ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

ADAPTIVE VOLTERRA FILTERING WITH COMPLETE LATTICE ORTHOGONALIZATION

Ph.D. THESIS

Mehmet Tahir ÖZDEN

Date of Submission : 29 November 1995
Date of Presentation : 13 March 1996

Thesis Advisor : Prof. Dr. Ahmet H. KAYRAN
Other Jury Members : Prof. Dr. Erdal PANAYIRCI
: Prof. Dr. Yorgo İSTEFANOPULOS
: Prof. Dr. Bülent SANKUR
: Doç. Dr. Cüneyt GÜZELİŞ

MART 1996

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TAM KAFES DİKLEŞTİRMESİ İLE UYARLANIR VOLTERRA SÜZGEÇLEMESİ

DOKTORA TEZİ

Mehmet Tahir ÖZDEN

Tezin Enstitüye Verildiği Tarih : 29 Kasım 1995
Tezin Savunulduğu Tarih : 13 Mart 1996

Tez Danışmanı : Prof. Dr. Ahmet H. KAYRAN
Diğer Jüri Üyeleri : Prof. Dr. Erdal PANAYIRCI
: Prof. Dr. Yorgo İSTEFANOPULOS
: Prof. Dr. Bülent SANKUR
: Doç. Dr. Cüneyt GÜZELİŞ

MART 1996

ACKNOWLEDGEMENTS

I am greatly indepted to Prof. Dr. Ahmet Hamdi KAYRAN and Prof. Dr. Erdal PANAYIRCI for guiding and supporting me during the research and edition phases of my thesis. Their helpful and constructive critism all along is highly appreciated. I would especially like to thank my family, who have been patient and understanding beyond measure.

CONTENTS

LIST OF ABBREVIATIONS

LMS	LEAST MEAN SQUARE
LS	LEAST SQUARES
RLS	RECURSIVE LEAST SQUARES
VLSI	VERY LARGE SCALE SCALE
SMPLS	SEQUENTIAL PROCESSING MULTICHANNEL LATTICE STAGE
PSK	PHASE SHIFT KEYING
GAL	GRADIENT ADAPTIVE LATTICE
SOP	SELF ORTHOGONAL PROCESSOR
ROP	REFERENCE ORTHOGONAL PROCESSOR
FPE	FORWARD PREDICTION ERROR
BPE	BACKWARD PREDICTION ERROR
TDMA	TIME DIVISION MULTIPLE ACCESS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis presents a new recursive least squares (RLS) adaptive nonlinear filter, based on Volterra series expansion. The main approach is to transform the nonlinear filtering problem into an equivalent multichannel, but linear, filtering problem. Then, the multichannel input signal is completely orthogonalized using sequential processing multichannel lattice stages. With the complete orthogonalization of the input signal, only scalar operations are required, instability problems due to matrix inversion are avoided and, good numerical properties are achieved. Avoidance of matrix inversion and vector operations reduce the complexity considerably, make the filter simple, highly modular and suitable for VLSI implementation.

First, adaptive filtering algorithms employing truncated Volterra series representation of nonlinear systems are considered. The LMS and RLS adaptive second-order Volterra filtering algorithms are presented. The Gram-Schmidth orthogonalization algorithm is reviewed. The new lattice structure, sequential processing multichannel lattice stages, is introduced. The implementation of the modified Gram-Schmidth orthogonalization algorithm with SPMLS's is given. The nonlinear filtering problem is transformed into an equivalent multichannel, but linear, adaptive filtering problem. Different forms of the sequential processing multichannel lattice stage algorithm are discussed.

Then, the main contribution appears as the application of sequential processing multichannel lattice stages to the second-order Volterra system identification and channel equalization problem. In each case, several experiments demonstrating the fast convergence properties of the filters are included. In system identification, the adaptive filter is defined with the same structure and the same number of coefficients as that of the system that is to be identified and, the performance of the filter is demonstrated with the mean squared error curves and the mean trajectories of estimated coefficients. In channel equalization, the adaptive filter is so defined that it also has second-order

nonlinearity and the performance of the filter is demonstrated with the mean squared error curves and the number of iterations needed for no error reception.

Finally, the equalization and identification of digital satellite channels are investigated. These types of channels have such a nonlinear structure that they can be modeled with higher order Volterra series. The structure of the Volterra filter is so defined that it can identify or equalize higher-order Volterra satellite channels. The performance of the higher-order Volterra filter is tested for the identification and the equalization of a 4-PSK nonlinear channel.

ÖZET

TAM KAFES DİKLEŞTİRMESİ İLE UYARLANIR VOLTERRA SÜZGEÇLEMESİ

Bu tezde, Volterra serisi açılımına dayalı yeni bir özyineli en küçük kareler uyarlanır doğrusal olmayan süzgeç sunulmaktadır. Ana yaklaşım, doğrusal olmayan süzgeçleme problemini, eşdenik çok kanallı, ancak doğrusal, süzgeçleme problemine dönüştürmektir. Daha sonra, çok kanallı giriş işareti, ardışık işlem yapan çok kanallı kafes kademeleri (AİYÇKKK) ile tam olarak dikleştirmektir. Giriş işaretinin tam olarak dikleştirilmesi ile, yalnızca rakkamsal işlemlere ihtiyaç duyulmakta, matris tersi alınmasına bağlı kararsızlık problemleri ortadan kaldırılmakta ve iyi sayısal özellikler elde edilmektedir. Matris tersi işlemi ve vektör işlemlerinin kullanılmaması ile, karmaşıklık oldukça azalmakta, süzgeç basitleşmekte, modüler ve çok büyük ölçekli tümleşim uygulamalarında kullanılabilir hale gelmektedir.

İkinci Bölüm'de, kesik Volterra serileri ile temsil edilen doğrusal olmayan sistemler için uyarlanır süzgeçleme algoritmaları ele alınmıştır. Kısım 2.3 ve 2.4'de LMS ve RLS uyarlanır ikinci derece Volterra süzgeçleme algoritmaları sunulmuştur. Kısım 2.5'de, Gram-Schmidth dikleştirme algoritması gözden geçirilmiş, kafes yapılarının avantajlarından bahsedilmiş ve yeni kafes yapısı, ardışık işlem yapan çok kanallı kafes kademeleri anlatılmıştır. Ayrıca, değiştirilmiş Gram-Schmidth dikleştirme algoritmasının, ardışık işlem yapan çokkanallı kafes kademeleri ile Volterra süzgeçleme problemine uygulanması anlatılmış ; doğrusal olmayan süzgeçleme problemi, eşdenik çok kanallı, ancak doğrusal, süzgeçleme problemine dönüştürülmüş, ardışık işlem yapan çok kanallı kafes kademelerinin farklı formları tartışılmıştır.

Üçüncü Bölüm, AİYÇKKK kullanan, özyineli en küçük kareler Volterra süzgecinin sistem tanıma uygulamasını sunmaktadır. Kısım 3.1'de, genel en küçük kareler uyarlanır Volterra sistem tanıma problemi tanıtılmıştır. Daha sonra, kısım 3.2, AİYÇKKK ile özyineli, uyarlanır ikinci-derece Volterra sistem tanıma

süzgeçlemesini sunmaktadır. Kısım 3.3'de ise, yeni kafes süzgecinin performansını gösteren deney sonuçları bulunmaktadır. Bu deneylerde, uyarlanır süzgeç tanınacak sistem ile aynı yapı ve aynı sayıda katsayı ile çalıştırılmıştır. Algoritmanın, tüm özyineli en küçük kareler algoritmalarının ortak özelliği olan hızlı yakınsama özelliğini paylaştığı, hatta sayısal kararsızlık söz konusu olmaksızın, özyineli en küçük kareler transversal Volterra süzgecine nazaran daha iyi performans gösterdiği gözlenmiştir.

Dördüncü Bölüm'de, Volterra tipi doğrusal olmayan kanalların dengelenmesi düşünülmüştür. Kısım 4.1, genel en küçük kareler uyarlanır ikinci-derece Volterra kanal dengeleme problemini sunmaktadır. Kısım 4.2'de, doğrusal olmayan dengeleme problemi, eşdenik çok kanallı, ancak doğrusal, uyarlanır dengeleme problemine çevrilmektedir. Daha sonra, AİYÇKKK ile özyineli en küçük kareler ikinci-derece Volterra kanal dengeleme problemi ele alınmaktadır. Kısım 4.3, farklı kanal distorsiyonları için, dengeleyicinin performansını gösteren deney sonuçları ile ilgilidir.

Son olarak Beşinci Bölüm'de, sayısal uydu kanallarının dengelenmesi ve tanınması problemi ele alınmaktadır. Bu tip kanallar, daha yüksek dereceden Volterra serileri ile modellenebilen bir doğrusal olmayan yapıya sahiptir. Volterra süzgecinin yapısı, daha yüksek dereceli Volterra kanallarını tanıyacak yada dengeleyecek şekilde tanımlanmıştır. Kanal katsayılarının karmaşık olması ve kanal giriş işaretinin karmaşık olması nedenleri ile, AİYÇKKK algoritmasının karmaşık formu kullanılmıştır. Daha yüksek dereceli Volterra süzgecinin performansı, faz kaydırmalı anahtarlamalı (PSK) modülasyonlu doğrusal olmayan bir uydu kanalı örneği ile deney yapılarak incelenmiştir. Sonuçlar, daha yüksek dereceli Volterra dengeleyicisinin, ikinci derece Volterra dengeleyicisine nazaran daha geç yakınsadığını göstermektedir. Kanal tanımada ise performans değişmemektedir.

CHAPTER 1

INTRODUCTION

Linear filters have played a very crucial role in the development of various signal processing techniques. The obvious advantage of linear filters is their inherent simplicity. Design, analysis, and implementation of such filters are relatively straightforward tasks in many applications. However, there are several situations in which the performance of linear filters is unacceptable. A simple but highly pervasive type of nonlinearity is the saturation-type nonlinearity. Trying to identify these types of systems using linear models can often give misleading results. Another situation where nonlinear models will do well when linear models will fail miserably is that of trying to relate two signals with nonoverlapping spectral components. In digital satellite links, the satellite amplifiers are usually driven to near the saturation point and they exhibit highly nonlinear characteristics.

When confronted with a nonlinear systems problem, many engineers shy away from the situation, hoping that the problem will go away, mainly because the solutions are often difficult from analytical and/or computational point of view. Moreover, the rich variety of highly developed tools available for solving linear systems engineering problems are just not there when it comes to most nonlinear systems problems. The difficulties mentioned above are much more magnified in the case of adaptive nonlinear systems.

Unlike the case of linear systems which are completely characterized by the system's unit impulse response function, it is impossible to find a unified framework for describing arbitrary nonlinear systems. Consequently, the researchers working on nonlinear filters are forced to restrict themselves to certain nonlinear system models that are less general.

Application of Volterra system theory has played an ever increasing role in nonlinear system modelling. This due, in part to the fact that Volterra series has firm mathematical foundation and, in many cases, a gently (or lower order) nonlinear time-invariant system can be described with reasonable accuracy by a truncated version of the Volterra series, which considerably reduces the complexity of the problem.

Furthermore, since the output of a Volterra filter depends linearly on the linear, quadratic, and higher-order filter coefficients (but nonlinearly on the input), many concepts originally developed for linear filters can be extended to Volterra filters.

1.1 LITERATURE REVIEW

Several researchers have used Volterra series representation of nonlinear systems to implement nonlinear channel equalizers [1,2,3,4,5]. Other applications of nonlinear models and filtering in communication problems include echo cancellation [6,7,8,9], performance analysis of data transmission systems [10,11,12,13], adaptive noise cancellation [14,15], and detection of nonlinear functions of Gaussian processes [16]. Nonlinear filters are very useful in modeling biological phenomena [17,18], myoelectric signal processing [19], characterization of semiconductor devices [20,21], image processing [22], modeling drift oscillations in random seas [23] and when trying to relate two signals with nonoverlapping spectral components [24]. Nonlinear filters developed using such models include order statistics filters [25,26,27] filters based on Volterra, Bilinear [28] and other polynomial descriptions of the nonlinearities involved.

Early works on adaptive Volterra filters [24], [29] were based on the LMS algorithm. The LMS algorithm is an important member of the family of stochastic gradient-based algorithms. A significant feature of the LMS algorithm is its simplicity. It does not require measurements of the pertinent correlation functions nor does it require matrix inversion. Indeed, it is the simplicity of the LMS algorithm that has made it the standart against which other adaptive filtering algorithms are benchmarked. However, it suffers from convergence behavior that is dependent on the statistics of the input signal.

One approach to achieve a convergence behavior that is independent on the statistics of the input signal is to use lattice (or other orthogonolized) structures in place of the LMS adaptive filter. The gradient adaptive lattice (GAL) algorithm is formulated around a lattice structure. Also as the name implies, its derivation is motivated by that of the LMS algorithm. Özgünel, Kayran and Panayırcı [30],[31] applied the GAL second-order Volterra filter to the channel equalization and identification problems. The GAL Volterra filter in these applications converges faster than the LMS filter, but in channel identification application, it was not run with the same structure and the same number of coefficients as that of the channel to be identified.

Another alternative is to use recursive least squares (RLS) algorithms which are implementations of the stochastic Gauss-Newton method. While LMS and gradient lattice adaptation algorithms, based on the steepest-descent method, provide a gradual iterative minimization of the performance index, RLS algorithms are based on the exact minimization of least-squares criteria. Accordingly, while the adaptive coefficients in LMS and GAL algorithms are optimal only after convergence, in RLS algorithms they are optimal at each time instant. An important feature of RLS algorithms is that they utilize information contained in the input data, extending back to the instant of time when the algorithm is initiated. The resulting rate of convergence is therefore much faster than the LMS algorithm. This improvement in performance, however, is achieved at the expense of a large increase in computational complexity. The recent fast reformulations of RLS algorithms, Lee and Mathews [32] and, Syed and Mathews [33], which are based on transversal and lattice structures respectively, reduced the complexity from $O(N^6)$ to $O(N^3)$ where N is the length of Volterra filter. Even though they are rapidly convergent and computationally simpler, the fast RLS filter suffers from poor numerical properties, and the lattice RLS filter can not compute the Volterra system coefficients efficiently and directly. The QR-decomposition based RLS algorithms for adaptive Volterra filtering has also been reported [34],[33]. The QR algorithm in [34] is based on transversal structure and it performs the QR-decomposition using a sequence of Givens rotations. The QR-based lattice algorithm in [33] transforms the conventional least squares algorithm to the QR-RLS algorithm by using the Cholesky factorization of the estimation error covariances and uses the rotation-based algorithms in [35] and multichannel LS lattice stages proposed in [36]. While QR-decomposition based RLS algorithms are rapidly convergent and inherently exhibit good numerical behaviour, both suffer from relative complexity and implementational simplicity. In addition, the QR-based lattice algorithm still needs parametric conversion for the computation of Volterra coefficients.

1.2 CONTRIBUTIONS OF THE THESIS

This thesis presents a new RLS adaptive lattice Volterra filter. The main objective in the thesis is to design a fast convergent, highly modular, and simple Volterra filter with good numerical properties.

The main contributions of the thesis are as follows :

1. The sequential processing multichannel lattice stages (SPMLS), first proposed by Ling and Proakis [37], [38] are applied to the Volterra filtering problem.

2. The different forms of lattice algorithms are adapted to the SPMLS.

3. A complete modified Gram-Schmidth orthogonalization of the input signal is achieved with SPMLS's.

4. With the complete orthogonalization, matrix inversion and vector operations in the Volterra filtering problem are avoided, only scalar operations are required and, a highly modular and simple Volterra filter with good numerical properties is obtained. Also, the filter in identification mode runs with the same structure and the same number of coefficients as that of the system to be identified and, the system coefficients are identified efficiently and directly.

5. The RLS adaptive algorithm proposed in this thesis shares the characteristic fast convergence property of all RLS algorithms with an additional improvement due to the complete lattice orthogonalization with SPMLS's.

6. The new filter is applied to the second-order Volterra channel equalization problem.

7. The new filter is applied to the identification and equalization of the satellite channels modelled with higher-order complex Volterra expansion.

8. The new filter is presented in such a way that it can be applied to adaptive echo cancellation, and adaptive noise cancellation problems in the future research.

The remainder of the thesis is organized as follows. The following chapter describes the LMS and RLS adaptive second-order Volterra filtering algorithms and introduces different forms of SPMLS algorithm. Chapter 3 presents the system identification application of the second-order Volterra filters using SPMLS's. In chapter 4, the equalization of the second-order Volterra channels with the new filter is considered. Chapter 5 presents the identification and equalization of the digital Volterra type satellite channels. Finally, the concluding remarks are made in chapter 6.

CHAPTER 2

VOLTERRA NONLINEAR FILTERING

In this chapter, the adaptive nonlinear filtering algorithms are introduced. Nonlinearity is modelled with Volterra series expansion. As the infinite series Volterra expansion is not useful in filtering applications, nonlinearity is modelled with truncated Volterra series expansion. A new lattice structure is proposed for Volterra filtering applications.

First, Volterra series expansion for nonlinear systems is introduced. Then, adaptive filtering algorithms using truncated Volterra series expansion are discussed. Least Mean Squares (LMS), Recursive Least Squares (RLS) algorithms for second-order Volterra filtering are given. Advantages and disadvantages of LMS and RLS for Volterra filtering are discussed. The advantages of lattice filters are mentioned. The Gram-Schmidth orthogonalization algorithm is reviewed. The new lattice structure, sequential processing multichannel lattice stages, for Volterra filtering is proposed. In order to develop the lattice parametrization of Volterra filters, the nonlinear filtering characterized as a linear multichannel filtering problem. The different versions of the SPMLS algorithm and a comparative complexity analysis are presented. Finally, the implementation of the modified Gram-Schmidth orthogonalization algorithm is discussed.

2.1 VOLTERRA SERIES EXPANSION FOR NONLINEAR SYSTEMS

In the Volterra series representation of systems, which is an extension of linear system theory, the output $y(n)$ of any causal, discrete-time, time invariant nonlinear system can be represented as a function of the input sequence $x(n)$

$$y(n) = h_0 + \sum_{m_1=0}^{\infty} h_1(m_1)x(n-m_1) + \sum_{m_1=0}^{\infty}\sum_{m_1=0}^{\infty} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \cdots + \sum_{m_1=0}^{\infty}\cdots\sum_{m_p=0}^{\infty} h_p(m_1,m_2,\ldots,m_p)x(n-m_1)\cdots x(n-m_p) + \cdots \quad (2.1)$$

where $h_p(m_1, m_2, \ldots, m_p)$ is the p th order Volterra kernel [39],[40] of the system. Without any loss of generality, one can assume that the Volterra kernels are symmetric, i.e., $h_p(m_1, m_2, \ldots, m_p)$ is left unchanged for any of the $p!$ permutations of the indices $m_1, m_2, \ldots, m_p$. One can think of the Volterra series expansion as a Taylor series expansion with memory.

Since an infinite series expansion like (2.1) is not useful in filtering applications, one must work with truncated Volterra series expansions of the form

$$y(n) = h_0 + \sum_{m_1=0}^{N-1} h_1(m_1)x(n-m_1) + \sum_{m_1=0}^{N-1}\sum_{m_1=0}^{N-1} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \cdots + \sum_{m_1=0}^{N-1}\cdots\sum_{m_p=0}^{N-1} h_p(m_1,m_2,\ldots,m_p)x(n-m_1)\cdots x(n-m_p) + \cdots \quad (2.2)$$

Block diagram of a second-order truncated Volterra system with $N = 3$ and h_0 is assumed zero, is shown in Figure 2.1. Note that this system is linear in the input signal to each coefficient. This fact highly simplifies the design problems involving Volterra series representations. On the other hand, even for moderately large values of N and p, the number of coefficients becomes very large. Consequently, the truncated Volterra series representation is most useful in applications where the values of N and p are relatively small.

2.2 ADAPTIVE FILTERS USING TRUNCATED VOLTERRA SERIES EXPANSIONS

The operation of an adaptive Volterra filter is descriptive of a feedback control system. Basically, it consists of a combination of two basic processes. An adaptive process, which involves the automatic adjustment of a set of filter coefficients. A filtering process, which involves forming the inner product of a set of filter coefficients emerging from the adaptive process to produce an estimate of a desired response, and generating an estimation error by comparing this estimate with the actual value of the desired response. Correspondingly, one may identify two basic components in the structural constitution of the adaptive filter as illustrated in Figure 2.2. First, there is a filter around which the adaptive algorithm is built. This component is responsible for performing the filtering process. Second, there is a mechanism for performing the adaptive control process on the coefficients of the filter. During the filtering process, the desired

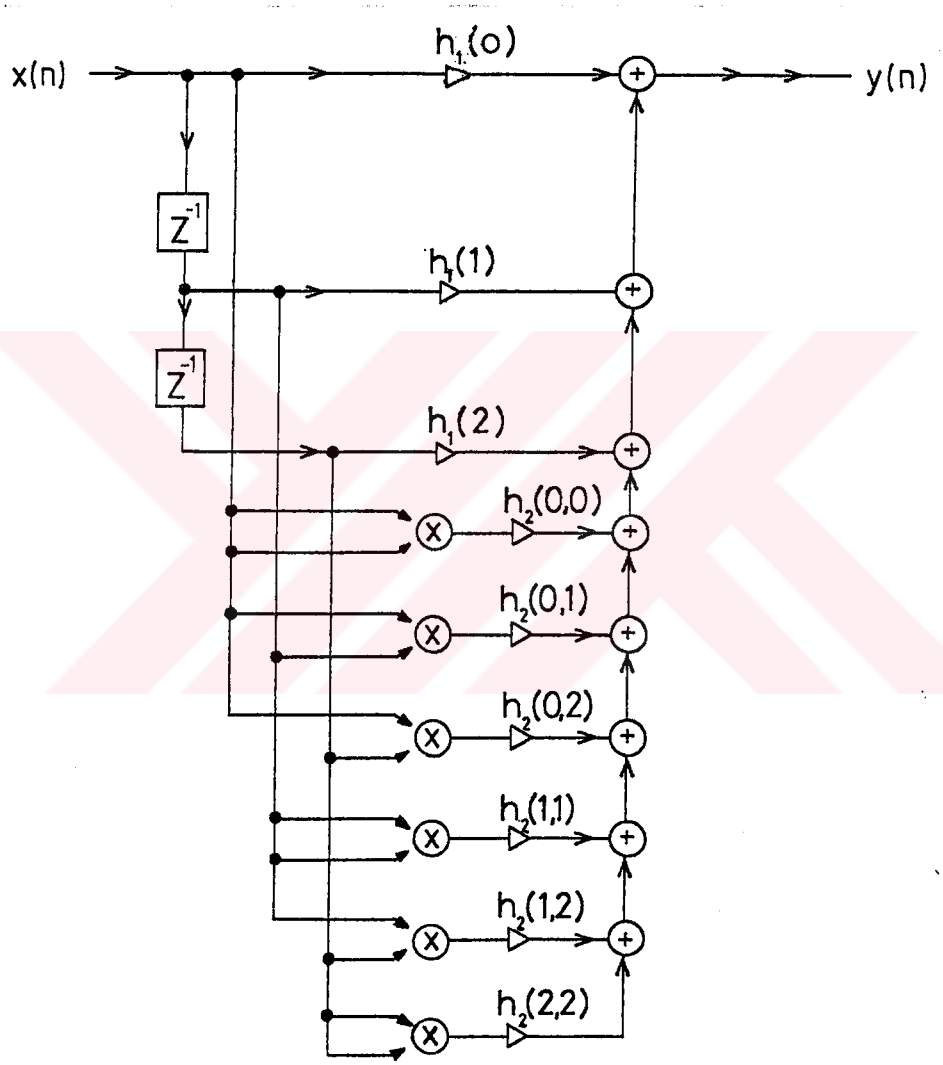


Figure 2.1: Block diagram of a second-order truncated Volterra system with $N = 3$

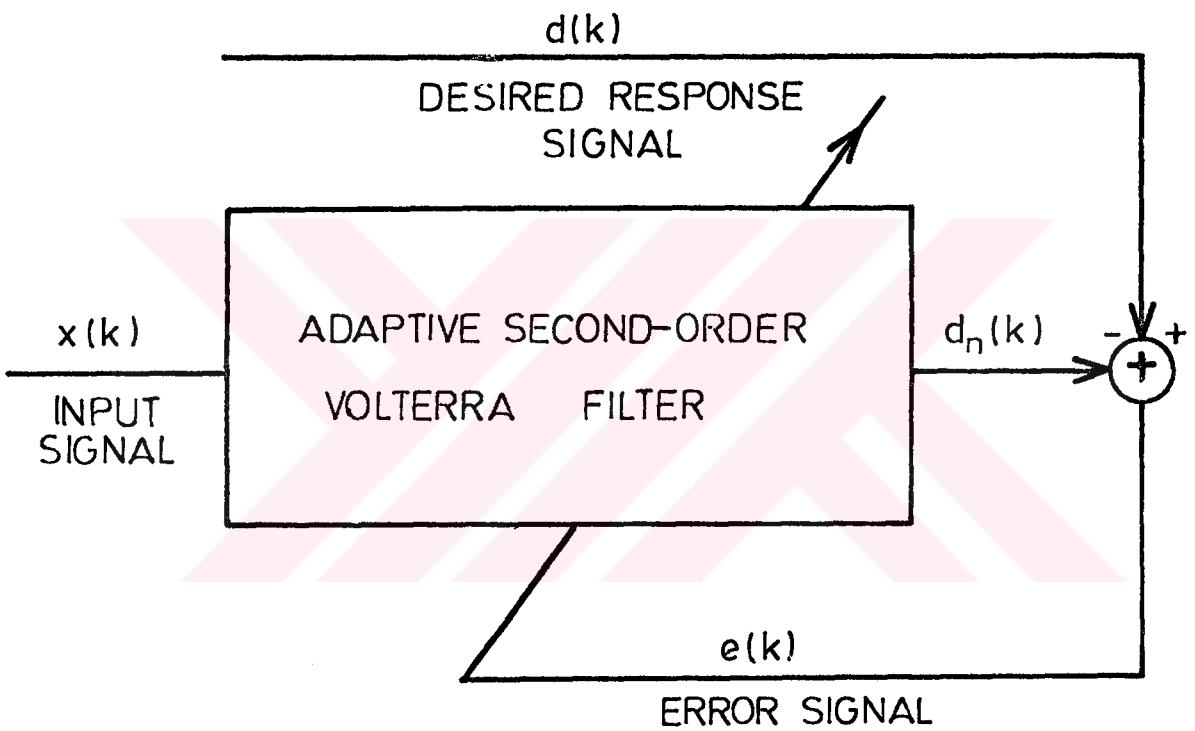


Figure 2.2: Block diagram of a general adaptive filter

signal $d(k)$ is supplied for processing alonside the input signal. Given this input, the filter produces an output $d_n(k)$ used as an estimate of the desired signal $d(k)$.

