NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

TENSOR FORMULATIONS FOR THE MODELLING OF DISCRETE-TIME NONLINEAR AND MULTIDIMENSIONAL SYSTEMS

by

Peter John Lenk

September 1985

Thesis Advisor: S.R. Parker

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

Tensor Formulations for the Modelling of Discrete-Time Nonlinear and Multidimensional Systems

5. TYPE OF REPORT & PERIOD COVERED

Ph.D. Dissertation
September 1985

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

Peter John Lenk

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Naval Postgraduate School
Monterey, California, 93943-5100

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Postgraduate School
Monterey, California, 93943-5100

12. REPORT DATE

September 1985

13. NUMBER OF PAGES

196

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Nonlinear system, nonlinear system modelling, Volterra series, tensor form of Volterra series, alternate coordinate systems, moving average, autoregressive, RLS algorithm, LMS algorithm, multidimensional system modelling, generalized lattice models, Levinson algorithm, Schur algorithm, 2-D lattice models, nonlinear lattice models, systolic arrays.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The modelling of nonlinear and multidimensional systems from input and or output measurements is considered. Tensor concepts are used to reformulate old results and develop several new ones. These results are verified through non-trivial computer simulations.

A generalized tensor formulation for the modelling of discrete-time stationary nonlinear systems is presented. Tensor equivalents of the normal equations are derived and several efficient methods for their solution are discussed. Conditions are established that ensure a diagonal correlation tensor so that a solution can be obtained directly without matrix inversion.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Using a tensor formulation, a new proof of the Generalized Lattice Theory is obtained. Tensor extensions of the Levinson and Schur algorithms are presented.

New two-dimensional (2-D) lattice parameter models are derived. Using the tensor form of the Generalized Lattice Theory the 2-D multi-point error order-updates are decomposed into $O(N^2)$ single point updates. 2-D extensions of the Levinson and Schur algorithms are given. The quarter plane lattice is considered in detail, first in a general form, then in forms which reduce the computational complexity by assuming shift-invariance.

Based on the 2-D lattice, a new nonlinear lattice model is developed. The model is capable of updates in the nonlinear as well as time order.

Approved for public release; distribution is unlimited

Tensor Formulations for the Modelling of Discrete-Time Nonlinear and Multidimensional Systems

by

Peter John Lenk
Lieutenant(N), Canadian Armed Forces
B.Eng., Royal Military College of Canada, 1978

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

The modelling of nonlinear and multidimensional systems from input and/or output measurements is considered. Tensor concepts are used to reformulate old results and develop several new ones. These results are verified through non-trivial computer simulations.

A generalized tensor formulation for the modelling of discrete-time stationary nonlinear systems is presented. Tensor equivalents of the normal equations are derived and several efficient methods for their solution are discussed. Conditions are established that ensure a diagonal correlation tensor so that a solution can be obtained directly without matrix inversion.

Using a tensor formulation, a new proof of the Generalized Lattice Theory is obtained. Tensor extensions of the Levinson and Schur algorithms are presented.

New two-dimensional (2-D) lattice parameter models are derived. Using the tensor form of the Generalized Lattice Theory the 2-D multi-point error order-updates are decomposed into $O(N^2)$ single point updates. 2-D extensions of the Levinson and Schur algorithms are given. The quarter plane lattice is considered in detail, first in a general form, then in forms which reduce the computational complexity by assuming shift-invariance.

Based on the 2-D lattice, a new nonlinear lattice model is developed. The model is capable of updates in the nonlinear as well as time order.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I owe a great debt to Dr. S.R. Parker. Without his constant encouragement and remarkable insight this work would not have been completed. I take this opportunity to acknowledge this debt and to thank him.

I would also like to thank the members of my Doctoral Commitee. In particular, I wish to express my gratitude to Professor E.C. Crittenden for his friendship and support, to Professor P.H. Moose for his friendship and the countless hours which he spent clarifying many of my misconceptions, and to Professor Ziomek for his careful reading of this thesis.

Doctors Srbijanka Turajlic and Bharat Madan have both added much to my understanding of Electrical Engineering. I would be remiss if I did not mention their names.

Professor R.D. Strum has also contributed substantially to my education. However, more importantly, he gave me his friendship.

I would also like to acknowledge Capt(N) J. Dean for his support of my proposal to remain in Monterey and persue my Doctorate. In today's modern Navy a requirement for Officers trained to the Doctorate level exists and I hope that this training will be made available to others in the future.

Finally. I must thank my wife, Kim, and daughters, Kaarina. Alexis. Teegan. and Tess. who endured much while I studied. Without this stable home environment I believe this thesis would not have been possible.

I. INTRODUCTION

Linear, time(shift)-invariant systems have been exhaustively studied and their properties and behaviour are well known. These systems form the foundation of all engineering and scientific disciplines. However, they represent only an approximation of reality. This fact, of course, does not diminish their utility. Mathematical models employing the assumptions of linearity and shift-invariance often provide results sufficiently accurate to be of practical use. For a large class of systems, however, these assumptions cannot be justified and so alternate models must be used. Mathematical models capable of representing nonlinearities and methods for their identification from system measurements are the major topics explored in this thesis. Due to the particular treatment of nonlinear models chosen, multidimensional linear system modelling is also investigated. The assumption of shift-invariance will not be relaxed.

It will be useful to formulate a geometric framework in which to solve the nonlinear modelling problem. The motivation for this is simple. The transition from physical problems to geometric ones allows many diverse phenomena to be handled with a common set of mathematical tools. This relieves the burden of having to invent new mathematics for each new situation. Instead, the well understood language of geometry is used to tackle many classes of problems. Kron [Ref. 1: p. 197] states very clearly the rationale that allows the real physical problem to be converted to an equivalent geometric one in the following passage.

> A set of n equations with n variables (with time as a parameter) may represent either the performance of a dynamical system with n degrees of freedom or the motion of a point along a curve located in an n-dimensional hypothetical space and expressed along some frame of reference.

The basic approach taken in this dissertation, with respect to the modelling of nonlinear systems, is to represent them as a linear combination of nonlinear functions of the data. This allows linear algorithms to be applied in the solution of the modelling problem. In the process of solving the nonlinear problem several

new multidimensional lattice structures are developed.

The inputs and outputs of the unknown system will be treated as random signals (not in general gaussian.) The problem of transforming random processes into geometric quantities has two solutions. These are introduced here in order to avoid confusion later and also in part to justify the tensor formulation that is used in the sequel.

A random process **X** may be defined as the assignment of a function $\{x(t.\omega), t\in T\}$, to every outcome, ω in a sample space Ω. Of interest in this dissertation is the case when T is a discrete and finite index set.

One way of geometrically visualizing this random process is to consider a Hilbert (or inner product) space, **S**, (in general infinite dimensional) of random variables. that is the vectors or elements of the space are random variables. Fixing $t=t_0$, $x(t_0,\omega)$ is a random variable and so is a vector in **S**. The random process, **X**, a time series of random variables, is a series of vectors in **S**. or a curve in **S** [Ref. 2: p. 27]. The components of the vectors comprising **X** are indexed by the parameter ω. The required inner product on this space is defined in terms of the statistical correlation, ie; $E\{x(t_1,\omega)x(t_2,\omega)\}$. This approach has proved highly successful in many applications [Ref. 3].

An alternate approach is to consider that the random process X, consists of vectors in a function space. In this interpretation the random process is an ensemble of time functions, $\{x(t,\omega),t\in T\}$, indexed by ω. Each of these time functions (generally referred to as **realizations**) is a vector in the function space. There will exist a large (in general infinite) number of such vectors corresponding to each possible outcome, $\omega\in\Omega$. The components of the vectors are indexed by the parameter t. There is no need to define a metric on this space. Any expectations that are required must be calculated over the ensemble of vectors.

This second approach will be the one that is followed throughout this work. It will lead to many interesting and novel interpretations of known algorithms and also will be used to derive several significant new results.

While vectors are sufficient to provide a complete characterization of discrete-time, one-dimensional linear systems, general nonlinear systems with

memory require the use of higher order **geometric objects** to obtain convenient descriptions. It is shown in this dissertation that a particular class of these **geometric objects** that extend vector concepts in a natural way and provide an ability to deal with nonlinearities in an organized fashion are tensors.

The contraversial **Sapir-Whorf** hypothesis from linguistics [Ref. 4] states that the constructs of a language define **the boundaries of thought.** Mathematics is a legitimate language. It is a well defined set of rules used to communicate ideas. If the mathematics that is employed in the solution of a problem is constrained, then it is conceivable that certain solutions may not be arrived at, or even that the problem may remain unsolved. In Electrical Engineering, vector calculus and linear algebra are the major mathematical tools. They are adequate to explain such diverse phenomena as the propagation of electromagnetic waves or the behaviour of one dimensional linear systems. More complex problems have also been solved using this theory by forcing them to fit, but the notation can become awkward. Tensor analysis is a convenient mathematical framework in which to deal with nonlinear and multidimensional signal processing problems. It provides an algebra for manipulating objects of higher dimension than two, which is all that can easily be handled using linear algebra. Importantly, tensor algebra furnishes a system of notation which is powerful, yet compact.

The Electrical Engineer's experience is usually limited to **ordinary**, Euclidean geometries. Physicists, around the turn of the century, began to realize that other more complex types of geometries were equally valid and important. In fact, Einstein showed that the world we live in is neither Euclidean nor is it simply three dimensional.

The arguments outlined above provide the motivation for this study of the utility of tensor concepts in Electrical Engineering, specifically in the area of discrete signal processing. Although this work examines but a fraction of the possible applications in this field, it proves that tensors can lead to useful results and that they warrant further consideration, particularly in problems involving spaces of higher dimensions.

A. HISTORICAL BACKGROUND

Tensor analysis has evolved in this century, originating with two Italian mathematicians in 1900; Ricci and Levi-Civita. Many of the early contributions are due to Einstein who required tensor concepts in the development of his general theory of relativity. Recent books on the subject include Golab, Synge and Schild, and Young [Ref. 5,6,7].

Kron [Ref. 1] in the early 1930's made use of tensor concepts in Electrical Engineering. He appears to be first to do so. His work was mainly concerned with the analysis and design of electrical networks and rotating machinery. Since that time there have been few papers that deal with tensors in the context of Electrical Engineering.

Volterra [Ref. 8] laid the foundation for nonlinear system analysis in the late nineteenth century. He studied functionals or functions of functions. He proposed a series of increasing order functionals as an approximation to any other functional. Frechet [Ref. 9: p. 517] later showed that this series was a complete representation and converged uniformly. This series has since become known as the Volterra series. We shall study this series in detail in Chapter 3.

The first application of the Volterra series to nonlinear systems was done by Norbert Wiener in the 1930's. Wiener also made several other significant contributions to nonlinear theory, such as the introduction of the Wiener G-functionals [Ref. 10]. They posses the property of orthogonality when the system input is white Gaussian noise. The two theories (Volterra and Wiener) form the basis of almost all significant work to date on nonlinear systems.

One of the first practical methods of system identification was proposed by Lee and Schetzen [Ref. 11]. Their method takes advantage of the orthogonality of the Weiner G-functionals by employing a cross-correlation technique to identify system parameters.

The study of discrete-time nonlinear systems has gained importance with the advent of the digital computer. It appears that the idea of a discrete Volterra series first appeared in the mid 1960's (see for example [Ref. 12].) The use of tensor techniques in the study of nonlinear systems has received little attention.

Sandor and Williamson [Ref. 13] made some use of them in the study of continuous systems. More recently Parker and Thomas [Ref. 14] proposed the idea of using tensor methods for the analysis of nonlinear discrete-time systems. Their techniques for system identification involved the use of deterministic input signals to extract system parameters.

The Volterra series is non-recursive and so a discrete form cannot represent an infinite memory system. This is equivalent to trying to represent an infinite memory linear system with a finite length impulse response. This can pose implementation difficulties for systems with long memories. One possible solution is the use of a recursive model. There has until very recently been little written about this because of the difficulty in analysing system stability. Parker and Perry [Ref. 15] have proposed a discrete nonlinear ARMA (auto-regressive moving-average) model, however, no stability implications were considered. Also Parker, Mayoral and Thomas [Ref. 16] proposed an Adaptive Kalman Identifier or RLS (Recursive Least Square) type algorithm for the identification of non-linear ARMA systems. Zarzycki and Dewilde [Ref. 17] and Zarzycki [Ref. 18,19,20,21] have proposed a nonlinear lattice structure. Again the stability of the resulting models is not discussed. Some nonlinear systems are inherently recursive (eg: the phase locked loop) so that this remains an important area for research.

Recently, several books dealing exclusively with nonlinear systems theory have been published. The book by Schetzen [Ref. 9] concerns itself with continuous systems. It provides a very thorough but readable development of the classical concepts. Also of interest is a short appendix outlining the history of nonlinear systems theory. A book by Rugh [Ref. 22], is an important contribution as it includes discussions of discrete theory.

B. DISSERTATION OVERVIEW

Because the typical reader of this dissertation will not have a background which includes tensor calculus it was felt that a chapter covering some fundamental concepts should be included. This material was considered to be of

central importance to the work that followed so it remains as a chapter rather than being relegated to an appendix. Readers familiar with tensor concepts may wish to skip most of Chapter 2, although, a cursory look is recommended to ensure that the notation is clearly understood.

Chapter 3 begins with a review of the traditional Volterra theory of nonlinear systems. Both continuous and discrete-time systems are discussed. The discrete-time tensor equivalent of the discrete Volterra series is deduced. An alternate nonlinear tensor formulation is presented along with a discussion of its relationship to the Volterra series. This alternate formulation will be used in most of the work that follows.

Next, methods for the identification of model parameters are examined. A tensor equivalent of the normal or Weiner-Hopf equations is formulated. Several recursive in time algorithms are included as examples of the application of traditional linear modelling methods to the nonlinear tensor formulation. Nontrivial numerical simulation results are included.

The advantages of using alternate coordinate systems are then investigated. It is shown that by proper choice of coordinate systems and input signals the identification process can be significantly simplified. The nonlinear tensor formulation is extended to include recursive type models. It is shown that the Yule-Walker equations have a tensor counterpart which can be solved for the model parameters. Several of the new results of this chapter have already been published [Ref. 23].

In Chapter 4 a review of modern lattice theory is presented. Although the results themselves are not new the approach is novel. Tensor concepts are used to derive the lattice filters presented by considering orthogonalizing coordinate transformations. Generalized forms of the Levinson and Schur algorithms are also presented and proven. These important algorithms are well known in linear matrix theory [Ref. 3] and their generalization in tensor form is a significant result.

Chapter 5 breaks new ground by applying the lattice theory of Chapter 4 to the problem of modelling two-dimensional data fields. Simplifications due to an

assumption of shift-invariance are studied. Several different configurations are considered. Simulation results are included to prove the validity of the theory. Some implementational aspects of the algorithms are considered. In particular a systolic array is deduced for one of the two-dimensional lattice algorithms presented. This result demonstrates that the new algortihms are amenable to implementation in dedicated VLSI hardware.

In Chapter 6 a nonlinear lattice is formulated, again based upon the theory presented in Chapter 4 and 5. This is a new result. The lattice structure proposed differs from those of previous researchers in that it is recursive, not only in time order, but also in nonlinear order. For example, one can obtain the optimal cubic model from a knowledge of the optimal quadratic model. Once again nontrivial simulation results are included.

Chapter 7 is a summary of the new results presented in this dissertation. It draws conclusions about these results and outlines some important unanswered questions as possible topics of future research.

Two appendices are included.

Appendix A contains an alternate proof of Theorem 4.4. This proof uses the Hilbert space formulation described in this introduction. It is included for two reasons: first. to illustrate this alternate formulation and second. to provide additional insight into this theorem which forms the foundation of Chapters 5 and 6.

Appendix B contains listings of the FORTRAN programs used in the simulations presented in this thesis.

II. MATHEMATICAL BACKGROUND

This chapter presents a brief overview of the mathematical tools that are used in the dissertation. It begins with a discussion of linear, bilinear and multilinear functionals, and it is shown that these can be represented by tensors. Some customary conventions which simplify notation are introduced, and some useful tensor operations are presented and discussed. This is meant to be an introduction to the subject of tensor analysis. Only those concepts that will be used in the remainder of the dissertation are presented. Some proofs are included to act as examples. Many others are not presented here. since the interested reader can find them in the references [Ref. 5,6,7]. The discussion assumes a thorough knowledge of linear algebra.

A. LINEAR FUNCTIONALS

We say that, $\mathbf{V}$, is a vector space over a field of scalars, F. if the operations of scalar multiplication and vector addition are defined such that the axioms of a vector space are satisfied (see for example [Ref. 24,25].)

Consider a vector space, $\mathbf{V}$, over a field of scalars, F. The elements of $\mathbf{V}$ are called vectors and will be denoted by use of boldface type, viz $\mathbf{T}$. If we restrict ourselves to spaces of finite dimension, N, then we may write a vector $\mathbf{T}$ as an N-tuple of components and denote the vector space by $\mathbf{V}^N$. The components of a vector $\mathbf{T}$ will be denoted by T^λ, where $\lambda = 1....N$. Writing a vector as a set of components implies the existence of a basis. We will denote a basis for $\mathbf{V}^N$ as the set of vectors $A = \{a_1, ... a_N\}$. Thus an arbitrary vector $\mathbf{T}$ in $\mathbf{V}^N$ can be written as a linear combination of these basis vectors, viz..

$$\mathbf{T} \quad \sum_{\lambda=1}^{N} T^\lambda \mathbf{a}_\lambda \tag{2.1}$$

In order to maintain generality it is not necessary to commit to any specific vector space at this point. Likewise we allow the basis to remain arbitrary.

The following definitions and theorems are presented essentially without proof.

1. Definition 2.1: Linear Functional

If $\mathbf{V}^N$ is a vector space over a field of scalars F, then, a linear transformation, **H** (the reason for the boldface will become apparent shortly, (see eqn (2.6))), of $\mathbf{V}^N$ into F, is known as a **linear functional** or **linear form** on $\mathbf{V}^N$. We can indicate this transformation by

$$\mathbf{H}(\mathbf{T}) = c \quad \text{where } c \in F \text{ and } \mathbf{T} \in \mathbf{V}^N \tag{2.2}$$

2. Theorem 2.1

The set of all linear functionals on $\mathbf{V}^N$ forms a vector space of the same dimension as $\mathbf{V}^N$. This space is known as the **dual vector space** and is indicated by $\hat{\mathbf{V}}^N$.

3. Theorem 2.2

If $\mathbf{V}^N$ is a vector space over the field of scalars F, with basis $A = \{\mathbf{a}_1, ..., \mathbf{a}_N\}$, then the set of linear functionals $\hat{A} = \{\mathbf{b}^1, ... \mathbf{b}^N\}$, (defined so that the λ-th functional. $\mathbf{b}^\lambda$, operating on an arbitrary element of $\mathbf{V}^N$, say **T**, yields the λ-th component of **T**, namely T^λ), form a basis for $\hat{\mathbf{V}}^N$. The defining property can be expressed mathematically as

$$\mathbf{b}^\lambda(\mathbf{T}) = T^\lambda \quad \text{for all } \lambda \in 1, ... N \tag{2.3}$$

$$\text{where } \mathbf{T} = \sum_{\lambda=1}^{N} T^\lambda \mathbf{a}_\lambda$$

We call the functional $\mathbf{b}^\lambda$ the λ-th coordinate function since when applied to a vector **T** it yields the λ-th coordinate, namely T^λ. The set of these linear functionals, $\mathbf{b}^\lambda$, $\lambda \in \{1, ... N\}$ comprising $\hat{A}$ is known as the **dual basis** of A.

It is interesting to note that this choice of basis for the dual space leads to the property that

$$\mathbf{b}^\lambda(\mathbf{a}_\mu) = \delta^\lambda{}_\mu = \begin{cases} 1 & \text{when } \lambda = \mu \\ 0 & \text{when } \lambda \neq \mu \end{cases} \tag{2.4}$$

To show this we proceed as follows: from (2.1) it follows that

$$\mathbf{b}^{\lambda}(\mathbf{T}) = \mathbf{b}^{\lambda}\left(\sum_{\mu=1}^{N} T^{\mu}\mathbf{a}_{\mu}\right) \tag{2.5a}$$

$$= \sum_{\mu=1}^{N} \mathbf{b}^{\lambda}\left(T^{\mu}\mathbf{a}_{\mu}\right) \tag{2.5b}$$

Since $\mathbf{b}^{\lambda}$ is a linear functional (2.5b) can be written as,

$$\mathbf{b}^{\lambda}(\mathbf{T}) = \sum_{\mu=1}^{N} T^{\mu}\mathbf{b}^{\lambda}\left(\mathbf{a}_{\mu}\right) \tag{2.5c}$$

However, from (2.3) it is known that $\mathbf{b}^{\lambda}(\mathbf{T}) = T^{\lambda}$. Thus (2.5c) implies (2.4).

The existence of a basis for the dual space implies that any vector (linear functional) $\mathbf{H}$ in $\hat{\mathbf{V}}^{N}$ may be written uniquely as a linear combination of the elements of the dual basis. Therefore, any linear functional can be represented uniquely by an N-tuple of components. Thus

$$\mathbf{H} = \sum_{\lambda=1}^{N} H_{\lambda}\mathbf{b}^{\lambda} \tag{2.6}$$

From (2.6) one can write

$$\mathbf{H}(\mathbf{T}) = \sum_{\lambda=1}^{N} H_{\lambda}\mathbf{b}^{\lambda}(\mathbf{T}) \tag{2.7a}$$

Using (2.3), (2.7a) becomes

$$\mathbf{H}(\mathbf{T}) = \sum_{\lambda=1}^{N} H_{\lambda}T^{\lambda} \tag{2.7b}$$

Alternately, in matrix form, if

$$\mathbf{H} = [H_1\ H_2\ \cdots\ H_N \tag{2.8}$$

$$\mathbf{T} = [T^1\ T^2\ \cdots\ T^N]^T \tag{2.9}$$

then (2.6) can be written as:

$$\mathbf{H}(\mathbf{T}) = \mathbf{HT}. \tag{2.10}$$

We notice that the two vectors, $\mathbf{H}$ and $\mathbf{T}$, are defined relative to different basis and that they belong to different vector spaces. The vector $\mathbf{H}$, defined

according to the dual basis $\hat{A}$ is called a **covariant** vector. The vector **T**, defined according to the **regular** basis, **A**, is known as a **contravariant** vector. As a convention, whenever a **subscript** is used to index the components of a vector it is understood that the vector is being expressed according to the **dual** basis, $\hat{A}$, in the dual vector space $\hat{\mathbf{V}}^N$, and is a **covariant** vector. Similarly, when a **superscripted** index is used the components are assumed to represent a **contravariant** vector in the vector space $\mathbf{V}^N$ according to the **regular** basis **A**.

Equation (2.7b) represents what we normally think of as a vector inner product. We usually do not think of the two vectors as coming from different vector spaces. The reason for this is that in the familiar rectangular cartesian system of coordinates, the regular and dual basis are identical and so there is no need to differentiate between covariant and contravariant vectors. In other systems of coordinates the distinction must be made in order that the relationships have meaning. To perform a vector inner product, one vector must be covariant and the other must be contravariant. For example consider the vector **T** as illustrated in Figure 2.1. It has components $[1 \;\; 3]^T$ with respect to basis $\{\mathbf{e}_1,\mathbf{e}_2\}$ and components $[-1 \;\; 2]^T$ with respect to basis $\{\mathbf{e}_1',\mathbf{e}_2'\}$.

A measure of the length of vector **T** in the rectangular coordinate system, $\{\mathbf{e}_1,\mathbf{e}_2\}$, can be computed from the expression

$$\| \mathbf{T} \| = \left[\sum_{\lambda=1}^{N} T^{\lambda} T^{\lambda} \right]^{1/2} \tag{2.11a}$$

$$= \left[1^2 + 3^2 \right]^{1/2} \tag{2.11b}$$

$$= (10)^{1/2} \tag{2.11c}$$

The answer is correct because in the rectangular Cartesian coordinate system there is no need to distinguish between covariant and contravariant vectors, ie: $T^{\lambda} = T_{\lambda}$. However, an expression similar to (2.11c) in the oblique coordinate system, $\{\mathbf{e}_1',\mathbf{e}_2'\}$, does not yield a measure of the length of the vector.

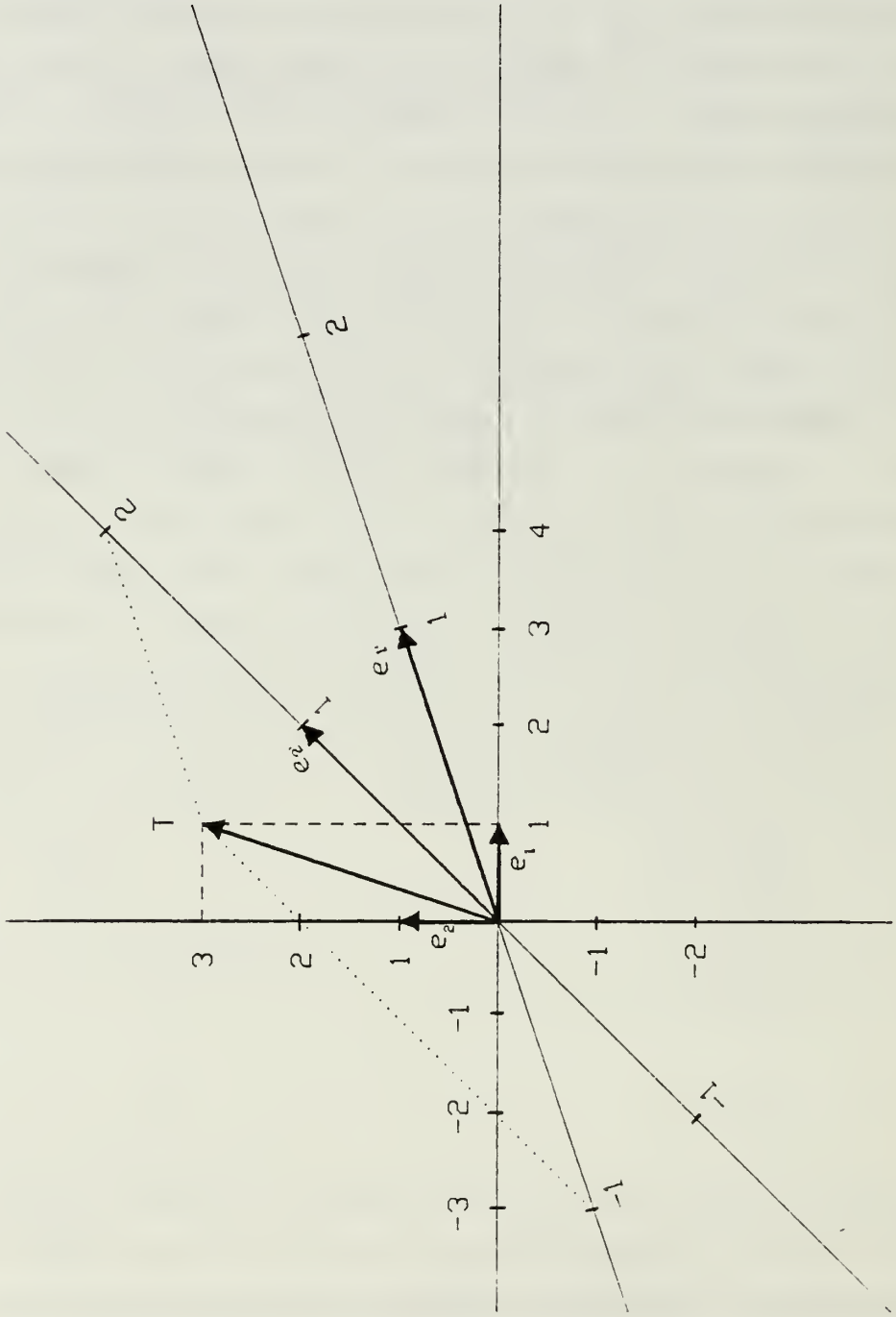


Figure 2.1: Vector **T** expressed according to the two bases $\{\mathbf{e}_1,\mathbf{e}_2\}$ and $\{\mathbf{e}_1',\mathbf{e}_2'\}$.

$$\sum_{\lambda=1}^{N} T^{\lambda}T^{\lambda} = \left[(-1)^2+2^2\right]^{1/2} \tag{2.12a}$$

$$= 5^{1/2} \tag{2.12b}$$

The answer is incorrect since both vectors in expression (2.12) are contravariant. To obtain a correct answer, one of the vectors would have to be made covariant. This involves introduction of a **metric tensor.** The interested reader can find a discussion of this concept in [Ref. 6]. In the case of rectangular coordinate systems this metric tensor is the identity matrix. Our intent here was only to indicate that in any system other than rectangular Cartesian, strict attention must be paid to the character of the vectors.

B. TENSORS

In the previous section the concept of covariant or contravariant vectors has been established. In this section a more formal definition of these quantities is presented.

Suppose we have N variables $x^1, x^2, \ldots, x^N$, then a set of values of these variables is called a point. The variables themselves are called coordinates (or components.) The totality of all points, as each of the variables (coordinates) x^{λ}, $\lambda = 1, \ldots, N$, vary over their entire specified range, constitutes an N-dimensional space, denoted by $\mathbf{V}^N$.

1. Definition 2.2: Contravariant Vector

A contravariant vector **T**, is defined on the basis of the transformation of its components upon transition from one coordinate system to another. For coordinate system (λ) the components of **T** are an N-tuple of numbers designated as

T^{λ} $(\lambda = 1, \ldots, N)$

Upon transition to another coordinate system (λ'), if the components of **T** transform according to the rule

$$T^{\lambda'} = \sum_{\lambda=1}^{N} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} T^{\lambda} \qquad \textbf{(2.13)}$$

where x^{λ} and $x^{\lambda'}$ define the coordinates of a point in the old (λ) and new (λ') coordinate systems, then **T** is said to be a contravariant vector.

2. Definition 2.3: Covariant Vector

A covariant vector **U**, is defined on the basis of the transformation of its components upon transition from one coordinate system to another. For coordinate system (λ) the components of **U** are an N-tuple of numbers designated as

$$U_{\lambda} \quad (\lambda = 1, \ldots, N)$$

Upon transition to another coordinate system (λ'), if the components of **U** transform according to the rule

$$U_{\lambda'} = \sum_{\lambda=1}^{N} \frac{\partial x^{\lambda}}{\partial x^{\lambda'}} U_{\lambda} \qquad (2.14)$$

then **U** is said to be a covariant vector.

In equation (2.13) the quantity $\frac{\partial x^{\lambda'}}{\partial x^{\lambda}}$ represents the partial derivative of the new (primed) coordinates with respect to the old coordinates. Similarly, in equation (2.14) $\frac{\partial x^{\lambda}}{\partial x^{\lambda'}}$ represents the partial derivative of the old coordinates with respect to the new, primed, coordinates. In general these quantities can be arranged into a two-dimensional matrix of numbers.

3. Example 2.1

Consider the following parametric description of a curve:

$$x^1 = f_1(t) \qquad (2.15a)$$

$$x^2 = f_2(t) \qquad (2.15b)$$

$$x^3 = f_3(t) \qquad (2.15c)$$

We consider the three quantities x^1, x^2, x^3, to be the coordinates of a point or equivalently the components of a vector in a three dimensional space. We leave

the basis arbitrary. Indeed the equations we write will be true regardless of the choice of basis. This is the inherent advantage of tensor analysis since it allows expressions to be written which are invariant with respect to the coordinate system.

We can now define new, primed, coordinates as functions of the old coordinates. For example, if we arbitrarily choose the following coordinate transformation

$$x^{1'} = g_1(x^1,x^2,x^3) = \sqrt{\frac{1}{\pi}}x^1 \tag{2.16a}$$

$$x^{2'} = g_2(x^1,x^2,x^3) = \sqrt{\frac{2}{\pi}}x^2 \tag{2.16b}$$

$$x^{3'} = g_3(x^1,x^2.x^3) = \sqrt{\frac{2}{\pi}}(2x^3-x^1) \tag{2.16c}$$

then, the quantities $\frac{\partial x^{\lambda'}}{\partial x^{\lambda}}$ can be written in matrix form as

$$\left[\frac{\partial x^{\lambda'}}{\partial x^{\lambda}}\right] = \begin{bmatrix} \sqrt{\frac{1}{\pi}} & 0 & 0 \\ 0 & \sqrt{\frac{2}{\pi}} & 0 \\ -\sqrt{\frac{2}{\pi}} & 0 & 2\sqrt{\frac{2}{\pi}} \end{bmatrix} \tag{2.17}$$

According to equation 2.13 any contravariant vector, say **T**, with components T^{λ}, can be expressed in the new (primed) coordinate system as

$$T^{\lambda'} = \sum_{\lambda=1}^{N} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} T^{\lambda}$$

or in terms of components

$$T^{1'} = \sqrt{\frac{1}{\pi}}T^1 + 0\cdot T^2 - 0\cdot T^3 = \sqrt{\frac{1}{\pi}}T^1 \tag{2.18a}$$

$$T^{2'} = 0\cdot T^1 + \sqrt{\frac{2}{\pi}}T^2 + 0\cdot T^3 = \sqrt{\frac{2}{\pi}}T^2 \tag{2.18b}$$

$$T^{3'} = -\sqrt{\frac{2}{\pi}}T^1 + 0\cdot T^2 + 2\sqrt{\frac{2}{\pi}}T^3 = \sqrt{\frac{2}{\pi}}\,2(T^3 - T^1) \tag{2.18c}$$

4. Definition 2.4: Contravariant Tensor of Order 2

A set of N^2 numbers $T^{\lambda\mu}$, where λ and $\mu = 1,...,N$ are said to be the components of a second order contravariant tensor if, upon transition to another coordinate system, they transform according to the rule

$$T^{\lambda'\mu'} = \sum_{\lambda=1}^{N}\sum_{\mu=1}^{N} T^{\lambda\mu}\frac{\partial x^{\lambda'}}{\partial x^{\lambda}}\frac{\partial x^{\mu'}}{\partial x^{\mu}} \tag{2.19}$$

5. Definition 2.5: Covariant Tensor of Order 2

A set of N^2 numbers $U_{\lambda\mu}$, where λ and $\mu = 1,....N$ are said to be the components of a second order covariant tensor if, upon transition to another coordinate system, they transform according to the rule

$$U_{\lambda'\mu'} = \sum_{\lambda=1}^{N}\sum_{\mu=1}^{N} U_{\lambda\mu}\frac{\partial x^{\lambda}}{\partial x^{\lambda'}}\frac{\partial x^{\mu}}{\partial x^{\mu'}} \tag{2.20}$$

Similarly, tensors of higher order can also be defined. In the general case, it will no longer be possible to use different letters to denote indices. In this case indices with sub-indices will be utilized, namely $\lambda_1, \lambda_2, ..., \lambda_N$.

