Then the following two relations hold:*

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} f_n(k) = 0 \tag{2.15}$$

and

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_k b_n(k) = 0. \tag{2.16}$$

Proof First, using (2.5), let us give the following formulation of the forward predictor coefficient vector:

$$\mathbf{a}_n = -\left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T\right)^{-1} \left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} v_k\right). \tag{2.17}$$

Let us now consider the term

$$\mathbf{a}_n^T \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} f_n(k), \tag{2.18}$$

which, using (2.17), becomes

$$-\left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1}^T v_k\right) \left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T\right)^{-1} \left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} f_n(k)\right). \tag{2.19}$$

Substituting v_k from (2.7) in (2.19) it follows that

$$\left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1}^T f_n(k)\right) \boldsymbol{\Omega}_{n-1}^{-1} \left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} f_n(k)\right) = 0. \tag{2.20}$$

Since $\boldsymbol{\Omega}_{n-1}^{-1}$ is positive definite, the first equation claimed in this Proposition follows immediately from (2.20). The derivation of the second equation can be readily obtained in the same way. □

Alternatively, equations (2.15) and (2.16) can also be proved by means of the principle of orthogonality that states the following [60]:

> "The necessary and sufficient condition for the cost function J_n to attain its minimum value is that the corresponding value of the estimation error is orthogonal to each input sample that enters into the estimation of the desired response at time n."

Proposition 2.3.2 *Let $\boldsymbol{\Omega}_n$ be the autocorrelation matrix, α_n, β_n be the autocorrelations of the forward and backward prediction errors and $\mathbf{a}_n$, $\mathbf{b}_n$ the corresponding predictors, as defined in Section 2.2. Then the following two relations hold:*

$$\sum_{k=0}^{n} \lambda^{n-k} v_k^2 = \alpha_n + \mathbf{a}_n^T \boldsymbol{\Omega}_{n-1} \mathbf{a}_n \tag{2.21}$$

and

$$\sum_{k=0}^{n} \lambda^{n-k} r_{k-1}^2 = \beta_n + \mathbf{b}_n^T \boldsymbol{\Omega}_n \mathbf{b}_n. \tag{2.22}$$

Proof Let us prove the first expression. Using (2.7), we obtain

$$\sum_{k=0}^{n} \lambda^{n-k} v_k^2 = \sum_{k=0}^{n} \lambda^{n-k} \left(f_n(k) - \mathbf{a}_n^T \mathbf{x}_{k-1} \right)^2 =$$

$$= \sum_{k=0}^{n} \lambda^{n-k} f_n^2(k) + \mathbf{a}_n^T \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T \mathbf{a}_n - 2\mathbf{a}_n^T \sum_{k=0}^{n} \lambda^{n-k} f_n(k) \mathbf{x}_{k-1}.$$

From Proposition 2.3.1 and using (2.14), the first expression claimed in this Proposition follows immediately. The second expression is proved very similarly. □

Proposition 2.3.3 *Let $\boldsymbol{\Omega}_n$, $\mathbf{x}_k$ and $\mathbf{a}_n$, $\mathbf{b}_n$ be the autocorrelation matrix, the input vector and the forward, backward predictors respectively, as defined in Section 2.2. Then the following two relations hold:*

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} v_k = -\boldsymbol{\Omega}_{n-1} \mathbf{a}_n \tag{2.23}$$

and

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_k^T r_{k-1} = -\mathbf{b}_n^T \boldsymbol{\Omega}_n. \tag{2.24}$$

Proof Using (2.7), substitute v_k in the first term of the first expression. Then the following relation is obtained:

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} v_k = \sum_{k=0}^{n} \lambda^{n-k} f_n(k) \mathbf{x}_{k-1} - \left(\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T \right) \mathbf{a}_n.$$

Therefore, using Proposition 2.3.1 the first relation is proved. The second expression can be derived in a similar way . □

2.3.2 Factorizations of the Autocorrelation Matrix

The fast algorithms described in this chapter are based on the development of different factorization forms of the augmented autocorrelation matrix $\overline{\mathbf{\Omega}}_n$, where

$$\overline{\mathbf{\Omega}}_n = \sum_{k=0}^{n} \lambda^{n-k} \overline{\mathbf{x}}_k \overline{\mathbf{x}}_k^T \tag{2.25}$$

and $\overline{\mathbf{x}}_k$ is defined in (2.13). In this section, we will describe different types of factorization, which will lead to different developments.

Square-Root factorization

It is worth recalling that, if a matrix can be factorized as follows

$$\overline{\mathbf{\Omega}}_n^{-1} = \overline{\mathbf{S}}_n \overline{\mathbf{S}}_n^T \tag{2.26}$$

then $\overline{\mathbf{S}}_n$ is said a square-root of $\overline{\mathbf{\Omega}}_n^{-1}$. Similarly

$$\mathbf{\Omega}_n^{-1} = \mathbf{S}_n \mathbf{S}_n^T, \tag{2.27}$$

where $\mathbf{S}_n$ is the square-root of $\mathbf{\Omega}_n^{-1}$. The algorithms we propose are based on the derivation of two factorizations of the inverse autocorrelation matrix. Note that these square-root matrices of $\mathbf{\Omega}_n^{-1}$ coincide only if they are both upper (or lower) triangular with positive diagonal (Cholesky factorization). If this condition is not met the square-root matrices differ for a rotation matrix $\mathbf{Q}$ ($\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$). It is important to point out that, in this chapter, we always assume $\mathbf{S}_n$ an upper triangular with positive diagonal matrix in order to ensure the uniqueness of the square-root factorization. The following Lemma describes a possible form for the extended square-root matrices.

Lemma 2.3.1 *Let us consider the square-root factorization of the inverse augmented autocorrelation matrix described in (2.26). Then* $\overline{\mathbf{S}}_n$ *and its inverse* $\overline{\mathbf{S}}_n^{-1}$ *can be computed as*

$$\overline{\mathbf{S}}_n = \begin{bmatrix} \mathbf{S}_n & \beta_n^{-\frac{1}{2}}\mathbf{b}_n \\ \mathbf{0}^T & \beta_y^{-\frac{1}{2}} \end{bmatrix} \tag{2.28}$$

and

$$\overline{\mathbf{S}}_n^{-1} = \begin{bmatrix} \mathbf{S}_n^{-1} & -\mathbf{y}_n \\ \mathbf{0}^T & \beta_n^{\frac{1}{2}} \end{bmatrix} \tag{2.29}$$

where $\mathbf{0}$ *is a vector with N zero elements,* $\mathbf{b}_n$, β_n *are defined in Section 2.2,* $\mathbf{S}_n$ *is the square-root of* $\boldsymbol{\Omega}_n^{-1}$ *as reported in (2.27) and* $\mathbf{y}_n = \mathbf{S}_n^{-1}\mathbf{b}_n$.

Proof From (2.25) and (2.13) it turns out that

$$\overline{\boldsymbol{\Omega}}_n = \sum_{k=0}^{n} \lambda^{n-k} \begin{bmatrix} \mathbf{x}_k \\ r_{k-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_k^T & r_{k-1} \end{bmatrix} =$$

$$= \begin{bmatrix} \sum_{k=0}^{n} \lambda^{n-k}\mathbf{x}_k\mathbf{x}_k^T & \sum_{k=0}^{n} \lambda^{n-k}\mathbf{x}_k r_{k-1} \\ \sum_{k=0}^{n} \lambda^{n-k}\mathbf{x}_k^T r_{k-1} & \sum_{k=0}^{n} \lambda^{n-k} r_{k-1}^2 \end{bmatrix}. \tag{2.30}$$

Using (2.27) and Propositions 2.3.2-2.3.3, (2.30) can be written as [1]:

$$\overline{\boldsymbol{\Omega}}_n = \begin{bmatrix} \mathbf{S}_n^{-T}\mathbf{S}_n^{-1} & -\mathbf{S}_n^{-T}\mathbf{S}_n^{-1}\mathbf{b}_n \\ -\mathbf{b}_n^T\mathbf{S}_n^{-T}\mathbf{S}_n^{-1} & \beta_n + \mathbf{b}_n^T\mathbf{S}_n^{-T}\mathbf{S}_n^{-1}\mathbf{b}_n \end{bmatrix}, \tag{2.31}$$

Let us now consider the following partitioned form of the $\overline{\mathbf{S}}_n^{-1}$ matrix:

$$\overline{\mathbf{S}}_n^{-1} = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\theta} \\ \boldsymbol{\psi}^T & \xi \end{bmatrix}, \tag{2.32}$$

where $\boldsymbol{\Phi}$ is N by N, $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are column vectors of N elements and ξ is a scalar. Therefore, from (2.26) and (2.32), we have

$$\overline{\boldsymbol{\Omega}}_n = \overline{\mathbf{S}}_n^{-T}\overline{\mathbf{S}}_n^{-1} = \begin{bmatrix} \boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\psi}\boldsymbol{\psi}^T & \boldsymbol{\Phi}^T\boldsymbol{\theta} + \boldsymbol{\psi}\xi \\ \boldsymbol{\theta}^T\boldsymbol{\Phi} + \xi\boldsymbol{\psi}^T & \boldsymbol{\theta}^T\boldsymbol{\theta} + \xi^2 \end{bmatrix}. \tag{2.33}$$

Equating (2.31) and (2.33) we obtain:

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\psi}\boldsymbol{\psi}^T = \mathbf{S}_n^{-T}\mathbf{S}_n^{-1}, \tag{2.34}$$

[1] Note that $\mathbf{S}^{-T}$ indicates the inverse transposed of matrix $\mathbf{S}$.

$$\boldsymbol{\Phi}^T\boldsymbol{\theta} + \boldsymbol{\psi}\xi = -\mathbf{S}_n^{-T}\mathbf{S}_n^{-1}\mathbf{b}_n, \tag{2.35}$$

$$\boldsymbol{\theta}^T\boldsymbol{\theta} + \xi^2 = \beta_n + \mathbf{b}_n^T\mathbf{S}_n^{-T}\mathbf{S}_n^{-1}\mathbf{b}_n. \tag{2.36}$$

If we impose $\boldsymbol{\psi} = \mathbf{0}$, (2.32) becomes upper triangular. Its elements are readily obtained as:

$$\boldsymbol{\Phi} = \mathbf{S}_n^{-1}, \tag{2.37}$$

$$\boldsymbol{\theta} = -\mathbf{S}_n^{-1}\mathbf{b}_n = -\mathbf{y}_n, \tag{2.38}$$

$$\xi = \beta_n^{\frac{1}{2}}. \tag{2.39}$$

This proves equation (2.29). The partitioned matrix (2.29) can be easily inverted, since one of the submatrices is zero. Equation (2.28) comes by inverting (2.29). □

In the following Lemma, two non upper-triangular square-root matrices for the autocorrelation and inverse autocorrelation augmented matrices are obtained.

Lemma 2.3.2 *Let us consider a factorization of the inverse augmented autocorrelation matrix, that is* $\overline{\boldsymbol{\Omega}}_n^{-1} = \tilde{\mathbf{S}}_n\tilde{\mathbf{S}}_n^T$. *Then, a possible form of the* $\tilde{\mathbf{S}}_n$ *matrix is the following:*

$$\tilde{\mathbf{S}}_n = \begin{bmatrix} \alpha_n^{-\frac{1}{2}} & \mathbf{0}^T \\ \alpha_n^{-\frac{1}{2}}\mathbf{a}_n & \mathbf{S}_{n-1} \end{bmatrix} \tag{2.40}$$

and the corresponding inverse matrix $\tilde{\mathbf{S}}_n^{-1}$ *is:*

$$\tilde{\mathbf{S}}_n^{-1} = \begin{bmatrix} \alpha_n^{\frac{1}{2}} & \mathbf{0}^T \\ -\mathbf{z}_n & \mathbf{S}_{n-1}^{-1} \end{bmatrix}. \tag{2.41}$$

where α_n *is the autocorrelation of the forward prediction error,* $\mathbf{a}_n$ *is the forward predictor,* $\mathbf{S}_n$ *is described in (2.27) and* $\mathbf{z}_n = \mathbf{S}_{n-1}^{-1}\mathbf{a}_n$.

Proof Similarly to Lemma 2.3.1, we can write

$$\overline{\boldsymbol{\Omega}}_n = \sum_{k=0}^{n} \lambda^{n-k} \begin{bmatrix} v_k \\ \mathbf{x}_{k-1} \end{bmatrix} \begin{bmatrix} v_k & \mathbf{x}_{k-1}^T \end{bmatrix} =$$

$$= \begin{bmatrix} \alpha_n + \mathbf{a}_n^T\mathbf{S}_{n-1}^{-T}\mathbf{S}_{n-1}^{-1}\mathbf{a}_n & -\mathbf{a}_n^T\mathbf{S}_{n-1}^{-T}\mathbf{S}_{n-1}^{-1} \\ -\mathbf{S}_{n-1}^{-T}\mathbf{S}_{n-1}^{-1}\mathbf{a}_n & \mathbf{S}_{n-1}^{-T}\mathbf{S}_{n-1}^{-1} \end{bmatrix}, \tag{2.42}$$

where Propositions 2.3.2-2.3.3 have been used. From (2.42), it turns out that $\widetilde{\mathbf{S}}_n^{-1}$ can be partitioned as follows:

$$\widetilde{\mathbf{S}}_n^{-1} = \begin{bmatrix} \xi & \boldsymbol{\theta}^T \\ \boldsymbol{\psi} & \boldsymbol{\Phi} \end{bmatrix}, \tag{2.43}$$

where ξ, $\boldsymbol{\theta}$, $\boldsymbol{\psi}$ and $\boldsymbol{\Phi}$ are sized as in Lemma 2.3.1. By computing the product $\widetilde{\mathbf{S}}_n^{-T}\widetilde{\mathbf{S}}_n$ we have:

$$\overline{\boldsymbol{\Omega}}_n = \begin{bmatrix} \xi^2 + \boldsymbol{\psi}^T\boldsymbol{\psi} & \xi\boldsymbol{\theta}^T + \boldsymbol{\psi}^T\boldsymbol{\Phi} \\ \xi\boldsymbol{\theta} + \boldsymbol{\Phi}^T\boldsymbol{\psi} & \boldsymbol{\theta}\boldsymbol{\theta}^T + \boldsymbol{\Phi}^T\boldsymbol{\Phi} \end{bmatrix}. \tag{2.44}$$

Hence, setting $\boldsymbol{\theta} = \mathbf{0}$ and equating (2.42) with (2.44), we obtain:

$$\widetilde{\mathbf{S}}_n^{-1} = \begin{bmatrix} \alpha_n^{\frac{1}{2}} & \mathbf{0}^T \\ -\mathbf{S}_{n-1}^{-1}\mathbf{a}_n & \mathbf{S}_{n-1}^{-1} \end{bmatrix}. \tag{2.45}$$

This proves equation (2.41). By inverting (2.41) we obtain immediately (2.40). □

Till now, two different factorizations of the autocorrelation matrix have been derived. In the following Lemma, we will describe another form of the upper triangular square-root factorization for $\overline{\boldsymbol{\Omega}}_n^{-1}$ which, due to the uniqueness of the upper triangular with positive diagonal factorization, can be equated to that derived in Lemma 2.3.1.

Lemma 2.3.3 *Let us consider the square-root factorization of the inverse extended autocorrelation matrix given in (2.26). Then, the upper triangular square-root matrix* $\overline{\mathbf{S}}_n$ *is the following:*

$$\overline{\mathbf{S}}_n = \begin{bmatrix} \sqrt{\frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}} & \frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}\mathbf{z}_n^T\mathbf{L}_n \\ \mathbf{0} & \mathbf{S}_{n-1}\mathbf{L}_n \end{bmatrix}, \tag{2.46}$$

where $\mathbf{z}_n$ *is defined as:*

$$\mathbf{z}_n = \mathbf{S}_{n-1}^{-1}\mathbf{a}_n \tag{2.47}$$

and $\mathbf{L}_n$ *is the following upper triangular factorization matrix:*

$$\left(\mathbf{I} + \alpha_n^{-1}\mathbf{z}_n\mathbf{z}_n^T\right) = \mathbf{L}_n\mathbf{L}_n^T. \tag{2.48}$$

Proof Using the factorization derived in Lemma 2.3.2, we obtain the following expression of the inverse autocorrelation matrix:

$$\overline{\boldsymbol{\Omega}}_n^{-1} = \begin{bmatrix} \alpha_n^{-1} & \alpha_n^{-1}\mathbf{a}_n^T \\ \alpha_n^{-1}\mathbf{a}_n & \mathbf{S}_{n-1}\mathbf{S}_{n-1}^T + \alpha_n^{-1}\mathbf{a}_n\mathbf{a}_n^T \end{bmatrix}. \tag{2.49}$$

Consider now (2.26), and assume that the square-root matrix is upper triangular:

$$\overline{\mathbf{S}}_n = \begin{bmatrix} \xi & \boldsymbol{\theta}^T \\ \mathbf{0} & \boldsymbol{\Phi} \end{bmatrix}. \tag{2.50}$$

From (2.26) and (2.50), we obtain

$$\overline{\boldsymbol{\Omega}}_n^{-1} = \begin{bmatrix} \xi^2 + \boldsymbol{\theta}^T\boldsymbol{\theta} & \boldsymbol{\theta}^T\boldsymbol{\Phi}^T \\ \boldsymbol{\Phi}\boldsymbol{\theta} & \boldsymbol{\Phi}\boldsymbol{\Phi}^T \end{bmatrix}. \tag{2.51}$$

Hence, equating (2.49) and (2.51):

$$\xi^2 + \boldsymbol{\theta}^T\boldsymbol{\theta} = \alpha_n^{-1}, \tag{2.52}$$

$$\boldsymbol{\Phi}\boldsymbol{\theta} = \alpha_n^{-1}\mathbf{a}_n, \tag{2.53}$$

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \mathbf{S}_{n-1}\mathbf{S}_{n-1}^T + \alpha_n^{-1}\mathbf{a}_n\mathbf{a}_n^T. \tag{2.54}$$

It is easy to show that, using (2.47), (2.54) can be written as

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \mathbf{S}_{n-1}\left(\mathbf{I} + \alpha_n^{-1}\mathbf{z}_n\mathbf{z}_n^T\right)\mathbf{S}_{n-1}^T. \tag{2.55}$$

From (2.55) and (2.48), $\boldsymbol{\Phi}$ can be represented as:

$$\boldsymbol{\Phi} = \mathbf{S}_{n-1}\mathbf{L}_n \tag{2.56}$$

and, from (2.53), (2.56) and (2.47), we get:

$$\boldsymbol{\theta} = \alpha_n^{-1}\mathbf{L}_n^{-1}\mathbf{z}_n. \tag{2.57}$$

Moreover, pre-multiplying the right term of (2.57) by $\mathbf{L}_n^T\mathbf{L}_n^{-T}$ and using (2.48), the $\boldsymbol{\theta}$ matrix can be represented as:

$$\boldsymbol{\theta} = \alpha_n^{-1}\mathbf{L}_n^T\left(\mathbf{I} + \alpha_n^{-1}\mathbf{z}_n\mathbf{z}_n^T\right)^{-1}\mathbf{z}_n. \tag{2.58}$$

On the other hand, using the matrix inversion lemma, we obtain

$$\left(\mathbf{I} + \alpha_n^{-1}\mathbf{z}_n\mathbf{z}_n^T\right)^{-1} = \mathbf{I} - \frac{\alpha_n^{-1}}{1 + \alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}\mathbf{z}_n\mathbf{z}_n^T. \tag{2.59}$$

Therefore, the final form of $\boldsymbol{\theta}$ is:

$$\boldsymbol{\theta} = \mathbf{L}_n^T\mathbf{z}_n\frac{\alpha_n^{-1}}{1 + \alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}. \tag{2.60}$$

By computing $\boldsymbol{\theta}^T\boldsymbol{\theta}$ and using (2.52), finally, the scalar term ξ can be computed. Using (2.60) and (2.48) we obtain:

$$\boldsymbol{\theta}^T\boldsymbol{\theta} = \alpha_n^{-2}\frac{\mathbf{z}_n^T\mathbf{z}_n}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}.$$

Hence, from (2.52) it follows that

$$\xi^2 = \alpha_n^{-1} - \boldsymbol{\theta}^T\boldsymbol{\theta} = \frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}.$$

Therefore:

$$\xi = \sqrt{\frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}}. \tag{2.61}$$

Thus, equations (2.61), (2.60) and (2.56) complete the proof. □

UD-factorization

Let us now consider another type of square-root factorization, the so called UD-factorization [12, 60]. According to the UD-factorization, the standard and extended inverse autocorrelation matrices can be factorized as follows:

$$\boldsymbol{\Omega}_n^{-1} = \mathbf{U}_n\mathbf{D}_n\mathbf{U}_n^T; \qquad \overline{\boldsymbol{\Omega}}_n^{-1} = \overline{\mathbf{U}}_n\overline{\mathbf{D}}_n\overline{\mathbf{U}}_n^T, \tag{2.62}$$

with $\mathbf{U}_n$, $\overline{\mathbf{U}}_n$ upper triangular with unit diagonal and $\mathbf{D}_n$, $\overline{\mathbf{D}}_n$ positive diagonal matrices. Clearly, the UD-factorization is itself a form of square-root factorization but, due to its structure, square-roots free fast algorithms will be obtained. We now derive different forms of the $\overline{\mathbf{U}}_n$ and $\overline{\mathbf{D}}_n$ matrices.

Lemma 2.3.4 *Let us consider the UD-factorization of the augmented autocorrelation matrix described in (2.62). Then, the* $\overline{\mathbf{U}}_n$ *and* $\overline{\mathbf{D}}_n$ *matrices are given by:*

$$\overline{\mathbf{U}}_n = \begin{bmatrix} \mathbf{U}_n & \mathbf{b}_n \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad \overline{\mathbf{D}}_n = \begin{bmatrix} \mathbf{D}_n & \mathbf{0} \\ \mathbf{0}^T & \beta_n^{-1} \end{bmatrix}, \tag{2.63}$$

the $\overline{\mathbf{U}}_n^{-1}$ *and* $\overline{\mathbf{D}}_n^{-1}$ *matrices are*

$$\overline{\mathbf{U}}_n^{-1} = \begin{bmatrix} \mathbf{U}_n^{-1} & -\mathbf{n}_n \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad \overline{\mathbf{D}}_n^{-1} = \begin{bmatrix} \mathbf{D}_n^{-1} & \mathbf{0} \\ \mathbf{0}^T & \beta_n \end{bmatrix}, \tag{2.64}$$

where $\mathbf{0}$ *is a vector of* N *zero elements,* β_n *is the autocorrelation of the backward prediction error,* $\mathbf{b}_n$ *is the backward prediction filter and* $\mathbf{n}_n = \mathbf{U}_n^{-1}\mathbf{b}_n$.

Proof We start observing that

$$\overline{\Omega}_n = \sum_{k=0}^{n} \lambda^{n-k} \overline{\mathbf{x}}_k \overline{\mathbf{x}}_k^T =$$

$$= \begin{bmatrix} \mathbf{U}_n^{-T}\mathbf{D}_n^{-1}\mathbf{U}_n^{-1} & -\mathbf{U}_n^{-T}\mathbf{D}_n^{-1}\mathbf{U}_n^{-1}\mathbf{b}_n \\ -\mathbf{b}_n^T\mathbf{U}_n^{-T}\mathbf{D}_n^{-1}\mathbf{U}_n^{-1} & \beta_n + \mathbf{b}_n^T\mathbf{U}_n^{-T}\mathbf{D}_n^{-1}\mathbf{U}_n^{-1}\mathbf{b}_n \end{bmatrix}. \tag{2.65}$$

where (2.13), (2.6), (2.62) and Proposition 2.3.2-2.3.3 have been used. Due to their internal structure, the $\overline{\mathbf{U}}_n^{-1}$ and $\overline{\mathbf{D}}_n^{-1}$ matrices in (2.62) can be partitioned as follows:

$$\overline{\mathbf{U}}_n^{-1} = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{\psi} \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad \overline{\mathbf{D}}_n^{-1} = \begin{bmatrix} \boldsymbol{\Theta} & \mathbf{0} \\ \mathbf{0}^T & \xi \end{bmatrix}, \tag{2.66}$$

where the N by N sized $\boldsymbol{\Phi}$ matrix is upper triangular with unit diagonal and $\boldsymbol{\psi}$ is a column vector. Moreover, $\boldsymbol{\Theta}$ is a N by N positive diagonal matrix and ξ is a scalar. Hence:

$$\overline{\Omega}_n = \begin{bmatrix} \boldsymbol{\Phi}^T\boldsymbol{\Theta}\boldsymbol{\Phi} & \boldsymbol{\Phi}^T\boldsymbol{\Theta}\boldsymbol{\psi} \\ \boldsymbol{\psi}^T\boldsymbol{\Theta}\boldsymbol{\Phi} & \boldsymbol{\psi}^T\boldsymbol{\Theta}\boldsymbol{\psi} + \xi \end{bmatrix}. \tag{2.67}$$

The matrices reported in (2.64) can be readily derived by realizing that, by equating (2.65) and (2.67), one can obtain:

$$\boldsymbol{\Phi} = \mathbf{U}_n^{-1}, \qquad \boldsymbol{\psi} = -\mathbf{U}_n^{-1}\mathbf{b}_n, \qquad \boldsymbol{\Theta} = \mathbf{D}_n^{-1}, \qquad \xi = \beta_n.$$

Thus, the derivation of (2.64) is concluded. The matrices in (2.63) can be found by inverting (2.64). □

Lemma 2.3.5 *A non upper triangular UDU^T factorization of the extended autocorrelation matrix $\overline{\Omega}_n = \widetilde{\mathbf{U}}_n^{-T}\widetilde{\mathbf{D}}_n^{-1}\widetilde{\mathbf{U}}_n^{-1}$ is given by:*

$$\widetilde{\mathbf{U}}_n^{-1} = \begin{bmatrix} 1 & \mathbf{0}^T \\ -\mathbf{m}_n & \mathbf{U}_{n-1}^{-1} \end{bmatrix}, \qquad \widetilde{\mathbf{D}}_n^{-1} = \begin{bmatrix} \alpha_n & \mathbf{0}^T \\ \mathbf{0} & \mathbf{D}_{n-1}^{-1} \end{bmatrix}, \tag{2.68}$$

where $\mathbf{0}$ is a vector of N zero elements, α_n is the autocorrelation of the forward prediction error and $\mathbf{m}_n = \mathbf{U}_{n-1}^{-1}\mathbf{a}_n$.

Proof From (2.13), (2.6) and Proposition 2.3.2-2.3.3 we have

$$\overline{\Omega}_n = \sum_{k=0}^{n} \lambda^{n-k} \overline{\mathbf{x}}_k \overline{\mathbf{x}}_k^T =$$

$$= \begin{bmatrix} \alpha_n + \mathbf{a}_n^T \mathbf{U}_{n-1}^{-T} \mathbf{D}_{n-1}^{-1} \mathbf{U}_{n-1}^{-1} \mathbf{a}_n & -\mathbf{a}_n^T \mathbf{U}_{n-1}^{-T} \mathbf{D}_{n-1}^{-1} \mathbf{U}_{n-1}^{-1} \\ -\mathbf{U}_{n-1}^{-T} \mathbf{D}_{n-1}^{-1} \mathbf{U}_{n-1}^{-1} \mathbf{a}_n & \mathbf{U}_{n-1}^{-T} \mathbf{D}_{n-1}^{-1} \mathbf{U}_{n-1}^{-1} \end{bmatrix}. \quad (2.69)$$

We are looking for a factorization $\overline{\mathbf{\Omega}}_n = \widetilde{\mathbf{U}}_n^{-T} \widetilde{\mathbf{D}}_n^{-1} \widetilde{\mathbf{U}}_n^{-1}$ with

$$\widetilde{\mathbf{U}}_n^{-1} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \boldsymbol{\psi} & \boldsymbol{\Phi} \end{bmatrix}, \qquad \widetilde{\mathbf{D}}_n^{-1} = \begin{bmatrix} \xi & \mathbf{0} \\ \mathbf{0}^T & \boldsymbol{\Theta} \end{bmatrix}, \quad (2.70)$$

where the N by N sized $\boldsymbol{\Phi}$ matrix is upper triangular with unit diagonal, $\boldsymbol{\psi}$ is a column vector of N elements, $\boldsymbol{\Theta}$ is a N by N positive diagonal matrix and ξ is a scalar. Hence:

$$\overline{\mathbf{\Omega}}_n = \begin{bmatrix} \boldsymbol{\psi}^T \boldsymbol{\Theta} \boldsymbol{\psi} + \xi & \boldsymbol{\psi}^T \boldsymbol{\Theta} \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^T \boldsymbol{\Theta} \boldsymbol{\psi} & \boldsymbol{\Phi}^T \boldsymbol{\Theta} \boldsymbol{\Phi} \end{bmatrix}. \quad (2.71)$$

By equating equations (2.71) and (2.69) we obtain:

$$\boldsymbol{\Phi} = \mathbf{U}_{n-1}^{-1}, \qquad \boldsymbol{\psi} = -\mathbf{U}_{n-1}^{-1} \mathbf{a}_n, \qquad \boldsymbol{\Theta} = \mathbf{D}_{n-1}^{-1}, \qquad \xi = \alpha_n.$$

This concludes the proof. □

A different derivation of the $\overline{\mathbf{U}}_n$ and $\overline{\mathbf{D}}_n$ matrices given in Lemma 2.3.4 and of their inverses can however be obtained, as shown in the following lemma.

Lemma 2.3.6 *Let us consider the UDU^T factorization of the inverse extended autocorrelation matrix given in (2.62). The upper triangular factorization is described as follows*

$$\overline{\mathbf{U}}_n = \begin{bmatrix} 1 & \frac{\alpha_n^{-1}}{1+\alpha_n^{-1} \mathbf{m}_n^T \mathbf{D}_{n-1}^{-1} \mathbf{m}_n} \mathbf{m}_n^T \mathbf{D}_{n-1}^{-1} \mathbf{L}_n \\ \mathbf{0} & \mathbf{U}_{n-1} \mathbf{L}_n \end{bmatrix},$$

$$\overline{\mathbf{D}}_n = \begin{bmatrix} \frac{\alpha_n^{-1}}{1+\alpha_n^{-1} \mathbf{m}_n^T \mathbf{D}_{n-1}^{-1} \mathbf{m}_n} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{W}_n \end{bmatrix}, \quad (2.72)$$

where $\mathbf{m}_n = \mathbf{U}_{n-1}^{-1} \mathbf{a}_n$. Moreover, $\mathbf{L}_n$ is an upper triangular with unit diagonal matrix and $\mathbf{W}_n$ is a positive diagonal matrix such that $\mathbf{L}_n \mathbf{W}_n \mathbf{L}_n^T = \mathbf{D}_{n-1} + \alpha_n^{-1} \mathbf{m}_n \mathbf{m}_n^T$.

Proof Using (2.49) and (2.62) we can write:

$$\overline{\mathbf{\Omega}}^{-1}(n) = \begin{bmatrix} \alpha_n^{-1} & \alpha_n^{-1} \mathbf{a}_n^T \\ \alpha_n^{-1} \mathbf{a}_n & \alpha_n^{-1} \mathbf{a}_n \mathbf{a}_n^T + \mathbf{U}_{n-1} \mathbf{D}_{n-1} \mathbf{U}_{n-1}^T \end{bmatrix}. \quad (2.73)$$

Let us define now the following partitioning of the UD factorization matrices:

$$\overline{\mathbf{U}}_n = \begin{bmatrix} 1 & \boldsymbol{\psi}^T \\ \mathbf{0} & \boldsymbol{\Phi} \end{bmatrix}, \qquad \overline{\mathbf{D}}_n = \begin{bmatrix} \xi & \mathbf{0}^T \\ \mathbf{0} & \boldsymbol{\Theta} \end{bmatrix}. \tag{2.74}$$

From (2.73) and (2.74) it turns out that

$$\xi + \boldsymbol{\psi}^T \boldsymbol{\Theta} \boldsymbol{\psi} = \alpha_n^{-1}, \qquad \boldsymbol{\Phi} \boldsymbol{\Theta} \boldsymbol{\psi} = \alpha_n^{-1} \mathbf{a}_n,$$

and

$$\boldsymbol{\Phi} \boldsymbol{\Theta} \boldsymbol{\Phi}^T = \alpha_n^{-1} \mathbf{a}_n \mathbf{a}_n^T + \mathbf{U}_{n-1} \mathbf{D}_{n-1} \mathbf{U}_{n-1}^T = \mathbf{U}_{n-1} (\mathbf{D}_{n-1} + \alpha_n^{-1} \mathbf{m}_n \mathbf{m}_n^T) \mathbf{U}_{n-1}^T,$$

where $\mathbf{m}_n = \mathbf{U}_{n-1}^{-1} \mathbf{a}_n$. Hence, introducing the factorization $\mathbf{D}_{n-1} + \alpha_n^{-1} \mathbf{m}_n \mathbf{m}_n^T = \mathbf{L}_n \mathbf{W}_n \mathbf{L}_n^T$ we immediately obtain

$$\boldsymbol{\Theta} = \mathbf{W}_n, \qquad \boldsymbol{\Phi} = \mathbf{U}_{n-1} \mathbf{L}_n$$

and, by the matrix inversion lemma, we finally obtain

$$\boldsymbol{\psi} = \alpha_n^{-1} \mathbf{W}_n^{-1} \mathbf{L}_n^{-1} \mathbf{m}_n = \alpha_n^{-1} \mathbf{L}_n^T (\mathbf{D}_{n-1} + \alpha_n^{-1} \mathbf{m}_n \mathbf{m}_n^T)^{-1} \mathbf{m}_n =$$

$$= \frac{\alpha_n^{-1}}{1 + \alpha_n^{-1} \mathbf{m}_n^T \mathbf{D}_{n-1}^{-1} \mathbf{m}_n} \mathbf{L}_n^T \mathbf{D}_{n-1}^{-1} \mathbf{m}_n$$

and

$$\xi = \frac{\alpha_n^{-1}}{1 + \alpha_n^{-1} \mathbf{m}_n^T \mathbf{D}_{n-1}^{-1} \mathbf{m}_n}$$

which concludes the derivation of (2.72). □

2.3.3 Review of Givens Rotations

Givens rotations are widely used in QR-RLS and Fast QR-RLS algorithms. Their success is due to the simplicity and the numerical robustness of the computations they perform.

We want to find a matrix rotation $\mathbf{Q}$ such that

$$\mathbf{Q} \begin{bmatrix} \xi_1 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \xi_{N+1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{2.75}$$

where $\mathbf{Q}\mathbf{Q}^T = 1$, $\xi_{N+1} = \sqrt{\sum_{i=1}^{N} x_i^2 + \xi_1^2}$ and we assume without any restriction that $\xi_1 > 0$.

Proposition 2.3.4 *The matrix* $\mathbf{Q}$ *of equation (2.75) can be decomposed into N Givens rotation matrices* $\mathbf{Q}_i$*. That is* $\mathbf{Q} = \mathbf{Q}_N \mathbf{Q}_{N-1} \dots \mathbf{Q}_1$*, where [60]*

$$\mathbf{Q}_i = \begin{bmatrix} c_i & & & & s_i & & & \\ & 1 & & & & \mathbf{0} & & \\ & & \ddots & & & & & \\ & & & 1 & & & & \\ -s_i & & & & c_i & & & \\ & & & & & 1 & & \\ & \mathbf{0} & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \leftarrow \textit{row 1} \\ \\ \\ \\ \leftarrow \textit{row } N+2-i \\ \\ \\ \\ \end{matrix} \tag{2.76}$$

and

$$c_i = \frac{\xi_i}{\xi_{i+1}}, \tag{2.77}$$

$$s_i = \frac{x_{N+1-i}}{\xi_{i+1}}, \tag{2.78}$$

$$\xi_{i+1} = \sqrt{\xi_i^2 + x_{N+1-i}^2}. \tag{2.79}$$

Proof The passage from the vector $\mathbf{x} = [\xi_1, x_1, \dots, x_N]^T$ to the pinning vector $[\xi_{N+1}, 0, \dots, 0]^T$ is obtained iteratively. At each iteration we annihilate an element x_i of $\mathbf{x}$ against the first element (the "pivot") of the same vector. On the i-th iteration the Givens rotation matrix $\mathbf{Q}_i$ has to perform the rotation:

$$\begin{bmatrix} c_i & & & & s_i & & & \\ & 1 & & & & \mathbf{0} & & \\ & & \ddots & & & & & \\ & & & 1 & & & & \\ -s_i & & & & c_i & & & \\ & & & & & 1 & & \\ & \mathbf{0} & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} \xi_i \\ x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \xi_{i+1} \\ x_1 \\ \vdots \\ x_{i-1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.80}$$

That is

$$\begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix} \cdot \begin{bmatrix} \xi_i \\ x_{N+1-i} \end{bmatrix} = \begin{bmatrix} \xi_{i+1} \\ 0 \end{bmatrix}. \tag{2.81}$$

The Proposition is proved by solving (2.81) for c_i and s_i. □

The Givens rotations are used in particular for matrix triangularization. Note, for instance, that we can triangulize the matrix in (2.40) by means of N Givens rotations. If we chose $\alpha_n^{\frac{1}{2}}$ as pivot and we annihilate the elements of $-\mathbf{z}_n$ from the last to the first against the pivot, we can set to zero all off-diagonal elements of the first column preserving the upper triangular structure of the remaining columns. For the uniqueness of the upper triangular factorization, the resulting matrix coincides with that of equation (2.28).

A similar procedure can be followed also to pass from the non upper triangular UDU^T factorization of the autocorrelation matrix in equation (2.68) to the upper triangular UDU^T factorization of equation (2.46). In this case we want to triangulize the matrix $\widetilde{\mathbf{U}}_n$ and, at the same time, we want to compute the diagonal matrix $\overline{\mathbf{D}}_n$. Again we can proceed iteratively by means of the Givens rotations. At each iteration we have to find the rotation matrix $\mathbf{Q}_i$ and the diagonal matrix $\mathbf{D} = \text{diag}[D_0, D_i]$ such that [2]

$$\mathbf{Q}_i \begin{bmatrix} \widetilde{D}_0 & 0 \\ 0 & \widetilde{D}_i \end{bmatrix}^{\frac{1}{2}} \cdot \begin{bmatrix} 1 & 0 \\ \widetilde{x}_i & 1 \end{bmatrix} = \begin{bmatrix} D_0 & 0 \\ 0 & D_i \end{bmatrix}^{\frac{1}{2}} \cdot \begin{bmatrix} 1 & \cdot \\ 0 & 1 \end{bmatrix}. \tag{2.82}$$

Alternatively, in order to avoid square-roots we can directly derive the matrix $\mathbf{D}$ and the pseudo-rotation matrix $\widehat{\mathbf{Q}}_i$ such that

$$\widehat{\mathbf{Q}}_i \cdot \begin{bmatrix} 1 & 0 \\ x_i & 1 \end{bmatrix} = \begin{bmatrix} 1 & \cdot \\ 0 & 1 \end{bmatrix} \tag{2.83}$$

and

$$\widehat{\mathbf{Q}}_i = \begin{bmatrix} D_0 & 0 \\ 0 & D_i \end{bmatrix}^{-\frac{1}{2}} \cdot \mathbf{Q}_i \cdot \begin{bmatrix} \widetilde{D}_0 & 0 \\ 0 & \widetilde{D}_i \end{bmatrix}^{\frac{1}{2}} \tag{2.84}$$

where $\mathbf{Q}_i$ is a rotation matrix, *i.e.* $\mathbf{Q}_i\mathbf{Q}_i^T = \mathbf{I}$

Proposition 2.3.5 *The matrix $\widehat{\mathbf{Q}}_i$ that solves the problem of equations (2.83) and (2.84) is*

$$\widehat{\mathbf{Q}}_i = \begin{bmatrix} \frac{\widetilde{D}_0}{D_0} & \frac{\widetilde{D}_i x_i}{D_0} \\ -x_i & 1 \end{bmatrix}, \tag{2.85}$$

where

$$D_0 = \widetilde{D}_0 + \widetilde{D}_i x_i^2 \tag{2.86}$$

[2] The $\cdot$ element is a don't care element.

and

$$D_i = \frac{\widetilde{D}_0 \widetilde{D}_i}{D_0}. \tag{2.87}$$

Moreover, the inverse transpose of matrix $\widehat{\mathbf{Q}}_i$ *is*

$$\widehat{\mathbf{Q}}_i^{-T} = \begin{bmatrix} 1 & x_i \\ -\frac{\widetilde{D}_i x_i}{D_0} & \frac{\widetilde{D}_0}{D_0} \end{bmatrix}. \tag{2.88}$$

Proof Let us consider

$$\mathbf{Q}_i = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}. \tag{2.89}$$

Form (2.84) and from the norm preserving property of square-root matrices we prove immediately equation (2.86) and we derive

$$D_i = c^2 \widetilde{D}_i. \tag{2.90}$$

From Proposition 2.3.4 it is

$$c = \frac{\widetilde{D}_0^{\frac{1}{2}}}{D_0^{\frac{1}{2}}} \tag{2.91}$$

and

$$s = \frac{\widetilde{D}_0^{\frac{1}{2}} x_i}{D_0^{\frac{1}{2}}}. \tag{2.92}$$

By substituting the expression in (2.91) in equation (2.90) we obtain equation (2.87). The expressions of the matrix $\widehat{\mathbf{Q}}_i$ and $\widehat{\mathbf{Q}}_i^{-T}$ can be derived by directly computing the products

$$\widehat{\mathbf{Q}}_i = \begin{bmatrix} D_0 & 0 \\ 0 & D_i \end{bmatrix}^{-\frac{1}{2}} \cdot \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \cdot \begin{bmatrix} \widetilde{D}_0 & 0 \\ 0 & \widetilde{D}_i \end{bmatrix}^{\frac{1}{2}} \tag{2.93}$$

and

$$\widehat{\mathbf{Q}}_i^{-T} = \begin{bmatrix} D_0^{\frac{1}{2}} & 0 \\ 0 & D_i^{\frac{1}{2}} \end{bmatrix} \cdot \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \cdot \begin{bmatrix} \widetilde{D}_0^{-\frac{1}{2}} & 0 \\ 0 & \widetilde{D}_i^{-\frac{1}{2}} \end{bmatrix}. \tag{2.94}$$

This completes the proof. □

2.4 Fast Square-Root RLS Algorithms

The use of a square-root factorization of the autocorrelation matrix is a classical approach to improve the numerical behaviour of RLS algorithms [60]. The inverse autocorrelation matrix, in this case, is factorized as described in (2.27). If the recursive equations are expressed in terms of the $\mathbf{S}_n$ matrix the positive definiteness of $\boldsymbol{\Omega}_n^{-1}$ is guaranteed. Let us now consider the vector

$$\mathbf{d}_n = \mathbf{S}_n^T \mathbf{x}_n, \tag{2.95}$$

where $\mathbf{x}_n$ is the input vector. Clearly, the positive definiteness of $\boldsymbol{\Omega}_n^{-1}$ is implicitly ensured if we recursively update $\mathbf{d}_n$. As suggested above, such updating relations can be easily obtained using the factorizations derived in Lemma 2.3.1 and Lemma 2.3.3 or Lemma 2.3.1 and Lemma 2.3.2.

2.4.1 Fast RLS Based upon Square-Root Factorization

We are now ready to derive some algorithms for performing fast and numerically stable RLS prediction and filtering.