For simplicity, attention in this chapter will be focused on second-order Volterra filtering. The adaptive filter in this case would try to estimate the desired response signal $d(k)$ using a second-order truncated Volterra series expansion in the input signal $x(n)$ as

$$d_n(k) = h_0 + \sum_{i=0}^{N-1} \hat{a}_i(n)x(k-i) + \sum_{i=0}^{N-1}\sum_{j=i}^{N-1} \hat{b}_{i,j}(n)x(k-i)x(k-j) \qquad (2.3)$$

$\{\hat{a}_i\}$ and $\{\hat{b}_{i,j}\}$ in (2.3) are the adaptive filter coefficients that are iteratively updated at each time so as to minimize some convex function of the error signal defined as

$$e(k) = d(k) - d_n(k) \qquad (2.4)$$

What makes the derivation of adaptive Volterra filters relatively straightforward is the fact that the error signal can be written as a linear combination of the input signal to each filter coefficient. In the case of the second-order Volterra filter, the relevant signals are $x(n), x(n-1), \ldots, x(n-N+1), x^2(n), x(n)x(n-1), \ldots, x(n)x(n-N+1), \ldots, x^2(n-N+1)$. This fact also makes the theoretical performance analysis of such filters a relatively straightforward extension of the linear filtering case.

2.3 THE LMS ADAPTIVE FILTER

The LMS adaptive filter updates the coefficients at each time using a steepest descent algorithm that tries to minimize $e^2(k)$ at each time. The update equations for the second-order Volterra filter can be easily shown to be [41],[42]

$$\begin{aligned} \hat{a}_i(k+1) &= \hat{a}_i(k) - \frac{\mu_1}{2}\frac{\partial e^2(k)}{\partial \hat{a}_i(k)} \\ \hat{a}_i(k+1) &= \hat{a}_i(k) + \mu_1 e(k)x(k-i) \end{aligned} \qquad (2.5)$$

and

$$\begin{aligned} \hat{b}_{i,j}(k+1) &= \hat{b}_{i,j}(k) - \frac{\mu_2}{2}\frac{\partial e^2(k)}{\partial \hat{b}_{i,j}(k)} \\ \hat{b}_{i,j}(k+1) &= \hat{b}_{i,j}(k) + \mu_2 e(k)x(k-i)x(k-j) \end{aligned} \qquad (2.6)$$

where μ_1 and μ_2 are small positive constants that control the speed of convergence and the steady state/tracking properties of the filter. For notational simplicity as well

as ease of performance analysis, it is usual to rewrite the adaptive filtering algorithm using vector notations. The relevant equations are shown in Table 2.1. Note that the structure of the adaptive filter is different from that of the linear case only in the way in which the vectors are defined.

The mean values of the adaptive filter coefficients converge (for stationary environments) to their optimal values if the convergence constant is chosen such that $0 < \mu_1, \mu_2 < 2/\lambda_{max}$, where λ_{max} is the maximum eigenvalue of the autocorrelation matrix of the input vector $\mathbf{X}(n)$. The problem, as is for the linear case, is that the eigenvalues of the autocorrelation matrix control the speed of convergence. In general, the larger the eigenvalue spread (the ratio of the maximum and minimum eigenvalues), the slower is the convergence speed. This is particularly troublesome in the nonlinear filtering case, since the eigenvalue spreads are in general very large. Even when the input signal is white, the presence of the nonlinear entries in the input vector will cause the eigenvalue spread to be more than one [42]. Consequently, it is important to seek alternate algorithms and structures that have convergence behaviors that are independent of or less dependent on the statistics of the input signal.

One approach is to use gradient adaptive lattice (GAL) filters. They are based on a LMS-like approach applied to the coeffients of the lattice representations rather than to the coefficients of the direct form representation. Taking advantage of the decoupling property of the lattice structures, and properly choosing the adaptation constants, all lattice coefficients can be made to converge fast and, in constrast to the LMS coefficients, with a convergence rate that is essentially independent of the eigenvalue spread of the autocorrelation input matrix.

Another alternative is to use RLS algorithms in place of the LMS adaptive filter. The LMS adaptive filter can be considered as an approximate solution to the statistical optimization problem that tries to minimize the mean squared value of the estimation error at each time. RLS adaptive filters, on the other hand, yield the exact solution to an optimization problem formulated in a deterministic fashion.

2.4 THE RLS ADAPTIVE FILTER

One such formulation gives rise to the exponentially weighted RLS adaptive filter and in the case of the second order Volterra filter, such adaptive systems minimize the

Table 2.1: THE LMS SECOND-ORDER VOLTERRA FILTER

Coefficient Vector

$\mathbf{W}(n) = [\hat{a}_0(n), \hat{a}_1(n), \ldots, \hat{a}_{N-1}(n), \hat{b}_{0,0}(n), \hat{b}_{0,1}(n),$
$\ldots, \hat{b}_{0,N-1}(n), \hat{b}_{1,1}(n), \ldots, \hat{b}_{N-1,N-1}(n)]^T$

$\mathbf{X}(n) = [x(n), x(n-1), \ldots, x(n-N+1), x^2(n), x(n)x(n-1), \ldots,$
$x(n)x(n-N+1), x^2(n-1), \ldots, x^2(n-N+1)]^T$

Initialization

$\mathbf{W}(0)$ can be arbitrarily chosen.

Algorithm

$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n)$

$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu\mathbf{X}(n)e(n)$

Note: $()^T$ denotes matrix transpose. μ is a diagonal matrix with μ_1 appearing in the first N diagonal entries and μ_2 appearing in the rest of the diagonal entries.

following cost function at each time

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - \mathbf{W}^T(n)\mathbf{X}(k) \mid^2 \tag{2.7}$$

where $\mathbf{W}(n)$ and $\mathbf{X}(n)$ are the coefficient and input signal vectors, which are defined as

$$\mathbf{W}(n) = [\hat{a}_0(n), \hat{a}_1(n), \ldots, \hat{a}_{N-1}(n), \hat{b}_{0,0}(n), \hat{b}_{0,1}(n), \ldots, \hat{b}_{0,N-1}(n), \hat{b}_{1,1}(n), \ldots, \hat{b}_{N-1,N-1}(n)]^T \tag{2.8}$$

and

$$\begin{aligned}\mathbf{X}(n) = [&x(n), x(n-1), \ldots, x(n-N+1), x^2(n), x(n)x(n-1), \\ &\ldots, x(n)x(n-N+1), x^2(n-1), \ldots, x^2(n-N+1)]^T\end{aligned} \tag{2.9}$$

respectively. λ $(0 < \lambda \leq 1)$ is the parameter of the exponential window that controls the rate at which the adaptive filter tracks the time varying parameters [43,44]. The solution to this problem at each time can be easily found by differentiating $J(n)$ with respect to $\mathbf{W}(n)$, setting the derivative to zero, and solving for $\mathbf{W}(n)$. The optimal solution at time n is given by

$$\mathbf{W}_{opt}(n) = \mathbf{R}^{-1}(n)\,\mathbf{P}(n) \tag{2.10}$$

where

$$\mathbf{R}(n) = \sum_{k=0}^{n} \lambda^{n-k} \mathbf{X}(k)\, \mathbf{X}(k)^T \tag{2.11}$$

and

$$\mathbf{P}(n) = \sum_{k=0}^{n} \lambda^{n-k} d(k) \mathbf{X}(k). \tag{2.12}$$

Here, $\mathbf{R}(n)$ is the LS autocorrelation matrix of the input vector $\mathbf{X}(n)$ and $\mathbf{P}(n)$ is the LS cross-correlation vector between the input vector $\mathbf{X}(n)$ and the desired signal $d(n)$.

In order to derive a recursion formula for updating the coefficient vector $\mathbf{W}(n)$, $\mathbf{R}(n)$ and $\mathbf{P}(n)$ is updated as

$$\mathbf{R}(n) = \lambda \mathbf{R}(n-1) + \mathbf{X}(n)\mathbf{X}^T(n) \tag{2.13}$$

and

$$\mathbf{P}(n) = \lambda \mathbf{P}(n-1) + d(n)\mathbf{X}(n) \tag{2.14}$$

where $\mathbf{R}(n-1)$ and $\mathbf{P}(n-1)$ are the "old" values of the correlation matrix and cross-correlation vector respectively and, the vector products $\mathbf{X}(n)\mathbf{X}^T(n)$ and $d(n)\mathbf{X}(n)$ play the role of a correction term in the updating operation. To avoid the direct matrix inversion in (2.10), which is a very time consuming task, a basic result in matrix algebra known as the matrix inversion lemma [41] is used. According to the matrix inversion lemma, the inverse of a positive definite $M \times M$ matrix $\mathbf{A}$ can be expressed as follows:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{BC}(\mathbf{D} + \mathbf{C}^{\mathbf{H}}\mathbf{BC})^{-1}\mathbf{C}^{\mathbf{H}}\mathbf{B} \tag{2.15}$$

where $\mathbf{B}$ is a positive definite $M \times M$ matrix, $\mathbf{D}$ is another positive definite $N \times N$ matrix, and $\mathbf{C}$ is an $M \times N$ matrix. Here, The superscript "H" denotes Hermitan transpose of a matrix. These matrices are related by

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{CD}^{-1}\mathbf{C}^{\mathbf{H}}. \tag{2.16}$$

Thus, $\mathbf{R}^{-1}(n-1)$ is updated as,

$$\mathbf{R}^{-1}(n) = \lambda^{-1}\mathbf{R}^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{R}^{-1}(n-1)\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{R}^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{X}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{X}(n)}. \tag{2.17}$$

and a new vector, which is called the time-varying gain vector, is defined as

$$\mathbf{k}(n) = \mathbf{R}^{-1}(n)\mathbf{X}(n). \tag{2.18}$$

Using (2.17) in (2.18), the time-varying gain vector $\mathbf{k}(n)$ is expressed as,

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{R}^{-1}(n-1)\mathbf{X}(n)}{1 + \lambda^{-1}\mathbf{X}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{X}(n)}. \tag{2.19}$$

Substituting (2.17) in (2.10), then using (2.19) for the time-varing gain vector, the desired recursive expression for updating the coefficient vector is,

$$\begin{aligned} \mathbf{W}(n) &= \mathbf{W}(n-1) + \mathbf{k}(n)[d(n) - \mathbf{X}^{\mathbf{T}}(n)\mathbf{W}(n-1)] \\ &= \mathbf{W}(n-1) + \mathbf{k}(n)\varepsilon(n) \end{aligned} \tag{2.20}$$

where $\varepsilon(n)$ is defined by

$$\varepsilon(n) = d(n) - \mathbf{W}^{T}(n-1)\mathbf{X}(n). \tag{2.21}$$

The inner product $\mathbf{W}^{\mathbf{T}}(n-1)\mathbf{X}(n)$ represents an estimate of the desired signal $d(n)$, based on the previous least-squares estimate of the coefficient vector that was made at time $n-1$. Accordingly, $\varepsilon(n)$ is referred to as the a priori estimation error. The algorithm has been summarized in Table 2.2 .

While direct evaluation of (2.10) requires $O(N^6)$ multiplications at each instant, using the matrix inversion lemma, this complexity is reduced to $O(N^4)$ multiplications per iteration [32]. Another approach, that exploits the relationships among the forward predictor, the backward predictor, and the gain vector to obtain the relevant update equations [32], reduces the complexity to $O(N^3)$ multiplications per iteration. Even-though it uses the matrix inversion lemma to avoid the direct matrix inversion, it still suffers from relative complexity due to matrix operations.

Aside from computational complexity problems of RLS adaptive filters, they may suffer from numerical instability problems in finite-precision arithmetic. The numerical instability or explosive divergence of RLS adaptive filters is of similar nature to that experienced in Kalman filtering, of which the RLS filter is a special case [41]. Indeed, the problem may be traced to the fact that the time-updated matrix $\mathbf{R}^{-1}(n)$ in equation (2.17) is computed as the difference between two nonnegative definite matrices. Accordingly, explosive divergence of the algorithm occurs when the matrix $\mathbf{R}^{-1}(n)$ loses the property of positive definiteness, which implies loss of nonsingularity for the matrix $\mathbf{R}(n)$. Another form of divergence observed in the finite-precision RLS algorithm is that of stalling. Stalling phenomenon occurs when the coeffients in the RLS algorithm stop adapting. In particular, this phenomenon occurs when the quantized elements of the matrix $\mathbf{R}^{-1}(n)$ become very small, such that multiplication by $\mathbf{R}^{-1}(n)$ is equivalent to multiplication by a zero matrix.

QR-decomposition based RLS algorithms solve matrix inversion induced numerical problems by bypassing the normal equation and working directly with the data matrix.

Table 2.2: THE RLS ADAPTIVE SECOND-ORDER VOLTERRA FILTER

Coefficient Vector

$\mathbf{W}(n) = [\hat{a}_0(n), \hat{a}_1(n), \ldots, \hat{a}_{N-1}(n), \hat{b}_{0,0}(n), \hat{b}_{0,1}(n),$
$\ldots, \hat{b}_{0,N-1}(n), \hat{b}_{1,1}(n), \ldots, \hat{b}_{N-1,N-1}(n)]^T$

$\mathbf{X}(n) = [x(n), x(n-1), \ldots, x(n-N+1), x^2(n), x(n)x(n-1), \ldots,$
$x(n)x(n-N+1), x^2(n-1), \ldots, x^2(n-N+1)]^T$

Initialization

$\mathbf{W}(0) = [0, 0, \ldots, 0]^T$

$\mathbf{R}^{-1}(0) = \delta^{-1}\mathbf{I}$

δ = a small positive constant

Algorithm

$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{R}^{-1}(n-1)\mathbf{X}(n)}{1+\lambda^{-1}\mathbf{X}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{X}(n)}$

$\varepsilon(n) = d(n) - \mathbf{W}^T(n-1)\mathbf{X}(n)$

$\mathbf{W}(n) = \mathbf{W}(n-1) + \mathbf{k}(n)\varepsilon(n)$

$\mathbf{R}^{-1}(n-1) = \lambda^{-1}\mathbf{R}^{-1}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{X}^T(n)\mathbf{R}^{-1}(n-1)$

$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n)$

This approach is employed because the eigenvalue spread of the data correlation matrix is the square of the eigenvalue spread of the corresponding data matrix [36]. Hence, a least squares solution based directly on the data matrix is better conditioned than one based on the data correlation matrix.

In general, one would like to develop computationally simple, numerically stable and modular algorithms such that matrix inversion and vector operations are avoided. One such algorithm that is based on a lattice structure and is related to QR-decomposition based RLS algorithms is proposed for Volterra filtering in the next section.

2.5 ADAPTIVE LATTICE FILTERS

Adaptive lattice filters implements the Gram-Schmidth orthogonalization algorithm for transforming the input vector consisting of correlated samples into an equivalent vector consisting of uncorrelated samples and, estimate the desired signal as a linear combination of the transformed signals that are orthogonal to each other. Lattice

filters, equipped with LMS type adaptation algorithms, which are called GAL filters, tend to show faster and less input signal dependent convergence behavior than their direct form counterparts. They tend to have better numerical properties than direct form adaptive filters. Least squares (LS) lattice filters on the other hand, converge even faster due to their exact decoupling property. In addition to their fast convergence and good numerical properties, lattice filters in general are fairly modular and suitable for VLSI implementations. It is the main purpose of this thesis to develop lattice structures for Volterra filtering, equipped with recursive least squares algorithms, so as to obtain the best of the computational efficiency of the former and the fast convergence property of the latter. Before proposing a new lattice structure for Volterra filtering, the Gram-Schmidth orthogonalization algorithm is first reviewed.

2.5.1 THE GRAM-SCHMIDTH ORTHOGONALIZATION ALGORITHM

In the Gram-Schmidth algorithm, the objective is to transform a sequence of correlated random variables denoted by $x(n), x(n-1), \ldots, x(n-M)$ into a new sequence, $b_0(n), b_1(n), \ldots, b_M(n)$, of random variables uncorrelated with each other. To accomplish the transformation, following series of steps are taken :

1.Let

$$b_0(n) = x(n) \tag{2.22}$$

2. $b_1(n)$ is expressed as a linear combination of $x(n)$ and $x(n-1)$ as

$$b_1(n) = x(n-1) + k_1(1)x(n) \tag{2.23}$$

where $k_1(1)$ is a constant to be determined. The cross-correlation of $b_0(n)$ and $b_1(n)$ equals

$$E[b_0(n)b_1(n)] = E[x(n)x(n-1)] + k_1(1)E[x^2(n)] \tag{2.24}$$

where $E[\bullet]$ is the expectation operator. For $b_0(n)$ to be orthogonal to $b_1(n)$, it is required that

$$E[b_0(n)b_1(n)] = 0. \tag{2.25}$$

This requirement is satisfied by choosing

$$k_1(1) = -\frac{E[x(n)x(n-1)]}{E[x^2(n)]}. \tag{2.26}$$

3. $b_2(n)$ is expressed as a linear combination of $x(n), x(n-1)$, and $x(n-2)$ as shown by

$$b_2(n) = x(n-2) + k_2(1)x(n-1) + k_2(2)x(n). \tag{2.27}$$

For $b_2(n)$ to be orthogonal to $b_0(n)$ and $b_1(n)$, it is required that

$$E[b_0(n)b_2(n)] = 0 \quad \text{and} \quad E[b_1(n)b_2(n)] = 0. \tag{2.28}$$

Solving for the constants $k_2(2)$ and $k_2(1)$,

$$k_2(2) = \frac{E^2[x(n)x(n-1)] - E[x(n)x(n)]E[x(n)x(n-2)]}{E^2[x(n)x(n)] - E^2[x(n)x(n-1)]} \tag{2.29}$$

$$k_2(1) = \frac{E[x(n)x(n-1)]E[x(n)x(n-2)] - E[x(n)x(n)]E[x(n)x(n-1)]}{E^2[x(n)x(n)] - E^2[x(n)x(n-1)]}. \tag{2.30}$$

4. Continue in this manner until the last random variable of interest, $b_M(n)$, has been accounted for. That is, $b_M(n)$ is expressed as a linear combination of $x(n), x(n-1), \ldots, x(n-M)$ as shown by

$$b_M(n) = k_M(M)x(n) + k_M(M-1)x(n-1) + \ldots + x(n-M). \tag{2.31}$$

Thus, the Gram-Schmidth orthogonalization procedure is completed with the generation of a set of random variables $b_0(n), b_1(n), \ldots, b_M(n)$ that are orthogonal to each other. The series of steps involved in the construction of the Gram-Schmidth algorithm may be summarized in matrix formulation as follows:

$$\begin{bmatrix} b_0(n) \\ b_1(n) \\ \vdots \\ b_M(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ k_1(1) & 1 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ k_M(M) & k_M(M-1) & \ldots & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-M) \end{bmatrix}. \tag{2.32}$$

Accordingly, the equation (2.32) can be rewritten simply as

$$\mathbf{b}(n) = \mathbf{L}\mathbf{x}(n) \tag{2.33}$$

where $\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-M)]^T$, $\mathbf{b}(n) = [b_0(n), b_1(n), \ldots, b_M(n)]^T$ and

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ k_1(1) & 1 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ k_M(M) & k_M(M-1) & \ldots & 1 \end{bmatrix}.$$

This equation states that the input vector $\mathbf{x}(n)$ is transformed into the output vector $\mathbf{b}(n)$ via multiplication by the unit lower triangular matrix $\mathbf{L}$. The construction of unit lower triangular matrix $\mathbf{L}$ in the Gram-Schmidth algorithm is built up row-wise. On the other hand, it is built up coloumn-wise in the modified Gram-Schmidth algorithm.

Let $\mathbf{R}_x$ denote the $(M+1) \times (M+1)$ correlation matrix of the $(M+1) \times 1$ input vector $\mathbf{x}(n)$. Let $\mathbf{R}_b$ denote the corresponding $(M+1) \times (M+1)$ correlation matrix of the $(M+1) \times 1$ output vector $\mathbf{b}(n)$ produced by the Gram-Schmidth algorithm. Accordingly, $\mathbf{R}_b$ can be written as

$$\begin{aligned} \mathbf{R}_b &= E[\mathbf{b}(n)\mathbf{b}(n)] \\ &= E[\mathbf{L}\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{L}^T] \\ &= \mathbf{L}E[\mathbf{x}(n)\mathbf{x}^T(n)]\mathbf{L}^T \\ &= \mathbf{L}\mathbf{R}_\mathbf{X}\mathbf{L}^T \end{aligned} \tag{2.34}$$

Since the output vector $\mathbf{b}(n)$ consists of uncorrelated random variables, the correlation matrix $\mathbf{R}_b$ is a diagonal matrix whose elements equal the mean-square values of the respective random variables that constitute the vector $\mathbf{b}(n)$. That is, $\mathbf{R}_b$ can be expressed as

$$\mathbf{R}_b = diag(D_0, D_1, \ldots, D_M) \tag{2.35}$$

where

$$D_i = E[b_i^2(n)], \quad i = 0, 1, \ldots, M. \tag{2.36}$$

Then, the inverse of the correlation matrix $\mathbf{R}_b$ becomes

$$\mathbf{R}_b^{-1} = diag(D_0^{-1}, D_1^{-1}, \ldots, D_M^{-1}). \tag{2.37}$$

Thus, with the implementation of the (modified) Gram-Schmidth orthogonalization algorithm, the matrix inversion operation is transformed to a simple scalar operation. Hence, numerical problems associated with matrix inversion are avoided and the complexity is reduced. In addition, regular, modular and simple structures such as lattice structures, which are suitable for VLSI implementations become available for matrix inversion problem.

2.5.2 ADAPTIVE VOLTERRA FILTERING WITH SPMLS

The objective in this section is to propose a new multichannel lattice structure for Volterra filtering which implements the modified Gram-Schmidth orthogonalization algorithm. To implement the modified Gram-Schmidth algorithm, the nonlinear filtering problem is first transformed into an equivalent multichannel, but linear, adaptive filtering problem [33]. The basic idea is to partition the input vector $\mathbf{X}(n)$ at time n into the following set of smaller $(N+1)$ vectors so that each of the vectors can be considered as being formed from successive samples of signals from a different input channel.

$$
\begin{aligned}
CH1: &\quad [x(n), x(n-1), \ldots, x(n-N+1)] \\
CH2: &\quad [x(n)x(n), x(n-1)x(n-1), \ldots, x(n-N+1)x(n-N+1)] \\
CH3: &\quad [x(n)x(n-1), x(n-1)x(n-2), \ldots, x(n-N+2)x(n-N+1)] \\
CH4: &\quad [x(n)x(n-2), x(n-1)x(n-3), \ldots, x(n-N+3)x(n-N+1)] \\
&\vdots \\
CH(N+1): &\quad [x(n)x(n-N+1)].
\end{aligned} \tag{2.38}
$$

The most important manner in which this multichannel characterization differs from traditional multichannel, adaptive linear filters is that the number of coefficients associated with each channel varies from channel-to-channel [37]. Channel 1 and 2 have N coefficients associated with it while channel 3 has $N-1$, channel 4 has $N-2$ and finally channel $N+1$ has only single coefficient associated with it.

After transforming the Volterra filtering problem into the equivalent multichannel filtering problem, a complete, modified Gram-Schmidth orthogonalization of the multichannel input vector is achieved using sequential processing multichannel lattice stages.

A sequential processing multichannel lattice stage, first proposed by Ling and Proakis [37], has a block structure as shown in Figure 2.3. It consists of two self-orthogonal processors (SOP) and two reference-orthogonal processors (ROP). While ROP's consist of orthogonal processing cells, which are represented as circles, SOP's in addition contain processing cells, which are represented as double circles. Input and output signals for cells are shown in Figure 2.4. Each processing cell represented as double circles has two input signals $s_r(n)$, $\gamma_i(n)$ and, two output signals $r(n)$ and $\gamma_o(n)$. The input signals $s_r(n)$ and $\gamma_i(n)$ are retransmitted. Each orthogonal processing cell represented as circles on the other hand, has one main input signal $s_i(n)$, three reference input signals $r(n)$, $s_r(n)$, $\gamma_i(n)$ and one output signal $s_o(n)$. The reference input signals are also retransmitted.