6. Definition 2.6: Contravariant Tensor of Order p

A set of N^p numbers $T^{\lambda_1 \cdots \lambda_p}$, where $\lambda_i = 1, ..., N$ for $i = 1,...,p$, are said to be the components of a p-th order contravariant tensor if, upon transition to another coordinate system, they transform according to the rule

$$T^{\lambda_1' \cdots \lambda_p'} = \sum_{\lambda_1=1}^{N} \cdots \sum_{\lambda_p=1}^{N} T^{\lambda_1 \cdots \lambda_p}\frac{\partial x^{\lambda_1'}}{\partial x^{\lambda_1}} \cdots \frac{\partial x^{\lambda_p'}}{\partial x^{\lambda_p}} \tag{2.21}$$

7. Definition 2.7: Covariant Tensor of Order p

A set of N^p numbers $U_{\lambda_1 \cdots \lambda_p}$, where $\lambda_i = 1, ..., N$ for $i = 1,...,p$, are said to be the components of a p-th order covariant tensor if, upon transition to another coordinate system, they transform according to the rule

$$U_{\lambda_1' \cdots \lambda_p'} = \sum_{\lambda_1=1}^{N} \cdots \sum_{\lambda_p=1}^{N} U_{\lambda_1 \cdots \lambda_p}\frac{\partial x^{\lambda_1}}{\partial x^{\lambda_1'}} \cdots \frac{\partial x^{\lambda_p}}{\partial x^{\lambda_p'}} \tag{2.22}$$

8. Definition 2.8: Mixed Tensor of Order p

A set of N^p numbers $S^{\lambda_1 \cdots \lambda_q}{}_{\lambda_{q+1} \quad \lambda_p}$ where $\lambda_i = 1, \ldots, N$ for $i = 1, \ldots, p$, are said to be the components of a p-th order mixed tensor, with q contravariant and (p-q) covariant indices if, upon transition to another coordinate system, they transform according to the rule

$$S^{\lambda'_1 \cdots \lambda'_q}{}_{\lambda'_{q+1} \cdots \lambda'_p}$$

$$= \sum_{\lambda_1=1}^{N} \cdots \sum_{\lambda_p=1}^{N} S^{\lambda_1 \cdots \lambda_q}{}_{\lambda_{q+1} \cdots \lambda_p} \frac{\partial x^{\lambda'_1}}{\partial x^{\lambda_1}} \cdots \frac{\partial x^{\lambda'_q}}{\partial x^{\lambda_q}} \frac{\partial x^{\lambda_{q+1}}}{\partial x^{\lambda'_{q+1}}} \cdots \frac{\partial x^{\lambda_p}}{\partial x^{\lambda'_p}} \tag{2.23}$$

We have already seen two examples of mixed tensors. The first is the Kronecker delta

$$\delta^{\lambda}{}_{\mu} = \begin{cases} 1 & \text{when} \lambda = \mu \\ 0 & \text{when} \lambda \neq \mu \end{cases} \tag{2.24}$$

To see that the Kronecker delta is in fact a mixed tensor we must test to see if it transforms according to the rule given in equation (2.23). We must prove that the following relation is true

$$\delta^{\lambda'}{}_{\mu'} = \sum_{\lambda=1}^{N} \sum_{\mu=1}^{N} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \frac{\partial x^{\mu}}{\partial x^{\mu'}} \delta^{\lambda}{}_{\mu} \tag{2.25}$$

We begin with the right hand side of (2.25)

$$\sum_{\lambda=1}^{N} \sum_{\mu=1}^{N} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \frac{\partial x^{\mu}}{\partial x^{\mu'}} \delta^{\lambda}{}_{\mu} = \sum_{\lambda=1}^{N} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \sum_{\mu=1}^{N} \delta^{\lambda}{}_{\mu} \frac{\partial x^{\mu}}{\partial x^{\mu'}} \tag{2.26a}$$

$$= \sum_{\lambda=1}^{N} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \frac{\partial x^{\lambda}}{\partial x^{\mu'}} \tag{2.26b}$$

$$= \frac{\partial x^{\lambda'}}{\partial x^{\mu'}} \tag{2.26c}$$

$$= \delta^{\lambda'}{}_{\mu'} \tag{2.26d}$$

Therefore, we conclude that relation (2.25) holds and so the Kronecker delta is in fact a second order tensor of mixed character.

The other mixed character tensor that we have already worked with is the one appearing in the formulae of transformation (definitions 2.2 through 2.8), that is the the partial derivative of the new coordinates with respect to the old (and the old with respect to the new). We will not prove that this is in fact a tensor of the type stated although the proof is straight forward. It is instructive to note, however, that the two quantities $\frac{\partial x^{\lambda}}{\partial x^{\lambda'}}$ and $\frac{\partial x^{\lambda'}}{\partial x^{\lambda}}$ are inverses of each other.

As a final note, vectors are tensors of order one. Also scalars are considered to be tensors of order zero. They are sometimes called invariants since their representation is independent of the coordinate systems used.

C. NOTATIONAL CONVENTIONS

There are two widely accepted conventions that simplify notation and unquestionably save much writing. The first is known as the summation convention. Historically, it was first used by Einstein. He noticed that in almost all cases there is really no need to explicitly write summation symbols. Summation can be implied whenever an index is repeated in an expression, once as a superscript and once as a subscript. The repeated index is allowed to take on all permissible values and the resulting terms are summed together. This type of index is often referred to as a dummy index. But what are permissible values for the index? This question leads to the second convention. Normally, the range of the index will not be explicitly stated. By convention it is understood that all greek subscripts and superscripts appearing in an expression will take on all values from 1 to N, where N is the dimension of the vector space in which we are working. In later chapters we will find it more convenient to allow indices to run from 0 to N. The dimensionality of the vector space will thus be N+1. An additional convention which we shall find useful is to reserve latin indices to indicate that we are dealing with a particular component of a quantity. In most books this is indicated by surrounding the particular index with parenthesis. However, we will reserve parenthesis to indicate exponentiation. The conventions adopted here will be used throughout the sequel. In exceptional cases, where

some deviation from them is required, we will explicitly state the meaning of the notation. As an example of the use of these conventions consider the expression

$$Y_\lambda = \sum_{\mu=1}^{N} H_{\lambda\mu} T^\mu \quad \text{for } \lambda = 1, \ldots, N \tag{2.27}$$

It can be written more succinctly as:

$$Y_\lambda = H_{\lambda\mu} T^\mu \tag{2.28}$$

Another convention which has already been used is now formally introduced. Every tensor quantity will be given a distinct base letter. Upon a change of coordinates, the base letter will be maintained in order to indicate that the quantity has not been modified, only the representation has changed. The coordinate system used is indicated by the sub- or superscript used to index the components of the quantity. We therefore, will refer to different coordinate systems simply by the index letter used to indicate the components. For example, a vector **T** has components T^λ in the (λ) set of coordinates, while it has components $T^{\lambda'}$ in the (λ') coordinate system. We note that using this representation, scalars appear identical in all coordinate systems, which is desirable.

1. Example 2.2

The following example serves not only as an illustration of the conventions presented in this section, but also as a concrete (and presumably a somewhat familiar) illustration of the two types of vector. Consider an invariant function of the coordinates, $f = f(x^1, x^2, x^3)$. The differential of this function is given by

$$df = \frac{\partial f}{\partial x^1} dx^1 + \frac{\partial f}{\partial x^2} dx^2 + \frac{\partial f}{\partial x^3} dx^3 \tag{2.29}$$

We can consider dx^λ to be the components of a contravariant vector representing an infinitesimal displacement expressed according to some basis $A = \{a_1, a_2, a_3\}$. We can write this as:

$$\mathbf{dx} = \sum_{\lambda=1}^{N} dx^{\lambda}\mathbf{a}_{\lambda} \tag{2.30}$$

We have already shown that components of a contravariant vector transform according to

$$dx^{\lambda'} = \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} dx^{\lambda} \tag{2.31}$$

The gradient is also a vector whose components are given by:

$$\nabla f_{\lambda} = \frac{\partial f}{\partial x^{\lambda}} \tag{2.32}$$

Upon a change of coordinates the values of the new components, $\nabla f_{\lambda'}$, can be deduced by application of the chain rule;

$$\nabla f_{\lambda'} = \frac{\partial f}{\partial x^{\lambda}} \frac{\partial x^{\lambda}}{\partial x^{\lambda'}} \tag{2.33}$$

It is clear that the components of the gradient transform according to the rule given in equation (2.14). The gradient must, therefore, be considered to be a covariant vector.

2. Example 2.3

Although it has already been stated (section 2.1) that linear functionals can be considered to be covariant vectors, this fact has not been proven. In this example we will show that any linear functional, say **H**, with components H_{λ} that transforms an arbitrary contravariant vector, say **T**, (according to equation (2.7b)) to yield an invariant, satisfies the definition of a covariant vector (equation (2.14)). Equation (2.7b), which defines a vector inner product is repeated here for convenience.

$$\mathbf{H}(\mathbf{T}) = H_{\lambda}T^{\lambda} \tag{2.34a}$$

The quantities T^{λ} are the components of an arbitrary contravariant vector. Because of the assumed invariance we may write

$$H_{\lambda}T^{\lambda} = H_{\lambda'}T^{\lambda} \tag{2.34b}$$

From the definition of a contravariant vector, (equation (2.13))

$$T^{\lambda'} = T^{\lambda} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \tag{2.34c}$$

Equation (2.34b) becomes

$$H_{\lambda} T^{\lambda} = H_{\lambda'} T^{\lambda} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \tag{2.34d}$$

Rearranging yields,

$$(H_{\lambda} - H_{\lambda'} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}}) T^{\lambda} = 0 \tag{2.34e}$$

Equation (2.34e) implies

$$H_{\lambda} = H_{\lambda'} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \tag{2.34f}$$

since the contravariant vector, **T**, was arbitrary. A simple change of variables in this last expression yields

$$H_{\lambda'} = H_{\lambda} \frac{\partial x^{\lambda}}{\partial x^{\lambda'}} \tag{2.34g}$$

which is identical to the equation defining a covariant vector. (2.14). We are thus justified in calling linear functionals covariant vectors.

D. BILINEAR AND MULTILINEAR FORMS

We next consider higher order functionals. In particular we will start with the **bilinear form** or **bilinear functional.**

1. Definition 2.9: Bilinear Functional

A mapping **f** of a pair of vectors, say $\mathbf{T} \in U^N$ and $\mathbf{S} \in V^M$ into a field of scalars, is a **bilinear functional** or **bilinear form** if **f**(**T**,**S**) is a linear function of **T** and **S** taken independently. We will only consider cases when N = M and $U^N = V^N$.

Once again choosing an arbitrary basis $A = \{\mathbf{a}_1, \ldots \mathbf{a}_N\}$ we may express two arbitrary contravariant vectors as linear combinations of these basis vectors.

$$\mathbf{T} = T^{\lambda}\mathbf{a}_{\lambda}, \text{ and } \mathbf{S} = S^{\mu}\mathbf{a}_{\mu}$$

The bilinear functional **f** can then be written

$$\mathbf{f}(\mathbf{S},\mathbf{T}) = \mathbf{f}(S^{\mu}\mathbf{a}_{\mu},T^{\lambda}\mathbf{a}_{\lambda}) \tag{2.35a}$$

$$= S^{\mu}\mathbf{f}(\mathbf{a}_{\mu},T^{\lambda}\mathbf{a}_{\lambda}) \tag{2.35b}$$

$$= S^{\mu}T^{\lambda}\mathbf{f}(\mathbf{a}_{\mu},\mathbf{a}_{\lambda}) \tag{2.35c}$$

The bilinear form is thus completely determined by the N^2 quantities $\mathbf{f}(\mathbf{a}_{\mu},\mathbf{a}_{\lambda})$. We will write these components of **f** as $f_{\mu\lambda}$. Using this shorthand, equation (2.35c) can be written as

$$\mathbf{f}(\mathbf{S},\mathbf{T}) = f_{\mu\lambda}S^{\mu}T^{\lambda} \tag{2.35d}$$

In matrix notation the bilinear form takes on the familiar appearance

$$\mathbf{f}(\mathbf{S},\mathbf{T}) = \mathbf{S}^{T}\mathbf{F}\mathbf{T} \tag{2.36}$$

where $\mathbf{F} = [f_{\mu\lambda}]$

If the two vectors **S** and **T** are equal then the bilinear form reduces to the well known quadratic form

$$\mathbf{f}(\mathbf{T},\mathbf{T}) = f_{\mu\lambda}T^{\mu}T^{\lambda} \tag{2.37a}$$

or in matrix notation

$$\mathbf{f}(\mathbf{T},\mathbf{T}) = \mathbf{T}^{T}\mathbf{F}\mathbf{T} \tag{2.37b}$$

We will be interested in the behaviour of the components, $f_{\mu\lambda}$, of the bilinear form, **f**, upon transition from one coordinate system to another. We establish their tensor character in the following example.

2. Example 2.4

In Example 2.3 we showed that a linear functional satisfied the definition of a covariant vector. Here we will show that a bilinear functional satisfies the definition of a covariant tensor of second order. It is necessary for the discussion that follows in later chapters to establish the tensor character of bilinear functionals. Since the bilinear functional yields an invariant (scalar), we may write

$$f_{\mu\lambda}S^{\mu}T^{\lambda} = f_{\mu'\lambda'}S^{\mu'}T^{\lambda'} \tag{2.38a}$$

$$= f_{\mu'\lambda'} S^{\mu} \frac{\partial x^{\mu'}}{\partial x^{\mu}} T^{\lambda} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} \tag{2.38b}$$

$$= \frac{\partial x^{\mu'}}{\partial x^{\mu}} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} f_{\mu'\lambda'} S^{\mu} T^{\lambda} \tag{2.38c}$$

Rearranging yields

$$(f_{\mu\lambda} - \frac{\partial x^{\mu'}}{\partial x^{\mu}} \frac{\partial x^{\lambda'}}{\partial x^{\lambda}} f_{\mu'\lambda'}) S^{\mu} T^{\lambda} = 0 \tag{2.38d}$$

Since the contravariant vectors **S** and **T** are arbitrary the quantity inside the parenthesis must vanish. This yields the relation

$$f_{\mu'\lambda'} = f_{\mu\lambda} \frac{\partial x^{\mu}}{\partial x^{\mu'}} \frac{\partial x^{\lambda}}{\partial x^{\lambda'}} \tag{2.38e}$$

This last equation is identical to the definition of a second order covariant tensor (equation (2.20).) We have thus proven that bilinear functionals are covariant tensors of order 2.

In general we can have m-linear functionals which map m contravariant vectors, **T**(1), ..., **T**(m), into a scalar and are linear functions of each of the m input vectors taken separately. They can be written as

$$\mathbf{f}(\mathbf{T}(1), ..., \mathbf{T}(m)) = T^{\lambda_1}(1) \cdots T^{\lambda_m}(m) f_{\lambda_1 \cdots \lambda_m} \tag{2.39}$$

Using identical arguments to the ones presented for linear and bilinear functionals, we can show that multilinear functionals are also covariant tensors.

E. TENSOR OPERATIONS

There are a few tensor operations that will be of considerable importance in later discussions. Although some have already been used we will formally define them here before proceeding. Only those operations that will be used in the sequel are presented. Others are possible and are discussed at length in the references (see for example [Refs. 1,5,6,7].)

1. Definition 2.10: Tensor Outer Product

Given the components of two tensors

$$T^{\lambda_1 \cdots \lambda_n}{}_{\mu_1 \cdots \mu_m} \text{ and } S^{\lambda_{n+1} \cdots \lambda_p}{}_{\mu_{m+1} \cdots \mu_q} \tag{2.40}$$

the $N^{(p+q)}$ numbers $R^{\lambda_1 \cdots \lambda_p}{}_{\mu_1 \cdots \mu_q}$ given by

$$R^{\lambda_1 \cdots \lambda_p}{}_{\mu_1 \cdots \mu_q} = T^{\lambda_1 \cdots \lambda_n}{}_{\mu_1 \cdots \mu_m} S^{\lambda_{n+1} \cdots \lambda_p}{}_{\mu_{m+1} \cdots \mu_q} \tag{2.41}$$

are components of a tensor of order p + q. The operation implied in the above is known as the **tensor outer product** or simply the **tensor product**.

2. Example 2.5

As an example of the tensor product operation consider two vectors defined as:

$$\mathbf{T} = [T^\lambda] \quad \text{and} \quad \mathbf{S} = [S^\mu] \tag{2.42}$$

The tensor product is given by (equation (2.41))

$$R^{\lambda\mu} = T^\lambda S^\mu \tag{2.43}$$

In this case the components of the tensor product can be arranged as a matrix of N^2 numbers. For simplicity consider the case when N=3. In matrix notation

$$\mathbf{R} = \mathbf{T}\mathbf{S}^T \tag{2.44a}$$

or

$$R^{\lambda\mu} = T^\lambda S^\mu \tag{2.44b}$$

$$= \begin{vmatrix} T^1S^1 & T^1S^2 & T^1S^3 \\ T^2S^1 & T^2S^2 & T^2S^3 \\ T^3S^1 & T^3S^2 & T^3S^3 \end{vmatrix} \tag{2.44c}$$

3. Definition 2.11: Contraction

The operation of setting two indices, one lower and the other upper, equal and summing the result is known as contraction. The result is a tensor which has the character indicated by the remaining indices. The contraction of a tensor of order p + 1 over two indices results in a tensor of order p - 1.

4. Example 2.6

We may contract the tensor

$$[H^{\lambda}{}_{\mu}] = \begin{bmatrix} H^1{}_1 & H^1{}_2 & h^1{}_3 \\ H^2{}_1 & H^2{}_2 & H^2{}_3 \\ H^3{}_1 & H^3{}_2 & H^3{}_3 \end{bmatrix} \tag{2.45}$$

over the indices λ and μ, resulting in

$$H^{\lambda}{}_{\lambda} = H^1{}_1 + H^2{}_2 + H^3{}_3 \tag{2.46a}$$

$$= \text{Trace}[H^{\lambda}{}_{\mu}] \tag{2.46b}$$

a scalar which represents the trace of the matrix.

We can consider a higher order example. Assuming a three dimensional vector space. consider contracting a tensor $U^{\lambda}{}_{\mu}{}^{\gamma}$ over the indices λ and μ. The result will be a vector whose components, U^{γ}, are given by

$$U^1 = U^1{}_1{}^1 + U^2{}_2{}^1 + U^3{}_3{}^1 \tag{2.47a}$$

$$U^2 = U^1{}_1{}^2 + U^2{}_2{}^2 + U^3{}_3{}^2 \tag{2.47b}$$

$$U^3 = U^1{}_1{}^3 + U^2{}_2{}^3 - U^3{}_3{}^3 \tag{2.47c}$$

5. Definition 2.12: Tensor Inner Product

Suppose that after we form the outer product of two arbitrary tensors **T** and **S**, with components $T^{\lambda_1 \cdots \lambda_n}{}_{\mu_1 \cdots \mu_m}$ and $S^{\lambda_{n+1} \cdots \lambda_p}{}_{\mu_{m+1} \cdots \mu_q}$, we set the indices λ_i and μ_j equal. This implies a contraction operation. The outer product is written (according to equation (2.41)) as

$$R^{\lambda_1 \cdots \lambda_p}{}_{\mu_1 \cdots \mu_q} = T^{\lambda_1 \cdots \lambda_n}{}_{\mu_1 \cdots \mu_m} S^{\lambda_{n+1} \cdots \lambda_p}{}_{\mu_{m+1} \cdots \mu_q} \tag{2.48a}$$

The contraction operation yields. (definition 2.11)

$$R^{\lambda_1 \cdots \lambda_{i-1} \lambda_{i+1} \cdots \lambda_p}{}_{\mu_1 \cdots \mu_{j-1} \mu_{j+1} \cdots \mu_q} = R^{\lambda_1 \cdots \lambda_{i-1} \lambda_i \lambda_{i+1} \cdots \lambda_p}{}_{\mu_1 \cdots \mu_{j-1} \lambda \mu_{j+1} \cdots \mu_q} \tag{2.48b}$$

It can be shown that the object. **R**. given in the last equation is a tensor of the character shown. If the original tensors. **T** and **S**. were of order m+n and p+q-(m+n) respectively. then the resulting tensor. **R**. will be of order (p+q-2). The

total operation. consisting of contracting the result of a tensor product over one (or more) pairs of indices. each member of the pair having come from a different tensor. is known as a **tensor inner product**, or simply **inner product**. This operation is also sometimes referred to as **transvection** (see for example [Ref. 5].)

This operation has been used previously in the discussion of linear, bilinear and multilinear functionals, (see equations (2.7b), (2.35d) and (2.39) respectively.) It will be particularly valuable in future discussions.

6. Example 2.7

Some insight into the inner product operation may be gained by explicitly performing the two steps described above for the simple case of the linear functional (equation (2.7b).)

$$y = H_\lambda T^\lambda \tag{2.49}$$

We can first form the outer product by replacing one of the λ indices appearing on the right hand side with a different letter. say μ. We may then write:

$$[H_\lambda T^\mu] = \begin{bmatrix} H_1T^1 & H_1T^2 & H_1T^3 \\ H_2T^1 & H_2T^2 & H_2T^3 \\ H_3T^1 & H_3T^2 & H_3T^3 \end{bmatrix} \tag{2.50}$$

The result is now contracted by equating λ and μ and performing the implied summation.

$$y = H_\lambda T^\lambda \tag{2.51a}$$

$$= H_1T^1 - H_2T^2 - H_3T^3 \tag{2.51b}$$

It is often useful to perform both these steps (tensor outer product and contraction) when calculating an inner product. particularly when dealing with higher order tensors. It may otherwise be difficult to keep track of how terms are combined or even which terms will be present in the final expression.

III. NONLINEAR SYSTEMS THEORY

In this chapter several models of nonlinear systems are described. The discussion begins with an overview of the classical continuous Volterra series. Several interesting aspects of the series are examined. Next, a discrete-time version of the Volterra series is introduced and is used to develop an equivalent tensor form. Finally, a new discrete-time, nonlinear tensor model is presented and its relationship to the Volterra series is discussed.

A. CONTINUOUS NONLINEAR SYSTEMS MODELS

Traditionally, non-linear systems have been modelled using the continuous Volterra series expansion [Ref. 26: p. 1559]. In its most general form this can be written as:

$$y(t) = h_0(t) - \int_{-\infty}^{\infty} h_1(t,\tau_1)x(\tau_1)d\tau_1 - \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(t,\tau_1,\tau_2)x(\tau_1)x(\tau_2)d\tau_1 d\tau_2 - \cdots \quad (3.1)$$

where $x(t)$ is the system input and $y(t)$ is the system output. The parameters $h_0(t)$, $h_1(t,\tau_1)$, $h_2(t,\tau_1,\tau_2)$, $\cdots$ are known as the **Volterra Kernels**. As we will only be concerned with time-invariant systems, the kernels will only be functions of the time difference, $(t-\tau)$, and not the actual time. The kernels then become: h_0, $h_1(t-\tau_1)$, $h_2(t-\tau_1,t-\tau_2)$, $\cdots$. Simple changes of variables then allow the series, (3.1), to be written in the form

$$y(t) = h_0$$

$$+ \int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1$$

$$+ \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2$$

$$+ \cdots \tag{3.2}$$

We note several things about the expansion. First, an infinite number of terms are required to represent the most general case. Second, the kernels h_0, $h_1(\tau_1)$, $h_2(\tau_1,\tau_2)$, $\cdots$ correspond to the constant, linear, quadratic, ... terms of the expansion, respectively. The familiar linear system appears as a special case of this more general expansion (ie; the case when all kernels except $h_1(\tau_1)$ are zero.) The third term:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 \tag{3.3}$$

is a bilinear term. that is it is linear in each variable $x(t-\tau_1)$ and $x(t-\tau_2)$ taken independently (ie: assume the other variable is a constant.) In general. the (i+1)-st term is i-linear. It is linear in each of the i variables $x(t-\tau_1)$, $x(t-\tau_2)$, ... $x(t-\tau_i)$ taken one at a time. Lastly. notice that the Volterra expansion is not orthogonal in the sense that the identification of the n-th order kernel depends on the values of all the other kernels. They cannot be identified independently.

The non-linear model represented by equation (3.1) can be visualized as shown in Figure 3.1. As can be seen it corresponds to a parallel connection of subsystems of increasing non-linearity. Each of the subsystems is homogeneous (except for the zeroth order subsystem) in the sense that increasing (multiplying) the input by a factor k results in the output of the p-th subsystem being increased by a factor k^p. This can easily be understood by examining the expression for the p-th term.

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_p(\tau_1, \ldots, \tau_p) x(t-\tau_1) \cdots x(t-\tau_p) d\tau_1 \cdots d\tau_p \tag{3.4}$$

Replacing x(t) by kx(t) yields

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_p(\tau_1, \ldots, \tau_p) kx(t-\tau_1) \cdots kx(t-\tau_p) d\tau_1 \cdots d\tau_p \tag{3.5a}$$

$$= k^p \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_p(\tau_1, \ldots, \tau_p) x(t-\tau_1) \cdots x(t-\tau_p) d\tau_1 \cdots d\tau_p \tag{3.5b}$$

The presence of a constant term in the Volterra expansion should not be unexpected. Consider, for example, a system whose response is given by:

$$y(t) = x(t) + h \tag{3.6}$$

It is easily shown that this system does not obey the principle of superposition and so cannot be considered linear. This necessitates the inclusion of a constant term in the Volterra expansion in order to handle the general case. The usual procedure adopted in linear analysis, if a constant term appears, is to define a new output function which is the actual output less the constant term. This new output function is then identified in the usual fashion. The constant term must be separately identified. If we admit that the system is non-linear then this will no longer be necessary.

There are many other aspects of continuous-time nonlinear system modelling that have not been discussed. In the next section discrete-time nonlinear systems are introduced. Many of the comments that will be made there are equally applicable to continuous-time nonlinear systems. However, we make them in the context of discrete-time, since that will make them more applicable to the sequel.

B. DISCRETE-TIME NONLINEAR SYSTEM MODELS

A discrete version of the time-invariant Volterra expansion can be written as:

$$y(k) = h_0$$

$$+ \sum_{\lambda_1=0}^{N} h_1(\lambda_1) x(k-\lambda_1)$$

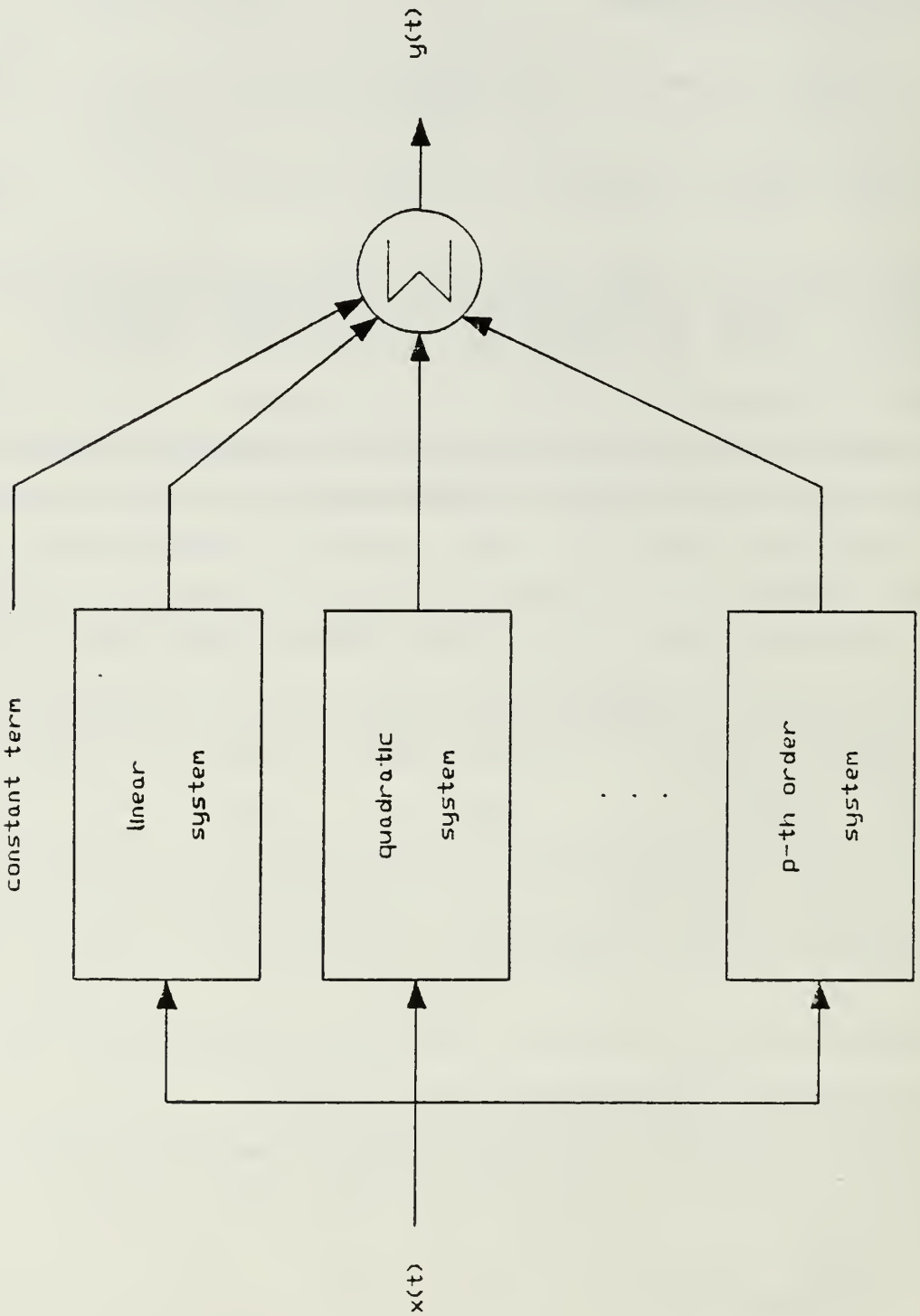


Figure 3.1: Nonlinear Volterra Series Model

$$+ \sum_{\lambda_1=0}^{N} \sum_{\lambda_2=0}^{N} h_2(\lambda_1,\lambda_1)x(k-\lambda_1)x(k-\lambda_2)$$

$$+ \cdots \tag{3.7}$$

where we have assumed that the system is causal.

For a large class of systems the summations can be truncated to N+1 terms. Certainly truncation is required for computer implementation. This truncation implies that only finite memory nonlinear systems can be represented. Equation (3.7) is an extension of the linear Moving Average (MA) type model. In fact the linear model is a special case of equation (3.7). This expansion is non-recursive in nature, that is, it expands the present output only in terms of the present and past input. Past outputs are not used.

The representation of the kernels in both the continuous and discrete forms of the Volterra expansion is not unique. There are, however, several special forms which are important. Consider a second order kernel for which $h_2(\lambda_1,\lambda_2) = h_2(\lambda_2,\lambda_1)$. This kernel is symmetric with respect to the two parameters λ_1 and λ_2. It turns out that the kernel can always be symmetrized with no loss of generality. For the p-th order kernel the procedure for obtaining the symmetric kernel from an asymmetric one is given by [Ref. 22]:

$$h_{sym}(\lambda_1, \ldots, \lambda_2) = \frac{1}{n!}\sum_{\pi()} h_p(\lambda_{\pi(1)}, \ldots, \lambda_{\pi(p)}) \tag{3.8}$$

where the summation is over all n! possible permutations of the p λ's. Although the symmetric form may contain more terms than an asymmetric form it is of importance because it is unique [Ref. 9: p. 43]. There can be many equivalent asymmetric forms of the kernel which all lead to the same symmetric kernel (through equation (3.8)). The symmetric kernel thus provides a standard form which can be used as a reference.

Other forms of interest are also possible. The symmetry of the symmetric kernel implies redundancy. This redundancy can be eliminated by use of a **triangular kernel**. Consider a kernel defined so that $h_{tri}(\lambda_1, \ldots, \lambda_p) = 0$ whenever

$\lambda_s > \lambda_t$ for $s < t$. The domain of a second order triangular kernel is illustrated in Figure 3.2a. For comparison, the equivalent symmetric kernel's domain is illustrated in Figure 3.2b.

Using the triangular kernel the output of the p-th order nonlinear subsystem is given by

$$y(k) = \sum_{\lambda_1=0}^{\lambda_2} \sum_{\lambda_2=0}^{\lambda_3} \cdots \sum_{\lambda_p=0}^{N} h_{tri}(\lambda_1, \ldots, \lambda_p) x(k-\lambda_1) \cdots x(k-\lambda_p) \tag{3.10}$$

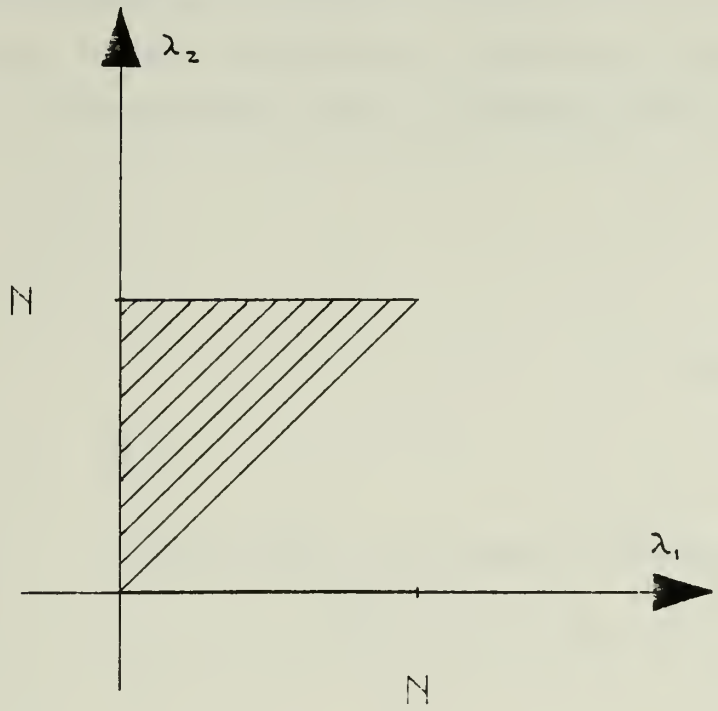
Notice that the limits of the summations reflect the triangular domain. This implies that fewer terms are included in the summations resulting in computational savings. The triangular kernel defined above, and used in equation (3.10), is not unique. Other triangular kernels can be formed by choosing alternate triangular domains. For example, the domain illustrated in Figure 3.3 could equivalently have been used. This choice corresponds to setting $h_{tri}(\lambda_1, \ldots, \lambda_p) = 0$ whenever $\lambda_s > \lambda_t$, for $s > t$.