Fast SQR-RLS Based on Identity

Theorem 2.4.1 *Given the linear filter described in (2.1), a numerically stable algorithm to perform the recursive minimization of (2.4) in $O(N)$ operations, is described in the following table, where the operation count is also shown*

ref.	*equation*	$\times$	$\div$	$\surd$
A.1	$f_{n-1}(n) = v_n + \mathbf{d}_{n-1}^T(\sqrt{\lambda}\mathbf{T}_{n-1}^{-1}\mathbf{z}_{n-1})$	$6N$	$2N$	N
A.2	$f_n(n) = \gamma_{n-1} f_{n-1}(n)$			
A.3	$\mathbf{z}_n = \sqrt{\lambda}\mathbf{T}_{n-1}^{-1}\mathbf{z}_{n-1} - \mathbf{d}_{n-1} f_{n-1}(n)$	N		
A.4	$\alpha_n = \lambda\alpha_{n-1} + f_n(n) f_{n-1}(n)$			
A.5	$\sigma_n = \lambda\sigma_{n-1} + v_n^2$			
A.6	$\overline{\mathbf{d}}_n = \begin{bmatrix} \mathbf{d}_n \\ \beta_n^{-\frac{1}{2}} b_n(n) \end{bmatrix} = \begin{bmatrix} v_n \sigma_n^{-\frac{1}{2}} \\ \mathbf{L}_n^T (v_n \sigma_n^{-1} \mathbf{z}_n + \mathbf{d}_{n-1}) \end{bmatrix}$	$5N$	$2N$	N
A.7	$\gamma_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n) + \beta_n^{-1} b_n^2(n)$			
A.8	$e_n(n) = \gamma_n e_{n-1}(n)$			
A.9	$e_{n-1}(n) = d(n) - \mathbf{d}_n^T(\sqrt{\lambda}\mathbf{T}_n^{-1}\mathbf{h}_{n-1})$	$4N$	N	
A.10	$\mathbf{h}_n = \sqrt{\lambda}\mathbf{T}_n^{-1}\mathbf{h}_{n-1} + \mathbf{d}_n e_{n-1}(n)$	N		
-	*total*	$17N$	$5N$	$2N$

In this table $\mathbf{T}_n$ *indicates an upper triangular matrix with positive diagonal such that*

$$\mathbf{T}_n^T\mathbf{T}_n = \mathbf{I} - \mathbf{d}_n\mathbf{d}_n^T, \tag{2.96}$$

and $\mathbf{L}_n$ *is an upper triangular matrix such that*

$$\mathbf{L}_n\mathbf{L}_n^T = \mathbf{I} + \alpha_n^{-1}\mathbf{z}_n\mathbf{z}_n^T. \tag{2.97}$$

Proof Equations (A.2), (A.4) and (A.8) can be found in several references, for example in [60].

Let us start with (A.1). Using the definition of $\boldsymbol{\Omega}_n$ given in (2.6) and the factorization (2.27), the following relation can be derived:

$$\mathbf{S}_n^{-T}\mathbf{S}_n^{-1} = \lambda\mathbf{S}_{n-1}^{-T}\mathbf{S}_{n-1}^{-1} + \mathbf{x}_n\mathbf{x}_n^T. \tag{2.98}$$

Let us now introduce the following factorization

$$\mathbf{I} - \mathbf{d}_n\mathbf{d}_n^T = \mathbf{T}_n^T\mathbf{T}_n \tag{2.99}$$

with $\mathbf{T}_n$ upper triangular. It is worth noting that, by inverting (2.99) and applying the matrix inversion lemma, we can write

$$\mathbf{I} + \gamma_n^{-1}\mathbf{d}_n\mathbf{d}_n^T = \mathbf{T}_n^{-1}\mathbf{T}_n^{-T}.$$

In other words, a form similar to (2.99) is obtained for the inverse factorization. Some issues concerning these forms of factorizations are described in Section 2.4.3, where efficient algorithms are also given.

Using (2.95) and (2.99), equation (2.98) becomes

$$\lambda\mathbf{S}_{n-1}^{-T}\mathbf{S}_{n-1}^{-1} = \mathbf{S}_n^{-T}\mathbf{T}_n^T\mathbf{T}_n\mathbf{S}_n^{-1}. \tag{2.100}$$

From this last equation, we can state that

$$\sqrt{\lambda}\mathbf{S}_{n-1}^{-1} = \mathbf{T}_n\mathbf{S}_n^{-1}, \tag{2.101}$$

which leads to the following expression:

$$\mathbf{S}_n^{-1} = \sqrt{\lambda}\mathbf{T}_n^{-1}\mathbf{S}_{n-1}^{-1}. \tag{2.102}$$

Moreover, from (2.7) we have

$$f_{n-1}(n) = v_n + \mathbf{x}_{n-1}^T\mathbf{a}_{n-1}. \tag{2.103}$$

Pre-multiplying $\mathbf{a}_{n-1}$ by $\mathbf{S}_{n-1}\mathbf{S}_{n-1}^{-1}$ and using (2.95), (2.102) and (2.47), the equation referenced to as (A.1) is obtained.

Since, from (2.11), $\mathbf{a}_n = \mathbf{a}_{n-1} - \mathbf{S}_{n-1}\mathbf{d}_{n-1}f_{n-1}(n)$, the $\mathbf{z}_n$ vector can be expressed as

$$\mathbf{z}_n = \mathbf{S}_{n-1}^{-1}\mathbf{a}_{n-1} - \mathbf{d}_{n-1}f_{n-1}(n) \tag{2.104}$$

and, using (2.102), this last equation becomes (A.3).

Using the definition of $\overline{\mathbf{d}}_n$ and using Lemma 2.3.1 and (2.8) we obtain

$$\overline{\mathbf{d}}_n = \overline{\mathbf{S}}_n^T \begin{bmatrix} \mathbf{x}_n \\ r_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_n \\ \beta_n^{-\frac{1}{2}} b_n(n) \end{bmatrix}. \tag{2.105}$$

By the way, from this last equation, we can see that $\mathbf{d}_n$ is the "normalized *a posteriori* backward prediction error vector" [60] whose elements are the normalized backward errors for different filter orders.

However, using Lemma 2.3.3, the $\overline{\mathbf{d}}_n$ vector can also be expressed as follows:

$$\overline{\mathbf{d}}_n = \overline{\mathbf{S}}_n^T \begin{bmatrix} v_n \\ \mathbf{x}_{n-1} \end{bmatrix} = \begin{bmatrix} v_n\sqrt{\frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}} \\ \mathbf{L}_n^T\left(\frac{\alpha_n^{-1}v_n}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}\mathbf{z}_n + \mathbf{d}_{n-1}\right) \end{bmatrix}. \tag{2.106}$$

Due to the uniqueness of the upper triangular square-root factorization, (2.105) and (2.106) are equal.

Equation (2.106) can be simplified by realizing that the first coefficient of $\mathbf{d}_n$ is the normalized backward prediction error of a zero-th order predictor or, in other words, the sample v_n itself divided by the backward error energy $\sqrt{\sigma_n}$, where

$$\sigma_n = \sum_{k=0}^{n} \lambda^{n-k} v_k^2 = \lambda\sigma_{n-1} + v_n^2.$$

which is reported in (A.5). Therefore, with (2.106) we can now state that

$$\frac{v_n}{\sqrt{\sigma_n}} = v_n\sqrt{\frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{z}_n^T\mathbf{z}_n}}.$$

Therefore, the update relation (A.6) for $\mathbf{d}_n$ and $\beta_n^{-\frac{1}{2}}b_n(n)$ is immediately obtained.

As regards the likelihood variable, moreover, from (2.10), (2.9) and (2.95) we obtain:

$$\gamma_n = 1 - \mathbf{c}_n^T\mathbf{x}_n = 1 - \mathbf{x}_n^T\boldsymbol{\Omega}_n^{-1}\mathbf{x}_n = 1 - \mathbf{d}_n^T\mathbf{d}_n. \tag{2.107}$$

Hence:

$$\overline{\gamma}_n = 1 - \overline{\mathbf{d}}_n^T\overline{\mathbf{d}}_n. \tag{2.108}$$

Using (2.105), expression (2.108) can be written as follows:

$$\overline{\gamma}_n = 1 - \mathbf{d}_n^T \mathbf{d}_n - \beta_n^{-1} b_n^2(n). \tag{2.109}$$

and, combining (2.107) and (2.109), it turns out that

$$\gamma_n = \overline{\gamma}_n + \beta_n^{-1} b_n^2(n) \tag{2.110}$$

Thus, since $\overline{\gamma}_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n)$ as reported, for instance, in [60], we get (A.7). It is important to note that (A.7) might not be sufficient to achieve numerical stability especially in particular conditions such as limited precision environment or long input sequences (in the latter case we can refer to a long term instability problem). The problem arises from a slow error accumulation effect that (A.7) introduces. One approach to overcome this problem is to monitor the value of γ_n, which must lay in the [0-1] range, and to recompute its value using relation (2.107) which is not recursive and therefore doesn't lead to the long term instability problem.

However, the error accumulation effect becomes relevant only on a long term scale; therefore, in another approach we can correct every L samples the value of γ_n by means of (2.107). In this way, the additional computational burden is reduced to L/N multiplications per input sample. For example, since (2.107) requires N multiplications per input sample, a reasonable choice can be $L = N$, such that the correction of the error accumulation effect requires only 1 multiplication for sample. However, computer simulations have shown that, on the average, the former approach is the best one. More on this topic is given in Section 2.5.

Equations (A.1) to (A.7) describe the prediction part of the algorithm. Let us now derive the filtering part, which pertains to the reconstruction error defined as

$$e_n(k) = d(k) - d_n(k) = d(k) - \mathbf{w}_n^T \mathbf{x}_k. \tag{2.111}$$

It can be easily shown that

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{S}_n \mathbf{d}_n e_{n-1}(n) \tag{2.112}$$

and

$$e_{n-1}(n) = d(n) - \mathbf{w}_{n-1}^T \mathbf{x}_n = d(n) - \mathbf{d}_n^T \mathbf{S}_n^{-1} \mathbf{w}_{n-1}. \tag{2.113}$$

By defining

$$\mathbf{h}_n = \mathbf{S}_n^{-1} \mathbf{w}_n \tag{2.114}$$

and using (2.102) and (2.26), from (2.113) we can obtain (A.9). Using (2.114) and (2.112), finally, we obtain (A.10). □

Fast SQR-RLS Based on Givens Rotations

Theorem 2.4.2 *Given the linear filter described in (2.1), a numerically stable algorithm to perform the recursive minimization of (2.4) in $O(N)$ operations, is described in the following table, where the operation count is also shown*

ref.	*equation*	$\times$	$\div$	$\surd$
B.1	$f_{n-1}(n) = v_n + \mathbf{d}_{n-1}^T(\sqrt{\lambda}\mathbf{T}_{n-1}^{-1}\mathbf{z}_{n-1})$	$6N$	$2N$	N
B.2	$f_n(n) = \gamma_{n-1} f_{n-1}(n)$			
B.3	$\mathbf{z}_n = \sqrt{\lambda}\mathbf{T}_{n-1}^{-1}\mathbf{z}_{n-1} - \mathbf{d}_{n-1} f_{n-1}(n)$	N		
B.4	$\alpha_n = \lambda\alpha_{n-1} + f_n(n) f_{n-1}(n)$			
B.5	$\prod_i \mathbf{Q}_i : \begin{bmatrix} \cdot \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \prod_i \mathbf{Q}_i \begin{bmatrix} \alpha_n^{\frac{1}{2}} \\ -\mathbf{z}_n \end{bmatrix}$	N	$2N$	N
B.6	$\begin{bmatrix} \mathbf{d}_n \\ \beta_n^{-\frac{1}{2}} b_n(n) \end{bmatrix} = \prod_i \mathbf{Q}_i \begin{bmatrix} \alpha_n^{-\frac{1}{2}} f_n(n) \\ \mathbf{d}_{n-1} \end{bmatrix}$	$4N$		
B.7	$\gamma_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n) + \beta_n^{-1} b_n^2(n)$			
B.8	$e_n(n) = \gamma_n e_{n-1}(n)$			
B.9	$e_{n-1}(n) = d(n) - \mathbf{d}_n^T(\sqrt{\lambda}\mathbf{T}_n^{-1}\mathbf{h}_{n-1})$	$4N$	N	
B.10	$\mathbf{h}_n = \sqrt{\lambda}\mathbf{T}_n^{-1}\mathbf{h}_{n-1} + \mathbf{d}_n e_{n-1}(n)$	N		
-	*total*	$17N$	$5N$	$2N$

In this table $\mathbf{T}_n$ indicates an upper triangular matrix with positive diagonal such that

$$\mathbf{T}_n^T\mathbf{T}_n = \mathbf{I} - \mathbf{d}_n\mathbf{d}_n^T. \tag{2.115}$$

Proof This algorithm differs from the previous one only in equations (B.5) and (B.6). The basis for these equations comes from the observation that the two matrices $\overline{\mathbf{S}}_n^{-1}$ and $\widetilde{\mathbf{S}}_n^{-1}$ in equations (2.29) and (2.41) differ only for a rotation matrix $\mathbf{Q}$. This rotation matrix can be decomposed in N Givens rotation matrices $\mathbf{Q}_i$ as shown in Section 2.3.3. Moreover, the identification of these rotation matrices requires only the knowledge of the first column of matrix $\widetilde{\mathbf{S}}_n^{-1}$, *i.e.* of $[\alpha_n^{\frac{1}{2}}, -\mathbf{z}_n^T]^T$. This justifies equation (B.5).

Since it is

$$\overline{\mathbf{S}}_n^{-1} = \prod_i \mathbf{Q}_i \widetilde{\mathbf{S}}_n^{-1}, \tag{2.116}$$

then it is also

$$\overline{\mathbf{S}}_n^{T} = \prod_i \mathbf{Q}_i \widetilde{\mathbf{S}}_n^{T}. \tag{2.117}$$

and

$$\overline{\mathbf{d}}_n = \overline{\mathbf{S}}_n^T \overline{\mathbf{x}}_n = \prod_i \mathbf{Q}_i \tilde{\mathbf{S}}_n^T \overline{\mathbf{x}}_n. \tag{2.118}$$

By substituting equations (2.13), (2.28), (2.40) in equation (2.118) we immediately derive (B.6). This completes the proof of the theorem. □

The following remarks apply to both the algorithms of this subsection.

Remark 2.4.1 The number of additions of the algorithms is comparable to the number of multiplications and, hence, it has not been included in the operations count.

Remark 2.4.2 The algorithms can be initialized as follows: the vectors $\mathbf{h}$ and $\mathbf{d}$ are set to zero, $\gamma = 1$, α can be set to a small positive constant and $\sigma = 0$.

Remark 2.4.3 It is important to note that the $\mathbf{h}_n$ vector is the same vector computed by QR and Lattice algorithms based on the normalized *a posteriori* prediction error vector [60]. Moreover, it is the vector used in the joint process for the estimation of the desired signal.

Remark 2.4.4 As explained in the proof of Theorem 2.4.1, the equation referred to as (A.7) or (B.7) for the computation of the likelihood variable can lead to a long term instability which can be avoided by computing the likelihood variable as $\gamma_n = 1 - \mathbf{d}_n^T \mathbf{d}_n$, when it becomes necessary. Furthermore, this correction becomes particularly necessary when the algorithm is implemented in a limited precision environment. Therefore, the step (A.7)-(B.7) of the algorithm introduced in Theorems 2.4.1 and 2.4.2 can be viewed as suitable for a standard floating point precision environment.

Remark 2.4.5 Equation (A.1)-(B.1) shows that the predicted signal is the first signal computed by the algorithm. The reader should note the direct dependency of the prediction error from the input signal v_n. Thus, equations (A.1)-(B.1) to (A.7)-(B.7) can be used in applications which require the computation of the predicted signal, such as ADPCM-like coding algorithms.

Remark 2.4.6 It is important to note that the recursive relations of Theorems 2.4.1 and 2.4.2 use many upper triangular matrix factorizations - such as the ones referred to as relations (2.48) and (2.99) - and

that the total operation count greatly depend upon these factorizations. Therefore, efficient factorization algorithms have been derived and are reported in Section 2.4.3. In fact, the operation count of (A.1)-(B.1), (A.3)-(B.3) and (A.6) reported in Theorems 2.4.1 and 2.4.2 takes into account the complexity of the factorization algorithms described in Section 2.4.3.

2.4.2 Fast RLS Based upon UD-Factorization

In this Subsection, a derivation of the fast RLS recursive relations on the basis of the UD-factorization is described. In this case, we consider the recursive updating of the

$$\mathbf{g}_n = \mathbf{U}_n^T \mathbf{x}_n \tag{2.119}$$

vector which takes the place of (2.95). The UD-RLS equations are somehow similar to the SQR-RLS ones described in Theorems 2.4.1 and 2.4.2, the main difference being the absence of square-root operations.

Fast UD-RLS Based on Identity

Theorem 2.4.3 *Given the linear filter described in (2.1), a square-roots free and numerically stable algorithm to perform the recursive minimization of (2.4) in $O(N)$ operations, is described in the following table, where the operation count is also shown*

ref.	*equation*	$\times$	$\div$
C.1	$f_{n-1}(n) = v_n + \mathbf{g}_{n-1}^T(\mathbf{T}_{n-1}^{-1}\mathbf{m}_{n-1})$	*5N*	*3N*
C.2	$f_n(n) = \gamma_{n-1} f_{n-1}(n)$		
C.3	$\mathbf{m}_n = \mathbf{T}_{n-1}^{-1}\mathbf{m}_{n-1} - \mathbf{D}_{n-1}\mathbf{g}_{n-1} f_{n-1}(n)$	*N*	
C.4	$\alpha_n = \lambda\alpha_{n-1} + f_n(n) f_{n-1}(n)$		
C.5	$\sigma_n = \lambda\sigma_{n-1} + v_n^2$		
C.6	$\begin{bmatrix} \mathbf{g}_n \\ b_n(n) \end{bmatrix} = \begin{bmatrix} v_n \\ \mathbf{L}_n^T\left(v_n\sigma_n^{-1}\mathbf{D}_{n-1}^{-1}\mathbf{m}_n + \mathbf{g}_{n-1}\right) \end{bmatrix}$	*7N*	*2N*
C.7	$\begin{bmatrix} \mathbf{D}_n & \mathbf{0} \\ \mathbf{0}^T & \beta_n^{-1} \end{bmatrix} = \begin{bmatrix} \sigma_n^{-1} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{W}_n \end{bmatrix}$		
C.8	$\gamma_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n) + \beta_n^{-1} b_n^2(n)$		
C.9	$e_n(n) = \gamma_n e_{n-1}(n)$		
C.10	$e_{n-1}(n) = d(n) - \mathbf{g}_n^T(\mathbf{T}_n^{-1}\mathbf{k}_{n-1})$	*3N*	
C.11	$\mathbf{k}_n = \mathbf{T}_n^{-1}\mathbf{k}_{n-1} + \mathbf{D}_n\mathbf{g}_n e_{n-1}(n)$	*N*	
-	*total*	*17N*	*5N*

In this table $\mathbf{T}_n$ *is an upper triangular with unit diagonal matrix such that*

$$\mathbf{T}_n^T \mathbf{V}_n \mathbf{T}_n = \mathbf{D}_n^{-1} - \mathbf{g}_n \mathbf{g}_n^T \tag{2.120}$$

and $\mathbf{V}_n$ *is a positive diagonal matrix. Moreover,* $\mathbf{L}_n$ *is an upper triangular with unit diagonal matrix and* $\mathbf{W}_n$ *is a diagonal matrix such that*

$$\mathbf{L}_n \mathbf{W}_n \mathbf{L}_n^T = \mathbf{D}_{n-1} + \alpha_n^{-1} \mathbf{m}_n \mathbf{m}_n^T. \tag{2.121}$$

Proof First, we will prove (C.1). From (2.7) and (2.119), we have

$$f_{n-1}(n) = v_n + \mathbf{g}_{n-1}^T \mathbf{U}_{n-1}^{-1} \mathbf{a}_{n-1}, \tag{2.122}$$

The UD counterpart of (2.101) is

$$\begin{aligned} \lambda \mathbf{U}_{n-1}^{-T} \mathbf{D}_{n-1}^{-1} \mathbf{U}_{n-1}^{-1} &= \mathbf{U}_n^{-T} \mathbf{D}_n^{-1} \mathbf{U}_n^{-1} - \mathbf{x}_n \mathbf{x}_n^T = \\ &= \mathbf{U}_n^{-T} (\mathbf{D}_n^{-1} - \mathbf{g}_n \mathbf{g}_n^T) \mathbf{U}_n^{-1}. \end{aligned} \tag{2.123}$$

The expression between brackets can be factorized as

$$\mathbf{T}_n^T \mathbf{V}_n \mathbf{T}_n = \mathbf{D}_n^{-1} - \mathbf{g}_n \mathbf{g}_n^T, \tag{2.124}$$

where $\mathbf{T}_n$ is an upper triangular with unit diagonal matrix and $\mathbf{V}_n$ is a positive diagonal matrix. From (2.123) it evinces that $\mathbf{U}_{n-1}^{-1} = \mathbf{T}_n \mathbf{U}_n^{-1}$ and $\lambda \mathbf{D}_{n-1}^{-1} = \mathbf{V}_n$. Hence

$$\mathbf{U}_n^{-1} = \mathbf{T}_n^{-1} \mathbf{U}_{n-1}^{-1}. \tag{2.125}$$

Combining (2.125) and (2.122) we obtain (C.1).

This last equation leads to the problem of computing the factorization matrix $\mathbf{T}_n^{-1}$. This problem, which is similar to the one introduced in Theorem 2.4.1 with relation (2.99), is addressed in the next section. For now, let us just note that, using the matrix inversion lemma, the inversion of (2.124) leads to

$$\mathbf{T}_n^{-1} \mathbf{V}_n^{-1} \mathbf{T}_n^{-T} = \mathbf{D}_n + \gamma_n^{-1} (\mathbf{D}_n \mathbf{g}_n)(\mathbf{g}_n^T \mathbf{D}_n). \tag{2.126}$$

Equation (C.3) can be easily obtained considering that $\mathbf{m}_n = \mathbf{U}_{n-1}^{-1} \mathbf{a}_n$ and recalling that, from (2.11), (2.9) and (2.62):

$$\mathbf{a}_n = \mathbf{a}_{n-1} - \mathbf{U}_{n-1} \mathbf{D}_{n-1} \mathbf{U}_{n-1}^T \mathbf{x}_{n-1} f_{n-1}(n). \tag{2.127}$$

As in Theorem 2.4.1, the updating equations for the $\mathbf{g}_n$ vector is obtained by equating the factorization forms derived in Lemma 2.3.4 and Lemma 2.3.6. In fact, from Lemma 2.3.4 we have that

$$\overline{P}_n = \begin{bmatrix} \mathbf{U}_n^T & \mathbf{0} \\ \mathbf{b}_n^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_n \\ r_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_n \\ \mathbf{b}_n^T \mathbf{x}_n + r_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_n \\ b_n(n) \end{bmatrix}. \tag{2.128}$$

On the other hand, from (2.13) and Lemma 2.3.6 we obtain:

$$\overline{P}_n = \begin{bmatrix} 1 & \mathbf{0}^T \\ \frac{\alpha_n^{-1}\mathbf{L}_n^T\mathbf{D}_{n-1}^{-1}\mathbf{m}_n}{1+\alpha_n^{-1}\mathbf{m}_n^T\mathbf{D}_{n-1}^{-1}\mathbf{m}_n} & \mathbf{L}_n^T\mathbf{U}_{n-1}^T \end{bmatrix} \begin{bmatrix} v_n \\ \mathbf{x}_{n-1} \end{bmatrix} =$$

$$= \begin{bmatrix} v_n \\ \mathbf{L}_n^T \left[\frac{\alpha_n^{-1} v_n \mathbf{D}_{n-1}^{-1}\mathbf{m}_n}{1+\alpha_n^{-1}\mathbf{m}_n^T\mathbf{D}_{n-1}^{-1}\mathbf{m}_n} + \mathbf{g}_{n-1} \right) \end{bmatrix}. \tag{2.129}$$

From (2.128) we have that $\mathbf{g}_n$ is the backward prediction error vector. Moreover from Lemma 2.3.4 and Lemma 2.3.6 we have:

$$\overline{\mathbf{D}}_n = \begin{bmatrix} \mathbf{D}_n & \mathbf{0} \\ \mathbf{0}^T & \beta_n^{-1} \end{bmatrix} = \begin{bmatrix} \frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{m}_n^T\mathbf{D}_{n-1}^{-1}\mathbf{m}_n} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{W}_n \end{bmatrix}. \tag{2.130}$$

It is worth recalling from Lemma 2.3.5 that

$$\mathbf{D}_{n-1} + \alpha_n^{-1}\mathbf{m}_n\mathbf{m}_n^T = \mathbf{L}_n\mathbf{W}_n\mathbf{L}_n^T. \tag{2.131}$$

The above expressions can be simplified if we realize that the diagonal of $\mathbf{D}_n^{-1}$ is the backward error energy and then

$$\frac{\alpha_n^{-1}}{1+\alpha_n^{-1}\mathbf{m}_n^T\mathbf{D}_{n-1}^{-1}\mathbf{m}_n} = \sigma_n^{-1}, \tag{2.132}$$

where σ_n is the signal energy computed with (C.5).

By the uniqueness of the UD-factorization, the extended vectors in (2.128) and (2.129) are equal. From this equality, and with (2.130), we obtain the updating relations for $\mathbf{D}_n$, $\mathbf{g}_n$ and $b_n(n)$ reported in (C.6) and (C.7).

Clearly, the same error accumulation problem introduced by (C.8) takes place in this algorithm. Also in this case we can monitor the likelihood variable and recompute its value, if necessary, with the UD-counterpart of (2.107) which is $\gamma_n = 1 - \mathbf{g}_n^T\mathbf{D}_n\mathbf{g}_n$.

The filtering error, finally, can be computed as $e_n(n) = \gamma_n e_{n-1}(n)$ where

$$e_{n-1}(n) = d(n) - \mathbf{x}_n^T\mathbf{w}_{n-1} = d(n) - \mathbf{g}_n^T\mathbf{U}_n^{-1}\mathbf{w}_{n-1}. \tag{2.133}$$

By defining $\mathbf{k}_n = \mathbf{U}_n^{-1}\mathbf{w}_n$, (2.133) reduces to (C.10). On the other hand, since

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{U}_n\mathbf{D}_n\mathbf{U}_n^T\mathbf{x}_n e_{n-1}(n), \tag{2.134}$$

equation (C.11) is immediately obtained. □

Fast UD-RLS based on Givens rotations

Theorem 2.4.4 *Given the linear filter described in (2.1), a square-roots free and numerically stable algorithm to perform the recursive minimization of (2.4) in $O(N)$ operations, is described in the following table, where the operation count is also shown*

ref.	*equation*	$\times$	$\div$
D.1	$f_{n-1}(n) = v_n + \mathbf{g}_{n-1}^T(\mathbf{T}_{n-1}^{-1}\mathbf{m}_{n-1})$	*5N*	*3N*
D.2	$f_n(n) = \gamma_{n-1} f_{n-1}(n)$		
D.3	$\mathbf{m}_n = \mathbf{T}_{n-1}^{-1}\mathbf{m}_{n-1} - \mathbf{D}_{n-1}\mathbf{g}_{n-1} f_{n-1}(n)$	*N*	
D.4	$\alpha_n = \lambda\alpha_{n-1} + f_n(n) f_{n-1}(n)$		
D.5	$\prod_i \widehat{\mathbf{Q}}_i : \begin{bmatrix} \cdot \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \prod_i \widehat{\mathbf{Q}}_i \begin{bmatrix} 1 \\ -\mathbf{z}_n \end{bmatrix}$	*2N*	*2N*
D.6	$\begin{bmatrix} \alpha_n^{-1} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{D}_{n-1} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{D}_n & \mathbf{0} \\ \mathbf{0}^T & \beta_n^{-1} \end{bmatrix}$	*N*	
D.7	$\begin{bmatrix} \mathbf{g}_n \\ b_n(n) \end{bmatrix} = \prod_i \widehat{\mathbf{Q}}_i^{-1} \begin{bmatrix} f_n(n) \\ \mathbf{g}_{n-1} \end{bmatrix}$	*3N*	
D.8	$\gamma_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n) + \beta_n^{-1} b_n^2(n)$		
D.9	$e_n(n) = \gamma_n e_{n-1}(n)$		
D.10	$e_{n-1}(n) = d(n) - \mathbf{g}_n^T(\mathbf{T}_n^{-1}\mathbf{k}_{n-1})$	*3N*	
D.11	$\mathbf{k}_n = \mathbf{T}_n^{-1}\mathbf{k}_{n-1} + \mathbf{D}_n\mathbf{g}_n e_{n-1}(n)$	*N*	
-	*total*	*16N*	*5N*

In this table $\mathbf{T}_n$ is an upper triangular with unit diagonal matrix such that

$$\mathbf{T}_n^T\mathbf{V}_n\mathbf{T}_n = \mathbf{D}_n^{-1} - \mathbf{g}_n\mathbf{g}_n^T \tag{2.135}$$

and $\mathbf{V}_n$ is a positive diagonal matrix.

Proof This algorithm differs from the previous one only in equations (D.5)-(D.7).

Note that both the matrices $\overline{\mathbf{D}}_N^{-\frac{1}{2}}\overline{\mathbf{U}}_N^{-1}$ and $\widetilde{\mathbf{D}}_N^{-\frac{1}{2}}\widetilde{\mathbf{U}}_N^{-1}$ provide a square-root factorization of the extended autocorrelation matrix. This means that $\overline{\mathbf{D}}_N^{-\frac{1}{2}}\overline{\mathbf{U}}_N^{-1}$ and $\widetilde{\mathbf{D}}_N^{-\frac{1}{2}}\widetilde{\mathbf{U}}_N^{-1}$ differ for a rotation matrix $\mathbf{Q}$. For the particular structure of this matrices, the rotation matrix $\mathbf{Q}$ can be decomposed in N Givens rotation matrices. As we proved in Subsection 2.3.3 with simple manipulations and with the introduction of the pseudo-rotation matrices $\widehat{\mathbf{Q}}_i$ we can easily compute $\overline{\mathbf{D}}_N^{-1}$ and $\overline{\mathbf{U}}_N^{-1}$ from $\widetilde{\mathbf{D}}_N^{-1}$ and $\widetilde{\mathbf{U}}_N^{-1}$ without the use of square-roots. Moreover, the computation of $\overline{\mathbf{D}}_N^{-1}$ and

$\overline{\mathbf{U}}_N^{-1}$ requires the knowledge of the diagonal matrix $\widetilde{\mathbf{D}}_N^{-1}$ and of the only first column of $\widetilde{\mathbf{U}}_N^{-1}$, *i.e.* $[1, -\mathbf{m}_n^T]^T$. This justifies the computations of equations (D.5) and (D.6). Furthermore, since it is

$$\overline{\mathbf{U}}_n^{-1} = \prod_i \widehat{\mathbf{Q}}_i \widetilde{\mathbf{U}}_n^{-1}, \tag{2.136}$$

we have that

$$\overline{\mathbf{U}}_n^T = \prod_i \widehat{\mathbf{Q}}_i^{-T} \widetilde{\mathbf{U}}_n^T \tag{2.137}$$

and also

$$\overline{\mathbf{g}}_n = \overline{\mathbf{U}}_n^T \overline{\mathbf{x}}_n = \prod_i \widehat{\mathbf{Q}}_i^{-T} \widetilde{\mathbf{U}}_n^T \overline{\mathbf{x}}_n \tag{2.138}$$

By substituting equations (2.13), (2.64), (2.68) in (2.138) we obtain (D.7). □

The following remark apply to both the algorithms of this subsection.

Remark 2.4.7 Basically, the same observations already remarked at the end of Theorems 2.4.1 and 2.4.2 can be repeated at this point. First, also in this case, the joint process coincides with that of QR and Lattice algorithms based on *a posteriori* backward prediction error vector.

Moreover, it is worth noting that also in this case the efficient algorithms described in Section 2.4.3 play a fundamental role in limiting the operation count.

Finally, we can note that the recursive algorithms described in Theorems 2.4.3 and 2.4.4 require a number of multiplications and division that is equal or lower than that of the algorithms of Theorems 2.4.1 and 2.4.2. However they do not require square-root operations at all.

2.4.3 Efficient Factorization Algorithms

As it has been shown in the previous sections, $n \times n$ matrices of the form

$$\boldsymbol{\Pi}_n = \mathbf{I} + c\mathbf{d}_n\mathbf{d}_n^T, \tag{2.139}$$

where c is a positive constant and $\mathbf{d}_n$ is a n-dimensional vector, are widely used in the adaptive filtering algorithms. Thus, the determination of square-root factorization matrices of $\boldsymbol{\Pi}_n$, namely $\boldsymbol{\Pi}_n = \boldsymbol{\Gamma}_n\boldsymbol{\Gamma}_n^T$ with $\boldsymbol{\Gamma}$ upper triangular, plays a fundamental role. In order to avoid confusion, the reader should note that, in this section, the subscripts 'n' indicate the matrix (or vector) dimension.

As shown in (A.1)-(B.1)-(C.1)-(D.1), (A.6)-(C.6) and (A.9)-(B.9)-(C.10)-(D.10), rather than the computation of $\boldsymbol{\Gamma}$, we need to compute the $\boldsymbol{\Gamma}^T\mathbf{x}$ and $\boldsymbol{\Gamma}\mathbf{x}$ products, where $\mathbf{x}$ is a given vector.

According to the Agee-Turner algorithm [12, 60], consider the quadratic form $\mathbf{x}^T\boldsymbol{\Pi}_n\mathbf{x}$. The following relation can thus be obtained:

$$\mathbf{x}_n^T\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T\mathbf{x}_n = \mathbf{x}_n^T(\mathbf{I}_n + c_n\mathbf{d}_n\mathbf{d}_n^T)\mathbf{x}_n =$$

$$= y_n^2 + [\mathbf{x}_{n-1}^T\left(\mathbf{I}_{n-1} + c_{n-1}\mathbf{d}_{n-1}\mathbf{d}_{n-1}^T\right)\mathbf{x}_{n-1}], \tag{2.140}$$

where $\mathbf{I}_n$ is a $n \times n$ identity matrix, $\mathbf{x}_n$ and $\mathbf{d}_n$ are n-dimensional vectors:

$$\mathbf{x}_n^T = [\ x_1 \quad x_2 \quad \dots \quad x_n\ ],$$

$$\mathbf{d}_n^T = [\ d_1 \quad d_2 \quad \dots \quad d_n\ ],$$

and $c_n = c$, $c_{n-1} = \frac{c_n}{e_n}$ where $e_n = 1 + c_n d_n^2$. Moreover

$$y_n = e_n^{-\frac{1}{2}}\left[x_n + c_n d_n \sum_{i=1}^{n} x_i d_i\right]. \tag{2.141}$$

Therefore, if we recursively apply (2.140), we obtain that

$$\mathbf{x}_n^T\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T\mathbf{x}_n = y_n^2 + y_{n-1}^2 + \ldots + y_1^2,$$

where y_i is given by (2.141). If we define $\mathbf{v}= \boldsymbol{\Gamma}^T\mathbf{x} = [v_1 v_2 \ldots v_n]$, hence, from (2.140) we have $v_i = y_i$.

As regards the $\boldsymbol{\Gamma}\mathbf{x}$ product it can be noted from (2.141) that the $\boldsymbol{\Gamma}$ matrix is given by the sum of a upper triangular matrix, whose generic element is given by $T(i,j) = d_i d_j c_j e_j^{-\frac{1}{2}}$ and a diagonal matrix whose diagonal elements are

$$[\ e_1^{-\frac{1}{2}} \quad e_2^{-\frac{1}{2}} \quad \dots \quad e_n^{-\frac{1}{2}}\ ].$$

Therefore, the product $\mathbf{u}= \boldsymbol{\Gamma}\mathbf{x} = [u_1 u_2 \ldots u_n]$ is given by:

$$u_i = e_i^{-\frac{1}{2}} x_i + d_i \sum_{k=i}^{n} c_k d_k e_k^{-\frac{1}{2}} x_k.$$

The above results can be summarized in the following two algorithms.

Algorithm 2.4.1 *Let us consider the following square-root factorization:* $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T = \mathbf{I} + c\mathbf{d}\mathbf{d}^T$ *where* $\boldsymbol{\Gamma}$ *is upper triangular,* c *a positive constant and* $\mathbf{d}$ *is a given* N *dimensional vector, whose generic element is* d_i*. Furthermore, let* $\mathbf{x}$ *be a given* N *dimensional vector, and* x_i *its generic element. An efficient algorithm for the* $\mathbf{v}= \boldsymbol{\Gamma}^T\mathbf{x}$ *product which requires* $4N$ *multiplications,* $2N$ *divisions and* N *square-roots, is the following:*

Initialize $c_N = c$ *and* $z_0 = 0$*;*
For i=1 to N
$z_i = z_{i-1} + x_i d_i$
End For;
For i=N Downto 1
$e_i = 1 + (c_i d_i) d_i$*;*
$v_i = e_i^{-\frac{1}{2}} [x_i + (c_i d_i) z_i]$*;*
$c_{i-1} = \frac{c_i}{e_i}$*;*
End For

Algorithm 2.4.2 *In the same conditions as the previous Lemma, the* $\mathbf{u} = \boldsymbol{\Gamma}\mathbf{x}$ *product can be computed using* $4N$ *multiplications,* $2N$ *divisions and* N *square-roots, with the following algorithm:*

$z_{N+1} = 0$*;* $c_N = c$*;*
For i=N Downto 1
$e_i = 1 + (c_i d_i) d_i$*;*
$z_i = (c_i d_i)(x_i e_i^{-\frac{1}{2}}) + z_{i+1}$*;*
$c_{i-1} = \frac{c_i}{e_i}$*;*
$u_i = x_i e_i^{-\frac{1}{2}} + d_i z_i$*;*
End For;

Let us now extend the last results to the UD factorization

$$\boldsymbol{\Phi} + c\mathbf{d}\mathbf{d}^T = \boldsymbol{\Gamma}\boldsymbol{\Theta}\boldsymbol{\Gamma}^T, \tag{2.142}$$

where $\boldsymbol{\Gamma}$ is an upper triangular matrix with unit diagonal, $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ are diagonal matrices, $\mathbf{d}$ is a vector and c is a positive number. Using similar developments as above, one can derive the results described in the following two Algorithms.

Algorithm 2.4.3 *The efficient computation (*$6N$ *multiplications and* N *divisions) of* $\boldsymbol{\Theta}$ *and of the product* $\mathbf{v} = \boldsymbol{\Gamma}^T\mathbf{x}$*, where* $\boldsymbol{\Gamma}$ *and* $\boldsymbol{\Theta}$ *are the UD factorization matrices shown in (2.142) and* $\mathbf{d}$*,* $\mathbf{x}$ *are given vectors, can be performed as follows:*

Initialize $c_N = c$ *and* $z_0 = 0$*;*
For i=1 to N
$z_i = z_{i-1} + x_i d_i$

End For;
For i=N Downto 1
$\epsilon_i = \phi_i + (c_i d_i) d_i;$
$v_i = x_i + \frac{c_i}{\epsilon_i} d_i z_{i-1}$
$c_{i-1} = \frac{c_i}{\epsilon_i} \phi_i;$
End For

Algorithm 2.4.4 *The efficient computation (3N multiplications and 3N divisions) of the product* $\mathbf{u} = \boldsymbol{\Gamma}\mathbf{x}$ *on its own, where* $\boldsymbol{\Gamma}$ *is the UD factorization matrix described in (2.142) and* $\mathbf{d}$, $\mathbf{x}$ *are given vectors, is described by the following pseudocode:*

Initialize $c_N = c$ *and* $z_{N+1} = 0;$
For i=N Downto 1
$\sigma_i = \frac{\phi_i}{c_i} + d_i^2;$
$c_{i-1} = \frac{\phi_i}{\sigma_i};$
$z_i = x_i \frac{d_i}{\sigma_i} + z_{i+1};$
$u_i = x_i + d_i z_{i+1}$
End For

Remark 2.4.8 In this section, we have described efficient algorithms for the determination of upper-triangular factorization of matrices of the form $\mathbf{I} + c\mathbf{d}\mathbf{d}^T$ inspired to the Agee-Turner factorization algorithm [12]. With respect to the Agee-Turner algorithm, however, the c constant is always positive, thus leading to more stable factorization algorithms.

Remark 2.4.9 It is worth noting that rearranging the expressions of the algorithms described in this section, we can somehow trade divisions with multiplications. Since the operation count of the fast adaptive algorithms heavily depends on the operations required by these factorization algorithms, we can simply obtain slightly different derivations with a different number of products and divisions. This can be very useful from an implementation point of view.

2.5 Computer Simulations

The algorithms described above have been implemented and widely tested in different experimental conditions. In order to assess their numerical performances, some results are described in this section. The simulations reported here were performed using the same conditions as described in

[111]. Namely, we performed an identification of a 3rd-order FIR filter described by the following difference equation:

$$z(n) = 2x(n) + x(n-1) - 0.5x(n-2), \tag{2.143}$$

where the input $x(n)$ was a non-white noise sequence generated by means of the equation

$$x(n) = 0.9x(n-1) + u(n), \tag{2.144}$$

where $u(n)$ is a white unit-variance Gaussian noise. An error signal which was a white unit variance Gaussian noise was also added. All the simulations presented in this section were performed using limited precision floating point arithmetic, which was implemented, as in [110, 111], by performing each arithmetic operation to the natural 32 bits floating point precision of the computer (sign + 24 bit mantissa + 8 bit exponent) and then immediately rounding the mantissa value of the result to reflect the required simulated precision. Only the number of bits in the mantissa are affected in the experiments, and the number of bits in the exponent is fixed at eight.

Figure 2.1 illustrates the initial convergence behaviour of the two fast algorithms described in Theorem 2.4.1 and Theorem 2.4.3 at different wordlengths, namely with 4, 8 and 16 bits. The two algorithms appear to perform quite similarly. The results shown in Figure 2.1 are ensemble averages taken over 1000 independent realizations of the experiment. Figure 2.2 illustrates the initial convergence behaviour of the two fast algorithms, described in Theorem 2.4.2 and Theorem 2.4.4, in the same experimental conditions. By comparing Figures 2.1 and 2.2 we see that the Fast SQR-RLS or UD-RLS algorithms of Theorems 2.4.1 and 2.4.3 are computationally more robust than the algorithms of Theorems 2.4.2 and 2.4.4 in a low mantissa precision environment. The two couples of algorithms differ only in the update of vector $\mathbf{d}_n$ or $\mathbf{g}_n$. Thus, the vector $\mathbf{d}_n$ ($\mathbf{g}_n$) update based on the identity of two square-root autocorrelation matrices is more robust than the update based on the passage between two square-root factorizations of the autocorrelation matrix realized by means of Givens rotations. However, even with a 4 bit mantissa precision all four algorithms are long term numerically stable.

In Figure 2.3 we describe the long term (0.5 million samples) *a priori* standard deviation of the identification error obtained with the UD algorithms described in Theorem 2.4.3 and plotted for two quantization levels, namely 4 and 16 bits. These two levels were chosen for the following reasons: at 16 bit mantissa the results are comparable to floating point while at 4 bit the possible error accumulation effects appear sooner.

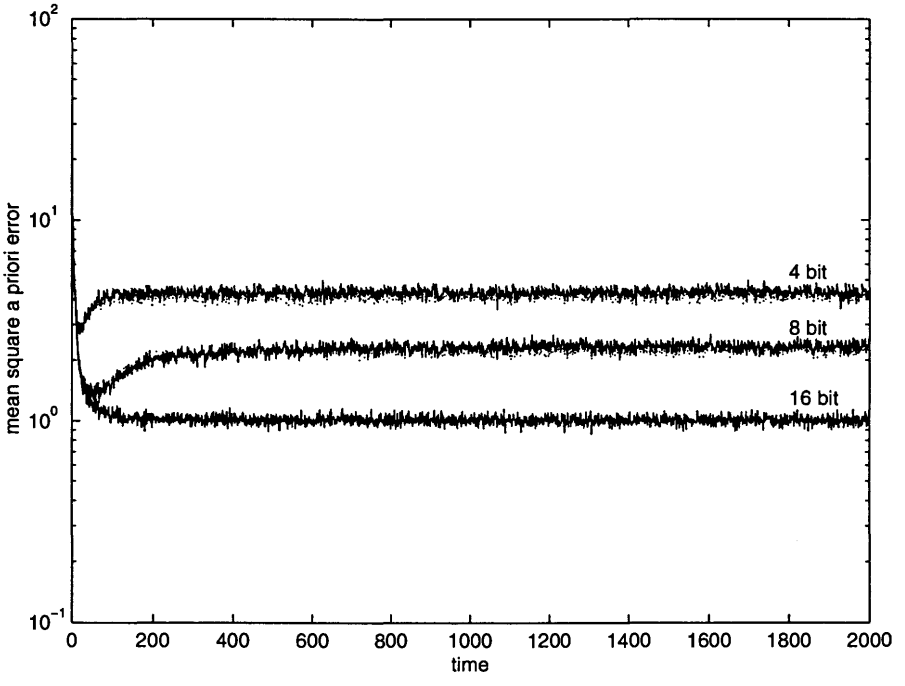


Figure 2.1: Initial convergence behaviour of the algorithms of Theorems 2.4.1 and 2.4.3 using 4, 8 and 16 bit mantissa. The square-root and UD algorithms are represented by solid and dotted lines respectively.