At each orthogonal processing cell represented as circles, the exponentially weighted LS cross-correlation, $\Delta(n)$, between the input signals $s_i(n)$ and $s_r(n)$ at time n is computed and the correlation associated with the input signal $s_r(n)$ is removed from the other input signal $s_i(n)$. Thus, the orthogonality between the output signal $s_o(n)$ and the input signal $s_r(n)$ is achieved. To accomplish the correlation cancelling operation

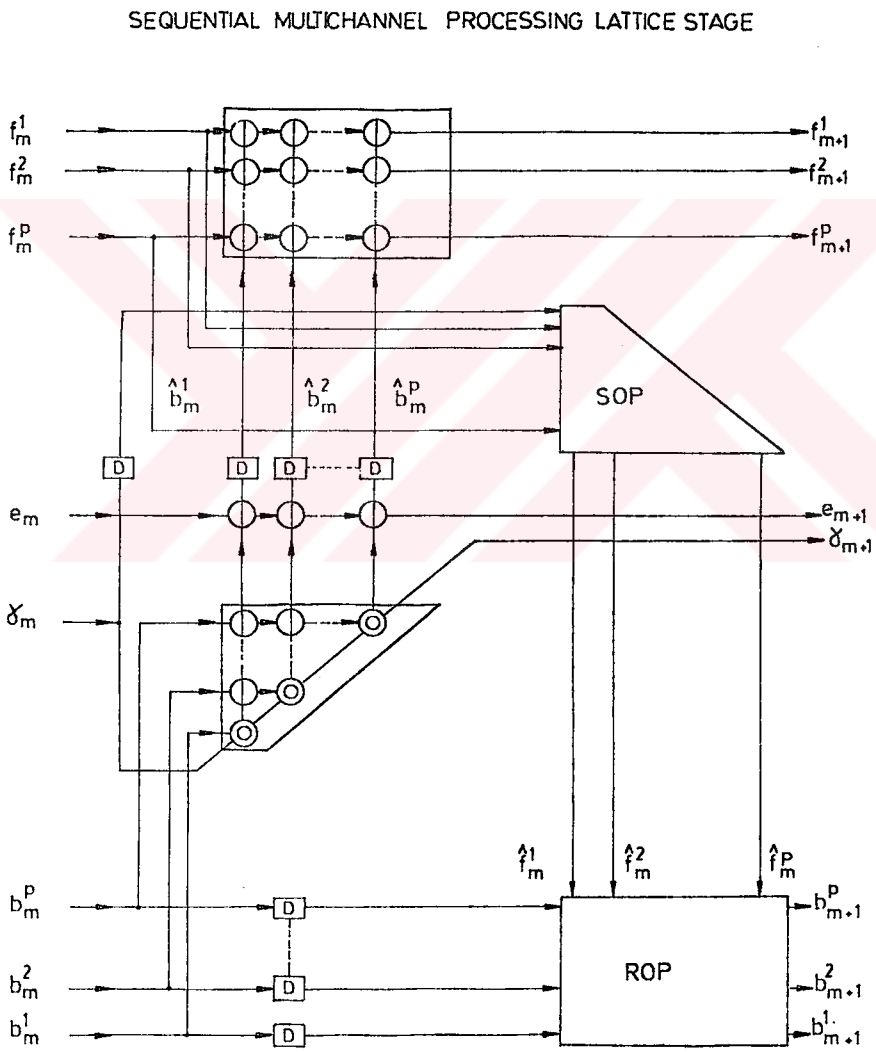


Figure 2.3: Block diagram of a sequential processing multichannel lattice stage, first proposed by Ling and Proakis [37]

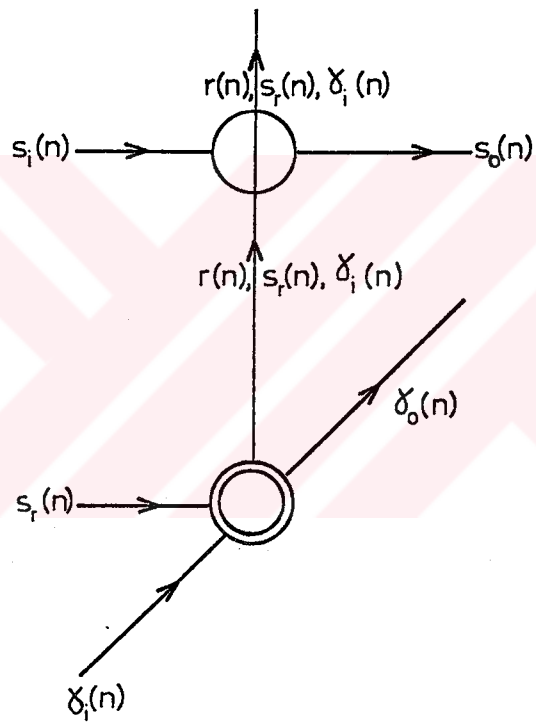


Figure 2.4: Block diagram of the sequential processing multichannel lattice stage cells

at each orthogonal processing cell represented as circles, the exponentially weighted LS autocorrelation, $r(n)$, of the input signal $s_r(n)$ and, the likelihood variable $\gamma_o(n)$ are computed at each orthogonal processing cell represented as double circles. The computations that are taking part in cells can be considered as the least squares equivalent of the Gram-Schmidth orthogonalization steps in the previous section.

The exponentially weighted LS cross-correlation $\Delta(n)$ and the exponentially weighted autocorrelation $r(n)$ are defined as

$$\Delta(n) = \sum_{i=1}^{n} \lambda^{n-i} s_r(n) s_i(n) \tag{2.39}$$

and

$$r(n) = \sum_{i=1}^{n} \lambda^{n-i} s_r^2(n) \tag{2.40}$$

respectively. The likelihood variable $\gamma_o(n)$ is defined as

$$\gamma_o(n) = \gamma_i(n) - s_r^2(n)/r(n) \tag{2.41}$$

where $\gamma_i(n)$ is the input likelihood variable and, the input likelihood variable to the first orthogonal processing cell represented as double circles at the first SPMLS is given by

$$\gamma_i(i) = \begin{cases} 1, & i = n \\ 0, & i = 1, 2, \ldots, n-1 \end{cases} \tag{2.42}$$

which is called the first unit vector. Thus, the likelihood variable can be interpretated as the estimation error signal for the generation of the LS estimate of the first unit vector and, acts as a gain factor which facilitates fast tracking of changes in the statistics of the observed data.

Processing equations for cells depend on what version of SPMLS algorithm is used. Mainly, there are four versions of the SPMLS algorithm [41],[45]. In version I, the variables are a posteriori forms of prediction and estimation errors, and the reflection coefficients are computed indirectly. In version II, the variables are a priori forms of prediction and estimation errors, and the reflection coefficients are again computed indirectly. In version III, the variables are a priori forms of prediction and estimation errors, but the reflection coefficients are computed directly. As a result of this direct computation, error feedback is introduced into the operation of the algorithm. In version IV, the variables are a posteriori forms of prediction and estimation errors, but the reflection coefficients are computed directly. Here again, error feedback is

introduced into the operation of the algorithm. In a posteriori forms, the output signals for cells represented as circles are computed using the reflection coefficients at the present time instant, n. In a priori forms, the output signals are computed using the reflection coefficients at the previous time instant, $n-1$. In indirect forms, cross-correlation between the input signals are first computed, then reflection coefficients are computed. On the other hand in direct forms, reflection coefficents are computed without computing cross-correlation between the input signals. In all forms, recursive formulations are used.

In version I, a posteriori, indirect form, the processing equations for cells represented as double circles are,

$$r(n) = \lambda\, r(n-1) + s_r^2(n)/\gamma_i(n) \tag{2.43}$$

$$\gamma_o(n) = \gamma_i(n) - s_r^2(n)/r(n) \tag{2.44}$$

where $s_r(n)$ is the input signal to the cell, r(n) is the exponentially weighted LS autocorrelation of the input signal, $\gamma_i(n)$ is the input likelihood variable, λ is the exponential weighting factor and $\gamma_o(n)$ is the output likelihood variable, all at time n [37]. Processing equations for cells represented as circles are,

$$\Delta(n) = \lambda\, \Delta(n-1) + s_i(n)s_r(n)/\gamma_i(n) \tag{2.45}$$

$$k(n) = \Delta(n)/r(n) \tag{2.46}$$

$$s_o(n) = s_i(n) - k(n)s_r(n) \tag{2.47}$$

where $s_i(n)$ and $s_r(n)$ are the input signal and the reference input signal to the cell respectively, $\Delta(n)$ is the exponentially weighted LS cross-correlation between the input signals, $s_o(n)$ is the output signal for the cell, and $k(n)$ is the reflection coefficient all at time n.

In version II, the a priori, indirect form, processing equations for cells represented as double circles are,

$$r(n) = \lambda\, r(n-1) + \gamma_i(n)s_r^2(n) \tag{2.48}$$

$$\gamma_o(n) = \gamma_i(n) - \gamma_i^2(n)s_r^2(n)/r(n) \tag{2.49}$$

where $s_r(n)$ is the input signal to the cell, r(n) is the exponentially weighted LS autocorrelation of the input signal, $\gamma_i(n)$ is the input likelihood variable, λ is the exponential

weighting factor and $\gamma_o(n)$ is the output likelihood variable, all at time n [37]. Processing equations for cells represented as circles are,

$$s_o(n) = s_i(n) - k(n-1)s_r(n) \tag{2.50}$$

$$\Delta(n) = \lambda\Delta(n-1) + \gamma_i(n)s_i(n)s_r(n) \tag{2.51}$$

$$k(n) = \Delta(n)/r(n) \tag{2.52}$$

where $s_i(n)$ and $s_r(n)$ are the input signal and the reference input signal to the cell respectively, $\Delta(n)$ is the exponentially weighted LS cross-correlation between the input signals, $s_o(n)$ is the output signal for the cell and, $k(n)$ is the reflection coefficient all at time n.

In version III, the a priori, direct form, processing equations for cells represented as double circles are,

$$r(n) = \lambda\, r(n-1) + \gamma_i(n)s_r^2(n) \tag{2.53}$$

$$\gamma_o(n) = \gamma_i(n) - \gamma_i^2(n)s_r^2(n)/r(n) \tag{2.54}$$

where $s_r(n)$ is the input signal to the cell, r(n) is the exponentially weighted LS autocorrelation of the input signal, $\gamma_i(n)$ is the input likelihood variable, λ is the exponential weighting factor and $\gamma_o(n)$ is the output likelihood variable, all at time n. Processing equations for cells represented as circles are,

$$s_o(n) = s_i(n) - k(n-1)s_r(n) \tag{2.55}$$

$$k(n) = k(n-1) + \gamma_i(n)s_o(n)s_r(n)/r(n) \tag{2.56}$$

where $s_i(n)$ and $s_r(n)$ are the input signal and the reference input signal to the cell respectively, $k(n)$ is the reflection coefficient, $s_o(n)$ is the output signal for the cell, all at time n.

In version IV, the a posteriori, direct form, processing equations for cells represented as double circles are,

$$r(n) = \lambda\, r(n-1) + s_r^2(n)/\gamma_i(n) \tag{2.57}$$

$$\gamma_o(n) = \gamma_i(n) - s_r^2(n)/r(n) \tag{2.58}$$

where $s_r(n)$ is the input signal to the cell, r(n) is the exponentially weighted LS autocorrelation of the input signal, $\gamma_i(n)$ is the input likelihood variable, λ is the exponential

weighting factor and $\gamma_o(n)$ is the output likelihood variable, all at time n. Processing equations for cells represented as circles are,

$$k(n) = k(n-1) - \frac{s_r(n)}{r(n)\gamma_i(n)}(s_i(n) + s_r(n)k(n-1)) \tag{2.59}$$

$$s_o(n) = s_i(n) - k(n)s_r(n) \tag{2.60}$$

where $s_i(n)$ and $s_r(n)$ are the input signal and the reference input signal to the cell respectively, $k(n)$ is the reflection coefficient, $s_o(n)$ is the output signal for the cell, all at time n.

In theory, assuming the use of infinite precision, all four versions of the SPMLS algorithm are mathematically equivalent. However, in a practical situation involving the use of finite-precision arithmetic, the four versions behave differently [41]. In particular, versions I and II may suffer from a numerical instability problem due to finite-precision effects. On the other hand, versions III and IV offer better numerical properties due to direct updating of reflection coefficients without first computing the cross correlation and autocorrelation of errors as done in version I and II. While the a priori forms are best suited for applications such as adaptive equalization and adaptive noise cancelling, a posteriori forms are used in system identification applications. Processing equations for all four versions are summarized in Table 2.3. The computational complexity of the SPMLS algorithm depends on which version is used. A comparative complexity analysis of the SPMLS algorithm is presented in Fig. 2.5.

Sequential processing multichannel lattice stage algorithm is also related to QR-decomposition based multichannel approach in [36]. Basically, in QR-decomposition, the Q matrix is implicitly formed and then used to compute the R matrix; whereas in the modified Gram-Schmidth approach used in this thesis, the inverse of the R matrix is implicitly formed and then used to compute the Q. Also, processing equations of the SPMLS algorithm are algebraically equivalent with the Givens rotation based algorithms in [46] which proves the equivalence for the a priori error-feed back version.

In the regular implementation of multichannel adaptive lattice filters, the components of forward prediction error vectors,

$$\mathbf{f}_m(n) = [f_m^1(n), f_m^2(n), \ldots, f_m^p(n)]^T \tag{2.61}$$

and backward prediction error vectors,

$$\mathbf{b}_m(n) = [b_m^1(n), b_m^2(n), \ldots, b_m^p(n)]^T \tag{2.62}$$

Table 2.3: PROCESSING EQUATIONS FOR THE SPMLS ALGORITHM

Version I

$$r(n) = \lambda\, r(n-1) + s_r^2(n)/\gamma_i(n)$$

$$\gamma_o(n) = \gamma_i(n) - s_r^2(n)/r(n)$$

$$\Delta(n) = \lambda\, \Delta(n-1) + s_i(n)s_r(n)/\gamma_i(n)$$

$$k(n) = \Delta(n)/r(n)$$

$$s_o(n) = s_i(n) - k(n)s_r(n)$$

Version II

$$r(n) = \lambda\, r(n-1) + \gamma_i(n)s_r^2(n)$$

$$\gamma_o(n) = \gamma_i(n) - \gamma_i^2(n)s_r^2(n)/r(n)$$

$$s_o(n) = s_i(n) - k(n-1)s_r(n)$$

$$\Delta(n) = \lambda\Delta(n-1) + \gamma_i(n)s_i(n)s_r(n)$$

$$k(n) = \Delta(n)/r(n)$$

Version III

$$r(n) = \lambda\, r(n-1) + \gamma_i(n)s_r^2(n)$$

$$\gamma_o(n) = \gamma_i(n) - \gamma_i^2(n)s_r^2(n)/r(n)$$

$$s_o(n) = s_i(n) - k(n-1)s_r(n)$$

$$k(n) = k(n-1) + \gamma_i(n)s_o(n)s_r(n)/r(n)$$

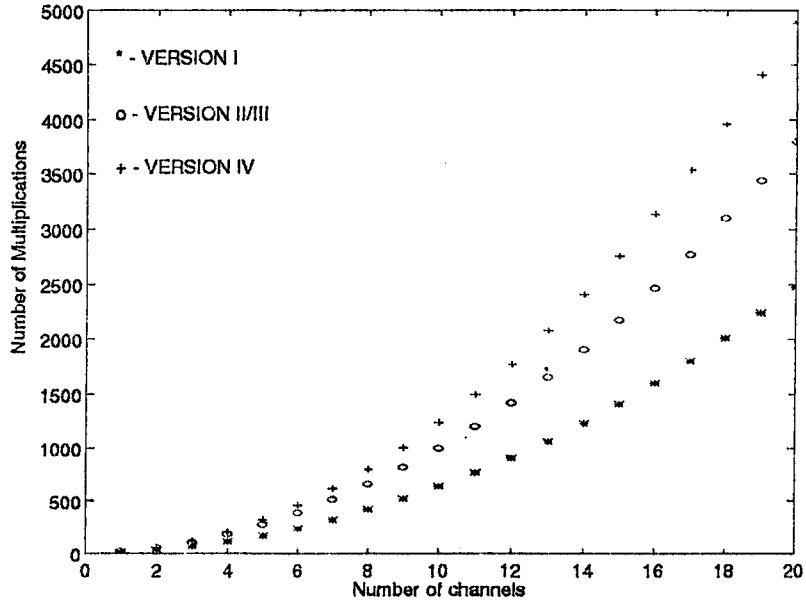
Version IV

$$r(n) = \lambda\, r(n-1) + s_r^2(n)/\gamma_i(n)$$

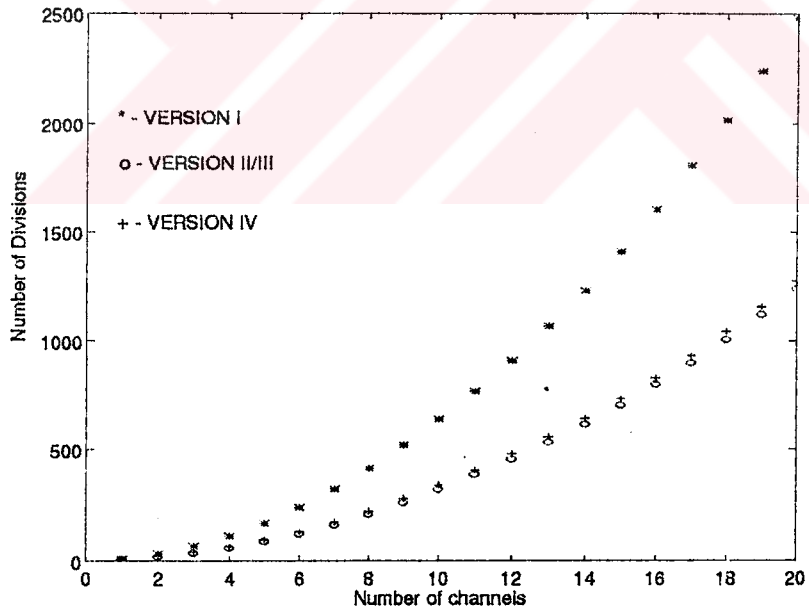
$$\gamma_o(n) = \gamma_i(n) - s_r^2(n)/r(n)$$

$$k(n) = k(n-1) - \frac{s_r(n)}{r(n)\gamma_i(n)}(s_i(n) + s_r(n)k(n-1))$$

$$s_o(n) = s_i(n) - k(n)s_r(n)$$



(a)



(b)

Figure 2.5: Complexity curves for the SPMLS algorithms

at the output of the m th stage and at time n are not orthogonal to each other. Here, p is the number of input channels. In the sequential processing lattice stage implementation, the new error vectors $\hat{\mathbf{f}}_m(n)$ and $\hat{\mathbf{b}}_m(n)$ are constructed from $\mathbf{f}_m(n)$ and $\mathbf{b}_m(n)$ at each stage as shown in Fig. 2.3. The construction of the new error vectors $\hat{\mathbf{f}}_m(n)$ and $\hat{\mathbf{b}}_m(n)$ from $\mathbf{f}_m(n)$ and $\mathbf{b}_m(n)$ is accomplished by two self-orthogonal processors at each stage. Self-orthogonal processors orthogonalize the components of the backward, $\mathbf{b}_m(n)$, and forward, $\mathbf{f}_m(n)$, error vectors, by coloumn-wise building up the unit lower triangular transformation matrices $\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)$ and $\mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)$ at each stage as

$$\hat{\mathbf{b}}_{\mathbf{m}}(n) = \mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)\mathbf{b}_{\mathbf{m}}(n) \tag{2.63}$$

and

$$\hat{\mathbf{f}}_{\mathbf{m}}(n) = \mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)\mathbf{f}_{\mathbf{m}}(n) \tag{2.64}$$

where

$$\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n) = \begin{bmatrix} 1 & 0 & \dots & 0 \\ k_{m,1}^{b}(n) & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ k_{m,p}^{b}(n) & k_{m,p-1}^{b}(n) & \dots & 1 \end{bmatrix} \text{ and } \mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n) = \begin{bmatrix} 1 & 0 & \dots & 0 \\ k_{m,1}^{f}(n) & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ k_{m,p}^{f}(n) & k_{m,p-1}^{f}(n) & \dots & 1 \end{bmatrix}. \tag{2.65}$$

The orthogonality between the components of prediction error vectors at different stages is achieved by two reference-orthogonal processors at each stage. At each reference orthogonal processor, the reflection coefficients at time n between orthogonalized backward $\hat{\mathbf{b}}_{\mathbf{m}}(n)$ (forward $\hat{\mathbf{f}}_{\mathbf{m}}(n)$) prediction error vector components and forward $\mathbf{f}_m(n)$ (backward $\mathbf{b}_{\mathbf{m}}(n)$) prediction error vector components are computed coloumn-wise and, a correlation cancelling operation is performed as follows,

$$\begin{bmatrix} b_{m+1}^{1}(n) \\ b_{m+1}^{2}(n) \\ \vdots \\ b_{m+1}^{p}(n) \end{bmatrix} = \begin{bmatrix} b_{m}^{1}(n-1) \\ b_{m}^{2}(n-1) \\ \vdots \\ b_{m}^{p}(n-1) \end{bmatrix} - \begin{bmatrix} k_{1,1}^{b}(n) & k_{1,2}^{b}(n) & \dots & k_{1,p}^{b}(n) \\ k_{2,1}^{b}(n) & k_{2,2}^{b}(n) & \dots & k_{2,p}^{b}(n) \\ \vdots & \vdots & & \vdots \\ k_{p,1}^{b}(n) & k_{p,2}^{b}(n) & \dots & k_{p,p}^{b}(n) \end{bmatrix} \begin{bmatrix} \hat{f}_{m}^{1}(n) \\ \hat{f}_{m}^{2}(n) \\ \vdots \\ \hat{f}_{m}^{p}(n) \end{bmatrix} \tag{2.66}$$

and

$$\begin{bmatrix} f_{m+1}^{1}(n) \\ f_{m+1}^{2}(n) \\ \vdots \\ f_{m+1}^{p}(n) \end{bmatrix} = \begin{bmatrix} f_{m}^{1}(n) \\ f_{m}^{2}(n) \\ \vdots \\ f_{m}^{p}(n) \end{bmatrix} - \begin{bmatrix} k_{1,1}^{f}(n) & \dots & k_{1,p}^{f}(n) \\ k_{2,1}^{f}(n) & \dots & k_{2,p}^{f}(n) \\ \vdots & \vdots & \vdots \\ k_{p,1}^{f}(n) & \dots & k_{p,p}^{f}(n) \end{bmatrix} \begin{bmatrix} \hat{b}_{m}^{1}(n-1) \\ \hat{b}_{m}^{2}(n-1) \\ \vdots \\ \hat{b}_{m}^{p}(n-1) \end{bmatrix}. \tag{2.67}$$

Thus, forward $\mathbf{f}_{m+1}(n)$ and backward $\mathbf{b}_{m+1}(n)$ prediction error vectors for the $(m+1)$th stage are obtained. The elements of the unit lower triangular matrices, $\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)$ and $\mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)$, in (2.65) and the square matrices in (2.66) and (2.67), which are called reflection coefficients, are all computed recursively in orthogonal processing cells represented as circles. In the coloumn-wise computation of the lower triangular matrices $\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)$ and $\mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)$, following series of steps are taken :

1. The RLS correlations of $b_m^1(n)$ and $f_m^1(n)$ are computed in the processing cells represented as double circles with the equations (2.43), (2.48), (2.53) or (2.57) depending on which version of the processing equations is implemented.

2. In versions I and II, The RLS cross-correlations between $b_m^1(n)$ and the other elements of $\mathbf{b}_m(n)$, which are $b_m^2(n),\ldots,b_m^p(n)$, and the RLS cross-correlation between $f_m^1(n)$ and the other elements of $\mathbf{f}_m(n)$, which are $f_m^2(n),\ldots,f_m^p(n)$, are computed in the processing cells represented as circles with the equations (2.45) or (2.51).

3. In versions I and II, the first coloumns of the matrices, $\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)$ and $\mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)$, are then computed in the cells represented as circles with the equations (2.46) or (2.52).

4. In versions III and IV, the first coloumns of the matrices, $\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)$ and $\mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)$, are computed directly without the computation of cross-correlations by using the equations (2.56) or (2.59).

5. After obtaining the new signals using the equations (2.47), (2.50), (2.55) or (2.60), the same steps are followed for the second coloumns of the matrices $\mathbf{k}_{\mathbf{m}}^{\mathbf{b}}(n)$ and $\mathbf{k}_{\mathbf{m}}^{\mathbf{f}}(n)$ and the third coloumns, so on.

Similarly, the coloumn-wise computation of the square transformation matrices in (2.66) and (2.67) is carried out by implementing the equations (2.46), (2.52), (2.56) or (2.59) in the processing cells represented as circles. Thus, each circular cell in SPMLS's takes part in the complete sequential orthogonalization of the input vector by computing a reflection coefficient in the unit lower triangular or square transformation matrices in the equations (2.65) or (2.66) and (2.67).

Since the output vectors $\hat{\mathbf{b}}_{\mathbf{m}}(n)$ and $\hat{\mathbf{f}}_{\mathbf{m}}(n)$ consists of uncorrelated random variables, the corresponding correlation matrices whose elements equal the mean-square values of the respective random variables that constitute the vector $\hat{\mathbf{b}}_{\mathbf{m}}(n)$ are diagonal. Thus, with the implementation of the (modified) Gram-Schmidth orthogonalization algorithm, the matrix inversion operation is transformed to a simple scalar operation. The desired signal $d(n)$ can now be estimated without matrix operations.

CHAPTER 3

IDENTIFICATION OF SECOND-ORDER VOLTERRA SYSTEMS USING LATTICE STRUCTURES

In this chapter, the identification of second-order Volterra systems using lattice structures is investigated. Second-order Volterra systems are identified with adaptive filters using sequential processing multichannel lattice stages. The adaptive filters have the same structure and the same number of coefficients as that of the system to be identified. A posteriori, indirect form of SPMLS algorithm is used in system identification problems in this chapter.

In the following sections, first, the general second-order adaptive Volterra system identification problem is described, then the RLS second-order Volterra system identification with SPMLS's is introduced. The second-order Volterra filtering algorithm and a block diagram of the filter structure is presented. In the last section, experimental results showing the performance of the new lattice filter are presented.