The output of nonrecursive models of the type presented in equation (3.7) is stable if the input is bounded and if the series is truncated to a finite number of summations. Consider an input x(n) which is bounded to be less than some constant M. If the series is truncated to p+1 terms then, in the worst case the output will be

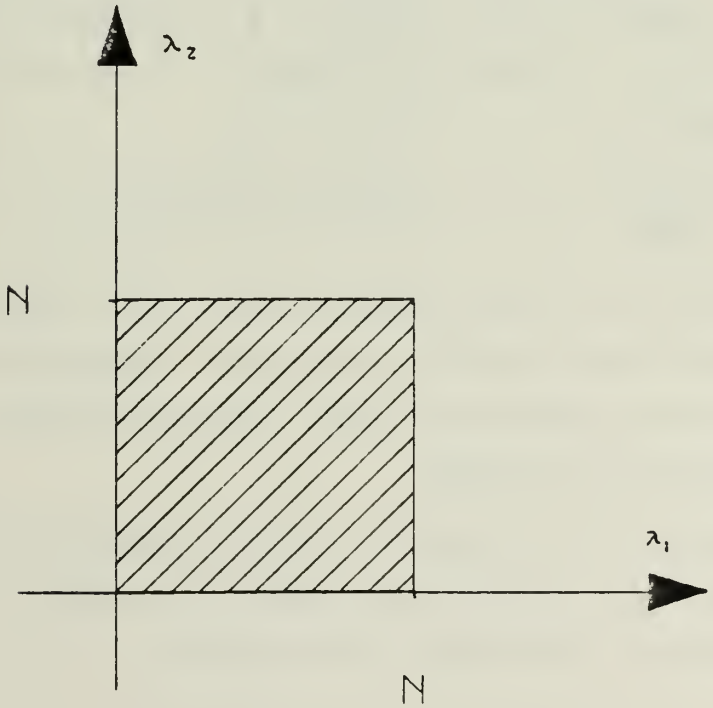
$$y(n) \leq h + M \sum_{\lambda_1=0}^{N} |h_1(\lambda_1)| + M^2 \sum_{\lambda_1=0}^{N} \sum_{\lambda_2=0}^{N} |h_2(\lambda_1, \lambda_2)| +$$

$$\cdots + M^p \sum_{\lambda_1=0}^{N} \cdots \sum_{\lambda_p=0}^{N} |h_p(\lambda_1, \ldots, \lambda_p)| \tag{3.11}$$

So, as long as the summations are bounded (which will generally be the case), the output will always remain finite. This guaranteed stability makes MA type models very attractive. As mentioned earlier, their shortcoming is their inability to accurately model infinite memory systems without using a large number of terms.



a. Second Order Triangular Kernel Domain



b. Second Order Symmetric Kernel Domain

Figure 3.2: Triangular and Symmetric Volterra Kernel Domains

1. Tensor Formulation

In order to adopt a tensor formulation of the problem we notice that equation (3.7) can be considered to consist of a series of increasing order functionals. As has been shown, these functionals can be expressed as tensors. We can therefore rewrite equation (3.7) as a tensor equation.

$$y(k) = H + H_{\lambda_1} x^{\lambda_1} + H_{\lambda_1\lambda_2} x^{\lambda_1} x^{\lambda_2} + \cdots \tag{3.12}$$

where we have defined the contravariant input vector as

$$\mathbf{x} = [x(k)\ x(k-1)\ \cdots\ x(k-\lambda)\ \cdots\ x(k-N)]^T \tag{3.13a}$$

$$= [x^{\lambda}]^T \tag{3.13b}$$

This choice for x has the effect of truncating the series to N+1 terms. The symbols $H, H_{\lambda_1}, H_{\lambda_1\lambda_2}, \cdots$ represent the components of covariant tensors of order 0, 1, 2, etc., respectively.

Examining equation (3.12) we make particular note of the following two aspects:

(1) The dimension of the vector space is related to the memory order of the system. The vector **x** has N+1 components implying that the nonlinear system contains no more than N delays. This fact is explicit in the way the input vectors have been defined.

(2) The nonlinearity of the expansion is provided implicitly by the tensor outer product operation. It is the outer product of the vector **x** with itself that makes the higher order (2 and up) terms nonlinear.

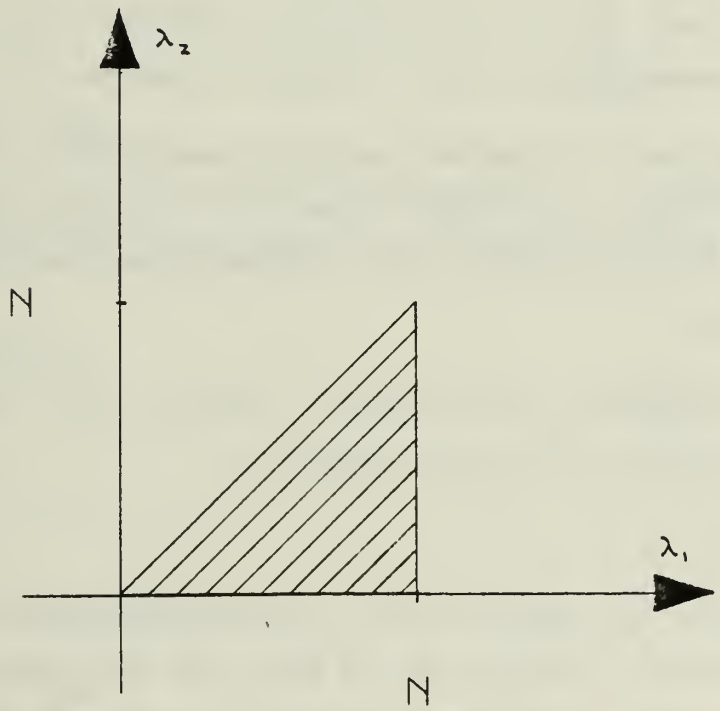


Figure 3.3: Alternate Second Order Triangular Volterra Kernel Domain

These observations provoke speculation about the possibility of an alternate formulation where the roles played by the dimensionality of the vector space and the outer product operation are interchanged.

2. Alternate Tensor Formulation

Consider a parametric description of a curve in $\mathbf{V}^{p+1}$, a p+1 dimensional vector space, given by;

$$\mathbf{x}(k) = \begin{bmatrix} x^0(k) = [x(k)]^{(0)} \\ x^1(k) = [x(k)]^{(1)} \\ \vdots \\ x^p(k) = [x(k)]^{(p)} \end{bmatrix} \tag{3.14}$$

where the superscript in parenthesis indicates exponentiation. Given any $\mathbf{x}(k)$, the components $x^{\lambda}(k)$, $\lambda = 0,\ldots,p$ define a point in $\mathbf{V}^{p+1}$. As $x(k)$ varies with k, the vector $\mathbf{x}(k)$ will describe a curve in $\mathbf{V}^{p+1}$. We define the components of another vector in $\mathbf{V}^{p+1}$ as

$$x^{\lambda_1}(k-1) = [x(k-1)]^{(\lambda_1)} \tag{3.15}$$

Similarly, the N-th such vector can be defined as,

$$x^{\lambda_N}(k-N) = [x(k-N)]^{(\lambda_N)} \tag{3.16}$$

The vectors in equations (3.14), (3.15), and (3.16) will be referred to as **observation** vectors. Although, at this point they only depend on past and present system input values, later, in Section D, they will be generalized to include past outputs as well. The input and output measurements represent the system observations, hence the name.

Using the theory developed in Chapter 2, concerning multilinear functionals, the following mathematical relation is proposed as a model of a finite order, finite memory nonlinear system:

$$\hat{y}(k) = x^{\lambda_0}(k) \cdots x^{\lambda_N}(k-N) H_{\lambda_0 \cdots \lambda_N} \tag{3.17}$$

where $H_{\lambda_0 \cdots \lambda_N}$ is an (p+1)-order covariant tensor representing a (p+1)-linear functional. This covariant tensor performs a role similar to that of a Volterra kernel.

This model is equivalent to the p+1 term Volterra series (equation (3.7) or (3.12)), although it does contain many additional terms. It has the advantage of notational simplicity. It replaces a p+1 term series with a single term and requires the specification of only a single **composite** kernel, $H_{\lambda_0 \cdots \lambda_N}$. As will be shown in Section D, equation (3.17) is considerably more general than equation (3.12). It will be shown that Wiener type models can be obtained from equation (3.17) by a simple coordinate transformation. Other choices of observation vectors yield Autoregressive (AR) or even Autoregressive-Moving Average (ARMA) type models.

In order to illustrate the correspondence of equation (3.17) to the standard Volterra type series expansion of equation (3.12) we present a simple example.

3. Example 3.1

Consider the truncated Volterra series expansion corresponding to equation (3.12):

$$y(k) = H + H_{\lambda_1} x^{\lambda_1} + H_{\lambda_1 \lambda_2} x^{\lambda_1} x^{\lambda_2} \tag{3.18}$$

where $\lambda_1 = 0,1$ and $\lambda_2 = 0,1$, and

$$x = \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} = \begin{bmatrix} x(k) \\ x(k-1) \end{bmatrix}$$

We may explicitly write out all the terms implied in equation (3.18). This yields

$$\begin{aligned} y(k) = {} & H \\ & + H_0 x(k) + H_1 x(k-1) \\ & + H_{00} x(k)x(k) + H_{01} x(k)x(k-1) \\ & + H_{10} x(k-1)x(k) + H_{11} x(k-1)x(k-1) \end{aligned} \tag{3.19}$$

Now consider a model of similar order given by equation (3.17).

$$y(k) = H_{\lambda_0\lambda_1} x^{\lambda_0}(k) x^{\lambda_1}(k-1) \tag{3.20}$$

where $\lambda_0, \lambda_1 = 0,1,2$ and

$$x^{\lambda_0}(k) = [x(k)]^{(\lambda_0)}$$

$$x^{\lambda_1}(k-1) = [x(k-1)]^{(\lambda_1)}$$

The tensor product, $x^{\lambda_0}(k)x^{\lambda_1}(k-1)$, appearing in equation (3.20) results in a contravariant tensor of second order. Its elements are

$$[x^{\lambda_0}(k)x^{\lambda_1}(k-1)] = \begin{bmatrix} 1 & x(k-1) & x^{(2)}(k-1) \\ x(k) & x(k)x(k-1) & x(k)x^{(2)}(k-1) \\ x^{(2)}(k) & x^{(2)}(k)x(k-1) & x^{(2)}(k)x^{(2)}(k-1) \end{bmatrix} \tag{3.21}$$

We notice that all the elements on or above the southwest- northeast diagonal are included in equation (3.19), the terms below this diagonal are not. This new form has not discarded any terms present in the latter version and so cannot represent a loss of generality. The extra terms that are included in this new form do not pose any significant problem. If the system does not contain terms involving these particular elements then the corresponding term in the kernel will go to zero during the identification process. Certainly in some classes of systems these additional elements will be important.

C. DISCRETE NONLINEAR SYSTEM IDENTIFICATION

In Section B a tensor formulation of the discrete nonlinear modelling problem was introduced. In this section methods for kernel identification are presented. All the methods that are discussed are statistical in nature and utilize a least-squares approach of parameter estimation. It will be shown that familiar methods used in linear systems modelling can be extended to handle the nonlinear case. In the first section a tensor equivalent of the **normal equations**, which can be solved for the unknown system parameters, is derived. Several recursive (in time) solutions to the problem are then presented. Finally, simulation results are offered to illustrate the effectiveness of the algorithms.

1. Derivation of Normal Equations

In Section B (equation (3.17)). the following tensor model for nonlinear systems was proposed:

$$y(k) = x^{\lambda_0}(k) \cdots x^{\lambda_N}(k-N)H_{\lambda_0 \cdots \lambda_N} \tag{3.22}$$

where $y(k)$ represents an estimate of the system output and the $x^{\lambda_i}(k-i)$'s are components of contravariant observation vectors, defined in equations (3.14) through (3.16).

Consider the analysis model of Figure 3.4. This diagram represents a conceptualization of the system identification process. The assumption is made that the unknown system can be represented by an equation identical to the model equation, (3.22), where the system parameters $H_{\lambda_0 \cdots \lambda_N}$ are unknown. The parameters of the model are adjusted to best match the actual system parameters. A convenient measure of how well the model represents the actual system is the mean-square error or MSE. The MSE is a quadratic function of the model parameters which implies that there exists a unique minimum. or optimal solution, to the problem. Minimizing the MSE yields a linear set of simultaneous equations which can be solved for the unknown model parameters. In addition. the quadratic nature of the MSE allows steepest descent type, adaptive algorithms to be used.

The error signal, $e(k)$. defined as the difference between the actual nonlinear system output, $y(k)$. and the output of the model, $\hat{y}(k)$, is given by

$$e(k) = y(k) - \hat{y}(k) \tag{3.23}$$

The mean-square value of this error signal is given by

$$E\left\{e^2(k)\right\} = E\left\{\left(y(k) - \hat{y}(k)\right)^2\right\} \tag{3.24a}$$

$$= E\left\{\left(y(k) - x^{\lambda_0}(k) \cdots x^{\lambda_N}(k-N)H_{\lambda_0 \cdots \lambda_N}\right)^2\right\} \tag{3.24b}$$

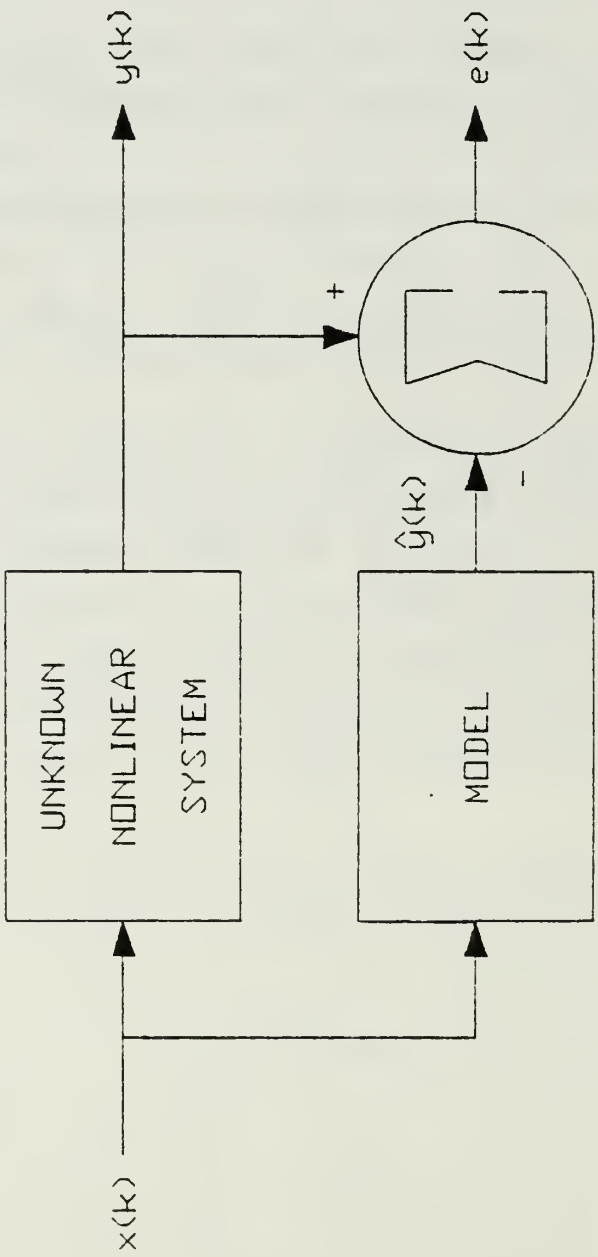


Figure 3.4: System Analysis Model

where E{ } indicates expectation. The optimal set of parameters can be found by minimizing the MSE, $E\{e^2(k)\}$, with respect to the parameters $H_{\lambda_0 \cdots \lambda_N}$. The gradient of the MSE is formed by differentiating with respect to the unknown parameters.

$$\frac{\partial E\left\{e^2(k)\right\}}{\partial H_{\lambda_0 \cdots \lambda_N}}$$

$$= 2E\left\{[y(k) - x^{\lambda_0}(k) \cdots x^{\lambda_N}(k-N)H_{\lambda_0 \cdots \lambda_N}][-x^{\mu_0}(k) \cdots x^{\mu_N}(k-N)]\right\} \qquad (3.25a)$$

Setting this equal to zero yields

$$2E\left\{[y(k)x^{\mu_0}(k)...x^{\mu_N}(k-N) - x^{\lambda_0}(k)...x^{\lambda_N}(k-N)x^{\mu_0}(k)...x^{\mu_N}(k-N)H_{\lambda_0 \cdots \lambda_N}]\right\}$$

$$= 0 \qquad (3.25b)$$

or.

$$E\left\{x^{\lambda_0}(k)...x^{\lambda_N}(k-N)x^{\mu_0}(k)...x^{\mu_N}(k-N)\right\}H_{\lambda_0 \cdots \lambda_N} = E\left\{y(k)x^{\mu_0}(k)...x^{\mu_N}(k-N)\right\}$$

(3.25c)

The expression, (3.25c), is a tensor equivalent of the Wiener-Hopf or **normal equations**. As will be shown in Section D, these equations can also be used to represent the Yule-Walker equations if a different choice is made for the observation vectors, $x(k-i)$. The term on the left-hand-side of equation (3.25c) is a nonlinear extension of the autocorrelation matrix. This contravariant tensor of order $2(N+1)$ includes various higher order correlations as well as the second order ones which arise in the linear case. The expectation on the right-hand-side represents a cross correlation between the output and various linear and nonlinear functions of the input.

The minimum MSE can be determined by substituting (3.25c) into (3.24b). This yields

$$E\left\{e^2(k)\right\}_{min} = E\left\{y^2(k)\right\} - 2E\left\{y(k)x^{\lambda_0}(k)\cdots x^{\lambda_N}(k-N)H_{\lambda_0\cdots\lambda_N}\right\}$$

$$+ H_{\lambda_0\cdots\lambda_N}H_{\mu_0\cdots\mu_N}E\left\{x^{\lambda_0}(k)\ldots x^{\lambda_N}(k-N)x^{\mu_0}(k)\ldots x^{\mu_N}(k-N)\right\} \tag{3.26a}$$

$$= E\left\{y^2(k)\right\} - E\left\{y(k)x^{\lambda_0}(k)\cdots x^{\lambda_N}(k-N)\right\}H_{\lambda_0\cdots\lambda_N} \tag{3.26b}$$

Equation (3.25c) represents $(p+1)^{(N+1)}$ equations in as many unknowns, and so can theoretically be solved for the unknown parameters $H_{\lambda_0\cdots\lambda_N}$. However, in practice the number of computations required to perform the matrix inversion becomes unwieldly. An $n\times n$ matrix inversion takes on the order of $O(n^3)$ operations [Ref. 27: p. 58], therefore, $O([p+1]^{3(N+1)})$ operations will be required in the solution of (3.25c). In order to make the task manageable, alternate algorithms that avoid direct matrix inversion, must be employed to solve the normal equations.

2. The Least Mean Square (LMS) Algorithm

The Least Mean Square (LMS) algorithm has successfully been employed in the solution of a variety of linear problems [Ref. 28]. It is a recursive algorithm based on a gradient, steepest descent type of strategy. The mean-square error is a hyperparaboloid which is concave upward. Steepest descent algorithms strive to descend down this quadratic surface, towards the minimum, by making adjustments to the unknown parameters proportional to the gradient. This can be expressed mathematically as

$$H_{\lambda_0\cdots\lambda_N}(k-1) = H_{\lambda_0\cdots\lambda_N}(k) - \mu\nabla_{\lambda_0\cdots\lambda_N}(k) \tag{3.27}$$

where $H_{\lambda_0\cdots\lambda_N}(k)$ is the value of the model parameters at the k-th time instant and μ is a parameter which controls the convergence of the algorithm. The symbol, $\nabla_{\lambda_0\cdots\lambda_N}(k)$, is used to represent the gradient.

The actual value of the gradient is given, according to equation (3.25a), by

$$\nabla_{\lambda_0 \quad \lambda_N}(k) = \quad 2E\left\{e(k)[x^{\lambda_0}(k) \cdot \cdot \cdot x^{\lambda_N}(k-N)\right\} \tag{3.28}$$

The LMS algorithm approximates the MSE by the instantaneous value of the error squared. The approximate value of the gradient is

$$\hat{\nabla}_{\lambda_0 \cdots \lambda_N}(k) = -\,2e(k)[x^{\lambda_0}(k) \cdots x^{\lambda_N}(k-N)] \tag{3.29}$$

where the circumflex indicates an estimated value. Using this approximation in equation (3.25) yields

$$H_{\lambda_0 \quad \lambda_N}(k+1) = H_{\lambda_0 \quad \lambda_N}(k) + 2\mu e(k)x^{\lambda_0}(k) \cdots x^{\lambda_N}(k-N) \tag{3.30}$$

Equation (3.30) gives a straight forward method of determining the system parameters. It involves no matrix inversion nor does it require the correlation tensor to be known. These two properties make the calculations required at each iteration very simple, so that it is possible to perform them in real time. It can be shown that the LMS algorithm converges to the optimal solution [Ref. 29: p. 578].

For the linear case, the algorithm will converge as long as the parameter, μ, is chosen to satisfy. [Ref. 29: p. 578].

$$0<\mu<\frac{1}{\lambda_{max}} \tag{3.31}$$

where λ_{max} is the largest eigenvalue of the correlation matrix. Since the correlation matrix is positive definite, all its eigenvalues are greater than zero. Therefore, the trace of the correlation matrix will always be greater than the largest eigenvalue. The following condition will, therefore, ensure stability.

$$0<\mu<\frac{1}{\text{Tr correlation matrix}} \tag{3.32}$$

For the nonlinear case this translates to

$$0<\mu<\frac{1}{\sum_{\lambda_0=0}^{N} \cdots \sum_{\lambda_N=0}^{N} E\left\{x^{\lambda_0}(k)\ldots x^{\lambda_N}(k \quad N)x^{\lambda_0}(k)\ldots x^{\lambda_N}(k-N)\right\}} \tag{3.33}$$

In practice a value considerably smaller than the maximum permitted by equation (3.33) is used to give slower, more accurate convergence.

3. Recursive Least Squares (RLS) Algorithm

The recursive least squares, or RLS algorithm, is similar to the well known Kalman filter except that it is used to estimate model parameters rather than the system state. The tensor form of the normal equations is not suitable for RLS implementation. The elements of the correlation tensor must first be rearranged in a two dimensional matrix. This can be accomplished by replacing the tensor outer products appearing in equation (3.25c) with matrix Kronecker products. The Kronecker product of an $\mathbf{n}\times\mathbf{m}$ matrix, $\mathbf{A} = [a_{\lambda_1\lambda_2}]$, and a $\mathbf{p}\times\mathbf{q}$ matrix, $\mathbf{B} = [b_{\mu_1\mu_2}]$, is an $\mathbf{np}\times\mathbf{mq}$ matrix given by [Ref. 30,31],

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1m}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2m}\mathbf{B} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{n1}\mathbf{B} & a_{n2}\mathbf{B} & \cdots & a_{nm}\mathbf{B} \end{bmatrix} \tag{3.34}$$

where the symbol $\otimes$ is used to denote the Kronecker product operation.

In order to rewrite the normal equations, (3.25c), in this matrix form, the covariant tensor of system parameters, $H_{\lambda_0 \cdots \lambda_N}$, must be put into a vector form by a lexicographic reordering of the elements. The resulting parameter vector is denoted $\mathbf{H}$. The normal equations become

$$E\left\{\mathbf{x}(k) \otimes \cdots \otimes \mathbf{x}(k-N) \otimes \mathbf{x}(k) \otimes \cdots \otimes \mathbf{x}(k-1)^T\right\}\mathbf{H}$$

$$= E\left\{y(k)\,\mathbf{x}(k) \otimes \cdots \otimes \mathbf{x}(k-N)\right\} \tag{3.35}$$

If an assumption of ergodicity is made then the statistical averages in (3.35) can be replaced with time averages.

$$\lim_{K\to\infty} \frac{1}{K} \left[\sum_{k=0}^{K-1} \mathbf{X}(k)\mathbf{X}^T(k) \right] \mathbf{H} = \lim_{K\to\infty} \frac{1}{K} \left[\sum_{k=0}^{K-1} y(k)\mathbf{X}(k) \right] \tag{3.36}$$

where $\mathbf{X}(k) = \mathbf{x}(k) \otimes \cdots \otimes \mathbf{x}(k - N)$. For computational purposes equation (3.36) is approximated as

$$\sum_{k=0}^{K-1} \mathbf{X}(k)\mathbf{X}^T(k) = \sum_{k=0}^{K-1} y(k)\mathbf{X}(k) \tag{3.37}$$

As a further notational convenience, the following two definitions are made,

$$\mathbf{X}_K = \begin{bmatrix} \mathbf{X}^T(0) \\ \mathbf{X}^T(1) \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{X}^T(K) \end{bmatrix} \tag{3.38}$$

and,

$$\mathbf{Y} = \begin{bmatrix} y(0) \\ y(1) \\ \cdot \\ \cdot \\ \cdot \\ y(K) \end{bmatrix} \tag{3.39}$$

Using these definitions the normal equations can be written in the compact form

$$\mathbf{X}_K^T\mathbf{X}_K\mathbf{H} = \mathbf{X}_K^T\mathbf{Y} \tag{3.40}$$

This last from of the normal equations is precisely the same as the one used by Goodwin and Payne [Ref. 32: p. 176] for the derivation of the RLS algorithm. The derivation involves the use of the **matrix inversion lemma** [Ref. 33: p. 247] which replaces a matrix inversion at each iterative step by a simple scalar division. It is this simplification which yields the efficiency of the RLS algorithm. The RLS equations are [Ref. 32: pp. 176-177].

$$\mathbf{H}_{K-1} \quad \mathbf{H}_K - \mathbf{Q}_{K-1}(y(K) \quad \mathbf{X}^T(K-1)\mathbf{H}_K) \tag{3.41a}$$

$$\mathbf{Q}_{K-1} \quad \frac{\mathbf{P}_K\mathbf{X}(K-1)}{1 + \mathbf{X}^T(K-1)\mathbf{P}_K\mathbf{X}(K-1)} \tag{3.41b}$$

$$\mathbf{P}_{K+1} = [\mathbf{I} - \mathbf{Q}_{K+1}\mathbf{X}^T(K+1)\,\mathbf{P}_K \qquad \textbf{(3.41c)}$$

$$= [\mathbf{X}_{K+1}^T\mathbf{X}_{K+1}]^{-1} \qquad \text{(3.41d)}$$

Equation (3.41a) deserves some comment. The term in the parentheses is normally known as the **innovation.** It is a scalar which represents the new information gained in the latest measurement. The term $\mathbf{X}^T(K+1)\mathbf{H}_K$ is an estimate of the current output given the new input measurement and the old estimate of the system parameters. The **innovation** is thus an error signal. The term in front of the bracket, $\mathbf{Q}_{K+1}$, is a gain vector. It gives an indication of how much faith is being put in the new information. If the gain is small, then the new estimate will be essentially the old estimate. Conversely, if the gain is large then the new estimate will depend to a great degree on the new information. When the algorithm has converged, the gain will be close to zero. If the system parameters change for any reason, the algorithm will not detect the change since it is ignoring the new information. In order to circumvent this problem a weighting can be applied to the data, so that new data is artificially emphasized. Exponential and rectangular windows have been successfully employed for this purpose. The time constant used in the case of the exponential window is often called a **forgetting factor** since it ensures that the algorithm's memory is finite. Use of windows will not be discussed further in this dissertation. The interested reader should consult reference [Ref. 32: pp. 179-185].

4. Simulation Results

In order to investigate the validity of the theoretical results, several computer simulations were performed. Two examples are presented, representing two different classes of systems. Many other systems were also tested, however, the results presented are typical of those obtained from all the simulations. The FORTRAN programs written allowed nonlinearities of up to fourth order, but were limited to systems involving only zero or one delay.

The first example was chosen to correspond exactly to the model equation (3.17). The system was excited by white, zero mean, uniform noise.

The problem was to identify the system parameters from only input and output measurements. The unknown covariant $H_{\lambda_1\lambda_2}$ tensor was chosen to be

$$\left[H_{\lambda_0\lambda_1}\right] = \begin{bmatrix} .2 & -.4 & .03 & -.7 & 0.0 \\ .5 & .35 & .11 & .9 & 0.0 \\ .01 & 1.3 & -.33 & .7 & 0.0 \\ .43 & .81 & -.05 & .4 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \tag{3.42}$$

Therefore, the output equation consists of 16 nonzero terms, ranging from $.2x^{(0)}(k)x^{(0)}(k-1)$ up to $.4x^{(3)}(k)x^{(3)}(k-1)$. This model will be called System I in the sequel.

The other type of nonlinearity tested was one that was known to require an infinite number of terms. The particular example chosen for this was the unit step function which simulates a saturation type of nonlinearity. It is a convenient choice since an analytical solution can be calculated.

The unit step function has different $H_{\lambda_0 \cdots \lambda_N}$ tensors depending on the order of the model chosen. This is a result of the chosen coordinates not being orthogonal (this will be clarified in Section D.) The second. third and fourth order (nonlinearity order) models are given by

$$\left[H_{\lambda_0}\right] = \left[\frac{1}{2} \quad \frac{3}{4} \quad 0\right]$$

$$= \left[.5 \quad .75 \quad 0.0\right] \tag{3.43a}$$

$$\left[H_{\lambda_0}\right] = \left[\frac{1}{2} \quad \frac{45}{32} \quad 0 \quad \frac{-35}{32}\right]$$

$$\left[.5 \quad 1.40625 \quad 0.0 \quad -1.09375\right] \tag{3.43b}$$

$$\left[H_{\lambda_0}\right] = \left[\frac{1}{2} \quad \frac{45}{32} \quad 0 \quad \frac{-35}{32} \quad 0\right]$$

$$= \left[.5 \quad 1.40625 \quad 0.0 \quad -1.09375 \quad 0.0\right] \tag{3.43c}$$

Notice that the unit step function has no memory. The expansion contains only odd powers of x(k) and a constant term since the unit step function can be written as a constant plus an odd function (the signum function.) We will refer to these three models collectively as System II in the sequel.

The direct solution of the normal equations was tested on these two models as was the LMS algorithm. To date the RLS algorithm has not been verified.

a. Direct Solution of Normal Equations

To verify the direct (matrix inversion) method of solution of the normal equations, a FORTRAN program was written which estimated the correlation tensors, and performed the required matrix inversion. The program was written so as to allow the number of points used to approximate the correlations to be varied. The maximum power of x(k) was also made to be adjustable so that the effect of over or under modelling could be studied. The final adjustable parameter was the magnitude of the uniform, zero mean, white noise that was used to excite the system. Adjusting this last parameter affects the range over which the resultant model is valid. In an actual application, something about the range of expected system inputs would have to be known in order to select a good value for this parameter.

The results for System I are given in Table 3.1 for several combinations of the three variable parameters. The results are remarkably good even with as few as 30 input samples. Since no measurement noise was introduced this is perhaps not surprising.

Overmodelling did not present any problems. The additional terms were identified by the algorithm to be essentially zero. This is evident in Table 3.1c. Undermodelling did introduce some inaccuracy. The coefficients identified are significantly different from the ones in the unknown system. However, the coefficients identified should represent the best second order approximation to the system, which will not in general be the same as the second order coefficients contained in the third order model.

a. 30 data points were used
The maximum power of x(k) was 3
The noise was uniform on (-1,1)

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .200 & -.400 & .030 & -.700 \\ .500 & .350 & .110 & .900 \\ .010 & 1.300 & -.330 & .700 \\ 430 & .810 & -.050 & .400 \end{bmatrix}$$

b. 500 data points were used
The maximum power of x(k) was 3
The noise was uniform on (-10,10)

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .19993 & -.39952 & .030016 & -.70001 \\ .49990 & .35014 & .11001 & .90000 \\ .010006 & 1.3000 & -.33000 & .70000 \\ .43000 & .81000 & -.050000 & .40000 \end{bmatrix}$$

c. 500 data points were used
The maximum power of x(k) was 4
The noise was uniform on (-1,1)

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .20000 & -.40000 & .03000 & -.70000 & -.26356E-06 \\ .50000 & .35000 & .11000 & .90000 & -.10016E-06 \\ .010000 & 1.30000 & -.33000 & .70000 & .33811E-05 \\ .43000 & .81000 & -.050000 & .40000 & -.83230E-07 \\ -.48140E-06 & -.43483E-06 & .34339E-05 & .52571E-06 & -.43294E-05 \end{bmatrix}$$

d. 500 data points were used
The maximum power of x(k) was 2
The noise was uniform on (-1,1)

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .20616 & -.80030 & .037553 \\ .75931 & 1.5108 & .069939 \\ -.033444 & 1.6619 & -.24457 \end{bmatrix}$$

TABLE 3.1: System I Results Using Full Matrix Inversion.

The results for System II are shown in Table 3.2. Significantly more points were required to accurately identify System II than were needed for System I. The unit step function cannot be represented exactly by a finite number of polynomials so it is not surprising that the solution is not precise. There is another factor that contributes to the poor accuracy. The choice of functions used as coordinates, equation (3.14), leads to ill conditioned equations (see Strang [Ref. 34: p. 135].) This problem will be corrected in Section D.

b. Simulation Using LMS Algorithm

To verify the LMS algorithm a FORTRAN program was written, which used equation (3.30), to adaptively identify the covariant tensor $H_{\lambda_0 \cdots \lambda_N}$. The program allowed the convergence parameter, μ, and the magnitude of the uniform, white noise to be varied. The results of the simulations are presented in Table 3.3 and 3.4 for System I and II respectively. Equation (3.33) was used to bound the convergence parameter, μ. The input excitation noise was chosen to be uniform on (-1,1) resulting in a bound for the convergence parameter of $0 < \mu < 0.3558$.

In general convergence was slow. The linear and "close to linear" terms were identified most rapidly. The highly nonlinear terms, involving high powers of x(k) or x(k-1) and their cross terms, were last to be identified. Their accuracy never reached that of the lower order terms. The algorithm was very sensitive to the setting of the convergence parameter, μ. A value smaller than the bound predicted above was used to achieve satisfactory performance.

D. GENERALIZED COORDINATE SYSTEMS

In Section B, a coordinate system was introduced that was closely related to the Volterra series (equation (3.14).) This system was subsequently used in the remainder of Section B and in Section C. There is really little motivation for choosing this particular set of coordinates. In fact there are very compelling reasons to search for other sets of coordinates. The set (3.14) can lead to poorly conditioned sets of equations (see Strang [Ref. 34: p. 135]), a fact that was mentioned in the last section. In higher order systems this can become a serious

a. 1000 data points were used.
The maximum power of x(k) was 3.
The input noise was uniform on (-1,1)

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .49278 & 1.4301 & .000519 & -1.1389 \\ -.037083 & -.10437 & .067337 & .18706 \\ -.025155 & -.072552 & -.035273 & .10310 \\ .084649 & .080036 & -.13334 & -.15251 \end{bmatrix}$$

b. 15000 data points were used.
The maximum power of x(k) was 3.
The input noise was uniform on (-1,1).

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .50380 & 1.4131 & -.0093646 & -1.1029 \\ -.0064558 & -.024451 & .030395 & .026762 \\ -.0016623 & -.021414 & .0044444 & .030779 \\ .00015024 & .024709 & -.029871 & -.018542 \end{bmatrix}$$

c. 500 data points were used.
The maximum power of x(k) was 2.
The input noise was uniform on (-1,1).

$$H_{\lambda_0\lambda_1} = \begin{bmatrix} .47416 & .77747 & .00063628 \\ .0049366 & -.0018654 & .0025596 \\ -.020319 & -.10698 & .076991 \end{bmatrix}$$

d. 5000 data points were used.
The maximum power of x(k) was 2.
The input noise was uniform on (-1.1).

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .49480 & .75558 & .0023054 \\ .017591 & -.0011427 & .024714 \\ .025590 & .0019598 & .032117 \end{bmatrix}$$

TABLE 3.2: System II Results Using Full Matrix Inversion.