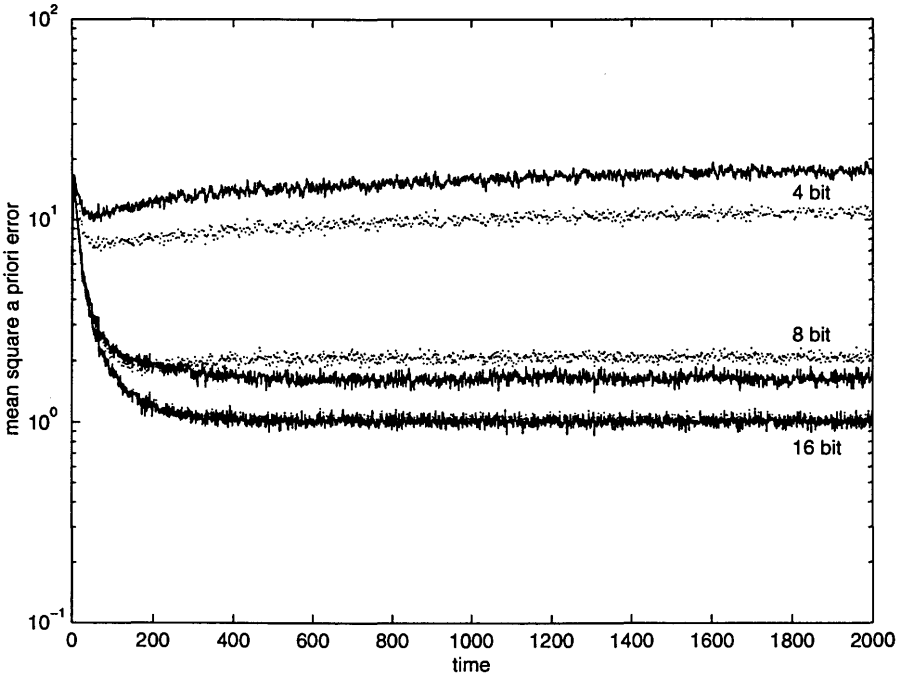


Figure 2.2: Initial convergence behaviour of the algorithms of Theorems 2.4.2 and 2.4.4 using 4, 8 and 16 bit mantissa. The square-root and UD algorithms are represented by solid and dotted lines respectively.

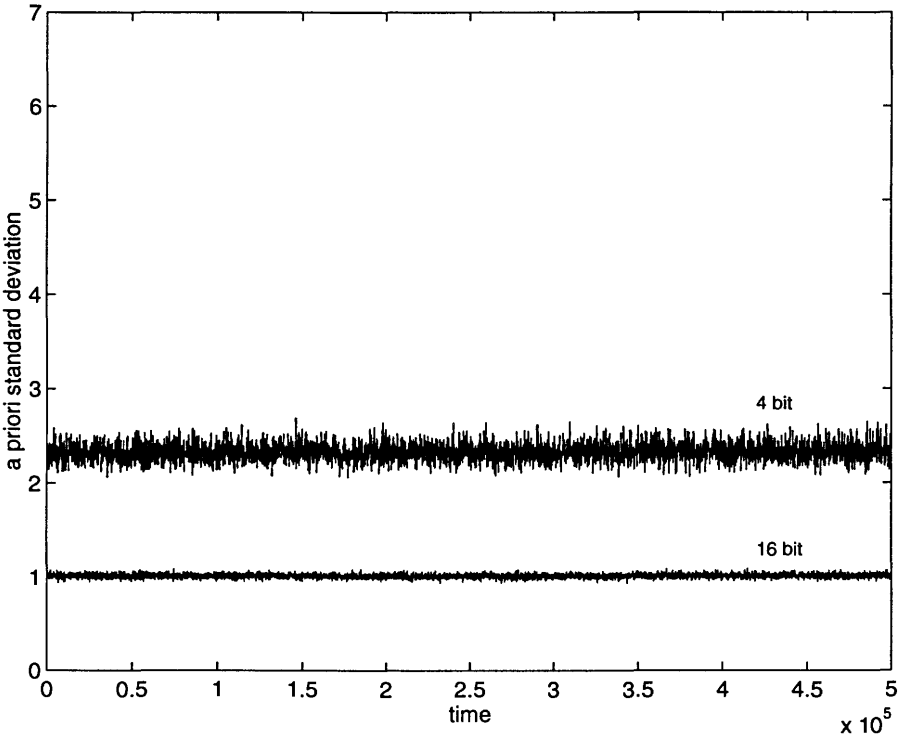


Figure 2.3: Long term (0.5 million samples) *a priori* standard deviation of the identification error for the UD algorithm described in Theorem 2.4.3.

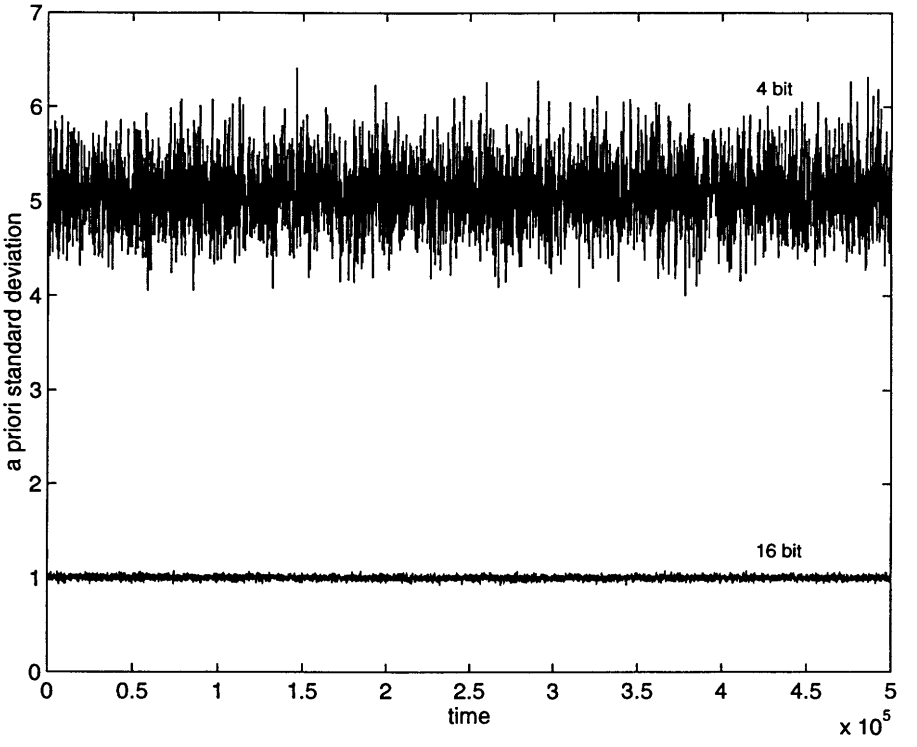


Figure 2.4: Long term (0.5 million samples) *a priori* standard deviation of the identification error for the SFTF algorithm reported in [135].

In Figure 2.4, the long term behaviour of the SFTF algorithm [135] is shown for comparison. The results shown in Figure 2.3 and Figure 2.4 are ensemble averages taken over 10 independent realizations of the experiment. The standard deviation was computed over blocks of 100 samples.

The first general observation is that, as it can be seen from Figure 2.3 and Figure 2.4, reducing the wordlength from 16 to 4 bits the variance of the identification error is increased, obviously because the roundoff noise propagation increases. Moreover, at 16 bits the behaviour of the two algorithms is identical. However, the UD-based fast algorithm described in Figure 2.4 is much more robust and much less biased than STFT at 4 bit precision, as the mean and variance are smaller. Therefore, the algorithms described in Theorems 2.4.1 and 2.4.3 appear suitable for a limited precision environment.

2.6 Final Remarks and Conclusions

Some fast square-root type algorithms which can be used in system identification applications as well as for adaptive prediction and filtering have been described in this dissertation These algorithms compute the filter coefficients in the form of a lattice filter based on the normalized or unnormalized a posteriori backward prediction error. The dual algorithms based on the a priori backward prediction error vector have been derived and they have computational and numerical properties that are very similar to those of the algorithms presented in this chapter. For this reasons they have not been included in the thesis.

The computational complexity of the algorithms presented in this chapter is $O(N)$, where N is the filter order. The fast RLS algorithms are numerically robust, and over 4 million samples long sequences have been analyzed with 4 bit mantissa without noticing any instability while maintaining notable accuracy. The algorithms have similar performances and a slight difference in computational complexity. A comparison with the algorithm described in [135] has been performed and a better accuracy in very low precision environment has been shown. Moreover, the numerical performances of the algorithms have been experimentally verified for several types of different signals such as speech and noise data. An application of the algorithm described in Theorem 2.4.2 for ADPCM coding of speech with RLS prediction has been developed on a DSP processor. These algorithms are particularly suited for such application because of the direct dependency of the forward prediction error from the input data v_n. To the authors knowledge, the only other RLS numerically stable algorithm that presents this property is the SFTF filter of [135]. Unfortunately this algorithm is numerically stable only for stationary input data. Input signals with fast changing statistic, like the human voice, can easily drive the adaptive SFTF filter into instability. On the contrary, the algorithms presented in this thesis maintains the numerical stability even in the presence of highly unstationary signals like the human voice.

Chapter 3

V-vector Algebra and its Application to Volterra Adaptive Filtering

3.1 Introduction

Adaptive Volterra filters are gaining importance both in signal processing theory and applications [92, 127, 132]. However, in spite of the numerous recent literature available on this topic several issues, such as the development of fast and numerically stable adaptive algorithms, need further research results.

Generally, adaptive algorithms for Volterra filters are obtained by extending classical algorithms for linear filters with a multichannel approach [85, 92, 121, 147, 146]. In the multichannel approach, the Volterra filter is realized by means of a linear filter bank, where each filter processes a product of samples of the input signal. The extension of the techniques for linear filters is straightforward in most of "slow" adaptive algorithms: we do refer to classical RLS, SQR-RLS, QR and Inverse QR algorithms [4, 60, 91] where the extension is achieved by simply substituting the linear filter input data vector with the corresponding Volterra filter input data vector. However, fast adaptive algorithms such as Fast RLS, FTF, Lattice RLS, Lattice QR, etc. [9, 23, 29, 31, 60, 89, 90,

---

Part of the content of this chapter was presented in
Alberto Carini and Enzo Mumolo, "A Novel Algebraic Formulation for the Development of Adaptive Volterra Filtering Algorithms," *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, June 20-22 1995, Neos Marmaras, Halkidiki, Greece, pp. 943-946
and submitted to *IEEE Trans. on Circuits and Systems II* in 1996

110, 115, 135, 149], use *ad hoc* studied derivations in order to achieve a fast implementation; these derivations generally can not be trivially extended to the Volterra filter. For example, Fast RLS and FTF make use of the concept of extended input data vector, which in the linear case is unique. However, in the non-linear case two different augmented input data vectors have to be defined [85]. Moreover, linear lattice algorithms use a filter order recursion for fast updating but with the multichannel Volterra formulation a contemporary filter order and channel order recursion has to be considered [147].

What makes it difficult to extend to the Volterra case the algorithms for linear filters is the loss of the time shift property in the input data vector. In the linear case, in order to pass form the input data vector at time n to that for time $n+1$ we have to discard the last element of the vector and we have to add the novel input at the beginning of the vector. This property does not apply to the input data vector of Volterra filters, which is constituted by different products of input samples. In this chapter we propose a novel approach that preserves the time shift property of linear data vectors. We derive a novel algebraic structure, called *V-vector algebra.* The V-vector algebra is a simple formalism which is suitable for the development of Volterra adaptive filter algorithms as an extension of linear adaptive techniques. In particular, the vectors of linear algebra are here substituted by a novel entity, the V-vector, which can be viewed as a non rectangular matrix. By the use of the V-vector formalism fast and numerically stable adaptive Volterra filtering algorithms can be easily derived from the known linear theory. Moreover, V-vector algebra can be applied also to the development of multichannel linear adaptive filters with channels of different memory lengths.

The novel algebra described in this chapter opens to new results and developments in the area of nonlinear adaptive filtering. In order to illustrate the merits of such a formalism, we first use V-vector algebra to reformulate the Lee-Mathews Fast RLS algorithm described in [85]. Thereafter, a new fast and stable Givens Rotation Based Square-Root RLS algorithm is worked out. This algorithm is the extension to the Volterra filter of the algorithm described in Theorem 2.4.2. The Fast RLS algorithm in [85] has a rather low computational complexity (in case of a second order homogeneous Volterra filter it requires $O(6N_2N_T)$ multiplications per sample, where N_2 is the memory length of the filter and N_T is the global number of coefficients of Volterra filter) but stability problems may occur. On the contrary, the fast SQR algorithm presented in this chapter shows very good stability properties, even with a modest word length precision, at the expense of a slightly higher computational

complexity ($(10 + \frac{1}{3})N_2N_T$ multiplications, $(3 + \frac{1}{3})N_2N_T$ divisions, $(1 + \frac{2}{3})N_2N_T$ square-roots).

In Section 3.2, after illustrating the motivations for the development of this novel algebraic structure, the V-vector algebra itself is presented with regard to a second order homogeneous Volterra filter: we first introduce the concepts of V-vector and V-matrix (which are the equivalent of the vector and the matrix of linear algebra), then the basic operations between V-vectors and V-matrices are defined; finally the linear algebra concepts of the inverse, transposed and triangular matrices are adapted to the V-vector algebra. In Section 3.3, the V-vectors for Volterra filters of any order and for multichannel linear filters are presented and, in particular, a recursive rule for the development of an N-th order homogeneous input data V-vector is given. In Section 3.4, the reformulation of the Lee-Mathews Fast RLS algorithm is worked out, while in Section 3.5 the novel Fast SQR RLS algorithm is developed. Conclusions follow in Section 3.6.

3.2 The V-vector Algebra

Let us first introduce the notation used in this chapter and some basic definitions. Vectors and V-vectors will be indicated with bold lower case letters, while matrices and V-matrices will be labelled with bold capital letters. A linear filter is defined by a N-th order coefficient vector $\mathbf{w}$. The input data vector of a linear filter is defined as a vector:

$$\mathbf{x}_n = \left[\; x(n), \;\; x(n-1), \;\; \ldots, \;\; x(n-N+1) \; \right]^T \tag{3.1}$$

such that the filter output signal is $y(n) = \mathbf{w}^T\mathbf{x}_n$. Note the time shift property of the input data vector for linear filters: at the time n the element $x(n)$ is added to the input data vector $\mathbf{x}_{n-1}$ while the element $x(n-N)$ is discarded. Many fast RLS adaptive algorithms use the notion of augmented or extended data vectors [29, 60, 85, 135]. The extended input data vector $\overline{\mathbf{x}}_n$ is defined as the vector obtained adding $x(n)$ to the top of $\mathbf{x}_{n-1}$ or adding $x(n-N)$ to the bottom of $\mathbf{x}_n$:

$$\overline{\mathbf{x}}_n = \begin{bmatrix} x(n) \\ \mathbf{x}_{n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_n \\ x(n-N) \end{bmatrix}. \tag{3.2}$$

On the contrary, in the case of Volterra filtering the identity (3.2) is not valid. For simplicity, we will refer here to a second order homogeneous

Volterra filter given by

$$y(n) = \sum_{i=0}^{N_2-1} \sum_{j=i}^{N_2-1} c_{ij} x(n-i)x(n-j), \tag{3.3}$$

where N_2 is the filter memory length. The extension to the most general Nth order Volterra filter will be discussed in Session 3.

The Volterra filter input data vector is defined by :

$$\begin{aligned} \mathbf{x}_n = \big[&x^2(n), \ldots, x^2(n-N_2+1), x(n)x(n-1), \ldots, \\ &x(n-N_2+2)x(n-N_2+1), \ldots, x(n)x(n-N_2+1)\big]^T, \end{aligned} \tag{3.4}$$

which does not satisfy the time shift property.

In this case, at the time n, the N_2 elements contained in the vector

$$\mathbf{r}_{n-1} = \big[x^2(n-N_2), x(n-N_2+1)x(n-N_2), \ldots, x(n-1)x(n-N_2)\big]^T \tag{3.5}$$

are discarded from $\mathbf{x}_{n-1}$ and the N_2 elements contained in the vector

$$\mathbf{v}_n = \big[x^2(n), x(n)x(n-1), \ldots, x(n)x(n-N_2+1)\big]^T \tag{3.6}$$

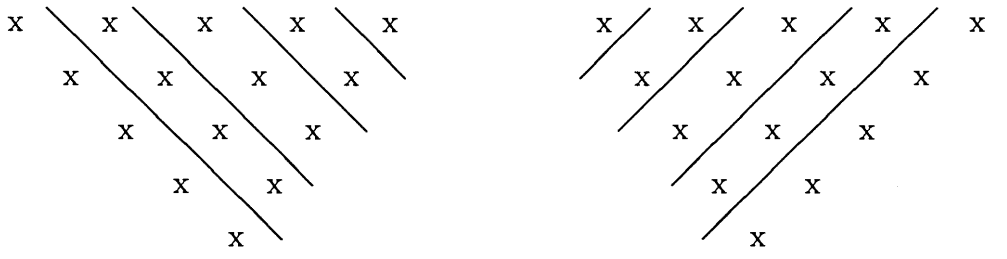
are added to the remaining elements [85]. Two extended input vectors are now defined at the time n: the first vector $\tilde{\mathbf{x}}_n$ is obtained by adding $\mathbf{v}_n$ to the top of $\mathbf{x}_{n-1}$ while the second one $\overline{\mathbf{x}}_n$ is obtained by adding $\mathbf{r}_{n-1}$ to the bottom of $\mathbf{x}_n$:

$$\tilde{\mathbf{x}}_n = \begin{bmatrix} \mathbf{v}_n \\ \mathbf{x}_{n-1} \end{bmatrix} \tag{3.7}$$

and

$$\overline{\mathbf{x}}_n = \begin{bmatrix} \mathbf{x}_n \\ \mathbf{r}_{n-1} \end{bmatrix}. \tag{3.8}$$

The two extended vectors, due to the presence of different products in the input data vector, do not coincide nor it is possible to make them coincide by an appropriate element arrangement of $\mathbf{x}_n$, $\mathbf{v}_n$ and $\mathbf{r}_{n-1}$. However, the augmented vectors contain the same elements and differ only by a permutation. The necessity for this permutation is due to the loss of the time shift property. In [85] a standard fast RLS adaptive algorithm has been extended to the Volterra case by taking into account the above mentioned permutation. The algorithm described in [85] is fast but it is not numerically stable. Fast and numerically stable algorithms can be obtained in the linear case by means of triangular matrices, *i.e.* we may refer to the Fast QR algorithms [9, 31, 90, 149] or QRD-based Lattice



a) The left columns b) The right columns

Figure 3.1: Definition of the left and right columns

algorithms [89, 110, 115]. If we have an adaptive algorithm which employs both the concepts of extended input data vector and of triangular matrix, as in our Fast SQR algorithm, then the extension to the Volterra case, by taking into account the upper mentioned permutation, becomes very difficult: in fact this permutation leads to the loss of the triangular structure of the matrices involved. In order to preserve the time shift property and to avoid permutations we can arrange the input data in a non-rectangular matrix, called *V-vector*:

$$\mathbf{x}_n = \begin{matrix} x^2(n) & x^2(n-1) & \dots & x^2(n-N_2+1) \\ x(n)x(n-1) & \dots & x(n-N_2+2)x(n-N_2+1) & \\ & \vdots & & \\ & x(n)x(n-N_2+1) & & \end{matrix} \tag{3.9}$$

where the diagonal brackets emphasise the non rectangular structure of the matrix.

For the non rectangular matrix in (3.9) we can define *left* and *right columns* as shown in Figure 3.1. It is clear that the first left column of $\mathbf{x}_n$ is formed by the elements which have been added going from $\mathbf{x}_{n-1}$ to $\mathbf{x}_n$ while the last right column of $\mathbf{x}_n$ is formed by the elements which will be discarded in the transition from $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$. Let us define the extended input V-vector as obtained by adding $\mathbf{v}_n$ to $\mathbf{x}_{n-1}$ as the first left column or by adding $\mathbf{r}_{n-1}$ to $\mathbf{x}_n$ as the last right column. Thus, it turns out that the two definitions of the extended input data V-vector are equal and therefore permutations can be avoided :

$$\widetilde{\mathbf{x}}_n = \left\backslash \mathbf{v}_n \setminus \mathbf{x}_{n-1} \right/ = \overline{\mathbf{x}}_n = \left\backslash \mathbf{x}_n \,/\, \mathbf{r}_{n-1} \right/ \tag{3.10}$$

Note the difference of notation between $\left\backslash \mathbf{a} \setminus \mathbf{b} \right/$ and $\left\backslash \mathbf{c} \,/\, \mathbf{d} \right/$; in the

first case **a** indicates the first left column and **b** the remaining columns of the V-vector while in the second case **d** stands for the last right column and **c** for the rest of columns. In what follows, for simplicity, the first left column and the last right column will be called first column and last column, respectively. For example, the extended input data V-vector for the second order homogeneous Volterra filter is given by

$$\overline{\mathbf{x}}_n = \begin{matrix} x^2(n) & x^2(n-1) & \dots & x^2(n-N_2) \\ x(n)x(n-1) & \dots & x(n-N_2+1)x(n-N_2) & \\ \vdots & & & \\ x(n)x(n-N_2+1) & x(n-1)x(n-N_2) & & \end{matrix} \tag{3.11}$$

In a more general context we can define a V-vector as a non-rectangular matrix in which the number of elements in each row does not increase going from the top to the bottom of the matrix. Even if some algorithms like Fast RLS, FTF are indifferent to V-vector rows arrangement, the V descendent structure becomes fundamental for the extension of adaptive algorithms like the lattice algorithms and the algorithms which use both the concepts of triangular matrix and of augmented data vector as shown in Section 3.4.

By varying the number of rows and the number of elements in each row we obtain V-vectors of different type. The *type* of a V-vector is the m-tupla of integers that defines the number of rows (m) and the number of elements in each row of the V-vector. For example, the type of the V-vector in (3.9) is the N_2-tupla $(N_2, N_2-1, \dots, 1)$. In the following, the type of a V-vector for simplicity will be designated with a capital letter.

After defining V-vectors, which replace the vectors, we can introduce the entity which replace the matrix of classical linear algebra. A *V-matrix* $M \times N$ is a V-vector of a certain type M whose elements are again V-vectors (subV-vectors) of a generally different type N. A V-matrix is depicted in Figure 3.2.

The elements of a V-vector $\backslash a_{ij} /$ can be identified by a couple of indexes: the first index indicates the row while the second indicates the column. Analogously the elements of a V-matrix $\backslash\backslash A_{ij_{lm}} //$ are identified by two couple of indexes, the first couple ij indicates the subV-vector while the second couple lm identifies the element in the subV-vector. Note that, when it is necessary, V-matrices are identified with double diagonal brackets.

In order to complete the definition of the novel algebraic structure we have to define the basic operations between V-vectors and V-matrices.

- Let **a** and **b** indicate two V-vectors of the same type, then the sum

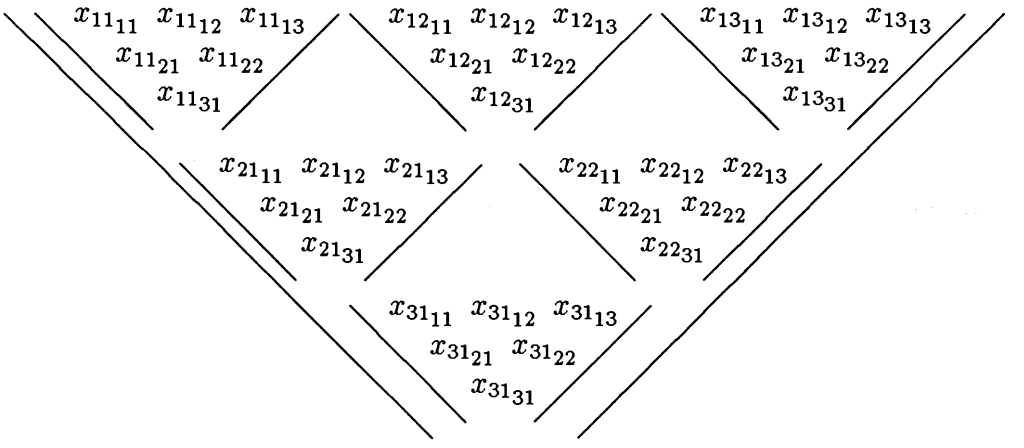


Figure 3.2: A V-matrix

of these two entities is a V-vector of the same type whose elements are given by :

$$c_{ij} = a_{ij} + b_{ij}.$$

- Let **A** and **B** indicate two V-matrices $M \times N$, then the sum between these two V-matrices is a V-matrix $M \times N$ whose elements are given by :

$$C_{ij_{lm}} = A_{ij_{lm}} + B_{ij_{lm}}.$$

- Let **a** and **b** indicate two V-vectors of the same type, then the inner product between these two entities is a scalar given by :

$$\mathbf{a} \cdot \mathbf{b} = \sum_{ij} a_{ij} b_{ij}.$$

- Let **A** indicate a V-matrix $M \times N$, and let **B** indicate a V-matrix $N \times R$ then the product $\mathbf{A} \cdot \mathbf{B}$ is a V-matrix $M \times R$ whose elements are given by :

$$C_{ij_{lm}} = \sum_{hk} A_{ij_{hk}} B_{hk_{lm}}.$$

- Let **A** indicate a V-matrix $M \times N$ and **b** indicate a V-vector of type N then the product $\mathbf{A} \cdot \mathbf{b}$ is a V-vector of type M whose elements are given by :

$$p_{ij} = \sum_{hk} A_{ij_{hk}} b_{hk}.$$

Moreover, in order to derive Volterra or multichannel linear adaptive RLS algorithms, we have to extend the linear algebra concepts of Transposed matrix, Identity matrix, Inverse matrix and Triangular matrix. These concepts are redefined as follows.

- The *Transposed V-matrix* of an $M \times N$ V-matrix $\mathbf{A} = \left\| a_{ij_{lm}} \right\|$ is a $N \times M$ V-matrix $\mathbf{A}^T$ with $\mathbf{A}^T = \left\| a_{lm_{ij}} \right\|$. That is, the transposed V-matrix has each subV-vector constituted by the elements of $\mathbf{A}$ which occupy in the different subV-vectors the same positions of the subV-vector in $\mathbf{A}^T$; furthermore each element is arranged with the same order of the corresponding subV-vector of $\mathbf{A}$.

There is an analogy between matrices and V-matrices: we may note that subV-vectors correspond to the rows of matrices. Unfortunately the "columns" of a V-matrix are more difficult to be visualised, however columns can be easily identified as the subV-vectors of the transposed V-matrix.

- The *Identity V-matrix* is an $M \times M$ V-matrix $\left\| I_{ij_{lm}} \right\|$ with all null elements except for the unit elements which present the couple of indexes ij equal to lm.

The definition of Inverse V-matrix is the same of linear algebra:

- The *Inverse V-matrix* of an $M \times M$ V-matrix $\mathbf{A}$ is the $M \times M$ V-matrix which pre- or post-multiplied by $\mathbf{A}$ gives the identity V-matrix.

It is straightforward to verify that, in case V-matrices or V-vectors reduce to matrices or vectors respectively, all these definitions coincide with those of linear algebra.

A class of V-matrices of particular interest is formed by *triangular V-matrices*. The great freedom in arranging the null elements allows the introduction of twelve different canonical triangular V-matrices:

Right Upper Triangular I	RUT I
Right Lower Triangular I	RLT I
Left Upper Triangular I	LUT I
Left Lower Triangular I	LLT I
Right Upper Triangular II	RUT II
Right Lower Triangular II	RLT II
Left Upper Triangular II	LUT II
Left Lower Triangular II	LLT II
Right Upper Triangular III	RUT III
Right Lower Triangular III	RLT III
Left Upper Triangular III	LUT III
Left Lower Triangular III	LLT III

Let us start defining triangular V-matrices of kind I. A V-matrix $\backslash\backslash A_{ij_{lm}} /\!/$ is:

- Right Upper Triangular I when all elements are null if

$$\begin{cases} m < j \\ l < i \text{ when } m = j. \end{cases}$$

- Right Lower Triangular I when all elements are null if

$$\begin{cases} m < j \\ l > i \text{ when } m = j. \end{cases}$$

- Left Upper Triangular I when all elements are null if

$$\begin{cases} m > j \\ l < i \text{ when } m = j. \end{cases}$$

- Left Lower Triangular I when all elements are null if

$$\begin{cases} m > j \\ l > i \text{ when } m = j. \end{cases}$$

With regard to the triangular V-matrices of kind II they are obtained from V-matrices of kind I by rotation around a vertical axis. Particularly

RUT II	comes from	LUT I
RLT II	comes from	LLT I
LUT II	comes from	RUT I
LLT II	comes from	RLT I

For triangular V-matrices of kind III we have that a V-matrix $\backslash\backslash A_{ij_{lm}} //$ is

- Right Upper Triangular III when all elements are null if

$$\begin{cases} l < i \\ m < j \text{ when } l = i. \end{cases}$$

- Right Lower Triangular III when all elements are null if

$$\begin{cases} l > i \\ m < j \text{ when } l = i. \end{cases}$$

- Left Upper Triangular III when all elements are null if

$$\begin{cases} l < i \\ m > j \text{ when } l = i. \end{cases}$$

- Left Lower Triangular III when all elements are null if

$$\begin{cases} l > i \\ m > j \text{ when } l = i. \end{cases}$$

The rotation around a vertical axis of a triangular V-matrix of kind III produces again a triangular V-matrix of kind III.

In Fig. 3.3-3.5 three cases of triangular V-matrices of different kind are graphically represented. As we may see from these figures, in triangular V-matrices of kind I and II we follow a routing order by columns while in triangular V-matrices of kind III we have a routing order by rows. Moreover we named Left (Right) Triangular V-matrices because the non-null elements tend to take place on the left (right) part of each subV-vector. Finally, we named Upper (Lower) Triangular V-matrix because:

- in V-matrices of kind I the matrix whose rows are equal to the first columns of subV-vectors of the first column is an upper (lower) triangular matrix;

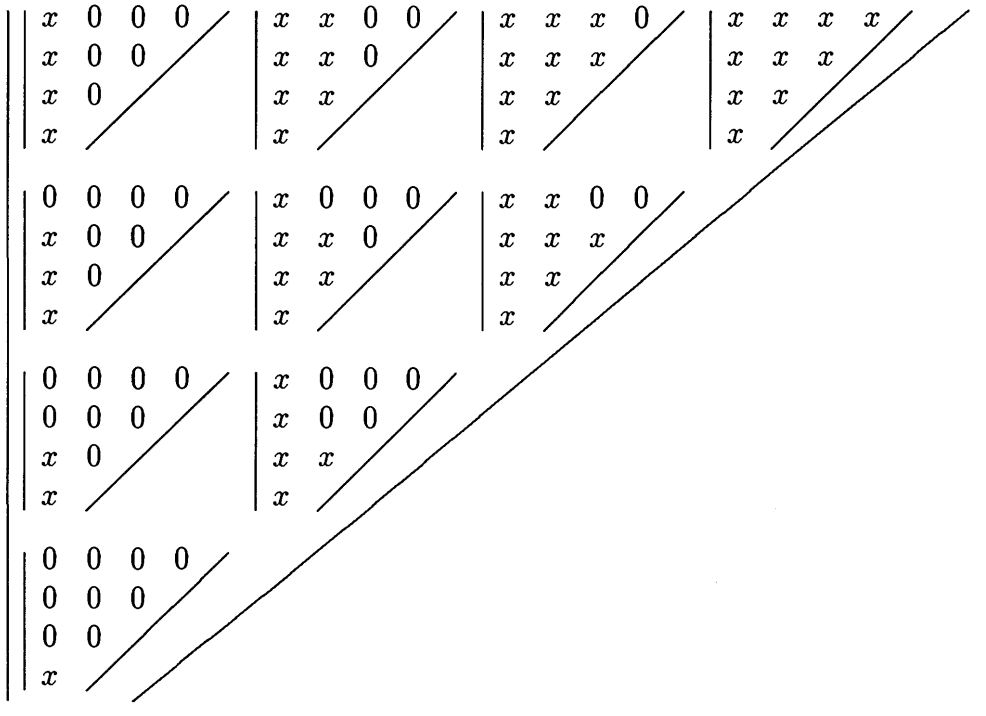


Figure 3.3: A Left Upper Triangular I V-matrix

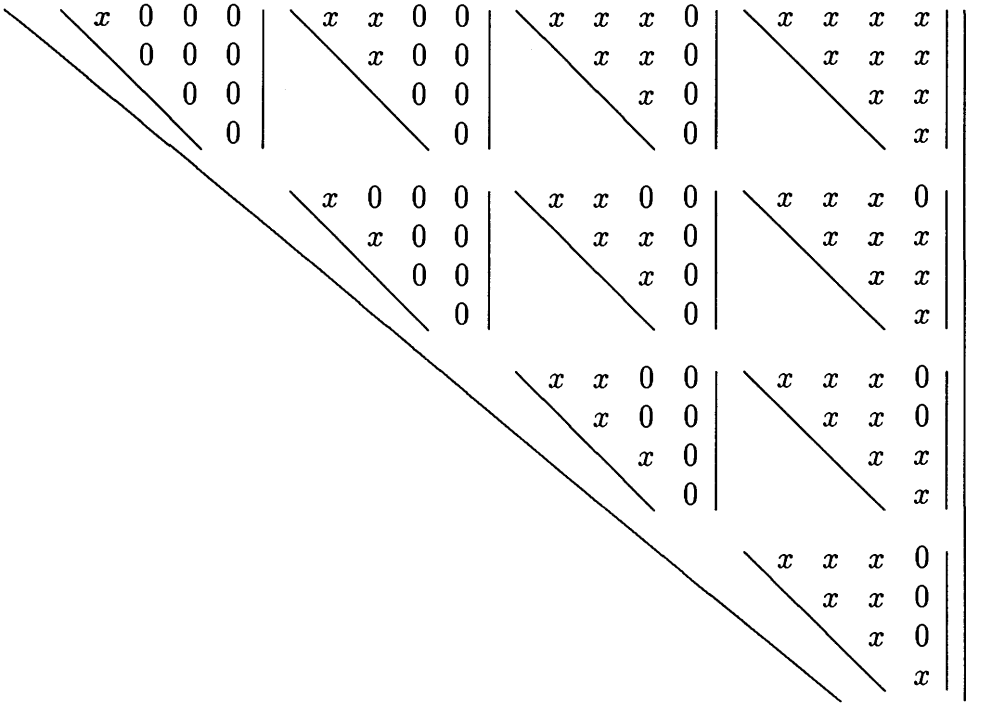


Figure 3.4: A Left Upper Triangular II V-matrix

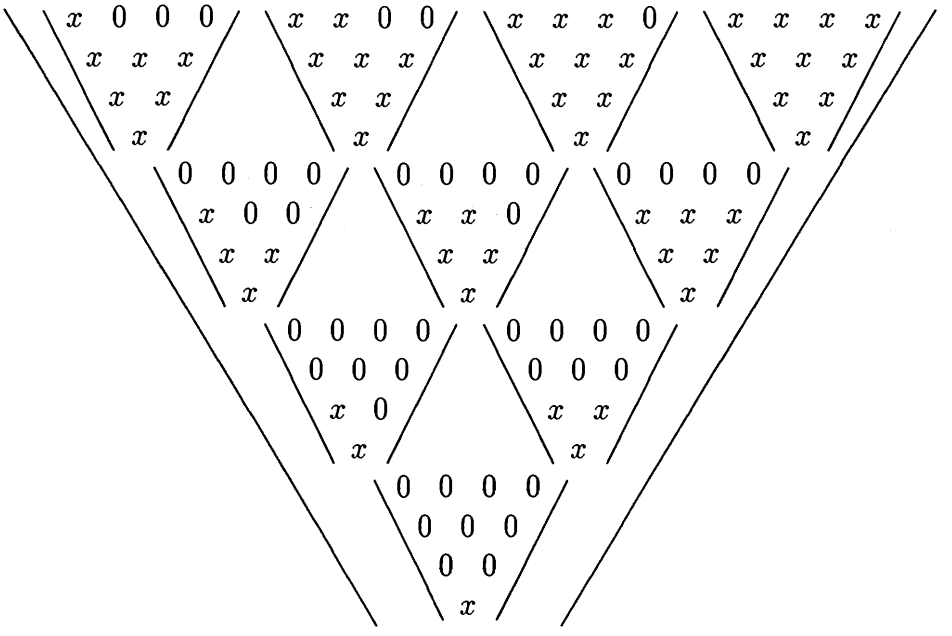


Figure 3.5: A Left Upper Triangular III V-matrix

- in V-matrices of kind II the matrix whose rows are equal to the last columns of subV-vectors of the last column is an upper (lower) triangular matrix;
- in V-matrices of kind III the matrix whose rows are equal to the first columns of subV-vectors of the first column and the matrix whose rows are equal to the last columns of subV-vectors of the last column are upper (lower) triangular matrices.

In right (left) strictly decreasing V-matrices of kind I the matrix whose rows are equal to the last columns of subV-vectors of the last column is lower (upper) triangular.

In right (left) strictly decreasing V-matrices of kind II the matrix whose rows are equal to the first columns of subV-vectors of the first column is upper (lower) triangular.

Moreover we can see that :

- the transposed of a right (left) triangular V-matrix is a left (right) triangular V-matrix;
- the transposed of an upper (lower) triangular V-matrix is a lower (upper) triangular V-matrix;

- the transposed of a triangular V-matrix of kind I (II) (III) is a triangular V-matrix of kind I (II) (III).

It is straightforward to demonstrate the validity of these two important properties that derive from linear algebra.

- The product between two triangular V-matrices with the same triangular structure is still a V-matrix with the same triangular structure.

- The inverse of a triangular V-matrix is a V-matrix with the same triangular structure.

The above mentioned V-matrices are not the unique triangular structures which satisfy these properties: in general any element routing order in a V-vector of type M will define a triangular structure for V-matrices $M \times M$. In particular for the following discussions it is important to define triangular V-matrices "row k MOD L" where L is the number of rows of the V-matrix. These triangular V-matrices are obtained by considering a routing order which starts from row k, instead of row 1, and scans columns in a cyclic manner.

For instance we define $\backslash\!\backslash a_{ijlm} /\!/$ a "row k MOD L" Left Upper Triangular III V-matrix when all elements are null if

$$\begin{cases} \mathrm{mod}_L(l-k+1) < \mathrm{mod}_L(i-k+1) \\ \qquad m > j \text{ when } l = i. \end{cases}$$

Note that the transposed of a "row k MOD L" triangular V-matrix is still a "row k MOD L" triangular V-matrix.

3.3 V-vectors for Volterra and Linear Multichannel Filters

The V-vector formalism can be applied to Volterra filters of arbitrary order. The problem we address now is the arrangement of the input data products of a k-th order homogeneous Volterra operator in a V decreasing structure, *i.e.* we want to derive the k-th order input data V-vector. This arrangement may be done in a simple manner by the use of a filter order/memory length recursion. We can demonstrate, in fact, this proposition :

Proposition 3.3.1 *In order to pass from a $(k-1)$-th memory length, i-th order input data V-vector to a k-th memory length, i-th order input data V-vector we have:*

1. *to add to the $(k-1)$-th V-vector a right column of products with the same building rule of the rows of our V-vector but translated of one unit in time;*

2. *to add to the bottom of the $(k-1)$-th V-vector the vector*

$$\mathbf{r}_n^{[k](i-1)}x(n) \tag{3.12}$$

or the vector

$$\mathbf{v}_n^{[k](i-1)}x(n-k+1), \tag{3.13}$$

where $\mathbf{r}_n^{[k](i-1)}$ indicates the last right column of the k-th memory length, $(i-1)$-th order V-vector and $\mathbf{v}_n^{[k](i-1)}$ the first left column of the same V-vector.

Note that $\mathbf{r}_n^{[k](i-1)}x(n)$ has the same elements of $\mathbf{v}_n^{[k](i-1)}x(n-k+1)$. With the rule of Proposition 3.3.1 it is trivial to build the i-th order input data V-vector from the knowledge of the $(i-1)$-th V-vector. Note that the i-th order V-vector with memory length 1 is equal to $\backslash x^i(n)/$.

Let us proof the validity of the proposition.

Proof The products of our k-th memory length Volterra operator can be divided into three different classes :

1. products which belong to the $(k-1)$-th memory length V-vector;

2. products constituted in the same manner of the rows of the $(k-1)$-th memory length V-vector but translated one unit in time;

3. products which do not belong to the previous classes and that, for this reason, must present both the input data $x(n)$ and $x(n-k+1)$.

The proposed recursive procedure simply translates this class division into an element arranging rule. In this way we have only to demonstrate that the third class coincides with the collection of elements of $x(n)\mathbf{r}_n^{[k](i-1)}$ and $x(n-k+1)\mathbf{v}_n^{[k](i-1)}$. An element of this vectors cannot appear in the first two classes because in every product are present both $x(n)$ and $x(n-k+1)$; moreover if we consider a product ξ of this third class then $\frac{\xi}{x(n)}$ is an $(i-1)$-th order product with $x(n-k+1)$

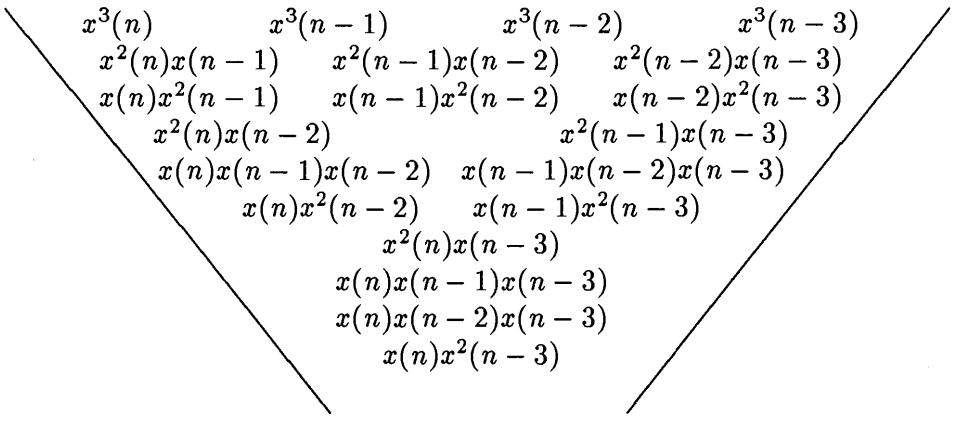


Figure 3.6: Input data V-vector of an order 3, memory length 4, homogeneous Volterra Filter.

as a factor, and for this reason it belongs to $\mathbf{r}_n^{[k](i-1)}$, while $\frac{\xi}{x(n-k+1)}$ analogously has $x(n)$ as factor and so it belongs to $\mathbf{v}_n^{[k](i-1)}$. □

In Figure 3.6 is presented an example of memory length 4, order 3 input data V-vector built up with the previous arranging rule based on $\mathbf{v}_n^{[k](i)}$.

Obviously the possible use of vector $\mathbf{r}_n^{[k](i)}$ instead of $\mathbf{v}_n^{[k](i)}$ leads to a different formulation of input data V-vector. The two V-vectors differ by a permutation of equal length rows, but this permutation does not have any influence on the algorithm development.

In most cases, we are not interested in a homogeneous Volterra operator but in a complete Nth memory length Kth order Volterra filter given by the following input-output relation

$$y(n) = \sum_{k=1}^{K} \sum_{i_1=1}^{N} \sum_{i_1=i_2}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} h_{i_1,\ldots,i_k} x(n-i_1)\ldots x(n-i_k). \qquad (3.14)$$

V-vector algebra can be applied also to the filter class of equation (3.14). We can derive the input data V-vector of the filter in (3.14) by first building the input data V-vector of each homogeneous Volterra operator of the filter in (3.14) and then by arranging all the rows of these V-vectors in a unique V descendant structure. In Figure 3.7 is depicted the V-vector of a 3-rd order, memory length 3 Volterra filter.