3.1 SYSTEM MODEL

A block diagram of the general second-order adaptive Volterra system identification problem is shown in Figure 3.1. Let $d(k)$ and $x(k)$ represent the desired response signal and the input signal, respectively, to the adaptive filter. The desired signal $d(k)$ to the adaptive filter is modelled by

$$d(k) = y(k) + e_o(k) \tag{3.1}$$

where $y(k)$ is the unknown system output and, $e_o(k)$ is the measurement noise. Both $d(k)$ and $x(k)$ are assumed to be wide-sense stationary with zero means. The problem is then to find an exponentially windowed, LS solution for the linear and quadratic coefficients of the adaptive filter that minimizes the cost function,

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - d_n(k) \mid^2 \tag{3.2}$$

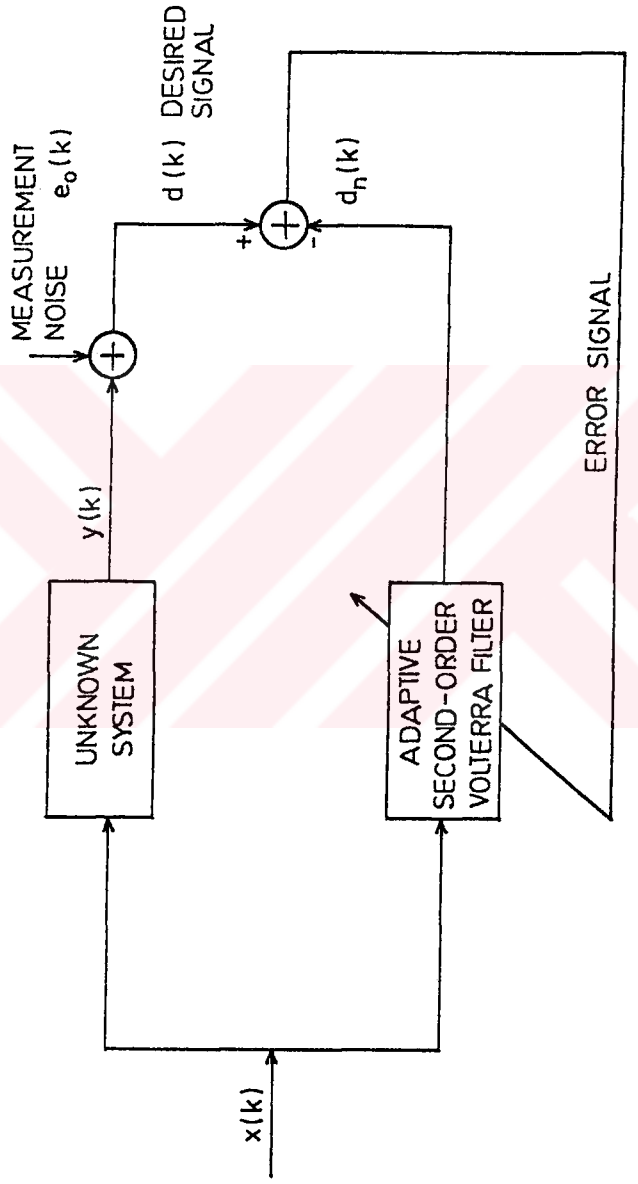


Figure 3.1: Block diagram of the general second-order adaptive Volterra system identification problem

at each time instant n. The filter output $d_n(k)$ is a second-order Volterra filter estimate of the desired signal $d(k)$ and,

$$d_n(k) = h_0 + \sum_{i=0}^{N-1} \hat{a}_i(n)x(k-i) + \sum_{i=0}^{N-1}\sum_{j=i}^{N-1} \hat{b}_{i,j}(n)x(k-i)x(k-j) \qquad (3.3)$$

where $\{\hat{a}_i\}$ and $\{\hat{b}_{i,j}\}$ are called the linear and quadratic filter weights, respectively, N is the filter length as in section 2.2. The first step in minimizing the cost function is to require the unbiasedness of the filter output. In other words, since the desired signal is zero mean, the filter output $d_n(k)$ should also be zero mean. The unbiased filter output is obtained by having the following relationship between h_0 and the quadratic filter weights [24],

$$h_0 = -\sum_{i=0}^{N-1}\sum_{j=i}^{N-1} \hat{b}_{i,j}(n)r_x(i-j) \qquad (3.4)$$

where $r_x(j) = E[x(n)x(n-j)]$ denotes the autocorrelation function of $x(n)$. By combining (3.2) and (3.3), the filter output can be expressed as

$$d_n(k) = \sum_{i=0}^{N-1} \hat{a}_i(n)x(k-i) + \sum_{i=0}^{N-1}\sum_{j=i}^{N-1} \hat{b}_{i,j}(n)[x(k-i)x(k-j) - r_x(i-j)]. \qquad (3.5)$$

Without the inclusion of h_o, the estimated coefficients of the filter would be biased and and hence, the error performance of the filter would worsen accordingly.

The input vector $\mathbf{X}(n)$ and the coefficient vector $\mathbf{W}(n)$, both having $N(N+3)/2$ elements, at time n are defined as

$$\mathbf{X}(n) = [x(n), x^2(n), x(n-1), x^2(n-1), x(n)x(n-1), \ldots, x(n)x(n-N+1)]^T \qquad (3.6)$$

and

$$\mathbf{W}(n) = [\hat{a}_0(n), \hat{b}_{0,0}(n), \hat{a}_1(n), \hat{b}_{1,1}(n), \hat{b}_{0,1}(n), \ldots, \hat{b}_{0,N-1}(n)]^T \qquad (3.7)$$

respectively. In the above "T" represents transpose of a matrix. Thus, the main concern of the exponentially weighted LS problem under consideration is to find, at each time n, the optimal coefficient vector $\mathbf{W}(n)$ that would minimize the cost function

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - \mathbf{W}^T(n)\mathbf{X}(k) \mid^2 . \qquad (3.8)$$

3.2 SYSTEM MODEL BASED ON SPMLS

The approach for developing a lattice structure for Volterra system identification is to transform the nonlinear system identification problem into an equivalent multichannel,

but linear, adaptive system identification problem. The basic idea is to partition the input vector $\mathbf{X}(n)$ at time n into the following set of smaller $(N+1)$ vectors so that each of the vectors can be considered as being formed from successive samples of signals from a different input channel as in section 2.5.

To avoid matrix inversion and vector operations and to achieve simplicity, good numerical properties and modularity, a complete, modified Gram-Schmidt orthogonalization of the input data $\mathbf{X}(n)$ is obtained using SPMLS.

The main objective in this section is to apply sequential processing lattice stages into the Volterra system identification problem. A block diagram of the second-order Volterra filter in system identification mode using SPMLS's with the filter length, $N = 3$ is shown in Figure 3.2. The basic idea employed here is to obtain a modified Gram-Schmidth orthogonalization of

$$\begin{aligned}\mathbf{X}(n) = [&x(n), x^2(n), x(n-1), x^2(n-1), x(n)x(n-1), x(n-2),\\ &x^2(n-2), x(n-1)x(n-2), x(n)x(n-2)]^T\end{aligned} \tag{3.9}$$

sequentially in such a way that an orthogonal basis set corresponding to $\mathbf{X}(n)$

$$\hat{\mathbf{b}}(n) = [\hat{b}_0^0(n), \hat{b}_0^1(n), \hat{b}_1^0(n), \hat{b}_1^1(n), \hat{b}_1^2(n), \hat{b}_2^0(n), \hat{b}_2^1(n), \hat{b}_2^2(n), \hat{b}_2^3(n)] \tag{3.10}$$

is obtained, so that the desired signal $d(n)$ can be estimated as a linear combination of the orthogonal basis set $\hat{\mathbf{b}}(n)$, instead of the elements of $\mathbf{X}(n)$.

To obtain the new orthogonal set sequentially, the elements of $\mathbf{X}(n)$ are grouped into three coloumns of a matrix as shown in (3.10).

$$\begin{bmatrix} x(n) & x(n-1) & x(n-2) \\ x(n)x(n) & x(n-1)x(n-1) & x(n-2)x(n-2) \\ & x(n)x(n-1) & x(n-1)x(n-2) \\ & & x(n)x(n-2) \end{bmatrix} \tag{3.11}$$

Each row in this matrix may be thought of as made up of samples of a signal belonging to a different channel in (2.13). Initially, the elements of the first coloumn, $x(n)$ and $x^2(n)$, are orthogonalized with a SOP and the new orthogonal signals, called backward prediction error (BPE) signals, $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$ are then obtained. The elements of the second coloumn are predicted from $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$, and the desired signal is estimated from $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$. Thus, the BPE signals, $b_1^0(n), b_1^1(n), b_1^2(n)$ and the first stage estimation error signal $e_1(n)$ are generated. Similarly, $x(n), x(n)x(n), x(n)x(n-1), x(n)x(n-2)$

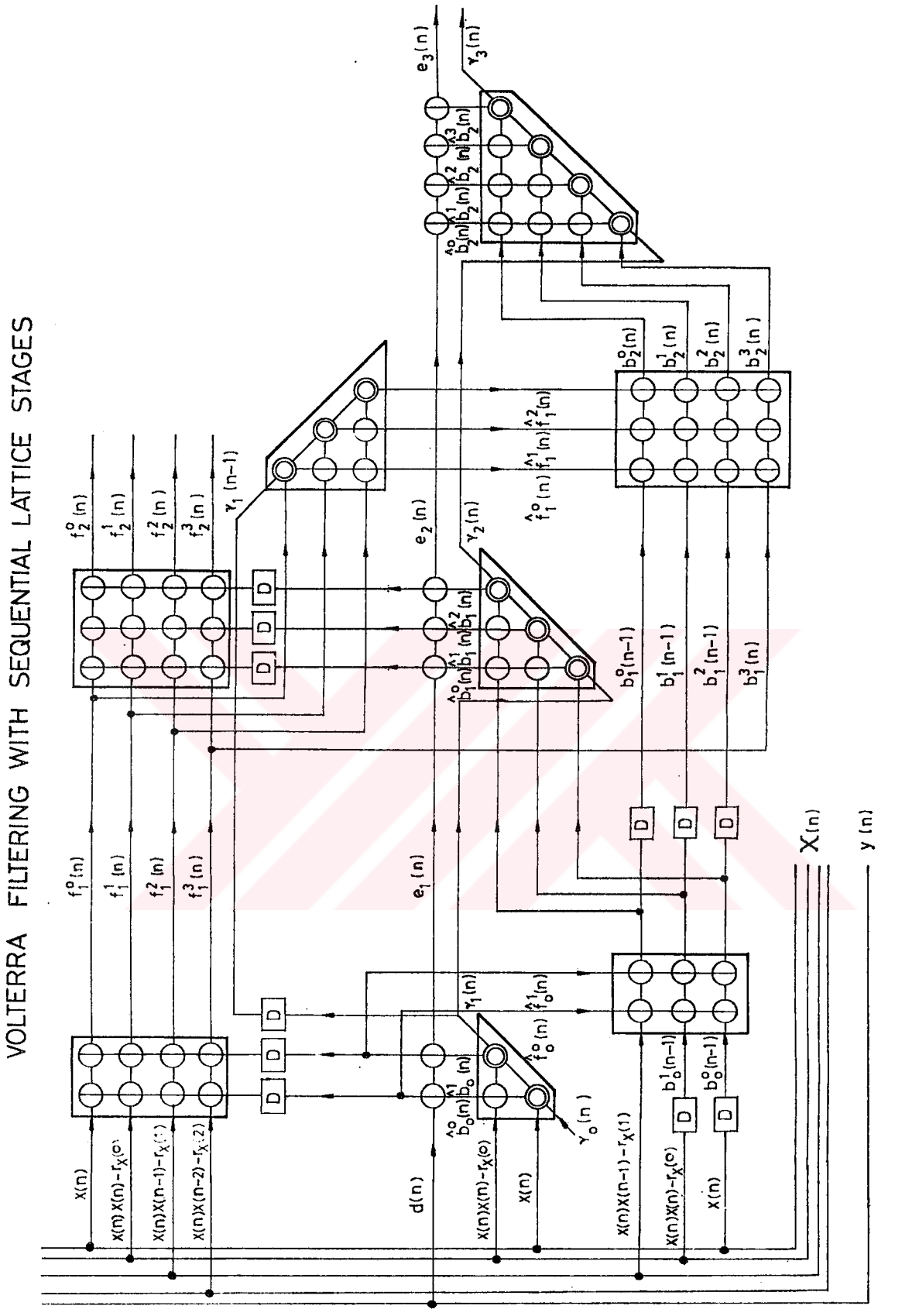


Figure 3.2: Block diagram of the filter structure for second-order Volterra systems with $N = 3$

are predicted from $\hat{b}_0^0(n-1)$, $\hat{b}_0^1(n-1)$ and the forward prediction errors (FPE) signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n)$ are obtained. Following these steps, the first stage BPE signals $b_1^0(n), b_1^1(n), b_1^2(n)$ are orthogonalized with a SOP, and the first stage FPE signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n)$ are predicted from the orthogonalized and delayed new BPE signals $\hat{b}_1^0(n-1), \hat{b}_1^1(n-1), \hat{b}_1^2(n-1)$. The first stage estimation error signal $e_1(n)$ is estimated from $\hat{b}_1^0(n), \hat{b}_1^1(n), \hat{b}_1^2(n)$ and then the second stage estimation error signal $e_2(n)$ is obtained. The FPE error signals $f_1^0(n), f_1^1(n), f_1^2(n)$ are similarly orthogonalized using a SOP. The delayed first stage backward prediction error signals, $b_1^0(n-1), b_1^1(n-1), b_1^2(n-1), b_1^3(n-1)$ are predicted from the orthogonolized FPE signals $\hat{f}_1^0(n), \hat{f}_1^1(n), \hat{f}_1^2(n)$ and the second stage BPE signals $b_2^0(n), b_2^1(n), b_2^2(n), b_2^3(n)$ are obtained.

The second stage backward prediction error signals are again orthogonalized with a SOP and $e_2(n)$ is estimated with the new orthogonalized BPE signals and so on. The complete algorithm for arbitrary N is presented in Table 3.1.

The key idea is that the complete orthogonalization of the data makes it possible to consider the LS optimization of each cell separately and the recursions defining the adaptive filter can be developed independently for each cell. With complete orthogonalization of the data, only scalar operations are required and instability problems due to matrix inversion are avoided, good numerical properties are achieved. Avoidance of matrix inversion and vector operations reduce the complexity considerably, make the filter simple, highly modular and suitable for VLSI implementation.

It is important to note that a cell in joint-process estimation part of the filter functions in no different manner than a cell in the prediction part. The only difference between them is the input signal. In the joint-process estimation part, the input signal is the desired signal or the relevant estimation error signal. In the prediction part, the input signal is the relevant backward or forward prediction error signal.

The computational complexity calculations for the second-order Volterra filter with SPMLS depends on the form of SPMLS algorithm is used and, on the number of channels. The number of channels in turn is related to the filter length N as shown in (2.13). The computational complexity for the filter can then be determined with the knowledge of the form of SPMLS algorithm and the number of channels. A comparative complexity analysis of the second-order Volterra filter is presented in Fig. 3.3. The version I of the SPMLS algorithm is used in the system identification mode of the filter.

Note that the number of multiplications required for the SPMLS algorithm is less than the conventional lattice algorithm in [33] and QR-decomposition based algorithm in [34]. On the other hand, the number of divisions required for the SPMLS algorithm is more than the conventional lattice algorithm.

The previous lattice filter structures for Volterra systems does not achieve a complete orthogonalization of the input data. In the previous lattice based structures as in [33], while signals in different stages of the filter are orthogonal to each other, signals in the same stage are not orthogonal to each other. This necessitates matrix inversion and vector operations, hence the matrix inversion lemma and makes the algorithm computationally complex. In the previous system identification applications with lattice structures, the linear and quadratic coefficients are obtained by converting the lattice reflections coefficients to the equivalent transversal filter coefficients. In the novel approach used here, the Volterra system coefficients are computed and identified efficiently and directly.

Table 3.1: SECOND ORDER VOLTERRA FILTERING ALGORITHM WITH SPMLS

Initialization

$e_0(n) = d(n)$

$\gamma_0(n) = 1.0$

$$\mathbf{b_0}(n) = \mathbf{f_0}(n) = \begin{bmatrix} x(n) \\ x^2(n) \end{bmatrix}$$

$$\hat{\mathbf{b}}_0(n) = \hat{\mathbf{f}}_0(n) = \mathbf{k}_0^{\mathbf{a}}(n)\mathbf{f_0}(n)$$

$$\mathbf{b_1}(n) = \begin{bmatrix} \mathbf{b_0}(n-1) \\ \cdots \\ x(n)x(n-1) \end{bmatrix} - \mathbf{k}_1^{\mathbf{b}}(n)\hat{\mathbf{f}}_0(n)$$

$$\mathbf{f_1}(n) = \begin{bmatrix} \mathbf{f_0}(n) \\ \cdots \\ x(n)x(n-1) \\ \vdots \\ x(n)x(n-j) \end{bmatrix} - \mathbf{k}_1^{\mathbf{f}}(n)\hat{\mathbf{b}}_0(n-1)$$

where $\mathbf{k}_0^{\mathbf{a}}(n)$ is 2 x 2 unit lower triangular coefficient matrix, $\mathbf{k}_1^{\mathbf{b}}(\mathbf{n})$ is 3 x 2 BPE coefficient matrix, $\mathbf{k}_1^{\mathbf{f}}(n)$ is (j+2) x 2 FPE coefficient matrix and j=2,3,...,N-1.

$e_1(n) = e_0(n) - \mathbf{k}_1^{\mathbf{d}}(n)\hat{\mathbf{b}}_0(n)$

$\gamma_1(n) = \gamma_0(n) - \hat{\mathbf{b}}_0^{\mathbf{T}}(n)\mathbf{r}_0^{-1}(n)\hat{\mathbf{b}}_0(n)$

where $\mathbf{r_0}(n)$ is the diagonal LS autocorrelation matrix of $\hat{\mathbf{b}}_0(n)$ and $\mathbf{k}_1^{\mathbf{d}}(n)$ is 1 x 2 estimation error coefficient matrix.

Stages 2 thru N-1

do for i=2,3,...,N-1

Backward Prediction Error Update

$$\mathbf{b_i}(n) = \begin{bmatrix} \mathbf{b_{i-1}}(n) \\ \cdots \\ f_{i-1}^{i+1}(n) \end{bmatrix} - \mathbf{k}_i^{\mathbf{b}}(n)\hat{\mathbf{f}}_{i-1}(n)$$

$$\hat{\mathbf{b}}_i(n) = \mathbf{k}_i^{\mathbf{a}}(n)\mathbf{b_i}(n)$$

Forward Prediction Error Update

$$\mathbf{f_i}(n) = \mathbf{f_{i-1}}(n) - \mathbf{k}_i^{\mathbf{f}}(n)\hat{\mathbf{b}}_{i-1}(n-1)$$

$$\hat{\mathbf{f}}_i(n) = \mathbf{k}_i^{\mathbf{a}}(n)\mathbf{f_i}(n)$$

where $\mathbf{k}_i^{\mathbf{b}}(n)$ and $\mathbf{k}_i^{\mathbf{f}}(n)$ are (i+2) x (i+1) prediction error coefficient matrices and $\mathbf{k}_i^{\mathbf{a}}(n)$'s are unit lower triangular coefficient matrices.

Likelihood Variable Update

$\gamma_i(n) = \gamma_{i-1}(n) - \hat{\mathbf{b}}_i^{\mathbf{T}}(n)\mathbf{r}_i^{-1}(n)\hat{\mathbf{b}}_i(n)$

where $\mathbf{r_i}(n)$ is the diagonal LS autocorrelation matrix of $\hat{\mathbf{b}}_i(n)$.

Joint Process Estimation Error Update

$e_i(n) = e_{i-1}(n) - \mathbf{k}_i^{\mathbf{d}}(n)\hat{\mathbf{b}}_{i-1}(n)$

where $\mathbf{k}_i^{\mathbf{d}}(n)$ is 1 x (i+1) estimation error coefficient matrix.

end

3.3 EXPERIMENTAL RESULTS

In this section we present several experimental results to show the performance of the second-order Volterra filtering algorithm with SPMLS. The MATLAB programming environment was used in the experiments. In these experiments the adaptive filters were run in the identification mode with the same structure and the same number of coefficients as that of the system that was to be identified with $N = 3$ and the exponential weighting factor is 0.995. The input signal is a white, zero-mean, pseudo-Gaussian noise, and the measurement noise signal is an additive white, zero-mean, pseudo-Gaussian, uncorrelated with the input signal. The system to be identified had three linear and six quadratic coefficients. These coefficients are given in Table 3.2.

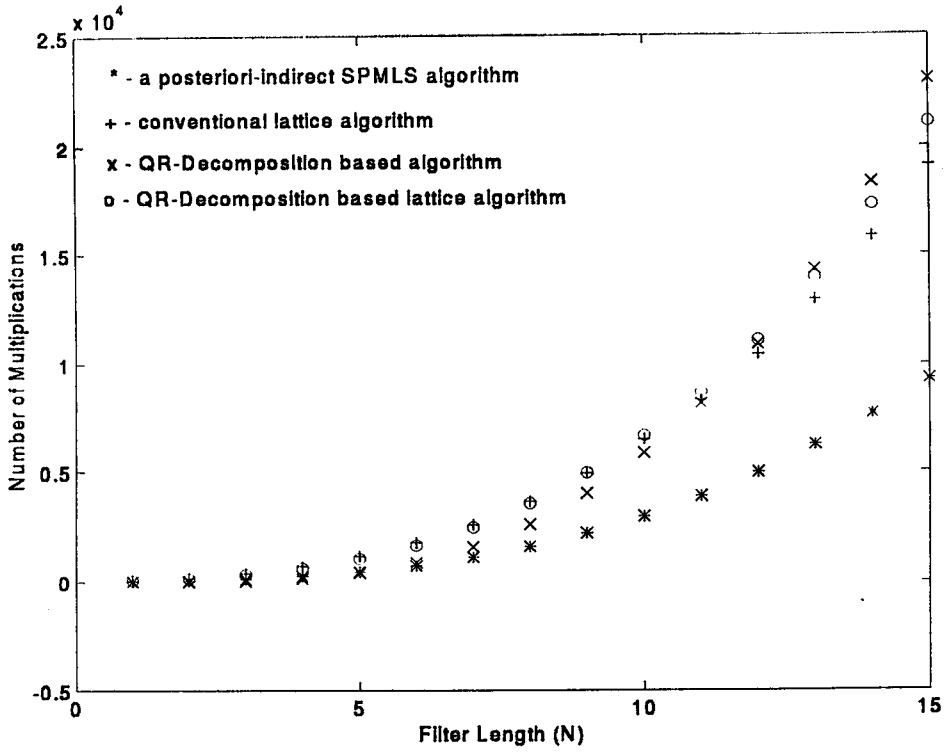
In the experiments, the parameters of the input signal were chosen such that the unknown system output power is approximately unity. The desired signal $d(n)$ was obtained by adding the measurement noise to the output signal of the unknown system. The variance of measurement noise was adjusted so as to obtain the desired signal-to-measurement noise ratio. The autocorrelation values, $r_x(j)$, for the estimation of the zeroth-order term h_o are computed by time-averaging the input vector components $x(n)x(n-j)$. The computed values of $r_x(j)$ are then subtracted from the input vector components, $x(n)x(n-j)$ at time n, before the first SPMLS. The results presented are ensemble averages of 50 independent runs. Performance evaluations were carried out by plotting the mean-squared value of the a posteriori estimation error and the estimated mean coefficient trajectories of the unknown system. The plots of the ensemble averaged a posteriori mean-squared error for two different signal-to-measurement noise ratios, 20 dB and 30 dB, are shown in Figure 3.4. Figures 3.5-3.6 show the estimated mean coefficient trajectories for 20 dB and 30 dB signal-to-measurement noise ratios. Observe that bias and fluctuations occur in the mean coefficient trajectories due to measurement noise and the choice of exponential weighting factor less than one, respectively.

Several experiments were conducted to compare the performance of the filter with that of the RLS transversal second-order Volterra filter, mentioned in section 2.4. In Figures 3.6-3.8, the mean-square error and the estimated mean coefficient trajectory performance of both filters are shown. Observe that the new lattice filter has better convergence due to the complete orthogonalization and the transversal filter has better self-noise performance. Note that the extensive simulation conducted reveal the fact that, the self-noise performance of the new lattice filter can be improved with a larger

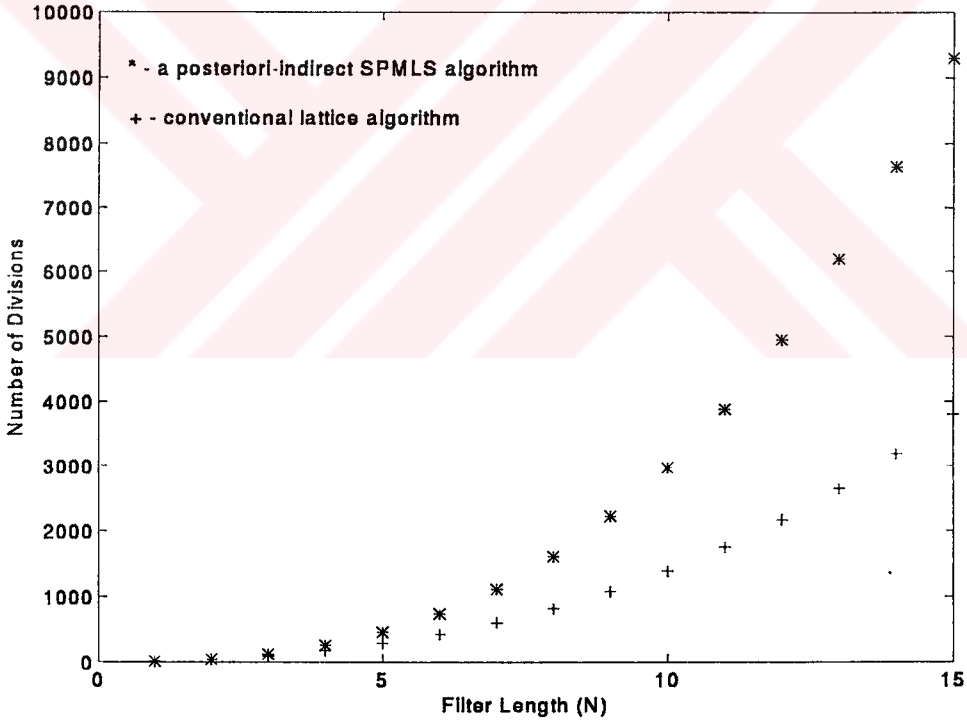
Table 3.2: LINEAR AND QUADRATIC COEFFICIENTS OF THE UNKNOWN SYSTEM

	a_i		$b_{i,j}$	
i	j=	0	1	2
0	-0.78	0.54	3.72	1.86
1	-1.48	0.00	-1.62	0.76
2	-1.39	0.00	0.00	1.41

exponential weighting factor at the expense of decreasing the convergence speed.