The maximum power of x(k) was 3.
The noise was uniform on (-1,1).
The convergence factor, μ, was chosen to be .2.

a. After 100 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .21191 & -.54901 & .23961 & -.58084 \\ .49735 & .55747 & -.22910 & .62263 \\ .11778 & 1.3372 & -.42404 & .85335 \\ .43573 & .41638 & -.29008 & .45359 \end{bmatrix}$$

b. After 300 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .19133 & -.45457 & -.089402 & -.69479 \\ .52472 & .65421 & .055704 & .71551 \\ .032839 & 1.2483 & -.34090 & .75055 \\ .44209 & .55546 & -.086633 & .53219 \end{bmatrix}$$

c. After 500 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .18995 & -.35146 & .088102 & -.68228 \\ .51438 & .55292 & .047074 & 66728 \\ .028302 & 1.2505 & -.28989 & .73386 \\ .43507 & .60299 & -.082106 & .56355 \end{bmatrix}$$

d. After 1000 Iterations

$$H_{\lambda_0\lambda_1} = \begin{bmatrix} .20060 & -.39481 & .028912 & -.70631 \\ .50360 & .46156 & .10901 & .74664 \\ .017045 & 1.2743 & -.32721 & .75039 \\ .43879 & 64136 & -.070640 & .64572 \end{bmatrix}$$

e. After 1800 Iterations

$$H_{\lambda_0\lambda_1} \begin{bmatrix} .19456 & -.38962 & .030956 & -.70562 \\ .48923 & .43032 & .11714 & .75506 \\ .0079652 & 1.2786 & -.33206 & .72930 \\ .43295 & .67660 & .053381 & .64097 \end{bmatrix}$$

TABLE 3.3: System I Results Using LMS Algorithm

The Maximum power of x(k) was 3.
The input noise was uniform on (-1,1).
The convergence factor, μ, was chosen to be .15.

a. After 100 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .27229 & .64437 & -.029974 & -.30266 \\ -.021558 & -.13134 & -.16022 & -.10782 \\ -.062328 & .049563 & -.069966 & -.17809 \\ .26624 & -.00231392 & -.093028 & -.048308 \end{bmatrix}$$

b. After 300 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .49886 & .96021 & -.11685 & -.88415 \\ -.010596 & .10195 & -.060408 & -.12604 \\ .036964 & .15373 & -.17940 & -.32627 \\ .14847 & .059296 & -.049165 & -.10986 \end{bmatrix}$$

c. After 500 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .54880 & 1.3925 & .010187 & -.88497 \\ -.091216 & .14549 & .10978 & -.0030069 \\ -.081292 & .23020 & -.14849 & -.33871 \\ .0091398 & .021484 & -.012741 & -.071461 \end{bmatrix}$$

d. After 1000 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .53253 & 1.5226 & -.044331 & -1.0004 \\ .13698 & -.060273 & .15947 & .036425 \\ .16666 & .16941 & -.096854 & -.35439 \\ 068181 & .018017 & .084221 & .0062654 \end{bmatrix}$$

e. After 1700 Iterations

$$H_{\lambda_0\lambda_1} = \begin{bmatrix} .48682 & 1.3719 & -.058312 & .95098 \\ .13368 & .025525 & .012328 & .037707 \\ .096072 & .20474 & -.031694 & -.21534 \\ .05853 & .047710 & -.033643 & .025036 \end{bmatrix}$$

TABLE 3.4: System II Results Using LMS Algorithm

problem. It is shown in this section that proper choices of coordinates lead to diagonal correlation matrices so that no costly matrix inversions are required to solve the normal equations. This makes the identification process almost trivial. It was stated in Section B.3 that the input **observation** vectors could be considered to describe a curve in a hypothetical (p+1)-dimensional space. The system output is estimated based on this curve (equation (3.17).) This chapter generalizes this idea to include cases where we have two curves, one depending on present and past inputs, the other depending on past outputs. Finally, the chapter concludes with several non-trivial numerical simulations.

1. Choices of Coordinate Systems

A point in (p+1)-dimensional space is defined by the variables $x^0, x^2, ..., x^P$. In Section B.3 (equation (3.14)) these variables were chosen to be parametric functions of the variable x(k), the system input. The particular choice presented there was chosen to correspond to the Volterra series. It is desirable to pick a system of coordinates that ensure a **diagonal** correlation tensor. as this allows solution of the normal equations without matrix inversion. A tensor **T** is diagonal if its components obey the rule

$$T^{\lambda_0 \cdots \lambda_N} = \begin{cases} K_\lambda & \text{for } \lambda_0 = \lambda_{\frac{N+1}{2}},\ \lambda_1 = \lambda_{\frac{N+1}{2}+1},\ \ldots\ \lambda_{\frac{N+1}{2}-1} = \lambda_N \\ 0 & \text{otherwise} \end{cases} \tag{3.44}$$

Note that only tensors of an even order (possessing an even number of indices) can possibly be diagonal. In general components satisfying the upper condition of equation (3.44) are called **diagonal elements**, or **diagonal components**. Components that are not diagonal are called **off diagonal.**

Two conditions are required in order to ensure a diagonal correlation tensor. The first is a result of the following theorem.

a. Theorem 3.1

If a set of functions $\{ f_0(x), f_1(x), ..., f_N(x) \}$ are defined with the property that

$$\int_{-\infty}^{\infty} f_\lambda(x) f_\mu(x) w(x) dx = \begin{cases} K_\lambda & \text{for } \lambda = \mu \\ 0 & \text{for } \lambda \neq \mu \end{cases} \tag{3.45}$$

where w(x) is a positive weighting function, then the set of random variables, Z^λ, defined as

$$\begin{aligned} Z^0 &= f_0(X) \\ Z^1 &= f_1(X) \\ &\vdots \\ Z^p &= f_p(X) \end{aligned} \tag{3.46}$$

where X is a random variable with probability distribution

$$p_X(x) = Cw(x) \tag{3.47}$$

are **uncorrelated**. In equation (3.47) the constant, C, and weighting function, w(x), must be chosen so that $p_X(x)$ satisfies the definition of a probability density function.

b. Proof

$$E\left\{Z^\lambda Z^\mu\right\} = E\left\{f_\lambda(X) f_\mu(X)\right\} \tag{3.48a}$$

$$= \int_{-\infty}^{\infty} f_\lambda(x) f_\mu(x) p_X(x) dx \tag{3.48b}$$

$$= \int_{-\infty}^{\infty} f_\lambda(x) f_\mu(x) C w(x) dx \tag{3.48c}$$

$$= C \int_{-\infty}^{\infty} f_\lambda(x) f_\mu(x) w(x) dx \tag{3.48d}$$

Substituting equation (3.45), yields

$$E\left\{Z^\lambda Z^\mu\right\} = \begin{cases} K_\lambda & \text{for} \lambda = \mu \\ 0 & \text{for} \lambda \neq \mu \end{cases} \tag{3.49}$$

Choosing a set of functions in accordance with equation (3.45) and ensuring that $f_0(x)$ is a constant function (ie: is a constant) is the first condition

that must be satisfied in order to obtain a diagonal correlation tensor. The system must be excited with samples drawn from a random process with probability distribution given by (3.47). In addition, if the random process is chosen to be zero mean and strictly white (ie, independent which implies uncorrelated), then the correlation tensor will be diagonal. This last condition is equivalent to requiring that

$$E\{x(m)x(n)\} = \delta(m-n) \tag{3.50a}$$

$$p(x(n),x(m)) = p(x(n))p(x(m)) \tag{3.50b}$$

and

$$E\{x(m)\} = 0 \tag{3.50c}$$

where x(m) and x(n) are two input samples taken at times m and n respectively. It is a straight forward matter to show that if these conditions are met, the correlation tensor will be diagonal.

The conditions presented above imply that different sets of coordinate functions should be used depending on the probability distribution of the noise used to excite the system. Different noise distributions have the effect of weighting the error differently. Consider that a Gaussian noise will contain samples of all amplitudes while uniform noise is bounded. If, for example, the system contains a saturation type of nonlinearity, the uniform noise may not detect its presence if its maximum amplitude is not sufficiently large. Theoretically, Gaussian noise contains samples of all amplitudes and will excite all modes. On the other hand it may be known that the system input never exceeds a certain maximum value and so a bounded input will be suitable.

If two models of a system are constructed, in two different coordinate systems but using the same input noise (only one set can possibly lead to a diagonal correlation tensor) then the two solutions will be equivalent. One solution can be transformed into the other by performing a change of coordinates

in accordance with equation (2.22). Therefore, it always makes sense to identify the system using the coordinate system which leads to a diagonal tensor. Other representations can then be calculated as desired. Solutions obtained by using different input noise density functions are not equivalent even if the coordinate systems used were the same. The effect of the different distributions is to weight the errors differently. A uniform input noise will weight the errors equally, while a Gaussian noise will emphasize the importance of errors made for small inputs. In general, transformations between solutions obtained using different excitation noise distributions cannot be found. Choosing an appropriate distribution requires knowledge of the expected system input signals.

The Hermite polynomials lead to a diagonal correlation tensor if the input is white, zero mean, Gaussian noise. Similarly, Legendre polynomials should be used in the case of uniform noise. It is convenient to normalize the coordinate functions so that the diagonal components of the correlation tensor are all ones. To identify the system parameters, only the cross-correlations of the right-hand-side need to be calculated. This fact was first discovered by Lee and Schetzen [Ref. 10].

2. Recursive Models

Recursive models have been used very successfully for modelling linear systems [Ref. 35]. Among their advantages is infinite memory, and the ability to model a system without knowledge of its input. The latter property allows these models to be employed in such areas as speech processing where input signals are difficult or impossible to measure. The assumption is made that the input is white noise.

Recursive discrete-time nonlinear expansions have also been proposed (see for example [Ref. 14,15,16]). Recursive models posses infinite memory, and so may require fewer terms to accurately represent long memory systems. However, nonlinear recursive models also posses infinite nonlinearity. To understand why this is true, consider the following example.

a. Example 3.2

Consider the following recursive, discrete, nonlinear system

$$y(k) = ay^2(k-1) + u(k) \tag{3.51}$$

where $u(k) = b\delta(k)$ (where $\delta(k)$ is a unit sample) and where a and b are arbitrary constants.

The output (y(k)) of the system for k = 1, ..., K is

$$y(0) = b \tag{3.52a}$$

$$y(1) = ab^2 \tag{3.52b}$$

$$y(2) = a(ab^2)^2 = a^3b^4 \tag{3.52c}$$

.

.

.

$$y(K) = a^{2^K-1}\, b^{2^K} \tag{3.52d}$$

It is clear that the nonlinearity of the system increases with time. Unlike a linear system, stability in a recursive nonlinear system is determined not only by the system parameter, a, but also by the input function. It is also difficult, in general, to predict for what classes of input a particular system will be stable. By analogy to the linear case we will refer to these types of models as Auto-Regressive or AR.

It is possible to also expand the output as a combination of both past and present inputs and past outputs. This type of model was proposed by Parker and Perry [Ref. 13]. We will refer to this type of model as an Auto-Regressive, Moving-Average or ARMA model. It will have the same stability problems as does the AR model.

Equation (3.17) can be used to model an AR nonlinear systems if the proper choice is made for the observation vectors. Using the same coordinate functions as given in equation (3.14) but using y(k - i), i = 1.....N, as an input parameter, yields appropriate observation vectors. That is,

$$y^{\lambda_i}(k-i) = [y(k-i)]^{(\lambda_i)} \tag{3.53}$$

defines the components of the i-th observation vector. The model equation, equivalent to equation (3.17), becomes

$$y(k) = y^{\lambda_1}(k-1)...y^{\lambda_N}(k-N)H_{\lambda_1 \cdots \lambda_N} \tag{3.54}$$

where $H_{\lambda_1 \cdots \lambda_N}$ is again a (p+1)-th order covariant tensor containing the system parameters. This model is an extension of the familiar linear autoregressive, or AR model.

The normal equations derived for the nonrecursive case can be used to solve for the $H_{\lambda_1 \cdots \lambda_N}$ tensor in this case as well. The identification process is described in Figure 3.5. The assumption is made that the system is recursive and driven by a white noise, u(k). Its output can then be described by

$$y(k) = y^{\lambda_1}(k-1)...y^{\lambda_N}(k-N)H_{\lambda_1 \cdots \lambda_N} + u(k) \tag{3.55}$$

This output signal is delayed and fed into the analysis model. The error signal e(k) is given by

$$e(k) = y(k) - \hat{y}(k) \tag{3.56a}$$

$$= y^{\lambda_1}(k-1)...y^{\lambda_N}(k-N)H_{\lambda_1 \cdots \lambda_N} + u(k) - y^{\lambda_1}(k-1)...y^{\lambda_N}(k-N)H_{\lambda_1 \cdots \lambda_N} \tag{3.56b}$$

When the model parameters exactly equal the actual system parameters the error signal will equal the input white noise. For this reason the analysis model is often called a **whitening** or **bleaching** filter. The analysis model is nonrecursive. It uses past values of the system output to make a prediction of the present system output. The normal equations (3.25c) apply to this situation with the observation vectors, x(k-i), replaced by the vectors defined in equation (3.53). The normal equations for the recursive nonlinear model can be written as

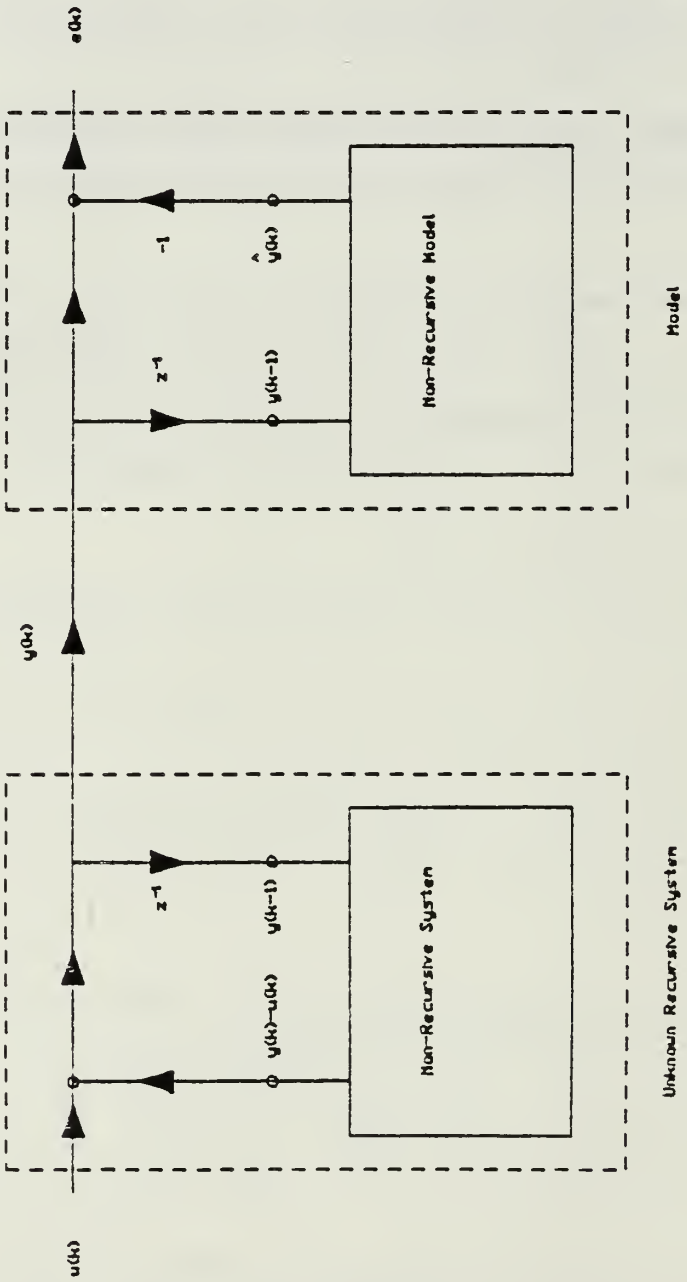


Figure 3.5: Recursive Model Identification

$$E\left\{y^{\lambda_1}(k-1)\ldots y^{\lambda_N}(k-N)y^{\mu_1}(k-1)\ldots y^{\mu_N}(k-N)\right\}H_{\lambda_1 \quad \lambda_N}$$

$$= E\left\{y(k)y^{\mu_1}(k-1)\ldots y^{\mu_N}(k-N)\right\} \tag{3.57}$$

These equations are a tensor equivalent of the Yule-Walker equations. The corresponding mean-square error is

$$E\left\{e^2(k)\right\} = E\left\{y^2(k)\right\} - E\left\{y(k)y^{\lambda_1}(k-1)\cdots y^{\lambda_N}(k-N)\right\}H_{\lambda_1 \quad \lambda_N} \tag{3.58}$$

In this case it is not obvious that one set of coordinates will yield better results than another. The correlation tensor appearing on the left-hand-side of equation (3.57) cannot be guaranteed to be diagonal since the probability distribution of y(k) cannot be controlled. Techniques must be devised which choose the coordinate system "on line" as the statistics of y(k) are determined. For example, in the linear lattice the coordinate system is chosen by orthogonalizing the sequence y(k) using a Gram-Schmidt procedure. In this way the coordinate system is not determined until run time. Extension of these ideas is left until Chapter 4.

The tensor model presented is also suitable to represent a nonlinear ARMA model. This type of model takes into account all available information (input and output) and so should be more accurate. It may also lead, in some cases, to a lower order solution than either the AR or MA model. An ARMA tensor model can be written as

$$y(k) = x^{\lambda_0}(k)\ldots x^{\lambda_M}(k-M)y^{\mu_1}(k-1)\ldots y^{\mu_N}(k-N)H_{\lambda_0 \quad \lambda_M\mu_1 \quad \mu_N} \tag{3.59}$$

Relations analogous to (3.57) and (3.58) for the normal equations and the minimum mean-square error for the ARMA model can be obtained in a straight forward manner.

3. Simulation Results

The concepts developed in Section D of this chapter were verified using FORTRAN simulations. The coordinate functions were chosen to be the

normalized Legendre polynomials because of the ease in generating uniform noise on a digital computer. By assuming that the correlation matrix was diagonal. the solution to the normal equations was determined without matrix inversion. This solution was verified by generating the correlation on the left-hand-side of the normal equations and performing the required matrix inversion. The LMS adaptive algorithm was also tested using the Legendre polynomials.

The Legendre polynomials used are given by

$$f_0(x) = 1.0 \tag{3.60a}$$

$$f_1(x) = \sqrt{3}\, x \tag{3.60b}$$

$$f_2(x) = \sqrt{\frac{45}{2}}\left(x^{(2)} - \frac{1}{3}\right) \tag{3.60c}$$

$$f_3(x) = \sqrt{\frac{175}{2}}\left(x^{(3)} - \frac{3}{5}x\right) \tag{3.60d}$$

These functions are normalized so that the correlation tensor will have ones along the diagonal when excited with zero mean white noise which is uniform on (-1.1).

The simulation models used were similar to those used in Section C. The coefficients for System I were identical to those given in (3.42), although this time they are coefficients of Legendre polynomials so they do not represent the same system. System II was again a unit step function which has a tensor representation, in terms of the Legendre polynomials given by [Ref. 26: p. 1526],

$$\left[H_{\lambda_0}\right] = \left[\frac{1}{2} \quad \frac{\sqrt{3}}{4} \quad 0 \quad -\frac{\sqrt{175}}{80}\right] \tag{3.61a}$$

$$= \left[0.5 \quad 0.433013 \quad 0.0 \quad 0.165359\right] \tag{3.61b}$$

The results of the simulations for System I are presented in Tables 3.5 and 3.6. Results for System II are given in Tables 3.7 and 3.8. In all cases it is obvious that the simulation results are very good.

15000 sample data points were used.
The maximum power of x(k) was 3.

a. Solution Without Using Matrix Inversion

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .19158 & -.38475 & .0041187 & -.67056 \\ .49067 & .38457 & .089293 & .93948 \\ -.012253 & 1.2925 & -.35452 & .71923 \\ .41049 & .85202 & -.074247 & .43011 \end{bmatrix}$$

b. Solution Using Matrix Inversion

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .20000 & -.40000 & .030000 & -.70000 \\ .50000 & .35000 & .11000 & .90000 \\ .01000 & 1.3000 & -.33000 & .70000 \\ .4[illegible]000 & .81000 & -.050000 & .40000 \end{bmatrix}$$

TABLE 3.5: System I Model Parameters Obtained
a. Using No Matrix Inversion, and
b. Using Matrix Inversion.

The maximum power of x(k) was 3.
The convergence factor was chosen to be .01.

a. After 200 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .14678 & -.45852 & -.036398 & -.72277 \\ .42720 & .25876 & .018320 & .83079 \\ -.077103 & 1.1898 & -.45388 & .61647 \\ .34264 & .68052 & -.16955 & .30546 \end{bmatrix}$$

b. After 500 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .20013 & -.40047 & .030509 & -.70066 \\ .50038 & .35077 & .11136 & .90025 \\ .0099732 & 1.2996 & -.32958 & .70026 \\ .43008 & .81014 & -.050063 & .39875 \end{bmatrix}$$

c. After 1000 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .20000 & -.40000 & .030007 & -.70000 \\ .50001 & .35000 & .11001 & .8999 \\ .010008 & 1.3000 & -.32999 & .69999 \\ .43001 & .81000 & -.049986 & .40000 \end{bmatrix}$$

TABLE 3.6: System I Model Parameters Using LMS Algorithma.

a. 500 data points were used.
The maximum power of x(k) was 3.

No Matrix Inversion Solution

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .52104 & .43930 & .0090070 & -.17515 \\ -.011098 & -.0046220 & .014464 & .020122 \\ .0011160 & -.010606 & -.010466 & .021310 \\ .040052 & -.0071009 & .058263 & .052566 \end{bmatrix}$$

Using Matrix Inversion

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .50157 & .43054 & -.011153 & -.16530 \\ -.0084508 & .0032387 & -.0000970 & -.0006363 \\ .0017041 & -.0009167 & .0062532 & -.0059126 \\ -.0066652 & -.0024676 & .012192 & .0014058 \end{bmatrix}$$

b. 15000 data point were used.
The maximum power of x(k) was 3.

No Matrix Inversion Solution

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .50050 & .43363 & -.0043279 & -.16772 \\ .0015150 & .0003272 & -.0012861 & -.0003738 \\ -.0071103 & -.0004279 & .012933 & .014358 \\ .0022021 & .0018209 & -.0021332 & -.0060674 \end{bmatrix}$$

Using Matrix Inversion

$$H_{\lambda_0\lambda_1} \begin{bmatrix} .50089 & .43306 & -.0001744 & -.16722 \\ .0018213 & -.0005741 & -.0014410 & -.0000789 \\ .0036401 & .0008143 & .0043811 & -.0008015 \\ .0003990 & .0001746 & -.0003974 & .0006467 \end{bmatrix}$$

TABLE 3.7: System II Model Parameters Obtained
a. Using No Matrix Inversion, and
b. Using Matrix Inversion.

The maximum power of x(k) was 3.
The convergence factor, μ, was chosen to be .005.

a. After 300 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .46464 & .41356 & .0066999 & -.16385 \\ .011790 & .012146 & .0067764 & -.0093346 \\ -.010943 & .014180 & .020212 & .018152 \\ .0014149 & -.0065302 & .0093879 & -.0047952 \end{bmatrix}$$

b. After 500 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .50556 & .43020 & -.00098824 & -.16796 \\ -.015344 & -.010684 & .020836 & .0098449 \\ -.0082813 & -.0052115 & .0090727 & -.010251 \\ .012877 & -.0094530 & -.0046525 & .0072316 \end{bmatrix}$$

c. After 1000 Iterations

$$[H_{\lambda_0\lambda_1}] = \begin{bmatrix} .49868 & .44629 & .0053940 & -.17370 \\ .020577 & .0091397 & -.010215 & .0065185 \\ .012857 & .020072 & -.0064996 & -.010541 \\ -.0020197 & .0031219 & -.0064275 & -.0037119 \end{bmatrix}$$

c. After 1800 Iterations

$$H_{\lambda_0\lambda_1} \begin{bmatrix} .50115 & .43727 & -.0059911 & -.16508 \\ .0000129 & -.0063172 & -.012713 & .010275 \\ .0065666 & .0084999 & -.0068403 & -.010135 \\ .0075634 & .0058037 & -.15156 & .018235 \end{bmatrix}$$

TABLE 3.8: System II Model Parameters Using LMS Algorithm

IV. REVIEW OF LATTICE FILTER STRUCTURES

This chapter reviews lattice filter theory. These filters were first proposed in connection with the linear prediction of speech by Itakura and Saito [Ref. 36] in 1971. They developed a new approach based on a partial correlation (PARCOR) coefficient. Since that time the filter structure they proposed has come to be known as a Lattice (or sometimes also called Ladder) filter. The properties of the filter have been exhaustively studied by many researchers [Ref. 37]. Lattice filters have been successfully applied to problems in various disciplines.

The great interest in the lattice approach stems from it's property of orthogonality. This property allows the filter to be updated in order, without recalculation of all the previous, lower order, filter coefficients. Orthogonality also leads to a nice physical structure, a cascade of first order sections, and so is appropriate for efficient hardware or software implementations. Finally, it has been shown that the lattice owes its robust numerical behaviour to this property of orthogonality [Ref. 38: pp. 128-136].

This chapter begins with a derivation of the one-dimensional (1-D) lattice. The approach taken in the derivation is somewhat novel in that it begins by expressing the linear prediction in terms of an uncorrelated error sequence. This is regarded as a change of coordinate systems and used to develop the Levinson algorithm and the lattice filter. In the following section generalized forms of the Levinson algorithm and lattice are derived. It is shown that these also correspond to coordinate transformation. Next, the Schur algorithm, which is a method for generating the required filter coefficients (Lattice parameters) directly, given only a knowledge of the correlation matrix, is reviewed. In the following chapter the generalized lattice filter is used to develop new, multidimensional (specifically 2-D) lattice filters. In Chapter 6, a new nonlinear lattice filter, based upon this generalized lattice formulation, is presented.

A. 1-D LINEAR AUTOREGRESSIVE LATTICE FILTER

We define the N-th order forward error sequence of an autoregressive model as

$$e^N(k) = y(k) - \sum_{\lambda=1}^{N} h_\lambda^N \, y(k-\lambda) \tag{4.1}$$

This can equivalently be written in tensor notation as,

$$e^N(k) = h_\lambda^N y^\lambda \tag{4.2}$$

where

$$[y^\lambda]^T = [y(k) \;\; y(k-1) \;\; \cdots \;\; y(k-K)] \tag{4.3}$$

and

$$[h_\lambda^N] = [1 \;\; -h_1^N \;\; -h_2^N \;\; \cdots \;\; -h_N^N \; 0 \ldots 0] \tag{4.4a}$$

where for convenience, we have made all vectors of length K+1 (ie; $\lambda = 0, \ldots, K$), where K is some arbitrary maximum length. Note that,

$$h_0^N = 1 \tag{4.4b}$$

The y^λ can be considered to comprise a coordinate system in an K+1 dimensional space. The vector $[y^\lambda]$ represents a single realization from a random process. As mentioned in the Chapter I, there will in general be many such vectors corresponding to all the possible realizations of the random process.

Because the error, $e^N(k)$, is a scalar (invariant) it must remain unaltered regardless of the choice of coordinate system. We may, therefore, write

$$e^N(k) = K_\lambda^N y^{\lambda'} \tag{4.5}$$

where the $y^{\lambda'}$ are defined as

$$[y^{\lambda'}] = \begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) - \tilde{h}_0^1 y(k-1) \\ y(k-3) - \tilde{h}_1^2 y(k-2) \tilde{h}_0^2 y(k-1) \\ \cdot \\ \cdot \\ \cdot \\ y(k-K) - \tilde{h}_{K-2}^{K-1} y(k-K+1) - \cdots - \tilde{h}_0^{K-1} y(k-1) \end{bmatrix} \tag{4.6}$$

and

$$[K_\lambda^N] = [1 \ -K_1 \ -K_2 \cdots -K_N \ 0...0] \tag{4.7}$$

where the $\tilde{h}$'s are chosen so that the components, $y^{\lambda'}$, are uncorrelated. The stochastic form of the Gram-Schmidt procedure can be used. A discussion of this method can be found in [Ref. 39: pp. 382-383]. By uncorrelated we mean

$$E\left\{y^{\lambda'} y^{\mu'}\right\} = \begin{cases} C_{\lambda'} & \text{for } \lambda' = \mu' \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

The reader familiar with lattice structures will recognize the components of $y^{\lambda'}$ as the backward prediction errors. That is they are the errors in predicting y(k-N) from the next N-1 values: y(k-N+1), ..., y(k-1).

It is a straight forward matter to solve the prediction problem given by equation (4.5) (ie; solve for the K's) because of the uncorrelatedness of the chosen coordinate system. Using an approach similar to that presented in Chapter III. Section C. a set of normal equations can be formulated for this problem. In this case. however. the correlation matrix is diagonal so that there is no inversion necessary to obtain the solution. Minimizing the mean square value of the error. $e^N(k)$. given by (4.5). with respect to the K_λ^N's. yields

$$K_\lambda^N = \left| 1 \quad \frac{E\left\{y(k)y^1\right\}}{E\left\{(y^1)^{(2)}\right\}} \quad \cdots \quad \frac{E\left\{y(k)y^N\right\}}{E\left\{(y^N)^{(2)}\right\}} \quad 0 \cdots 0 \right| \tag{4.9}$$

Having obtained a solution to the orthogonal problem. we wonder if it cannot be employed to advantage to simplify the calculations required to solve the

original autoregressive problem. From equation (2.13), we know that the relationship between the old and new components can be expressed as

$$y^{\lambda'} = \frac{\partial y^{\lambda'}}{\partial y^{\lambda}} y^{\lambda} \tag{4.10}$$

where

$$\left[\frac{\partial y^{\lambda'}}{\partial y^{\lambda}}\right] = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -\tilde{h}_0^1 & 1 & \cdots & 0 & 0 \\ 0 & -\tilde{h}_0^2 & -\tilde{h}_1^2 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ 0 & -\tilde{h}_0^{K-2} & -\tilde{h}_1^{K-2} & \cdots & 1 & 0 \\ 0 & -\tilde{h}_0^{K-1} & -\tilde{h}_1^{K-1} & \cdots & -\tilde{h}_{K-2}^{K-1} & 1 \end{bmatrix} \tag{4.11}$$

Now, since

$$e^N(k) = y^{\lambda} h_{\lambda}^N = y^{\lambda'} K_{\lambda'}^N \tag{4.12a}$$

$$= y^{\lambda} \frac{\partial y^{\lambda'}}{\partial y^{\lambda}} K_{\lambda'}^N \tag{4.12b}$$

then

$$h_{\lambda}^N = K_{\lambda'}^N \frac{\partial y^{\lambda'}}{\partial y^{\lambda}} \tag{4.13}$$

Equation (4.13) gives the relationship between the autoregressive model parameters and the $K_{\lambda'}^N$'s. This result will be used in the proof of Levinson's algorithm.

1. Levinson-Durbin Algorithm

In 1947 in his classic paper [Ref. 40] Levinson developed a method for recursively solving the normal equations. Beginning with a zero order solution successively higher order solutions are calculated. This algorithm can be used to exploit the Toeplitz nature of correlation matrices of stationary random processes in order to reduce the required number of computations. The algorithm as presented in this work is actually a simplified version of Levinson's original from.

It assumes that the equations being solved are the Yule-Walker equations ([Ref. 41], see also Chapter 3. Section D) so that the right-hand-side of the equations contains only terms that will be present in the correlation matrix (left-hand-side) of the next higher order model. This simplification is due to Durbin [Ref. 42]. We will refer to this algorithm often simply as the Levinson algorithm, however, Durbin's contribution is acknowledged.

a. Theorem 4.1 Levinson's Algorithm (Durbin's Form)

The autoregressive model parameters may be calculated recursively from the relation

$$[h_\lambda^{N+1}] = [h_\lambda^N] - K_{N+1}\mathbf{S}[\tilde{h}_\lambda^N] \tag{4.14a}$$

where

$$[\tilde{h}_\lambda^N] = [-\tilde{h}_0^N \ -\tilde{h}_1^N \cdots -\tilde{h}_{N-1}^N \ 1 \ 0 \cdots 0] \tag{4.14b}$$

and

$$\mathbf{S}[\tilde{h}_\lambda^N] = [0 \ -\tilde{h}_0^N \ -\tilde{h}_1^N \cdots -\tilde{h}_{N-1}^N \ 1 \ 0 \cdots 0] \tag{4.14c}$$

The operator S has the effect of shifting the components. $\tilde{h}_\lambda^N$ one position to the right. Note that

$$\tilde{h}_N^N = 1$$

For stationary processes the following simplification applies

$$\tilde{h}_\lambda^N = \tilde{h}_{N-\lambda-1}^N \quad \text{for } \lambda = 0, \ldots, N \tag{4.14d}$$

b. Proof (by Induction)

Using equation (4.13) for the first order model we have

$$h_\lambda^1 = h_0^1 \ h_1^1 \ 0 \cdots 0 \tag{4.15a}$$

$$= 1 \quad K_1 \ 0 \cdots 0 \tag{4.15b}$$

$$= 1 \ 0 \cdots 0 \quad K_1 \ 0 \ 1 \ 0 \cdots 0 \tag{4.15c}$$

$$= [h_\lambda^0] - K_1\mathbf{S}[\tilde{h}_\lambda^0] \tag{4.15d}$$

so that equation (4.14a) holds for N=1. We assume the recursion holds for the N-th order model

$$[h_\lambda^N] = [h_\lambda^{N-1}] - K_N \mathbf{S}[\tilde{h}_\lambda^{N-1}] \tag{4.16}$$

From equation (4.13) we have

$$h_\lambda^N = K_\lambda^N \frac{\partial y^{\lambda'}}{\partial y^\lambda} \qquad \lambda, \lambda' = 0, ..., K \tag{4.17}$$

Therefore,

$$[h_\lambda^N] = \begin{bmatrix} 1 \\ -K_1 + K_2\tilde{h}_0^1 + K_3\tilde{h}_0^2 + ... + K_N\tilde{h}_0^{N-1} \\ -K_2 + K_3\tilde{h}_1^2 + K_4\tilde{h}_1^3 + ... + K_N\tilde{h}_1^{N-2} \\ \cdot \\ \cdot \\ \cdot \\ -K_{N-1} + K_N\tilde{h}_{N-2}^{N-1} \\ -K_N \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}^T \tag{4.18}$$

Now, for the (N+1)-st order model

$$h_\lambda^{N+1} = \begin{bmatrix} 1 \\ -K_1 + K_2\tilde{h}_0^1 + K_3\tilde{h}_0^2 + ... + K_{N+1}\tilde{h}_0^N \\ -K_2 + K_3\tilde{h}_1^2 + K_4\tilde{h}_1^3 + ... + K_{N+1}\tilde{h}_1^N \\ \cdot \\ \cdot \\ \cdot \\ -K_N - K_{N-1}\tilde{h}_{N-1}^N \\ -K_{N+1} \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}^T \tag{4.19a}$$

$$= \begin{bmatrix} 1 \\ -K_1 + K_2\tilde{h}_0^1 + K_3\tilde{h}_0^2 + \ldots + K_N\tilde{h}_0^{N-1} \\ -K_2 + K_3\tilde{h}_1^2 + K_4\tilde{h}_1^3 + \ldots + K_N\tilde{h}_1^{N-1} \\ \cdot \\ \cdot \\ \cdot \\ -K_N \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}^T - K_{N+1} \begin{bmatrix} 0 \\ -\tilde{h}_0^N \\ -\tilde{h}_1^N \\ \cdot \\ \cdot \\ \cdot \\ -\tilde{h}_{N-1}^N \\ 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}^T \tag{4.19b}$$

$$= h_\lambda^N - K_{N+1}\mathbf{S}[\tilde{h}_\lambda^N] \tag{4.19c}$$

And so the desired result has been confirmed.