It is worth noting that the V-vector algebra can be used also to deal with multichannel linear filters with channels of different memory length.

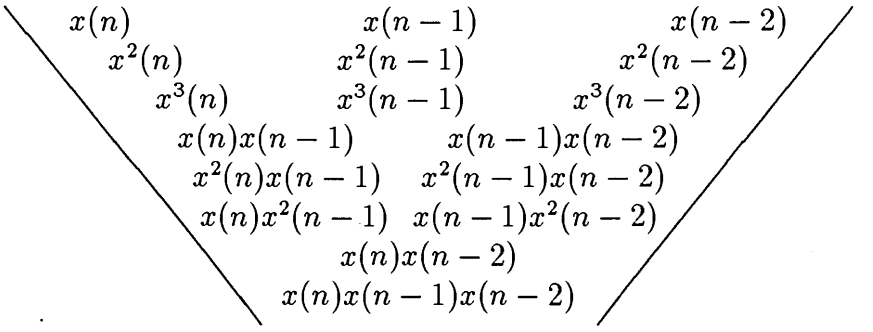


Figure 3.7: Input data V-vector of an order 3, memory length 3 Volterra filter.

The input data V-vector of the linear multichannel filter can be trivially obtained by arranging the input data vectors of the different channels in a unique V descendant structure.

3.4 A Reformulation of the Lee-Mathews' Adaptive Fast RLS Algorithm

The algorithms presented in this section and in Section 3.5 apply to any filter whose output is a linear function of an input data V-vector that satisfy the time shift property. This filter class includes both the Volterra and the linear multichannel filters.

Again we want to find at each time n a fast recursive solution for the exponentially windowed cost function :

$$J_n = \sum_{k=0}^{n} \lambda^{n-k} \left| d(k) - d_n(k) \right|^2 , \tag{3.15}$$

where λ is an exponential weight called "forgetting factor" that controls the rate of tracking time-varying signals, $d(k)$ is the desired adaptive filter output, and $d_n(k)$ is the adaptive filter output at time k:

$$d_n(k) = \mathbf{w}_n^T \mathbf{x}_k. \tag{3.16}$$

Note that in $d_n(k)$ the subscript n indicate that the output signal is evaluated from the optimal coefficient vector at time n. Moreover, $\mathbf{x}_k$ is the input data V-vector and $\mathbf{w}_n$ is the optimal filter coefficient V-vector.

The algorithm proposed in [85] is based on the relationship between the forward prediction filter, which estimates $\mathbf{v}_n$ from $\mathbf{x}_{n-1}$, and the

backward prediction filter, which estimates $\mathbf{r}_{n-1}$ from $\mathbf{x}_n$. The sets of optimal coefficients of the forward and backward prediction filters will be indicated by $\mathbf{A}_n$ and $\mathbf{B}_n$ respectively. In [85] $\mathbf{A}_n$ and $\mathbf{B}_n$ are matrices; on the contrary in this context they will be V-vectors of vectors, *i.e.* V-matrices whose subV-vectors are made up of a unique row.

The corresponding prediction error vectors at time k, denoted as $\mathbf{f}_n(k)$ and $\mathbf{b}_n(k)$, are then defined as

$$\mathbf{f}_n(k) = \mathbf{v}_k + \mathbf{A}_n^T \mathbf{x}_{k-1} \tag{3.17}$$

and

$$\mathbf{b}_n(k) = \mathbf{r}_{k-1} + \mathbf{B}_n^T \mathbf{x}_k. \tag{3.18}$$

A crucial role in the development of the coefficient update equations is played by the Kalman Gain $\mathbf{c}_n$, defined as

$$\mathbf{c}_n = \boldsymbol{\Omega}_n^{-1} \mathbf{x}_n, \tag{3.19}$$

where $\boldsymbol{\Omega}_n$ is the autocorrelation V-matrix

$$\boldsymbol{\Omega}_n = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_k \mathbf{x}_k^T. \tag{3.20}$$

The Kalman gain is now a V-vector and it may be viewed as the optimal coefficient V-vector of a transversal filter that estimates the pinning sequence. The corresponding estimation error γ_n , which is usually called "likelihood variable", is given by

$$\gamma_n = 1 - \mathbf{c}_n^T \mathbf{x}_n. \tag{3.21}$$

The likelihood variable assumes a great importance in all Fast Transversal Filter algorithms. In fact it monitors the numerical stability of the algorithm itself. According to [60], γ_n is a real value bounded by zero and one, $0 \leq \gamma_n \leq 1$, and instability arises when γ_n exceeds these bounds due to finite precision of processors and to error propagation.

The coefficients update can be immediately obtained from the knowledge of the Kalman gain V-vector and of the prediction or estimation errors :

$$\mathbf{A}_n = \mathbf{A}_{n-1} - \mathbf{c}_{n-1} \mathbf{f}_{n-1}^T(n), \tag{3.22}$$

$$\mathbf{B}_n = \mathbf{B}_{n-1} - \mathbf{c}_n \mathbf{b}_{n-1}^T(n), \tag{3.23}$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{c}_n e_{n-1}(n), \tag{3.24}$$

where the estimation error $e_n(k)$ is given by

$$e_n(k) = d(k) - \mathbf{w}_n^T \mathbf{x}_k. \tag{3.25}$$

In order to update the gain V-vector $\mathbf{c}_n$ let us introduce the extended Kalman gain V-vector $\overline{\mathbf{c}}_n$ that is the least square (LS) estimate for the pinning sequence using the extended input data V-vector. Two different derivations of $\overline{\mathbf{c}}_n$ may be obtained using the two definitions of $\overline{\mathbf{x}}_n$ given in (3.10):

$$\overline{\mathbf{c}}_n = \left\backslash\ \boldsymbol{\alpha}_n^{-1}\mathbf{f}_n(n)\ \backslash\ \mathbf{A}_n\boldsymbol{\alpha}_n^{-1}\mathbf{f}_n(n) + \mathbf{c}_{n-1}\ \right/, \tag{3.26}$$

$$\overline{\mathbf{c}}_n = \left\backslash\ \mathbf{c}_n + \mathbf{B}_n\boldsymbol{\beta}_n^{-1}\mathbf{b}_n(n)\ /\ \boldsymbol{\beta}_n^{-1}\mathbf{b}_n(n)\ \right/, \tag{3.27}$$

where $\boldsymbol{\alpha}_n$ and $\boldsymbol{\beta}_n$ are the autocorrelation matrices of, respectively, the forward and the backward prediction errors[1]:

$$\boldsymbol{\alpha}_n = \sum_{k=0}^{n} \lambda^{n-k}\mathbf{f}_n(k)\mathbf{f}_n^T(k), \tag{3.28}$$

$$\boldsymbol{\beta}_n = \sum_{k=0}^{n} \lambda^{n-k}\mathbf{b}_n(k)\mathbf{b}_n^T(k), \tag{3.29}$$

Let us first demonstrate expression (3.26). Remember that

$$\overline{\boldsymbol{\Omega}}_n = \sum_{k=0}^{n} \lambda^{n-k}\overline{\mathbf{x}}_k\overline{\mathbf{x}}_k^T \tag{3.30}$$

and

$$\overline{\mathbf{c}}_n = \overline{\boldsymbol{\Omega}}_n^{-1}\overline{\mathbf{x}}_n \tag{3.31}$$

then

$$\overline{\boldsymbol{\Omega}}_n\overline{\mathbf{c}}_n = \overline{\mathbf{x}}_n. \tag{3.32}$$

By considering

$$\overline{\mathbf{x}}_n = \left\backslash \mathbf{v}_n \backslash \mathbf{x}_{n-1} \right/ \tag{3.33}$$

and

$$\overline{\mathbf{c}}_n = \left\backslash \mathbf{a} \backslash \mathbf{b} \right/ \tag{3.34}$$

we have

$$\begin{aligned}
\overline{\boldsymbol{\Omega}}_n\overline{\mathbf{c}}_n &= \sum_{k=0}^{n} \lambda^{n-k}\left\backslash \mathbf{v}_k \backslash \mathbf{x}_{k-1} \right/ \cdot \left\backslash \mathbf{v}_k \backslash \mathbf{x}_{k-1} \right/^T \cdot \overline{\mathbf{c}}_n = \\
&= \sum_{k=0}^{n} \lambda^{n-k}\left\backslash \mathbf{v}_k \backslash \mathbf{x}_{k-1} \right/ \cdot (\mathbf{v}_k^T\mathbf{a} + \mathbf{x}_{k-1}^T\mathbf{b}) = \\
&= \sum_{k=0}^{n} \lambda^{n-k}\left\backslash \mathbf{v}_k(\mathbf{v}_k^T\mathbf{a} + \mathbf{x}_{k-1}^T\mathbf{b}) \right\backslash \mathbf{x}_{k-1}(\mathbf{v}_k^T\mathbf{a} + \mathbf{x}_{k-1}^T\mathbf{b}) \right/
\end{aligned} \tag{3.35}$$

[1] Please note that we use small bold letters to indicate the autocorrelation matrices of the prediction errors in order to maintain the same symbols used in [85].

From (3.32) with simple manipulations we obtain

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{v}_k \mathbf{v}_k^T \mathbf{a} + \sum_{k=0}^{n} \lambda^{n-k} \mathbf{v}_k \mathbf{x}_{k-1}^T \mathbf{b} = \mathbf{v}_n \tag{3.36}$$

and

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{v}_k^T \mathbf{a} + \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T \mathbf{b} = \mathbf{x}_{n-1}. \tag{3.37}$$

Following the derivations of Section 2.3, it is straightforward to demonstrate that,

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{v}_k \mathbf{v}_k^T = \boldsymbol{\alpha}_n + \mathbf{A}_n^T \boldsymbol{\Omega}_{n-1} \mathbf{A}_n, \tag{3.38}$$

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{v}_k^T = -\boldsymbol{\Omega}_{n-1} \mathbf{A}_n, \tag{3.39}$$

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T = \boldsymbol{\Omega}_{n-1}. \tag{3.40}$$

By substituting (3.38), (3.39) and (3.40) in (3.36) and (3.37), the following equations are derived

$$\mathbf{a} = \boldsymbol{\alpha}_n^{-1} \mathbf{f}_n(n), \tag{3.41}$$

$$\mathbf{b} = \mathbf{c}_{n-1} + \mathbf{A}_n(\boldsymbol{\alpha}_n^{-1} \mathbf{f}_n(n)), \tag{3.42}$$

which correspond to (3.26). In the same way from $\overline{\mathbf{x}}_n = \backslash \mathbf{x}_n / \mathbf{r}_{n-1} /$ we can demonstrate expression (3.27).

Note that the two expressions (3.26) and (3.27) do not differ for a permutation as in [85] and therefore they can be immediately equated as it happens in the case of linear filters.

Let $\boldsymbol{\mu}_n$ and $\mathbf{m}_n$ indicate respectively the last right column and the remaining first columns of $\overline{\mathbf{c}}_n$:

$$\backslash \mathbf{m}_n \; / \; \boldsymbol{\mu}_n / = \overline{\mathbf{c}}_n, \tag{3.43}$$

then it is possible to demonstrate (see [85]) that

$$\mathbf{c}_n = \left(1 - \mathbf{b}_{n-1}^T(n) \boldsymbol{\mu}_n\right)^{-1} [\mathbf{m}_n - \mathbf{B}_{n-1} \boldsymbol{\mu}_n] \tag{3.44}$$

and

$$\gamma_n = \left(1 - \mathbf{b}_{n-1}^T(n) \boldsymbol{\mu}_n\right)^{-1} \overline{\gamma}_n \tag{3.45}$$

where $\overline{\gamma}_n$ is the "extended" likelihood variable:

$$\overline{\gamma}_n = 1 - \overline{\mathbf{c}}_n^T \overline{\mathbf{x}}_n = \gamma_{n-1} - \mathbf{f}_n^T(n)\boldsymbol{\alpha}_n^{-1}\mathbf{f}_n(n), \tag{3.46}$$

In Table 3.1 the complete Volterra Fast RLS algorithm is summarized. The number of multiplications involved in each expression is also reported. We indicate with L the number of rows of the input data V-vector, *i.e.* the number of "channels" of the Volterra or multichannel linear filter, and with N_T the total number of coefficients. For example, in case of an homogeneous second order Volterra filter $L = N_2$ and $N_T = N_2(N_2 + 1)/2$.

As we can see, this algorithm is identical to that derived by Lee and Mathews [85], with the same computational complexity and the same stability properties. Practically, the principal difference with the use of V-vectors is that now permutations are avoided; the algorithm itself is really similar to that of linear filters because all derivations are identical. Furthermore the V-vector formalism is completely independent from the order of a Volterra filter or from the channels' number of a multichannel linear filter. Therefore from these viewpoints V-vectors appear as one of the most natural way to deal with Volterra and multichannel linear filters in order to develop adaptive algorithms.

3.5 A Volterra Givens Rotation Based Fast SQR RLS Filter

The Fast RLS algorithm requires a limited number of multiplications per sample ($6LN_T$), but it is unstable on the long term when implemented with a finite precision arithmetic. In order to obtain numerically stable algorithms many expedients have been devised [60]. As we remarked in the previous chapter, one of the most successful approaches is the Square Root (SQR) technique, in which the autocorrelation matrix is factorized as

$$\boldsymbol{\Omega}_n = \mathbf{R}_n^T \mathbf{R}_n. \tag{3.47}$$

This factorization is not unique: every $\mathbf{QR}$ matrix with $\mathbf{Q}$ orthogonal (or rotation) matrix ($\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$) fulfils the same relationship. In order to univocally identify the factor $\mathbf{R}_n$, we can choose to work with positive diagonal triangular matrices and in this case the factorization above mentioned is called Cholesky Factorization.

In this paragraph, we propose a new algorithm for adaptive Volterra prediction and filtering. The algorithm, which presents excellent numerical properties and belongs to the class of QR algorithms, is the extension

Algorithm	×
$\mathbf{f}_{n-1}(n) = \mathbf{v}_n + \mathbf{A}_{n-1}^T \mathbf{x}_{n-1}$	LN_T
$\mathbf{f}_n(n) = \gamma_{n-1}\mathbf{f}_{n-1}(n)$	
$\boldsymbol{\alpha}_n^{-1} = \lambda^{-1}\left[\boldsymbol{\alpha}_{n-1}^{-1} - \frac{\boldsymbol{\alpha}_{n-1}^{-1}\mathbf{f}_{n-1}(n)\mathbf{f}_{n-1}^T(n)\boldsymbol{\alpha}_{n-1}^{-1}}{\lambda\gamma_{n-1}^{-1}+\mathbf{f}_{n-1}^T(n)\boldsymbol{\alpha}_{n-1}^{-1}\mathbf{f}_{n-1}(n)}\right]$	$2L^2$
$\overline{\gamma}_n = \gamma_{n-1} - \mathbf{f}_n^T(n)\boldsymbol{\alpha}_n^{-1}\mathbf{f}_n(n)$	
$\mathbf{A}_n = \mathbf{A}_{n-1} - \mathbf{c}_{n-1}\mathbf{f}_{n-1}^T(n)$	LN_T
$\overline{\mathbf{c}}_n = \backslash\boldsymbol{\alpha}_n^{-1}\mathbf{f}_n(n)\backslash\mathbf{A}_n\boldsymbol{\alpha}_n^{-1}\mathbf{f}_n(n) + \mathbf{c}_{n-1}/ = \backslash\mathbf{m}_n/\boldsymbol{\mu}_n/$	LN_T
$\mathbf{b}_{n-1}(n) = \mathbf{r}_{n-1} + \mathbf{B}_{n-1}^T\mathbf{x}_n$	LN_T
$\mathbf{c}_n = \left(1 - \mathbf{b}_{n-1}^T(n)\boldsymbol{\mu}_n\right)^{-1}\left[\mathbf{m}_n - \mathbf{B}_{n-1}\boldsymbol{\mu}_n\right]$	LN_T
$\gamma_n = \left(1 - \mathbf{b}_{n-1}^T(n)\boldsymbol{\mu}_n\right)^{-1}\overline{\gamma}_n$	
$\mathbf{B}_n = \mathbf{B}_{n-1} - \mathbf{c}_n\mathbf{b}_{n-1}^T(n)$	LN_T
$e_{n-1}(n) = d(n) - \mathbf{w}_{n-1}^T\mathbf{x}_n$	N_T
$e_n(n) = \gamma_n e_{n-1}(n)$	
$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{c}_n e_{n-1}(n)$	N_T
TOTAL	$6LN_T$

Table 3.1: Algorithm for the computation of the Lee-Mathews Fast RLS, V-vector version. In the second column is reported the main term of each operation cost in multiplications.

to Volterra filters of the algorithm of Theorem 2.4.2. As in QR algorithms we do have a **Q** Givens rotation matrix and an **R** triangular matrix, which is the Cholesky factor of the autocorrelation matrix[2]; but differently from QR algorithms, the derivation of the filter is an algebraic one, based on the relationship between two different square-root factorizations of the extended autocorrelation matrix.

In order to obtain a fast algorithm, with LN_T operations per sample, the L rows of input data V-vector (the "channels" of the Volterra or multichannel linear filter) are updated in a sequential manner [146]. Practically, the filter update is divided into L steps and at each step a different channel is taken under consideration and updated.

Let us introduce first some new notations:

$\backslash a \backslash_i \mathbf{b} /$ means that a is an element placed before the first element of the i-th row of V-vector **b**;

$\backslash \mathbf{a} /_i b /$ means that b is an element placed after the last element of the i-th row of V-vector **a**;

$\left\| \begin{matrix} a & \backslash & \mathbf{b}^T \\ \mathbf{c} & \backslash_i & \mathbf{D} \end{matrix} \right\|$ means that **c** is a V-vector (a column in the matrix analogy) whose elements are placed before the first elements of the i-th rows of the corresponding subV-vectors of **D** and $\backslash a \backslash_i \mathbf{b} /$ is a subV-vector placed before the first subV-vector of the i-th row of V-matrix $\| \mathbf{c} \backslash_i \mathbf{D} \|$

$\left\| \begin{matrix} \mathbf{A} & / & \mathbf{b} \\ \mathbf{c}^T & /_i & d \end{matrix} \right\|$ means that **b** is a V-vector (a column in the matrix analogy) whose elements are placed after the last elements of the i-th rows of the corresponding subV-vectors of **A** and $\backslash \mathbf{c} /_i d /$ is a subV-vector placed before the first subV-vector of the i-th row of V-matrix $\| \mathbf{A} /_i \mathbf{b} \|$

Moreover, let us indicate with $\mathbf{x}_{n,i}$ the input data subV-vector in which only the first i-th rows/channels have been updated at time instant n; for $i = L$ it is $\mathbf{x}_{n,L} = \mathbf{x}_n$ and for $i = 0$ it is $\mathbf{x}_{n,0} = \mathbf{x}_{n-1}$.

All quantities with subscript i are referred to the i-th step and thus to the input data V-vector $\mathbf{x}_{n,i}$. The extended input data V-vector $\overline{\mathbf{x}}_{n,i}$

[2]In this chapter we prefer, for convenience, to deal with the square root matrix of $\boldsymbol{\Omega}_n$, *i.e.* $\boldsymbol{R}_n$, instead of the square root matrix of $\boldsymbol{\Omega}_n^{-1}$, *i.e.* $\boldsymbol{S}_n$. Note however that it is $\mathbf{R}_n^{-1} = \mathbf{S}_n$.

is obtained by placing $v_{n,i}$ (*i.e.* the i-th element of vector $\mathbf{v}_n$) in the left of the i-th row of $\mathbf{x}_{n,i-1}$ or placing $r_{n-1,i}$ in the right of the i-th row of $\mathbf{x}_{n,i}$.

The autocorrelation V-matrix, the Kalman gain V-vector, the forward prediction error, the forward prediction filter V-vector, the LS autocorrelation of forward prediction error, the backward prediction error and the backward prediction filter V-vector all at time instant n, step i are given respectively by:

$$\boldsymbol{\Omega}_{n,i} = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{n,i} \mathbf{x}_{n,i}^T, \tag{3.48}$$

$$\mathbf{c}_{n,i} = \boldsymbol{\Omega}_{n,i}^{-1} \mathbf{x}_{n,i}, \tag{3.49}$$

$$f_{n,i}(k) = v_{k,i} + \mathbf{a}_{n,i}^T \mathbf{x}_{k,i-1}, \tag{3.50}$$

$$\mathbf{a}_{n,i} = \mathbf{a}_{n-1,i} - \mathbf{c}_{n,i-1} f_{n-1,i}(n), \tag{3.51}$$

$$\alpha_{n,i} = \lambda \alpha_{n-1,i} + f_{n-1,i}(n) f_{n,i}(n), \tag{3.52}$$

$$b_{n,i}(k) = r_{k-1,i} + \mathbf{b}_{n,i}^T \mathbf{x}_{k,i}, \tag{3.53}$$

$$\mathbf{b}_{n,i} = \mathbf{b}_{n-1,i} - \mathbf{c}_{n,i} b_{n-1,i}(n). \tag{3.54}$$

This fast SQR algorithm is based on two different factorizations of the extended autocorrelation V-matrix [3]:

$$\overline{\boldsymbol{\Omega}}_{n,i} = \overline{\mathbf{R}}_{n,i}^T \overline{\mathbf{R}}_{n,i} = \widetilde{\mathbf{R}}_{n,i}^T \widetilde{\mathbf{R}}_{n,i}, \tag{3.55}$$

where

$$\widetilde{\mathbf{R}}_{n,i} = \left\| \begin{matrix} \alpha_{n,i}^{1/2} \\ -\mathbf{R}_{n,i-1} \mathbf{a}_{n,i} \end{matrix} \right. \left\backslash_i \begin{matrix} 0^T \\ \mathbf{R}_{n,i-1} \end{matrix} \right\| \tag{3.56}$$

$$\overline{\mathbf{R}}_{n,i} = \left\| \begin{matrix} \mathbf{R}_{n,i} \\ 0^T \end{matrix} \right. \left/_i \begin{matrix} -\mathbf{R}_{n,i} \mathbf{b}_{n,i} \\ \beta_{n,i}^{1/2} \end{matrix} \right\| \tag{3.57}$$

such that

$$\widetilde{\mathbf{R}}_{n,i}^{-T} = \left\| \begin{matrix} \alpha_{n,i}^{-1/2} \\ 0 \end{matrix} \right. \left\backslash_i \begin{matrix} \alpha_{n,i}^{-1/2} \mathbf{a}_{n,i}^T \\ \mathbf{R}_{n,i-1}^{-T} \end{matrix} \right\| \tag{3.58}$$

$$\overline{\mathbf{R}}_{n,i}^{-T} = \left\| \begin{matrix} \mathbf{R}_{n,i}^{-T} \\ \beta_{n,i}^{-1/2} \mathbf{b}_{n,i}^T \end{matrix} \right. \left/_i \begin{matrix} 0 \\ \beta_{n,i}^{-1/2} \end{matrix} \right\| \tag{3.59}$$

We prove first equation (3.56).[4] The extended autocorrelation matrix is given by

$$\overline{\boldsymbol{\Omega}}_{n,i} = \sum_{k=0}^{n} \lambda^{n-k} \left\backslash v_{k,i} \right\backslash_i \mathbf{x}_{k,i-1} \Big/ \cdot \left\backslash v_{k,i} \right\backslash_i \mathbf{x}_{k,i-1} \Big/^T$$

[3] In the following we indicate with $\mathbf{R}^{-T}$ the transposed of the inverse of $\mathbf{R}$.

[4] See Appendix 3.A for the fundamental operations on the above V-matrices.

$$= \sum_{k=0}^{n} \lambda^{n-k} \left\| \begin{matrix} v_{k,i}^2 \\ \mathbf{x}_{k,i-1} v_{k,i} \end{matrix} \right\backslash_i \left. \begin{matrix} v_{k,i}\mathbf{x}_{k,i-1}^T \\ \mathbf{x}_{k,i-1}\mathbf{x}_{k,i-1}^T \end{matrix} \right\| \tag{3.60}$$

Following the derivations of Section 2.3, it is straightforward to demonstrate that

$$\sum_{k=0}^{n} \lambda^{n-k} v_{k,i}^2 =, \alpha_{n,i} + \mathbf{a}_{n,i}^T \boldsymbol{\Omega}_{n,i-1} \mathbf{a}_{n,i} \tag{3.61}$$

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k,i-1} y_{k,i} = -\boldsymbol{\Omega}_{n,i-1} \mathbf{a}_{n,i} \tag{3.62}$$

$$\sum_{k=0}^{n} \lambda^{n-k} \mathbf{x}_{k,i-1} \mathbf{x}_{k,i-1}^T = \boldsymbol{\Omega}_{n,i-1} \tag{3.63}$$

Thus, it is

$$\overline{\boldsymbol{\Omega}}_{n,i} = \left\| \begin{matrix} \alpha_{n,i} + \mathbf{a}_{n,i}^T \boldsymbol{\Omega}_{n,i-1} \mathbf{a}_{n,i} \\ -\boldsymbol{\Omega}_{n,i-1} \mathbf{a}_{n,i} \end{matrix} \right\backslash_i \left. \begin{matrix} -\mathbf{a}_{n,i}^T \boldsymbol{\Omega}_{n,i-1} \\ \boldsymbol{\Omega}_{n,i-1} \end{matrix} \right\|. \tag{3.64}$$

It is trivial to verify that a SQR factorization of the extended autocorrelation V-matrix $\overline{\boldsymbol{\Omega}}_{n,i} = \widetilde{\mathbf{R}}_{n,i}^T \widetilde{\mathbf{R}}_{n,i}$ is given by

$$\widetilde{\mathbf{R}}_{n,i} = \left\| \begin{matrix} \alpha_{n,i}^{1/2} \\ -\mathbf{R}_{n,i-1} \mathbf{a}_{n,i} \end{matrix} \right\backslash_i \left. \begin{matrix} \mathbf{0}^T \\ \mathbf{R}_{n,i-1} \end{matrix} \right\|. \tag{3.65}$$

In the same way we can demonstrate (3.57).

The role played in Fast RLS by both the Kalman Gain vector and the input data vector is taken here by the V-vector $\mathbf{d}_n$ defined as

$$\mathbf{d}_n = \mathbf{R}_n^{-T} \mathbf{x}_n. \tag{3.66}$$

Obviously two different extended V-vectors $\mathbf{d}$ can be defined in correspondence to the V-matrices (3.58) and (3.59), as described in (3.67)-(3.68).

$$\widetilde{\mathbf{d}}_{n,i} = \widetilde{\mathbf{R}}_{n,i}^{-T} \cdot \left\| v_{n,i} \right\backslash_i \mathbf{x}_{n,i-1} \left/ \right\| = \left\| \alpha_{n,i}^{-1/2} f_{n,i}(n) \right\backslash_i \mathbf{d}_{n,i-1} \left/ \right\|, \tag{3.67}$$

$$\overline{\mathbf{d}}_{n,i} = \overline{\mathbf{R}}_{n,i}^{-T} \cdot \left\| \mathbf{x}_{n,i} \right/_i r_{n-1,i} \left/ \right\| = \left\| \mathbf{d}_{n,i} \right/_i \beta_{n,i}^{-1/2} b_{n,i}(n) \left/ \right\|. \tag{3.68}$$

From (3.68) we have that $\mathbf{d}_n$ is the normalized backward prediction error V-vector and that the identified filter is the same of a normalized lattice RLS algorithm [60].

All the above relations are proved to be correct whatever is the type M of the $M \times M$ V-matrix $\mathbf{R}$. In the rest of this paragraph however

$\mathbf{R}_{n,i}^T$ is a *row (i+1) MOD L Left Upper Triangular II V-matrix* such that starting form a row 1 MOD L LUT II V-matrix after L iterations we obtain again a row 1 MOD L LUT II V-matrix.

The V-matrices $\widetilde{\mathbf{R}}_{n,i}$ and $\overline{\mathbf{R}}_{n,i}$ do not coincide but differ by a rotation V-matrix $\mathbf{Q}_i$ ($\mathbf{Q}_i\mathbf{Q}_i^T = I$) and even the couples $\widetilde{\mathbf{R}}_{n,i}^{-T}$, $\overline{\mathbf{R}}_{n,i}^{-T}$ and $\tilde{\mathbf{d}}_{n,i}$, $\overline{\mathbf{d}}_{n,i}$ differ by the same V-matrix. If we determine $\mathbf{Q}_i$ that allows the passage from (3.56) to (3.57) then it is trivial to update $\mathbf{d}_{n,i}$ from $\mathbf{d}_{n,i-1}$. This rotation matrix will be decomposed into Givens rotations. Since the product $\mathbf{Q}_i\widetilde{\mathbf{R}}_{n,i}$ is a row $\times$ column product we have to proceed on the columns of $\widetilde{\mathbf{R}}_n$, *i.e.* the rows (subV-vectors) of $\widetilde{\mathbf{R}}_n^T$:

$$\overline{\mathbf{R}}_{n,i}^T = \left\| \begin{matrix} \mathbf{R}_{n,i}^T \\ -(\mathbf{R}_{n,i}\mathbf{b}_{n,i})^T \end{matrix} \right/ \begin{matrix} \mathbf{0} \\ \beta_{n,i}^{1/2} \end{matrix} \Big\|_i$$

$$= \widetilde{\mathbf{R}}_{n,i}^T\mathbf{Q}_i^T = \left\| \begin{matrix} \alpha_{n,i}^{1/2} \\ \mathbf{0} \end{matrix} \right\backslash \begin{matrix} -(\mathbf{R}_{n,i-1}\mathbf{a}_{n,i})^T \\ \mathbf{R}_{n,i-1}^T \end{matrix} \Big\|_i \cdot \mathbf{Q}_i^T \quad (3.69)$$

In particular we have to annihilate some elements of subV-vector

$$\left\backslash \alpha_{n,i}^{1/2} \right\backslash_i - \mathbf{R}_{n,i-1}\mathbf{a}_{n,i} \Big/ \quad (3.70)$$

preserving the row i MOD L LUT II structure of the remaining part of $\widetilde{\mathbf{R}}_{n,i}^T$ determined by $\mathbf{R}_{n,i-1}^T$. For this purpose we use as pivot element $\alpha_{n,i}^{1/2}$ and we have to rotate on this pivot all the elements at its right scanning V-vector (3.70) by right columns from right to left and (in a cyclic manner) from i-th row up to $(i+L-1)$ MOD L row (note that we stop scanning when we encounter the pivot).

After applying the Givens rotations, if we discard from $\overline{\mathbf{R}}_{n,i}^T$ the i-th row, last column subV-vector, and from every subV-vector the i-th row, last column element we obtain $\mathbf{R}_{n,i}^T$ which is a row $(i+1)$ MOD L LUT II V-matrix. Furthermore we can see that the update of $\mathbf{d}_{n,i}$ requires only the knowledge of the $\mathbf{Q}_i$ V-matrix, *i.e.* of V-vector (3.70) without building up $\widetilde{\mathbf{R}}_{n,i}^T$.

From the knowledge of $\mathbf{d}_{n,i}$ we can update all others parameters of the algorithm. The *a priori* forward prediction error can be written as:

$$\begin{aligned} f_{n-1,i}(n) &= v_{n,i} + \mathbf{a}_{n-1,i}^T\mathbf{R}_{n,i-1}^T\mathbf{R}_{n,i-1}^{-T}\mathbf{x}_{n,i-1} \\ &= v_{n,i} + \left(\mathbf{R}_{n,i-1}\mathbf{a}_{n-1,i}\right)^T\mathbf{d}_{n,i-1}. \end{aligned}$$

Let us call

$$\mathbf{z}_{n,i} = \mathbf{R}_{n,i-1}\mathbf{a}_{n,i}, \quad (3.71)$$

then we can show that

$$\mathbf{R}_{n,i-1}\mathbf{a}_{n-1,i} = \sqrt{\lambda}\mathbf{T}_{n,i-1}\mathbf{z}_{n-1,i}, \tag{3.72}$$

where $\mathbf{T}^T_{n,i-1}$ is the row i MOD L LUT II Cholesky factor of

$$\mathbf{I} + \gamma^{-1}_{n,i-1}\mathbf{d}_{n,i-1}\mathbf{d}^T_{n,i-1} = \mathbf{T}_{n,i-1}\mathbf{T}^T_{n,i-1}. \tag{3.73}$$

In fact, by omitting for simplicity the subscript $(i-1)$, we have

$$\begin{aligned}
\mathbf{R}^T_n\mathbf{R}_n &= \lambda\mathbf{R}^T_{n-1}\mathbf{R}_{n-1} + \mathbf{x}_n\mathbf{x}^T_n, \\
\lambda\mathbf{R}^T_{n-1}\mathbf{R}_{n-1} &= \mathbf{R}^T_n(\mathbf{I} - \mathbf{d}_n\mathbf{d}^T_n)\mathbf{R}_n = \\
&= \mathbf{R}^T_n\mathbf{T}^{-T}_n\mathbf{T}^{-1}_n\mathbf{R}_n, \\
\sqrt{\lambda}\mathbf{R}_{n-1} &= \mathbf{T}^{-1}_n\mathbf{R}_n, \\
\mathbf{R}_n &= \sqrt{\lambda}\mathbf{T}_n\mathbf{R}_{n-1}.
\end{aligned}$$

Being

$$\gamma_n = 1 - \mathbf{d}^T_n\mathbf{d}_n \tag{3.74}$$

and

$$(\mathbf{I} - \mathbf{d}_n\mathbf{d}^T_n)^{-1} = \mathbf{I} + \gamma^{-1}_n\mathbf{d}_n\mathbf{d}^T_n \tag{3.75}$$

we obtain immediately (3.73).

A computationally efficient procedure for the computation of (3.72) has been developed and is presented in Table 3.2. The derivation of this procedure is similar to that of the algorithms presented in Section 2.4.3. From (3.51) we immediately derive

$$\mathbf{z}_{n,i} = \sqrt{\lambda}\mathbf{T}_{n,i-1}\mathbf{z}_{n-1,i} - \mathbf{d}_{n,i-1}f_{n-1,i}(n). \tag{3.76}$$

The update of $\gamma_{n,i}$ is critical for the numerical stability of the algorithm. We have

$$\begin{aligned}
\overline{\gamma}_{n,i} &= 1 - \overline{\mathbf{d}}^T_{n,i}\overline{\mathbf{d}}_{n,i} = 1 - \mathbf{d}^T_{n,i}\mathbf{d}_{n,i} - \beta^{-1}_{n,i}b^2_{n,i}(n) = \\
&= 1 - \tilde{\mathbf{d}}^T_{n,i}\tilde{\mathbf{d}}_{n,i} = 1 - \mathbf{d}^T_{n,i-1}\mathbf{d}_{n,i-1} - \alpha^{-1}_{n,i}f^2_{n,i}(n),
\end{aligned}$$

and we can update $\gamma_{n,i}$ according to the following equation

$$\gamma_{n,i} = \gamma_{n,i-1} + \beta^{-1}_{n,i}b^2_{n,i}(n) - \alpha^{-1}_{n,i}f^2_{n,i}(n). \tag{3.77}$$

But if $\gamma_{n,i}$ is evaluated with (3.77) an error accumulation on the likelihood variable determines a long term numerical instability of the algorithm. This error accumulation can be interrupted recomputing γ_n after a certain amount of steps (for instance after L steps) as in equation (3.74).

$z = 0$

From the last column of V-vector to the first
From the i-th row to the $(i + L - 1)$ MOD L
Compute:

cd_{hk}
$l_{hk} = 1 + (cd_{hk})d_{hk}$
$l_{hk}^{1/2}$
$x_{hk}/l_{hk}^{1/2}$
$z = z + (cd_{hk})(x_{hk}/l_{hk}^{1/2})$
$c = c/l_{hk}$
$y_{hk} = (x_{hk}/l_{hk}^{1/2}) + d_{hk}z$

Table 3.2: Algorithm for the computation of the $\mathbf{y} = \mathbf{Tx}$ product, with $\mathbf{TT}^T = \mathbf{I} + c\mathbf{d}\mathbf{d}^T$ and $\mathbf{T}^T$ a row i MOD L LUT II V-matrix.

The use of this accommodation technique gives a numerically stable algorithm.

The desired signal estimation is produced by a joint process which act only after a complete update cycle of the prediction scheme. Proceeding with the same developments as for $\mathbf{z}_{n,i}$, the *a priori* estimation error is now given by

$$e_{n-1}(n) = d(n) - \mathbf{d}_{n,L}^T \sqrt{\lambda} \mathbf{T}_{n,L} \mathbf{h}_{n-1} \tag{3.78}$$

where

$$\mathbf{h}_n = \mathbf{R}_{n,L} \mathbf{w}_n \tag{3.79}$$

is the filter which estimate $d(n)$ from the normalized prediction errors, and it is

$$\mathbf{h}_n = \sqrt{\lambda} \mathbf{T}_{n,L} \mathbf{h}_{n-1} + \mathbf{d}_{n,L} e_{n-1}(n) \tag{3.80}$$

Furthermore, as $\mathbf{d}_n$ is the normalized backward prediction error filter, the joint process part coincides with that of normalized Lattice RLS and Fast QR algorithms [115]. So even if the filter coefficient vector is not directly evaluated, we can still apply this algorithm for system identification as well as prediction and filtering. In particular a direct dependency of the forward *a priori* prediction error at time n from the input sample at the same time makes the algorithm suitable for the

ADPCM application in signal coding [96], even if this direct dependency is payed with a not pipelineable structure due to the presence of sums of products.

For what regards the initialization of the algorithm we can choose

$$\begin{cases} \mathbf{d}_{1,0} = 0 \\ \mathbf{z}_{0,i} = 0 \\ \mathbf{h}_0 = 0 \\ \gamma_{1,0} = 1 \\ \alpha_{0,i} = \delta \quad \text{with} \quad \delta << 1. \end{cases}$$

This choice leads to a limited memory of initial conditions during the transitory convergence period, but even to sharply varying parameters in the same period, which can overflow the computational precision of processors especially in presence of limited wordlength. This problem can be avoided by taking

$$\alpha_{0,i} = \Delta \quad \text{with} \quad \Delta >> 1;$$

this initialization, in fact, gives slowly varying parameters during the transitory convergence period, which however is extended proportionally to Δ.

The final algorithm and operation count of all the equations is presented in Table 3.3. The computational burden of the algorithm in case of a strictly decreasing V descendant data vector is $(10+\frac{1}{3})LN_T$ multiplications, $(3+\frac{1}{3})LN_T$ divisions and $(1+\frac{2}{3})LN_T$ square-roots. The addition count is comparable to the multiplication count. By the use of an array of processors, however, filter adaptation can be performed in a limited $O(LN_T)$ number of machine cycles. The computational complexity of the algorithm presented in this section is similar to or lower than the computational complexity of the algorithms described in [146] and [121] as reported in Table 3.4.

A different formulation of the algorithm which does not require any square-root has been developed and it is the extension to Volterra filters of the algorithm reported in Theorem 2.4.2.

The numerical stability of the algorithm has been verified by several experiments with different types of data signals. A finite precision arithmetic was simulated as in [110] by implementing a floating point arithmetic with a mantissa of 16, 8 and 4 bits, respectively. The longest simulation performed with a 4 bits mantissa had more than 4 millions samples and in no one of all the considered simulations any instability has been observed.

Algorithm	$\times$	$\div$	$\surd$
$\mathbf{d}_{n,0} = \mathbf{d}_{n-1,L}$			
$\gamma_{n,0} = \gamma_{n-1,L}$			
For i=1 to L Compute			
$\sqrt{\lambda}\mathbf{T}_{n,i-1}\mathbf{z}_{n-1,i}$	$5LN_T$	$2LN_T$	LN_T
$f_{n-1,i}(n) = v_{n,i} + (\sqrt{\lambda}\mathbf{T}_{n,i-1}\mathbf{z}_{n-1,i})^T\mathbf{d}_{n,i-1}$	LN_T		
$f_{n,i}(n) = \gamma_{n,i-1}f_{n-1,i}(n)$			
$\mathbf{z}_{n,i} = \sqrt{\lambda}\mathbf{T}_{n,i-1}\mathbf{z}_{n-1,i} - \mathbf{d}_{n,i-1}f_{n-1,i}(n)$	LN_T		
$\alpha_{n,i} = \lambda\alpha_{n-1,i} + f_{n-1,i}(n)f_{n,i}(n)$			
$\mathbf{Q}_i$ from $\left\backslash \alpha_{n,i}^{1/2} \right\backslash_i - \mathbf{z}_{n,i} \Big/$	$\frac{2}{3}LN_T$		$\frac{2}{3}LN_T$
$\left\backslash \mathbf{d}_{n,i} \Big/_i \beta_{n,i}^{-1/2} b_{n,i}(n) \right\backslash =$			
$= \mathbf{Q}_i \cdot \left\backslash \alpha_{n,i}^{-1/2} f_{n,i}(n) \right\backslash_i \mathbf{d}_{n,i-1} \Big/$	$\frac{8}{3}LN_T$	$\frac{4}{3}LN_T$	
$\gamma_{n,i} = \gamma_{n,i-1} + (\beta_{n,i}^{-1/2} b_{n,i}(n))^2 +$			
$-(\alpha_{n,i}^{-1/2} f_{n,i}(n))^2$			
End For			
$\gamma_{n,L} = 1 - \mathbf{d}_{n,L}^T\mathbf{d}_{n,L}$	N_T		
$\sqrt{\lambda}\mathbf{T}_{n,L}\mathbf{h}_{n-1}$	$3N_T$	N_T	
$e_{n-1}(n) = d(n) - (\sqrt{\lambda}\mathbf{T}_{n,L}\mathbf{h}_{n-1})^T\mathbf{d}_{n,L}$	N_T		
$\mathbf{h}_n = \sqrt{\lambda}\mathbf{T}_{n,L}\mathbf{h}_{n-1} + \mathbf{d}_{n,L}e_{n-1}(n)$	N_T		
$e_n(n) = \gamma_{n,L}e_{n-1}(n)$			
TOTAL	$\frac{31}{3}LN_T$	$\frac{10}{3}LN_T$	$\frac{5}{3}LN_T$

Table 3.3: Algorithm for the computation of the Givens Rotation Based Fast SQR-RLS. In the second, the third and the fourth column is reported the main term of each operation cost in multiplications, divisions and square-roots.

	$\times$	$\div$	$\surd$
[146]	$12LN_{tot}$	$(3+\frac{1}{3})LN_{tot}$	$(1+\frac{2}{3})LN_{tot}$
[121]	$(10+\frac{1}{3})LN_{tot}$	$(3+\frac{1}{3})LN_{tot}$	$(1+\frac{1}{3})LN_{tot}$
NEW	$(10+\frac{1}{3})LN_{tot}$	$(3+\frac{1}{3})LN_{tot}$	$(1+\frac{2}{3})LN_{tot}$

Table 3.4: Comparison of the computational complexity of the algorithms in [146] and [121] with the computational complexity of the new algorithm.

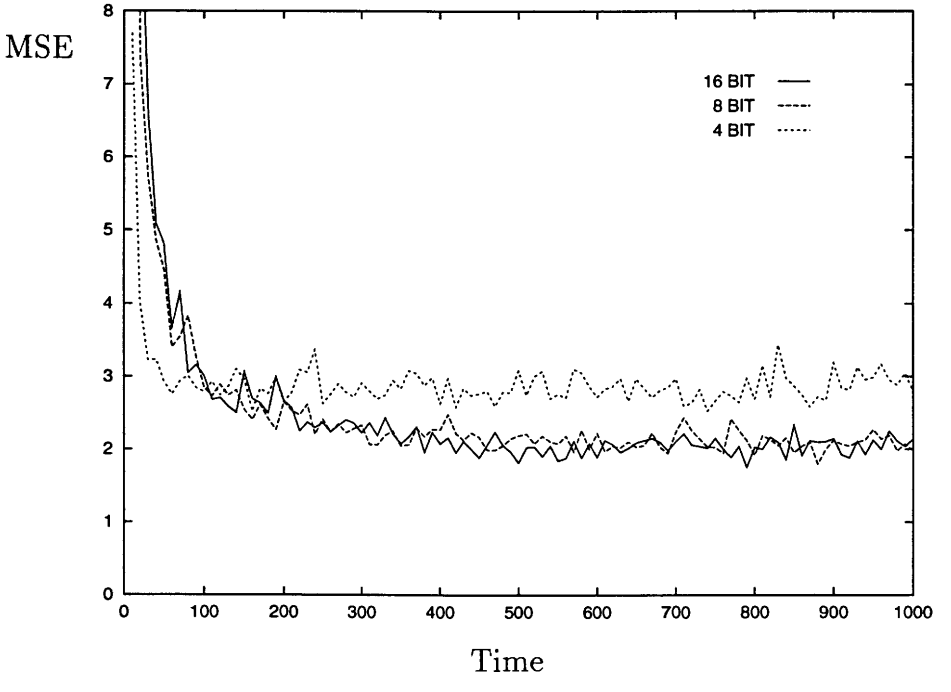


Figure 3.8: Arithmetic mean of the *a priori* forward prediction mean square error as a function of time. The arithmetic mean is evaluated on twenty different non-white-Gaussian noise signals while the mean square error is computed on data segments of 10 samples.