(a)



(b)

Figure 3.3: Complexity curves for second-order Volterra systems

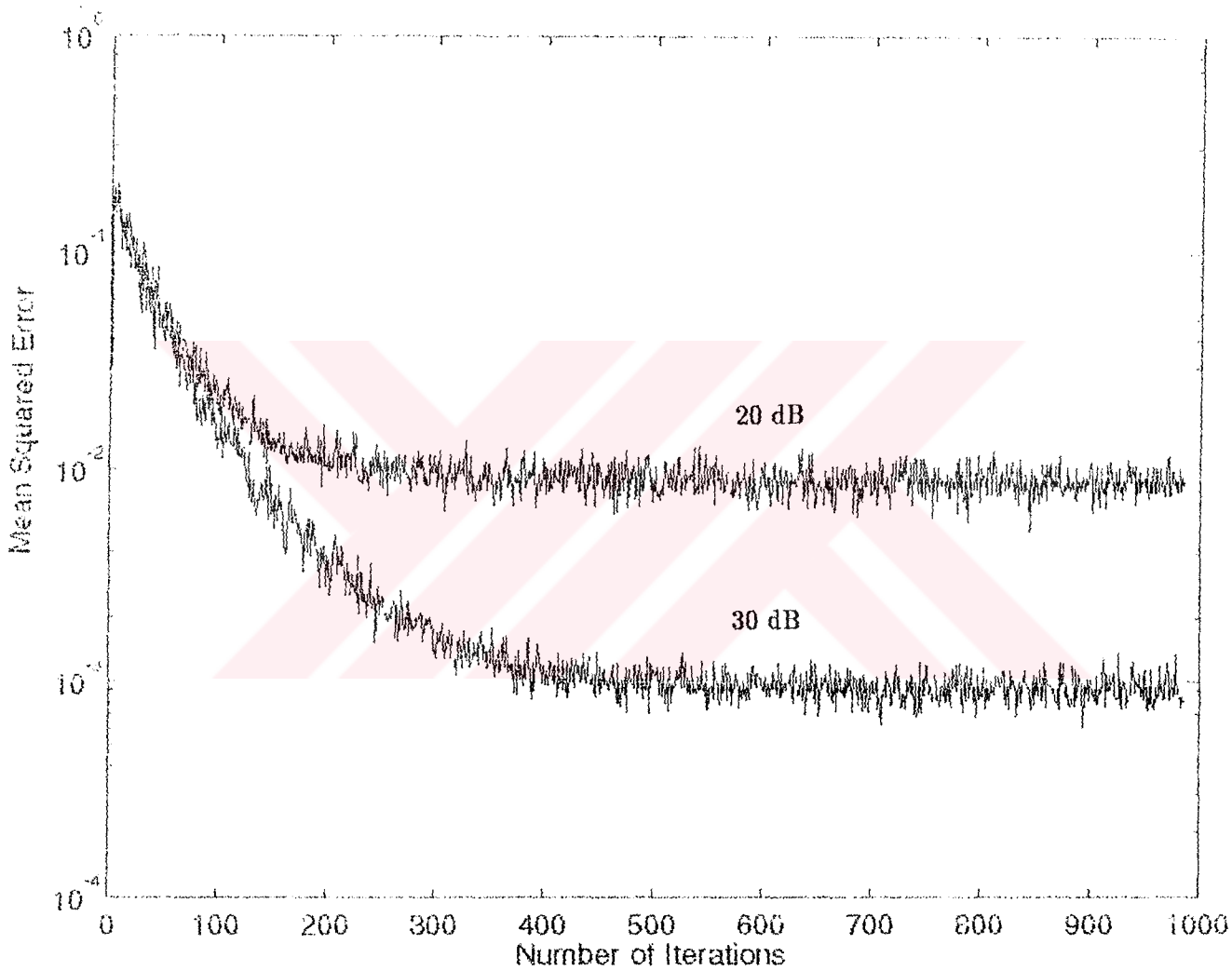
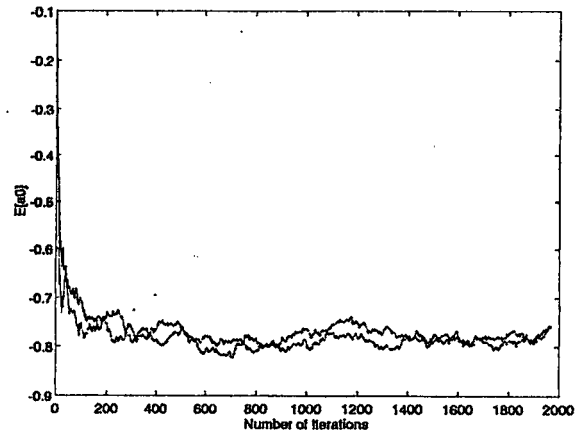
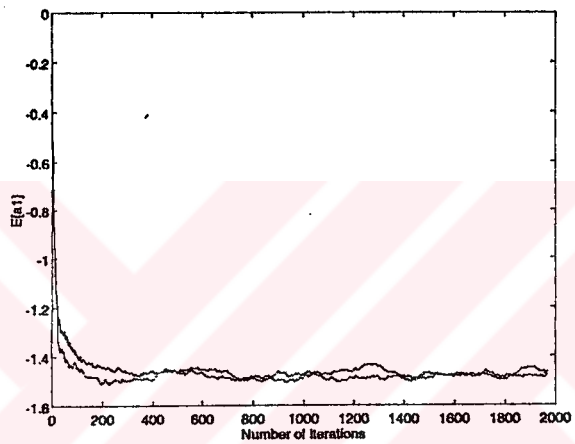


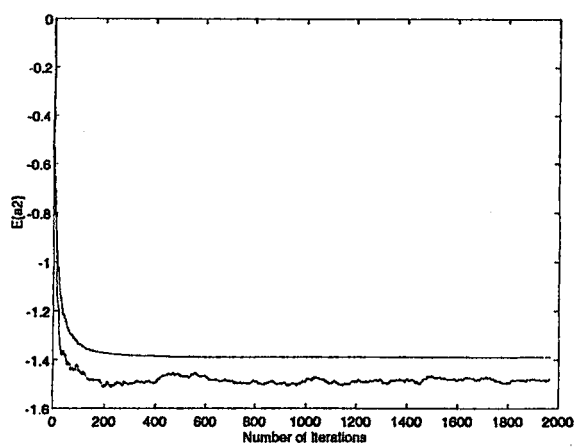
Figure 3.4: Learning curves for second-order Volterra system identification with $N = 3$ and SNR $= 20, 30$ dB



(a)

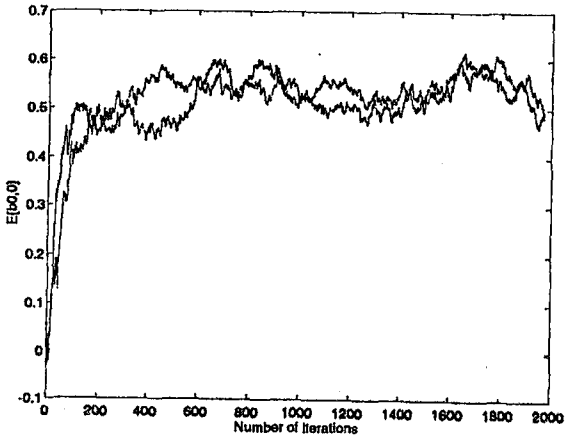


(b)

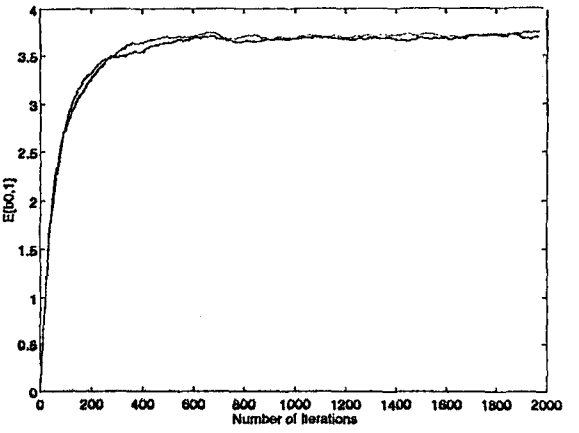


(c)

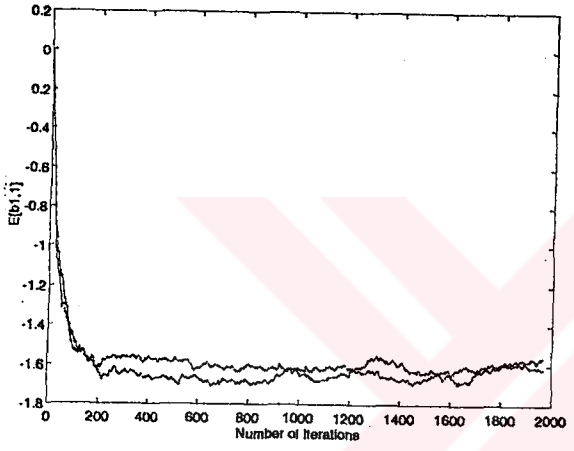
Figure 3.5: Mean trajectories of linear coefficients for 20 dB and 30 SNR's



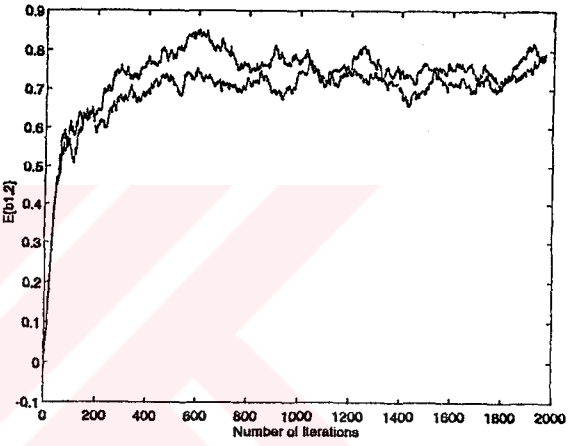
(a)



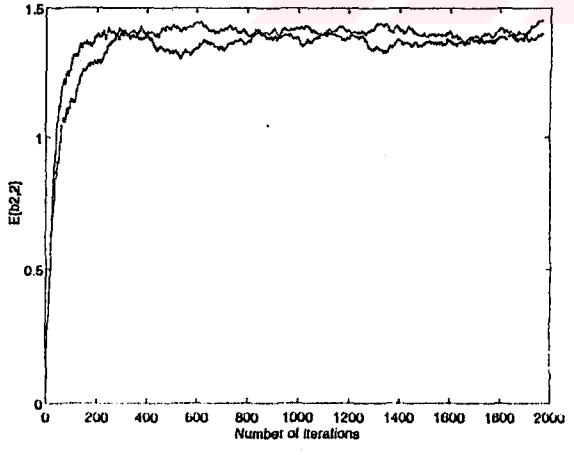
(b)



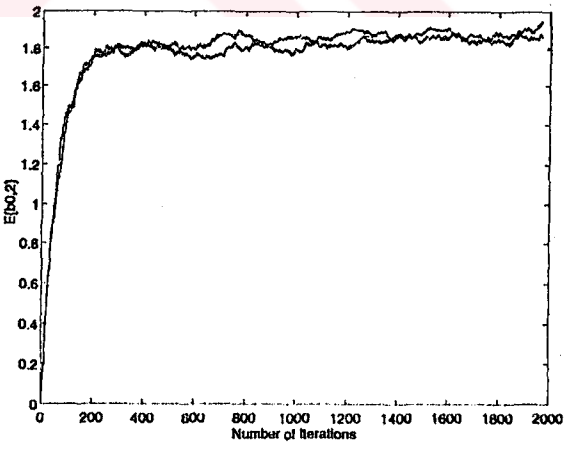
(c)



(d)



(e)



(f)

Figure 3.6: Mean trajectories of quadratic coefficients for 20 dB and 30 SNR's

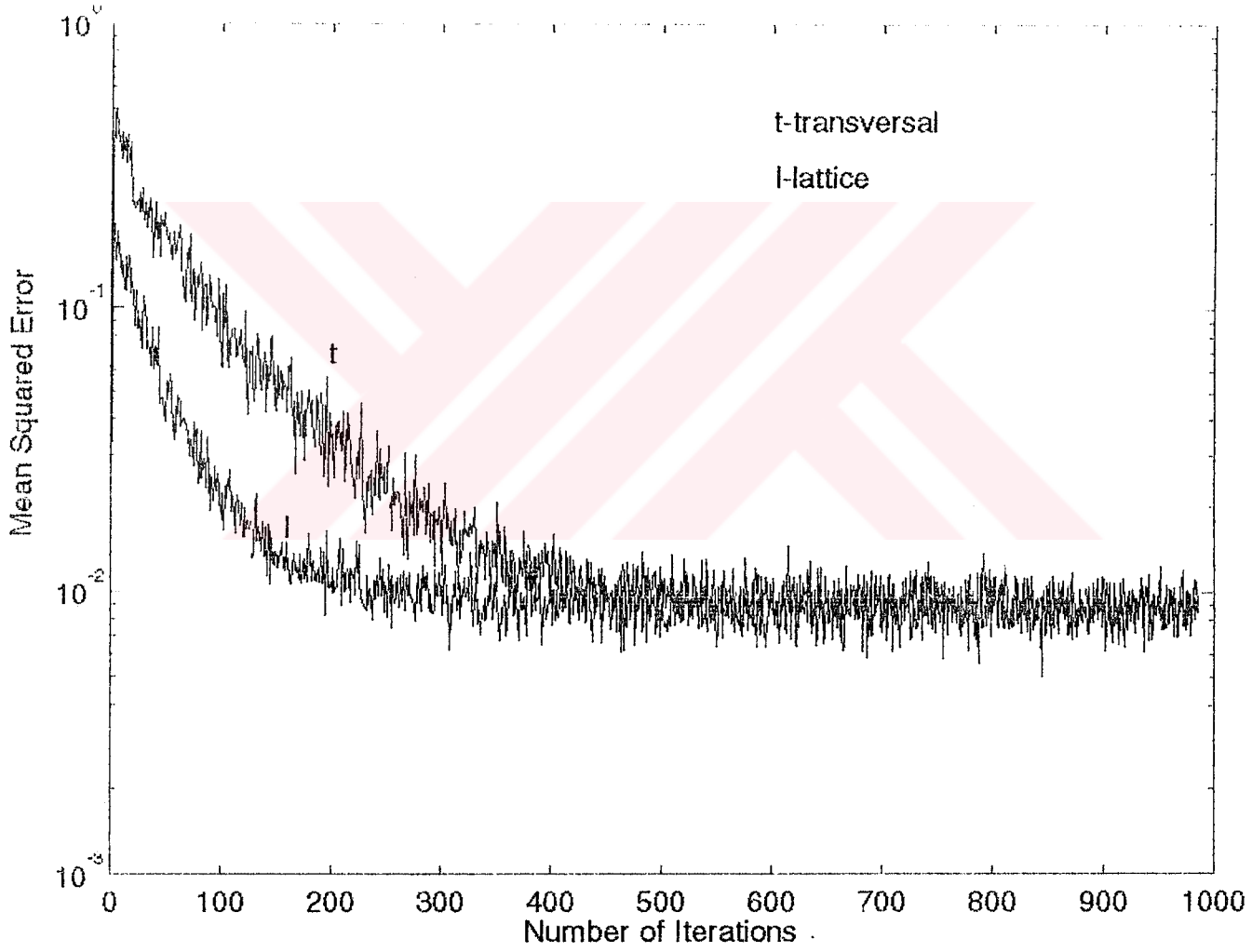
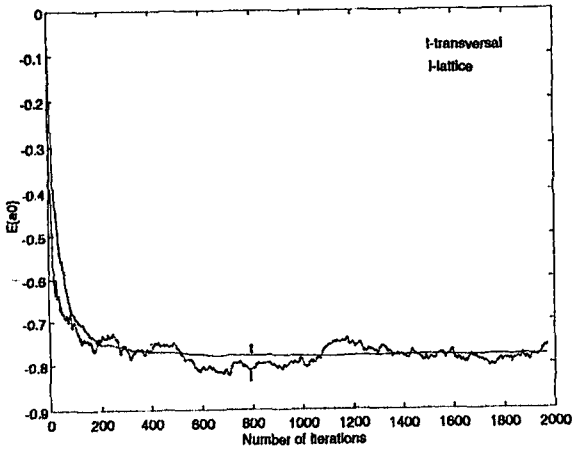
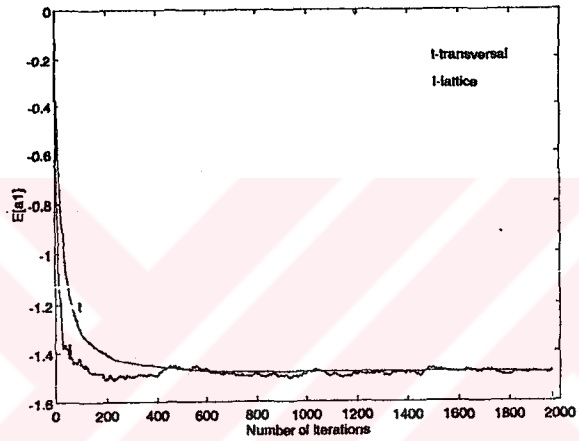


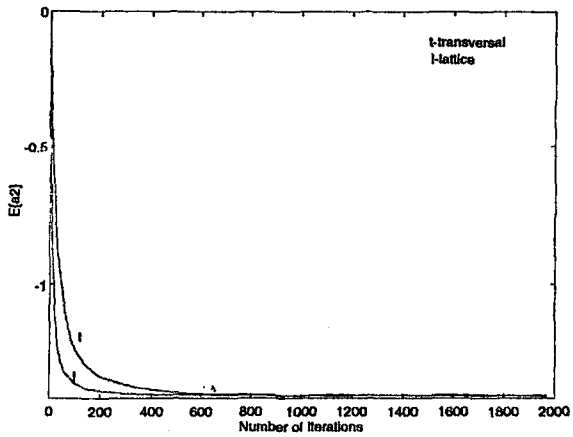
Figure 3.7: Learning curves for second-order Volterra system identification with $N = 3$ and SNR = 20 dB



(a)

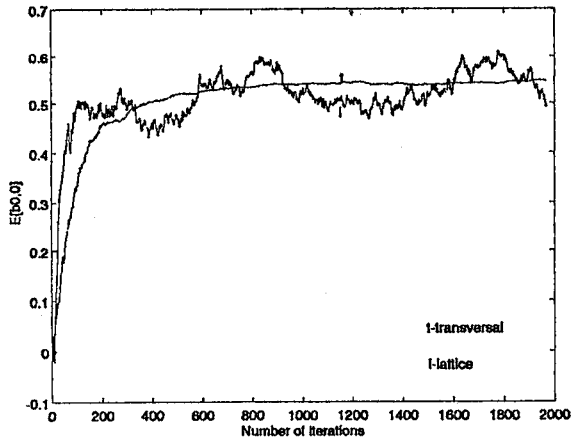


(b)

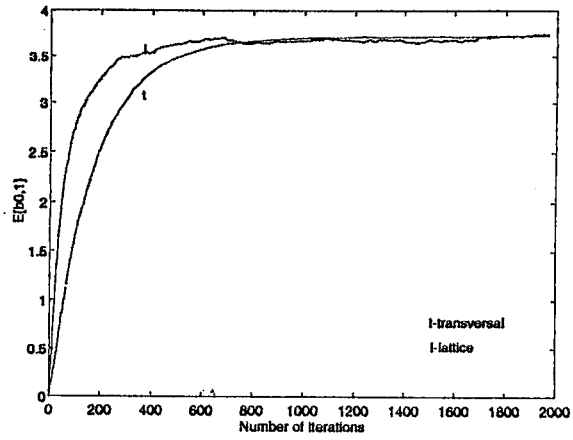


(c)

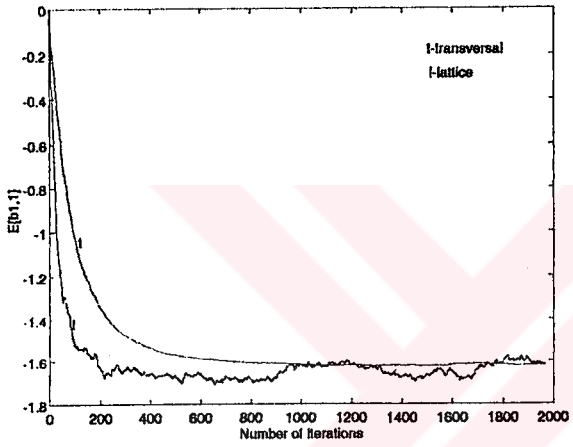
Figure 3.8: Mean trajectories of linear coefficients for 20 dB SNR



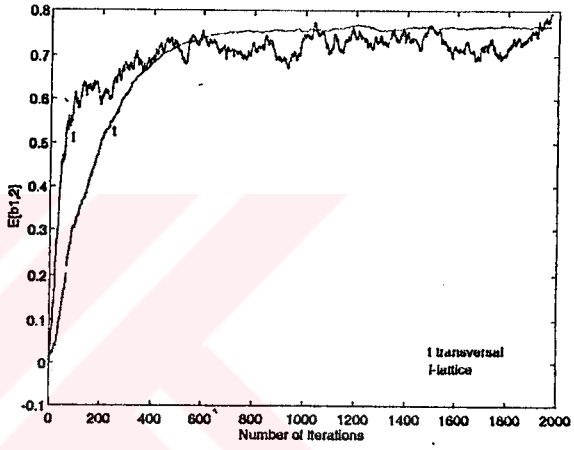
(a)



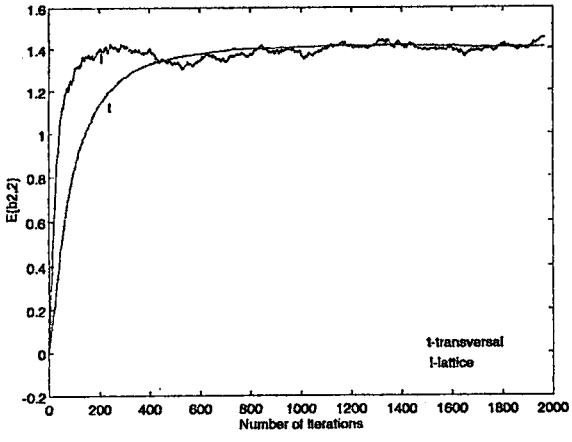
(b)



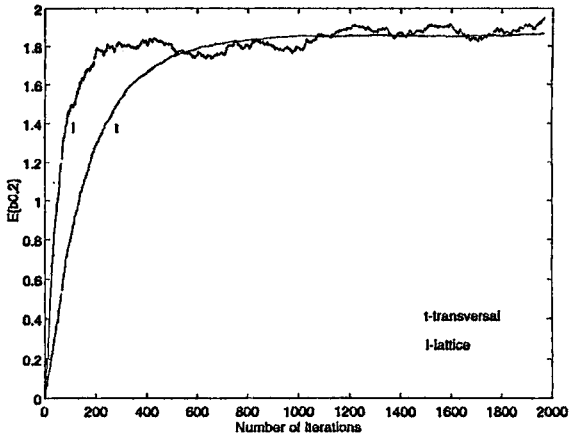
(c)



(d)



(e)



(f)

Figure 3.9: Mean trajectories of quadratic coefficients for 20 dB SNR

CHAPTER 4

SECOND-ORDER LS ADAPTIVE CHANNEL EQUALIZATION

The problem of equalizing a channel whose channel correlation matrix has a large eigenvalue spread is well known. Channels with nonlinear entries will cause the eigenvalue spread to be large even when the input signal is white. A typical channel equalizer is a transversal filter with enough taps to approximate the inverse transfer function of the channel. In many situations, the channel is not known in advance, or it may be time-varying as in the case of multipath channels. Therefore, it is desirable to design adaptive equalizers.

Lattice structures offer a number of potential advantages over transversal filters in equalization applications. One advantage of lattice structures is their modularity. This property of lattice structures allows the dynamic assignment of the particular length of the lattice equalizer which proves the most effective at any instant of equalization. A second advantage of lattice structures is that longer lattice filters may be built up from shorter ones by simply adding on more lattice stages. These properties prove useful in designing VLSI implementations. Another important advantage of lattice structures is their robust numerical properties, and their insensitivity to roundoff noise.

In this chapter, the equalization of Volterra type nonlinear channels are considered. In the following sections, first the general adaptive second-order Volterra channel equalization problem is described, then the RLS second-order adaptive Volterra channel equalization using SPMLS's is introduced. In the final section, experimental results showing the performance of the adaptive second-order Volterra equalization filter are presented.