The proof presented does not require that the process be stationary and so the condition of (4.14c) is not necessary. One is then faced with the problem of how to determine the $\tilde{h}_\lambda^N$'s. This question is answered in the next section where a generalized form of the algorithm is derived. We note that the condition of (4.14c) implies the following

$$h_N^N = K_N = \tilde{h}_0^N \tag{4.20}$$

or that the second column of the matrix given in equation (4.11) contains all the lattice coefficients.

The following significant observation about equation (4.14a) is made. The transpose of the term within the second bracket on the right hand side corresponds exactly with the rows of the coordinate transformation matrix of equation (4.11). The Levinson algorithm, therefore, represents a recursive method for finding an orthogonalizing coordinate transformation.

2. The 1-D Lattice Structure

It has been shown, in the previous section, that the autoregressive model parameters can be calculated in a recursive manner using the Levinson algorithm. It was also shown that a model could be built in an orthogonal coordinate system where the model parameters were given by the K_λ's. The question arises, can a filter structure be devised which represents y(k) in the orthogonal coordinate system? The answer is affirmative. It will be shown in this section that the Lattice filter is the required structure for the stationary case. A more general solution is presented in Section B of this chapter.

The desired result is obtained in a straight forward manner by multiplying both sides of equation (4.14a) by y^λ. This yields

$$e^{N+1}(k) = e^N(k) - K_{N+1}r^N(k-N-1) \tag{4.21}$$

where the quantity $r^N(k-N-1) = y^{\lambda'}$ evaluated at $\lambda' = N+1$. As was mentioned earlier in this chapter, the quantity $r^N(k-N-1)$, is generally known as the backward prediction error since it corresponds to the prediction of the point y(k-N-1) from the N future points y(k-1). ....y(k-N).

Assuming stationarity, we can use (4.14c) and (4.14a) to obtain

$$\tilde{h}_\lambda^{N+1} = S[\tilde{h}_\lambda^N] - K_{N+1}\, h_\lambda^N \tag{4.22}$$

which leads directly to the equation

$$r^{N+1}(k-N-1) = r^N(k-N-1) - K_{N+1}e^N(k) \tag{4.23}$$

Equations (4.21) and (4.23) define the Lattice form of the whitening filter (see Chapter 3, Section D). This is also sometimes referred to as the analysis model. The structure is illustrated in Figure 4.1.

In order to develop a synthesis model we need only to solve equation (4.21) for $e^N(k)$ since we know that $e^0(k)$ is equivalent to $y(k)$. Therefore, the synthesis equations are

$$e^N(k) = e^{N+1}(k) + K_{N+1}r^N(k-N-1) \tag{4.24a}$$

$$r^{N+1}(k-N-1) = r^N(k-N-1) - K_{N+1}e^N(k) \tag{4.24b}$$

The resulting structure is as illustrated in Figure 4.2. Note that in this model, $y(k)$ is indeed being expressed as a weighted sum of the backwards errors, which represent an orthogonal coordinate system.

This concludes the discussion of the classical Lattice formulation.

B. GENERALIZED ORDER UPDATE RECURSIONS

In this section a more general linear prediction problem is considered. No assumptions are made as to the origin of the data. In fact, the data need not represent a time series at all, and certainly shift invariance is not required. The ordering of the data is simply chosen in some convenient fashion. The generality of this formulation will allow its application to multidimensional and nonlinear problems.

1. Definitions and Formulation

In this section quantities are defined that will be required to complete the statements and proofs contained in the remainder of the chapter.

A realization of the random process **Y** is given by the vector $\{y^\lambda\}$, $0 \leq \lambda \leq K$.

The error, e^N_{k-1}, in predicting the element y^{k+1} from the previous N elements of **Y** is given by

$$e^N_{k+1} = h^N_\lambda(k+1)y^\lambda \quad \text{for } \lambda = 0, \ldots, K \tag{4.25}$$

where

$$h^N_\lambda(k+1) = [0 \cdots 0 \; -h^N_{k-N-1} \; -h^N_{k-N+2} \cdots -h^N_k \; 1 \; 0 \cdots 0] \tag{4.26}$$

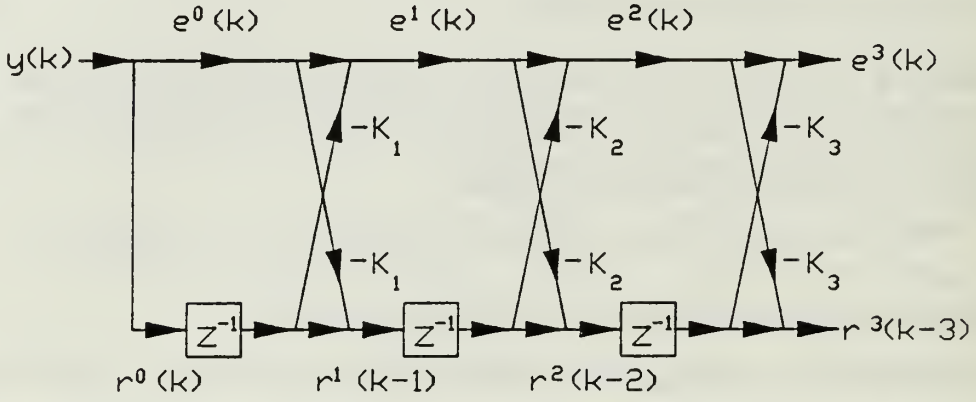


Figure 4.1: 1-D Lattice Analysis Model.

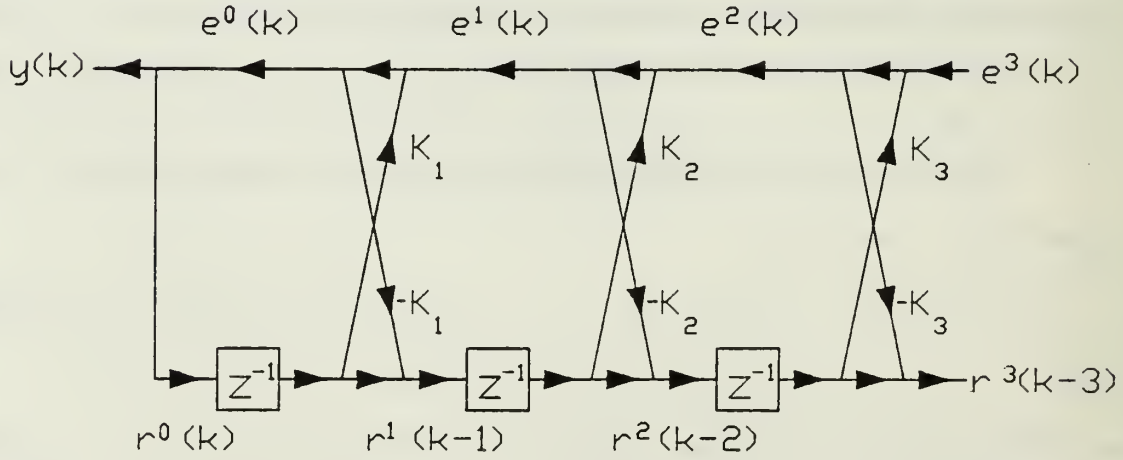


Figure 4.2: 1-D Lattice Synthesis Model.

The norm of this prediction error is given by

$$||e_{k+1}^N|| = \left[E\left\{(e_{k-1}^N)^2\right\} \right]^{1/2} \tag{4.27}$$

A normalized version of the forward error is given by

$$\bar{e}_{k+1}^N = \frac{e_{k+1}^N}{||e_{k+1}^N||} \tag{4.28a}$$

$$= a_\lambda^N(k+1)y^\lambda \tag{4.28b}$$

where

$$a_\lambda^N(k+1) = \frac{1}{||e_{k+1}^N||} h_\lambda^N(k+1) \tag{4.29}$$

The backwards prediction error, r_{k-N}^N, is the error associated with the prediction of y^{k-N} given the next N elements of **Y**. It is given by,

$$r_{k-N}^N = \tilde{h}_\lambda^N(k-N)y^\lambda \tag{4.30}$$

where

$$[\tilde{h}_\lambda^N(k-N)] = [0 \cdots 0 \ 1 \ -\hat{h}_{k-N+1}^N \ -\tilde{h}_{k-N+2}^N \cdots -\tilde{h}_k^N \ 0 \cdots 0] \tag{4.31}$$

Again, a normalized version can be defined as

$$\bar{r}_{k-N}^N = \frac{r_{k-N}^N}{|r_{k-N}^N|} \tag{4.32a}$$

$$= b_\lambda^N(k-N)y^\lambda \tag{4.32b}$$

where

$$b_\lambda^N = \frac{\tilde{h}_\lambda^N(k-N)}{r_{k-N}^N} \tag{4.33}$$

and

$$||r_{k-N}^N|| = \left[E\left\{(r_{k-N}^N)^2\right\} \right]^{1/2} \tag{4.34}$$

In order to generalize the results of Section A of this chapter we need to introduce two different families of coordinate systems. We will refer to members of these families as either forward or backward local coordinates. The term local is used because a different coordinate system will be associated with every value of k and N. We define the (k+1) - indexed, N-th order forward coordinate system as

$$
\left[y^{\lambda'}(k+1,N)\right] = \begin{bmatrix} y^0 \\ y^1 \\ \cdot \\ \cdot \\ \cdot \\ y^{k-N} \\ y^{k-N+1} - \tilde{h}_{k-N+2}^{N-1} y^{k-N+2} - \ldots - \tilde{h}_k^{N-1} y^k \\ \cdot \\ \cdot \\ \cdot \\ y^{k-1} - \tilde{h}_k^1 y^k \\ y^k \\ y^{k+1} \\ \cdot \\ \cdot \\ \cdot \\ y^K \end{bmatrix} \tag{4.35}
$$

The corresponding coordinate transformation from the unprimed coordinate system to this local forward coordinate system is given by

$$
\left[\frac{\partial y^{\lambda'}(k-1,N)}{\partial y^{\lambda}} \right] =
$$

$$\left[\begin{array}{c|c|ccccc|c} \mathbf{I} & & & \mathbf{0} & & & & \mathbf{0} \\ \hline & 1 & -\tilde{h}_{k-N+2}^{N-1} & \tilde{h}_{k-N+3}^{N-1} & \cdots & -\tilde{h}_{k-1}^{N-1} & -\tilde{h}_{k}^{N-1} & \\ & 0 & 1 & -\tilde{h}_{k-N+3}^{N-2} & \cdots & -\tilde{h}_{k-1}^{N-2} & -\tilde{h}_{k}^{N-2} & \\ & 0 & 0 & 1 & \cdots & -\tilde{h}_{k-1}^{N-3} & -\tilde{h}_{k}^{N-3} & \\ & \cdot & \cdot & \cdot & & \cdot & \cdot & \\ \mathbf{0} & \cdot & \cdot & \cdot & & \cdot & \cdot & \mathbf{0} \\ & \cdot & \cdot & \cdot & & \cdot & \cdot & \\ & 0 & 0 & 0 & \cdots & 1 & -\tilde{h}_{k}^{1} & \\ & 0 & 0 & 0 & \cdots & 0 & 1 & \\ \hline \mathbf{0} & & & \mathbf{0} & & & & \mathbf{I} \end{array}\right] \tag{4.36}$$

where the **0**'s are zero matrices and the **I**'s represent identity matrices.

Similarly, we define the (k-N)-indexed, N-th order backward coordinate system as

$$y^{\lambda''}(k-N,N) = \begin{bmatrix} y^0 \\ y^1 \\ \cdot \\ \cdot \\ y^{k-N} \\ y^{k-N-1} \\ y^{k-N+2} - h_{k-N+1}^{1} y^{k-N+1} \\ \cdot \\ \cdot \\ \cdot \\ y^{k} - h_{k-1}^{N-1} y^{k-1} - \ldots - h_{k-N-1}^{N-1} y^{k-N+1} \\ y^{k+1} \\ \cdot \\ \cdot \\ \cdot \\ y^{K} \end{bmatrix} \tag{4.37}$$

The corresponding coordinate transformation is given by

$$\left[\frac{\partial y^{\lambda''}(k-N,N)}{\partial y^{\lambda}}\right] =$$

$$\left[\begin{array}{c|cccccc|c} \mathbf{I} & & & \mathbf{0} & & & & \mathbf{0} \\ \hline & 1 & 0 & 0 & \cdots & 0 & 0 & \\ & -h^{1}_{k-N+1} & 1 & 0 & \cdots & 0 & 0 & \\ & -h^{2}_{k-N+1} & -h^{2}_{k-N+2} & 1 & \cdots & 0 & 0 & \\ \mathbf{0} & \cdot & \cdot & \cdot & & \cdot & \cdot & \mathbf{0} \\ & \cdot & \cdot & \cdot & & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & & \cdot & \cdot & \\ & -h^{N-2}_{k-N+1} & -h^{N-2}_{k-N+2} & -h^{N-2}_{k-N+3} & \cdots & 1 & 0 & \\ & -h^{N-1}_{k-N+1} & -h^{N-1}_{k-N+2} & -h^{N-1}_{k-N+3} & \cdots & -h^{N-1}_{k-1} & 1 & \\ \hline \mathbf{0} & & & \mathbf{0} & & & & \mathbf{I} \end{array}\right] \tag{4.38}$$

These definitions are sufficient to state and prove the theorems presented in the remainder of this chapter.

2. Generalized Levinson Algorithm

The Levinson algorithm (equation (4.14)) can be extended to recursively compute the forward and backward prediction coefficients defined in equations (4.26) and (4.31). In this section two forms of the algorithm are presented and proved. First a non-normalized version is introduced, then using this result a normalized algorithm is developed.

a. Theorem 4.2: Generalized Levinson Recursion (Regular Form)

The forward and backward prediction coefficients defined in equations (4.26) and (4.31) can be updated using the following recursive equations.

$$\begin{bmatrix} h^{N+1}_{\lambda}(k+1) \\ \tilde{h}^{N+1}_{\lambda}(k-N) \end{bmatrix} = \begin{bmatrix} h^{N}_{\lambda}(k+1) \\ \tilde{h}^{N}_{\lambda}(k-N) \end{bmatrix} - \rho^{k}_{N+1} \begin{bmatrix} \tilde{h}^{N}_{\lambda}(k-N) \\ h^{N}_{\lambda}(k+1) \end{bmatrix} \frac{\begin{bmatrix} e^{N}_{k+1} \\ r^{N}_{k-N} \end{bmatrix}}{\begin{bmatrix} r^{N}_{k-N} \\ e^{N}_{k+1} \end{bmatrix}} \tag{4.39a}$$

$$\begin{bmatrix} h^{N+1}_{\lambda}(k-N) \\ \tilde{h}^{N+1}_{\lambda}(k-N) \end{bmatrix} = \begin{bmatrix} h^{N}_{\lambda}(k-N) \\ \tilde{h}^{N}_{\lambda}(k-N) \end{bmatrix} - \rho^{k}_{N+1} \begin{bmatrix} \tilde{h}^{N}_{\lambda}(k-1) \\ h^{N}_{\lambda}(k-N) \end{bmatrix} \frac{\begin{bmatrix} r^{N}_{k-N} \\ e^{N}_{k+1} \end{bmatrix}}{\begin{bmatrix} e^{N}_{k+1} \\ r^{N}_{k-N} \end{bmatrix}} \tag{4.39b}$$

where

$$\rho^{k}_{N+1} = E\left\{\bar{e}^{N}_{k+1}\bar{r}^{N}_{k-N}\right\} \tag{4.40}$$

b. Proof

The forward and backward prediction errors are scalars so their representation is identical in all frames of reference. Therefore, we can write

$$e_{k+1}^{N} = h_{\lambda}^{N}(k+1)y^{\lambda} \tag{4.41a}$$

$$= K_{\lambda'}^{N}(k+1)y^{\lambda'}(k+1,N) \tag{4.41b}$$

and

$$r_{k-N}^{N} = h_{\lambda}^{N}(k-N)y^{\lambda} \tag{4.42a}$$

$$= K_{\lambda''}^{N}(k-N)y^{\lambda''}(k-N,N) \tag{4.42b}$$

where

$$K_{\lambda'}^{N}(k+1) = h_{\lambda}^{N}(k+1)\frac{\partial y^{\lambda}}{\partial y^{\lambda'}(k+1,N)} \tag{4.43a}$$

$$= [0 \cdots 0 \; -K_{k-N+1}^{N} \cdots -K_{k}^{N} \; 1 \; 0 \cdots 0] \tag{4.43b}$$

$$h_{\lambda}^{N}(k+1) = K_{\lambda'}^{N}(k+1)\frac{\partial y^{\lambda'}(k+1,N)}{\partial y^{\lambda}} \tag{4.44}$$

and

$$\tilde{K}_{\lambda''}^{N}(k-N) = \tilde{h}_{\lambda}^{N}(k-N)\frac{\partial y^{\lambda}}{\partial y^{\lambda''}(k-N,N)} \tag{4.45a}$$

$$= [0 \cdots 0 \; 1 \; -\tilde{K}_{k-N+1}^{N} \cdots -\tilde{K}_{k}^{N} \; 0 \cdots 0] \tag{4.45b}$$

$$\tilde{h}_{\lambda}^{N}(k-N) = \tilde{K}_{\lambda''}^{N}(k-N)\frac{\partial y^{\lambda''}(k-N,N)}{\partial y^{\lambda}} \tag{4.46}$$

The normal equations in the primed and unprimed coordinate systems can be solved for $K_{\lambda}^{N}(k-1)$ and $K_{\lambda''}^{N}(k-N)$. This is once again (see Section A of this chapter) a straight forward matter because the correlation matrices. $[E\{y^{\lambda'}(k+1,N)y^{\mu'}(k+1,N)\}]$ and $[E\{y^{\lambda''}(k-N,N)y^{\mu''}(k-N,N)\}]$. will contain only diagonal elements. The solutions are

$K_{\lambda'}^{N}(k+1) =$

$$\left[0 \cdots 0 \quad \frac{E\left\{y(k-1)y^{k-N-1}(k+1,N)\right\}}{E\left\{(y^{k-N+1}(k+1,N))^2\right\}} \quad \cdots \quad \frac{E\left\{y(k+1)y^{k}(k-1,N)\right\}}{E\left\{(y^{k}(k-1.N))^2\right\}} \quad 1 \quad 0 \cdots 0 \right]$$

(4.47)

$\tilde{K}_{\lambda'}^{N}(k-N) =$

$$\left[0 \cdots 0 \quad 1 \quad \frac{E\left\{y(k-N)y^{k-N+1}(k-N,N)\right\}}{E\left\{(y^{k-N+1}(k-N,N))^2\right\}} \quad \cdots \quad \frac{E\left\{y(k-N)y^{k}(k-N,N)\right\}}{E\left\{(y^{k}(k-N,N))^2\right\}} \quad 0 \cdots 0 \right]$$

(4.48)

Consider the (k-N)-th term of the (N+1) order version of (4.47).

$$K_{k-N}^{N+1}(k+1) = \frac{E\left\{y(k+1)y^{k-N}(k+1,N)\right\}}{E\left\{(y^{k-N}(k-1,N))^2\right\}} \tag{4.49a}$$

$$= \frac{E\left\{y(k-1)r_{k-N}^{N}\right\}}{E\left\{(r_{k-N}^{N})^2\right\}} \tag{4.49b}$$

$$= \frac{E\left\{e_{k-1}^{N}r_{k-N}^{N}\right\}}{E\left\{(r_{k-N}^{N})^2\right\}} \tag{4.49c}$$

$$= \frac{e_{k+1}^{N}}{r_{k-N}^{N}}\rho_{N-1}^{k} \tag{4.49d}$$

Similarly,

$$\tilde{K}_{k+1}^{N+1}(k-N) = \frac{\| r_{k-N}^{N} \|}{\| e_{k+1}^{N} \|} \rho_{N+1}^{k} \tag{4.50}$$

For the N=1 model the forward error prediction coefficients are (from equation (4.44))

$$[h_{\lambda}^{1}(k+1)] = K_{\lambda'}^{1}(k+1) \frac{\partial y^{\lambda'}(k+1,N)}{\partial y^{\lambda}} \tag{4.51a}$$

$$= [0 \cdots 0 \ -K_k^1 \ 1 \ 0 \cdots 0] \tag{4.51b}$$

$$= [0 \cdots 0 \ \frac{\| e_{k+1}^{0} \|}{\| r_k^0 \|} \rho_1^k \ 1 \ 0 \cdots 0] \tag{4.51c}$$

Equation (4.39a) yields

$$[h_{\lambda}^{1}(k+1)] = [h_{\lambda}^{0}(k+1)] - \rho_1^k [\tilde{h}_{\lambda}^{0}(k)] \frac{\| e_{k+1}^{0} \|}{\| r_k^0 \|} \tag{4.52a}$$

$$= [0 \cdots 0 \ 1 \ 0 \cdots 0] - \rho_1^k \frac{\| e_{k+1}^{0} \|}{\| r_k^0 \|} [0 \cdots 0 \ 1 \ 0 \cdots 0 \tag{4.52b}$$

$$= [0 \cdots 0 \ -\rho_1^k \frac{\| e_{k+1}^{0} \|}{\| r_k^0 \|} \ 1 \ 0 \cdots \tag{4.52c}$$

Therefore, the recursion of equation (4.39a) holds for N=1. We assume it is valid for N and verify it for N+1. From equation (4.44) we have

$$h_{\lambda}^{N+1}(k-1) = K_{\lambda'}^{N+1}(k-1) \frac{\partial y^{\lambda'}(k-1,N)}{\partial y^{\lambda}} \tag{4.53a}$$

$$= \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ -K_{k-N}^{N+1}(k+1) \\ -K_{k-N+1}^{N+1}(k+1) + \tilde{h}_{k-N+1}^{N} K_{k-N}^{N+1}(k+1) \\ \cdot \\ \cdot \\ \cdot \\ -K_{k}^{N+1}(k+1) + \tilde{h}_{k}^{1} K_{k-1}^{N+1}(k+1) + \cdots + \tilde{h}_{k}^{N} K_{k-N}^{N+1}(k+1) \\ 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}^{T} \qquad \textbf{(4.53b)}$$

$$= \left[h_{\lambda}^{N}(k-1) \; - K_{k-N}^{N+1}(k-1) \; 0 \quad \cdots \quad 0 \quad 1 \quad -\tilde{h}_{k-N+1}^{N} \quad \cdots \quad -\tilde{h}_{k}^{N} \quad 0 \quad \cdots \quad 0 \right] \qquad (4.53c)$$

$$= h_{\lambda}^{N}(k-1) \; - \frac{e_{k-1}^{N}}{r_{k-N}^{N}} \rho_{N+1}^{k} \tilde{h}_{\lambda}^{N}(k-N) \qquad (4.53d)$$

This completes the proof of equation (4.39a). Using similar arguments the backwards recursion of equation (4.39b) can be verified.

c. Theorem 4.3: Generalized Levinson Recursion (Normalized Form)

The normalized prediction error coefficients defined in equations (4.30) and (4.34) can be recursively updated using the following recurence relations:

$$\begin{bmatrix} a_{\lambda}^{N-1}(k-1) \\ b_{\lambda}^{N-1}(k-N) \end{bmatrix} = \Theta(\rho_{N-1}^{k}) \begin{bmatrix} a_{\lambda}^{N}(k-1) \\ b_{\lambda}^{N}(k-N) \end{bmatrix} \qquad (4.54)$$

where

$$\Theta(\rho_{N-1}^{k}) \quad \frac{1}{\sqrt{1 \quad (\rho_{N-1}^{k})^{2}}} \begin{bmatrix} 1 & \rho_{N-1}^{k} \\ \rho_{N-1}^{k} & 1 \end{bmatrix} \qquad (4.55)$$

and ρ_{N+1}^{k} is given by equation (4.40).

d. Proof

The proof follows directly from the non- normalized version of the generalized Levinson algorithm (equations (4.39)) and the definitions of the normalized prediction error coefficients (equations (4.29) and (4.33)).

Using (4.29) in (4.39a) we obtain

$$a_\lambda^{N+1}(k+1)$$

$$= \frac{1}{\| e_{k+1}^{N+1} \|}\left[\| e_{k+1}^{N} \| \, a_\lambda^N(k+1) - \rho_{n+1}^k \frac{\| e_{k+1}^N \|}{\| r_{k-N}^N \|} \| r_{k-N}^N \| \, b_\lambda^N(k-N)\right] \tag{4.56a}$$

$$= \frac{\| e_{k+1}^N \|}{\| e_{k+1}^{N+1} \|} a_\lambda^N(k+1) - \rho_{N+1}^k b_\lambda^N(k-N)] \tag{4.56b}$$

Similarly, using (4.33) in (4.39b) we obtain

$$b_\lambda^{N+1}(k-N) = \frac{\| r_{k-N}^N \|}{\| r_{k-N}^{N+1} \|}[b_\lambda^N(k-N) - \rho_{N+1}^k a_\lambda^N(k+1)] \tag{4.57}$$

The proof of the fact that

$$\frac{\| e_{k+1}^N \|}{e_{k-1}^{N+1}} = \frac{\| r_{k-N}^N \|}{r_{k-N}^{N+1}} = \frac{1}{\sqrt{1 - (\rho_{N-1}^k)^2}} \tag{4.58}$$

is relegated to the next section.

The initial values for forward and backward normalized autoregressive parameters are obtained by setting N=0 in equations (4.29) and (4.33). This yields

$$a_\lambda^0(k+1) = [0 \cdots 0 \; \frac{1}{\| y^{k+1} \|} \; 0 \cdots 0] \tag{4.59a}$$

$$b_\lambda^0(k) = 0 \cdots 0 \; \frac{1}{\| y^k \|} \; 0 \cdots 0 \tag{4.59b}$$

We note the similarities in the two generalized forms of the Levinson algorithm presented in this section with that presented earlier in this chapter. A little reflection will convince that equations (4.14) are simply a special case of equations (4.39).

3. Error Order Update Recursions

a. Theorem 4.4

The (N+1) order errors can be calculated from the N-th order errors through the recursion

$$\begin{bmatrix} \bar{e}_{k+1}^{N+1} \\ \bar{r}_{k-N}^{N+1} \end{bmatrix} = \Theta(\rho_{N+1}^{k}) \begin{bmatrix} \bar{e}_{k+1}^{N} \\ \bar{r}_{k-N}^{N} \end{bmatrix} \tag{4.60}$$

where $\Theta(\rho_{N+1}^{k})$ is defined in equation (4.55).

b. Proof (Outline)

Using the normalized form of the Levinson algorithm (equation (4.54)) and the following relationships

$$\bar{e}_{k+1}^{N} = a_{\lambda}^{N}(k+1)y^{\lambda} \tag{4.28b}$$

$$\bar{e}_{k+1}^{N+1} = a_{\lambda}^{N+1}(k+1)y^{\lambda}$$

$$\bar{r}_{k-N}^{N} = b_{\lambda}^{N}(k-N)y^{\lambda} \tag{4.32b}$$

$$\bar{r}_{k-N}^{N-1} = b_{\lambda}^{N+1}(k-N)y^{\lambda}$$

equation (4.60) is easily verified.

We now return to the proof of equation (4.58). That is.

$$\frac{\left| e_{k+1}^{N} \right|}{\left| e_{k+1}^{N+1} \right|} = \frac{\left| r_{k-N}^{N} \right|}{\left| r_{k-N}^{N+1} \right|} = \frac{1}{\sqrt{1 - (\rho_{N+1}^{k})^{2}}} \tag{4.58}$$

Working with the term on the left-hand side of the equation

$$\frac{\left| e_{k+1}^{N} \right|}{\left| e_{k+1}^{N-1} \right|} = \frac{E\left\{ (e_{k+1}^{N})^{2} \right\}^{1/2}}{E\left\{ (e_{k+1}^{N-1})^{2} \right\}^{1/2}} \tag{4.61a}$$

$$= \frac{E\left\{ y^{k+1} e_{k+1}^{N} \right\}^{1/2}}{E\left\{ y^{k+1} e_{k+1}^{N+1} \right\}^{1/2}} \tag{4.61b}$$

$$= \frac{1}{\left[1 - \frac{\rho_{N+1}^{k} E\left\{y^{k+1} \bar{r}_{k-N}^{N}\right\} \Big| \Big| e_{k-1}^{N} \Big| \Big|}{E\left\{y^{k+1} e_{k+1}^{N}\right\}}\right]^{1/2}} \tag{4.61c}$$

$$= \frac{1}{\sqrt{1 - (\rho_{N+1}^{k})^2}} \tag{4.61d}$$

Similar arguments can be used to verify the other relationship given in equation (4.58).

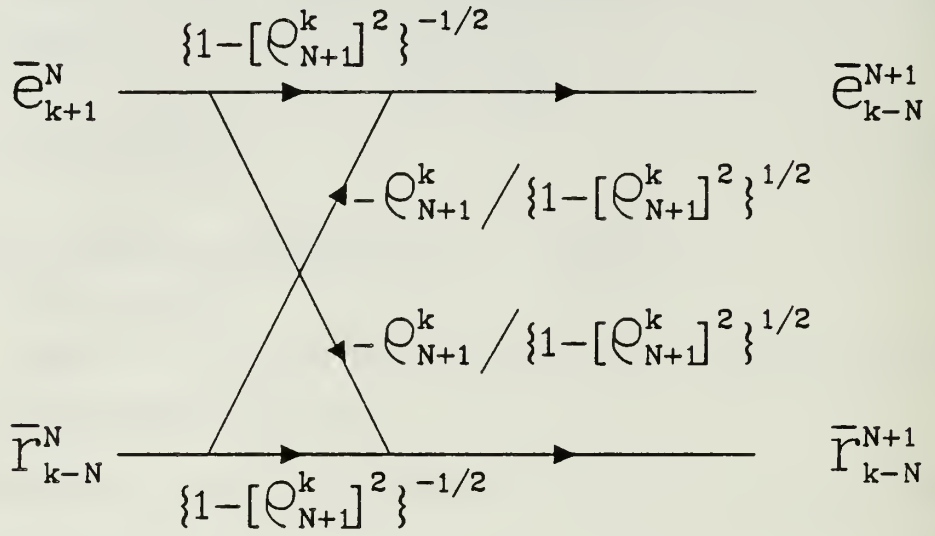
Figure 4.3 symbolizes a single stage of the recursions of equation (4.60) while Figure 4.4 illustrates a third order analysis model.

The interesting feature of equations (4.60) is that they do not make any assumptions about the nature of the given data. The data values need not be delayed versions of each other, as is the case for the autoregressive model. Any set of data values can be used. This fact will be significant when we deal with 2-D and nonlinear lattices.

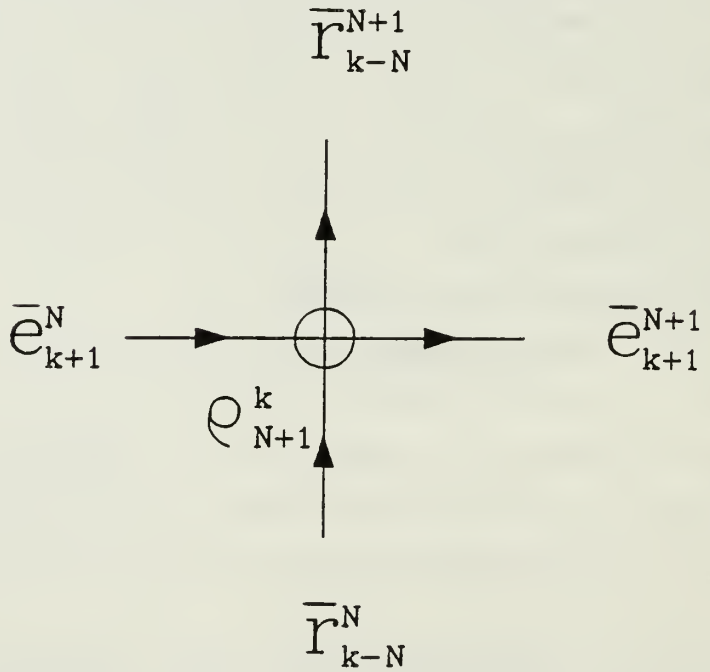
4. The Generalized Schur Algorithm

In this section an algorithm will be presented which allows the calculation of the partial correlation coefficients in a direct manner. It will be shown that knowledge of the correlation matrix is sufficient to calculate all the reflection factors and thus solve the normal equations (by use of the Levinson algorithm). The method used has come to be known as the Schur algorithm [Ref. 3].