In Figure 3.8 is represented, as a function of time, the arithmetic mean of the *a priori* forward prediction mean square error. The arithmetic mean is evaluated on twenty different non-white-Gaussian noise signals while the mean square error is computed on data segments of 10 samples. All noise signals are obtained by filtering a zero mean, unit variance white Gaussian noise $N(n)$ with the cascade of two linear filters given by

$$x(n) = N(n) + 0.9x(n-1), \tag{3.81}$$
$$y(n) = 2x(n) + x(n-1) - 0.5x(n-2). \tag{3.82}$$

The different plots refers to different mantissa precision of the processor. Figure 3.8 illustrates how the wordlength affects the performances of the algorithm and shows the good convergence properties even with a low mantissa precision. The same performance can be obtained both from an 8 bit mantissa wordlength and the standard floating point precision.

3.6 Conclusions

A novel algebraic structure based on non-rectangular matrices has been developed. All fundamental objects and basic operations of this algebra have been presented in this chapter. The V-vector algebra has revealed a powerful tool to cope with adaptation algorithms for Volterra filters. With this algebra, in fact, Volterra filters are treated exactly in the same way as classical linear filters. Furthermore V-vector algebra, even though initially developed for Volterra filters, can be applied also to linear multichannel systems with different memory length in the various channels.

As a matter of fact, it is worth noting that the RLS algorithms developed taking into account the permutation between the vectors in (3.7)-(3.8) coincide with the RLS algorithms developed with the V-vector algebra. However, the permutation in (3.7)-(3.8) has to be designed *ad hoc* for each one of the considered filters. On the contrary, the developments of the V-vector formalism are the same whichever the Volterra or linear multichannel adaptive filters may be.

These considerations make it possible to devise, in addition to the easy extension of adaptive algorithms already known for linear filters, the derivation of new powerful adaptation techniques for Volterra or linear multichannel filters.

Appendices

3.A The Fundamental Operations between V-Matrices

The aim of this appendix is to show how to compute the principal operations between the V-matrix structures defined in Section 3.5. First of all it is straightforward to show by the definition of the transposed V-matrix that

$$\left\| \begin{matrix} a & \mathbf{b}^T \\ \mathbf{c} & \mathbf{D} \end{matrix} \right\|^T = \left\| \begin{matrix} a & \mathbf{c}^T \\ \mathbf{b} & \mathbf{D}^T \end{matrix} \right\| \tag{3.83}$$

$$\left\| \begin{matrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^T & d \end{matrix} \right\|^T = \left\| \begin{matrix} \mathbf{A}^T & \mathbf{c} \\ \mathbf{b}^T & d \end{matrix} \right\| \tag{3.84}$$

Then the product between two V-matrices is given by

$$\left\| \begin{matrix} a & \mathbf{b}^T \\ \mathbf{c} & \mathbf{D} \end{matrix} \right\| \left\| \begin{matrix} e & \mathbf{f}^T \\ \mathbf{g} & \mathbf{H} \end{matrix} \right\| = \left\| \begin{matrix} ae + \mathbf{b}^T\mathbf{g} & a\mathbf{f}^T + \mathbf{b}^T\mathbf{H} \\ \mathbf{c}e + \mathbf{D}\mathbf{g} & \mathbf{c}\mathbf{f}^T + \mathbf{D}\mathbf{H} \end{matrix} \right\| \tag{3.85}$$

$$\left\| \begin{matrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^T & d \end{matrix} \right\| \left\| \begin{matrix} \mathbf{E} & \mathbf{f} \\ \mathbf{g}^T & h \end{matrix} \right\| = \left\| \begin{matrix} \mathbf{A}\mathbf{E} + \mathbf{b}\mathbf{g}^T & \mathbf{A}\mathbf{f} + \mathbf{b}h \\ \mathbf{c}^T\mathbf{E} + d\mathbf{g}^T & \mathbf{c}^T\mathbf{f} + dh \end{matrix} \right\| \tag{3.86}$$

This is a row per column product: if we make the product between every subV-vector of the first factor and of the transposed of the second factor we obtain immediately (3.85) and (3.86). Finally, the inverse V-matrices are obtained by solving the following systems

$$\left\| \begin{matrix} a & \mathbf{b}^T \\ \mathbf{c} & \mathbf{D} \end{matrix} \right\| \left\| \begin{matrix} u & \mathbf{v}^T \\ \mathbf{w} & \mathbf{Z} \end{matrix} \right\| = \left\| \begin{matrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{I} \end{matrix} \right\| \tag{3.87}$$

$$\left\| \begin{matrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^T & d \end{matrix} \right\| \left\| \begin{matrix} \mathbf{U} & \mathbf{v} \\ \mathbf{w}^T & z \end{matrix} \right\| = \left\| \begin{matrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{matrix} \right\| \tag{3.88}$$

Chapter 4

Sufficient Stability Bounds for Slowly-Varying Direct-Form Recursive Linear Filters and Their Applications in Adaptive IIR Filters

4.1 Introduction

Another interesting area in adaptive signal processing is represented by adaptive IIR filtering. Adaptive IIR filters have been the subject of active research over the last three decades [67, 91, 102, 119, 130]. Despite a large amount of work that has been done, some open issues still remain. One of these issues is that of ensuring the stability of the time-varying IIR filter that results from the identification process.

Researchers have attempted to derive adaptive IIR filters that operate in a stable manner in several different ways. One class of algorithms is obtained by means of the equation-error technique. In the equation-error technique, the IIR filter is identified by the use of a two-channel

---

Part of the content of this chapter was presented in
Alberto Carini, V. John Mathews e Giovanni L. Sicuranza, "Sufficient Stability Bounds for Slowly Varying Discrete-Time Recursive Linear Filters," *Proceedings of ICASSP 97*, April 21-24 1997, Munich, Germany
and submitted to *IEEE Trans. Signal Processing* in 1996

adaptive FIR filter that operates on samples of the input and the desired response signals. Since the system model employed in equation-error methods is not recursive, the adaptive filter can operate in a stable manner when the step size is properly selected. However, this fact does not ensure the stability of the resulting IIR filter. Moreover, it is well-known that equation-error adaptive algorithms give biased solutions when the desired response signal is corrupted by noise.

Output error algorithms have become popular in adaptive IIR filtering research in recent years. In output error techniques, the adaptive filter operates in a recursive manner on the input signal to provide an estimate of the desired response signal. A class of such methods requires a certain system transfer function to be strictly positive real (SPR) in order to avoid problems with instability and to ensure the convergence of the algorithm. This class of algorithms includes the pseudo-linear regression algorithm (PRA) [45], also known as Feintuch's algorithm, Landau's algorithm [77], the hyperstable adaptive recursive filter (HARF) [65] and the simplified HARF (SHARF) [80]. An SPR condition is not easy to guarantee in practice. In the PRA, the SPR condition limits the range of the location of the poles of the unknown system for which convergence is guaranteed. This problem is avoided in HARF and SHARF algorithms. However, some *a priori* knowledge of the underlying system model is required in order to meet the SPR condition. Moreover, if the system consists of two or more parallel sections, the SPR condition is not sufficient to guarantee stability [156]. A second class of adaptive output-error direct-form filters employ stability monitoring by checking the location of the instantaneous poles of the system and projecting the coefficients back to a region for which the instantaneous poles are within the unit circle [91]. Unfortunately, time-varying filters may be unstable even when the instantaneous poles are within the unit circle. A simple example is given by the time-varying recursive linear system with two coincident poles located at $(-1)^{k-1}0.5$ at time k with input-output relationship

$$y(k) = (-1)^{k-1}y(k-1) - 0.25y(k-2) + x(k). \tag{4.1}$$

Even though the instantaneous poles are always bounded by one and far from the unit circle itself, it is straightforward to show that the response of this system to a unit impulse signal diverges exponentially. Consequently, even though projection-based techniques that force the instantaneous poles of the system to stay within the unit circle work well in a large number of situations, they are not guaranteed to operate in a stable manner in all situations. A third class of output-error algorithms employs lattice structures [93, 119]. Normalized lattice filters are

guaranteed to be stable if the reflection coefficients are bounded by one. Similar conditions can be established also for other filter structures such as power wave digital filters [76] and normal forms [119]. However, direct form filters are particularly suited for the multiply-accumulate architectures found in most digital signal processors, and for this reason are often preferred to the above-mentioned filter structures.

In this chapter, we present a method for controlling the adaptation step size to guarantee bounded-input bounded-output stability of output-error adaptive IIR filters. It is well-known [6, 35, 119, 122, 123] that a recursive time-varying homogeneous linear system is exponentially stable if its instantaneous poles are always inside the unit circle and if they are sufficiently slowly-varying. We first derive a new upperbound on the maximum allowable coefficient variation for the stability of a direct-form linear recursive filter, and then apply the results to control the step size of an adaptive IIR filter to ensure stable operation. Experimental results demonstrating the good convergence characteristics of the adaptive filter so derived as well as comparing our stability bound with previously available results are also included in the chapter.

4.2 Sufficient Conditions for the Stability of Slowly-Varying Direct-Form Recursive Systems

We consider a time-varying recursive linear system with input-output relationship given by

$$y(k) = \sum_{i=0}^{N-1} b_i(k)x(k-i) + \sum_{i=1}^{N-1} a_i(k)y(k-i). \tag{4.2}$$

Let

$$\boldsymbol{\theta}(k) = [b_0(k), b_1(k), \ldots, b_{N-1}(k), a_1(k), a_2(k), \ldots, a_{N-1}(k)]^T \tag{4.3}$$

denote the coefficient vector and let the evolution of the coefficients be of the form

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \mu_k \boldsymbol{\psi}(k), \tag{4.4}$$

where μ_k is a time-varying scalar sequence. Our objective is to find a sufficient bound on the squared-norm of the increment vector $\mu_k\boldsymbol{\psi}(k)$ given by $\mu_k^2\boldsymbol{\psi}^T(k)\boldsymbol{\psi}(k)$ such that the time-varying system of (4.2) is stable in the bounded-input, bounded-output (BIBO) sense. From such a result,

we then find a bound on μ_k for guaranteeing the stability of the system. An adaptive filter with coefficient update as in (4.4) will be BIBO stable if μ_k is chosen smaller than or equal to such a bound. The basis for our work is the following theorem proved in [123]:

Theorem 4.2.1 *The linear state equation*

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k), \quad \mathbf{x}(k_0) = \mathbf{x}_0 \tag{4.5}$$

is uniformly exponentially stable if and only if there exists an $N \times N$ matrix sequence $\mathbf{Q}(k)$ that is symmetric for all k and such that

$$\eta\mathbf{I} \leq \mathbf{Q}(k) \leq \rho\mathbf{I} \tag{4.6}$$

and

$$\mathbf{A}^T(k+1)\mathbf{Q}(k+1)\mathbf{A}(k+1) - \mathbf{Q}(k) \leq -\phi\mathbf{I}, \tag{4.7}$$

where η, ρ and ϕ are finite positive constants.

The condition "matrix $\mathbf{Q} \leq \rho\mathbf{I}$" in the theorem implies that $\mathbf{x}^T\mathbf{Q}\mathbf{x} \leq \rho\mathbf{x}^T\mathbf{x}$ for all vectors $\mathbf{x}$. Exponential stability of the homogeneous system implies BIBO stability of the more general system in (4.2) provided that the coefficients of the non-recursive part are bounded [119].

Theorem 4.2.1 is expressed in terms of the state transition matrix $\mathbf{A}(k)$ while we are interested in the direct form realization. However, it is trivial to transform the direct form representation in (4.2) to the state space representation by considering the state vector

$$\mathbf{x}(k) = [y(k-1), y(k-2), \ldots, y(k-N)]^T \tag{4.8}$$

and the corresponding state transition matrix

$$\mathbf{A}(k) = \begin{bmatrix} a_1(k) & a_2(k) & \ldots & a_{N-1}(k) & a_N(k) \\ 1 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}. \tag{4.9}$$

Finite and bounded choices of the feedforward coefficients $b_i(k)$ do not affect the stability of the system. Consequently, ignoring these coefficients in the rest of the analysis does not result in any loss of generality. In the derivations that follow, we find a sequence of candidate matrices $\mathbf{Q}(k)$ that meets the Lyapunov conditions given by (4.6) and (4.7) for slowly-varying recursive linear filters. We assume that the instantaneous poles of the recursive system are always inside the unit circle.

4.2.1 The Lyapunov Candidate Sequence

Since the poles of the system (4.2) are by hypothesis always inside the unit circle, we can consider as Lyapunov candidate the unique, symmetric and positive definite solution of the discrete-time Lyapunov equation

$$\mathbf{A}^T(k)\mathbf{Q}_{\nu_k}(k)\mathbf{A}(k) - \mathbf{Q}_{\nu_k}(k) = -\nu_k \mathbf{I}_N, \tag{4.10}$$

where ν_k is a bounded positive sequence with $\nu_k \in [\nu_{min}, \nu_{max}]$ and $[\nu_{min}, \nu_{max}]$ is an arbitrary interval of the positive real axis.

Following the derivations in [35, 123], it can be shown that $\mathbf{Q}_{\nu_k}(k)$ is given by

$$\mathbf{Q}_{\nu_k}(k) = \nu_k \sum_{i=0}^{+\infty} [\mathbf{A}^T(k)]^i \mathbf{A}^i(k). \tag{4.11}$$

The closed form solution for $\mathbf{Q}_{\nu_k}(k)$ is given by

$$\text{vec}[\mathbf{Q}_{\nu_k}(k)] = -\nu_k \left[\mathbf{A}^T(k) \otimes \mathbf{A}^T(k) - \mathbf{I}_{N^2}\right]^{-1} \text{vec}[\mathbf{I}_N], \tag{4.12}$$

where $\otimes$ indicates the Kronecker product and "vec[$\mathbf{Q}$]" is a vector operator that stores the columns of $\mathbf{Q}$ in a predetermined order.

Since ν_k is bounded by the positive and finite values ν_{min} and ν_{max} and the eigenvalues of $A(k)$ are in the open set $(0, 1)$, it can be shown that the Lyapunov conditions in (4.6) are always satisfied. This result can be proved easily by following the derivations in [35, 123]. As for the condition in (4.7), let us consider (4.10) at time $k+1$ and add $\mathbf{Q}_{\nu_{k+1}}(k+1) - \mathbf{Q}_{\nu_k}(k)$ to both sides of the expression. It follows trivially that the condition of (4.7) can be rewritten as

$$\begin{aligned} &\mathbf{A}^T(k+1)\mathbf{Q}_{\nu_{k+1}}(k+1)\mathbf{A}(k+1) - \mathbf{Q}_{\nu_k}(k) = \\ &= \mathbf{Q}_{\nu_{k+1}}(k+1) - \mathbf{Q}_{\nu_k}(k) - \nu_{k+1}\mathbf{I}_N \leq -\phi \mathbf{I}_N. \end{aligned} \tag{4.13}$$

This condition is met if

$$\|\mathbf{Q}_{\nu_{k+1}}(k+1) - \mathbf{Q}_{\nu_k}(k)\| \leq \xi \nu_{k+1}. \tag{4.14}$$

where ξ is a real positive constant, $\xi < 1$ and $\|(\cdot)\|$ is the induced L_2 norm of the matrix $(\cdot)$. Dividing both sides of (4.14) by ν_{k+1}, (4.14) becomes

$$\|\mathbf{Q}_1(k+1) - \nu_{k+1}^{-1}\mathbf{Q}_{\nu_k}(k)\| \leq \xi < 1, \tag{4.15}$$

for all k, where $\mathbf{Q}_1(k)$ is given by

$$\mathbf{Q}_1(k) = \sum_{i=0}^{+\infty} [\mathbf{A}^T(k)]^i \mathbf{A}^i(k). \tag{4.16}$$

The above condition can be used to enable BIBO stable operation of an adaptive IIR filter equipped with a projection technique. Any projection technique that can move the coefficients back to a space that satisfies the condition in (4.15) can be used for this purpose. However, since most projection techniques have unpredictable computational complexity, and since they may result in coefficient stalling, we now derive a closed-form bound for μ_k when the parameter $\nu_k = 1$, under the assumption that the coefficient vary slowly. In this case the inequality in (4.15) reduces to

$$\|\mathbf{Q}_1(k+1) - \mathbf{Q}_1(k)\| \leq \xi < 1 \tag{4.17}$$

for all k. Since ν_k is fixed, we drop the subscript on $\mathbf{Q}$ such that $\mathbf{Q}(k) = \mathbf{Q}_1(k)$ from now onwards.

To derive the result, we note that

$$\|\mathbf{Q}(k+1) - \mathbf{Q}(k)\| \leq \|\text{vec}[\mathbf{Q}(k+1)] - \text{vec}[\mathbf{Q}(k)]\|. \tag{4.18}$$

Combining (4.17) and (4.18), we derive a sufficient condition for the exponential stability of the system in (4.5) to be

$$\|\text{vec}[\mathbf{Q}(k+1)] - \text{vec}[\mathbf{Q}(k)]\| \leq \xi < 1, \tag{4.19}$$

for all k. In the hypothesis of slowly-varying coefficients, the following approximation can be applied:

$$\text{vec}[\mathbf{Q}(k+1)] - \text{vec}[\mathbf{Q}(k)] \simeq \nabla_{\boldsymbol{\theta}} \text{vec}[\mathbf{Q}(k)] \cdot \Delta\boldsymbol{\theta}(k), \tag{4.20}$$

where $\nabla_{\boldsymbol{\theta}}$ indicates the gradient vector operator with respect to the coefficient vector $\boldsymbol{\theta}$ and $\Delta\boldsymbol{\theta}(k) = \boldsymbol{\theta}(k+1) - \boldsymbol{\theta}(k)$. Recall from (4.4) that $\Delta\boldsymbol{\theta}(k) = \mu_k \boldsymbol{\psi}(k)$. Substituting the approximation of (4.20) in (4.19) and manipulating the resulting expression gives an explicit condition on μ_k for the stability of (4.5) to be

$$\mu_k \leq \frac{\xi}{\|\nabla_{\boldsymbol{\theta}} \text{vec}[\mathbf{Q}(k)] \cdot \boldsymbol{\psi}(k)\|}, \tag{4.21}$$

for all k.

The stability conditions of (4.15) and (4.17) are derived without resorting to any approximation. However, these conditions can be employed in adaptive recursive filtering applications only with the help of projection techniques. Even though the derivation of (4.21) employs an approximation that is based on slow variations in the coefficients, this condition has the advantage of being useful in directly controlling the step size of adaptation.

The Second-Order Case

The stability conditions derived in the previous subsection hold for any filter order. Computationally simple expressions that relate the filter coefficients to the stability bound can be derived for the second-order case. Therefore, the implementation of the stability conditions is simplest when the adaptive filter is realized as a cascade or parallel connection of second-order sections. Since the non-recursive part does not affect the stability of the resulting system provided that the coefficients are bounded, we consider the following second-order filter:

$$y(k) = a_1(k)y(k-1) + a_2(k)y(k-2) + x(k). \tag{4.22}$$

For the instantaneous poles of this system to be inside the unit circle, the coefficients must satisfy the inequalities

$$|a_1(k)| + a_2(k) < 1 \tag{4.23}$$

and

$$a_2(k) > -1. \tag{4.24}$$

The candidate Lyapunov matrix $\mathbf{Q}(k)$ in (4.16) can be shown to be given by

$$\mathbf{Q}(k) = \begin{bmatrix} 2\frac{a_2(k)-1}{r(k)} & -2\frac{a_1(k)a_2(k)}{r(k)} \\ -2\frac{a_1(k)a_2(k)}{r(k)} & \frac{s(k)}{r(k)} \end{bmatrix}, \tag{4.25}$$

where

$$r(k) = -a_2^3(k) + a_2^2(k) + a_1^2(k)a_2(k) + a_2(k) + a_1^2(k) - 1 \tag{4.26}$$

and

$$s(k) = a_2^3(k) - a_2^2(k) + a_1^2(k)a_2(k) + a_2(k) + a_1^2(k) - 1. \tag{4.27}$$

Substituting (4.25) into (4.21) results in the following bound for μ_k for the second-order system:

$$\mu_k \leq \frac{r^2(k)}{\sqrt{\begin{array}{c} 4(r(k)\cdot\ \psi_2(k) - (a_2(k)-1)\cdot v(k))^2 \\ +8((a_1(k)\ \psi_2(k) + a_2(k)\ \psi_1(k))\cdot r(k) - a_1(k)a_2(k)\cdot v(k))^2 \\ +(r(k)\cdot w(k) - v(k)\cdot s(k))^2 \end{array}}}, \tag{4.28}$$

where

$$w(k) = (3a_2^2(k) + a_1^2(k) - 2a_2(k) + 1)\ \psi_2(k) + 2a_1(k)(a_2(k)+1)\ \psi_1(k) \tag{4.29}$$

and

$$v(k) = (-3a_2^2(k) + 2a_2(k) + a_1^2(k) + 1)\psi_2(k) + 2a_1(k)(a_2(k)+1)\psi_1(k). \tag{4.30}$$

4.3 Stabilized Output Error Adaptive IIR Filters

In this section, we apply the stability condition derived in Section 4.2 to the stabilization of output-error adaptive IIR filters. Even though the ideas presented here are applicable to almost all adaptive IIR filters, we describe our approach using the Gauss-Newton output error adaptation algorithm [91]. Furthermore, since the implementation of the stability condition is simplest when the adaptive filter is realized as a cascade or parallel connection of second-order sections, we have considered adaptive IIR filters employing parallel second-order sections.

Each second-order section is of the form

$$y_i(k) = a_{1i}(k)y_i(k-1)+a_{2i}(k)y_i(k-2)+b_{0i}u(k)+b_{1i}u(k-1)+b_{2i}u(k-2). \tag{4.31}$$

where $u(k)$ denotes the input to the section. Let the data vector and the coefficient vector of the i-th section be given by[1]

$$\mathbf{X}_i(k) = [u(k), u(k-1), u(k-2), y_i(k-1), y_i(k-2)]^T \tag{4.32}$$

and

$$\boldsymbol{\theta}_i(k) = [b_{0i}(k), b_{1i}(k), b_{2i}(k), a_{1i}(k), a_{2i}(k)]^T, \tag{4.33}$$

respectively. We define the data and coefficient vectors for the overall structure to be

$$\mathbf{X}(k) = [\mathbf{X}_1^T(k), \mathbf{X}_2^T(k), \ldots, \mathbf{X}_L^T(k)]^T \tag{4.34}$$

and

$$\boldsymbol{\theta}(k) = [\boldsymbol{\theta}_1^T(k), \boldsymbol{\theta}_2^T(k), \ldots, \boldsymbol{\theta}_L^T(k)]^T, \tag{4.35}$$

respectively, where L denotes the number of parallel sections.

The coefficients are updated in this method as

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \boldsymbol{\mu}(k)\mathbf{R}^{-1}(k+1)\boldsymbol{\psi}(k)e(k) \tag{4.36}$$

where $\boldsymbol{\mu}(k)$ is a time-varying step size matrix of the adaptive filter defined as

$$\boldsymbol{\mu}(k) = \text{diag}[\mu_1(k), \mu_2(k), \ldots, \mu_{5L}(k)], \tag{4.37}$$

$e(k)$ is the *a priori* estimation error defined as

$$e(k) = d(k) - \sum_{i=1}^{L} y_i(k), \tag{4.38}$$

[1]Note the exception in our notation. We use the capital letter $\mathbf{X}$ to indicate the data vector because $\mathbf{x}$ was used in Section 4.2 in order to indicate the state vector.

and

$$\boldsymbol{\psi}(k) = \left[\boldsymbol{\psi}_1^T(k), \boldsymbol{\psi}_2^T(k), \ldots, \boldsymbol{\psi}_L^T(k)\right]^T \tag{4.39}$$

is the information vector whose i-th vector element $\boldsymbol{\psi}_i(k)$ is

$$\boldsymbol{\psi}_i(k) = \frac{1}{1 - a_{1i}(k)q^{-1} - a_{2i}(k)q^{-2}}\mathbf{X}_i(k). \tag{4.40}$$

In the above expression, the notation q^{-1} refers to a unit delay operator. The matrix $\mathbf{R}(k)$ is an estimate of the autocorrelation matrix of the information vector and it is recursively computed as

$$\mathbf{R}(k) = \lambda\mathbf{R}(k-1) + (1-\lambda)\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k), \tag{4.41}$$

where $0 \ll \lambda < 1$ is a parameter that controls the convergence and tracking speed of the estimation of the autocorrelation matrix. Its inverse may be evaluated recursively using the matrix inversion lemma as

$$\mathbf{R}^{-1}(k+1) = \frac{1}{\lambda}\left(\mathbf{R}^{-1}(k) - \frac{\mathbf{R}^{-1}(k)\boldsymbol{\psi}(k)\boldsymbol{\psi}^T(k)\mathbf{R}^{-1}(k)}{\frac{\lambda}{1-\lambda} + \boldsymbol{\psi}^T(k)\mathbf{R}^{-1}(k)\boldsymbol{\psi}(k)}\right). \tag{4.42}$$

The BIBO stability of the above adaptive filter can be achieved by constraining the step sizes associated with the recursive component of each section to meet the conditions specified by (4.28). We point out again that the instantaneous poles must always lie within the unit circle and that the coefficients of the feedforward part must be bounded. In all our experiments, we have also limited the maximum step size value. Doing so has two advantages: *(i)* it allows the designer to control the steady-state behaviour of the adaptive filter independently of the characteristics of the adaptive filter coefficients, and *(ii)* it ensures that the coefficients vary slowly so that the approximations in the derivation are valid.

The computational complexity of calculating the step size bound in (4.28) correspond to 16 multiplications, one square-root operation and one division per second-order section. Consequently, the complexity of implementing the stability bounds for a cascade or parallel adaptive filter is linearly proportional to the order of the filter. Furthermore, this complexity is comparable to or smaller than the complexity of adapting the coefficients in many adaptive IIR filtering algorithms.

4.4 Experimental Results

In the first set of results presented below, the adaptive filters were employed to identify an unknown, fourth-order IIR filter with transfer func-

tion

$$H(z) = \frac{1}{1 - 1.86z^{-1} + 0.8698z^{-2}} + \frac{2}{1 - z^{-1} + 0.5z^{-2}} \tag{4.43}$$

using measurements of the input and output signals. The poles of the above system are located at $[0.93 \pm 0.07j]$ and $[0.5 \pm 0.5j]$. The adaptive filters employed a parallel connection of two second-order systems and were adapted using the Gauss-Newton algorithm. The input of the unknown system is a coloured Gaussian signal with zero mean value obtained by filtering a white Gaussian signal with zero mean value and unit variance with the FIR filter of transfer function given by

$$W(z) = 1 + 0.5z^{-1}. \tag{4.44}$$

The desired response signal was generated by processing this signal with the unknown system and then corrupting the output with an additive, zero-mean and white Gaussian noise sequence that is statistically independent of the input signal. The variance of the measurement noise was such that the output signal-to-noise ratio was 30 dB. The adaptive filter employed a different step size sequence for each second-order section and for the recursive and non-recursive part of each section. The step size of the recursive part was selected to be the minimum of 0.001 or the bound suggested by our conditions, while that of the moving average part was fixed at 0.0005. The forgetting factor in the evaluation of the inverse of the autocorrelation matrix was chosen to be 0.9999. Almost all output error adaptive recursive filters are susceptible to converging to the wrong local minima of the squared estimation error surface. In the results presented here, all the experiments that resulted in convergence to wrong local minima were eliminated from the calculation of the ensemble averages. In this way, we are able to observe the speed of convergence of the adaptive filter when it converged to the true solution. The results displayed in the figures are averages of the first fifty experiments in which the coefficients converged to the correct solution.

In addition to constraining the step size to values below the stability bound at each time, we must also verify that the instantaneous poles of the updated filter are within the unit circle. In the experiments described below the updates for a particular iteration was simply skipped whenever one or more poles crossed the unit circle. In order to ensure the BIBO stability of the adaptive filter feedforward part, we also imposed an upper bound on the absolute value of feedforward coefficients. The upper bound chosen was 1000, and in no experiment the feedforward coefficients reached this bound. The algorithm was initialized with the

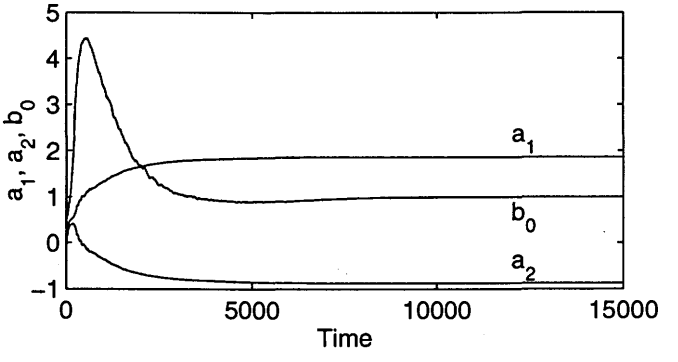


Figure 4.1: Evolution of the feedback coefficients of one of the parallel sections of the stabilized adaptive IIR filters.

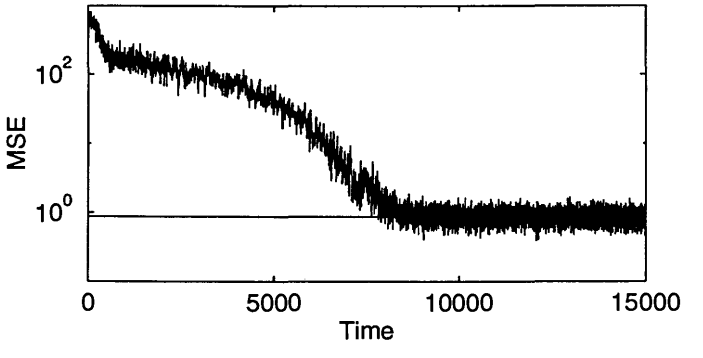


Figure 4.2: Evolution of the mean-square estimation error in the stabilized adaptive IIR filters.

coefficients of the feedforward parts equal to 0.5 and the poles of the recursive part equal to $[0.1 \pm 0.1i]$ and $[-0.1 \pm 0.1i]$ respectively for each second-order section.

Figure 4.1 shows the ensemble averaged behaviour of the coefficients of the parallel section that correspond to the coefficients 1.86 and 0.8698 (corresponding to the poles located at $[0.93 \pm 0.07j]$) of the unknown system. Figure 4.2 shows the ensemble averaged, squared estimation error at the output of the adaptive filters. The horizontal line in the figure represents the noise floor. Figure 4.3 displays the ensemble averaged step size sequence for the parallel section tracking the poles of the unknown system at $[0.93 \pm 0.07j]$. The results indicate that step size selection using the closed form conditions in (4.28) results in stable operation of

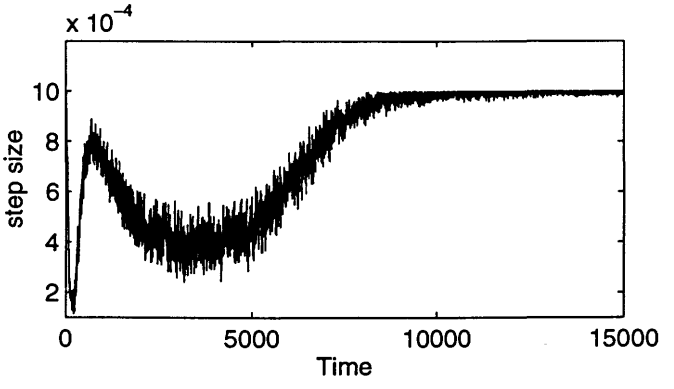


Figure 4.3: Evolution of the step size sequence of the stabilized adaptive IIR filters.

the adaptive filter. The initial values of the step size are small in this example because the initial estimation error is large. Combined with the large error, the initial values of the step size produced the largest changes possible that still maintained the exponential stability of the system.

We now compare the performance of the stabilized adaptive IIR filter with that of the SHARF algorithm. In order to make the comparisons as fair as possible, we used a single second order system for identifying an unknown second-order system with transfer function

$$H(z) = \frac{1}{1 - 1.9z^{-1} + 0.905z^{-2}}. \tag{4.45}$$

We used the same experimental conditions as in the previous example, with the difference that we employed the same step size sequence for adapting the moving average and the recursive coefficients of the system. The coefficient update in the SHARF algorithm was implemented as in [130] in the following manner:

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \mu \mathbf{R}^{-1}(k+1)\mathbf{X}(k)e_f(k), \tag{4.46}$$

where

$$\mathbf{R}^{-1}(k+1) = \frac{1}{\lambda}\left(\mathbf{R}^{-1}(k) - \frac{\mathbf{R}^{-1}(k)\mathbf{X}(k)\mathbf{X}^T(k)\mathbf{R}^{-1}(k)}{\frac{\lambda}{1-\lambda} + \mathbf{X}^T(k)\mathbf{R}^{-1}(k)\mathbf{X}(k)}\right), \tag{4.47}$$

$$e(k) = d(k) - \boldsymbol{\theta}^T(k)\mathbf{X}(k), \tag{4.48}$$

and

$$e_f(k) = \sum_{m=0}^{p-1} c(m)e(k-m). \tag{4.49}$$

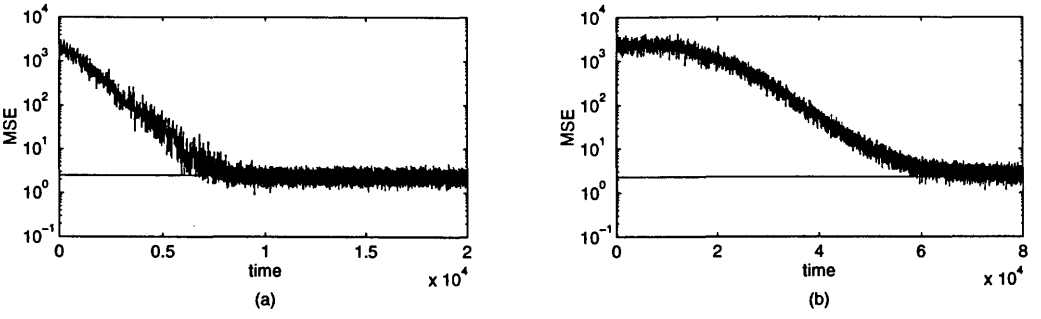


Figure 4.4: Comparison of the stabilized adaptive filter of this chapter and the SHARF algorithm.

The input vector $\mathbf{X}(k)$ is defined in this case as $[x(k), y(k-1), y(k-2)]^T$. In order to obtain satisfactory convergence of the SHARF algorithm, the FIR filter with transfer function $C(z)$ that gives the filtered estimation error $e_f(k)$, must be such that $C(z) \simeq A(z)$, where $A(z)$ is the denominator of the transfer function of the unknown filter. In our case we have considered the optimal choice of $C(z) = A(z)$.

Both the algorithms were initialized with the coefficient of the feedforward part equal to 0.5 and with the two poles at the origin.

The step size μ was selected to be 0.00008 for the SHARF algorithm so that the steady-state excess mean-square error was identical to that of the stabilized adaptive IIR filter of this chapter. We note that the coefficient update equations (4.46) and (4.47) have the form of a Gauss-Newton update. The similarity of this set of update equations to those in (4.36)–(4.42) and the choices of the step sizes so that the steady-state errors are almost identical make it possible to make fair comparisons of the performance of the two algorithms.

Figure 4.4 plots the evolution of the mean-squared estimation error for the two algorithms. We can see from this figure that the SHARF algorithm converges much slower than the method introduced in this chapter. Note that the time scales used in the two plots are different from each other. In general, when the instantaneous poles are initialized to be sufficiently removed from the unit circle, we have observed that our method converges significantly faster than the SHARF. However, it is possible to slow the initial convergence rate of our method by initializing the instantaneous poles to be very close to the unit circle. Such an initialization will force the initial values of the step size to be very small, and therefore will result in slow convergence.

We also studied the convergence behaviour of the Gauss-Newton output error algorithm with fixed step size using the same experimental

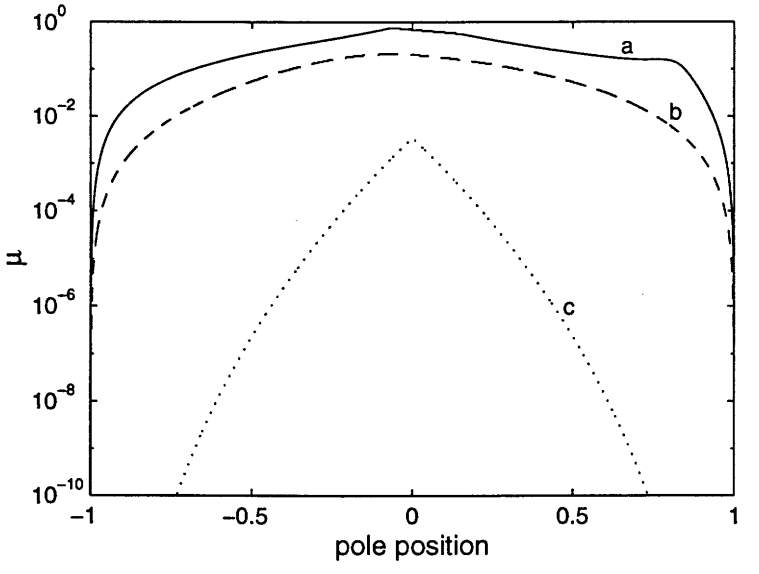


Figure 4.5: Comparison of the bound in (4.28) with that derived in [6] and [35].

conditions. Without pole projection inside the unit circle, the algorithm became unstable in all 50 experiments we performed with a step size equal to $6 \cdot 10^{-4}$, which was lower than the maximum step size value we allowed for the stabilized adaptive filter. With pole projection and a step size equal to $8 \cdot 10^{-4}$, the instantaneous poles moved to locations outside the unit circle so often that the overall speed of convergence was much slower than that of the stabilized algorithm. Furthermore, the evolution of the coefficients toward their steady-state values was very erratic for the method employing only pole projection. In our experiments with the adaptive filter employing the new step size bound, only in one of the fifty-one experiments did the coefficients of the system not converge to the correct coefficient values after 20,000 samples. With the fixed step size, the coefficients in sixteen of sixty-six experiments did not converge to the correct values during the same time span.

Finally, we compare the bounds given by (4.28) with the bounds derived in [6] and [35] for the maximum allowable variations in the coefficients of an exponentially stable, second-order linear system with time-varying coefficients. The stability bounds in [6] and [35] are expressed in terms of the state transition matrix. In order to make the comparison as fair as possible we derived the maximum allowable coefficient variation for the state transition matrix defined in (4.9). Figure 4.5 displays the three bounds as a function of the magnitude of the complex instantaneous

poles of the system. The curve a refers to the bound given by (4.28) while the curves b and c refers to the stability bounds derived in [6] and [35], respectively.

The bounds were obtained for the case when the complex conjugate pole pair moved along straight lines located at ± 45 degrees to the real axis. The coefficients were assumed to change as in (4.4), and the vector $\psi(k)$ was assumed to have unit magnitude at each time. The bounds are for the scaling factor μ_k in the coefficient evolution equation (4.4). It is clear from the results of Figure 4.5 that all three stability bounds converge to zero as the instantaneous poles approach the unit circle. However, the rate at which $\mu(k)$ in (4.28) approaches zero when the poles tend to the unit circle is several orders of magnitude slower than the bound derived in [35]. Furthermore, the bound for $\mu(k)$ given by (4.28) is much greater than the bound in [6]. This result is an additional demonstration of the usefulness of the sufficient stability bounds derived in this thesis.

4.5 Concluding Remarks

This chapter presented a novel stability condition for time-varying direct-form recursive linear systems. This condition was successfully applied for designing bounded-input, bounded-output stable adaptive IIR filters. The experimental results not only confirmed the reliability of the derived bound, but also demonstrated the better convergence characteristics of the stabilized algorithm when compared with other stable adaptive IIR filters. The time-varying bound derived in this thesis may be incorporated into any practical adaptive IIR filter. It is well-known that certain adaptive IIR filtering algorithms such as Feintuch's method diverge for all choices of the step size for certain input signals [156]. Experimental results as well as theoretical considerations indicate that the step size bound derived in this chapter eventually goes to zero in such situations, thus preserving the BIBO stability of the adaptive filter.

Chapter 5

Sufficient Stability Conditions for Discrete-Time Recursive Polynomial Filters

5.1 Introduction

In Chapter 1 we have introduced the Volterra filters, which are polynomial filters originated from the truncation of the Volterra series expansion. One drawback of these polynomial filters is that they require a large number of coefficients to characterize a nonlinear process. This problem can be alleviated by using recursive polynomial structures. In fact many real-world nonlinear systems have an infinite memory for the input signal history. In this situation, system modelling by means of recursive polynomial structures requires a much lower number of coefficients than the non-recursive counterpart. Moreover, as shown in Chapter 6, recursive polynomial structures are also obtained from the exact inversion of Volterra filters. However, recursive polynomial filters are inherently unstable, in the sense that it is always possible to find bounded input

---

Part of the content of this chapter was presented in
Enzo Mumolo e Alberto Carini, "A Stability Condition for Adaptive Recursive Second Order Volterra Filters," *Signal Processing*, Elsevier, Vol.54, No. 1, Oct. 1996, pp. 85-90
Alberto Carini e Enzo Mumolo, "Adaptive Stabilization of Recursive Second Order Polynomial Filters by Means of a Stability Test," *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, June 20-22 1995, Neos Marmaras, Halkidiki, Greece, pp. 939-942
Enzo Mumolo e Alberto Carini, "Recursive Volterra Filters with Stability Monitoring," *Proceedings of EUSIPCO 96*, September 10-13 1996, Trieste, Italia

signals which drive the nonlinear filter to instability. As a matter of fact, the stability of recursive polynomial filters depends not only on the filter coefficients as it happens in linear systems, but also on the input signal which must belong to a certain class to produce a bounded and well behaved filter output. This subset of input signals depends only from the recursive filter structure. Lee and Mathews introduced the notion of input-dependent stability for nonlinear recursive filters; some results concerning the bilinear systems are described in [84, 86, 87, 88]. Bilinear systems are the simplest form of recursive nonlinear model [92]. In [84, 86, 87] input dependent sufficient stability conditions have been reported; in [88] the conditions of [86] have been used to monitor the stability of a bilinear system model in a system identification task.

In [75], sufficient conditions for certain classes of discrete time nonlinear systems, where the output is computed from past input and output values plus mixed product of input and outputs, have been described. The conditions can be applied to any polynomial recursive system; however, a direct derivation of the output of the filter as a series of product of the input and past output samples is required. Generally, it is not easy to derive this series of products nor it is simple to verify the conditions given in [75].

In [68] other stability conditions for bilinear and quadratic filters are presented. In particular, conditions are given under which asymptotically periodic inputs produce asymptotically periodic outputs with the same period.