4.1 MODEL

The purpose of a channel equalizer is to undo the distorting effects of the channel and recover, from the received signal, the input signal. The block diagram of a general second-order adaptive Volterra channel equalization problem is shown in Figure 4.1. The desired signal $d(k)$ in channel equalization problem is the appropriately delayed

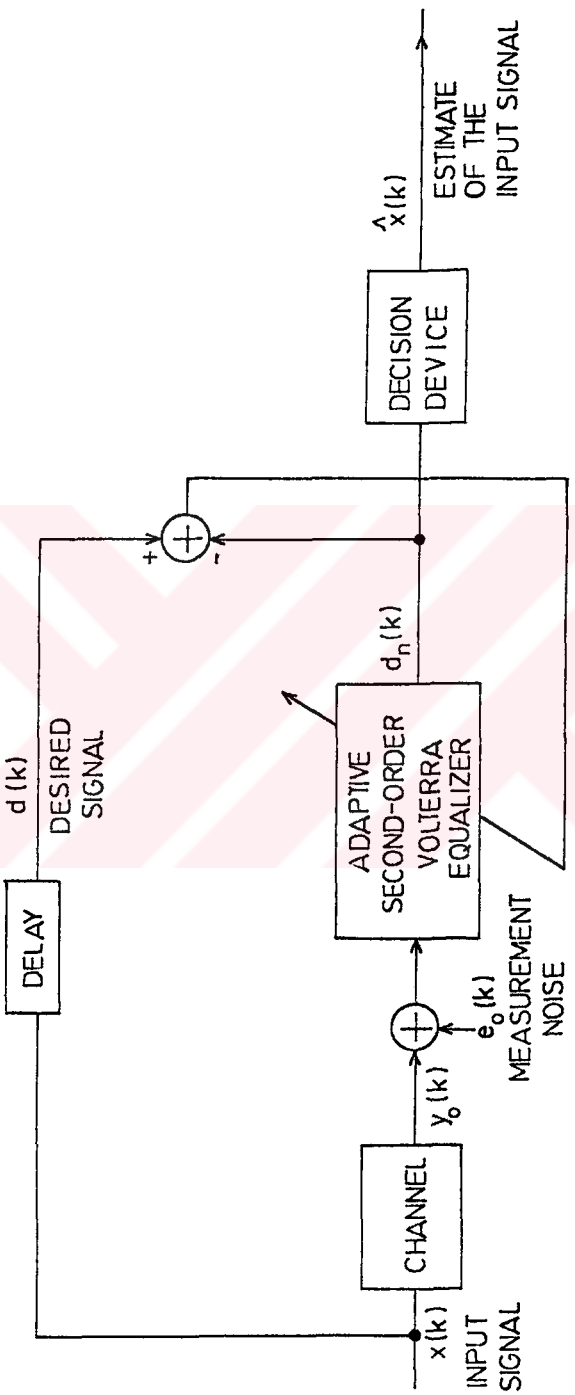


Figure 4.1: Block diagram of a general channel equalization problem

input signal, $x(k-D)$. The input signal $y(k)$ to the equalizer is modeled by

$$y(k) = y_o(k) + e_o(k) \tag{4.1}$$

where $y_o(k)$ is the output signal from the channel and $e_o(k)$ is the measurement noise. The input signal, $x(k)$, to the channel to be equalized and the input signal, $y(k)$, to the equalizer are both wide-sense stationary and zero mean. The problem is then to find an exponentially windowed, LS solution for the linear and quadratic coefficients of the adaptive equalizer that minimizes the cost function,

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - d_n(k) \mid^2 \tag{4.2}$$

at each time instant n. The filter output $d_n(k)$ is a second-order Volterra filter estimate of the desired signal $d(k)$ and,

$$d_n(k) = h_0 + \sum_{i=0}^{N-1} \hat{a}_i(n) y(k-i) + \sum_{i=0}^{N-1} \sum_{j=i}^{N-1} \hat{b}_{i,j}(n) y(k-i) y(k-j) \tag{4.3}$$

where $\{\hat{a}_i\}$ and $\{\hat{b}_{i,j}\}$ are called the linear and quadratic filter weights, respectively, N is the equalizer length as in section 2.2. The first step in minimizing the cost function is to require the unbiasness of the equalizer output. The unbiased filter output is obtained by having the following relationship between h_0 and the quadratic filter weights

$$h_0 = -\sum_{i=0}^{N-1} \sum_{j=i}^{N-1} \hat{b}_{i,j}(n) r_y(i-j) \tag{4.4}$$

where $r_y(j) = E[y(n)y(n-j)]$ denotes the autocorrelation function of $y(n)$. By combining (4.2) and (4.3), the equalizer output is expressed as

$$d_n(k) = \sum_{i=0}^{N-1} \hat{a}_i(n) y(k-i) + \sum_{i=0}^{N-1} \sum_{j=i}^{N-1} \hat{b}_{i,j}(n)[y(k-i)y(k-j) - r_y(i-j)]. \tag{4.5}$$

The input vector $\mathbf{Y}(n)$ and the coefficient vector $\mathbf{W}(n)$, both having $N(N+3)/2$ elements, at time n are defined as

$$\mathbf{Y}(n) = [y(n), y^2(n), y(n-1), y^2(n-1), y(n)y(n-1), \ldots, y(n)y(n-N+1)]^T \tag{4.6}$$

and

$$\mathbf{W}(n) = [\hat{a}_0(n), \hat{b}_{0,0}(n), \hat{a}_1(n), \hat{b}_{1,1}(n), \hat{b}_{0,1}(n), \ldots, \hat{b}_{0,N-1}(n)]^T \tag{4.7}$$

respectively. In the above "T" represents transpose of a matrix. Thus, the main concern of the exponentially weighted LS problem under consideration is to find, at each time n, the optimal coefficient vector $\mathbf{W}(n)$ that would minimize the cost function

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - \mathbf{W}^T(n)\mathbf{Y}(k) \mid^2 . \tag{4.8}$$

4.2 MODEL BASED ON SPMLS

The approach for developing a lattice structure for Volterra channel equalization is to transform the nonlinear system equalization problem into an equivalent multichannel, but linear, adaptive channel equalization problem. The basic idea is to partition the input vector $\mathbf{Y}(n)$ at time n into the following set of smaller $(N+1)$ vectors

$$\begin{aligned}
CH1: &\quad [y(n), y(n-1), \ldots, y(n-N+1)] \\
CH2: &\quad [y(n)y(n), y(n-1)y(n-1), \ldots, y(n-N+1)y(n-N+1)] \\
CH3: &\quad [y(n)y(n-1), y(n-1)y(n-2), \ldots, y(n-N+2)y(n-N+1)] \\
CH4: &\quad [y(n)y(n-2), y(n-1)y(n-3), \ldots, y(n-N+3)y(n-N+1)] \\
CH5: &\quad [y(n)y(n-3), y(n-1)y(n-4), \ldots, y(n-N+4)y(n-N+1)] \\
&\vdots \\
CH(N+1): &\quad [y(n)y(n-N+1)]
\end{aligned} \tag{4.9}$$

so that each of the vectors can be considered as being formed from successive samples of signals from a different input channel, where N is the length of the equalization filter.

To avoid matrix inversion and vector operations and to achieve simplicity, good numerical properties and modularity, a complete, modified Gram-Schmidt orthogonalization of the input data $\mathbf{Y}(n)$ is obtained using SPMLS.

The main objective in this section is to apply sequential processing lattice stages into the Volterra channel equalization problem. A block diagram of the second-order Volterra equalization filter using SPMLS's with the filter length, $N = 5$ is shown in Appendix A. The basic idea employed here is to obtain a modified Gram-Schmidth orthogonalization of

$$\begin{aligned}
\mathbf{Y}(n) = [&y(n), y^2(n), y(n-1), y^2(n-1), y(n)y(n-1), y(n-2), \\
&y^2(n-2), y(n-1)x(n-2), y(n)y(n-2), y(n-3), y(n-3)y(n-3), \\
&y(n-2)y(n-3), y(n-1)y(n-3), y(n)y(n-3), y(n-4), y(n-4)y(n-4), \\
&y(n-3)y(n-4), y(n-2)y(n-4), y(n-1)y(n-4), y(n)y(n-4)]^T
\end{aligned} \tag{4.10}$$

sequentially in such a way that an orthogonal basis set corresponding to $\mathbf{Y}(n)$

$$\begin{aligned}
\hat{\mathbf{b}}(n) = [&\hat{b}_0^0(n), \hat{b}_0^1(n), \hat{b}_1^0(n), \hat{b}_1^1(n), \hat{b}_1^2(n), \hat{b}_2^0(n), \hat{b}_2^1(n), \hat{b}_2^2(n), \hat{b}_2^3(n), \hat{b}_3^0(n), \\
&\hat{b}_3^1(n), \hat{b}_3^2(n), \hat{b}_3^3(n), \hat{b}_3^4(n), \hat{b}_4^0(n), \hat{b}_4^1(n), \hat{b}_4^2(n), \hat{b}_4^3(n), \hat{b}_4^4(n), \hat{b}_4^5(n)]
\end{aligned} \tag{4.11}$$

is obtained, so that the desired signal $x(n-D)$ can be estimated as a linear combination of the orthogonal basis set $\hat{\mathbf{b}}(n)$, instead of the elements of $\mathbf{Y}(n)$.

To obtain the new orthogonal set sequentially, the elements of $\mathbf{Y}(n)$ are grouped into five coloumns of a matrix as shown in (4.11).

$$\begin{bmatrix} y(n) & y(n-1) & y(n-2) & y(n-3) & y(n-4) \\ y(n)y(n) & y(n-1)y(n-1) & y(n-2)y(n-2) & y(n-3)y(n-3) & y(n-4)y(N-4) \\ & y(n)y(n-1) & y(n-1)y(n-2) & y(n-2)y(n-3) & y(n-3)y(N-4) \\ & & y(n)y(n-2) & y(n-1)y(n-3) & y(n-2)y(n-4) \\ & & & y(n)y(n-3) & y(n-1)y(n-4) \\ & & & & y(n)y(n-4) \end{bmatrix} \tag{4.12}$$

Each row in this matrix may be thought of as made up of samples of a signal belonging to a different channel in (4.8). Initially, the elements of the first coloumn ,$y(n)$ and $y^2(n)$, are orthogonalized with a SOP and the new orthogonal signals, called backward prediction error (BPE) signals, $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$ are then obtained. The elements of the second coloumn are predicted from $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$, and the desired signal $x(n-D)$ is estimated from $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$. Thus, the BPE signals, $b_1^0(n), b_1^1(n), b_1^2(n)$ and the first stage estimation error signal $e_1(n)$ are generated. Similarly, $y(n), y(n)y(n), y(n)y(n-1), y(n)y(n-2), y(n)y(n-3), y(n)y(n-4)$ are predicted from $\hat{b}_0^0(n-1)$, $\hat{b}_0^1(n-1)$ and the forward prediction errors (FPE) signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n), f_1^4(n), f_1^5(n)$ are obtained. Following these steps, the first stage BPE signals $b_1^0(n), b_1^1(n), b_1^2(n)$ are orthogonalized with a SOP, and the first stage FPE signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n), f_1^4(n), f_1^5(n)$ are predicted from the orthogonalized and delayed new BPE signals $\hat{b}_1^0(n-1), \hat{b}_1^1(n-1), \hat{b}_1^2(n-1)$. The first stage estimation error signal $e_1(n)$ is estimated from $\hat{b}_1^0(n), \hat{b}_1^1(n), \hat{b}_1^2(n)$ and then the second stage estimation error signal $e_2(n)$ is obtained. The FPE error signals $f_1^0(n), f_1^1(n), f_1^2(n)$ are similarly orthogonalized using a SOP. The delayed first stage backward prediction error signals, $b_1^0(n-1), b_1^1(n-1), b_1^2(n-1), b_1^3(n-1)$ are predicted from the orthogonolized FPE signals $\hat{f}_1^0(n), \hat{f}_1^1(n), \hat{f}_1^2(n)$ and the second stage BPE signals $b_2^0(n), b_2^1(n), b_2^2(n), b_2^3(n)$ are obtained.

The second stage backward prediction error signals are again orthogonalized with a SOP and $e_2(n)$ is estimated with the new orthogonalized BPE signals and so on.

With the a priori, direct form of SPMLS algorithm is used and the longer filter length, the complexity increases comparing to the system identification problem in the previous chapter.

4.3 EXPERIMENTAL RESULTS

In this section, several experimental results are presented to show the performance of the second-order Volterra equalization filter. The channel input signal is a white, zero-mean, pseudorandom bipolar (+/- 1) signal. The measurement noise signal is an additive, white, zero-mean, pseudo-Gaussian, uncorrelated with the input signal. The noise variance in all experiments was adjusted to be 0.001. The channel to be equalized had three linear and six quadratic coefficients. The linear coefficients of the second-order Volterra channel are defined by the raised cosine shape [41],

$$a_i = \begin{cases} \frac{1}{2}[1 + \cos(\frac{2\pi}{W}(i-1))], & \text{i=0,1,2} \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

and, the quadratic coefficients are defined by $b_{i,j} = a_i.a_j$ and $b_{i,j} = b_{j,i}$. W in (4.13) controls the amount of distortion and therefore the eigenvalue spread produced by the channel. The desired signal $d(n)$ was obtained by delaying the input signal suitably so that the smallest mean squared error for the equalizer is obtained. The delay parameter for the smallest mean square error was three. The length, N, of the second-order Volterra channel to be equalized was three. Exploiting the modularity property of lattice structures , the particular length of the adaptive equalizer which proved most effective was assigned dynamically and, the most effective equalizer length, N, was five. The exponential weighting factor was chosen as 0.995. Experiments were carried out for $W = 2.9, 3.1, 3.3, 3.5$ and $W = 2.9$ with nonlinear coefficients zero. The autocorrelation values, $r_y(j)$, for the estimation of the zeroth-order term h_o are computed by time-averaging the input vector components $y(n)y(n-j)$. The computed values of $r_y(j)$ are then subtracted from the input vector components, $y(n)y(n-j)$ at time n, before the first SPMLS. The results presented are ensemble averages of 50 independent runs. Performance evaluations were carried out by plotting the mean squared values of the a priori estimation errors. Figure 4.3 shows the mean squared error curves for different channel distortions. Notice that the performance of the equalizer worsens as the eigenvalue spread or distortion increases. The input signal, estimated signal and the threshold detector output signal when $W = 2.9$ is shown in Figure 4.4.

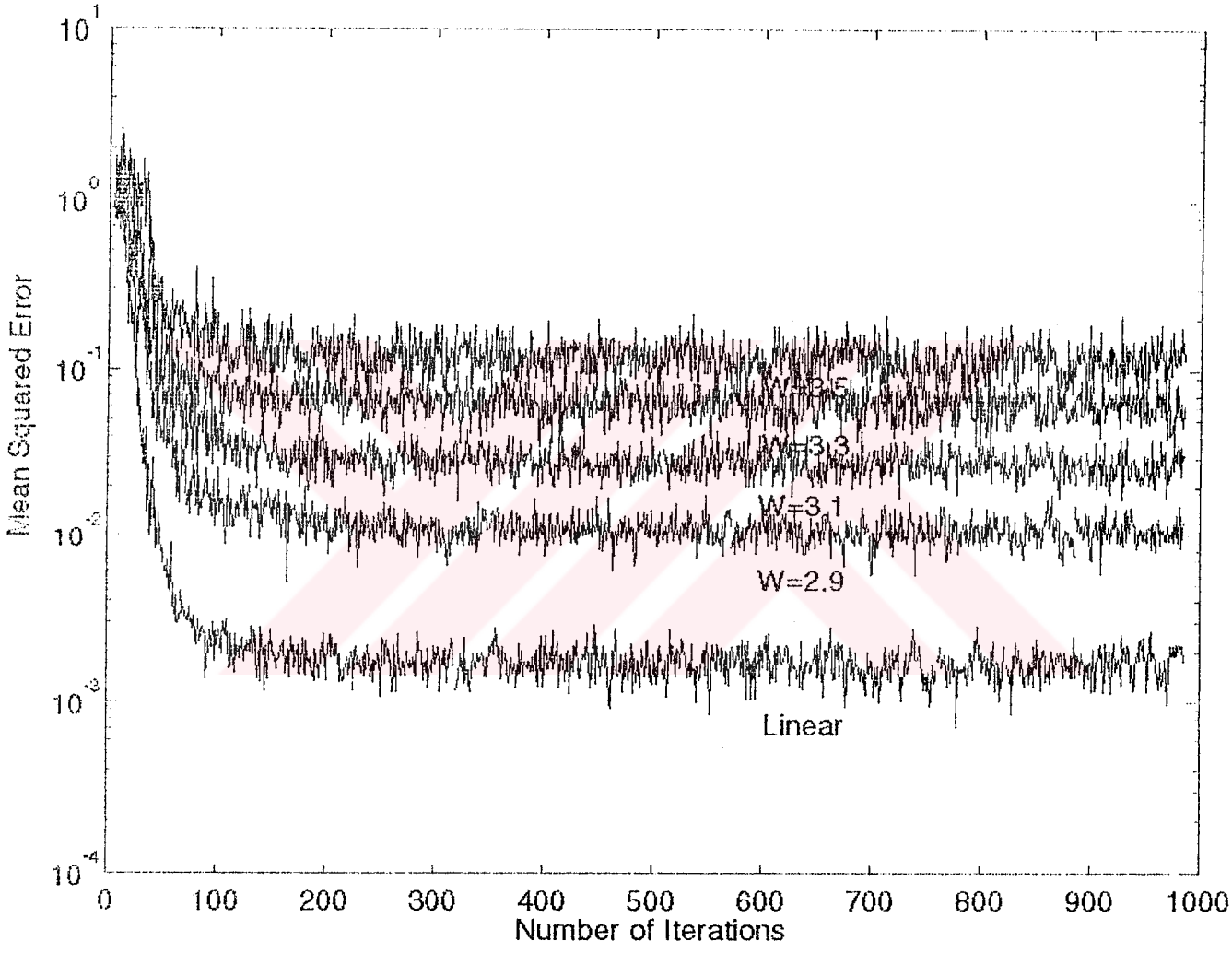
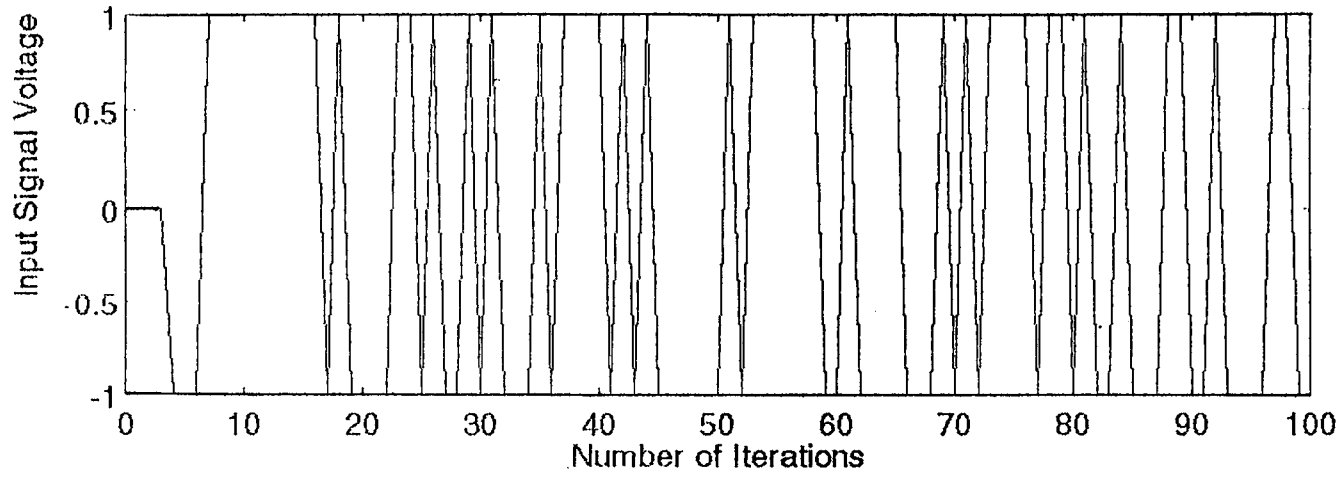
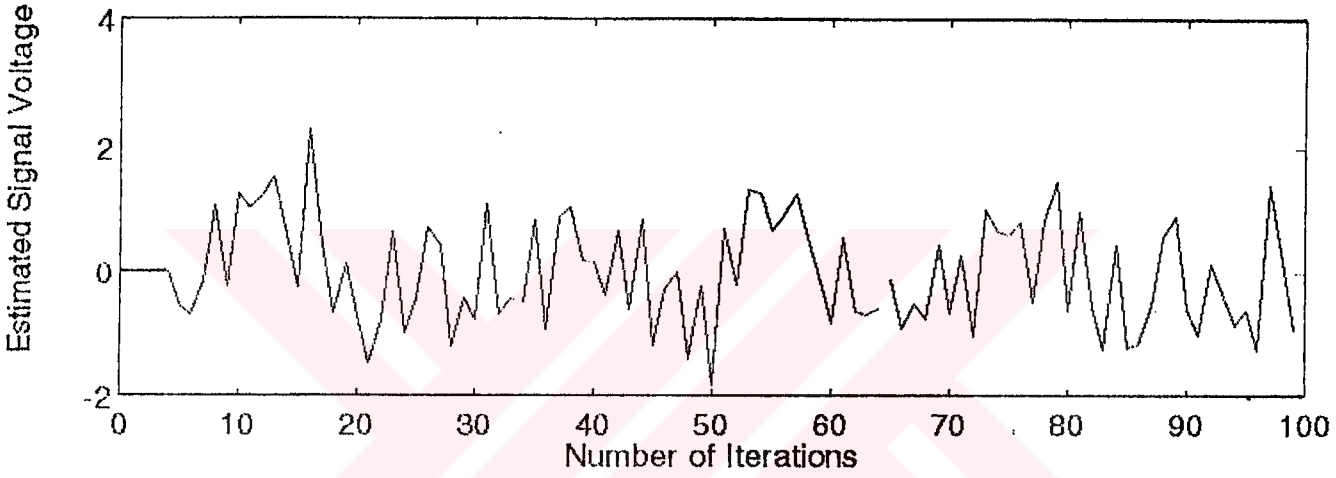


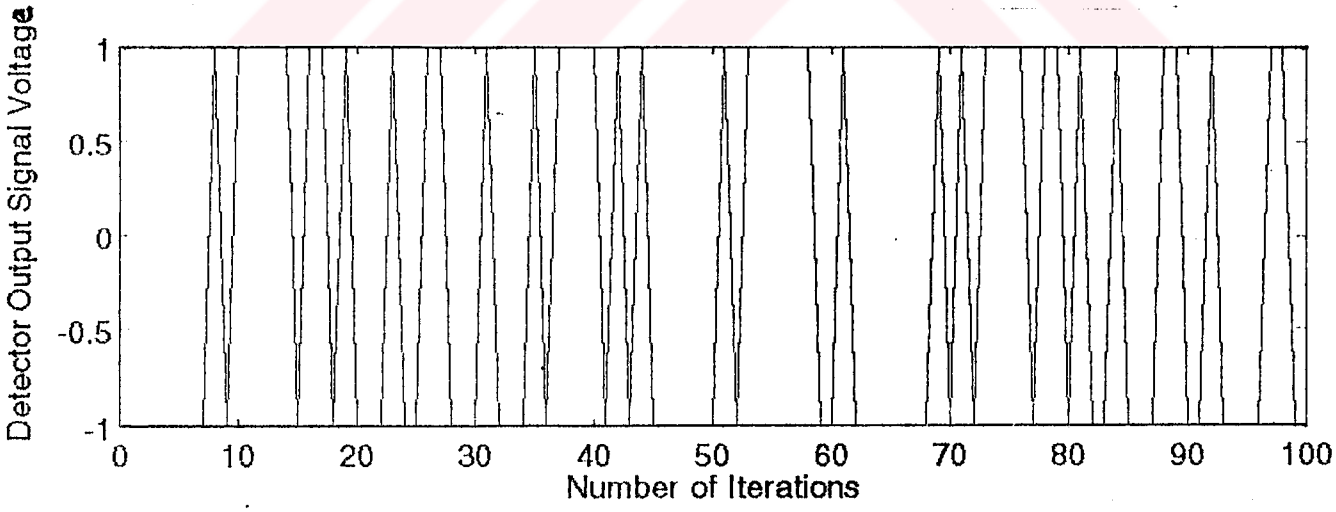
Figure 4.3: Learning curves for second-order Volterra channel equalization with $N = 5$



(a)



(b)



(c)

Figure 4.4: The input (a), the estimated (b) and the threshold detector output (c) signals

Table 4.1: THE NUMBER OF ITERATIONS NEEDED FOR EQUALIZATION

λ	W	Noise variance	τ
0.995	2.9	0.001	27.0
0.995	3.1	0.001	31.9
0.995	3.3	0.001	51.2
0.995	3.5	0.001	78.0
0.995	2.9	0.100	79.3

The number of iterations, τ, needed before the adaptive equalizer makes no error was determined for two different measurement noise variances and different channel distortions. First, the measurement noise variance was constant but, the channel distortion was varied. Then, the channel distortion was constant but, the measurement noise variance was varied. In both cases, the exponential weighting factor was 0.995. Notice that τ gets large either channel distortion or measurement noise variance increases. The results are averages of 10 independent runs and summarized in Table 4.1.

CHAPTER 5

EQUALIZATION AND IDENTIFICATION OF DIGITAL SATELLITE CHANNELS

It has been long recognized that PSK/TDMA provides highly efficient use of power and bandwidth, for digital satellite communication, through the sharing of a single transponder by several earth stations accessing it [1]. The increasing demand for digital satellite services requires the use of new frequency bands together with economical earth stations. In such a band-limited, nonlinear channel, a critical factor influencing the performance is the choice of transmit and receive filters, which must be selected in order to minimize the nonlinear distortions.

Satellite channels differ from voiceband channels in the sense that the nonlinearity is stronger for satellites, but the memory is shorter. In this chapter, digital satellite channels which are modelled by higher-order Volterra series are examined. In the following sections, first digital satellite channels modeled with higher-order Volterra series are described, then RLS higher-order adaptive Volterra identification and equalization filters using SPMLS's are introduced. The results of experiments for each case are also included at each section.

5.1 DIGITAL SATELLITE CHANNELS

The input-output relationship for satellite channels which are modelled by higher-order Volterra series can be expressed as [1,3]

$$y(n) = h_0 + \sum_{m=0}^{\infty} \sum_{k_1} \cdots \sum_{k_{2m+1}} h_{2m+1}(n-k_1,\ldots,n-k_{2m+1})x(k_1)\cdots x(k_1)x^*(k_2)\cdots x^*(k_{2m+1}) \qquad (5.1)$$

The structure of (5.1) reflects how the channel output depends both on the channel (through the Volterra coefficients) and on the information symbols. Particularly, the symbol structure of phase shift keying (PSK) modulation is insensitive to certain kinds of nonlinearities. In phase shift keying, the complex baseband equivalent input signal

is represented by

$$x(n) = e^{j\psi_n} \tag{5.2}$$

where ψ_n takes values in the set

$$\left\{\frac{2\pi}{M}(i-1)+\phi\right\}_{i=1}^{M}$$

and ϕ is an arbitrary constant phase. In binary coherent phase shift keying, where $M = 2$, since $x(n)x^*(n) = 1$, it is apparent from (5.1) that certain Volterra coefficients h_{2m+1} will contribute to nonlinearities of order less than $2m+1$. To be more specific, if the third-order Volterra coefficients $h_3(n-k_1, n-k_2, n-k_3)$ are considered for $k_1 = k_3$ or $k_2 = k_3$, the channel nonlinearities reflected by these coefficients do not affect the PSK signal, because

$$x(k_1)x(k_2)x^*(k_3)h_3(n-k_1, n-k_2, n-k_3) = \begin{cases} x(k_2)h_3(n-k_1, n-k_2, n-k_3), & \text{if } k_1 = k_3 \\ x(k_1)h_3(n-k_1, n-k_2, n-k_3), & \text{if } k_2 = k_3 \end{cases} \tag{5.3}$$

and the only contribution is to the linear part of the channel. Similar considerations on the higher-order coefficients show that some of them contribute to the linear part, others only to the third-order nonlinearity, and so on. These considerations can be further pursued if it is observed that, for an M-ary coherent phase shift keying (PSK), $x^M(n) = 1$ (for instance in 4-PSK, $x(n)x(n)x^*(n)x^*(n) = 1$), which results in a further reduction of sensitivity of PSK to certain nonlinearities. This leads to the noteworthy conclusion that, for PSK, certain coefficients need only linear compansation, while others affect the signal to a lower degree than other modulation schemes. The overall effect is a further reduction of the number of Volterra coefficients to be taken into account in the channel model.