In order to obtain the desired result we must introduce two new quantities, defined as



a. Single Stage of Generalized Lattice Filter



b. Alternate Representation

Figure 4.3: Generalized Lattice Filter Sections

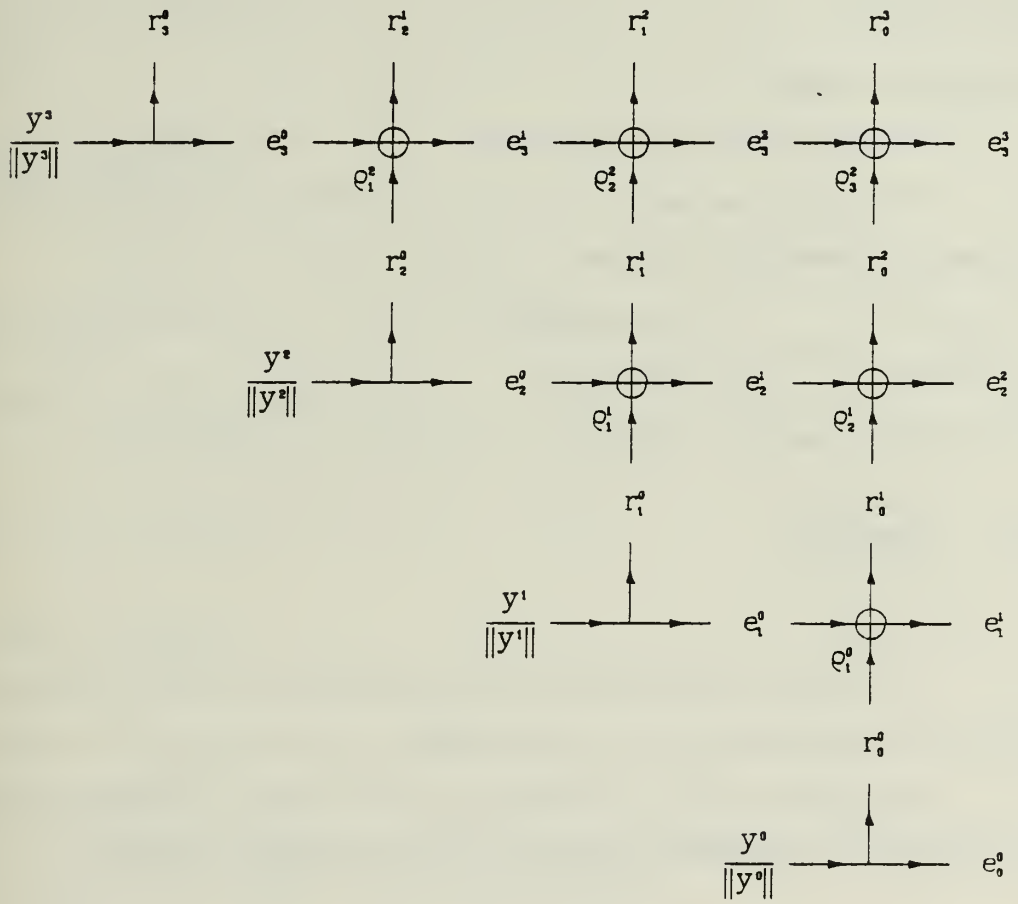


Figure 4.4: 3-rd Order Generalized Lattice Filter

$$\alpha_{N+1}^{\lambda}(k+1) = E\left\{\bar{e}_{k+1}^{N} y^{\lambda}\right\} = a_{\mu}^{N}(k+1)R^{\mu\lambda} \tag{4.62}$$

$$\beta_{N+1}^{\lambda}(k-N) = E\left\{\bar{r}_{k-N}^{N} y^{\lambda}\right\} = b_{\mu}^{N}(k-N)R^{\mu\lambda} \tag{4.63}$$

where

$$R^{\mu\lambda} = E\left\{y^{\mu} y^{\lambda}\right\} \tag{4.64}$$

is a correlation matrix.

a. Theorem 4.5: Schur Recursions

The quantities $\alpha_N^{\lambda}(k+1)$ and $\beta_N^{\lambda}(k-N)$ defined in equations (4.62) and (4.63) can be updated according to

$$\begin{bmatrix} \alpha_{N+1}^{\lambda}(k+1) \\ \beta_{N+1}^{\lambda}(k-N) \end{bmatrix} = \Theta(\rho_{N+1}^{k}) \begin{bmatrix} \alpha_N^{\lambda}(k+1) \\ \beta_N^{\lambda}(k-N) \end{bmatrix} \tag{4.65}$$

and the partial correlation coefficient, ρ_{N+1}^{k}, can be calculated from

$$\rho_{N+1}^{k} = \frac{\alpha_N^{k-N}(k+1)}{\beta_N^{k-N}(k-N)} \tag{4.66}$$

b. Proof

The proof of equations (4.65) follows directly from equations (4.60) and the above definitions for $\alpha_N^{\lambda}(k+1)$ and $\beta_N^{\lambda}(k-N)$.

Beginning with the definition of the partial correlation coefficient given in (4.40). the relationship given in (4.66) is verified as follows

$$\rho_{N+1}^{k} = E\left\{\bar{e}_{k+1}^{N} \bar{r}_{k-N}^{N}\right\} \tag{4.67a}$$

$$= E\left\{\bar{e}_{k+1}^{N} y^{\lambda}\right\} b_{\lambda}^{N}(k-N) \tag{4.67b}$$

$$= E\left\{\bar{e}_{k+1}^{N} y^{k-N}\right\} b_{k-N}^{N}(k-N) \tag{4.67c}$$

$$= \alpha_N^{k-N}(k+1)b_{k-N}^N(k-N) \tag{4.67d}$$

But

$$b_{k-N}^N(k-N) = \frac{1}{\| y^{k-N} \|} = \frac{1}{[R^{(k-N)(k-N)}]^{1/2}} \tag{4.68}$$

and

$$\beta_N^{k-N}(k-N) = [R^{(k-N)(k-N)}]^{1/2} \tag{4.69}$$

Therefore,

$$\rho_{N+1}^k = \frac{\alpha_N^{k-N}(k+1)}{\beta_N^{k-N}(k-N)} \tag{4.70}$$

Using the initial conditions given by (4.59) the following initial values are determined for the Schur algorithm

$$\alpha_0^\lambda(k+1) = \frac{1}{\| y^{k+1} \|} R^{(k+1)\lambda} \tag{4.71a}$$

$$\alpha_0^\lambda(k-1) = \frac{1}{\| y^{k+1} \|} [R^{(k+1)0} \quad R^{(k+1)1} \quad \cdots \quad R^{(k+1)K} \tag{4.71b}$$

$$\beta_0^\lambda(k) = \frac{1}{\| y^k \|} R^{k\lambda} \tag{4.72a}$$

$$\beta_0^\lambda(k) = \frac{1}{\| y^k \|} [R^{k0} \quad R^{k1} \quad \cdots \quad R^{kK} \tag{4.72b}$$

where the parenthesis used to surround the first indices in equation (4.71) simply indicate that they are fixed at the value indicated.

The Schur algorithm implies a filter structure identical to that of Figure 4.4. In this case the input vectors are the rows of the correlation matrix (normalized by the square root of the diagonal elements).

5. Synthesis Model

The original data, Y, can be synthesised from the model parameters obtained from the Levinson algorithm, equation (4.54). It is also possible to regenerate the y^λ directly from the lattice parameters. The desired result is

obtained by solving equations (4.60) for $\bar{e}_{k-1}^{N}$ and $\bar{r}_{k-N}^{N+1}$.

$$\bar{e}_{k-1}^{N} = \sqrt{1 - (\rho_{N+1}^{k})^2}\, \bar{e}_{k-1}^{N+1} + \rho_{N-1}^{k} \bar{r}_{k-N}^{N} \tag{4.73a}$$

$$\bar{r}_{k-N}^{N+1} = \sqrt{1 - (\rho_{N+1}^{k})^2}\, \bar{r}_{k-N}^{N} - \rho_{N+1}^{k} \bar{e}_{k+1}^{N+1} \tag{4.73b}$$

Equations (4.73) constitute the synthesis model. They imply a structure similar to the analysis model of Figure 4.4. but with the direction of flow of the forward error signals reversed. The processing performed at each stage can be visualized as in Figure 4.5. A complete third order synthesis model is shown in Figure 4.6. Each horizontal path in Figure 4.6 represents a separate synthesis model. synthesising a different component of Y. The coordinate system for each of these models depends only on values of y^{λ} which appear farther down, that is they have a smaller value of the index, λ.

Compare Figure 4.3b and 4.5b. It is apparent that the behaviour of the backwards error signals is identical in the two cases. Hence, it is possible to construct a synthesis model that only reverses the direction of the forward error corresponding to the point being predicted. This assumes knowledge of the other inputs (zero order forward errors) to the lattice. Such a configuration is illustrated in Figure 4.7.

The amount of knowledge possessed about the signals used for the predictions dictates which form of the synthesis model should be used. If little is known. then estimates must first be generated which can then be used in the prediction. This corresponds to the model of Figure 4.6. If complete knowledge is available (either from initial conditions or previous predictions) then Figure 4.7 can be used. It is also possible to construct models which exploit partial knowledge of the input signals and thus fall between these two extremes. In this case the known signals should be input as zero order forward errors while the unknown ones must be estimated.

6. Stochastic Fourier Series Interpretation

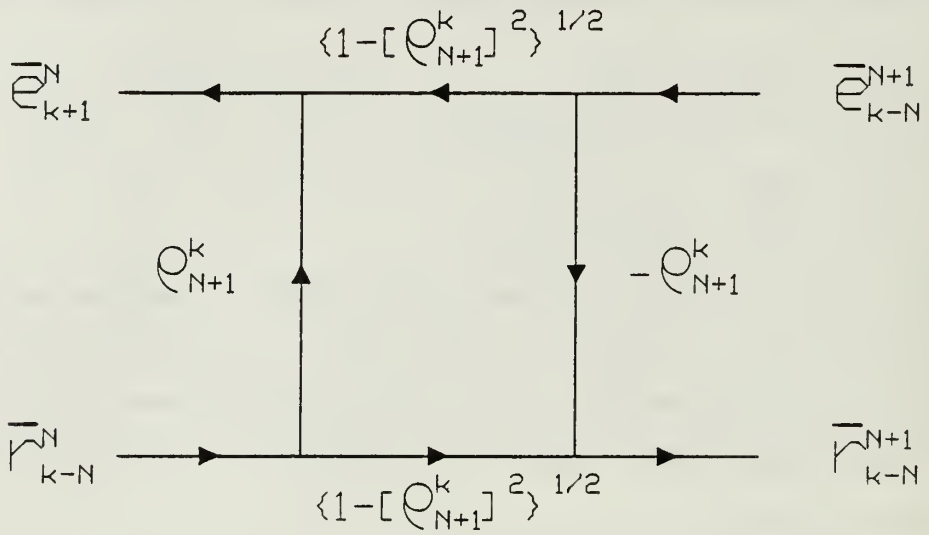
From equations (4.72) and (4.40) we can deduce the following result

$$e_{k+1}^{0} = e_{k+1}^{N+1} + \sum_{\lambda=1}^{N} E\left\{e_{K+1}^{\lambda}\bar{r}_{k-N}^{\lambda}\right\}\bar{r}_{k-\lambda}^{\lambda} \tag{4.74a}$$

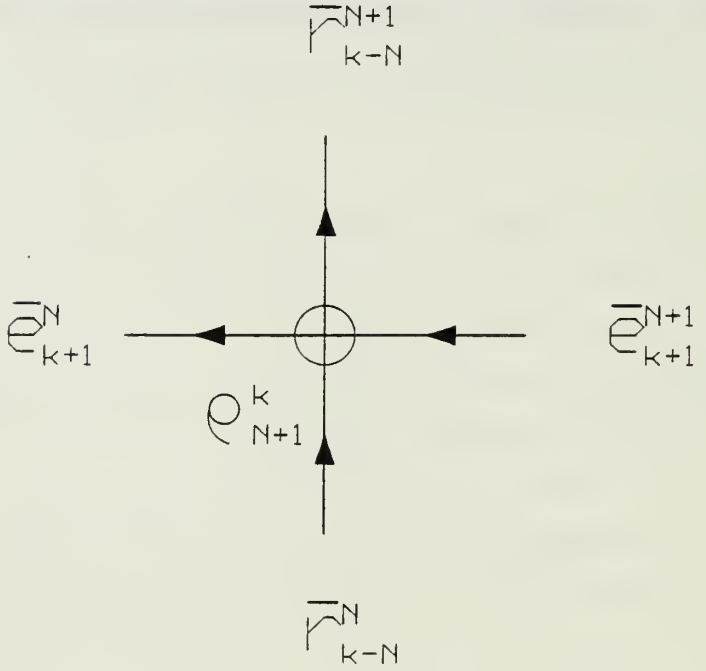
$$= e_{k+1}^{N+1} + \sum_{\lambda=1}^{N} \| e_{k+1}^{\lambda} \| \| \rho_{\lambda+1}^{k}\bar{r}_{k-\lambda}^{\lambda} \tag{4.74b}$$

where e_{k+1}^{0} is equivalent to y^{k+1}. These expressions offer an alternate interpretation of the lattice filter. Equations (4.74) describe a stochastic Fourier series expansion of the forward error sequence where the basis functions are given by the backwards error signals. The Fourier coefficients are related to the partial correlation coefficients.

This concludes the review of existing lattice formulations. In the next chapter new. multidimensional extensions of this theory are presented. In Chapter VI these results are used to derive original nonlinear lattice structures.



a. Single Stage of Generalized Lattice Synthesis Filter



b. Alternate Representation

Figure 4.5: Generalized Lattice Synthesis Filter Sections

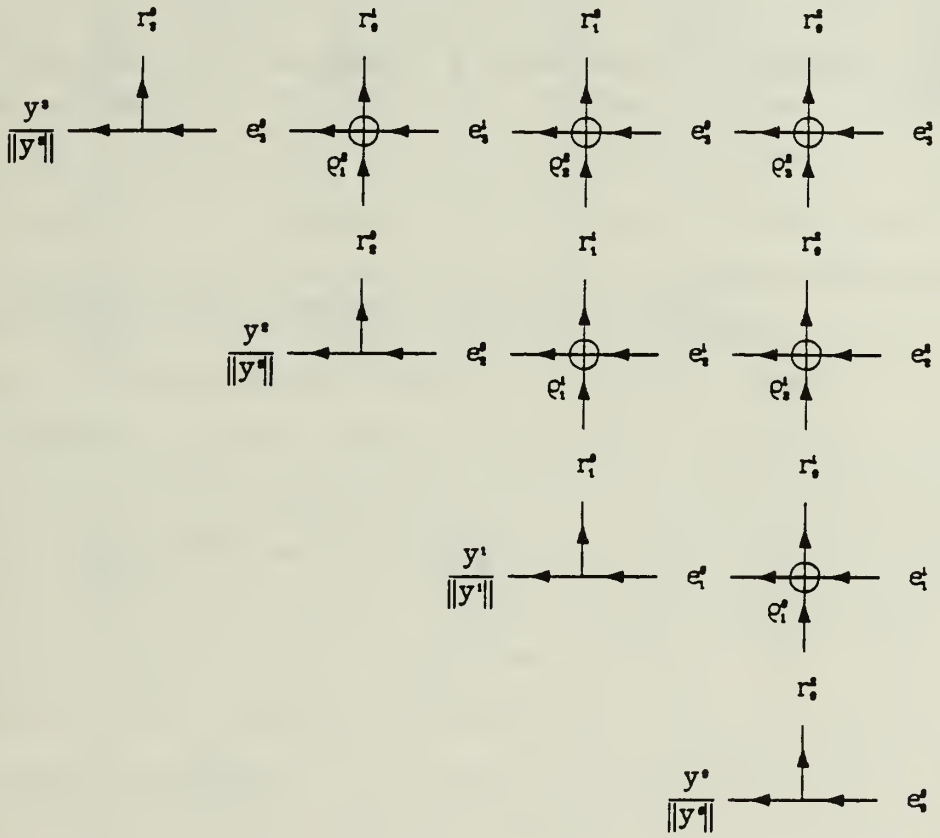


Figure 4.6: 3-rd Order Generalized Lattice Synthesis Filter

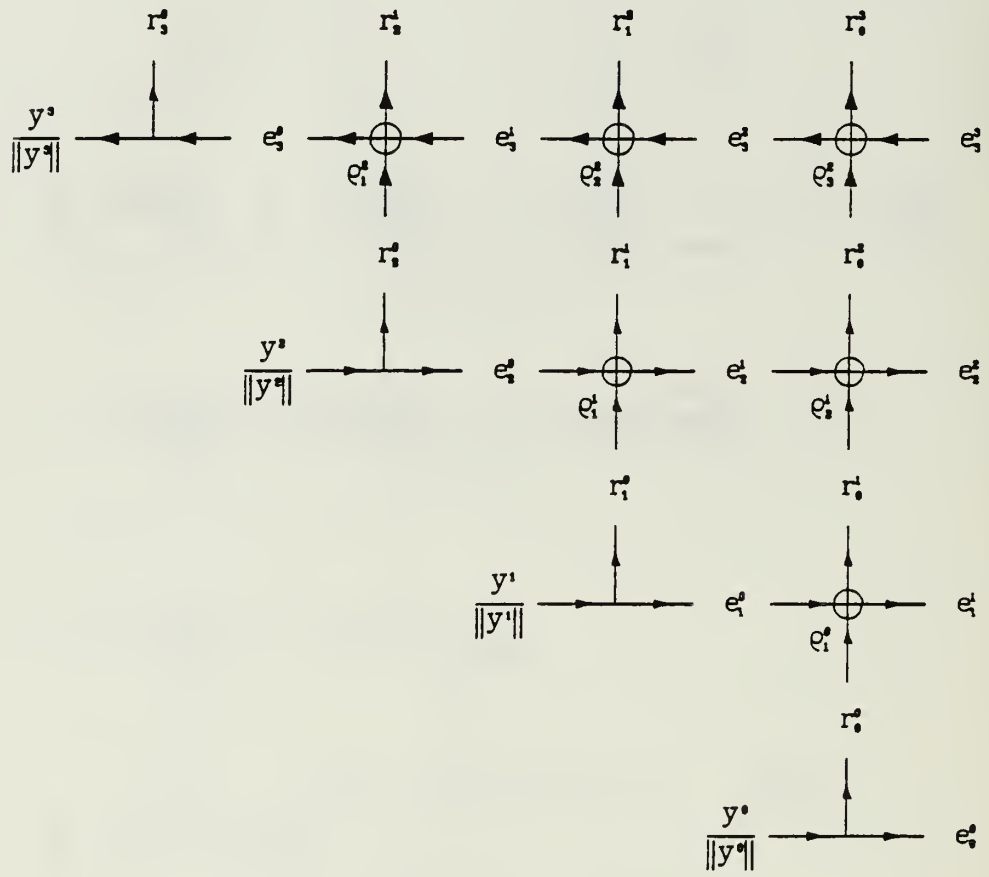


Figure 4.7: 3-rd Order Generalized Lattice Synthesis Filter

V. TWO DIMENSIONAL LATTICE STRUCTURES

In this chapter a new two-dimensional lattice structure will be derived and discussed. Lattice modelling of two dimensional fields has recently received considerable attention [Ref. 43,44,45]. In one dimensional lattice modelling, updating the order introduces only one new point into the model support. An order update in a 2-D lattice model must introduce O(N) new points, where N is the model order. Several different solutions have been suggested to this problem. The first due to Marzetta [Ref. 43,44], uses a particular ordering of the data to reduce the problem to one dimension. He proposed a half-plane support which is infinite in one of the two dimensions. This approach, while maintaining several of the nice characteristics of 1-D lattices, such as correlation matching and producing a minimum phase filter, leads to very long delay filters.

A different approach, proposed by Parker and Kayran [Ref. 45], simultaneously introduces many points into the support when the model order is increased. Their filter uses a quarter plane support and introduces three parameters at each order update. Therefore, it lacks sufficient parameters to represent all classes of 2-D autoregressive quarter plane filters. More importantly, it lacks the property of orthogonality so that the cascading of stages does not lead to an optimum filter (better filters are possible using an equivalent number of parameters). It's simplicity is attractive and good results have been reported using this approach [Ref. 46].

The theory presented here maintains features of both previous approaches. It utilizes the generalized lattice theory presented in Chapter 4 to decompose the **global** $O(N)$ point update into $O(N^2)$ single point **local** updates. It maintains the important property of orthogonality so that the solution at all stages is optimum. Although only the quarter plane support case is presented here in detail, the theory can be used for any shaped support. It is shown that the Levinson and Schur algorithms (see Chapter 4) can be used to solve the 2-D linear prediction

problem. In its most general form the lattice contains $O(N^4)$ parameters while there are only $O(N^2)$ points in the filter support. Several structures are presented which take advantage of shift-invariance and reduce this requirement to $O(N^3)$.

A. GENERAL FORM OF 2-D LATTICE FILTER

The theory used is exactly that presented in the previous chapter. The 2-D structure results from a careful selection of input data. To illustrate the proposed 2-D lattice structure we will consider a 2-D linear prediction problem which utilizes a quarter plane support. The 2-D data field is given by

$$\mathbf{Y} = \left\{ y^{\lambda_1 \lambda_2} = y(\lambda_1, \lambda_2) \right\} \tag{5.1}$$

where $(\lambda_1, \lambda_2) \in {}^2L^K = L^K \times L^K$

where L^K is an index set given by

$$L^K = \left\{ 0, \ldots, K \right\} \tag{5.2}$$

Points will be used from this data field in a particular, convenient order. We define an ordered index set

$$\begin{aligned} {}^2_{\diamond}L^K = \Big\{ & (0,0),(1,0),(0,1),(2,0),(0,2), \ldots, (N,0),(0,N), \\ & (1,1),(2,1),(1,2), \ldots, (N,1),(1,N), \ldots, \\ & (K-2,K),(K-1,K-1),(K,K-1),(K-1,K),(K,K) \Big\} \end{aligned} \tag{5.3}$$

Other orderings are possible and equally valid. This one is chosen merely to illustrate the concepts. The desired ordering of ${}^2L^K$, to obtain ${}^2_{\diamond}L^K$, can be accomplished by the following, computationally efficient algorithm

```
k = 0
for m0 = 0 to K
   for n0 = 0 to 2*m0
      if (mod(n0.2) = 0)
         then begin
            i = m0
            j = n0/2
         end
         else begin
            i = n0/2
            j = m0
         end
      ²ₒL^K (k) = ²L^K (i,j)
      k = k + 1
   next n0
next m0
```

In this algorithm ${}^{2}_{o}L^{K}(k)$ has been used to describe the k-th element of the index set ${}^{2}_{o}L^{K}$, while ${}^{2}L^{K}(i,j)$ has been used to indicate the (i,j)-th element of ${}^{2}L^{K}$. The $(K+1)^2$ elements of ${}^{2}L^{K}$ have been ordered into a one dimensional index set, ${}^{2}_{o}L^{K}$. The elements of ${}^{2}_{o}L^{K}$ can be numbered, consecutively, from 0 to $(K+1)^2-1$. The notation (k,l) - q will be used to mean: the element of the index set corresponding to the q-th element prior to the element (k,l). For example, (2,0)-3 would indicate the element (1,0) (see equation (5.3)). Occasionally this notation will be abbreviated to simply, kl-q.

Define the (q-1) order, normalized, forward error associated with the prediction of the point y(k,l) from the previous (in the sense of ${}^{2}_{o}L^{K}$) (q-1) points, as

$$\bar{e}_{kl}^{q-1} = a_{\lambda_1\lambda_2}^{q-1}(k,l)y^{\lambda_1\lambda_2} \qquad (5.4)$$

where the implied summation over $(\lambda_1,\lambda_2) \in {}^{2}L^{K}$, can be carried out in any order, as long as all components are considered. It is preferred to think of (λ_1,λ_2) belonging to ${}^{2}L^{K}$ rather than ${}^{2}_{o}L^{K}$ as this maintains the two-dimensional character of the problem.

The $a_{\lambda_1\lambda_2}^{q-1}(k,l)$ can be interpreted as the components of a second order covariant tensor. These components, for a range of indices $(\lambda_1,\lambda_2) \leq (k,l) - q$ (in the

sense of (5.3)), or for indices $(\lambda_1,\lambda_2) > (k,l)$. are equal to zero. When $(\lambda_1,\lambda_2) = (k,l)$. the component

$$a_{kl}^{q-1}(k,l) = \frac{1}{\| e_{kl}^{q-1} \|} \tag{5.5}$$

where e_{kl}^{q-1} is an unnormalized version of $\bar{e}_{kl}^{q-1}$.

A normalized backward error associated with the prediction of $y((k,l)-q)$ from the next q-1 points of ${}_{\delta}^{2}L^{K}$, is defined by

$$\bar{r}_{kl-q}^{q-1} = b_{\lambda_1\lambda_2}^{q-1}((k,l)-q)y^{\lambda_1\lambda_2} \tag{5.6}$$

As with the forward error prediction coefficients, the $b_{\lambda_1\lambda_2}^{q-1}((k,l)-q)$ can be interpreted as components of a second order covariant tensor. The components $b_{\lambda_1\lambda_2}^{q-1}((k.l)-q)$ equal zero for the range of indices $(\lambda_1,\lambda_2) \leqslant ((k,l)-q)$ or $(\lambda_1,\lambda_2) > (k,l)$. For the case when the index $(\lambda_1,\lambda_2) = ((k,l)-q)$ the component

$$b_{kl-q}^{q-1}((k.l)-q) = \frac{1}{\| r_{kl-q}^{q-1} \|} \tag{5.7}$$

where r_{kl-q}^{q-1} is an unnormalized version of $\bar{r}_{kl-q}^{q-1}$.

1. Normalized 2-D Levinson Algorithm

   a. Theorem 5.1

   The 2-D prediction error coefficients can be updated in order using the following recursions

$$\begin{bmatrix} a_{\lambda_1\lambda_2}^{q}(k.l) \\ b_{\lambda_1\lambda_2}^{q}((k.l)-q) \end{bmatrix} = \Theta(\rho_q^{kl}) \begin{bmatrix} a_{\lambda_1\lambda_2}^{q-1}(k.l) \\ b_{\lambda_1\lambda_2}^{q-1}((k.l)-q) \end{bmatrix} \tag{5.8}$$

where

$$\Theta(\rho_q^{kl}) = \frac{1}{\sqrt{1-(\rho_q^{kl})^2}} \begin{bmatrix} 1 & -\rho_q^{kl} \\ -\rho_q^{kl} & 1 \end{bmatrix} \tag{5.9}$$

and

$$\rho_q^{kl} = E\left\{\bar{e}_{kl}^{q-1}\bar{r}_{kl-q}^{q-1}\right\} \tag{5.10}$$

b. Proof (Outline)

The proof follows directly if the index (k.l) is replaced by a single index which runs from 0 to $(K+1)^2-1$ and thus indexes the elements of ${}_{\delta}^{2}L^{K}$. The equations (5.8) are identical in form to equations (4.54) and the proof presented there can be applied. This approach is equivalent to reordering the data field into a vector in the order specified by ${}_{\delta}^{2}L^{K}$.

An alternate proof is possible by generalizing the methods of Chapter 4. Alternate coordinate systems could be introduced (similar to (4.35) and (4.37)). The necessary transformations can be found and all the steps of the proof of Theorem 4.2 can be generalized to these higher order objects. These concepts will not be explored further here except to note that they could be extented to solve problems in any number of dimensions. they are not restricted to the two dimensional case being studied here.

2. Normalized 2-D Error Order Updates

a. Theorem 5.2

The two-dimensional prediction errors can be updated in order according to the following equation

$$\begin{bmatrix} \bar{e}_{kl}^{q} \\ \bar{r}_{kl-q}^{q} \end{bmatrix} = \Theta(\rho_q^{kl}) \begin{bmatrix} \bar{e}_{kl}^{q-1} \\ \bar{r}_{kl-q}^{q-1} \end{bmatrix} \quad . \tag{5.11}$$

b. Proof (Outline)

The proof follows from equations (5.8). If an inner product is formed on both sides of the equation (5.8a) with the data field $\mathbf{Y}$ equation (5.11a) results. Similar arguments can be employed in the verification of (5.11b).

3. 2-D Form of Schur Recursion

Define the quantities

$$\alpha_q^{\lambda_1\lambda_2}(k.l) = E\left\{\bar{e}_{kl}^{q-1} y^{\lambda_1\lambda_2}\right\} = a_{\lambda_3\lambda_4}^{q-1}(k.l) R^{\lambda_1\lambda_2\lambda_3\lambda_4} \tag{5.12a}$$

$$\beta_q^{\lambda_1\lambda_2}((k,l)-q) = E\left\{\bar{r}_{kl-q}^{q-1} y^{\lambda_1\lambda_2}\right\} = b_{\lambda_3\lambda_4}^{q-1}(kl-q) R^{\lambda_1\lambda_2\lambda_3\lambda_4} \tag{5.12b}$$

where

$$R^{\lambda_1\lambda_2\lambda_3\lambda_4} = E\left\{y^{\lambda_1\lambda_2}y^{\lambda_3\lambda_4}\right\} \tag{5.13}$$

We note that for the 2-D case the correlation is a fourth order tensor.

The Schur recursions for the 2-D case are then given by the following theorem.

a. Theorem 5.3

The generalized 2-D Schur recursions are given by

$$\begin{bmatrix} \alpha_q^{\lambda_1\lambda_2}(k,l) \\ \beta_q^{\lambda_1\lambda_2}((k,l)-q) \end{bmatrix} = \Theta(\rho_q^{kl}) \begin{bmatrix} \alpha_{q-1}^{\lambda_1\lambda_2}(k,l) \\ \beta_{q-1}^{\lambda_1\lambda_2}((k,l)-q) \end{bmatrix} \tag{5.14}$$

and ρ_q^{kl} can be calculated from

$$\rho_q^{kl} = \frac{\alpha_{q-1}^{kl-q}(k,l)}{\beta_{q-1}^{kl-q}((k,l)-q)} \tag{5.15}$$

b. Proof (Outline)

The proof follows identical arguments to those of Theorem 4.5 and so is not given here.

4. 2-D Lattice Structures

The derivations presented do not assume shift- invariance. Models are built for each point in the data field starting with the point y(K.K) and ending with the point y(0.0). All the models are not equivalent. however. In fact, no two models are identical. The only model that is quarter plane is the first one. that is the model corresponding to the point y(K.K). A quarter plane model for any point. y(m.n), in the field can be built by considering an appropriate subset of the set **Y** (equation (5.1)). The subset would start with the point y(m-N.n-N) and continue in a quarter plane manner until the point y(m.n). for an N-th (global) order model. This support can be written

$$\mathbf{D} = \left\{y(\lambda_1,\lambda_2)\right\} \tag{5.16}$$

where

$$(\lambda_1,\lambda_2) \in {}^2L_{mn}^N = L_m^N \times L_n^N$$

where

$$L_m^N = \Big\{(m-N),(m-N+1), \ldots, m\Big\} \tag{5.17a}$$

and

$$L_n^N = \Big\{(n-N),(n-N+1), \ldots, n\Big\} \tag{5.17b}$$

The $(N+1)^2$ terms of this index set can be ordered to form

$${}_0^2L_{mn}^N = \Big\{(m-N,n-N),(m-N+1,n-N),(m-N,n-N+1),\ldots.$$

$$(m-1,n-1),(m,n-1),(m-1,n),(m,n)\Big\} \tag{5.18}$$

The subset of Y given by (5.16) and the ordering of (5.18) are illustrated in Figure 5.1. This could be done for each point in the entire data field Y, yielding $(K+1)^2$ models. If the process is shift-invariant, then all the models would be identical. Other simplifications in the model are possible if shift-invariance can be assumed. These will be the topic of the next section. In addition, if ergodicity is assumed the required statistical averages can be replaced by appropriate spatial ones.

A second order quarter plane 2-D lattice model for generating the prediction error associated with an arbitrary point y(m,n) is illustrated in Figure 5.2. In this diagram the forward and backward error fields are indicated pictorially rather than symbolically. The icons used are defined in the legend on the diagram. The large squares, at each stage, indicate the entire support for the second order model. The small blank squares indicate the additional support (besides the error field squares) used to generate the given error fields. For example, the forward error field in the upper right hand square is generated by predicting y(m,n) from all the remaining data points in the large square, including the one indicated as a backward prediction error.

(2N) $y(m-N, n)$	• • •	(4) $y(m-N, n-N+2)$	(2) $y(m-N, n-N+1)$	(0) $y(m-N, n-N)$
(4N-1) $y(m-N+1, n)$	• • •	(2N+3) $y(m-N+1, n-N+2)$	(2N+1) $y(m-N+1, n-N+1)$	(1) $y(m-N+1, n-N)$
(6N-4) $y(m-N+2, n)$	• • •	(4N) $y(m-N+2, n-N+2)$	(2N+2) $y(m-N+2, n-N+1)$	(3) $y(m-N+2, n-N)$
• • •	• • •	• • •	• • •	• • •
$(N+1)^2-1$ $y(m, n)$	• • •	(6N-5) $y(m, n-N+2)$	(4N-2) $y(m, n-N+1)$	(2N-1) $y(m, n-N)$

Figure 5.1: Filter Support Ordering (see equation 5.18).

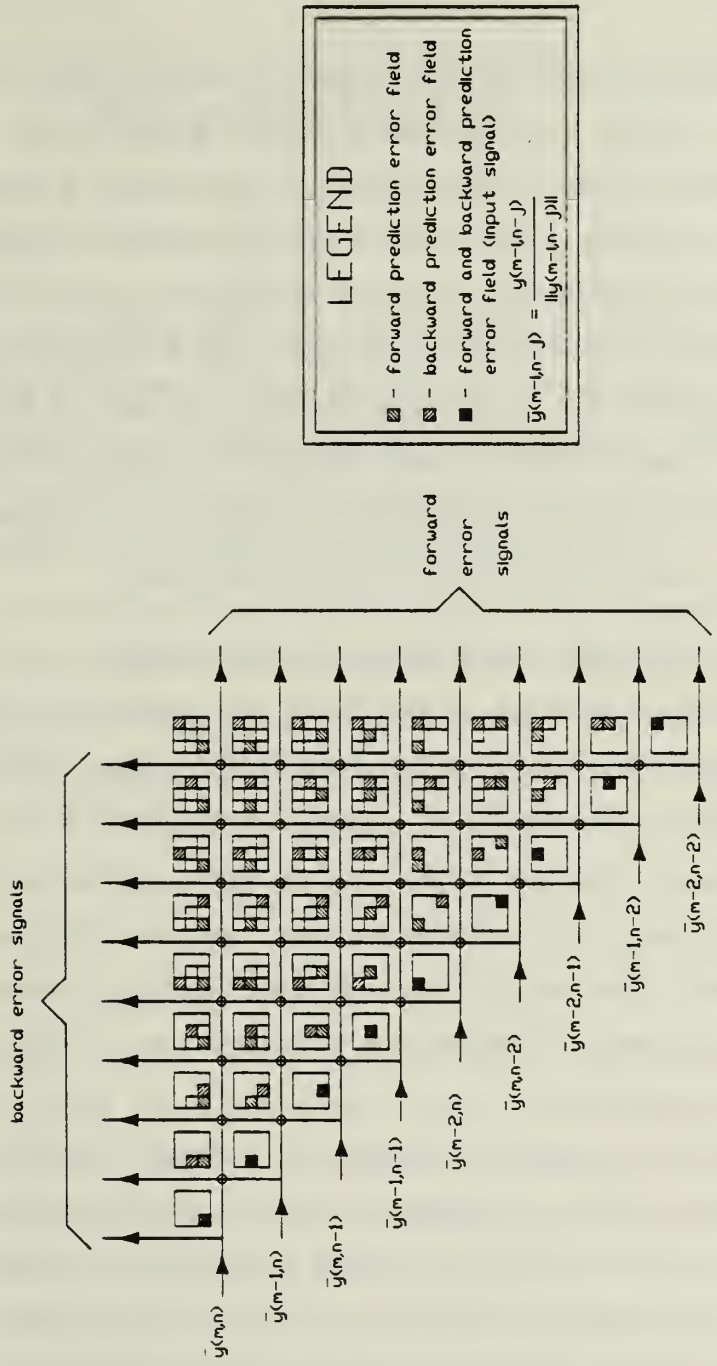


Figure 5.2: General 2-D Quarter Plane Lattice Filter.

The doubly hatched squares, corresponding in this diagram to zero order errors, are inputs to the filter. In general, (in later diagrams) doubly hatched squares are considered to be inputs, although, they may not always correspond to zero order errors.

The ordering chosen for ${}^{2}_{O}L^{K}$ (equation (5.3)) is only one of many that could have been chosen to implement a quarter plane model. This particular selection was made for ease of implementation and because it guaranteed that for every 2N+1 local updates, a global order update would be completed. In Figure 5.2 the filter can be visualized as a cascade of increasing order filter sections. For every global update, $O(N^3)$ local order updates must be completed. This implies a total of $O(N^4)$ updates for an N-th order filter. In general this is too large a number to allow these filters to be used for any real time applications.

In the next section the problem of reducing the complexity of the 2-D lattice filter is examined and some solutions are proposed.

B. REDUCED COMPLEXITY 2-D LATTICE FILTERS

The assumption of shift-invariance allows certain of the backward prediction errors to be considered to be shifted versions of each other. This eliminates some calculations. Various structures are possible depending on the types of shifts introduced. We note that no single type of shift (neither z_1^{-1}, z_2^{-1}, $z_1^{-1}z_2^{-1}$) will introduce all the new data elements into the support that are required for a global order update. Because of this, additional prediction errors will have to be introduced at each stage. This reduces somewhat the advantage gained by the shift-invariance assumption.