In this chapter we derive input-dependent stability conditions for recursive polynomial filters. In Section 5.2, we first describe a simple sufficient stability condition for recursive second order polynomial filters given by the following difference equation:

$$\begin{aligned} y(n) &= \sum_{i=0}^{M_1} \alpha_1(i)x(n-i) + \sum_{i=0}^{M_2}\sum_{j=i}^{M_2} \alpha_2(i,j)x(n-i)x(n-j) + \\ &\quad \sum_{i=1}^{N_1} h_1(i)y(n-i) + \sum_{i=1}^{N_2}\sum_{j=i}^{N_2} h_2(i,j)y(n-i)y(n-j). \end{aligned} \tag{5.1}$$

In Section 5.3 a similar stability condition is derived for the more general recursive polynomial filter described by the difference equation

$$\begin{aligned} y(n) &= \sum_{i_1=0}^{M_1} \alpha_1(i_1)x(n-i_1) + \sum_{i_1=0}^{M_2}\sum_{i_2=0}^{M_2} \alpha_2(i_1,i_2)x(n-i_1)x(n-i_2) + \\ &\quad \cdots + \sum_{i_1=0}^{M_p} \cdots \sum_{i_p=0}^{M_p} \alpha_p(i_1,\cdots,i_p)x(n-i_1)\cdots x(n-i_p) + \end{aligned}$$

$$+\sum_{i_1=1}^{N_1} h_1(i_1)y(n-i_1)+\sum_{i_1=1}^{N_2}\sum_{i_2=1}^{N_2} h_2(i_1,i_2)y(n-i_1)y(n-i_2)+$$
$$\cdots+\sum_{i_1=1}^{N_q}\cdots\sum_{i_q=1}^{N_q} h_q(i_1,\cdots,i_q)y(n-i_1)\cdots y(n-i_q). \qquad (5.2)$$

The system described in (5.1) and (5.2) are also referred to as "recursive Volterra filters". It will be shown in Section 5.2 and 5.3 that, provided that the stability of the linear part is guaranteed, the output of the filters in (5.1) and (5.2) is bounded for every n if the input signal is bounded by a certain value which can be very efficiently computed from the filter coefficients. No requirement is imposed on the filter coefficients.

5.2 Sufficient Stability Conditions for Recursive Quadratic Filters

For simplicity, let us introduce the time-invariant operator q^{-1} such that

$$\sum_{i=1}^{N_1} h_1(i)y(n-i)=\left(\sum_{i=1}^{N_1} h_1(i)q^{-i}\right)y(n)$$

Moreover, let us call p_i the zeros of the polynomial

$$\left(1-\sum_{i=1}^{N_1} h_1(i)q^{-i}\right)q^{N_1} \qquad (5.3)$$

and introduce the following terms:

$$\alpha=\prod_{i=1}^{N_1}(1-|p_i|), \qquad \beta=\sum_{i=0}^{M_1}|\alpha_1(i)|,$$
$$\gamma=\sum_{i=1}^{N_2}\sum_{j=i}^{N_2}|h_2(i,j)|, \qquad \delta=\sum_{i=0}^{M_2}\sum_{j=i}^{M_2}|\alpha_2(i,j)| \qquad (5.4)$$

We now prove the following preliminary result, which is a sufficient stability condition for the recursive system

$$y(n)=\xi(n)+\sum_{i=1}^{N_1} h_1(i)y(n-i)+\sum_{i=1}^{N_2}\sum_{j=i}^{N_2} h_2(i,j)y(n-i)y(n-j). \qquad (5.5)$$

Theorem 5.2.1 *If*

i *it is* $|\xi(n)| \leq M_\xi$ *and* $|y(n)| \leq M_y$ *for every* $n < 0$, *where* $M_\xi = \frac{\alpha^2}{4\gamma}$ *and* $M_y = \frac{\alpha}{2\gamma}$,

ii *the zeros* p_i *are inside the unit circle,* $\forall i = 1 \ldots N_1$,

iii *the input signal* $\xi(n)$ *is bounded by* M_ξ *for every* $n \geq 0$.

then the output signal $y(n)$ *of the recursive polynomial filter described in (5.5) is bounded by* M_y *for every* n.

Proof With the q^{-i} operator described above, the recursive quadratic system reported in (5.5) can be rewritten in the following way:

$$\left(1 - \sum_{i=1}^{N_1} h_1(i) q^{-i}\right) y(n) = \xi(n) + \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} h_2(i,j) y(n-i) y(n-j) \quad (5.6)$$

and, by defining

$$k(n) = \xi(n) + \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} h_2(i,j) y(n-i) y(n-j) \quad (5.7)$$

one can write

$$\left(1 - \sum_{i=1}^{N_1} h_1(i) q^{-i}\right) y(n) = k(n) \quad (5.8)$$

or

$$y(n) = \left\{ \prod_{i=1}^{N_1} \left(1 - p_i q^{-1}\right)^{-1} \right\} k(n). \quad (5.9)$$

By induction on the order of the system, one can easily show [84] that this last expression can also be written as follows:

$$y(n) = \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} k(n). \quad (5.10)$$

In fact, considering a first order system, with $N_1 = 1$, we have

$$y(n) = \left(1 - p_i q^{-1}\right)^{-1} k(n) \quad (5.11)$$

or, by successive substitutions,

$$y(n) = \left\{ \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} k(n). \tag{5.12}$$

In other words, (5.10) holds for $N_1 = 1$. Suppose now that (5.10) is true for all the system orders from 1 to $(N_1 - 1)$ and let us see if (5.10) holds for all the system orders. The N_1-th order system can be viewed as the cascade of a first order system with a pole p_{N_1} and a $(N_1 - 1)$-th order system with poles $p_1 \cdots p_{N_1 - 1}$. Hence

$$y(n) = \left\{ \sum_{l=0}^{\infty} p_{N_1}^l q^{-l} \right\} \left\{ \prod_{i=1}^{N_1 - 1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} k(n) = \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} k(n). \tag{5.13}$$

Substituting (5.7) in (5.10), it follows that

$$y(n) = \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} \xi(n) + \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} h_2(i,j) y(n-i) y(n-j) \tag{5.14}$$

and, from (5.14)

$$\begin{aligned} |y(n)| \leq & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l q^{-l} \right\} |\xi(n)| + \\ & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l q^{-l} \right\} \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} |h_2(i,j)| |y(n-i)| |y(n-j)| \end{aligned} \tag{5.15}$$

Let us now suppose that $|\xi(n)| \leq M_\xi$ for every n. By hypothesis, $|y(n)| \leq M_y$ for $n < 0$. Furthermore, let us suppose that $|y(n-i)| \leq M_y$ for each $i \geq 1$. Then, also the current output sample $y(n)$ is bounded by M_y. In fact, from (5.15):

$$\begin{aligned} |y(n)| \leq & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l \right\} M_\xi + \\ & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l q^{-l} \right\} \sum_{i=1}^{N_2} \sum_{j=i}^{N_2} |h_2(i,j)| M_y^2 \end{aligned} \tag{5.16}$$

Since $|p_i| < 1$, we have $\sum_{l=0}^{\infty} |p_i|^l = (1 - |p_i|)^{-1}$ and, by substituting into (10) and using (3) and (5), we obtain

$$|y(n)| \leq \frac{1}{\alpha} M_\xi + \frac{\gamma}{\alpha} M_y^2 = \frac{\alpha}{2\gamma} \tag{5.17}$$

In conclusion, if the input signal $\xi(n)$ is bounded by M_ξ for every n, by induction the output signal $y(n)$ is bounded by M_y for every n. □

If, in particular, the linear recursive part in (5.5) is missing, the input and output bounds are given by the following Corollary.

Corollary 5.2.1 *With the hypothesis that* $|\xi(n)| \leq \frac{1}{4\gamma}$ *and* $|y(n)| \leq \frac{1}{2\gamma}$ *for every* $n < 0$, *the output signal* $y(n)$ *of the following recursive nonlinear filter:*

$$y(n) = \xi(n) + \sum_{i=1}^{M_2} \sum_{j=i}^{M_2} h_2(i,j) y(n-i) y(n-j) \tag{5.18}$$

is bounded by $\frac{1}{2\gamma}$ *for every* $n \geq 0$ *if the input signal* $\xi(n)$ *is bounded by* $\frac{1}{4\gamma}$ *for every* $n \geq 0$.

Proof From (5.18) it follows that

$$|y(n)| \leq |\xi(n)| + \sum_{i=1}^{M_2} \sum_{j=i}^{M_2} |h_2(i,j)||y(n-i)||y(n-j)| \tag{5.19}$$

As in Theorem 5.2.1, this corollary can easily be proved by induction.□

Remark 5.2.1 The stability condition can be shown not to be necessary for stability by showing that there exist stable recursive polynomial filters that do not satisfy the input bound. For example, consider the system $y(n) = x(n) + y^2(n-1)$. According to Corollary 5.2.1, the input bound for stability is 1/4. However, if we consider the impulse response of this system, it can be readily shown that the system is stable for input amplitudes less than or equal to 1.

A sufficient condition for the stability of (5.1) is, finally, given in the following theorem.

Theorem 5.2.2 *If*

i *for every* $n < 0$ *it is* $|x(n)| \leq M_x$ *and* $|y(n)| \leq M_y$, *where*

$$M_x = \sqrt{\frac{\beta^2}{4\delta^2} + \frac{\alpha^2}{4\gamma\delta}} - \frac{\beta}{2\delta} \quad \text{and} \quad M_y = \frac{\alpha}{2\gamma},$$

ii *the zeros p_i are inside the unit circle, $\forall i = 1 \dots N_1$,*

iii *the input signal $x(n)$ is bounded by M_x for every $n \geq 0$.*

then the output of the system in (5.1), is bounded by M_y for every $n \geq 0$.

Proof The recursive equation described in (5.1) is obtained using (5.5) where $\xi(n)$ is defined as follows:

$$\xi(n) = \sum_{i=0}^{M_1} \alpha_1(i)x(n-i) + \sum_{i=0}^{M_2}\sum_{j=i}^{M_2} \alpha_2(i,j)x(n-i)x(n-j) \qquad (5.20)$$

By Theorem 5.2.1, if $\xi(n)$ is bounded by $M_\xi = \frac{\alpha^2}{4\gamma}$ for every n the output signal $y(n)$ of the system in (5.1) is bounded by M_y. Thus, we find a bound of $x(n)$, M_x, which assures that the output signal $\xi(n)$ of (5.20) is bounded by M_ξ. From (5.20)

$$|\xi(n)| \leq \beta M_x + \delta M_x^2 \qquad \forall n. \qquad (5.21)$$

The bound we are looking for is the maximum M_x such that

$$\beta M_x + \delta M_x^2 \leq M_\xi. \qquad (5.22)$$

This is equal to the positive root of $\beta M_x + \delta M_x^2 - M_\xi = 0$ which is

$$M_x = \sqrt{\left(\frac{\beta}{2\delta}\right)^2 + \frac{M_\xi}{\delta}} - \frac{\beta}{2\delta}$$

Substituting M_ξ in this last expression, we finally obtain

$$M_x = \sqrt{\frac{\beta^2}{4\delta^2} + \frac{\alpha^2}{4\gamma\delta}} - \frac{\beta}{2\delta}.$$

□

5.3 Sufficient Stability Conditions for General Order Recursive Volterra Filters

Let us call p_i the zeros of the polynomial

$$\left(1 - \sum_{i=1}^{N_1} h_1(i)q^{-i}\right) q^{N_1}. \qquad (5.23)$$

Moreover let us call

$$\gamma_1 = \prod_{i=1}^{N_1}(1 - |p_i|)$$

and introduce the absolute summation of the Volterra kernels as follows:

$$\gamma_2 = -\sum_{i_1}\sum_{i_2}|h_2(i_1, i_2)|, \quad \ldots, \quad \gamma_p = -\sum_{i_1}\cdots\sum_{i_p}|h_q(i_1, \cdots, i_q)|.$$

With these quantities, let us introduce the following polynomial

$$P_1(z) = \sum_{i=1}^{q} i\gamma_i z^{i-1}, \tag{5.24}$$

which for the Descartes' rule has a unique real positive zero. Let us call M_y the unique positive zero of (5.24) and, with it, compute the following value:

$$M_\xi = \sum_{i=1}^{q} \gamma_i M_y^i. \tag{5.25}$$

Finally, let us define the following quantities:

$$\delta_0 = -M_\xi, \qquad \delta_1 = \sum_{i_1}|\alpha_1(i_1)|,$$

$$\delta_2 = \sum_{i_1}\sum_{i_2}|\alpha_2(i_1, i_2)|, \quad \ldots, \quad \delta_p = \sum_{i_1}\cdots\sum_{i_p}|\alpha_p(i_1, \cdots, i_p)|$$

and, with such quantities, let us introduce the polynomial

$$P_2(z) = \sum_{i=0}^{p} \delta_i z^i. \tag{5.26}$$

Again, for the Descartes' rule the polynomial in (5.26) has a unique positive zero which will be called M_x.

The system described in (5.2) can be written also as follows:

$$y(n) = A\{x(n)\} + H\{y(n)\} \tag{5.27}$$

where

$$\begin{aligned} A\{x(n)\} \;=\; & \sum_{i_1=0}^{M_1} \alpha_1(i_1)x(n - i_1) + \sum_{i_1=0}^{M_2}\sum_{i_2=0}^{M_2} \alpha_2(i_1, i_2)x(n - i_1)x(n - i_2) \\ & + \cdots + \sum_{i_1=0}^{M_p}\cdots\sum_{i_p=0}^{M_p} \alpha_p(i_1, \cdots, i_p)x(n - i_1)\cdots x(n - i_p) \end{aligned} \tag{5.28}$$

and

$$H\{y(n)\} = \sum_{i_1=1}^{N_1} h_1(i_1)y(n-i_1) + \sum_{i_1=1}^{N_2}\sum_{i_2=1}^{N_2} h_2(i_1,i_2)y(n-i_1)y(n-i_2) + \cdots + \sum_{i_1=1}^{N_q}\cdots\sum_{i_q=1}^{N_q} h_q(i_1,\cdots,i_q)y(n-i_1)\cdots y(n-i_q). \tag{5.29}$$

Using the above definitions, we now prove the following results.

Theorem 5.3.1 *The output of the system*

$$y(n) = \xi(n) + H\{y(n)\} \tag{5.30}$$

is bounded for every n if

i *it is* $|\xi(n)| \leq M_\xi$ *and* $|y(n)| \leq M_y$ *for every* $n < 0$, *where* M_y *is the unique positive root of the polynomial in (5.24) and* M_ξ *is defined in (5.25),*

ii *the zeros* p_i *are inside the unit circle,* $\forall i = 1 \ldots N_1$,

iii *the input signal* $\xi(n)$ *is bounded by* M_ξ *for every* $n \geq 0$.

Moreover, in these conditions the output of the system in (5.30) is bounded by M_y *for every* $n \geq 0$.

Proof The proof is similar to that of Theorem 5.2.1. Using (5.29), (5.30) we have that

$$\left(1-\sum_{i_1=1}^{N_1} h_1(i_1)q^{-i_1}\right) y(n) = \xi(n)+\sum_{i_1=1}^{N_2}\sum_{i_2=1}^{N_2} h_2(i_1,i_2)y(n-i_1)y(n-i_2)+\cdots+ \sum_{i_1=1}^{N_q}\cdots\sum_{i_q=1}^{N_q} h_q(i_1,\cdots,i_q)y(n-i_1)\cdots y(n-i_q). \tag{5.31}$$

By defining

$$k(n) = \xi(n) + \sum_{i_1=1}^{N_2}\sum_{i_2=1}^{N_2} h_2(i_1,i_2)y(n-i_1)y(n-i_2) + \cdots + \sum_{i_1=1}^{N_q}\cdots\sum_{i_q=1}^{N_q} h_q(i_1,\cdots,i_q)y(n-i_1)\cdots y(n-i_q) \tag{5.32}$$

and, introducing the p_i's defined above,

$$y(n) = \left\{ \prod_{i=1}^{N_1} \left(1 - p_i q^{-1}\right)^{-1} \right\} k(n). \tag{5.33}$$

Since the poles p_i are inside the unit circle, by induction on the order of the system, one can easily show that this last expression can also be written as follows:

$$y(n) = \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} k(n). \tag{5.34}$$

Substituting the expression of $k(n)$ in (5.34), it follows that

$$\begin{aligned} y(n) \;=\; & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} \xi(n) + \\ & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} \sum_{i_1=1}^{N_2} \sum_{i_2=1}^{N_2} h_2(i_1, i_2) y(n-i_1) y(n-i_2) + \cdots + \\ & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} p_i^l q^{-l} \right\} \sum_{i_1=1}^{N_q} \cdots \sum_{i_q=1}^{N_q} h_q(i_1, \cdots, i_q) y(n-i_1) \cdots y(n-i_q) \end{aligned} \tag{5.35}$$

and, from (5.35)

$$\begin{aligned} |y(n)| \;\leq\; & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l q^{-l} \right\} |\xi(n)| + \\ & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l q^{-l} \right\} \sum_{i_1=1}^{N_2} \sum_{i_2=1}^{N_2} |h_2(i_1, i_2)| |y(n-i_1)| |y(n-i_2)| + \cdots + \\ & \left\{ \prod_{i=1}^{N_1} \sum_{l=0}^{\infty} |p_i|^l q^{-l} \right\} \sum_{i_1=1}^{N_q} \cdots \sum_{i_q=1}^{N_q} |h_q(i_1, \cdots, i_q)| |y(n-i_1)| \cdots |y(n-i_q)|. \end{aligned} \tag{5.36}$$

Let us now suppose that a bound $\mathcal{M}_\xi$ exists such that $|\xi(n)| \leq \mathcal{M}_\xi$ for every n and let us look for a bound of $y(n)$. We proceed by induction on the sample number. We suppose $|y(n)| \leq \mathcal{M}_y \; \forall n < 0$. Assume that $|y(n-i)| \leq \mathcal{M}_y$ for each $i \geq 1$. We will show that, in these hypothesis, a bound $\mathcal{M}_y$ that holds for any output sample $y(n)$ can be found. In fact, from (5.36):

$$|y(n)| \leq \left\{\prod_{i=1}^{N_1}\sum_{l=0}^{\infty}|p_i|^l\right\}\mathcal{M}_\xi + \left\{\prod_{i=1}^{N_1}\sum_{l=0}^{\infty}|p_i|^l\right\}\sum_{i_1=1}^{N_2}\sum_{i_2=1}^{N_2}|h_2(i_1,i_2)|\mathcal{M}_y^2 + \cdots +$$

$$+\left\{\prod_{i=1}^{N_1}\sum_{l=0}^{\infty}|p_i|^l\right\}\sum_{i_1=1}^{N_q}\cdots\sum_{i_q=1}^{N_q}|h_q(i_1,\cdots,i_q)|\mathcal{M}_y^q. \tag{5.37}$$

Since $|p_i| < 1$, we have obviously $\sum_{l=0}^{\infty}|p_i|^l = (1-|p_i|)^{-1}$. Thus, by substituting this last expression into (5.37), we obtain

$$|y(n)| \leq \frac{1}{\gamma_1}\mathcal{M}_\xi - \frac{\gamma_2}{\gamma_1}\mathcal{M}_y^2 - \cdots - \frac{\gamma_q}{\gamma_1}\mathcal{M}_y^q. \tag{5.38}$$

Thus, the system output $y(n)$ is always bounded if the right term of (5.38) is bounded by $\mathcal{M}_y$. By imposing this condition, from (5.38) we get

$$\mathcal{M}_\xi \leq \sum_{i=1}^{q}\gamma_i\mathcal{M}_y^i \tag{5.39}$$

The output of the system (5.30) is bounded for every choice of $\mathcal{M}_\xi$, $\mathcal{M}_y$ that satisfy (5.39). In particular, we are interested in the maximum value of $\mathcal{M}_\xi$ that meets condition (5.39), which is given by the $\mathcal{M}_y$ that maximizes the following function

$$\mathcal{F} = \sum_{i=1}^{q}\gamma_i\mathcal{M}_y^i \tag{5.40}$$

This $\mathcal{M}_y$ can be found by setting to zero the first derivative of $\mathcal{F}$. That is,

$$\frac{d\mathcal{F}}{d\mathcal{M}_y} = \sum_{i=1}^{q} i\gamma_i\mathcal{M}_y^{i-1} = 0 \tag{5.41}$$

According to the definitions given above, the unique positive solution of (5.41) is called M_y. The corresponding extrema of $\mathcal{F}$ is the upper bound on $\xi(n)$ we are looking for, and it is given by M_ξ as defined in (5.25). □

Theorem 5.3.2 *The output of the system*

$$y(n) = A\{x(n)\} + H\{y(n)\} \tag{5.42}$$

is bounded for every n if

i *it is* $|x(n)| \leq M_x$ *and* $|y(n)| \leq M_y$ *for every* $n < 0$, *where* M_x *and* M_y *are the unique positive roots of the polynomials in (5.24) and (5.26), respectively,*

ii *the zeros* p_i *are inside the unit circle,* $\forall i = 1 \dots N_1$,

iii *the input signal* $x(n)$ *is bounded by* M_x *for every* $n \geq 0$.

Moreover, in these conditions the output of the system in (5.42) is bounded by M_y *for every* $n \geq 0$.

Proof By using the result in Theorem 5.3.1, we find the maximum bound of the input to the system

$$\xi(n) = A\{x(n)\} \tag{5.43}$$

such that its output $\xi(n)$ is bounded by M_ξ. Therefore, assuming that $|x(n)| \leq \mathcal{M}_x$ for each n, from (5.28), it comes out that

$$|\xi(n)| \leq \sum_{i_1=0}^{M_1} |\alpha_1(i_1)|\mathcal{M}_x + \sum_{i_1=0}^{M_2}\sum_{i_2=0}^{M_2} |\alpha_2(i_1,i_2)|\mathcal{M}_x^2 + \cdots +$$

$$+ \sum_{i_1=0}^{M_p} \cdots \sum_{i_p=0}^{M_p} |\alpha_p(i_1,\cdots,i_p)|\mathcal{M}_x^p \tag{5.44}$$

or

$$|\xi(n)| \leq \sum_{i=1}^{p} \delta_i \mathcal{M}_x^i. \tag{5.45}$$

If we impose that the right term of (5.45) is bounded by M_ξ, we make sure that $|\xi(n)| \leq M_\xi$ for every n. Hence, using the definition of δ_0 given above, the following condition must hold:

$$\sum_{i=0}^{p} \delta_i \mathcal{M}_x^i \leq 0 \tag{5.46}$$

The maximum value of $\mathcal{M}_x$ that satisfies (5.46) is the unique positive zero of $P_2(z)$ described in (5.26). □

Remark 5.3.1 The conditions of Theorems 5.3.1 and 5.3.2 are not necessary for the stability of the systems (5.30) and (5.42), respectively.

Remark 5.3.2 The bounds are based on the computation of the roots of polynomials (5.24) and (5.26), respectively. The coefficients of these polynomials present just one change of the sign and therefore, for the Descartes rule of sign, the real and positive roots are unique. Accordingly, the zeros of the polynomials P_1 and P_2 can be computed very efficiently by numerical methods.

5.4 Final Remarks and Conclusion

Simple stability conditions for recursive polynomial filters have been described in this chapter. The sufficient conditions proved in Section 5.2 are expressed in a closed form, while those in Section 5.3 require the computation of the real positive zero of two polynomials. However, for the particular form of that polynomials, the real positive zero can be computed very efficiently by numerical methods.

As it will be pointed out in Chapter 6, the theorems presented in this chapter can be used to find conditions for the stability of the exact inverse of a truncated Volterra filter.

Chapter 6

The Inverse of Certain Nonlinear Systems

6.1 Introduction

The equalization and linearization of nonlinear system is a subject of exploding interest in Signal Processing. In Chapter 7 we will present a theory for the equalization and linearization of a wide class of nonlinear systems. In this chapter, instead, we present some theorems for the inversion of certain nonlinear systems; these theorems constitute the starting point for the development of the equalization and linearization theory introduced in Chapter 7. In particular, we present some results for the exact inversion of the nonlinear systems described by the input-output relationship

$$y(n) = g[x(n)]h[x(n-1), y(n-1)] + f[x(n-1), y(n-1)], \qquad (6.1)$$

where $g[\cdot]$, $h[\cdot,\cdot]$ and $f[\cdot,\cdot]$ are causal, discrete-time and nonlinear operators, and the inverse function $g^{-1}[\cdot]$ exists. We also present expressions for the pth order inverses of systems of the form

$$y(n) = g[x(n)] + f[x(n-1), y(n-1)]. \qquad (6.2)$$

The second result is useful in situations where the exact inverse system does not exist, or is not stable. Even when the exact inverse does not

---

Part of the content of this chapter was presented in
Alberto Carini, Giovanni L. Sicuranza and V. John Mathews "On the Inversion of Certain Nonlinear Systems," *IEEE Signal Processing Letters*, Dec. 97
Alberto Carini, Giovanni L. Sicuranza and V. John Mathews "On the Exact Inverse and the pth Order Inverse of Certain Nonlinear Systems," *Proceedings of NSIP 97*, September 7-11 1997, Michigan, USA

exist, the class of filters in (6.2) admits efficient realizations of their pth order inverses.

6.2 The Inverse of Certain Nonlinear Systems

In all of our discussions, we assume causal signals, *i.e.*, all the signals are identically zero for time indices less than zero. The following theorem shows how to evaluate the exact inverse of (6.1).

Theorem 6.2.1 *Let $g[\cdot]$, $h[\cdot,\cdot]$ and $f[\cdot,\cdot]$ be causal nonlinear discrete operators and let the inverse operator $g^{-1}[\cdot]$ exist. Then, the exact inverse of the system in (6.1) is described by the input-output relationship*

$$z(n) = g^{-1}\left[\frac{u(n) - f[z(n-1), u(n-1)]}{h[z(n-1), u(n-1)]}\right], \tag{6.3}$$

where $u(n)$ and $z(n)$ are the input signal and output signal, respectively, of the system.

Proof We demonstrate first that the system in (6.3) is the post-inverse of (6.1), *i.e.*, a cascade interconnection of the system in (6.1) followed by the system in (6.3) results in an identity system. We proceed by mathematical induction. Let $x(n)$ and $y(n)$ represent the input and output signals, respectively, of the system in (6.1). To prove the theorem using induction, we assume that

$$z(n-i) = x(n-i) \qquad \forall i > 0. \tag{6.4}$$

We must now show using (6.4) that

$$z(n) = x(n) \tag{6.5}$$

when $u(k) = y(k)$ for $k \leq n$. Now,

$$\begin{aligned} z(n) &= g^{-1}\left[\frac{y(n) - f[z(n-1), y(n-1)]}{h[z(n-1), y(n-1)]}\right] \\ &= g^{-1}\left[\frac{\{g[x(n)]h[x(n-1), y(n-1)] + \\ f[x(n-1), y(n-1)] - f[z(n-1), y(n-1)]\}}{h[z(n-1), y(n-1)]}\right]. \end{aligned} \tag{6.6}$$

By substituting $z(n-i) = x(n-i)$ from (6.4) into (6.6), it follows in a straightforward manner that $z(n) = x(n)$. We can prove in a similar manner that the system in (6.3) is also the pre-inverse of the system in (6.1), *i.e.* a cascade interconnection of the system in (6.3) followed by the system in (6.1) results in an identity system. This completes the proof. □

Remark 6.2.1 The inverse of the system in (6.1) may not exist or may not be stable. For example, if

$$h[z(n-1), u(n-1)] = 0 \tag{6.7}$$

at any time for some specific input signal, the inverse system of (6.1) is unstable.

Example 6.2.1 We wish to find the inverse of the bilinear system

$$\begin{aligned} y(n) &= x(n) + \sum_{i=1}^{N-1} a_i x(n-i) + \sum_{i=1}^{N-1} b_i y(n-i) + \\ &\quad \sum_{i=0}^{N-1}\sum_{j=1}^{N-1} c_{ij} x(n-i) y(n-j). \end{aligned} \tag{6.8}$$

Let us define f, h and g to be

$$\begin{aligned} f[x(n-1), y(n-1)] &= \sum_{i=1}^{N-1} a_i x(n-i) + \sum_{i=1}^{N-1} b_i y(n-i) + \\ &\quad \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} c_{ij} x(n-i) y(n-j), \end{aligned} \tag{6.9}$$

$$h[x(n-1), y(n-1)] = 1 + \sum_{j=1}^{N-1} c_{0j} y(n-j) \tag{6.10}$$

and

$$g[x(n)] = x(n), \tag{6.11}$$

respectively. Then, we can utilize Theorem 6.2.1 to find the inverse of the bilinear system to be

$$z(n) = \frac{\left\{u(n) - \sum_{i=1}^{N-1} b_i u(n-i) - \sum_{i=1}^{N-1} a_i z(n-i) + \right.}{1 + \sum_{j=1}^{N-1} c_{0j} y(n-j)} \frac{\left. - \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} c_{ij} z(n-i) u(n-j) \right\}}{1 + \sum_{j=1}^{N-1} c_{0j} y(n-j)}. \tag{6.12}$$

A simpler expression can be found for the inverse filter of the system in (6.2). The following corollary can be immediately derived from Theorem 6.2.1.

Corollary 6.2.1 *Let $g[\cdot]$ and $f[\cdot,\cdot]$ be causal nonlinear discrete operators and let the inverse operator $g^{-1}[\cdot]$ exist. Then, the causal discrete nonlinear system described in (6.2) has the inverse system whose input-output relationship is given by*

$$z(n) = g^{-1}\Big[u(n) - f[z(n-1), u(n-1)]\Big]. \tag{6.13}$$

Example 6.2.2 The inverse of the bilinear system

$$\begin{aligned} y(n) &= x(n) + \sum_{i=1}^{N-1} a_i x(n-i) + \sum_{i=1}^{N-1} b_i y(n-i) + \\ &\quad \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} c_{ij} x(n-i) y(n-j) \end{aligned} \tag{6.14}$$

is the bilinear system

$$\begin{aligned} z(n) &= u(n) - \sum_{i=1}^{N-1} b_i u(n-i) - \sum_{i=1}^{N-1} a_i z(n-i) + \\ &\quad - \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} c_{ij} z(n-i) u(n-j). \end{aligned} \tag{6.15}$$

Note that the double summation in (6.14) is slightly different from the double summation in (6.8), and this difference contributes to the simpler inverse system in (6.15).

6.3 pth Order Inverses

Not all nonlinear systems possess an inverse and many nonlinear systems admit an inverse only for a certain subset of input signals. For these reasons, Schetzen developed the theory of the pth order inverse of a nonlinear system whose input-output relation can be represented using a Volterra series expansion [126, 127]. The pth order inverse of a nonlinear system H is defined in [126, 127] as the pth order system which, connected in cascade with H, results in a system whose linear kernel is the identity system and whose Volterra kernels from the second up to the pth order are zero. A pth order system is one in which all the Volterra

kernels of order greater than p are zero. The definition of the pth order inverse was relaxed in [125] by allowing the inverse system to possess non-zero Volterra operators of order greater than p. These operators do not affect the first p Volterra operators of the cascade system. This relaxed definition of the pth order inverse was employed in [125] to derive simpler and computationally more efficient expressions for the inverse system. However, because of the presence of higher order components, the definition of the pth order inverse in [125] does not result in a unique inverse system.

The following theorem presents an efficient method of computing a pth order inverse of the system in (6.2). Note that this system is a special case of the system in (6.1) when $h[\cdot,\cdot] = 1$.

Theorem 6.3.1 *Let $g[\cdot]$ and $f[\cdot,\cdot]$ be causal, discrete-time nonlinear operators with convergent Volterra series expansions with respect to all the arguments. Moreover, let the pth order inverse $g_p^{-1}[\cdot]$ of the system $g[\cdot]$ exist. Then a pth order inverse of the causal, discrete-time nonlinear system described in (6.2) is given by the following input-output relationship:*

$$z(n) = g_p^{-1}\Big[u(n) - f[z(n-1), u(n-1)]\Big]. \tag{6.16}$$

Proof As was the case for the Theorem 6.2.1, we first show that the system in (6.16) is the pth order post-inverse of the system in (6.2). Using the same variables as in the derivation of Theorem 6.2.1, we express $z(n)$ as

$$\begin{aligned} z(n) &= g_p^{-1}\Big[y(n) - f[z(n-1), y(n-1)]\Big] \\ &= g_p^{-1}\Big[g[x(n)] + f[x(n-1), y(n-1)] + \\ &\quad -f[z(n-1), y(n-1)]\Big]. \end{aligned} \tag{6.17}$$

We proceed by mathematical induction. We assume that, for any i greater than zero, the output $z(n-i)$ differs from $x(n-i)$ only by $T_p(n-i)$, a term whose Volterra series expansion in $x(n)$ contains only kernels of order larger than p, *i.e.*,

$$z(n-i) = x(n-i) + T_p(n-i) \qquad \forall i > 0. \tag{6.18}$$

We have to prove that the Volterra series expansion of $z(n) - x(n)$ have zero kernels of order up to p. Since $f[\cdot,\cdot]$ admits a convergent Volterra series expansion, we have from (6.18) that the Volterra series expansion

of the difference $f[x(n-1), y(n-1)] - f[z(n-1), y(n-1)]$ contains only kernels of order greater than p, *i.e.*,

$$f[x(n-1), y(n-1)] - f[z(n-1), y(n-1)] = 0 + T'_p(n), \tag{6.19}$$

where the Volterra kernels of $T'_p(n)$ up to order p are zero. Substituting (6.19) in (6.17), we get

$$z(n) = g_p^{-1}\Big[g[x(n)] + T'_p(n)\Big]. \tag{6.20}$$

The pth order inverse of the operator $g[\cdot]$ derived in [126] is given by a pth order truncated Volterra series whose kernels depend only on the first p kernels of the Volterra series expansion of $g[\cdot]$. The pth order inverse derived in [125] may have Volterra kernels of order greater than p. However, the inverse still has a Volterra series expansion with finite order of nonlinearity, and it depends only on the first p kernels of the Volterra series expansion of $g[\cdot]$. Consequently, it immediately follows from (6.20) that

$$z(n) = x(n) + T_p(n) \tag{6.21}$$

and that the system in (6.16) is the pth order post-inverse of the system in (6.2). We can prove in a similar manner that it is also a pre-inverse of the system in (6.2). □

Remark 6.3.1 Due to the rational structure of the system in (6.3), a similar expression for the pth order inverse of the system in (6.1) does not exist.

Example 6.3.1 We wish to derive a pth order inverse for the second-order Volterra filter given by the following expression:

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) + \sum_{i=0}^{N-1}\sum_{j=i}^{N-1} b_{ij} x(n-i)x(n-j). \tag{6.22}$$

Let

$$g[x(n)] = a_0 x(n) + x(n)\sum_{j=0}^{N-1} b_{0j} x(n-j) \tag{6.23}$$

and

$$f[x(n-1)] = \sum_{i=1}^{N-1} a_i x(n-i) + \sum_{i=1}^{N-1}\sum_{j=i}^{N-1} b_{ij} x(n-i)x(n-j). \tag{6.24}$$

According to the Theorem 6.3.1, a pth order inverse of (6.22), is

$$z(n) = g_p^{-1}\left[u(n) - \sum_{i=1}^{N-1} a_i z(n-i) - \sum_{i=1}^{N-1}\sum_{j=i}^{N-1} b_{ij} z(n-i)z(n-j)\right]. \tag{6.25}$$

The pth order inverse $g_p^{-1}[\cdot]$ can be computed iteratively as in [125] and is given by

$$g_p^{-1}[u(n)] = -g_1^{-1}\left[q_p\left[g_{p-1}^{-1}[u(n)]\right] - u(n)\right], \tag{6.26}$$

where $g_1^{-1}[\cdot]$ is the inverse of the first Volterra operator of $g[\cdot]$ (*i.e.*, a_0^{-1} in our case) and $q_p[\cdot]$ is the truncated Volterra series expansion of the system $g[\cdot]$ that contains only the second through pth order Volterra kernels.

While it is possible to compute the pth order inverse of the system of (6.22) as in [125], using the structure in [125] for inverting a smaller subsystem and then using Theorem 6.3.1, as we have done here, is a more efficient procedure in most situations.

The computational cost for the evaluation of (6.25) is $2(N-1) + \frac{(N-1)N}{2} + (N+2)(p-1)$ multiplications per time instant. The corresponding computational cost for directly computing the pth order inverse of (6.22) as in (6.26) is $N + \left(2N + \frac{N(N+1)}{2}\right)(p-1)$ multiplications per time instant. Implementing (6.25) has a computational cost of $O(N^2 + pN)$ multiplications per time instant while for the method in [125] the computational cost is $O(N^2 p)$. The methodology suggested by Theorem 6.3.1 is more efficient for evaluating the pth order inverse of a Volterra filter of order q when p is greater than q. On the other hand, when $p < q$, only the first p Volterra operators are significant for the evaluation of the pth order inverse. In this situation, both methods of inversion require almost the same number of multiplications per sample.

6.4 An Experimental Result

We consider the pth order inversion of the second order Volterra filter with input-output relationship

$$\begin{aligned} y(n) = {} & x(n) - x(n-1) - 0.125x(n-2) + \\ & 0.3125x(n-3) + x^2(n) - 0.3x(n)x(n-1) + \\ & 0.2x(n)x(n-2) - 0.5x(n)x(n-3) + \\ & 0.5x^2(n-1) - 0.3x(n-1)x(n-2) + \end{aligned}$$

$$-0.6x(n-1)x(n-3) - 0.6x^2(n-2) + 0.5x(n-2)x(n-3) - 0.1x^2(n-3). \quad (6.27)$$

The pth order inverse derived applying Theorem 6.3.1, where $g_p^{-1}[\cdot]$ is computed as in equation (6.26), is compared with the pth order inverse obtained by directly using the method in [125]. In Figure 6.1 the points identified with o refer to the pth order inverse of the Theorem 6.3.1, while the points indicated with + refer to the pth order inverse of [125]. The plots in Figure 6.1a compare the computational cost in multiplications for different orders p of the inversion. The computational efficiency of the pth order inverse of Theorem 6.3.1 over the inverse suggested in [125] can be clearly seen in this figure. Figures 6.1b and 6.1c displays the mean-squared error (MSE) between the input signal of the system in (6.27) and the output of its pth order inverse when connected in cascade to the system. The input signal was white and Gaussian-distributed with zero mean value. Figure 6.1b presents the MSE in the reconstruction of the input for different values of the inverse filter order p when the standard deviation of the input signal was 0.05. Figure 6.1c shows the mean-square error values for different standard deviations of the input signal for a fifth-order inverse system. All the results presented are time averages of 1,000 samples of the ensemble averages computed over fifty independent experiments. Values of the standard deviations for which a corresponding MSE value is absent correspond to instability situations. We can see that our approach give the similar or better performances as the method in [125] till instability arises in the inverse system. In such situations, the performance of the pth order inverse of [125] are also unacceptable.

6.5 Concluding Remarks

This chapter presented expressions for the exact inverse and the pth order inverse of a wide class of discrete-time nonlinear systems. This class includes most causal polynomial systems with finite order as well as many nonlinear filters with nonpolynomial input-output relationships. In particular, Theorem 6.2.1 allows the inversion of all recursive polynomial systems whose dependence on the input sample $x(n)$ can be characterized using an invertible component $g[x(n)]$. The computational cost of the exact inverse filter coincides with the cost of implementing the direct system and the operator $g^{-1}[\cdot]$. Theorem 6.3.1 applies to all recursive polynomial systems with the same characteristic as described above, as

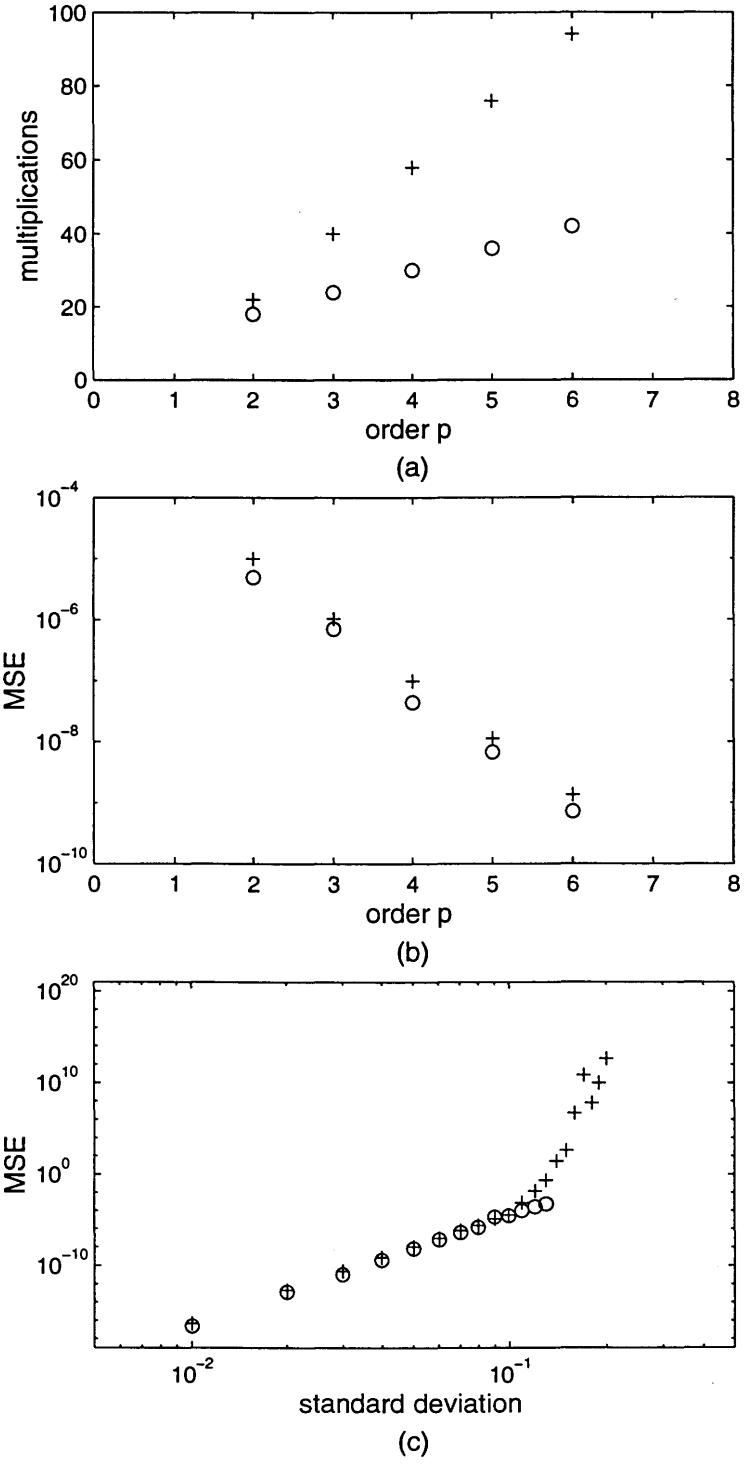


Figure 6.1: Experimental Results.

well as many other nonlinear systems. In this case also, the cost of implementing the inverse filter is that of implementing the direct system and the pth order inverse of $g[\cdot]$. All the inverse filters presented in this chapter are recursive and therefore may possess poor stability properties. Consequently, the stability of such systems must be tested after the inversion of the filter. Stability of recursive nonlinear systems is a topic of active research. Some useful stability results for recursive polynomial filters can be found in [19, 75, 86, 87, 133] and in Chapter 5.

Chapter 7

Equalization and Linearization of Nonlinear Systems

7.1 Introduction

Equalization of linear systems has been studied for several years [94]. Many real channels, however, possess non-negligible nonlinearities that make it impossible for linear equalization procedures to provide acceptable results. Examples of real world systems in which nonlinear effects are present include satellite communication channels [10, 11], voiceband data transmission systems [13], high density magnetic recordings [16], high density optical systems [1] and loudspeaker systems [49, 53, 69], drift oscillations in random seas [71], semiconductor devices [64, 99, 100] and biological phenomena [63, 74], to name but a few.

The need to compensate for such nonlinearities in these and other applications has made the problem of nonlinear equalization one of the most active research areas of digital signal processing. This chapter presents a theory for the exact and the pth order equalization or linearization of nonlinear systems with known recursive or nonrecursive polynomial input-output relationship. Extension to more general nonlinear system models such as those considered in [26] is possible. However such exten-

---

Part of the content of this chapter was presented in
Alberto Carini, Giovanni L. Sicuranza and V. John Mathews "Equalization and Linearization of Nonlinear Systems," *Proceedings of ICASSP-98, International Conference on Acoustics Speech and Signal Processing*, May 12-15 1998, Seattle, Washington, USA.

sions are not presented here. The theory we present is an extension of the standard equalization technique for linear systems.

Definition 7.1.1 *A* nonlinear equalizer *is a filter which, when connected in cascade before or after a nonlinear system, results in an overall system whose characteristics correspond to those of an identity system in the band of frequencies and in the range of input signal amplitudes of interest.*

Unlike the linear case, we impose a limit on the range of input amplitudes because of the nonlinear nature of the problem. Nonlinear systems, are often amplitude dependent. Some nonlinear systems may be stable for a certain range of input signal amplitudes, but their outputs may not be defined or they may be unstable outside that range of amplitudes [68, 75, 86, 87, 97].