In this chapter, an example of modelling a 4-PSK nonlinear satellite channel using Volterra series is considered. The computed Volterra coefficients for a PSK channel, after deletion of the smallest (the Volterra coefficients less than 0.001 for linear part and 0.005 for nonlinear parts are neglected), are shown in Table 5.1 [1]. After a further reduction that takes into account the structure of 4-PSK modulation, the surviving coefficients are shown in Table 5.2 [1]. It is seen that the size of the reduction is relevant.

5.2 SYSTEM MODEL AND PARAMETERS

In this section, the LS identification of digital satellite channels modelled with higher-order Volterra series is investigated. A procedure similar to the identification of second-order Volterra systems will be followed for the reduced Volterra channel given in the previous section. The desired signal $d(k)$ to the adaptive filter as before is modelled by

$$d(k) = y(k) + e_o(k) \tag{5.4}$$

where $y(k)$ is the unknown system output and, $e_o(k)$ is the measurement noise. An 4-PSK signal, $x(k)$, is used to probe the channel. Both $d(k)$ and $x(k)$ are assumed to be wide-stationary with zero means.

The problem is to find an exponentially windowed, LS solution for the linear and higher-order nonlinear coefficients of the adaptive filter that minimizes the cost function,

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - d_n(k) \mid^2 \tag{5.5}$$

at each time instant n. The filter output $d_n(k)$ is a higher-order Volterra filter estimate of the desired signal $d(k)$ and,

$$\begin{aligned} d_n(k) = h_0 + \sum_{n_1=0}^{3} \hat{a}_{n_1}(n)\, x(k-n_1) + \\ \sum_{n_1}\sum_{n_2}\sum_{n_3} \hat{b}_{n_1,n_2,n_3}(n)\, x(k-n_1)x(k-n_2)x^*(k-n_3) \\ + \hat{c}_{0,0,0,1,1}(n)\, x(k)x(k)x(k)x^*(k-1)x^*(k-1) \end{aligned} \tag{5.6}$$

where $\{\hat{a}_{n_1}\}$ and $\{\hat{b}_{n_1,n_2,n_3}\}$, $\{\hat{c}_{0,0,0,1,1}\}$ are called the linear and higher-order filter weights. The unbiased filter output is obtained by having the following relationship between h_0 and the higher-order filter weights,

$$h_0 = -\sum_{n_1}\sum_{n_2}\sum_{n_3} \hat{b}_{n_1,n_2,n_3}(n)\, r_{xxx}(n_1-n_2-n_3) - \hat{c}_{0,0,0,1,1}(n)\, r_{xxxxx}(0) \tag{5.7}$$

where $r_{xxx}(n_1-n_2-n_3) = E[x(n-n_1)x(n-n_2)x^*(n-n_3]$ and $r_{xxxxx}(0) = E[x(n)x(n)x(n)$ $x^*(n-1)x^*(n-1)]$ respectively. By combining (5.4) and (5.5), the filter output is expressed as

$$\begin{aligned} d_n(k) = \sum_{n_1=0}^{3} \hat{a}_{n_1}(n)x(k-n_1) + \\ \sum_{n_1}\sum_{n_2}\sum_{n_3} \hat{b}_{n_1,n_2,n_3}(n)\, [x(k-n_1)x(k-n_2)x^*(k-n_3) - r_{xxx}(n_1-n_2-n_3)] + \\ \hat{c}_{0,0,0,1,1}(n)\, [x(k)x(k)x(k)x^*(k-1)x^*(k-1) \\ -r_{xxxxx}(0)]. \end{aligned} \tag{5.8}$$

Table 5.1: VOLTERRA COEFFICIENTS FOR A PSK CHANNEL

Linear part	
$h_1(0) = 3.400 + j0.381$	$h_5(0,0,0,0,0) = 3.920 - j2.210$
$h_1(1) = 0.052 + j0.006$	$h_5(1,0,0,0,0) = -0.690 + j0.388$
$h_1(2) = -0.048 - j0.005$	$h_5(1,0,0,1,0) = 0.236 - j0.133$
$h_1(3) = 0.178 + j0.020$	$h_5(1,1,0,1,0) = 0.070 - j0.066$
$h_3(0,0,0) = -4.296 + j1.741$	$h_5(1,1,0,1,1) = 0.074 - j0.022$
$h_3(1,0,0) = 0.388 - j0.137$	$h_5(1,1,1,1,1) = 0.039 - j0.022$
$h_3(1,0,1) = -0.230 + j0.081$	$h_5(2,0,0,0,0) = 0.059 - j0.033$
$h_3(1,1,1) = -0.105 + j0.037$	$h_5(2,2,2,2,2) = -0.053 + j0.030$
$h_3(2,0,0) = -0.056 + j0.020$	$h_5(3,0,0,0,0) = 0.349 - j0.197$
$h_3(2,2,2) = 0.074 - j0.026$	$h_5(3,0,0,3,0) = 0.118 - j0.066$
$h_3(3,0,0) = -0.384 + j0.136$	$h_7(0,0,0,0,0,0,0) = -1.140 + j0.764$
$h_3(3,0,3) = -0.033 + j0.029$	$h_7(1,0,0,0,0,0,0) = 0.309 - j0.207$
	$h_7(1,0,0,0,1,0,0) = -0.106 + j0.072$
	$h_7(3,0,0,0,1,0,0) = -0.107 + j0.072$
	$h_7(3,0,0,0,3,0,0) = -0.043 + j0.029$
Third-order nonlinearities	
$h_3(0,0,1) = 0.194 - j0.068$	$h_5(0,0,0,3,0) = 0.233 - j0.131$
$h_3(0,0,3) = -0.192 + j0.068$	$h_5(1,1,0,0,0) = 0.118 - j0.066$
$h_3(1,1,0) = -0.115 + j0.041$	$h_5(1,1,1,1,0) = 0.049 - j0.028$
$h_3(3,3,0) = -0.041 + j0.015$	$h_5(3,3,0,0,0) = 0.059 - j0.033$
$h_5(0,0,0,1,0) = -0.460 + j0.259$	$h_7(0,0,0,0,1,0,0) = 0.231 - j0.156$
$h_5(0,0,0,2,0) = 0.039 - j0.022$	$h_7(0,0,0,0,3,0,0) = -0.081 + j0.054$
	$h_7(1,1,0,0,0,0,0) = -0.053 + j0.036$
Fifth-order nonlinearities	
$h_5(0,0,0,1,1) = 0.039 - j0.022$	

Table 5.2: REDUCED VOLTERRA COEFFICIENTS FOR A PSK CHANNEL

Linear part
$h_1(0) = 1.220 + j0.646$
$h_1(1) = 0.063 - j0.001$
$h_1(2) = -0.024 - j0.014$
$h_1(3) = 0.036 + j0.031$
Third-order nonlinearities
$h_3(0,0,2) = 0.039 - j0.022$
$h_3(3,3,0) = 0.018 - j0.018$
$h_3(0,0,1) = -0.035 + j0.035$
$h_3(0,0,3) = -0.040 - j0.009$
$h_5(1,1,0) = -0.001 - j0.017$
Fifth-order nonlinearities
$h_5(0,0,0,1,1) = 0.039 - j0.022$

The input vector $\mathbf{X}(n)$ and the coefficient vector $\mathbf{W}(n)$, at time n are defined as

$$\begin{aligned}\mathbf{X}(n) = [&x(n), x(n)x(n)x^*(n-2), x(n-1), x(n-3)x(n-3)x^*(n),\\ &x(n)x(n)x^*(n-1), x(n)x(n)x^*(n-3), x(n-1)x(n-1)x^*(n),\\ &x(n)x(n)x(n)x^*(n-1)x^*(n-1),\\ &x(n-2), x(n-3)]^T \end{aligned} \tag{5.9}$$

and

$$\begin{aligned}\mathbf{W}(n) = [&\hat{a}_0(n), \hat{b}_{0,0,2}(n), \hat{a}_1(n), \hat{b}_{3,3,0}(n), \hat{b}_{0,0,1}(n),\\ &\hat{b}_{0,0,1}(n), \hat{b}_{0,0,3}(n), \hat{b}_{1,1,0}(n), \hat{c}_{0,0,0,1,1}(n), \hat{a}_2(n), \hat{a}_3(n)]^T \end{aligned} \tag{5.10}$$

respectively. In the above "T" represents transpose of a matrix. Thus, the main concern of the exponentially weighted LS problem under consideration is to find, at each time n, the optimal coefficient vector $\mathbf{W}(n)$ that would minimize the cost function

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \mid d(k) - \mathbf{W}^H(n)\mathbf{X}(k) \mid^2 \tag{5.11}$$

where the superscript "H" denotes Hermitan transpose of a matrix as in Chapter 2.

5.2.1 RLS ADAPTIVE VOLTERRA SATELLITE CHANNEL IDENTIFICATION USING SPMLS

The approach for developing a lattice structure for Volterra satellite channel identification is to transform the higher-order nonlinear channel identification problem into an equivalent multichannel, but linear, adaptive channel identification problem. The basic idea is to partition the input vector $\mathbf{X}(n)$ at time n into the following set of smaller vectors,

$$\begin{aligned}
CH1: &\quad [x(n), x(n-1), x(n-2), x(n-3)] \\
CH2: &\quad [x(n)x(n)x^*(n-2)] \\
CH3: &\quad [x(n-3)x(n-3)x^*(n)] \\
CH4: &\quad [x(n)x(n)x^*(n-1)] \\
CH5: &\quad [x(n)x(n)x^*(n-3)] \\
CH6: &\quad [x(n-1)x(n-1)x^*(n)] \\
CH7: &\quad [x(n)x(n)x(n)x^*(n-1)x^*(n-1)]
\end{aligned} \tag{5.12}$$

so that each of the vectors can be considered as being formed from successive samples of signals from a different input channel. To avoid matrix inversion and vector operations and to achieve simplicity, good numerical properties and modularity, a complete, modified Gram-Schmidt orthogonalization of the input data $\mathbf{X}(n)$ is obtained using SPMLS.

The main objective in this section is to apply sequential processing lattice stages into the Volterra satellite channel identification problem. A block diagram of the Volterra satellite channel identification problem using SPMLS's is shown in Figure 5.1. The complex, a posteriori and indirect form of SPMLS algorithm is used in the Volterra satellite channel identification. In complex, a posteriori, indirect form, processing equations for cells represented as double circles are,

$$r(n) = \lambda\, r(n-1) + |s_r(n)|^2 / \gamma_i(n) \tag{5.13}$$

$$\gamma_o(n) = \gamma_i(n) - |s_r(n)|^2 / r(n) \tag{5.14}$$

where $s_r(n)$ is the input signal to the cell, r(n) is the LS autocorrelation of the input signal, $\gamma_i(n)$ is the input likelihood variable, λ is the exponential weighting factor and $\gamma_o(n)$ is the output likelihood variable, all at time n. Processing equations for cells

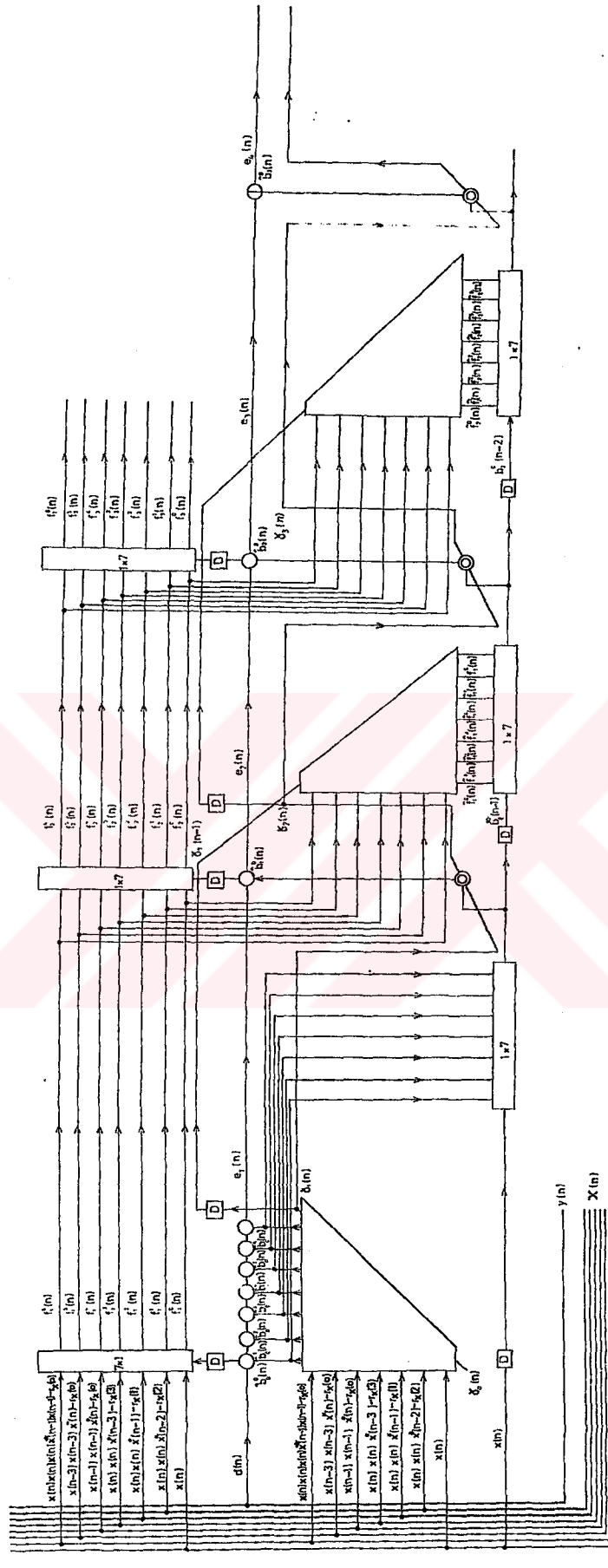


Figure 5.1: Block diagram of the Volterra satellite channel identification with SPMLS

represented as circles are,

$$\Delta(n) = \lambda\Delta(n-1) + s_i^*(n)s_r(n)/\gamma_i(n) \tag{5.15}$$

$$k(n) = \Delta(n)/r(n) \tag{5.16}$$

$$s_o(n) = s_i(n) - k^*(n)s_r(n) \tag{5.17}$$

where $s_i(n)$ and $s_r(n)$ are the input signal and the reference input signal to the cell respectively, $\Delta(n)$ is the LS crosscorrelation between the input signals, $s_o(n)$ is the output signal from the cell, all at time n.

The basic idea employed is to obtain a modified Gram-Schmidt orthogonalization of $\mathbf{X}(n)$ sequentially in such a way that an orthogonal basis set corresponding to $\mathbf{X}(n)$,

$$\hat{\mathbf{b}}(n) = [\hat{b}_0^0(n), \hat{b}_0^1(n), \hat{b}_0^2(n), \hat{b}_0^3(n), \hat{b}_0^4(n), \hat{b}_0^5(n), \hat{b}_0^6(n), \hat{b}_1^0(n), \hat{b}_2^0(n), \hat{b}_3^0(n)] \tag{5.18}$$

is obtained, so that the desired signal $d(n)$ can be estimated as a linear combination of the orthogonal basis set $\hat{\mathbf{b}}(n)$, instead of the elements of $\mathbf{X}(n)$. To obtain the new orthogonal set sequentially, the elements of $\mathbf{X}(n)$ are grouped into three coloumns of a matrix as shown in (5.19).

$$\begin{bmatrix} x(n) & x(n-1) & x(n-2) & x(n-3) \\ x(n)x(n)x^*(n-2) & & & \\ x(n-3)x(n-3)x^*(n) & & & \\ x(n)x(n)x^*(n-1) & & & \\ x(n)x(n)x^*(n-3) & & & \\ x(n-1)x(n-1)x^*(n) & & & \\ x(n)x(n)x(n)x^*(n-1)x^*(n-1) & & & \end{bmatrix} \tag{5.19}$$

Each row in this matrix may be thought of as made up of samples of a signal belonging to a different channel in (5.12). Initially, the elements of the first coloumn are orthogonalized with a SOP and the new orthogonal signals, called backward prediction error (BPE) signals, $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$, $\hat{b}_0^2(n)$, $\hat{b}_0^3(n)$,$\hat{b}_0^4(n)$, $\hat{b}_0^5(n)$,$\hat{b}_0^6(n)$ are then obtained. The second coloumn, $x(n-1)$, is predicted from $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$, $\hat{b}_0^2(n)$, $\hat{b}_0^3(n)$,$\hat{b}_0^4(n)$, $\hat{b}_0^5(n)$,$\hat{b}_0^6(n)$ and the desired signal is estimated from $\hat{b}_0^0(n)$, $\hat{b}_0^1(n)$, $\hat{b}_0^2(n)$, $\hat{b}_0^3(n)$,$\hat{b}_0^4(n)$, $\hat{b}_0^5(n)$,$\hat{b}_0^6(n)$. Thus, the BPE signal, $b_1^0(n)$ and the first stage estimation error signal $e_1(n)$ are generated. Similarly, the elements of the first coloumn are predicted from $\hat{b}_0^0(n-1)$ and the forward prediction errors (FPE) signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n), f_1^4(n), f_1^5(n), f_1^6(n)$ are obtained.

Following these steps, the first stage FPE signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n), f_1^4(n), f_1^5(n), f_1^6(n)$ are predicted from $\hat{b}_1^0(n-1)$. The first stage estimation error signal $e_1(n)$ is estimated from $\hat{b}_1^0(n)$ and then the second stage estimation error signal $e_2(n)$ is obtained. The FPE error signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n), f_1^4(n), f_1^5(n), f_1^6(n)$ are similarly orthogonalized using a SOP. The delayed first stage backward prediction error signal, $b_1^0(n-1)$ are predicted from the orthogonolized FPE signals $f_1^0(n), f_1^1(n), f_1^2(n), f_1^3(n), f_1^4(n), f_1^5(n), f_1^6(n)$ and the second stage BPE signal $b_2^0(n)$ is obtained. $e_2(n)$ is estimated with the new BPE signal $\hat{b}_2^0(n)$ and so on.

The computational complexity of the satellite channel identification filter is larger than the second-order case for the same channel length N, due to higher order Volterra components.

5.2.2 EXPERIMENTAL RESULTS

Several experiments were conducted to show the performance of the satellite channel identification filter. In the experiments, the exponential weighting factor is 0.995, the input signal $x(n)$ is a 4-PSK signal defined as

$$x(n) = e^{j\psi_n} \tag{5.20}$$

where ψ_n takes values from the set

$$\left\{\frac{2\pi}{4}(i-1)+\phi\right\}_{i=1}^{4} \tag{5.21}$$

and ϕ is an arbitrary constant phase. If $\phi = 0$, then ψ_n takes values from the set $\left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}$. So, $x(n)$ takes values from the set

$$\left\{1, e^{j\pi/2}, e^{j\pi}, e^{j3\pi/2}\right\}. \tag{5.22}$$

The measurement noise signal is an additive white, zero-mean, pseudo-Gaussian, uncorrelated with the input signal. The variance of measurement noise is 0.001. The autocorrelation values, $r_{xxx}(n_1 - n_2 - n_3)$ and $r_{xxxxx}(0)$, for the estimation of the zeroth-order term h_o are computed by time-averaging the input vector components $x(n-n_1)x(n-n_2)x^*(n-n_3)$ and $x(n)x(n)x(n)x^*(n-1)x^*(n-1)$ respectively. The computed values of $r_{xxx}(n_1-n_2-n_3)$ and $r_{xxxxx}(0)$ are then subtracted from the input vector components, $x(n-n_1)x(n-n_2)x^*(n-n_3)$ and $x(n)x(n)x(n)x^*(n-1)x^*(n-1)$ at time n, before the first SPMLS. The results presented are ensemble averages of 50 independent runs. Performance evaluations were carried out by plotting the mean-squared

Table 5.3: ESTIMATED LINEAR AND NONLINEAR WEIGHTS OF THE 4-PSK CHANNEL

Linear part
$\hat{a}_0 = 1.220 + j0.64445$
$\hat{a}_1 = 0.063 - j0.008$
$\hat{a}_2 = -0.0243 - j0.0145$
$\hat{a}_3 = 0.0368 + j0.0316$
Third-order nonlinearities
$\hat{b}_{0,0,2} = 0.04 - j0.0228$
$\hat{b}_{3,3,0} = 0.0179 - j0.0177$
$\hat{b}_{0,0,1} = -0.0348 + j0.0352$
$\hat{b}_{0,0,3} = -0.0406 - j0.0093$
$\hat{b}_{1,1,0} = 0.0376 - j0.038$
Fifth-order nonlinearities
$\hat{c}_{0,0,0,1,1} = 0.0188 - j0.0193$

value of the a posteriori estimation error, in Figure 5.2 and the estimated coefficients of the channel, in Table 5.3. Note that the filter has a very good convergence speed and there is bias in some coefficients due to the choice of exponential weighting factor and measurement noise.

5.3 RLS ADAPTIVE VOLTERRA SATELLITE CHANNEL EQUALIZATION USING SPMLS

In this section, SPMLS'S are applied into the digital satellite channel equalization problem. The channel is modeled as a 4-PSK with the reduced Volterra coefficients in Table 5.2. A procedure similar to the equalization of second-order Volterra channels will be followed.

To obtain the multichannel input signal to the adaptive equalizer, the input vector at time n is partitioned as follows

$$CH1: \quad [y(n), y(n-1), \ldots, y(n-N-2)]$$

$$CH2: \quad [y(n)y(n)y^*(n-2), \ldots, y(n-N+1)y(n-N+1)y^*(n-N-1)]$$

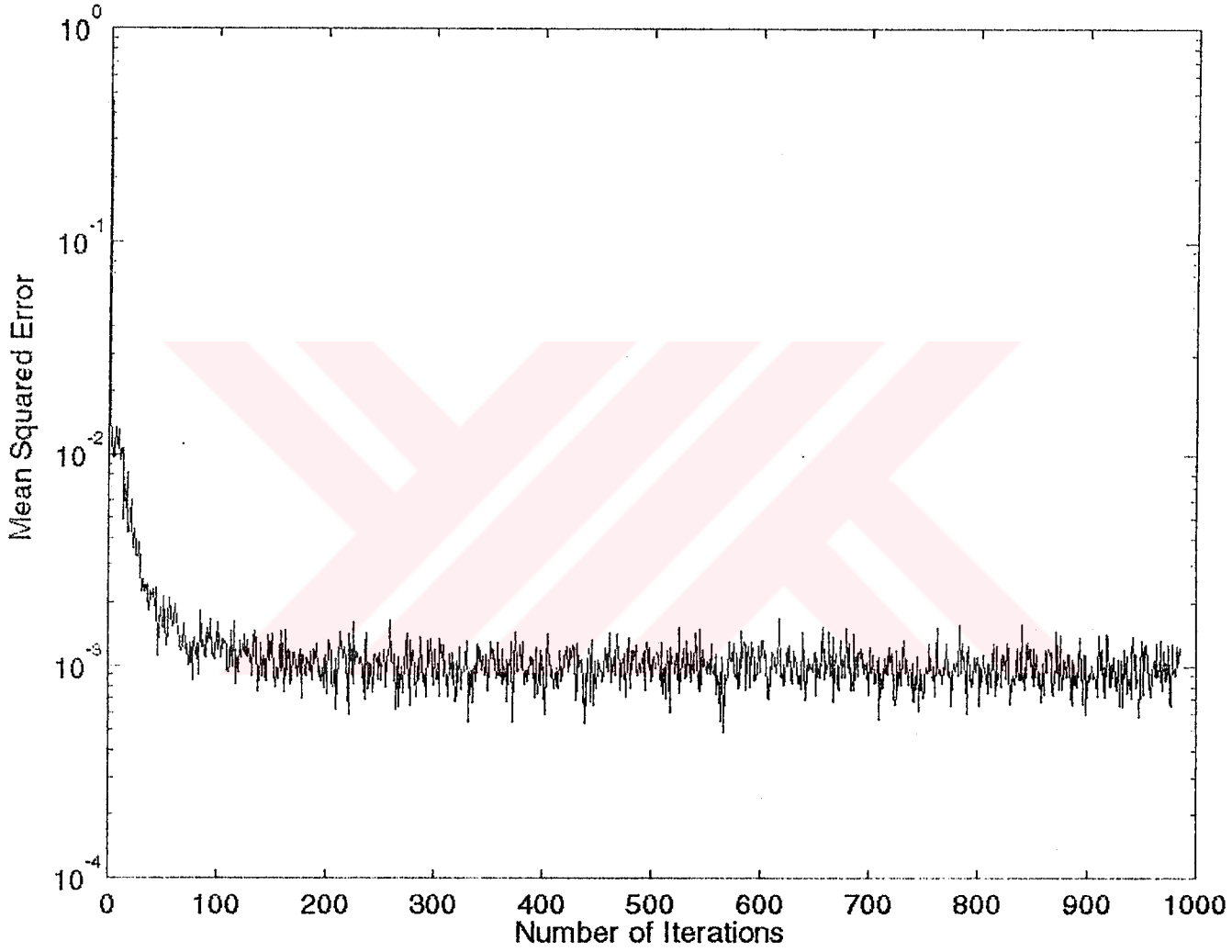


Figure 5.2: Learning curve for the adaptive Volterra satellite channel identification filter.

$$
\begin{aligned}
CH3: &\ [y(n-3)y(n-3)y^*(n), \ldots, y(n-N-2)y(n-N-2)y^*(n-N+1)] \\
CH4: &\ [y(n)y(n)y^*(n-1), \ldots, y(n-N+1)y(n-N+1)y^*(n-N-1)] \\
CH5: &\ [y(n)y(n)y^*(n-3), \ldots, y(n-N+1)y(n-N+1)y^*(n-N-2)] \\
CH6: &\ [y(n-1)y(n-1)y^*(n), \ldots, y(n-N)y(n-N)y^*(n-N+1)] \\
CH7: &\ [y(n)y(n)y(n)y^*(n-1)y^*(n-1), \ldots, \\
&\ y(n-N+1)y(n-N+1)y(n-N+1)y^*(n-N)y^*(n-N)]
\end{aligned} \tag{5.23}
$$

so that each of the vectors can be considered as being formed from successive samples of signals from a different input channel.