Two types of delays will be considered in detail. Initially, several models involving **diagonal** shifts are examined. Later, a model involving a **horizontal** shift is discussed. We begin by introducing a diagonal shift operator, D. This is equivalent to multiplication by $z_1^{-1}z_2^{-1}$ in the bivariate z-transform domain.

Because of the assumed shift invariance the following statements can be made

$$R(m,n,m-i,n-j) = E\left\{y(m,n)y(m-i,n-j)\right\} \tag{5.19a}$$

$$= E\left\{\mathbf{D}[y(m,n)]\mathbf{D}[y(m-i,n-j)]\right\} \tag{5.19b}$$

$$= E\left\{y(m-1,n-1)y(m-i-1,n-j-1)\right\} \tag{5.19c}$$

$$= R(i,j) \tag{5.19d}$$

where R() is a correlation function. The correlation is only a function of the relative positions of the two points not their absolute positions. If we adopt for a moment the Hilbert space formulation (see Appendix A) we conclude that the diagonal shift operator is an inner-product preserving operator and so the use of the shifted versions of the backward error signals is justified.

Consider the structure illustrated in Figure 5.3. It represents a third order quarter plane lattice filter. At each global stage, $(2N-1)^2-3$ lattice coefficients are introduced. Therefore, an N-th order model requires $O(N^3)$ parameters. Notice that at each stage two new errors are introduced. They each require the solution of an $(N-1)^2$ point prediction problem. For small N this is an insignificant number, however, for large N it becomes overwhelming and the required number of parameters again becomes $O(N^4)$. It is difficult to analyze this structure analytically, as the index sets for each prediction error are different. The support for different errors follow different patterns. This, and the complexity issue make it a structure that is really only of academic interest.

The structure of Figure 5.4 is a true $O(N^3)$ parameter model. It avoids the addition of the new error signals at each stage by introducing them at the outset. The structure has a support that differs slightly from quarter plane. A more significant drawback, however, is that the maximum order of the filter must be fixed at the start. If the maximum order is overestimated then some unnecessary computations will have been performed. If on the other hand, the maximum

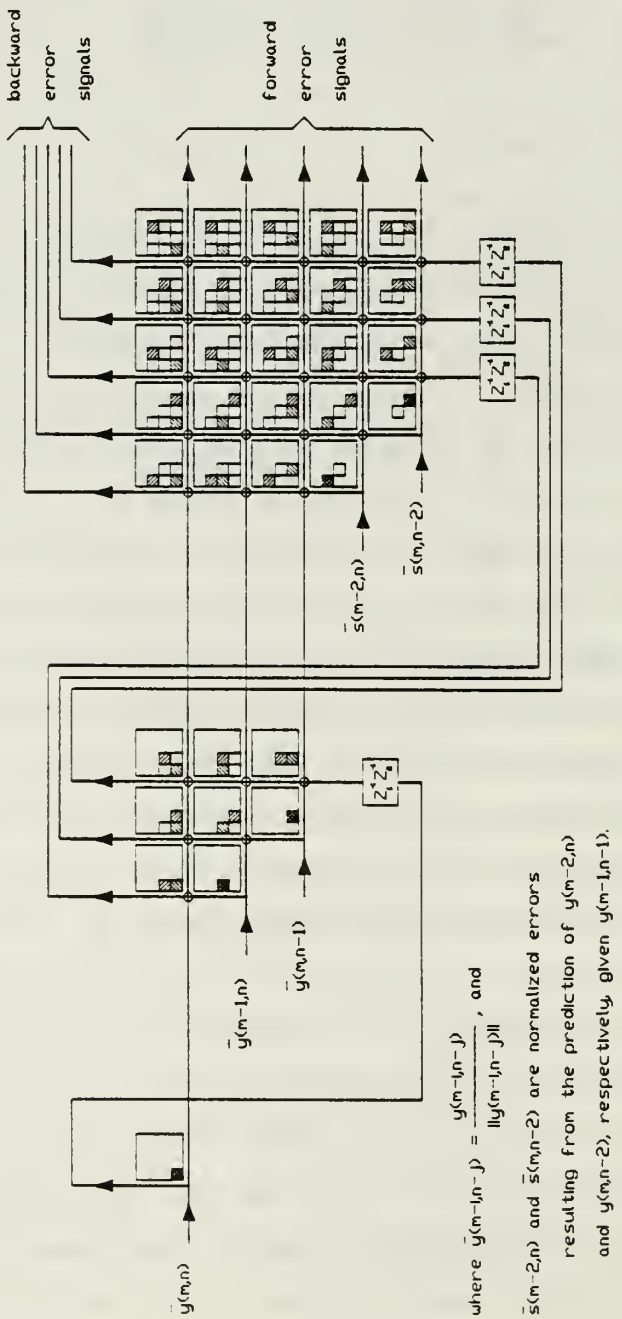


Figure 5.3: Reduced Complexity, 2-D Quarter Plane Lattice Filter.

required order was underestimated, then a great price must be paid to increase it. However, this is considered to be a superior structure because of the regularity and complexity reduction it offers. Therefore, a more detailed analysis of this model will be performed.

The ordered index set (equivalent to (5.18)) in this case (for N=2) is given by

$$ {}^{2}_{O}L^{2}_{mn} = \Big\{(m-2,n-2),(m-1,n-3),(m-3,n-1),(m-1,n-2),(m-2,n-1),$$

$$(m-1,n-1),(m,n-2),(m-2,n),(m,n-1),(m-1,n),(m,n)\Big\} \tag{5.20}$$

The following relations can then be used to advantage to update the backwards errors

$$\bar{r}^{0}_{(m-1,n-1)} = \mathbf{D}\left[\bar{r}^{0}_{(m,n)}\right] \tag{5.21a}$$

$$\bar{r}^{1}_{(m-2,n-1)} = \mathbf{D}\left[\bar{r}^{1}_{(m-1,n)}\right] \tag{5.21b}$$

$$\bar{r}^{2}_{(m-1,n-2)} = \mathbf{D}\left[\bar{r}^{2}_{(m,n-1)}\right] \tag{5.21c}$$

$$\bar{r}^{3}_{(m-3,n-1)} = \mathbf{D}\left[\bar{r}^{3}_{(m-2,n)}\right] \tag{5.21d}$$

$$\bar{r}^{4}_{(m-1,n-3)} = \mathbf{D}\left[\bar{r}^{4}_{(m,n-2)}\right] \tag{5.21e}$$

$$\bar{r}^{5}_{(m-2,n-2)} = \mathbf{D}\left[\bar{r}^{5}_{(m-1,n-1)}\right] \tag{5.21f}$$

In general, whenever (m-i,n-j) equals (m,n)-q, then

$$\bar{r}^{q}_{(m-i-1,n-j-1)} = \mathbf{D}\,\bar{r}^{q}_{(m-i,n-j)} \tag{5.22}$$

Equation (5.22) is a simple rule for exploiting the shift- invariance of the data field.

One final reduced complexity lattice model will be introduced. It will serve to illustrate the variety of structures possible and in particular will yield a model for which an especially convenient synthesis model can be constructed. The model incorporates a horizontal (z_2^{-1}) shift, rather than the diagonal shift used in

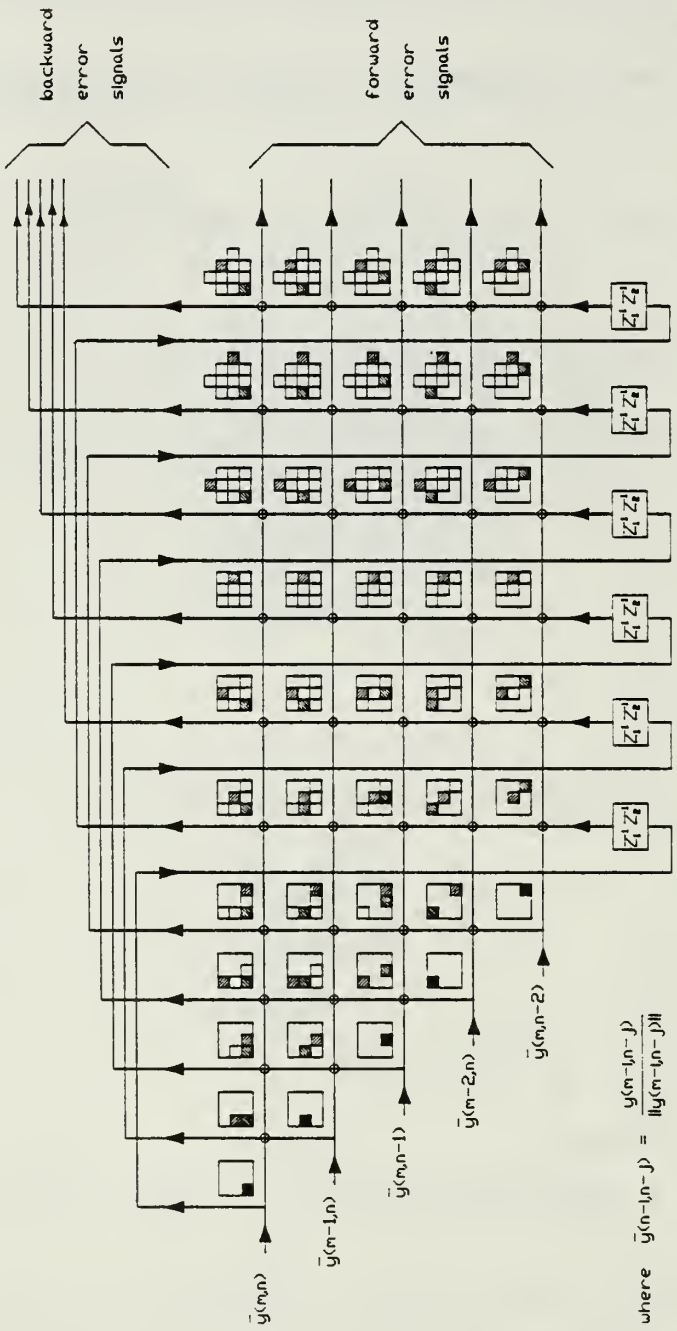


Figure 5.4: Reduced Complexity, $O(N^3)$, 2-D Quarter Plane Lattice Filter

the previous models. A horizontal shift also is a inner product preserving mapping so that its use is permitted.

The ordered index set used for this model is given by (for the second order case)

$$ {}_0^2L_{mn}^2 = \Big\{(m-2,n-2),(m-1,n-2),(m,n-2),(m-2,n-1),$$

$$(m-1,n-1),(m,n-1),(m-2,n),(m-1,n),(m,n)\Big\} \tag{5.23}$$

Define a horizontal shift operator, **H**, by the relation

$$y(m,n-1) = \mathbf{H}[y(m,n)] \tag{5.24}$$

The following relations hold due the assumed shift- invariance of the given data field

$$\bar{r}^0_{(m,n-1)} = \mathbf{H}[\bar{r}^0_{(m,n)}] \tag{5.25a}$$

$$\bar{r}^1_{(m-1,n-1)} = \mathbf{H}\,\bar{r}^1_{(m-1,n)} \tag{5.25b}$$

$$\bar{r}^2_{(m-2,n-1)} = \mathbf{H}\,\bar{r}^2_{(m-2,n)} \tag{5.25c}$$

$$\bar{r}^3_{(m-3,n-1)} = \mathbf{H}\,\bar{r}^3_{(m-3,n)} \tag{5.25d}$$

$$\bar{r}^4_{(m-4,n-1)} = \mathbf{H}\,\bar{r}^4_{(m-4,n)} \tag{5.25e}$$

$$\bar{r}^5_{(m-5,n-1)} = \mathbf{H}\,\bar{r}^5_{(m-5,n)} \tag{5.25f}$$

In general, whenever, (m-i,n-j) equals (m,n)-q then

$$\bar{r}^q_{(m-i,n-j-1)} = \mathbf{H}\,r^q_{(m-i,n-j)} \tag{5.26}$$

Using these simplifying relations and equations (5.11), the model of Figure 5.5 can be deduced. This model still contains $O(N^3)$ parameters, however, the actual number is only about one quarter of that required by the previous structure (Figure 5.4.)

This algorithm shares with the previous algorithm the shortcoming of having to estimate the maximum order of the filter at the outset. Despite this, it is believed that this model offers a good compromise (probably the best to date) between model accuracy and implementation complexity. In the next section it is demonstrated that the synthesis form of this algorithm has some particularly desirable properties. In Chapter 7, it will be shown that this algorithm is well suited for highly parallel VLSI implementation.

C. SYNTHESIS MODEL

The synthesis results of the previous chapter are easily extended to the 2-D case being considered. The data fields can be regenerated from a knowledge of the lattice coefficients and the forward error fields, it is not necessary to explicitly calculate the forward and backward error prediction coefficients.

The desired result is obtained by solving equation (5.11) for $\bar{e}_{kl}^{q-1}$ and $\bar{r}_{kl-q}^{q}$. This yields

$$\bar{e}_{kl}^{q-1} = \sqrt{1 - (\rho_q^{kl})^2}\, \bar{e}_{kl}^{q} + \rho_q^{kl} \bar{r}_{kl-q}^{q-1} \tag{5.27a}$$

$$\bar{r}_{kl-q}^{q} = \sqrt{1 - (\rho_q^{kl})^2}\, \bar{r}_{kl-q}^{q-1} - \rho_q^{kl} \bar{e}_{kl}^{q} \tag{5.27b}$$

These equations establish the method for regenerating the original data field. They describe the processing that must be carried out at each stage of the synthesis process. As an example of their application, consider the second order synthesis model pictured in Figure 5.6. It is the synthesis counterpart of the reduced complexity analysis model of Figure 5.5. In order to regenerate the original data field processing should be carried out horizontally by rows. For this second order model, two rows and two columns of initial conditions must be specified. The required zero order forward error sequences will always be available from either the given initial conditions or from previous estimates.

It will be shown that for VLSI implementations it will be more convenient to estimate all the zero order forward error sequences. This necessitates that three residuals to be input and that all forward error channels be reversed. Such a structure is illustrated in Figure 5.7.

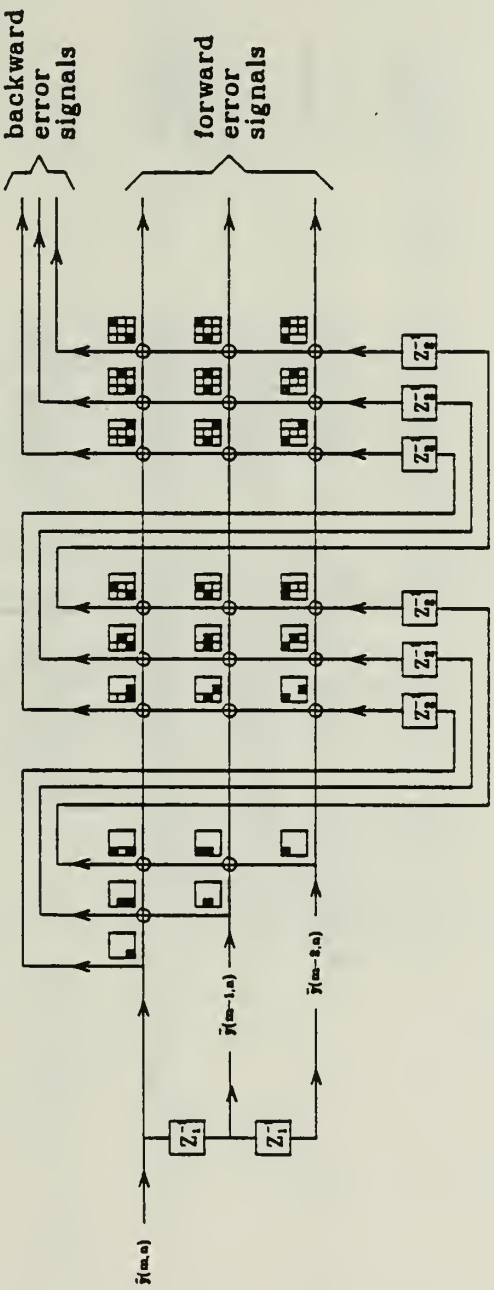


Figure 5.5: Alternate, Reduced Complexity, $O(N^3)$, Quarter Plane Lattice Filter.

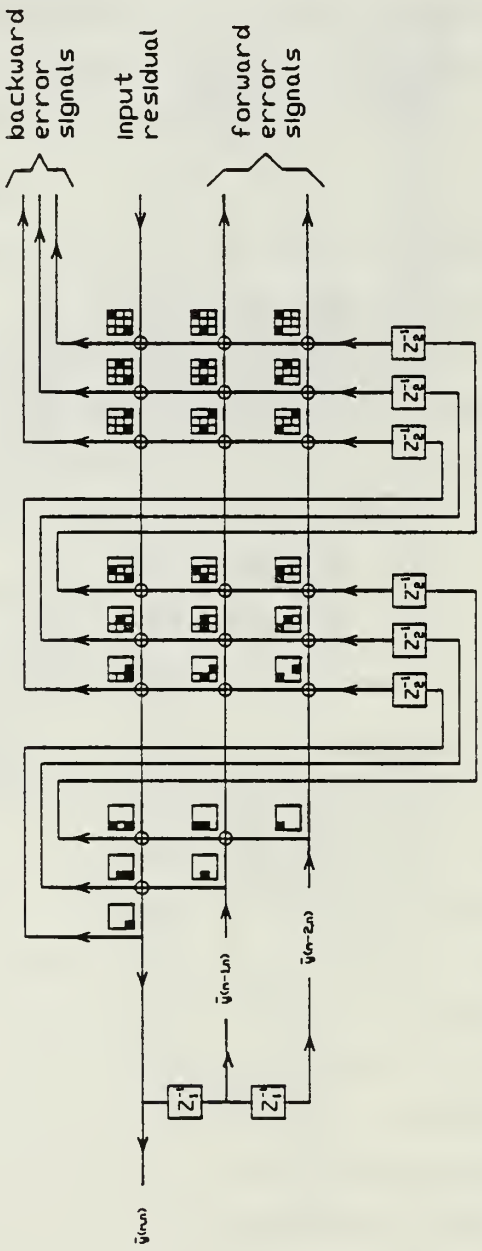


Figure 5.6: 2-D Quarter Plane Lattice Synthesis Model
Involving Only a Single Input Residual

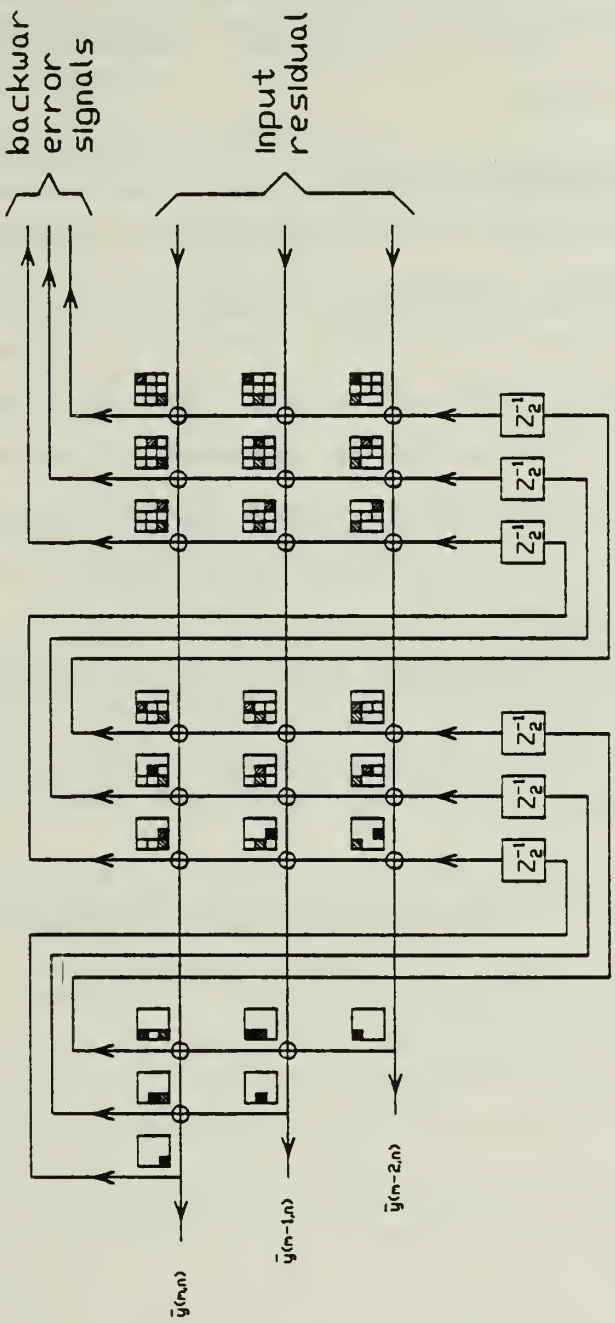


Figure 5.7: 2-D Quarter Plane Lattice Synthesis Model
Involving Three Input Residuals

A synthesis model corresponding to the analysis model of Figure 5.4 would have to process the data along diagonals. The required zero order forward error sequences would not be available at each stage in this case and so only a model analogous to the one of Figure 5.7 can be used. This would require the provision of five input residual signals.

D. SYSTOLIC IMPLEMENTATIONS

In the past, most signal processing algorithms were implemented largely in software due to their high complexity. More recently, with the advent of VLSI technology, there has been a shift towards specialized hardware implementations. These offer increased performance at a low per-unit cost. A particularly promising class of implementations, first suggested by Kung and Leiserson [Ref. 49]. are the so called systolic arrays. These attempt to partition the required computations in time and space over an array of identical processing elements, in order to increase throughput.

Kung [Ref. 50: pp. 869] defines a systolic array as a network possessing the following features:

> a)Synchrony: The data are rhythmically computed (timed by a global clock) and passed through the network.
>
> b)Regularity (i.e., Modularity and Local Interconnections): The array should consist of modular processing units with regular and (spatially) local interconnections. Moreover, the computing network may be extended indefinitely.
>
> c)Temporal Locality: There will be at least one unit-time delay allotted so that signal transactions from one node to the next can be completed.
>
> d) Pipelinability (i.e., O(M) Execution-Time Speed-Up): A good measure for the efficiency of the array is the following
>
> $$\text{Speed-Up Factor} = \frac{\text{Processing Time in a Single Processor}}{\text{Processing Time in the Array Processor}}$$
>
> A systolic array should exhibit a linear-rate pipelinability, i.e., it should achieve an O(M) speed-up, in terms of processing rates, where M is the number of processor elements (PE's).

Methods have been proposed for transforming algorithms into Systolic implementations beginning with either an **algorithmic** description (see Moldovan [Ref. 51]) or a **signal-flow-graph** (SFG) description (see Kung [Ref. 50]). In this chapter we shall be using the second of these methods to transform

one of the 2-D lattice structures into systolic form. Rather than present a detailed discussion of the rules used in this method, we shall simply state them and then apply them to produce the desired systolic array. It is hoped that this example will clarify the procedures used. If further insight is desired the reader is referred to the cited reference.

1. SFG Transformation Procedure

The procedure used is based on a **cut-set** approach. According to Kung [Ref. 50, pp. 870] a cut-set is defined as:

> A cut-set in an SFG is a minimal set of edges which partitions the SFG into two parts.

He proposes and proves that the following two rules that can be used to transform any computable SFG into a systolic array:

> Rule (i) Time-Scaling: All delays D may be scaled, i.e., D → D′, by a single positive integer . Correspondingly, the input and output rates also have to be scaled by a factor (with respect to the new time unit D′) ... .
>
> Rule (ii) Delay-Transfer: Given any cut-set of the SFG, we can group the edges of the cut-set into "inbound edges" and "outbound edges", depending upon the directions assigned to the edges. Rule (ii) allows advancing k (D′) time units on all the outbound edges and delaying k time units on the inbound edges, and vice versa. It is clear that, for a (time-invariant) SFG, the general system behaviour is not affected because the effects of the lags and advances cancel each other in the overall timing. Note that the input-input and input-output timing relationships will also remain exactly the same only if they are located on the same side. Otherwise they should be adjusted by a lag of −k time units or an advance of -k time units.

2. Systolic Implementation of 2-D Lattice Filter

Using the two rules given in the previous section the 2-D lattice synthesis model of Figure 5.7 will be mapped into a systolic array. There is some flexibility in the design, the result not being unique. The first choice that must be made is that of the operation that is to be performed by the basic PE. In the case being considered several convenient choices are possible. The simplest element would be a multiplier-adder. A slightly higher level operation would be that of the single lattice section given by Figure 4.5. A still higher level operation is conceivable by grouping several of the lattice sections together. We shall use the second of these choices as it illustrates the general procedure without the added complexity inherent in the lower level implementation.

The mapping will be done in stages. Initially, the diagram of Figure 5.7 will be redrawn in an SFG format. Rule (i) will be used to scale all the delays appropriately. Then, by successive application of Rule (ii), the SFG will be temporally localized (it is already spatially local.) The steps used are outlined as follows;

(1) In Figure 5.8 the algorithm of Figure 5.7 has been redrawn in a SFG form. In this and subsequent diagrams delays are indicated by the letter D. The lattice section of Figure 4.5, as before, is indicated by the circles at the nodes.

(2) Using rule(i), all delays are scaled by a factor of 6. That is, $D \rightarrow 6D'$. This is indicated in Figure 5.9. The input and output signal rates must also be scaled by this factor of 6.

(3) Using the cut-sets indicated in Figure 5.10 the delays are redistributed so that temporal locality is achieved. The resulting SFG is indicated in Figure 5.11. Until now the processing at each node was assumed not to take any time. The delays going into each node can be combined with the lattice sections and be used to account for the processing time. In this way, the structure will appear as in Figure 5.12. In this last figure, the nodes have been shaded to indicate that the operation being performed within them consumes one time unit.

3. Additional Remarks

In this section we have shown that the 2-D Lattice structures derived in this chapter are amenable to a systolic implementation. This is significant as the processing of 2-D data fields such as images in real-time requires high data rates. These rates can only be achieved in practice through the use of super-computers or specialized hardware. Due to the high cost of super-computers the second alternative is the more practical. With the costs of VLSI production rapidly decreasing, it is now cost effective to produce dedicated chips even in very small quantities. For large scale productions the cost can be amortized over a large number of chips, yielding a low per-unit cost.

Although only a single specific implementation has been presented here, it indicates the ease with which the other algorithms discussed in this thesis may be transformed into forms which can be efficiently implemented in silicon.

E. SIMULATION RESULTS

The theory has been proven by computer simulation. Two different order models were excited by a white noise process. Several different order estimates of each spectrum were generated and compared to the originals.

1. Example 5.1

The first model simulated was described by

$$\begin{aligned} y(m,n) = {} & .295y(m-1,n) - .470y(m,n-1) + 0.0y(m-1,n-1) \\ & - .055y(m-2,n) + .007y(m,n-2) + .003y(m-2,n-1) \\ & - .015y(m-1,n-2) + .022y(m-2,n-2) + u(m,n) \end{aligned} \tag{5.28}$$

where u(m,n) was a 2-D zero-mean white noise process.

The original spectrum is illustrated in Figure 5.13 while the first, second, third, and fourth order estimates are given in Figure 5.14. Notice that the original model is only second order so that the third and fourth order estimates can be used to examine the effects of over modelling. As can be seen from the figures the estimated spectra correspond very closely to the originals. Over modelling did not noticeably degrade the accuracy of the estimates.

The actual algorithm used was that of Figure 5.2. Although it is unnecessarily complex, it is the most straight forward to implement. The generalized 2-D Schur algorithm was used to generate all the required lattice parameters. The computer subroutines used to accomplish this simulation are included in Appendix B.

2. Example 5.2

The second simulation used the following higher order autoregressive equation to generate the data

$$y(m,n) = -.47y(m-1,n) - .03y(m,n-1) + .195y(m-1,n-1)$$

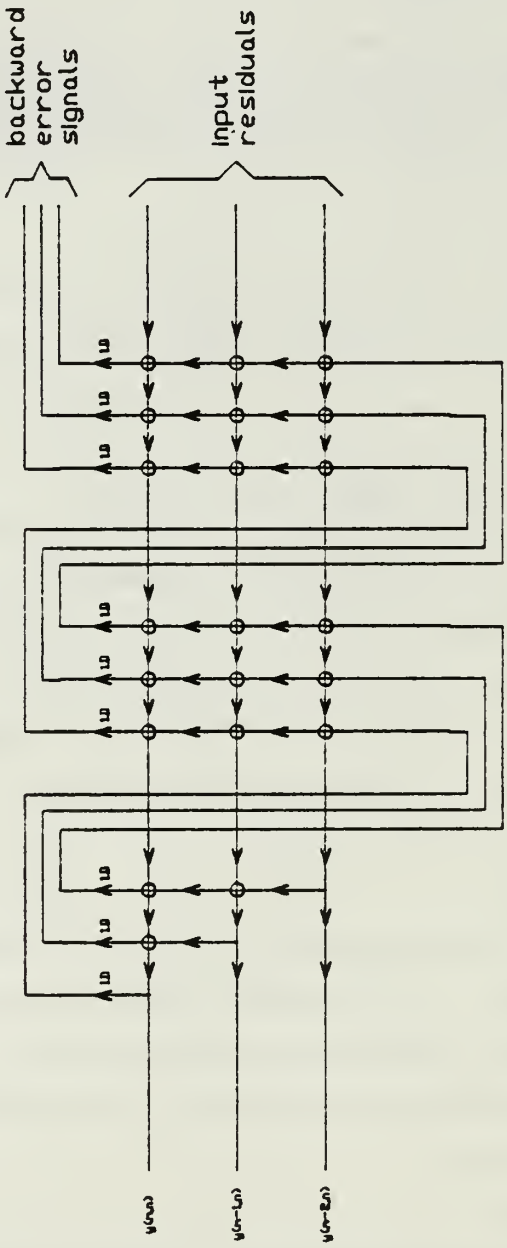


Figure 5.8: 2-D Lattice Filter of Figure 5.7 in SFG Form.

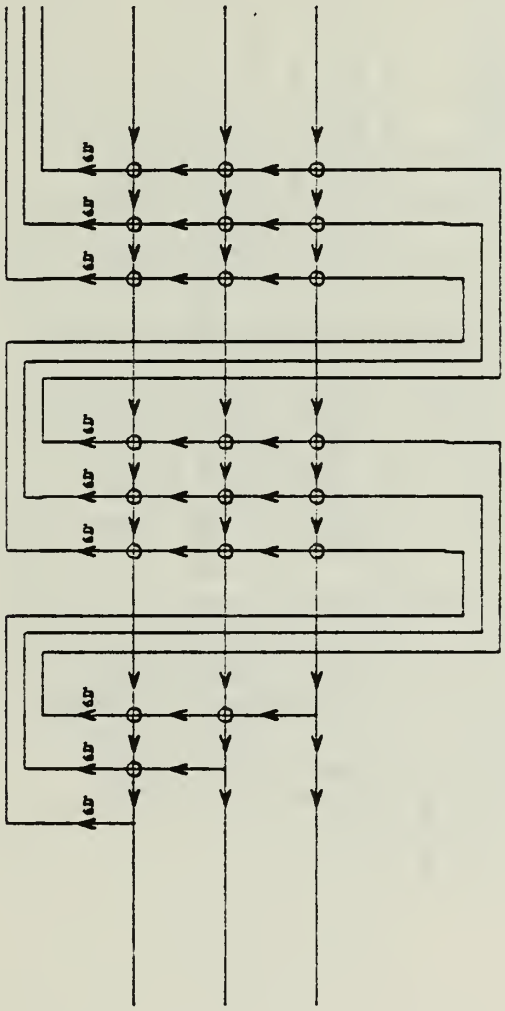


Figure 5.9: The Delays are Scaled by a Factor of Six.