Definition 7.1.2 *A* linearizer *is a filter which, when connected in cascade before or after the unknown system, results in an overall system whose characteristics correspond to those of a linear system in the frequency band and in the range of input signal amplitudes of interest.*

When a equalizer (linearizer) is connected before the unknown system it is called a *pre-equalizer* (*pre-linearizer*). When it is connected after a nonlinear system, it is called a *post-equalizer* (*post-linearizer*).

Several equalization/linearization procedures are available in the literature [11, 13, 16, 49, 53, 56]. Many such techniques start from the identification of a model which describes the input-output relationship of the unknown system. One exception to this framework is the work of Giannakis and Serpedin [56]. In [56] a method for the blind equalization of truncated Volterra channels by means of a bank of linear filters is presented. The technique of [56] equalizes the system channel using only its output signal. We note, however that the method in [56] cannot be used for pre-equalization or pre-linearization of nonlinear systems. In addition, our method is useful for a much larger class of channel models.

One of the first attempts of nonlinear equalization was performed in digital communication channels. Two different techniques were derived. One method performs equalization by designing a truncated Volterra filter which minimizes the mean square error between the decision device input and the correct symbol value [11]. The second technique performs a cancellation of the estimated linear and nonlinear intersymbol interferences obtained from preliminary decision performed after linear equalization [13, 16].

A nonlinear equalization/linearization technique that is popular with many researchers is described in [49] and [53]. First, the unknown system is identified by means of a truncated Volterra system model. Then an approximate equalizer or linearizer is derived by means of the pth order inverse [126, 127] of the estimated forward system. We note here that the system in [53] uses a second-order inverse, even though [53] does not explicitly state so. A pth order system is one in which all the Volterra kernels of order greater than p are zero. The pth order inverse is an approximation of the true inverse of the nonlinear model. The pth order inverse of a nonlinear system H is defined as the pth order truncated Volterra system, which when connected in cascade with H results in a system whose Volterra kernels from the second up to the pth order are zero. A generalized pth order inverse that allows the inverse filter to have kernels of order greater than p has also been derived [125]. The higher-order kernels do not affect the first p kernels of the cascade of the pth order inverse and the unknown system. The computation of the pth order inverse requires the inversion of the linear component of the unknown system model. This linear inverse filter is typically determined by approximating the desired transfer function with an FIR filter [49, 53]. Both amplitude and phase errors highly degrade the performances of the resulting linearizer.

This chapter introduces methods for equalizing and linearizing a class of nonlinear systems whose input-output relationship is given by

$$\begin{aligned} y(n) &= \sum_{i=0}^{N} a_i x(n-i) - \sum_{i=1}^{N} b_i y(n-i) + \\ &\sum_{k=2}^{L} \sum_{i_1=0}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} h_{i_1 i_2 \ldots i_k} x(n-i_1) x(n-i_2) \cdot \ldots \cdot x(n-i_k) \\ &- \sum_{k=2}^{L} \sum_{i_1=0}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} r_{i_1 i_2 \ldots i_k} y(n-i_1) y(n-i_2) \cdot \ldots \cdot y(n-i_k) \\ &+ \sum_{i=0}^{N} \sum_{j=1}^{N} c_{ij} x(n-i) y(n-j), \end{aligned} \tag{7.1}$$

where we consider only a mixed product term for ease of presentation. We first present exact equalizers and linearizers for such systems. Such systems may not always be realizable, and even the realizable systems may not be stable. Consequently, we introduce the concepts of the p-th order equalization and linearization, and present algorithms for developing realizable and stable pth order equalizers and linearizers. We assumes that

the characteristics of the nonlinear system to be equalized (linearized) are completely known. Adaptive algorithms that estimate the parameters of the system online are under investigation and will be the content of a future paper.

The rest of the chapter is organized as follows. The theory of exact equalization and linearization for nonrecursive polynomial system models is presented in Section 7.2. Section 7.3 derives the pth order equalization (linearization) technique from a power series expansion of the exact equalization (linearization) method. In Section 7.4, we extend the exact and pth order equalization and linearization theory to recursive polynomial filters. Section 7.5 presents an experimental result that illustrates the capabilities of our techniques. Concluding remarks are given in Section 7.6.

7.2 Ideal equalization and linearization

In Chapter 6, it was shown that the inverse of the system

$$y(n) = G[x(n)]H[x(n-1), y(n-1)] + F[x(n-1), y(n-1)], \qquad (7.2)$$

where $G[\cdot]$, $H[\cdot,\cdot]$ and $F[\cdot,\cdot]$ are causal, discrete-time and nonlinear operators and the inverse function $G^{-1}[\cdot]$ exists, always exists and is given by

$$w(n) = G^{-1}\left[\frac{u(n) - F[w(n-1), u(n-1)]}{H[w(n-1), u(n-1)]}\right]. \qquad (7.3)$$

We restrict ourselves to the nonrecursive system model with input-output relationship

$$\begin{aligned} y(n) &= \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{N}\sum_{j=i}^{N} c_{ij} x(n-i)x(n-j) + \dots \\ &\quad + \sum_{i_1=1}^{N}\sum_{i_2=i_1}^{N} \dots \sum_{i_L=i_{L-1}}^{N} h_{i_1 i_2 \dots i_L} x(n-i_1)x(n-i_2)\cdot \dots \cdot x(n-i_L). \end{aligned} \qquad (7.4)$$

in this section. Extension of the results to more general system models are considered in Section 7.4.

According to (7.3), the inverse of the system in (7.4) always exists and is given by

$$w(n) = \frac{1}{a_0}\left[u(n) - \sum_{i=1}^{N} a_i w(n-i) - \sum_{i=1}^{N}\sum_{j=i}^{N} c_{ij} w(n-i)w(n-j) - \dots\right.$$

$$-\sum_{i_1=1}^{N}\sum_{i_2=i_1}^{N}\cdots\sum_{i_L=i_{L-1}}^{N} h_{i_1 i_2 \ldots i_L} w(n-i_1)w(n-i_2)\cdot\ldots\cdot w(n-i_L)\Bigg], \tag{7.5}$$

where $u(n)$ and $w(n)$ are the input and output signal, respectively. The inverse filter in (7.5) is usually computationally less expensive than the pth order inverse of (7.4), especially when p takes high values. While the pth order inverse always introduces new harmonic components of order greater than p, the exact inverse of (7.5) does not give rise to such harmonics. However, the recursive system of (7.5) may not be stable. This issue will be discussed later in the chapter. An explicit expression for the output of the inverse system as in (7.5) is possible because the system model does not depend on the input sample $x(n)$ in a nonlinear manner.

An implicit expression for the inverse of the more general system

$$\begin{aligned} y(n) &= \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=0}^{N}\sum_{j=i}^{N} c_{ij} x(n-i)x(n-j) + \ldots \\ &\quad + \sum_{i_1=0}^{N}\sum_{i_2=i_1}^{N}\cdots\sum_{i_L=i_{L-1}}^{N} h_{i_1 i_2 \ldots i_L} x(n-i_1)x(n-i_2)\cdot\ldots\cdot x(n-i_L) \end{aligned} \tag{7.6}$$

if it exists, is given by

$$\begin{aligned} w(n) &= \frac{1}{a_0}\Bigg[u(n) - \sum_{i=1}^{N} a_i w(n-i) - \sum_{i=0}^{N}\sum_{j=i}^{N} c_{ij} w(n-i)w(n-j) - \ldots \\ &\quad - \sum_{i_1=0}^{N}\sum_{i_2=i_1}^{N}\cdots\sum_{i_L=i_{L-1}}^{N} h_{i_1 i_2 \ldots i_L} w(n-i_1)w(n-i_2)\cdot\ldots\cdot w(n-i_L)\Bigg]. \end{aligned} \tag{7.7}$$

In what follows, we employ a compact expression given by

$$y(n) = A(q)x(n) + \mathcal{N}[x(n)] \tag{7.8}$$

to represent the systems in (7.4) and (7.6). In the above equation, q^{-1} is the delay operator, $x(n)$ and $y(n)$ are the input and output signals, respectively,

$$A(q) = \sum_{i=0}^{N} a_i q^{-i} \tag{7.9}$$

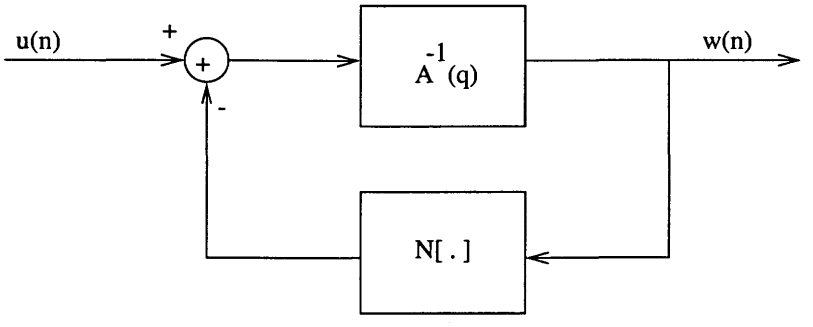


Figure 7.1: The ideal equalizer.

represents the linear component of the nonlinear system, and

$$\mathcal{N}[x(n)] = \sum_{i=r}^{N}\sum_{j=i}^{N} c_{ij}x(n-i)x(n-j) + \ldots + \sum_{i_1=r}^{N}\sum_{i_2=i_1}^{N}\cdots\sum_{i_L=i_{L-1}}^{N} h_{i_1 i_2 \ldots i_L}x(n-i_1)x(n-i_2)\cdot\ldots\cdot x(n-i_L) \tag{7.10}$$

is the component of the nonlinear system obtained by removing the effects of the linear component from the output signal. If r is equal to 1, (7.10) corresponds to the system in (7.4), and if r is equal to 0, the expression corresponds to the system in (7.6). Using the above notation, we can express the input-output relationship of the inverse of the system in (7.8) as

$$A(q)w(n) = u(n) - \mathcal{N}[w(n)]. \tag{7.11}$$

We can also express the output signal explicitly as

$$w(n) = A^{-1}(q)u(n) - A^{-1}(q)\mathcal{N}[w(n)]. \tag{7.12}$$

Figure 7.1 shows a block diagram for the recursive polynomial filter in (7.12). We note that the overall system can be described as a feedback system in which the feedback loop contains a nonlinear operator and the feedforward loop contains the inverse of the linear component of the quadratic filter. The inverse system in (7.12) will not be stable unless the linear part of the system model in (7.8) has minimum phase characteristics. When $A(q)$ represents a minimum phase system, the system in (7.12) can be shown to operate in a stable manner whenever the input signal is sufficiently small. The bound on the input signal depends on the zeros of $A(q)$ and on the coefficients of $\mathcal{N}[\cdot]$ [68, 86, 87, 97]. Thus,

our inverse system will equalize the above nonlinear system only on the range of amplitudes for which it is stable.

In many applications, we are interested in equalizing the unknown system only on a certain band of frequencies. For example it may be known that the input signal is band-limited. Such an equalizer may be designed by replacing $A^{-1}(q)$ in Figure 7.1 with another linear filter such that the overall system response corresponds to that of an identity system in the band of interest and possibly to zero outside the band. The following two theorems characterize the structure of the post and pre-equalizers for a specific range of frequencies.

Theorem 7.2.1 *Let the input signal $x(n)$ of the system in (7.8) be band-limited with spectrum inside a certain frequency band B and let $u(n)$ be its output. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside the band B. A post-equalizer for the system of (7.8) in the band B is given by*

$$w(n) = \tilde{A}^{-1}(q)u(n) - \tilde{A}^{-1}(q)\mathcal{N}[w(n)]. \tag{7.13}$$

Proof We consider the post-equalization of the nonlinear system in (7.8) in the band B and the elimination of all other frequencies at the output. For this purpose, we first cascade the system in (7.8) with the linear filter $\tilde{A}^{-1}(q)$ and then equalize the resulting nonlinear system. Cascading (7.8) with the linear system $\tilde{A}^{-1}(q)$ eliminates all frequencies outside the band B. The resulting system has input-output relationship

$$z(n) = \tilde{A}^{-1}(q)A(q)x(n) + \tilde{A}^{-1}(q)\mathcal{N}[x(n)]. \tag{7.14}$$

Since $x(n)$ has frequency components only on B, the above system is equivalent to

$$z(n) = x(n) + \tilde{A}^{-1}(q)\mathcal{N}[x(n)], \tag{7.15}$$

whose post-inverse system is given by

$$w(n) = z(n) - \tilde{A}^{-1}(q)\mathcal{N}[w(n)]. \tag{7.16}$$

Since $z(n)$ is band-limited to the band B, the output of the system in (7.16) is also band-limited to the band B. Thus, the cascade of $z(n) = \tilde{A}^{-1}(q)u(n)$ (where we assume $u(n) = y(n)$) and the system in (7.16) is an exact post-equalizer for the system in (7.8) in the band B. It is trivial to prove that this system is identical to (7.13). □

When the input signal is not band-limited to the frequency band B, the filter in (7.13) is only an approximate post-equalizer for the system in (7.8). The equalization, in particular, is affected by the frequency components of the input signal outside the band B since the output of the nonlinear filter (7.8) in the frequency band B is affected by the input signal components outside the band. If the contribution of the out-of-band is negligible, the system in (7.13) will still provide good equalization of the system in (7.8).

Theorem 7.2.2 *Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside the band B. A pre-equalizer in the frequency band B for the system of (7.8) is given by*

$$w(n) = \tilde{A}^{-1}(q)u(n) - \tilde{A}^{-1}(q)\mathcal{N}[w(n)]. \tag{7.17}$$

The proof is similar to that of Theorem 7.2.1. Since the pre-equalizer operates directly on the input signal, there is no need to assume that it is band-limited to achieve an exact equalization. The pre-equalizer will directly eliminate the frequency components of the input signal that fall outside the band B. However, the pre-equalizer is unable to eliminate the frequency components of $y(n)$ that fall outside the band B. Both amplitudes and phase errors between the linear equalizer $\tilde{A}^{-1}(q)$ and the linear part of the true inverse filter $A^{-1}(q)$ affect the quality of the nonlinear equalization. Consequently, it is important to design the linear system $\tilde{A}^{-1}(q)$ with particular care.

7.2.1 The ideal pre- and post-linearizers

There are many situations in which we do not have to perfectly equalize the nonlinear system, but desire only to compensate for the nonlinearities introduced by the unknown system. As we defined earlier, a pre-linearizer (post-linearizer) is a filter, which when connected before (after) a nonlinear system, results in an overall system whose characteristics corresponds to those of a linear filter in the frequency band and in the range of input signal amplitudes of interest. In this chapter, the linear system that results from the linearization process will always have transfer function equal to the linear part of the nonlinear system that is linearized.

Definition 7.2.1 *The* ideal *pre-linearizer (post-linearizer) is the filter that pre-linearizes (post-linearizes) the nonlinear system in all of the frequency domain.*

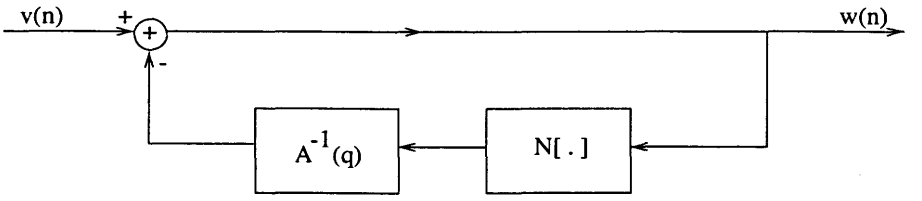


Figure 7.2: The ideal pre-linearizer.

The ideal pre-linearizing filter for the system in (7.8) is given by the following expression:

$$w(n) = v(n) - A^{-1}(q)\mathcal{N}[w(n)], \tag{7.18}$$

where $v(n)$ and $w(n)$ are the input and the output signal, respectively, of the prelinearizer.

Proof Consider a cascade of a linear system

$$u(n) = A(q)v(n) \tag{7.19}$$

followed by an identity system obtained by the cascade connection of the pre-equalizer of (7.8) with the system in (7.8). Obviously, the overall system characteristic is also given by (7.19). Thus, the cascade connection of the linear system in (7.19) and the ideal pre-equalizer given by (7.11) is an ideal pre-linearizer for the nonlinear system in (7.8). The overall input-output relationship of this pre-linearizer is given by

$$A(q)w(n) = A(q)v(n) - \mathcal{N}[w(n)] \tag{7.20}$$

which is identical to the system of (7.18). □

In the same way we can prove that the ideal post-linearizing filter for (7.8) is given by

$$w(n) = v(n) - \mathcal{N}[A^{-1}(q)w(n)]. \tag{7.21}$$

Figures 7.2 and 7.3 illustrate the block diagrams of the ideal pre- and post-linearizer for the system in (7.8).

We now consider the problem of designing pre- and post-linearizers when the input signal is known to be bandlimited. The following results can be proved in a manner similar to that employed for proving Theorem 7.2.1.

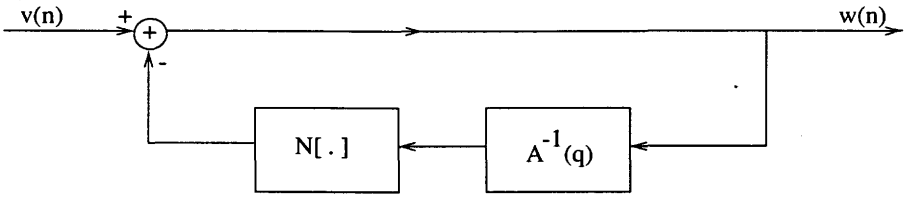


Figure 7.3: The ideal post-linearizer.

Theorem 7.2.3 *Let the input signal $x(n)$ of the system in (7.8) be band-limited with spectrum inside a certain frequency band B and let $v(n)$ be its output. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside the band B. A post-linearizing filter in the band B for the system of (7.8) is given by the system with input-output relationship*

$$w(n) = v(n) - \mathcal{N}[\tilde{A}^{-1}(q)w(n)]. \tag{7.22}$$

Theorem 7.2.4 *Let the input signal $v(n)$ be band-limited to a certain frequency band B. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside the band B. A pre-linearizing filter in the band B for the system of (7.8) is given by the system with input-output relationship*

$$w(n) = v(n) - \tilde{A}^{-1}(q)\mathcal{N}[w(n)]. \tag{7.23}$$

If the input signal of the nonlinear system in (7.8) is not band-limited, the filter in (7.22) is only an approximation to the post-linearizer of the system in (7.8). If the input signal $v(n)$ of (7.23) is not band-limited in the band B, the output $w(n)$ of the linearizer is not band-limited. However, the system in (7.23) is still a pre-linearizer for the nonlinear system in (7.8) in the band B. In any case, the output $y(n)$ of the cascade of the system in (7.23) followed by (7.8) has non null frequency components outside the band B.

7.3 pth Order Equalizers and Linearizers

In the previous section, we introduced some filters for the exact equalization or linearization of a certain class of nonlinear systems. These filters may not always be realizable. For example, when $r = 0$ in (7.8), the equalizers and linearizers do not have explicit input-output relationship. In this situation the filters of Figures 7.1, 7.2 and 7.3 are not realizable because the feedback loop depends on output samples that have

not yet been computed. Furthermore, because of the recursive structure of the equalizers/linearizers, these filters, if they exist, may also be unstable. In what follows, we present a theory for the pth order equalization/linearization of nonlinear systems [126, 127]. By means of pth order equalization/linearization we derive filters that are always realizable and are bounded input bounded output stable.

Definition 7.3.1 *A* pth order equalizer *is a filter which, when connected in cascade before or after a nonlinear system, results in an overall system whose characteristics, in the band of frequencies and in the range of the input signal amplitudes of interest, correspond to those of a parallel connection of an identity system and a nonlinear component whose Volterra kernels of order smaller than or equal to p are all zero.*

Definition 7.3.2 *A* pth order linearizer *is a filter which, when connected in cascade before or after a nonlinear system, results in an overall system whose characteristics, in the band of frequencies and in the range of the input signal amplitudes of interest, correspond to those of a parallel connection of a linear system and a nonlinear component whose Volterra kernels of order smaller than or equal to p are all zero.*

When a pth order equalizer (linearizer) is connected before the nonlinear system in (7.8) it is called a *pth order pre-equalizer* (*pth order pre-linearizer*). When it is connected after the nonlinear system, it is called a *pth order post-equalizer* (*pth order post-linearizer*). We now present several theorems that are the counterparts of the results in the previous section to the case of the pth order equalizer and linearizer.

Theorem 7.3.1 *Let the linear inverse filter $A^{-1}(q)$ be bounded input bounded output stable. The sequence of systems defined by*

$$w_1(n) = A^{-1}(q)u(n), \tag{7.24}$$

$$w_p(n) = A^{-1}(q)u(n) - A^{-1}(q)\mathcal{N}[w_{p-1}(n)]; \quad p > 1 \tag{7.25}$$

converges to the system in (7.12) when the input signal is bounded by some finite constant $\tau > 0$. Moreover, the system in (7.25) is a generalized pth order inverse of the system in (7.8) in the sense of Sarti and Pupolin [125].

Recall that the generalized pth order inverse may have Volterra kernels of order larger than p. A cascade connection of a nonlinear system

and its (generalized) pth order inverse results in a nonlinear system that is an identity system plus a residual nonlinearity whose Volterra kernels of order 0 through p are zero.

Proof of Theorem 7.3.1 We prove by induction that (7.25) is the p-th order inverse of (7.8). Let us process the output of the system of (7.8) with the system defined by (7.24) and (7.25). By considering $u(n) = y(n)$, for $p = 1$ we obtain the following input-output relationship:

$$\begin{aligned} w_1(n) &= x(n) + A^{-1}(q)\mathcal{N}[x(n)], \\ &= x(n) + T_1(n), \end{aligned} \tag{7.26}$$

where $T_1(n)$ is a Volterra operator of order greater than 1. This proves that $w_1(n)$ is the output of a first order inverse in the sense of Sarti and Pupolin. Let us suppose that $w_p(n)$ is the output of a pth order inverse, *i.e.*,

$$w_p(n) = x(n) + T_p(n), \tag{7.27}$$

where $T_p(n)$ is a Volterra operator of order greater than p. We want to prove that the system defined by

$$w_{p+1}(n) = A^{-1}(q)u(n) - A^{-1}(q)\mathcal{N}[w_p(n)] \tag{7.28}$$

is a $(p+1)$th order inverse of (7.8). By substituting $y(n)$ in (7.8) for $u(n)$ and (7.27) for $w_p(n)$ in (7.28), we have

$$\begin{aligned} w_{p+1}(n) &= x(n) + A^{-1}(q)\mathcal{N}[x(n)] - A^{-1}(q)\mathcal{N}[x(n) + T_p(n)] \\ &= x(n) + A^{-1}(q)\mathcal{N}[x(n)] - A^{-1}(q)\mathcal{N}[x(n)] + T_{p+1}(n), \end{aligned} \tag{7.29}$$

where T_{p+1} is an operator of order greater than $p+1$ and we have taken into account the fact that $\mathcal{N}[\cdot]$ is an operator of order greater than 1. Thus, we have shown that the sequence of systems defined by (7.24)-(7.25) define pth order inverses of (7.8). If $A^{-1}(q)$ is stable, the sequence of systems (7.24)-(7.25) will converge to (7.11) at least in an amplitude interval around zero. In order to prove convergence we have simply to prove that $\|T_{p+1}(n)\|_\infty$ tends to zero when the input signal $x(n)$ is bounded by some finite constant $\tau > 0$, *i.e.* when $\|x(n)\|_\infty < \tau$. Note that if $A^{-1}(q)$ is stable and $\|x(n)\|_\infty < 1$, it is[1]

$$\|T_1(n)\|_\infty < k_1\|x(n)\|_\infty^2 \tag{7.30}$$

[1]These inequalities can be easily proved following the derivations in [86] or [98].

for some positive constant k_1 and, if we suppose also $\|T_p(n)\|_\infty < 1$,

$$\begin{aligned} \|T_{p+1}(n)\|_\infty &= \|A^{-1}(q)\mathcal{N}[x(n)] - A^{-1}(q)\mathcal{N}[x(n)+T_p(n)]\|_\infty \\ &= \|\mathcal{G}[x(n),T_p(n)]\|_\infty \\ &< k_2\|x(n)\|_\infty\|T_p(n)\|_\infty + k_3\|T_p(n)\|_\infty^2, \end{aligned} \tag{7.31}$$

where k_2 and k_3 are some positive constants and $\mathcal{G}[x(n),T_p(n)]$ is a finite order polynomial with no constant or linear term and with each term containing at least a factor $T_p(n)$. Let k be a positive constant less than 1 and let $\|x(n)\|_\infty < \tau = \max(\frac{1}{k_1}, \frac{k}{k_2+k_3}, 1)$; it is easy to prove by induction that $\|T_p(n)\|_\infty < 1$ for every p and

$$\|T_{p+1}(n)\|_\infty < k^p\|x(n)\|_\infty. \tag{7.32}$$

Thus, $\|T_{p+1}(n)\|_\infty$ converges to zero when $\|x(n)\|_\infty < \tau$. □

The sequence (7.24)-(7.25) corresponds to a systolic cascade of cells as shown in Figure 7.4. Thus the pth order inverse can be easily implemented using VLSI circuits. Furthermore, each cell is always realizable while we recall that the ideal equalizer is not always realizable for the system in (7.8).

The following theorems deal with the pth order inverses and linearizers for systems with band-limited input signals, and can be proved in a manner similar to the derivations for Theorem 7.3.1.

Theorem 7.3.2 *Let the input signal $x(n)$ to the system in (7.8) be band-limited to a certain frequency band B and let $u(n)$ be its output. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside B. If $\tilde{A}^{-1}(q)$ is stable, the sequence of systems defined by*

$$w_1(n) = \tilde{A}^{-1}(q)u(n), \tag{7.33}$$

$$w_p(n) = \tilde{A}^{-1}(q)u(n) - \tilde{A}^{-1}(q)\mathcal{N}[w_{p-1}(n)]; \quad p > 1 \tag{7.34}$$

converges to the system (7.13) when $u(n)$ is bounded by some finite constant $\tau > 0$. Moreover, the system in (7.34) is a pth order post-equalizer for the system in (7.8).

Theorem 7.3.3 *Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside B. If $\tilde{A}^{-1}(q)$ is stable, the sequence of systems defined by*

$$w_1(n) = \tilde{A}^{-1}(q)u(n), \tag{7.35}$$

$$w_p(n) = \tilde{A}^{-1}(q)u(n) + \tilde{A}^{-1}(q)\mathcal{N}[w_{p-1}(n)]; \quad p > 1 \tag{7.36}$$

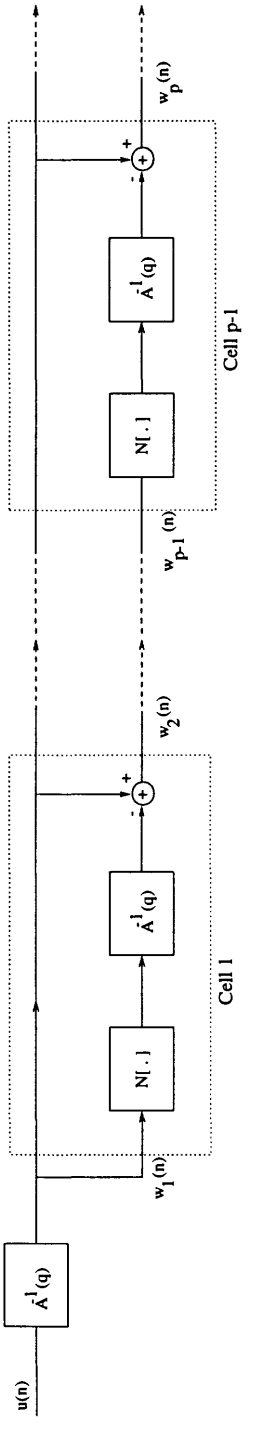


Figure 7.4: The pth order inverse implemented as a cascade connection of systolic cells.

converges to the system in (7.17) when $u(n)$ is bounded by some finite constant $\tau > 0$. Moreover, the system in (7.36) is a pth order pre-equalizer of the system in (7.8).

The block-diagrams of the pth order pre- and post- equalizers are identical to that in Figure 7.4 with the difference that the system $A^{-1}(q)$ is now substituted by the linear equalizer $\tilde{A}^{-1}(q)$.

Theorem 7.3.4 *Let the input signal $x(n)$ to the system in (7.8) be band-limited to a certain frequency band B and let $v(n)$ be its output. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the frequency band B with zero response outside B. If $\tilde{A}^{-1}(q)$ is stable, the sequence of systems defined by*

$$w_1(n) = v(n), \tag{7.37}$$
$$w_p(n) = v(n) - \mathcal{N}[\tilde{A}^{-1}(q)w_{p-1}(n)]; \quad p > 1 \tag{7.38}$$

converges to the system in (7.22) when $v(n)$ is bounded by some finite constant $\tau > 0$. Moreover, the system in (7.38) is a pth order post-linearizer of the system in (7.8).

Theorem 7.3.5 *Let the input signal $v(n)$ be band-limited. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside B. If $\tilde{A}^{-1}(q)$ is stable, the sequence of systems defined by*

$$w_1(n) = v(n), \tag{7.39}$$
$$w_p(n) = v(n) - \tilde{A}^{-1}(q)\mathcal{N}[w_{p-1}(n)]; \quad p > 1 \tag{7.40}$$

converges to the system in (7.23) when $v(n)$ is bounded by some finite constant $\tau > 0$. Moreover, the system in (7.40) is a pth order pre-linearizer of the system in (7.8).

In a similar manner we can define the pth order pre- and post- linearizers. The structure of the pre-linearizer is the same as in Figure 7.4, with the difference that the initial linear block $A^{-1}(q)$ is now absent. A block diagram of the post-linearizer is shown in Figure 7.5.

The pth order equalizers/linearizers of Figures 7.4 and 7.5 are quite attractive because of their systolic structure which allows for modularity as well as cell reuse in hardware realizations. However, if we are interested only on a pth order inversion/equalization, the structure of Figures 7.4 and 7.5 is slightly redundant.

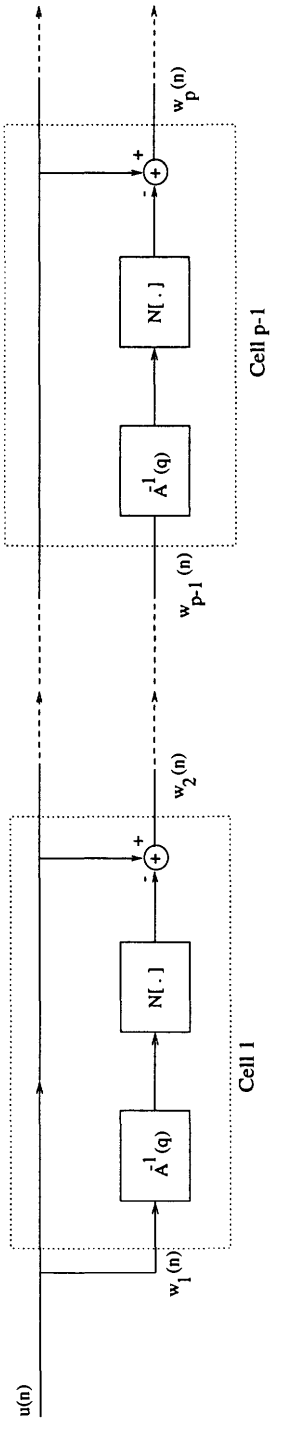


Figure 7.5: The pth order post-linearizer.

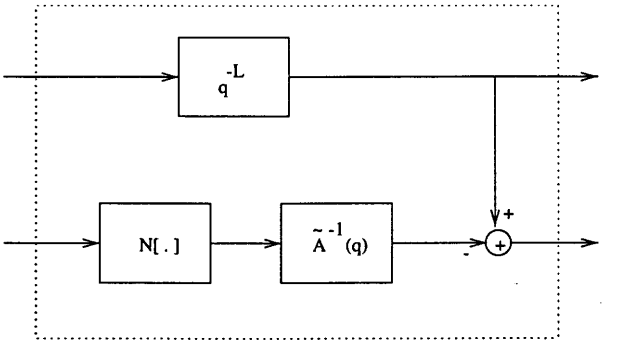


Figure 7.6: The delayed cell.

Remark 7.3.1 *Let us number the nonlinear cells in Figure 7.4 from the left to the right from 1 to* $(p-1)$. *We can drop all the Volterra kernels of* $\mathcal{N}[w_i(n)]$ *of order greater than* $i+1$ *from the* i*th cell and still obtain a generalized* p*th order inverse in the sense of Sarti and Pupolin. This is a direct consequence of the fact that the Volterra kernels of order larger than* $i+1$ *in the* i*th cell generates no nonlinear component of order lower than* $p+1$ *at the output.*

The pth order equalizers and linearizers discussed in this section presents some disadvantages when compared with the ideal equalizers and linearizers, but exhibit several characteristics that may make them more useful in practical applications. The pth order inverses are computationally more expensive than the ideal inverses. The computational cost is almost p times larger for pth order systems. They also do not perfectly compensate for the distortions introduced by the nonlinearities since they do not attempt to reduce distortions of order larger than p. The advantage of the pth order equalizers and linearizers is that we can guarantee their stability in many practical situations. One other advantage of the pth order systems is that it is possible to delay the response of the linear equalizer in order to simplify its design. In the case of the ideal equalizers the response of the linear equalizer cannot be delayed because the linear equalizer appears in the feedback loop and the introduction of a delay would completely modify the response of the system. Figure 7.6 presents a cell with delayed equalizer. This cell may be used to replace each of the cells shown in Figure 7.4. Note that the delay q^{-L} of the upper branch is used to compensate for the delay added to the linear equalizer.

7.4 Equalization and Linearization of Recursive Polynomial Systems

In this section, we briefly extend the results of Sections 7.2 and 7.3 to the case of recursive polynomial systems. We consider the following system model:

$$\begin{aligned} y(n) &= \sum_{i=0}^{N} a_i x(n-i) - \sum_{i=1}^{N} b_i y(n-i) + \\ &\sum_{k=2}^{L} \sum_{i_1=1}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} h_{i_1 i_2 \ldots i_k} x(n-i_1)x(n-i_2)\cdot \ldots \cdot x(n-i_k) \\ &- \sum_{k=2}^{L} \sum_{i_1=1}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} r_{i_1 i_2 \ldots i_k} y(n-i_1)y(n-i_2)\cdot \ldots \cdot y(n-i_k) \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x(n-i)y(n-j), \end{aligned} \tag{7.41}$$

where we consider only a mixed product term for ease of presentation. According to equation (7.3) the inverse of the system always exists and is given by

$$\begin{aligned} w(n) &= \frac{1}{a_0}\Bigg[u(n) + \sum_{i=1}^{N} b_i u(n-i) - \sum_{i=1}^{N} a_i w(n-i) + \\ &\sum_{k=2}^{L} \sum_{i_1=1}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} r_{i_1 i_2 \ldots i_k} u(n-i_1)u(n-i_2)\cdot \ldots \cdot u(n-i_k) \\ &- \sum_{k=2}^{L} \sum_{i_1=1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} h_{i_1 i_2 \ldots i_k} w(n-i_1)w(n-i_2)\cdot \ldots \cdot w(n-i_k) \\ &- \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} w(n-i)u(n-j)\Bigg]. \end{aligned} \tag{7.42}$$

An implicit expression for the inverse of the more general model

$$\begin{aligned} y(n) &= \sum_{i=0}^{N} a_i x(n-i) - \sum_{i=1}^{N} b_i y(n-i) + \\ &\sum_{k=2}^{L} \sum_{i_1=0}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} h_{i_1 i_2 \ldots i_k} x(n-i_1)x(n-i_2)\cdot \ldots \cdot x(n-i_k) \\ &- \sum_{k=2}^{L} \sum_{i_1=1}^{N} \sum_{i_2=i_1}^{N} \cdots \sum_{i_k=i_{k-1}}^{N} r_{i_1 i_2 \ldots i_k} y(n-i_1)y(n-i_2)\cdot \ldots \cdot y(n-i_k) \end{aligned}$$

$$+\sum_{i=0}^{N}\sum_{j=1}^{N} c_{ij}x(n-i)y(n-j), \tag{7.43}$$

if it exists, is given by

$$\begin{aligned} w(n) \;=\; \frac{1}{a_0}\Bigg[& u(n)+\sum_{i=1}^{N} b_i u(n-i)-\sum_{i=1}^{N} a_i w(n-i)+ \\ & \sum_{k=2}^{L}\sum_{i_1=1}^{N}\sum_{i_2=i_1}^{N}\cdots\sum_{i_k=i_{k-1}}^{N} r_{i_1 i_2\ldots i_k}u(n-i_1)u(n-i_2)\cdot\ldots\cdot u(n-i_k) \\ & -\sum_{k=2}^{L}\sum_{i_1=0}^{N}\cdots\sum_{i_k=i_{k-1}}^{N} h_{i_1 i_2\ldots i_k}w(n-i_1)w(n-i_2)\cdot\ldots\cdot w(n-i_k) \\ & -\sum_{i=0}^{N}\sum_{j=1}^{N} c_{ij}w(n-i)u(n-j)\Bigg]. \end{aligned} \tag{7.44}$$

In what follows, we employ the compact expression given by

$$B(q)y(n) = A(q)x(n) + \mathcal{N}[x(n), y(n)] \tag{7.45}$$

to represent the system in (7.41) and (7.43). In the above expression,

$$A(q) = \sum_{i=0}^{N} a_i q^{-i}, \tag{7.46}$$

$$B(q) = 1 + \sum_{i=1}^{N} b_i q^{-i} \tag{7.47}$$

and

$$\begin{aligned} \mathcal{N}[x(n), y(n)] \;=\; & \sum_{k=2}^{L}\sum_{i_1=r}^{N}\cdots\sum_{i_k=i_{k-1}}^{N} h_{i_1 i_2\ldots i_k}x(n-i_1)\cdot\ldots\cdot x(n-i_k) \\ & -\sum_{k=2}^{L}\sum_{i_1=r}^{N}\cdots\sum_{i_k=i_{k-1}}^{N} r_{i_1 i_2\ldots i_k}y(n-i_1)\cdot\ldots\cdot y(n-i_k) \\ & +\sum_{i=r}^{N}\sum_{j=1}^{N} c_{ij}x(n-i)y(n-j). \end{aligned} \tag{7.48}$$

When r is equal to 1, (7.45) corresponds to the system in (7.41), and when r is equal to 0, the expression corresponds to the system in (7.43). Using the above notation we can express the input-output relationship of the inverse of the system in (7.45) as

$$A(q)w(n) = B(q)u(n) - \mathcal{N}[w(n), u(n)], \tag{7.49}$$

where $u(n)$ and $w(n)$ are the input and output signals respectively of the inverse system. We can also express the output signal as

$$w(n) = A^{-1}(q)B(q)u(n) - A^{-1}(q)\mathcal{N}[w(n), u(n)]. \tag{7.50}$$

The following result holds for the post-equalizer of the system in (7.45).

Theorem 7.4.1 *Let the input signal $x(n)$ of the system in (7.45) be band-limited to a certain frequency band B and let $u(n)$ be its output. Let $\tilde{A}^{-1}(q)$ be the linear equalizer of the system $A(q)$ in the band B with zero response outside B. The system defined by*

$$w(n) = \tilde{A}^{-1}(q)B(q)u(n) - \tilde{A}^{-1}(q)\mathcal{N}[w(n), u(n)], \tag{7.51}$$

is a post-equalizer in the band B for the system of (7.45).

A similar theorem holds for the pre-equalizer.

We are also interested in deriving a pre- or post-linearizing filter. In this part also, the linear system that results from the linearization process will have transfer function equal to the linear part of the nonlinear system that is linearized, *i.e.*, equal to $\frac{A(q)}{B(q)}$. Following the derivations in Section 7.2, it can be easily verified that the ideal pre-linearizing filter is given by the system with the following input-output relationship:

$$w(n) = v(n) - A^{-1}(q)\mathcal{N}[w(n), \frac{A(q)}{B(q)}v(n)], \tag{7.52}$$

where $v(n)$ and $w(n)$ are the input and the output signal, respectively. Similarly, the ideal post-linearizing filter is given by

$$w(n) = v(n) - B^{-1}(q)\mathcal{N}[\frac{B(q)}{A(q)}w(n), v(n)]. \tag{7.53}$$

We can easily extend the results of Theorems 7.2.3 and 7.2.4 to the case of recursive nonlinear systems using (7.52) and (7.53).

Finally, the following theorem holds for the pth order inversion of a recursive nonlinear system.

Theorem 7.4.2 *Let the linear inverse filter $A^{-1}(q)$ be bounded input bounded output stable. The sequence of systems defined by*

$$w_1(n) = \frac{B(q)}{A(q)}u(n), \tag{7.54}$$

$$w_p(n) = w_1(n) - A^{-1}(q)\mathcal{N}[w_{p-1}(n), u(n)] \tag{7.55}$$

converges to the system in (7.50) when $u(n)$ is bounded by some finite constant $\tau > 0$. Moreover, the system in (7.55) is a generalized pth order inverse of the system of (7.45) in the sense of Sarti and Pupolin [125].

Similar theorems can be derived for the pth order equalizers and linearizers also.

7.5 An application in Linearization of Loudspeakers

Harmonic distortion caused by non-ideal behaviour of loudspeakers can significantly affect the perceptual quality of audio signals reproduced by the loudspeaker. In this example, we consider the linearization of the nonlinearities associated with a synthetic loudspeaker using the pth order pre-linearizer of Figure 7.4. Previous work [69] has shown that loudspeaker nonlinearities can be efficiently modeled with good accuracy using low-order, truncated Volterra systems. Many linearization procedures for loudspeakers that have been proposed in literature use pth order linearization [49, 53]. The main causes of harmonic distortions in loudspeakers are the nonuniform flux density of the permanent magnet and the nonlinearity of the suspensions. Such distortions can be controlled by a careful design that imposes expensive constraints or by limiting the output power. Another approach that is less expensive and also does not limit the output power is to use digital linearization techniques.

Typically, a single loudspeaker is modeled with the help of a good microphone or a laser vibrometer. Such an approach does not consider the true environment in which the loudspeakers operate. In general we do not have a single loudspeaker but at least two or more often three loudspeakers that cover the acoustic band. The signal that comes from the power amplifier is separated by a crossover filter in the two/three components that are fed to the corresponding loudspeakers. Since it is much easier to design a low distortion mid frequency (midrange) and high frequency (tweeter) loudspeakers than woofers, problems with harmonic distortions are usually more dominant at low frequencies. It is therefore not unusual to compensate only for the distortions caused by the woofer. However, the woofer will not only generate audio frequency components in its passband, but also in the passband of the mid-frequency loudspeaker due to the harmonic distortion. Such distortions cannot be compensated by the woofer, and must be corrected by the midrange loudspeaker. For this reason, the most reasonable way to operate is not to

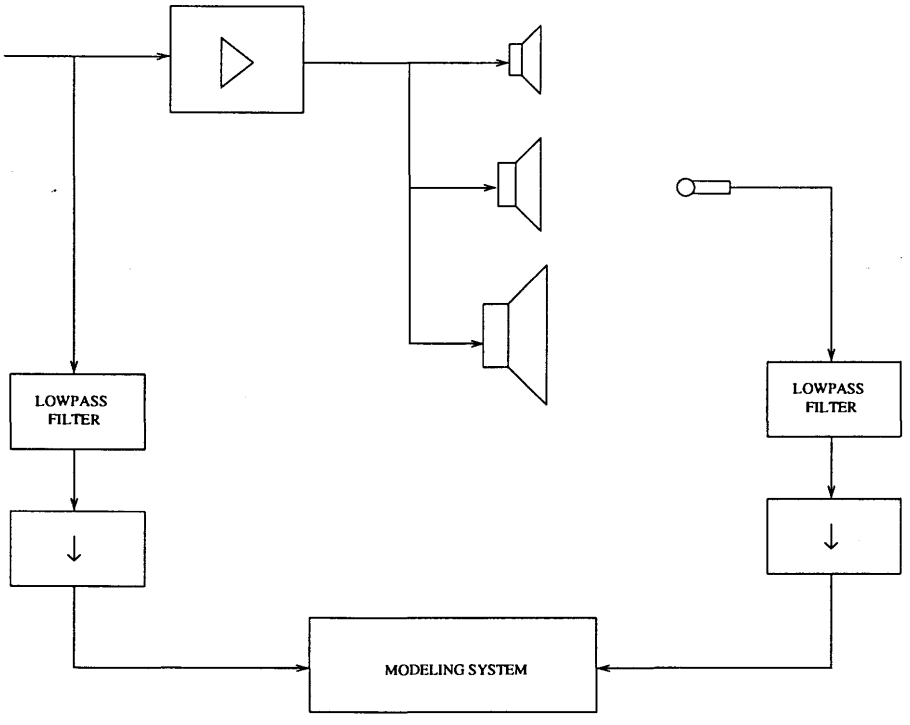


Figure 7.7: Loudspeaker modeling.

model the woofer response alone, but the complete frequency response of the bank of loudspeakers by means of a good quality microphone. In this way the crossover filter and the particular configuration of loudspeakers will be completely transparent to the modeling process. After determining the range of frequencies in which the distortions are severe and have to be compensated, it is convenient to model the loudspeaker system in this band only. To reduce the computational complexity associated with the modeling and linearization process, we have chosen to work with a lower-rate signal obtained by decimating an appropriately bandpass filtered version of the microphone output.