To avoid matrix inversion and vector operations and to achieve simplicity, good numerical properties and modularity, a complete, modified Gram-Schmidt orthogonalization of the input data is obtained using SPMLS.

A block diagram of the Volterra satellite channel equalization problem using SPMLS's for $N = 2$ is shown in Appendix B. The complex, a priori and direct form of SPMLS algorithm is used in the Volterra satellite channel equalization. In complex, a priori, direct form, processing equations for cells represented as double circles are,

$$r(n) = \lambda\, r(n-1) + \gamma_i(n)\,|s_r(n)|^2 \tag{5.24}$$

$$\gamma_o(n) = \gamma_i(n) - \gamma_i^2(n)\,|s_r(n)|^2\,/r(n) \tag{5.25}$$

where $s_r(n)$ is the input signal to the cell, r(n) is the LS autocorrelation of the input signal, $\gamma_i(n)$ is the input likelihood variable, λ is the exponential weighting factor and $\gamma_o(n)$ is the output likelihood variable, all at time n. Processing equations for cells represented as circles are,

$$s_o(n) = s_i(n) - k^*(n-1)s_r(n) \tag{5.26}$$

$$\Delta(n) = \Delta(n-1) + \gamma_i(n)s_o^*(n)s_r(n)/r(n) \tag{5.27}$$

where $s_i(n)$ and $s_r(n)$ are the input signal and the reference input signal to the cell respectively, $\Delta(n)$ is the LS crosscorrelation between the input signals, $s_o(n)$ is the output signal from the cell, all at time n.

The computational complexity of the Volterra satellite channel equalization filter using SPMLS is larger than the second-order case for the same channel length N, due to higher order Volterra components.

5.3.1 EXPERIMENTAL RESULTS

Several experiments were conducted to show the performance of the satellite channel equalization filter. In the experiments, the exponential weighting factor is 0.995, the input signal $x(n)$ to the satellite channel is the same 4-PSK signal used in the previous section. The measurement noise signal is an additive white, zero-mean, pseudo-Gaussian, uncorrelated with the input signal. The variance of measurement noise is 0.001. The autocorrelation values, $r_{yyy}(n_1-n_2-n_3)$ and $r_{yyyyy}(0)$, for the estimation of the zeroth-order term h_o are computed by time-averaging the input vector components $y(n-n_1)y(n-n_2)y^*(n-n_3)$ and $y(n)y(n)y(n)y^*(n-1)y^*(n-1)$ respectively. The computed values of $r_{yyy}(n_1-n_2-n_3)$ and $r_{yyyyy}(0)$ are then subtracted from the input vector components, $y(n-n_1)y(n-n_2)y^*(n-n_3)$ and $y(n)y(n)y(n)y^*(n-1)y^*(n-1)$ at time n, before the first SPMLS. The results presented are ensemble averages of 50 independent runs. Performance evaluations were carried out by plotting the mean-squared value of the a priori estimation error, in Figure 5.4. The desired signal was obtained by delaying the input signal suitably so that the smallest mean squared error for the equalizer is obtained. The delay parameter for the smallest mean squared error was four. Notice that the higher-order Volterra equalizer filter converges slower comparing to the second-order Volterra equalizer in Chapter 4.

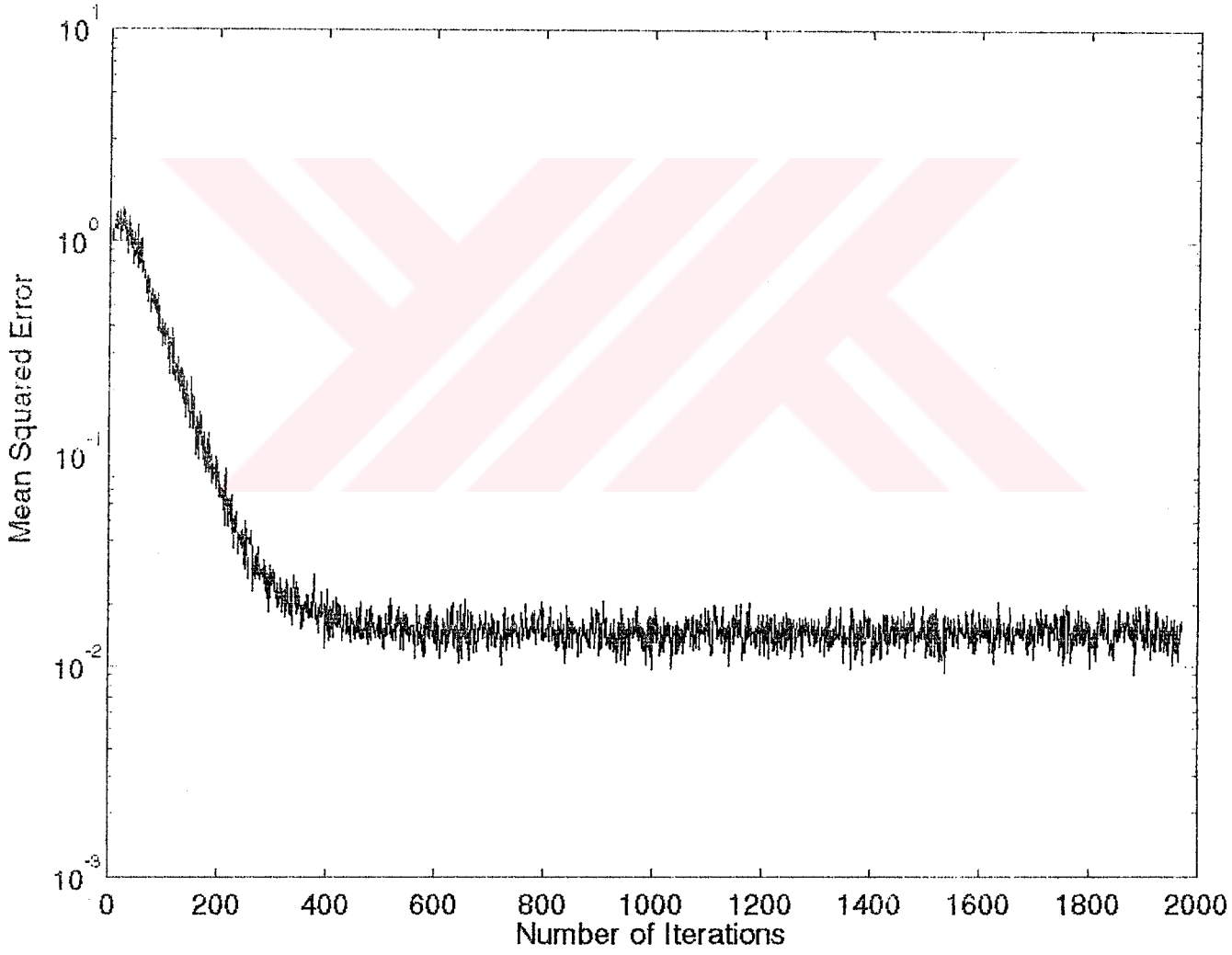


Figure 5.4: Learning curve for the adaptive Volterra satellite channel equalization filter.

CHAPTER 6

CONCLUSIONS

In this thesis, a new multichannel lattice structure for nonlinear systems modeled with Volterra series representation is presented. The structure is different from most previously published Volterra structures in that it completely orthogonalizes the multichannel input signal vector resulting in a less complex, very modular and suitable for VLSI implementations. It is also applicable to arbitrary input signal and planes of support of the Volterra kernels extending to higher order nonlinearities.

In Chapter 2, adaptive filtering algorithms employing truncated Volterra series representation of nonlinear systems were considered. Section 2.3 and 2.4 presented the LMS and RLS adaptive second-order Volterra filtering algorithms respectively. In section 2.5, the benefits of lattice structures in Volterra filtering applications were mentioned, the Gram-Schmidth orthogonalization algorithm was reviewed and the new lattice structure, sequential processing multichannel lattice stages, were introduced. The nonlinear filtering problem was transformed into an equivalent multichannel, but linear, adaptive filtering problem. The implementation of the modified Gram-Schmidth orthogonalization algorithm with SPMLS's were discussed. Different forms of the SPMLS algorithm were given.

Chapter 3 presents a system identification application of the RLS filter based on SPMLS. In section 3.1, the general LS adaptive second-order Volterra system identification problem was described. Then, section 3.2 presented the RLS adaptive second-order Volterra system identification with SPMLS's. In section 3.3, the results of several experiments showing the performance of the new lattice filter was investigated. In these experiments, the adaptive filters were run in with the same structure and the same number of coefficients as that of the system that was to be identified. It was observed that the algorithm shared the fast convergence property of all RLS algorithms and showed even better performance than RLS transversal Volterra filters without suffering numerical instability.

In Chapter 4, the equalization of the Volterra type of nonlinear channels were considered. Section 4.1 presented the general LS adaptive second-order Volterra channel equalization problem. In section 4.2, the nonlinear equalization problem was tranformed into the equivalent multichannel, but linear, adaptive equalization problem. Then, the RLS second-order Volterra channel equalization problem with SPMLS's were introduced. Section 4.3 was concerned with the experimental results showing the performance of the equalizer for different channel distortions.

The final contributions appeared in Chapter 5, in which the equalization and identification of digital satellite channels were investigated. These types of channels have such a nonlinear structure that they can be modeled with higher order Volterra series. The structure of the Volterra filter was so defined that it can identify or equalize higher-order Volterra channels. The complex forms of SPMLS algorithm were used due to complex channel coefficients and complex channel input signal. The performance of the higher-order Volterra filter was tested for the identification and the equalization of a 4-PSK nonlinear channel. It was observed that the learning curve for the equalizer converged to steady state later, comparing to the second-order Volterra equalizer.

The algorithm was not tested under finite precision conditions, but one could expect that the numerical properties of the nonlinear filters using SPMLS algorithm would be similar to their linear counterparts and the analysis of the numerical properties of such filters [36] have shown that they indeed possess good finite precision characteristics.

As the approach in this thesis achieves the desired modularity and suitability for VLSI implementation, the estimation of the order of memory, the order of nonlinearity and the effects of order and model mismatch issues should also be investigated, in the future research.

REFERENCES

[1] BENEDETTO, S. and BIGLIERI, E., "Nonlinear equalization of digital satellite channels," *IEEE J. Selected Areas on Communications*, Vol. SAC-1, pp. 57-62, 1983.

[2] BELLAFEMINA, M. and BENEDETTO, S., "Identification and equalization of nonlinear channels for digital transmission," *Proc. IEEE Int. Symp. Circuits and Systems*, Kyoto, Japan, pp. 1477-1480, June 1985.

[3] BENEDETTO, S., BIGLIERI, E., and CASTELLINI, V., *Digital Transmission Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[4] BIGLIERI, E., GERSHO, A., GITLIN, R.D. and LIM, T.L., "Adaptive cancellation of nonlinear, intersymbol interference for voiceband data transmission," *IEEE J. Selected Areas on Communications*, Vol. SAC-2, pp. 765-77, 1984.

[5] ÖZGÜNEL, S., *Equalization and identification of Volterra type of nonlinear channels using multichannel adaptive lattice algorithms*, Ph.D. dissertation, Istanbul Technical University, Istanbul, Turkey, October 1992.

[6] AGAZZI,O., MESSERSCHMITT, D.G. and HODGES, D.A., "Nonlinear echo cancellation of data signals," *IEEE Trans. Communications*, Vol. COM-30, pp. 2421-2433, November 1982.

[7] CASAR-CORREDERA, J.R., GARCIA-OTERA, M. and FIGUEIRAS-VIDAL, A.R., "Data echo nonlinear cancellation," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Tampa, Florida, pp. 32.4.1-4, March 1985.

[8] SICURANZA, G.L., BUCCONI, A. and MITRI, P.,"Adaptive echo cancellation with nonlinear digital filters," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, San Diego, California, pp. 3.10.1-4, March 1984.

[9] SMITH, M.J., COWAN, C.F.N. and ADAMS, P.F.,"Nonlinear echo cancellers based on transpose distributed arithmetic," *IEEE Transactions on Circuit and Systems*, Vol. CAS-35, No.1, pp. 6-18, January 1988.

[10] BENEDETTO, S., BIGLIERI, E. and DAFFARA, R.,"Performance of multilevel baseband digital systems in nonlinear environment," *IEEE Trans. Communications*, Vol. COM-24, pp. 1166-1175, 1976.

[11] BENEDETTO, S., BIGLIERI, E. and DAFFARA, R., "Modeling and performance evaluation of nonlinear satellite links - A Volterra series approach," *IEEE Trans. Aerospace and Electronic Systems*, Vol. AES-15, pp. 494-507, 1979.

[12] KIM, T.L. and OMURA, J.K., "Error-rate estimates in digital communication over a nonlinear channel with memory," *IEEE Transactions in Communications*, Vol.COM-31, No.3, pp. 407-412, March 1983.

[13] MAQUSI, M., "Performance of baseband digital data transmission in nonlinear channels with memory," *IEEE Transactions on Communications*, Vol. COM-33, No.7, pp. 715-719, July 1985.

[14] COKER, M.J. and SIMKINS, D.N., "A nonlinear adaptive noise canceller," *Proceedings of the 1980 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 470-473, 1980.

[15] STAPLETON, J.C. and BASS, S.C., "Adaptive noise cancellation for a class of nonlinear, dynamic reference channels," *IEEE Transactions on Circuits and Systems*, Vol. CAS-32, No.2, pp. 143-150, February 1985.

[16] KENEFIC, R.J. and WEINER, D.D. "Application of the Volterra functional expansion in the detection of nonlinear functions of Gaussian processes," *IEEE Transactions on Communications*, Vol.COM-33, No.3, pp.276-279, March 1985.

[17] HUNTER, I.W. and KORENBERG, M.J., "The identification of nonlinear biological systems: Weiner Hammerstein cascade models," *Biological Cybernetics*, Vol.55, pp. 135-144, 1986.

[18] KORENBERG, M.J. and HUNTER, I.W., "The identification of nonlinear biological systems: LNL cascade models," *Biological Cybernetics*, Vol.55, pp. 125-134, 1986.

[19] JACOBSEN, S.C., MEEK, S.G. and FULMER, R.R., "An adaptive myoelectric filter," *6th IEEE Conf.Eng. in Med.and Biol. Soc.*, 1984.

[20] JAVED, A., SYRETT, B.A. and GOUD, P.A., "Intermodulation distortion analysis of reflection-type IMPATT amplifiers using Volterra series representation," *IEEE Transactions on Microwave Thery and Techniques*, Vol. MTT-25 No.9, pp. 729-733, September 1977.

[21] NARAYANAN, S., "Transistor distortion analysis using Volterra series representation," *Bell Systems Technical Journal*, Vol.46, pp. 991-1204, May-June 1967.

[22] RAMBONI, G. and SICURANZA, G.L.,"Decision-directed nonlinear filters for image processing," *Electronic letters*, Vol. 23, No. 23, pp. 1218-1219, November 5, 1987.

[23] KOH, T., POWERS, E.J. and MIKSAD, R.W., "An approach to time-domain modeling of nonlinear drift oscillations in random seas," *Proceedings of the International Symposium on Offshore Engineering*, Rio de Janeiro, Brazil, 1983.

[24] KOH, T. and POWERS, E.J. , "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASPP-33, No.6, pp. 1445-1455, December 1985.

[25] BOVIK, A.C., HUANG, T.S. and MUNSON, D.C., "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, Signal Proc.*, Vol. ASPP-31, No.6, pp. 1342-1349, December 1983.

[26] LEE, Y.H. and KASSAM, S.A., "Generalized median filtering and related nonlinear filtering technique," *IEEE Trans. Acoust., Speech, Signal Proc.*, Vol. ASPP-33, No.3, pp. 672-683, June 1985.

[27] NODES, T.A. and GALLARHER, N.C., "Median filters: Some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Proc.*, Vol. ASPP-30, No.5, pp. 739-746, October 1982.

[28] BAIK, H.K. and MATHEWS, V.J.,"Adaptive Bilinear Filters," *IEEE Trans. Sig. Proc.*, vol.41, no. 6, pp. 2033-2046, June 1993.

[29] COKER, M.J. and SIMKINS, D.N., "A nonlinear adaptive noise canceller," in *Proc. 1980 IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 470-473, 1980.

[30] ÖZGÜNEL, S., KAYRAN, A.H. and PANAYIRCI, E., "Nonlinear channel equalization using multichannel adaptive lattice algorithms," *Proc. IEEE Int. Symp. on Circuits and Systems, Singapore*, pp. 2826-2829, June 1991.

[31] ÖZGÜNEL, S., KAYRAN, A.H. and PANAYIRCI, E., "Nonlinear channel equalization and identification," *Proc. Int. Conf. on Digital Signal Processing, Florence*, pp. 260-265,

[32] LEE, J. and MATHEWS, V.J., "A fast recursive least squares adaptive second-order Volterra filter and its performance analysis," *IEEE Trans. Sig.Proc.*, vol. 41, no. 3, pp. 1087-1101, March 1993.

[33] SYED, M.A. and MATHEWS, V.J., "Lattice algorithms for recursive least squares adaptive second-order volterra filtering," *IEEE Trans. Circuits Syst.-II: Analog and Digital Sig. Proc.*, vol. 41, no. 3, March 1994.

[34] SYED, M.A. and MATHEWS, V.J., "QR-Decomposition based algorithms for adaptive Volterra filtering," *IEEE Trans. Circuits Syst.-I: Fundamental Theory and Applications*, vol. 40, no. 6, June 1993.

[35] YANG, B. and B"OHME, J.F., "Rotation-based RLS algorithms: Unified derivations, numerical properties, and parallel implementations" *IEEE Trans. Sig. Proc.*, vol. 40, no. 5, May 1992.

[36] LEWIS, P.S., "QR-based algorithms for multichannel adaptive least squares lattice filters," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. 38, no.3, pp. 421-432, March 1990.

[37] LING, F. and PROAKIS, J.G., "A generalized multichannel least squares lattice algorithm based on sequential processing stages," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-32, no.2, pp. 381-389, April 1984.

[38] LING, F., et al., "A recursive modified Gram-Schmidt algorithm for least-squares estimation," *IEEE Trans. Acoust.,Speech, Sig. Proc.*, vol. ASSP-34, no.4, pp. 829-835, August 1986.

[39] SCHETZEN, M.,"Nonlinear system modelling based on the Wiener theory," *Proc. IEEE*, vol. 69, no. 12, pp. 1557-1573, December 1981.

[40] BOYD, S., TANG, Y.S. and CHUA, L.O., " Measuring volterra kernels," *IEEE Trans. Circuits Syst.*, vol. CAS-30, no. 8, pp. 571-577, August 1983.

[41] HAYKIN, S., *Adaptive filter theory*, Prentice Hall, second ed., Englewood Cliffs, New Jersey, 1991.

[42] MATHEWS, V.J., "Adaptive polynomial filters," *IEEE Signal Proc. Mag.*, vol.8, no.3 pp. 10-26, July 1991.

[43] ELEFTHERIOU, E. and FALCONER, D.D., "Tracking properties and steady-state performance of RLS adaptive filter algorithms," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no.5, pp. 1097 - 1110, October 1986.

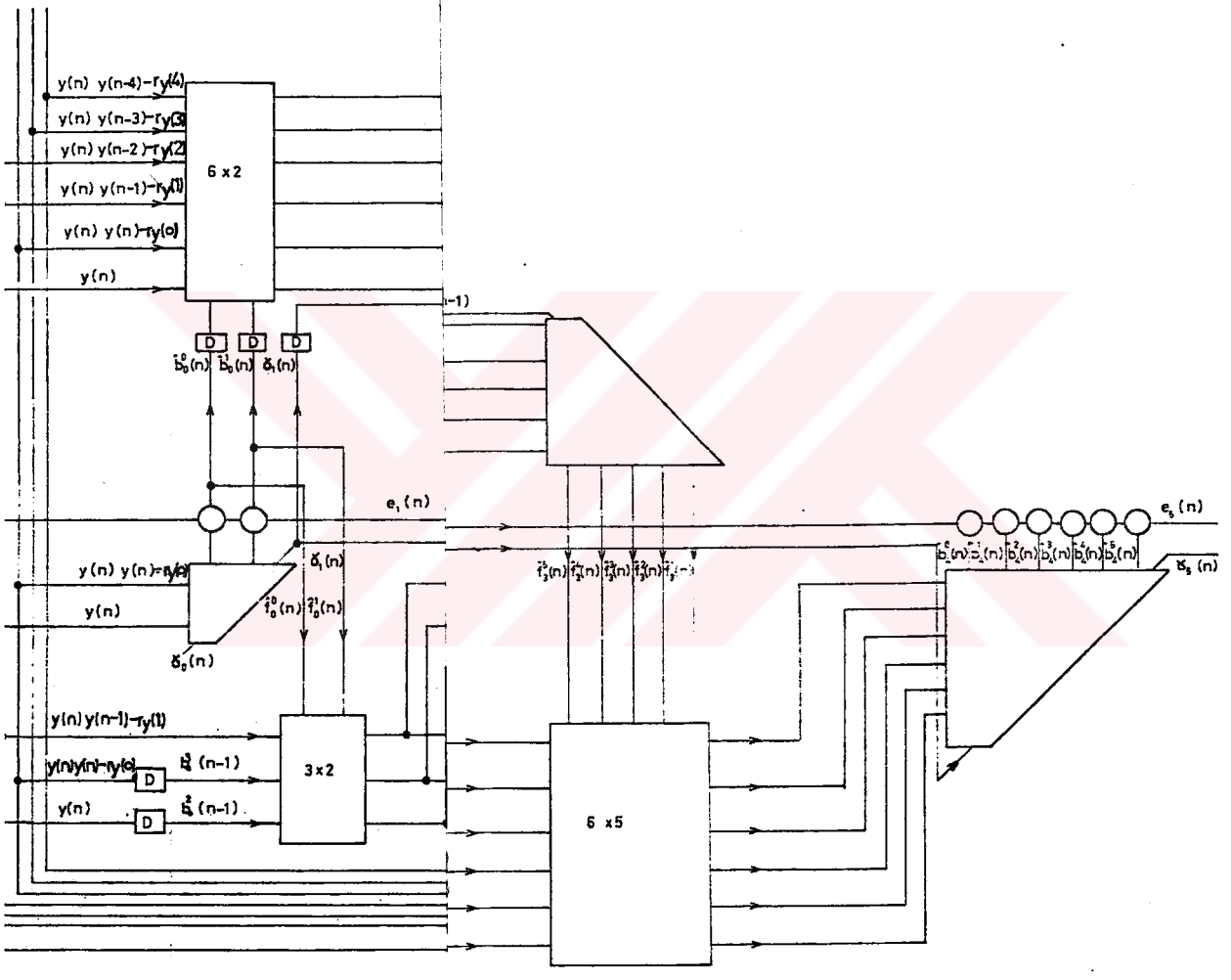
[44] FRIEDLANDER, B., "Lattice filters for adaptive processing," *Proceedings IEEE*, Vol. 70, no.8, August 1982.

[45] LING, F., et al., "Numerically robust least squares lattice-ladder algorithms with direct updating of the reflection coefficients," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 4, pp. 837-845, August 1986.

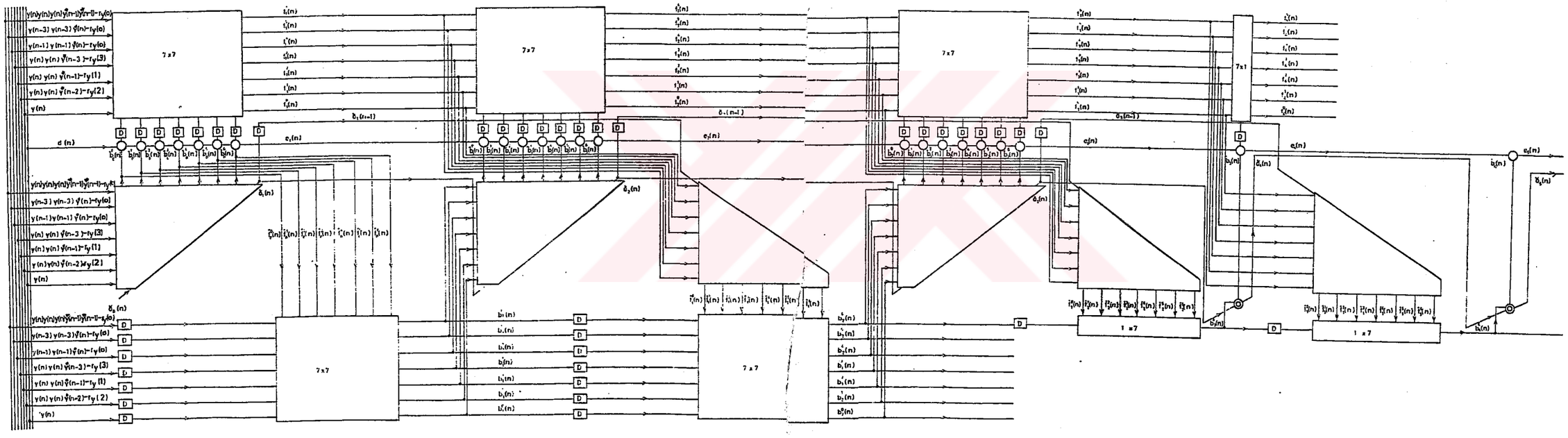
[46] LING, F., "Givens rotation-based least squares lattice and related algorithms," *IEEE Trans. Sig. Proc.*, vol. 39, no. 7, July 1991.

APPENDICES

APPENDIX A


y(n) y(n-4)–r_y(4)
y(n) y(n-3)–r_y(3)
y(n) y(n-2)–r_y(2)
y(n) y(n-1)–r_y(1)
y(n) y(n)–r_y(0)
y(n)
6 x 2
D
D
D
e_1(n)
e_5(n)
y(n) y(n)–r_y(0)
y(n)
3x2
y(n)y(n-1)–r_y(1)
y(n)y(n)–r_y(0)
y(n)
6 x5

APPENDIX B



Block diagram of the Volterra satellite channel equalization with SPMLS's.

BIOGRAPHY

Mehmet Tahir Özden was born in Tokat, in 1962. He was graduated from Deniz Lisesi, Istanbul. He received the B.S. degree in 1984 from Deniz Harp Okulu, Istanbul and the M.S. degree in 1990 from Naval Postgraduate School, Monterey, California, U.S.A.