Figure 7.7 displays the block diagram of an experimental set-up. The two lowpass filters of Figure 7.7 used for modeling the loudspeaker system must meet stringent amplitude and linear phase characteristics in the passband in order to avoid large errors in the estimated nonlinear model, which in turn would highly degrade the quality of the linearization. After the loudspeaker is modeled in the range of frequencies where the distortions has to be compensated, the digital circuit of Figure 7.8 is used for the linearization.

The block marked `L.P.1` and `H.P.1` in Figure 7.8 are complementary

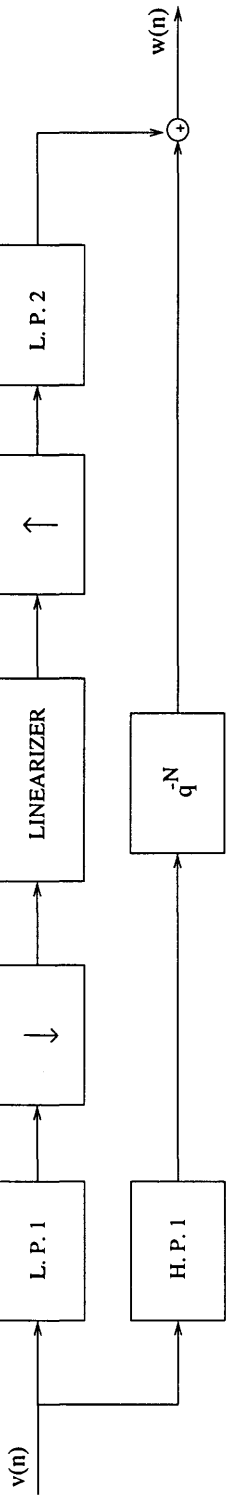


Figure 7.8: The subband linearizer.

filters with lowpass and highpass characteristics, respectively. `L.P.1` selects the frequency band in which the system attempts to compensate for the distortions. The signal then is subsampled with the same decimation factor employed for modeling the loudspeaker and processed by the linearizer that corresponds to the estimated loudspeaker model. By further interpolation and addition of the high frequency component, we obtain the predistorted signal. For the linearization the filter requirements on `L.P.1` and `H.P.1` are not severe. The amplitude and phase distortions of these filters affect the audio signal but not the linearization process. On the other hand, the filter `L.P.2` must meet stringent constant amplitude and a linear phase requirements to provide a good linearization performance. The delay q^{-N} of the lower branch compensate for the delay introduced in the upper branch.

The linearization procedure was simulated with a synthetic loudspeaker system. The loudspeaker was modeled using a second-order truncated Volterra filter with memory lengths 101 and 40 samples for the linear and quadratic parts, respectively. The second-order harmonic distortion of the system is shown in Figure 7.9. Since the distortions were primarily in the range $[0, f_N/3]$ Hz., where f_N denotes the Nyquist frequency, the band $[0, f_N/3]$ was selected by means of a lowpass filter. We employed a PCAS filter [83] for this purpose. The PCAS filter consists of two parallel allpass filters. In our system, the first allpass filter was a simple delay of 100 samples while the the second one was an allpass filter of memory length 99 samples that is used to obtain the desired amplitude and phase characteristics of the overall response. In order to select the high band $[f_N/3, f_N]$ we used the complementary filter for the above lowpass system, which is a PCAS filter also. The output of the lowpass filter was subsampled by a factor of three. The parameters of the loudspeaker were estimated using a quadratic filter with memory length equal to 51 samples for the linear component and 40 samples for the second-order nonlinearity from the subsampled versions of the input to the loudspeaker and its output in the presence of uncorrelated, 30 dB measurement noise. Experiments were conducted with several other values of memory lengths of the linear and quadratic components of the model, and the above values gave the best results for loudspeaker identification. The estimated model was then used to pre-linearize the system using second, third, fourth and fifth order linearizers. The linear equalizer was realized by means of a PCAS filter constituted by two allpass filters with memory lengths 81 samples each.

Figure 7.9 also shows the second-order harmonic distortion measured

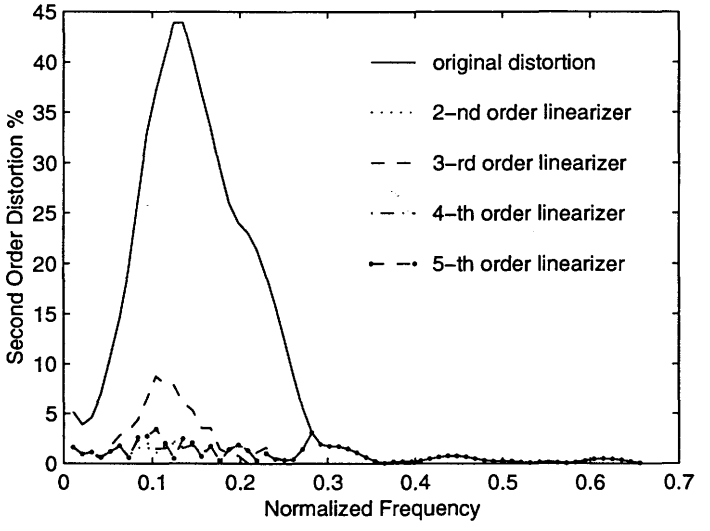


Figure 7.9: The Second Order Distortion.

at the output of the linearized systems. The second-order distortion is the smallest in the case of the second-order linearizer. This linearizer is sufficient to correct for the second-order distortions and it produces the most compact spectrum for the predistorted signal. The third-order linearizer exhibits a higher second-order distortion in this experiment. This is due to model mismatch, our approximations and the wider band of the predistorted signal whose intermodulation contributions alter the amplitude of the fundamental frequency components. The higher-order linearizers exhibit comparable second-order distortions to the second-order linearizer. The improvement due to the use of the linearizer is evident from all the experimental results.

7.6 Concluding Remarks

This chapter presented a theory for the exact and the pth order equalization or linearization of nonlinear systems with known polynomial input-output relationships. An attractive aspect of the results in the chapter is that the equalizers and linearizers can be implemented by cascading modular and stable components. Thus, the pth order equalizers and linearizers can be easily implemented using VLSI circuits. This chapter also included the application of the theory developed in a problem involving linearization to compensate for harmonic distortions introduced by loudspeakers. Adaptive algorithms for equalization and linearization

are under investigation and will be the content of a future paper.

Chapter 8

Conclusions

8.1 Summary

Several contributions to the research areas of Adaptive Filtering and Nonlinear Filtering have been presented in this dissertation.

The candidate has first developed some novel fast and stable RLS algorithms for adaptive linear filtering. The algorithms present a very robust numerical stability combined with an efficient computational complexity. Some of these algorithms have been extended to adaptive Volterra filtering by means of V-vector algebra. This is a novel formalism that allows the development of Volterra and linear multichannel adaptive filter algorithms as an extension of linear adaptive techniques. Successively, contributions have been given to the field of adaptive IIR filtering. In particular, the candidate has developed a sufficient time-varying bound on the maximum variation of the coefficients of an exponentially stable time-varying direct-form homogeneous linear recursive filter. This bound was then applied to control the step size of output error adaptive IIR filters to achieve bounded input bounded output stability of adaptive filters.

Maybe the most important contributions of this dissertation are in the area of nonlinear equalization. The candidate has first derived some theorems for the exact and pth order inversion of a wide class of nonlinear systems. This class includes most causal polynomial systems with finite order as well as many nonlinear filters with nonpolynomial input-output relationship. In particular, it was proved that Volterra filters possess an inverse that in many cases is a recursive polynomial filter. The stability of many polynomial filters is input dependent: if the linear part of the filter is stable and the input signal is sufficiently small the non-

linear filter is stable. In this dissertation we have also presented some theorems that quantify "how much small" the input signal should be in order to guarantee the stability of recursive Volterra filters. Eventually, by exploiting the expression of the exact inverse of recursive or nonrecursive polynomial filters that was recently derived, a theory for the exact and pth order equalization and linearization of nonlinear systems with known recursive or nonrecursive polynomial input-output relationship was developed. The theory is an extension of the standard equalization technique for linear systems. Moreover, the proposed pth order linearizers and equalizers can be implemented by cascading modular and stable components that can be easily realized using VLSI circuits.

8.2 Suggestions for Future Research

All the areas the student has coped with are still very active. Industry needs fast converging stable adaptive linear and nonlinear filters with low computational complexity. The greatest limitation of the class of Lattice QR algorithms is the $O(N)$ number of divisions (where N is the linear filter memory length) which are needed for the adaptation. Divisions are not suited to the architecture found in most digital signal processors. The SFTF algorithm of [135] is a fast RLS algorithm that employs only multiplications but unfortunately stability is ensured only with stationary signals. A very promising class of adaptive filters is that of the Affine Projection Algorithm (APA) [107] and its fast implementations (FAPA) [8, 95]. These algorithms are an extension of the Normalized Least Mean Square (NLMS) algorithm. They require only an order $O(N) + O(P)$ of multiplications (where P is the order of the algorithm, $P = 1$ for the NLMS algorithm) and, when the order P of the algorithm is sufficiently high, they have a speed of convergence that is comparable with that of RLS adaptive filters.

The main limitation of Volterra filters is that their computational complexity increases exponentially with the filter order. In [48] and [49] a very interesting approximation of Volterra filters was presented. In case of a second order Volterra filter, the filter structure of [48, 49] is constituted only of three linear filters with memory length equal to that of the Volterra filter which is approximated. A very interesting research area is the development of adaptive algorithms for this filter structures.

Despite the large amount of work that has been done for adaptive IIR filters, in most cases the performances of the algorithms that have been derived are still unsatisfactory. Due to the high nonlinearity of

the optimization problem, researchers have to cope with problems of biased solution, of local minima, of instability of the filter that results from the identification process. To the author experience one of the most well performing algorithms is that in [116]. The above mentioned problems are avoided in [116], but still the speed of convergence can be unsatisfactory, especially in situations when the filter we want to identify has poles very close to the unit circle. Consequently, much work has still to be done in the area of adaptive IIR filtering.

The stability of recursive polynomial filters is a very interesting subject especially for the implications with nonlinear equalization. Several contributions have been presented in literature [21, 68, 75, 84, 86, 87, 88, 97, 98], but these results in most cases are over conservative, in the sense that the input signal stability region that is identified is much smaller than the real one. Further research has to be done in this area.

Inversion, equalization and linearization on nonlinear systems is a subject of exploding interest in Signal Processing. Novel applications of nonlinear equalization are currently investigated. The theory we have presented allows the exact and pth order equalization and linearization of most recursive polynomial systems with known input-output relationship. The proposed technique applies a linear filter design procedure with combined amplitude and phase specifications in order to design a linear equalizer. Novel and more efficient linear filter design techniques with these characteristics should be developed. The extension of the equalization/linearization theory to adaptive filters is currently under investigation and is another interesting research field.

Bibliography

[1] L. Agarossi, S. Bellini, C. Beoni, P. Migliorati, "Non Linear Effects in High Density Optical Recording," *Int. Workshop on HDTV-96*, 8-9 October 1996, Los Angeles, U.S.A.

[2] O. E. Agazzi and D. G. Messerschmitt, "Nonlinear Echo Cancellation of Data Signals," *IEEE Trans. on Communications*, Vol. COM-30, No. 11, Nov. 1982, pp. 2421-2433

[3] O. E. Agazzi and N. Seshadri, "On the Use of Tentative Decisions to Cancel Intersymbol Interference and Nonlinear Distortion (With Application to Magnetic Recording Channels)," *IEEE Trans. on Information Theory*, Vol. 43, No. 2, March 1997, pp. 394-408

[4] S. T. Alexander, 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, January 1993, pp. 20-30

[5] P. Alper, "A Consideration of the Discrete Volterra Series," *IEEE Trans. on Automatic Control*, Vol. 10, No. 3, July 1965, pp. 322-327

[6] F. Amato, G. Celentano and F. Garofalo, "New Sufficient Conditions for the Stability of Slowly-Varying Linear Systems," *IEEE Trans. on Automat. Contr.*, Vol. 38, No. 9, Sep. 1993, pp. 1409-1411

[7] P. Bauer, M. Mansour and J. Duran "Stability of Polynomials with Time-Variant Coefficients," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 40, No. 6, June 1993, pp. 423-426

[8] B. Baykal and A. G. Constantinides, "Underdetermined-Order Recursive Least-Squares Adaptive Filtering: The Concept and Al-

gorithms," *IEEE Trans. on Signal Processing*, Vol. 45, No. 2, Feb. 1997, pp. 346-362

[9] M. G. Bellanger, "The FLS-QR Algorithm for Adaptive Filtering", *Signal Processing*, Vol. 17, No. 4, Aug. 1989, pp. 291-304

[10] S. Benedetto, E. Biglieri and R. Daffara, "Modeling and Performance Evaluation of Nonlinear Satellite Links - A Volterra Series Approach," *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-15, No. 4, July 1979, pp. 494-507

[11] S. Benedetto and E. Biglieri, "Nonlinear Equalization of Digital Satellite Channels," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. 1 Jan. 1983, pp. 57-62

[12] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977

[13] E. Biglieri, A. Gersho, R. D. Gitlin and T. L. Lim, "Adaptive Cancellation of Nonlinear Intersymbol Interference for Voiceband Data Transmission," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, No. 5, Sep. 1984, pp. 765-777

[14] E. Biglieri, S. Barberis and M. Catena, " Analysis and Compensation of Nonlinearities in Digital Transmission Systems", *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 1, Jan. 1988, pp. 42-51

[15] E. Biglieri, M. Elia and L. Lopresti, "The Optimal Linear Receiving Filter for Digital Transmission Over Nonlinear Channels," *IEEE Trans. on Information Theory*, Vol. 35, No. 3, May 1989, pp. 620-625

[16] E. Biglieri, E. Chiaberto, G. P. Maccone and E. Viterbo "Compensation of Nonlinearities in High-Density Magnetic Recording Channels," *IEEE Trans. on Magnetics*, Vol. 30, No. 6, Nov. 1994, pp. 5079-5086

[17] R. R. Bitmead and B. D. O. Anderson, "Lyapunov Techniques for the Exponential Stability of Linear Difference Equations with Random Coefficients," *IEEE Trans. on Automatic Control,* Vol. AC-25, No. 4, Aug. 1980, pp.782-787

[18] T. Bose and D. A. Trautman "Stability of the Quantized LMS Algorithm," *Circuits Systems and Signal Processing*, Vol. 14, No. 5, 1995, pp. 587-602

[19] T. Bose, M. Q. Chen, "BIBO Stability of the Discrete Bilinear System," *Digital Signal Processing: a Review Journal*, Vol. 5, No. 3, July 1995, pp. 160-166.

[20] A. Carini and E. Mumolo, "A Novel Algebraic Formulation for the Development of Adaptive Volterra Filtering Algorithms," *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, June 20-22 1995, Neos Marmaras, Halkidiki, Greece, pp. 943-946

[21] A. Carini e E. Mumolo, "Adaptive Stabilization of Recursive Second Order Polynomial Filters by Means of a Stability Test," *Proceedings of 1995 IEEE Workshop on Nonlinear Signal and Image Processing*, June 20-22 1995, Neos Marmaras, Halkidiki, Greece, pp. 939-942

[22] A. Carini, "A Novel Givens Rotation Based Fast SQR-RLS Algorithm," *Proc. of EUSIPCO-96, VIII European Signal Processing Conference*, Trieste, Italy, September 10-13 1996, pp. 1235-1238

[23] A. Carini and E. Mumolo, "Fast Square-Root RLS Adaptive Filtering Algorithms," *Signal Processing*, Vol. 57, No. 3, Mar. 1997, pp. 233-250

[24] Alberto Carini, V. John Mathews e Giovanni L. Sicuranza, "Sufficient Stability Bounds for Slowly Varying Discrete-Time Recursive Linear Filters," *Proc. of 1997 International Conference on Acoustics, Speech and Signal Processing*, April 21-24 1997, Munich, Germany

[25] Alberto Carini, Giovanni L. Sicuranza and V. John Mathews "On the Exact Inverse and the pth Order Inverse of Certain Nonlinear Systems," *Proceedings of NSIP 97*, September 7-11 1997, Michigan, USA

[26] A. Carini, G. L. Sicuranza and V. J. Mathews, "On the Inversion of Certain Nonlinear Systems," *IEEE Signal Processing Letters*, Dec. 1997

[27] Alberto Carini, Giovanni L. Sicuranza and V. John Mathews "Equalization and Linearization of Nonlinear Systems," *Proceedings of 1998 International Conference on Acoustics Speech and Signal Processing*, May 12-15 1998, Seattle, Washington, USA.

[28] L. K. Chen, X. Yang and R. R. Mohler, "Stability Analysis of Bilinear Systems," *IEEE Trans. on Automatic Control,* Vol. 36, No.11, Nov. 1991, pp.1310-1315

[29] J. M. Cioffi and T. Kailath, "Fast, Recursive-Least-Squares Transversal Filters for Adaptive Filtering," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-32, No. 2, Apr. 1984, pp. 304-337

[30] J. M. Cioffi and T. Kailath, "Windowed Fast Transversal Filters Adaptive Algorithms with Normalization," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-33, No. 3, June 1985, pp. 607-625

[31] J. M. Cioffi, "The Fast Adaptive ROTOR's RLS Algorithm", *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 38, No. 4, April 1990, pp. 631-653

[32] P. M. Clarkson and M. V. Dokic, "Stability and Convergence Behaviour of Second-Order LMS Volterra Filter," *Electronics Letters*, Vol. 27, No. 5, 28th Feb. 1991, pp. 441-443

[33] M. Dahleh and M. A. Dahleh, "On Slowly Time-Varying Systems," *Automatica*, Vol. 27, No. 1, Jan. 1991, pp. 201-205

[34] C. E. Davila, "An Algorithm for Efficient, Unbiased, Equation-Error Infinite Impulse Response Adaptive Filtering," *IEEE Trans. on Signal Processing*, Vol. 42, No. 5, May 1994, pp. 1221-1226

[35] C. A. Desöer, "Slowly-Varying Discrete System $x_{i+1} = A_i x_i$," *Electronic Letters*, Vol. 6, No. 11, May 1970, pp. 339-340

[36] A. G. Deczky "Synthesis of Recursive Digital Filters Using the Minimum p-Error Criterion," *IEEE Trans. on Audio and Electroacoustics,* Vol. AU-20, No. 4, Oct. 72, pp. 257-263

[37] R. D. DeGroat, L. R. Hunt, D. A. Linebarger and M. Verma, "Discete-Time Nonlinear System Stability" *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 39, No. 10, Oct. 1992, pp. 834-840

[38] P. S. R. Diniz and M. G. Siqueira, "Fixed-Point Error Analysis of the QR-Recursive Least Square Algorithm," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 42, No. 5, May 1995, pp. 334-348

[39] S. C. Douglas, "A Family of Normalized LMS Algorthms," *IEEE Signal Processing Letters*, Vol. 1, No.3, March 1994, pp. 49-51

[40] L. Dugard and G. C. Goodwin, "Global Convergence and Landau's "Output Error with Adjustable Compensator" Adaptive Algorithm," *IEEE Trans. on Automatic Control,* Vol. AC-30, No. 6, June 1985, pp.593-595

[41] P. Fabre and C. Gueguen, "Improvement of the Fast Recursive Least-Squares Algorithms Via Normalization: A Coparative Study," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 2, April 1986, pp. 296-308

[42] D. D. Falconer, "Adaptive Equalization of Channel Nonlinearities in QAM Data Trasmission Systems," *The Bell System Technical Journal*, Vol. 57, No. 7, Sep. 1978, pp. 2589-2611

[43] H. Fan and W. K. Jenkins, "A New Adaptive IIR Filter," *IEEE Tran. on Circuits and Systems*, Vol. CAS-33, No. 10, Oct. 1986, pp. 939-946

[44] H. Fan "Application of Benveniste's Convergence Results in the Study of Adaptive IIR Filtering Algorithms," *IEEE Trans. on Information Theory*, Vol. 34, No. 4, July 1988, pp. 692-709

[45] P. L. Feintuch, "An Adaptive Recursive LMS Filter," *Proc. IEEE,* Vol. 64, No. 11, Nov. 1976, pp. 1622-1624

[46] R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, Vol. 6, No. 2, July 1963, pp. 163-168

[47] F. Fnaiech and L. Ljung, "Recursive Identification of Bilinear Systems," *Int. J. Control*, Vol. 45, No. 2, Feb. 1987, pp. 453-470

[48] W. A. Frank, "MMD - An Efficient Approximation to the 2nd Order Volterra Filter," *Proc. of 1994 International Conference on Acoustics, Speech and Signal Processing*, Adelaide, Australia, April 1994, Vol. 3, pp. 517-520

[49] W. A. Frank, "An Efficient Approximation to the Quadratic Volterra Filter and Its Application in Real-Time Loudspeaker Linearization," *Signal Processing* Elsevier, Vol. 45, No. 1, Jan. 1995, pp. 97-113

[50] W. A. Frank "Sampling Requirements for Volterra System Identification" *IEEE Signal Processing Letters*, Vol. 3, No. 9, Sep. 1996, pp. 266-268

[51] E. N. Frantzeskakis, 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, Sep. 1994, pp. 2455-2469

[52] B. Friedlander, "Lattice Filters for Adaptive Processing," *Proc. IEEE*, Vol. 70, No. 8, Aug. 1982, pp.829-867

[53] F. X. Y. Gao and W. M. Snelgrove, "Adaptive Linearization of a Loudspeaker," *Proceedings of the 1991 International Conference on Acoustics Speech and Signal Processing*, Toronto, Canada, May 1991, pp. 3589-3592

[54] F. X. Y. Gao and W. M. Snelgrove, "An Adaptive Backpropagation Cascade IIR Filter," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 39, No. 9, Sep. 1992, pp. 606-610

[55] F. X. Y. Gao and W. M. Snelgrove, "Adaptive Nonlinear Recursive State-Space Filters," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 41, No. 11, Nov.1994, pp. 760-764

[56] G. B. Giannakis and E. Serpedin, "Linear Multichannel Blind Equalizers of Nonlinear FIR Volterra Channels," *IEEE Trans. on Signal Processing*, Vol. 45, No. 1, Jan. 1997, pp. 67-81

[57] C. Gong and S. Thompson, "Stability Margin Evaluation for Uncertain Systems," *IEEE Trans. on Automatic Control,* Vol. 39, No. 3, March 1994, pp. 548-550

[58] A. H. Gray Jr. and J. D. Markel, "Digital Lattice and Ladder Filter Synthesis," *IEEE Trans. on Audio and Electroacoustics*, Vol. AU-21, No. 6, Dec. 1973, pp. 491-500

[59] A. H. Gray Jr. and J. D. Markel, "A Normalized Digital Filter Structure," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-23, No. 3, June 1975, pp. 268-277

[60] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

[61] R. Hermann, "Volterra Modeling of Digital Magnetic Saturation Recording Channels," *IEEE Trans. on Magnetics,* Vol. 26, No. 5, Sep. 1990, pp. 2125-2127

[62] L. R. Hunt, R. D. DeGroat and D. A. Linebarger, "Nonlinear AR Modeling," *Circuits Systems Signal Processign*, Vol. 14, No. 5, May 1995, pp. 689-705

[63] I. W. Hunter and M. J. Korenberg, "The Identification of Nonlinear Biological Systems: Wiener and Hammerstein Cascade Models," *Biological Cybernetics*, Vol 55, 1986, pp. 135-144

[64] A. Javed, B. A. Syrett and P. A. Goud, "Intermodulation Distortion Analysis of Reflection-Type IMPATT Amplifiers Using Volterra Series Representation," *IEEE Trans. on Microwave Theory and Techniques,* Vol. MTT25, No. 9, Sep. 1977, pp. 729-733

[65] C. R. Johnson Jr, "A Convergence Proof for a Hyperstable Adaptive Recursive Filter," *IEEE Trans. Inform. Theory*, Vol. IT-25, No. 6, Dec. 1979, pp. 745-749

[66] C. R. Johnson Jr and M. G. Larimore, "Comments on and Additions to "An Adaptive Recursive LMS Filter"," *Proc. IEEE*, Vol. 65, No. 9, Sep. 1977, pp. 1399-1402

[67] C. R. Johnson Jr, "Adaptive IIR Filtering: Current Results and Open Issues," *IEEE Trans. on Inform. Theory*, Vol. IT-30, No. 2, March 1984, pp. 237-250

[68] K. K. Johnson and I. W. Sandberg, "The Stability of Bilinear and Quadratic Filters", *Proceedings of NSIP 97*, September 7-11 1997, Michigan, USA

[69] A. J. M. Kaizer "Modeling of the Nonlinear Response of an Electrodynamic Loudspeaker by a Volterra Series Expansion," *AES Journal of Audio Engineering Society*, Vol. 35 No. 6, June 1987, pp. 421-433

[70] E. W. Kamen, P. P. Khargonekar and A. Tannenbaum, "Control of Slowly-Varying Linear Systems," *IEEE Trans. on Automatic Control,* Vol. 34, No. 12, Dec. 1989, pp. 1283-1285

[71] K. I. Kim, E. J. Powers, C. P. Ritz, R. W. Miksad and F. J. Fischer "Modeling of the Nonlinear Drift Oscillations of Moored Vessels Subject to Non-Gaussian Random Sea-Wave Excitation," *IEEE Journal of Oceanic Engineering*, Vol. OE-12, No. 4, Oct. 1987, pp. 568-575

[72] T. Koh and E. J. Powers, "Second-Order Volterra Filtering and Its Application to Nonlinear System Identification," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-33, No. 6, Dec. 1985, pp. 1445-1455

[73] S. .R. Kolla, R. K. Yedavalli and J. B. Farison, "Robust Stability Bounds on Time-Varying Perturbations for State-Spece Models of Linear Discete-Time Systems," *Int. J. Control*, Vol. 50, No. 1, 1989, pp. 151-159

[74] M. J. Korenberg and I. W. Hunter, "The Identification of Nonlinear Biological Systems: LNL Cascade Models," *Biological Cybernetics*, Vol. 55, 1986, pp. 125-135

[75] S. Kotsios and N. Kalouptsidis, "BIBO Stability Criteria for a Certain Class of Discrete Nonlinear Systems," *Int. J. Control,* Vol. 58, No. 3, Sep. 1993, pp. 707-730

[76] G. Kubin, "On the Stability of Wave Digital Filters with Time-Varying Coefficients," *Proc. European Conf. Circuit Theory and Design*, 2-6 September 1985, Prague, Czechoslovakia, pp. 499-502

[77] I. D. Landau,"Unbiased Recursive Identification Using Model Reference Adaptive Techniques," *IEEE Trans. on Automatic Control,* Vol. AC-21, No. 2, Apr. 1976, pp. 194-202

[78] S. W. Lang and J. H. McClellan, "A Simple Proof of Stability for All-Pole Linear Predition Models," *Proc. IEEE*, Vol. 67, No. 5, May 1982, pp.860-861

[79] M. Lang and T. I. Laakso, "Simple and Robust Method for the Design of Allpass Filters Using Least-Squares Phase Error Criterion," *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 41, No. 1, Jan. 1994, pp. 40-48

[80] M. G. Larimore, J. R. Treichler and C. R. Johnson Jr., "SHARF: An algorithm for adapting IIR digital filters," *IEEE Trans. Acoust., Speech, Signal Processing,* Vol. ASSP-30, Apr. 1982, pp. 240-246

[81] D. A. Lawrence and W. J. Rugh, "Input-Output Pseudolinearization for Nonlinear Systems," *IEEE Trans. on Automatic Control,* Vol. 39, No. 11, Nov. 1994, pp. 2207-2218

[82] S. S. Lawson and A. Wicks, "Design of Efficient Digital Filters Satisfying Arbitrary Loss and Delay Specifications," *IEE Proceedings-G,* Vol. 139, No. 5, Oct. 1992, pp. 611-620

[83] S. S. Lawson, "Direct Approach to Design of PCAS Filters with Combined Gain and Phase Specification," *IEE Proc. Vis. Image Signal Process.*, Vol. 141, No. 3, June 1994, pp. 161-167

[84] J. Lee, *Adaptive Polynomial Filtering Algorithms*, PhD Dissertation, University of Utah, 1992

[85] J. Lee and V. J. Mathews, "A Fast Recursive Least Squares Second-Order Volterra Filter and its Performance Analysis," *IEEE Trans. on Signal Proc.*, Vol. 41, No. 3, Mar. 1993, pp. 1087-1101

[86] J. Lee, V. J. Mathews, "A Stability Condition for Certain Bilinear Systems," *IEEE Trans. on Signal Processing,* Vol. 42, No. 7, July 1994, pp. 1871-1873.

[87] J. Lee, V. J. Mathews, "A Stability Result for RLS Adaptive Bilinear Filters," *IEEE Signal Processing Letters,* Vol. 1, No. 12, Dec. 1994, pp. 191-193.

[88] J. Lee, V. J. Mathews, "Output-Error LMS Bilinear Filters with Stability Monitoring", *Proc. of 1995 International Conference on Acoustics, Speech and Signal Processing*, Detroit, Michigan, 8-12 May 1995, pp. 965-968

[89] F. Ling, "Givens Rotation Based Least Squares Lattice and Related Algorithms", *IEEE Transaction on Signal Processing*, Vol. 39, No. 7, July 1991, pp. 1541-1551

[90] Z. Liu, "QR Methods of O(N) Complexity in Adaptive Parameters Estimation", *IEEE Transactions on Signal Processing*, Vol. 43, No. 3, March 1995, pp. 720-728

[91] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, M.I.T. Press, Cambridge MA, 1983.

[92] V. J. Mathews, "Adaptive Polynomial Filters", *IEEE Signal Proc. Magazine*, Vol. 8, No. 3, July 1991, pp. 10-26

[93] K. X. Miao, H. Fan and M. Doroslovački, "Cascade Lattice IIR Adaptive Filters," *IEEE Trans. Signal Processing*, Vol. 42, No. 4, Apr. 1994, pp. 721-742

[94] S. K. Mitra and J. F. Kaiser, *Handbook for Digital Signal Processing*, John Wiley and Sons, New York NY, 1993

[95] D. R. Morgan and S. G. Kratzer, "On a Class of Computationally Efficient, Rapidly Converging, Generalized NLMS Algorithms," *IEEE Signal Processing Letters*, Vol. 3, No. 8 Aug. 1996, pp. 245-247

[96] E. Mumolo and A. Carini, "Volterra Adaptive Prediction of Speech with Application to Waveform Coding," *European Transactions on Telecomunications*, Vol. 6, No. 6, Nov.-Dec. 1995, pp. 685-693

[97] E. Mumolo and A. Carini, "Recursive Volterra Filters with Stability Monitoring," *Proceedings of EUSIPCO 96*, September 10-13 1996, Trieste, Italia

[98] E. Mumolo, A. Carini, "A Stability Condition for Adaptive Recursive Second-Order Polynomial Filters," *Signal Processing*, Vol. 54, No. 1, Oct. 1996, pp. 85-90.

[99] S. Narayanan, "Transistor Distortion Analysis Using Volterra Series Representation," *Bell Systems Technical Journal,* Vol. 46, May-June 1967, pp. 991-1204

[100] S. Narayanan, "Application of Volterra Series to Intermodulation Distortion of Transistor Feedback Amplifiers," *IEEE Trans. on Circuit Theory,* Vol. CT-17, Nov.1970, pp. 518-527

[101] A. Nehorai and D. Starer, "Adaptive Pole Estimation," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 38, No. 5, May 1990, pp. 825-838

[102] S. L. Netto, P. S. R. Diniz and P. Agathoklis, "Adaptive IIR Filtering Algorithms for System Identification: A General Framework," *IEEE Trans. on Education*, Vol. 38, No. 1, Feb. 1995, pp. 54-66

[103] R. D. Nowak and B. D. Van Veen, "Random and Pseudorandom Inputs for Volterra Filter Identification," *IEEE Trans. on Signal Processing*, Vol. 42, No. 8, Aug. 1994, pp. 2124-2135

[104] R. D. Nowak and B. D. Van Veen, "Invertibility of Higer Order Moment Matrices," *IEEE Trans. on Signal Processing*, Vol. 43, No. 3, March 1995, pp. 705-708

[105] R. D. Nowak and B. D. Van Veen, "Volterra Filter Equalization: A Fixed Point Approach," *IEEE Trans. on Signal Processing*, Vol. 45, No. 2, Feb. 1997, pp. 377-388

[106] S. J. Orfanidis and L. M. Vail "Zero-Tracking Adaptive Filters," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 6, Dec. 1986, pp. 1566-1571

[107] K. Ozeki and T. Umeda, "An Adaptive Filtering Algorithm Using an Orthogonal Projection to an Affine Subspace and Its Properties," *Electronics and Communications in Japan* Vol. 67-A, No. 5, 1984, pp. 19-27

[108] P. Park and T. Kailath, "A Lattice Algorithm Dual to the Extended Inverse QR Algorithm," *Signal Processing*, Vol. 47, 1995, pp. 115-133

[109] K. A. Prabhu, "A Predictor Switching Scheme for DPCM Coding of Video Signals," *IEEE Trans. on Communications,* Vol. COM-33, No. 4, April 1985, pp. 373-379

[110] 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, August 1991, pp. 341-353

[111] I. K. Proudler, "Fast Time-Series Adaptive-Filtering Algorithm Based on the QRD Inverse-Updates Method", *IEE Proc. Vis. Image Signal Process.*, Vol. 141, No. 5, Oct. 1994, pp. 325-332

[112] G. Ramponi and G. L. Sicuranza, "Decision-Directed Nonlinear Filters for Image Processing," *Electronic Letters*, Vol. 23, No. 23, Nov. 5 1987, pp. 1218-1219

[113] G. M. Raz and B. D. Van Veen, "Baseband Volterra Filters for Implementing Carrier Based Nonlinearities," *Proceedings of NSIP 97*, September 7-11 1997, Michigan, USA

[114] P. A. Regalia, S. K. Mitra and P. P. Vaidyanathan, "The Digital All-Pass Filter: A Versatile Signal Processing Building Block," *Proc. IEEE*, Vol. 76, No. 1, jan. 1988, pp.19-37

[115] P. A. Regalia and M. G. Bellanger, "On the Duality Between fast QR Methods and Lattice Methods in Least Squares Adaptive Filtering", *IEEE Transaction on Signal Process.*, Vol. 39, No. 4, April 1991, pp. 879-891

[116] P. A. Regalia, "Stable and Efficient Lattice Algorithms for Adaptive IIR Filtering," *IEEE Trans. on Signal Processing*, Vol. 40, No. 2, Feb. 1992, pp. 375-388

[117] P. A. Regalia, "Numerical Stability Properties of a QR-Based Fast Least Squares Algorithm," *IEEE Trans. on Signal Processing*, Vol. 41, No. 6, June 1993, pp. 2096-2109

[118] P. A. Regalia, "An Unbiased Equation Error Identifier and Reduced-Order Approximations," *IEEE Trans. on Signal Processing*, Vol. 42, No. 6, June 1994, pp. 1397-1412

[119] P. A. Regalia, *Adaptive IIR Filtering in Signal Processing and Control*, Marcel Dekker Inc., New York, 1995.

[120] M. Renfors, S. K. Mitra, P. A. Regalia and Y. Neuvo, "Generalized Doubly Complementary IIR Digital Filters," *Signal Processing* Vol. 29, 1992, pp. 173-181

[121] A. A. Rontogiannis and S. Theodoridis, "A New Highly Parallel Multichannel Fast QRD-LS Adaptive Algorithm," *Proc. of EUSIPCO-96, VIII European Signal Processing Conference*, Trieste, Italy, Sep. 10-13 1996, pp. 1381-1384

[122] B. H. H. Rosenbrock, "The Stability of Linear Time-Dependent Control Systems," *J. Electron. and Control*, Vol. 15, No. 1, July 1963, pp. 73-80

[123] W. J. Rugh, *Linear System Theory*, Prentice Hall, Upper Saddle River NJ, 1995.

[124] M. Rupp, "Normalization and Convergence of Gradient-Based Algorithms for Adaptive IIR Filters," *Signal Processing*, Vol. 46, 1995, pp. 15-30

[125] A. Sarti and S. Pupolin, "Recursive Techniques for the Synthesis of a p^{th}-Order Inverse of a Volterra System," *European Trans. on Telecomm.*, Vol. 3, No. 4, Jul.-Aug. 1992, pp. 315-322.

[126] M. Schetzen, "Theory of pth-Order Inverses of Nonlinear Systems," *IEEE Trans. on Circuits and Systems*, Vol. CAS-23, No. 5, May 1976, pp. 285-291.

[127] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, John Wiley and Sons, Inc. New York, NY, 1980.

[128] H. Schurer, A .P. Berkhoff, C. H. Slump and O. E. Herrmann, "Modeling and Compensation of Nonlinear Distortion in Horn Loudspeaker," *Proceedings of EUSIPCO '94*, Signal Processing VII: Theories and Applications, pp. 1468-1471.

[129] H. Schütze and Z. Ren, "Easy and Effective Stabilization Measure for Fast Recursive Least Squares Algorithms for Adaptive Transversal Filters," *Electronics Letters*, Vol. 27, No. 16, 1st August 1991, pp. 1397-1399

[130] J. J. Shynk, "Adaptive IIR Filtering," *IEEE ASSP Mag.*, Vol. 6, No. 2, April 1989, pp. 4-21

[131] J. J. Shynk, "Adaptive IIR Filtering Using Parallel-Form Realizations," *IEEE Trans. Acoust., Speech, Signal Processing,* vol.37, No. 4, Apr. 1989, pp. 519-533

[132] G. L. Sicuranza, "Quadratic Filters for Signal Processing", *Proc. IEEE*, Vol. 80, No. 8, August 1992, pp. 1263-1285

[133] T. Siu, M. Schetzen, "Convergence of Volterra Series Representation and BIBO Stability of Bilinear Systems," *Int. J. Systems Science*, Vol. 22, No. 12, Dec. 1991, pp. 2679-2684.

[134] D. T. M. Slock and T. Kailath, "Fast Transversal Filters with Data Sequence Weighting," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. 37, No. 3, March 1989, pp. 346-359

[135] D. T. M. Slock and T. Kailath, "Numerically Stable Fast Transversal Filters for Recursive Least Squares Adaptive Filtering", *IEEE Transactions on Signal Processing*, Vol. 39, No. 1, Jan. 1991, pp. 92-113

[136] D. T. M. Slock and K. Maouche, "The Fast Subsampled-Updating Stabilized Fast Transversal Filter (FSU SFTF) RLS Algorithm," *Proceedings of EUSIPCO '94*, Signal Processing VII: Theories and Applications, pp. 740-743.

[137] T. Söderström and P. Stoica, "Some Properties of the Output Error Method," *Automatica*, Vol. 18, No. 1, Jan. 1982, pp. 93-99

[138] D. Starer, A. Nehorai, "Adaptive Polynomial Factorization by Coefficient Matching", *IEEE Trans. on Signal Processing*, Vol. 39, N.2, Feb.1991, pp. 527-530

[139] S. D. Stearns "Error Surfaces of Recursive Adaptive Filters," *IEEE Trans. on Acoustics, Speech and Signal Processing*, Vol. ASSP-29, No. 3, June 1981, pp. 763-766

[140] K. Steiglitz and L. E. McBride, "A Technique for the Identification of Linear Systems," *IEEE Trans. on Automatic Control,* Vol. AC-10, Oct. 1965, pp. 461-464

[141] P. Stoica and T. Söderström, "The Steiglitz-McBride Identification Algorithm Revisited- Convergence Analysis and Accuracy Aspects," *IEEE Trans. on Automatic Control,* Vol. AC-26, No. 3, June 1987, pp. 712-717

[142] V. L. Stonick, "Time-Varying Performance Surfaces for Adaptive IIR Filters: Geometric Properties and Implications for Filter Stability," *IEEE Trans. on Signal Processing*, Vol. 43, No. 1, Jan. 1995, pp. 29-42

[143] P. Strobach, "The Square-Root Schur RLS Adaptive Filter", *Proc. of 1991 International Conference on Acoustics, Speech and Signal Processing*, pp. 1845-1848.

[144] P. Strobach, "Two-fold Normalized Square-Root Schur RLS Adaptive Filter", *IEEE Transactions on Signal Processing*, Vol. 42, No. 5, May 1994, pp. 1230-1233

[145] J. H. Su, "Comments on "Stability Margin Evaluation for Uncertain Systems" " *IEEE Trans. on Automatic Control,* Vol. 39, No. 12, Dec. 1994, pp. 2523-2524

[146] M. A. Syed and V. J. Mathews, "QR-Decomposition Based Algorithms for Adaptive Volterra Filtering," *IEEE Trans. on Circuits and Systems–I: Fundamental Theory and Applications*, Vol. 40, No. 6, June 1993, pp. 372-382

[147] M. A. Syed and V. J. Mathews, "Lattice Algorithms for Recursive Least Squares Adaptive Second Order Volterra Filtering," *IEEE Trans. on Circuits and Systems–II: Analog and Digital Signal Processing*, Vol. 41, No. 3, Mar. 1994, pp. 202-214

[148] M. Tarrab and A. Feuer, "Convergence and Performance Analysis of the Normalized LMS Algorithm with Uncorrelated Gaussian Data," *IEEE Trans. on Information Theory*, Vol. 34, No. 4, July 1988, pp. 680-691

[149] M. Terrè and M. Bellanger, "A Fast Least Squares QRD-Based Algorithm for Complex Data", *IEEE Transaction on Signal Processing*, Vol. 42, No. 11, Nov. 1994, pp. 3272-3273

[150] B. E. J. Thomas, "An Adaptive Echo Canceller in a Nonideal Environment (Nonlinear or Time Variant)," *The Bell System Technical Journal*, Vol. 50, No. 8, Oct. 1971, pp. 2779-2805

[151] C. H. Tseng and E. J. Powers, "Nonlinear Channel Equalization in Digital Satellite Systems," *Proceedings of 1993 IEEE Global Telecommunication Conference*, Nov. 29-DEc. 2 1993, Houston, Tx, USA

[152] J. R. Treichler, C. R. Johnson Jr and M. G. Larimore, *Theory and Design of Adaptive Filters*, A Wiley-Interscience publication, New York, 1987

[153] B. Widrow and J. M. McCool, "Comments on "An Adaptive Recursive LMS Filter"," *Proc. IEEE*, Vol. 65, No. 9, Sep. 1977, pp. 1402-1404

[154] B. Widrow and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation," *Proc. IEEE*, Vol. 78, No. 9, Sep. 1990, pp. 1415-1442

[155] G. A. Williamson, B. D. O. Anderson and C. R. Johnson Jr., "On the Local Stability Properties of Adaptive Parameter Estimators with Composite Errors and Split Algorithms," *IEEE Trans. on Automatic Control*, Vol. 36, No. 4, Apr. 1991, pp. 463-473

[156] G. A. Williamson, "Gradient-Descent Adaptation of Cascade-Form Adaptive Filters," *Proc. of 27th Asilomar Conference*, Pacific Grove, CA, November 1993, pp. 1-5

[157] R. Zarour and M. M. Fahmy, "A design Technique for Variable Digital Filters," *IEEE Trans. on Circuits and Systems*, Vol. 36, No. 11, Nov. 1989, pp. 1473-1478.

[158] D. Zwillinger, *Standard Mathematical Tables and Formulae*, CRC Press, Boca Raton, Florida, 1996