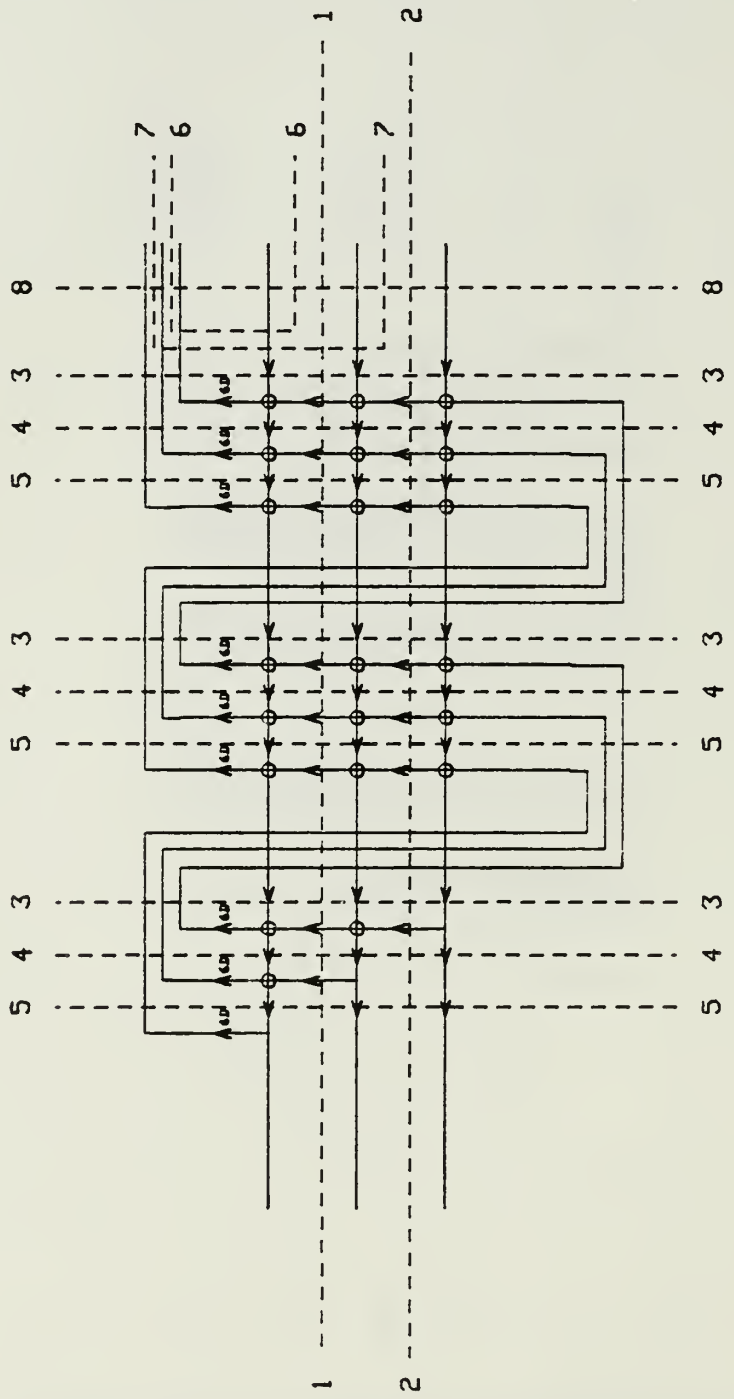


Figure 5.10: SFG for 2-D Lattice Filter Showing Cut-Sets to be Used

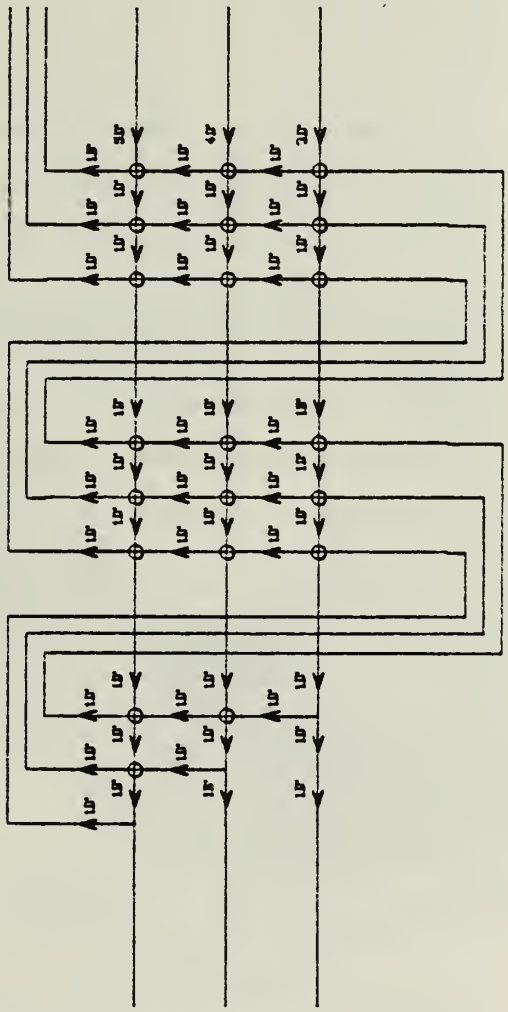


Figure 5.11: Temporally Local Version of 2-D Lattice Filter

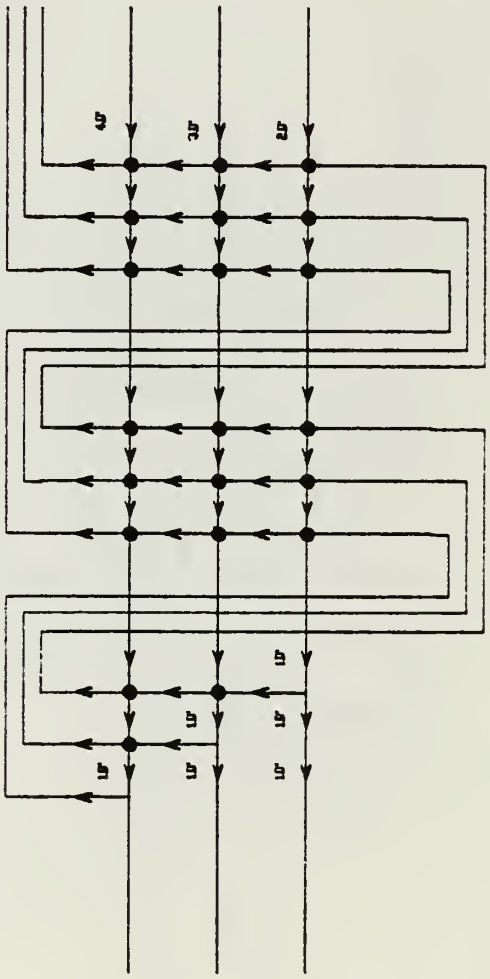


Figure 5.12: Temporally Local 2-D Lattice Filter.
Shaded Nodes Indicate the Inclusion of a Delay

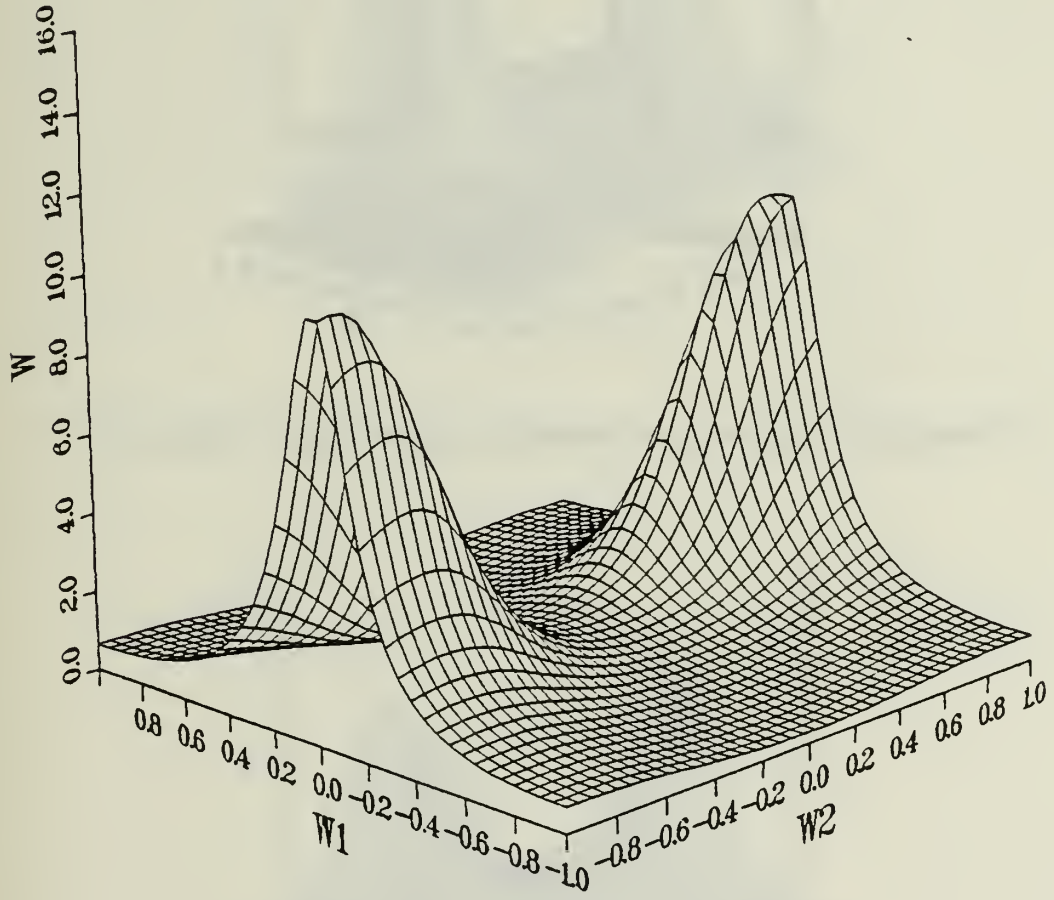
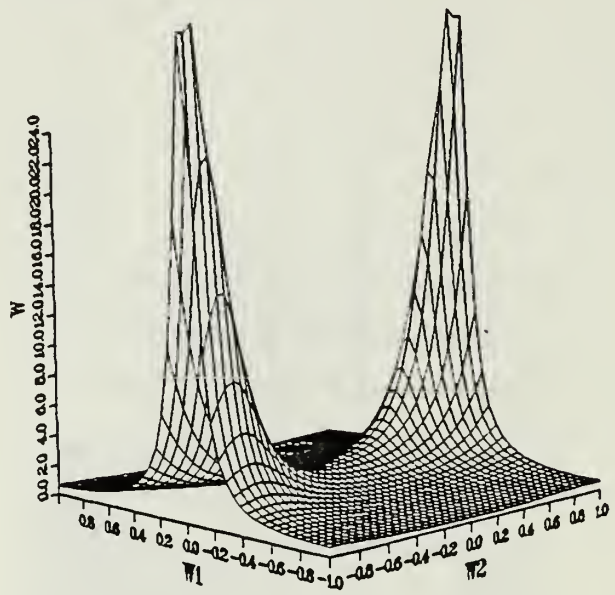
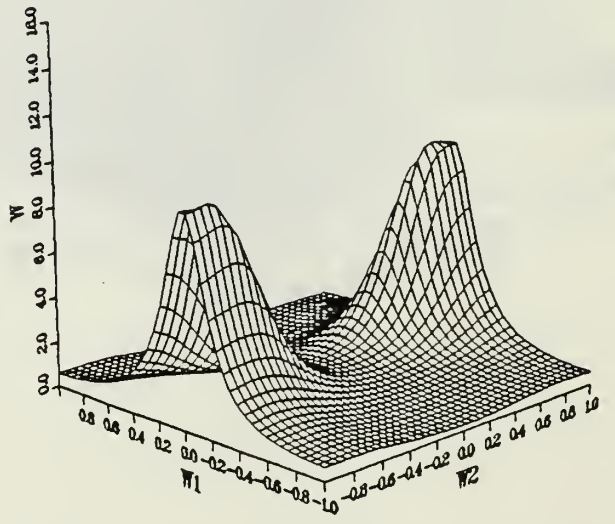


Figure 5.13: Original Spectrum For Example 5.1.

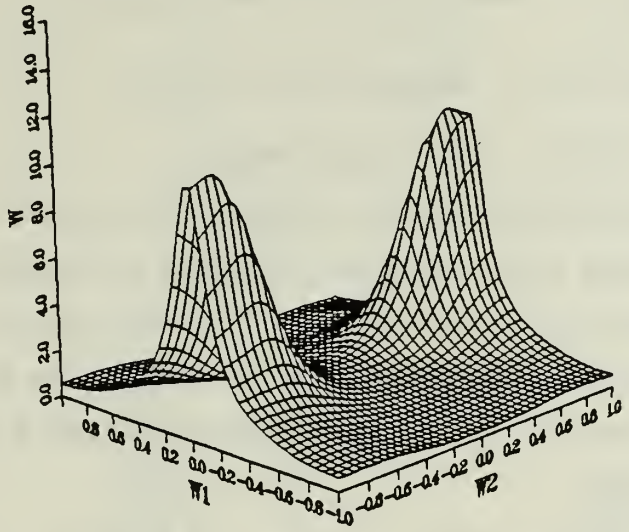


a. First Order Lattice Approximation of Spectrum of Example 5.1.

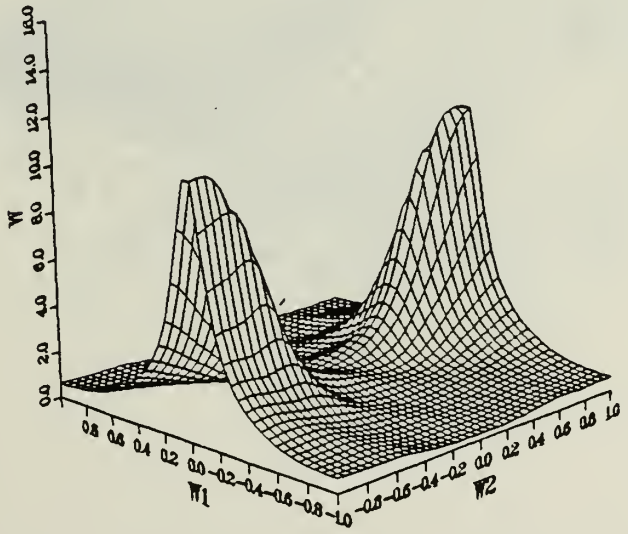


b. Second Order Lattice Approximation of Spectrum of Example 5.1.

Figure 5.14: Lattice Approximations Of Spectrum of Example 5.1



c. Third Order Lattice Approximation of Spectrum of Example 5.1.



d. Fourth Order Lattice Approximation of Spectrum of Example 5.1.

$$- .015y(m,n-2) + .055y(m-1,n-2) - .003y(m-2,n-2)$$

$$+ .0067y(m-3,n) - .015y(m,n-3) + .022y(m-2,n-3)$$

$$+ .033y(m,n-4) - .085y(m-1,n-4) - .002y(m-4,n-2)$$

$$- .0001y(m-2,n-4) + .0001y(m-4,n-4) + u(m,n) \qquad (5.29)$$

where u(m,n) is a zero-mean white noise process.

The spectrum of this model is shown in Figure 5.15. The first through fourth order estimates of the spectrum are shown in Figures 5.16. The general shape of the spectrum is identified in the first order model, although the fine detail is not introduced until the fourth order model. The position and relative magnitude of the peaks in the spectra are identified with great accuracy in the fourth order estimate.

In the next chapter lattice models are applied to the solution of the autoregressive nonlinear modelling problem.

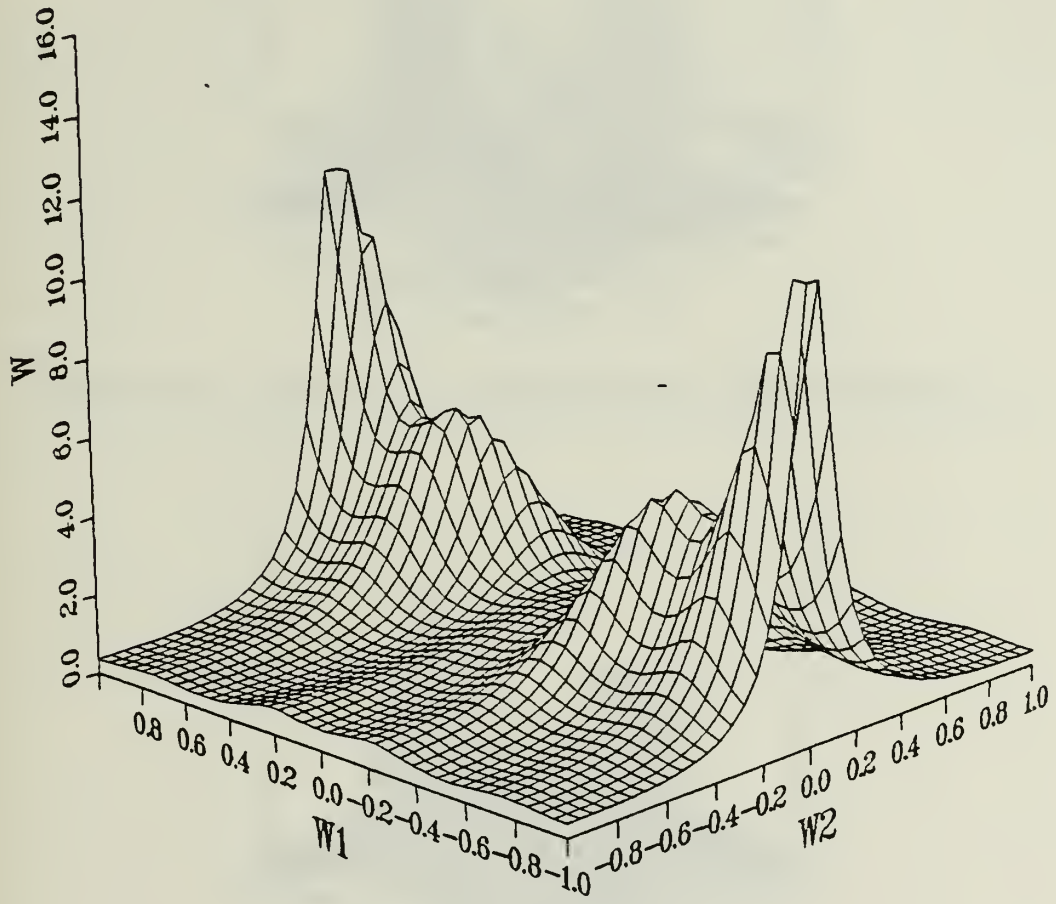
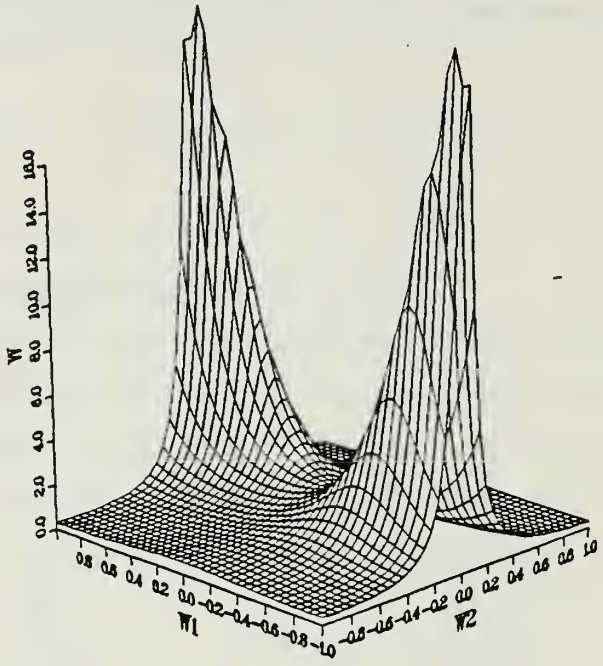
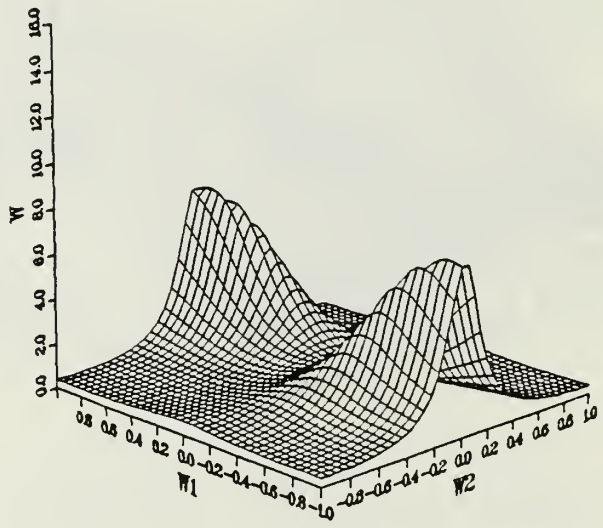


Figure 5.15: Original Spectrum For Example 5.2.

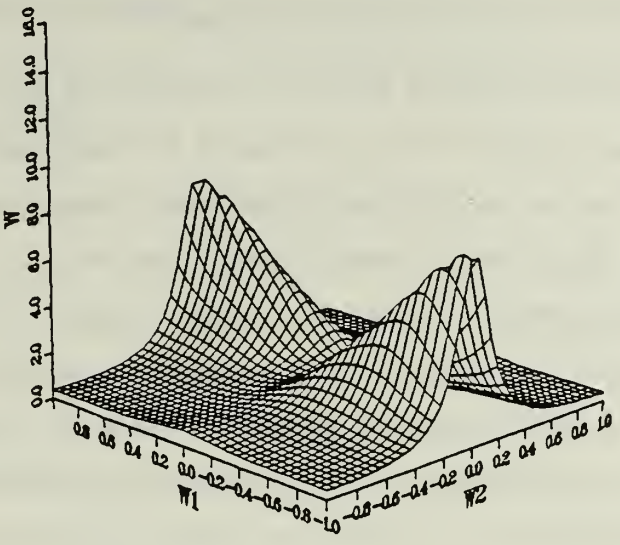


a. First Order Lattice Approximation of Spectrum of Example 5.2.

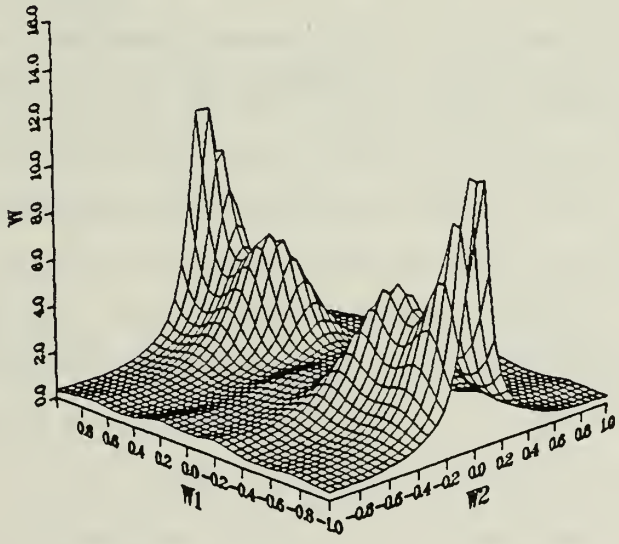


b. Second Order Lattice Approximation of Spectrum of Example 5.2.

Figure 5.16: Lattice Approximations Of Spectrum of Example 5.2



c. Third Order Lattice Approximation of Spectrum of Example 5.2.



d. Fourth Order Lattice Approximation of Spectrum of Example 5.2.

VI. NONLINEAR LATTICE STRUCTURES

In the previous two chapters lattice structures have been examined in great length and have been applied to the problem of 2-D autoregressive modelling. In this chapter we return to the problem of nonlinear autoregressive modelling with the hope of solving the problem using a lattice structure.

There have been several attempts in the literature to fit lattice filters to a Volterra like model. The first was due to Parker and Perry [Ref. 15]. They proposed a multichannel lattice implementation of an autoregressive-moving average nonlinear model. Their model was capable of providing an update in time of some terms of the expansion. It, however, does not introduce all the terms that are needed to constitute the next higher order model.

A recent proposal by Zarzycki and Dewilde [Ref. 17] is a true generalization of the Levinson algorithm to the case of the autoregressive Volterra type model. In their work they considered a non-stationary nonlinear structure and showed that the stationary and linear models can be treated as special cases. Their model provides a true update in time order, introducing all the required terms. They found. as we did with the 2-D model. that not all terms could be recursively generated and that some would have to be introduced at each stage by other means. For these non-linear models only one error signal must be injected at each stage not two as in the 2-D case (if a triangular kernel is used.)

The model proposed in this chapter is based on the alternate tensor form of the nonlinear model developed in Chapter 3. Recall that the model was based on interchanging the roles played by time order and nonlinear order. The lattice model presented here thus becomes recursive in nonlinear order. This means. for example. that the optimum cubic model is calculated from a knowledge of the optimum quadratic model. The theory is based on the generalized lattice concepts of Chapter 4 and the 2-D lattice structures developed in Chapter 5.

Although only the nonlinear order update model is discussed. some reflection will show that a time update can also be performed using the proposed model.

For simplicity in illustrating the concepts only models involving two delays will be considered in this chapter. The theory is equally valid and easily extended to more complex situations.

A. GENERAL NONLINEAR LATTICE MODEL

As with the 2-D model of the previous chapter, the theory used is that of the generalized error order updates given in Chapter 4. A careful choice of the input data will yield the desired results. We begin by summarizing briefly the nonlinear model to be used, for the case of two delays.

The estimate of the model output is given by (3.53)

$$y(k) = y^{\lambda_1}(k-1) \cdots y^{\lambda_N}(k-N)H_{\lambda_1 \cdots \lambda_N} \tag{6.1}$$

To avoid unnecessarily complex algebra we consider the case when N=2.

$$y(k) = y^{\lambda_1}(k-1)y^{\lambda_2}(k-2)H_{\lambda_1\lambda_2} \tag{6.2}$$

The tensor product $y^{\lambda_1}(k-1)y^{\lambda_2}(k-2)$ defines a second order tensor, $\mathbf{Y}(k)$, with components given by

$$\mathbf{Y}(k) = [y^{\lambda_1}(k-1)y^{\lambda_2}(k-2)] =$$

$$= \begin{bmatrix} 1 & y(k-2) & y^{(2)}(k-2) & \cdots & y^{(p)}(k-2) \\ y(k-1) & y(k-1)y(k-2) & y(k-1)y^{(2)}(k-2) & \cdots & y(k-1)y^{(p)}(k-2) \\ y^{(2)}(k-1) & y^{(2)}(k-1)y(k-2) & y^{(2)}(k-1)y^{(2)}(k-2) & \cdots & y^{(2)}(k-1)y^{(p)}(k-2) \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ y^{(p)}(k-1) & y^{(p)}(k-1)y(k-2) & y^{(p)}(k-1)y^{(2)}(k-2) & \cdots & y^{(p)}(k-1)y^{(p)}(k-2) \end{bmatrix} \tag{6.3}$$

This tensor can be considered to be a shift-varying data field and can thus be modelled using the 2-D lattice model of Figure 5.2. A tensor of the form given in (6.3) can be formed for each time, k. If the process is time-varying then each of

these tensors must be separately modelled. If time-invariance can be assumed, then only one model need be developed. In that case the required correlations can be calculated over the ensemble of such tensors.

A shift along the diagonal (or horizontal or vertical) of the $\mathbf{Y}(k)$ tensor is not a inner-product preserving operation. This prevents the application of the reduced complexity algorithms of Chapter 5 to this nonlinear problem.

The 2-D lattice model of the tensor $\mathbf{Y}(k)$ does not offer a complete solution. Notice that $y(k)$, the sample that is to be estimated, does not appear in equation (6.3). Two possible solutions to this problem exist. First, the ordered index set, (5.3), can be extended by one element so that $y(k)$ is included in the model. This has the effect of adding a channel to the general 2-D lattice structure of Figure 5.2. A conceptually different solution is to first model the tensor $\mathbf{Y}(k)$ using the results of the previous chapter. The backwards error signals that result can be used as a basis for a Fourier expansion of the the sample $y(k)$ (see equation (4.73) for details.) Both of these approaches lead to identical models but do provide alternate interpretations. The second approach will be the one used in the derivations of this chapter as it allows the results of the previous chapter to be applied with little modification.

The tensor, $\mathbf{Y}(k)$, can be considered to be a two- dimensional data field given by

$$\mathbf{Y}(k) = \left\{ y^{\lambda_1}(k-1) y^{\lambda_2}(k-2) \right\} \tag{6.4}$$

where $(\lambda_1, \lambda_2) \in {}^2L^P = L^P \times L^P$

where L^P is an index set given by

$$L^P = \left\{ 0, \ldots, p \right\} \tag{6.5}$$

Points will be used from this data field in a particular, convenient order. We define an ordered index set

$$\overset{2}{\circ}L^P = \Big\{(p,p),(p-1,p),(p,p-1),\ \ldots,\ (0,p),(p,0),$$

$$(p-1,p-1),(p-2,p-1),(p-1,p-2),\ \ldots,\ (0,p-1),(p-1,0),$$

$$\ldots,(1,1),(0,1),(1,0),(0,0)\Big\} \tag{6.6}$$

The $(p+1)^2$ elements of ${}^2L^P$ have been ordered into a one dimensional index set, $\overset{2}{\circ}L^P$. The elements of $\overset{2}{\circ}L^P$ can be numbered, consecutively, from 0 to $(p+1)^2-1$.

As in Chapter 5, the notation (m,n) - q will be used to mean; the element of the index set corresponding to the q-th element prior to the element (m,n). For example, (1.0) - 2 would indicate the element (1,1) (see equation (6.6)). This notation will often be abbreviated to simply mn-q.

Define the (q-1) order, normalized, forward error associated with the prediction of the element $y^m(k-1)y^n(k-2)$ from the previous (in the sense of $\overset{2}{\circ}L^P$) (q-1) points, as

$$\bar{e}_{mn}^{q-1} = a_{\lambda_1\lambda_2}^{q-1}(m,n)y^{\lambda_1}(k-1)y^{\lambda_2}(k-2) \tag{6.7}$$

where the implied summation over $(\lambda_1,\lambda_2) \in {}^2L^P$, can be carried out in any order, as long as all components are considered. It is preferred to think of (λ_1,λ_2) belonging to ${}^2L^P$ rather than $\overset{2}{\circ}L^P$ as this maintains the multi-dimensional character of the problem.

The $a_{\lambda_1\lambda_2}^{q-1}(m,n)$ can be interpreted as the components of a second order covariant tensor. These components, for a range of indices $(\lambda_1,\lambda_2) < mn-q$ (in the sense of (6.6)), or for indices $(\lambda_1,\lambda_2) > (m,n)$ are equal to zero. For the case when $(\lambda_1,\lambda_2) = (m,n)$, the component

$$a_{mn}^{q-1}(m,n) = \frac{1}{e_{mn}^{q-1}} \tag{6.8}$$

where e_{mn}^{q-1} is an unnormalized version of $\bar{e}_{mn}^{q-1}$.

A normalized backward error associated with the prediction of y((m,n)-q) from the next q-1 points of ${}_{\delta}^{2}L^{P}$, is defined by

$$\bar{r}_{mn-q}^{q-1} = b_{\lambda_1\lambda_2}^{q-1}(mn-q)y^{\lambda_1}(k-1)y^{\lambda_2}(k-2) \tag{6.9}$$

As with the forward error prediction coefficients, the $b_{\lambda_1\lambda_2}^{q-1}(mn-q)$ can be interpreted as components of a second order covariant tensor. The components $b_{\lambda_1\lambda_2}^{q-1}(mn-q)$ equal zero for the range of indices $(\lambda_1,\lambda_2) < (mn-q)$ or $(\lambda_1,\lambda_2) > (m,n)$. For the case when the index $(\lambda_1,\lambda_2) = (mn-q)$ the component

$$b_{mn-q}^{q-1}((m,n)-q) = \frac{1}{\| r_{mn-q}^{q-1} \|} \tag{6.10}$$

where r_{mn-q}^{q-1} is an unnormalized version of $\bar{r}_{mn-q}^{q-1}$.

1. Normalized Order Update Recursions

The following two theorems are stated without proof. The proofs are identical to those of Theorems 5.1 and 5.2 of the previous chapter and so are not repeated.

a. Theorem 6.1: Normalized Nonlinear Levinson Algorithm

The nonlinear prediction error coefficients can be updated in order using the following recursions

$$\begin{bmatrix} a_{\lambda_1\lambda_2}^{q}(m,n) \\ b_{\lambda_1\lambda_2}^{q}((m.n)-q) \end{bmatrix} = \Theta(\rho_q^{mn}) \begin{bmatrix} a_{\lambda_1\lambda_2}^{q-1}(m,n) \\ b_{\lambda_1\lambda_2}^{q}(m,n) \end{bmatrix} \tag{6.11}$$

where

$$\Theta(\rho_q^{mn}) \quad \frac{1}{\sqrt{1 - (\rho_q^{mn})^2}} \begin{bmatrix} 1 & -\rho_q^{mn} \\ -\rho_q^{mn} & 1 \end{bmatrix} \tag{6.12}$$

and

$$\rho_q^{mn} = E\left\{ e_{mn}^{q-1} \bar{r}_{mn-q}^{q-1} \right\} \tag{6.13}$$

b. Theorem 6.2: Normalized Error Order Update Algorithm

The nonlinear prediction errors can be updated in order according to the following equation

$$\begin{bmatrix} \bar{e}^{q}_{mn} \\ \bar{}^{q}_{mn-q} \end{bmatrix} = \Theta(\rho_q^{mn}) \begin{bmatrix} \bar{e}^{q-1}_{mn} \\ \bar{}^{q-1}_{mn-q} \end{bmatrix} \tag{6.14}$$

In order to introduce the sample y(k) into the model we recognize the backwards errors as an alternate coordinate system. In particular the following vector is defined.

$$[y^{\lambda}] = [\bar{r}^{\lambda'}_{(0,0)-\lambda'}] = \begin{bmatrix} \bar{r}^{0}_{(0,0)} \\ \bar{r}^{1}_{(0,0)-1} \\ \bar{r}^{2}_{(0,0)-2} \\ \cdot \\ \cdot \\ \cdot \\ \bar{r}^{(p+1)^2-1}_{(0,0)-(p+1)^2+1} \end{bmatrix} \tag{6.15}$$

Equations (6.14) correspond to a model structure identical with that given in Figure 6.2. In this diagram, the backwards errors given in equations (6.15) correspond to those that are shown leaving the uppermost stages of the filter.

The forward error in predicting y(k) from the $\mathbf{Y}(k)$ tensor is given by

$$\bar{e}_k^{(p-1)^2} = y(k) - K_{\lambda'} \cdot y^{\lambda'} \tag{6.16}$$

Because the backward errors are uncorrelated it is a straight forward matter to find the (Fourier) coefficients, $K_{\lambda'}$. They are given by

$$K_{\lambda'} = \left[E\left\{ y(k) r^{0}_{(0,0)} \right\} \quad E\left\{ y(k) r^{1}_{(0,0)-1} \right\} \quad \cdots \quad E\left\{ y(k) r^{(p+1)^2-1}_{(0,0)-(p+1)^2+1} \right\} \right] \tag{6.17}$$

The m-th component is

$$K_m = E\left\{y(k)\bar{r}^m_{(0,0)-m}\right\} \tag{6.18a}$$

$$= E\left\{e_k^m \bar{r}^m_{(0,0)-m}\right\} \tag{6.18b}$$

$$= \| e_k^m \| E\left\{\bar{e}_k^m \bar{r}^m_{(0,0)-m}\right\} \tag{6.18c}$$

$$= \| e_k^m \| \rho_m^k \tag{6.18d}$$

A normalized version of the forward error defined in (6.16) is thus given by

$$\bar{e}_k^m = \frac{1}{\| e_k^m \|}\left[y(k) - \sum_{\lambda'=0}^{m-1} \rho^k_{\lambda'} \bar{r}^{\lambda'}_{(0,0)-\lambda'} \| e_k^{\lambda'} \| \right] \tag{6.19}$$

Recognizing that y(k) is a zero order forward error we can write a normalized version of the q-th order forward error as

$$\bar{e}_k^q = \frac{\| e_k^{q-1}}{\| e_k^q} \left[e_k^{q-1} - \rho_q^k \bar{r}^{q-1}_{(0,0)-q} \right] \tag{6.20}$$

This last result follows if (6.19) is iterated. calculating successively higher order errors or from equation (4.73) where the equivalence of the lattice model and Fourier series was established.

We conclude that expressing y(k) as a stochastic Fourier series (with the backward errors as a basis) amounts to nothing more than the addition of a supplementary channel to the existing lattice structure. Equations (6.11) and (6.14) hold for this additional channel and can thus be used to update the normalized error signals and model parameters. Figure 6.1 illustrates a quadratic nonlinear lattice filter.

2. Uniqueness Of Lattice Parameters

a. Theorem 6.3

The lattice parameterization of the sequence y(k) is unique. That is. the lattice parameters given by (6.13) are unique.

b. Proof (Outline)

The truth of this theorem is a consequence of the fact that the lattice can be interpreted as a Fourier series expansion of the sequence y(k) in terms of the orthonormal backwards errors (see equation (6.17).) The uniqueness of the Fourier series is well known [Ref. 47].

3. Synthesis Models

A model analogous to that illustrated in Figure 4.7 can be used to regenerate the original sequence. This requires knowledge of the forward error residual sequence. The other inputs (zero order forward errors) required can all be obtained from past estimates of the sequence or from initial conditions.

The probability distributions of the output forward error sequences must be known if a synthesis model is to be constructed which accurately reproduces the original signal statistics. It is not clear that any general statements can be made about the nature of these distributions. It has been shown [Ref. 48: p. 357] that in the continuous case they will be gaussian and of the same variance as the original additive gaussian noise source. In the same reference it is also shown that for the discrete case the error sequence will not be gaussian although it will be white. The inaccuracies introduced through the use of gaussian noise need to be investigated. The stability of these lattice models is also left as an open question.

B. SIMULATION RESULTS

In order to verify the theory, a computer simulation was performed. Uniform white noise was used to excite the system. This choice provides a bounded input so that the response of the system used for the simulation could be guaranteed to be stable for a suitable choice of system parameters. A sufficient condition for the AR model to be stable, given a driving signal whose absolute value is bounded on (-a,a), where $0 < a < 1$, is given by

$$\sum_{\lambda_1=1}^{F} \cdots \sum_{\lambda_N=1}^{F} H_{\lambda_1 \cdots \lambda_N} + |a| < 1 \qquad (6.21)$$

This condition guarantees that the output never exceeds unity and thus remains

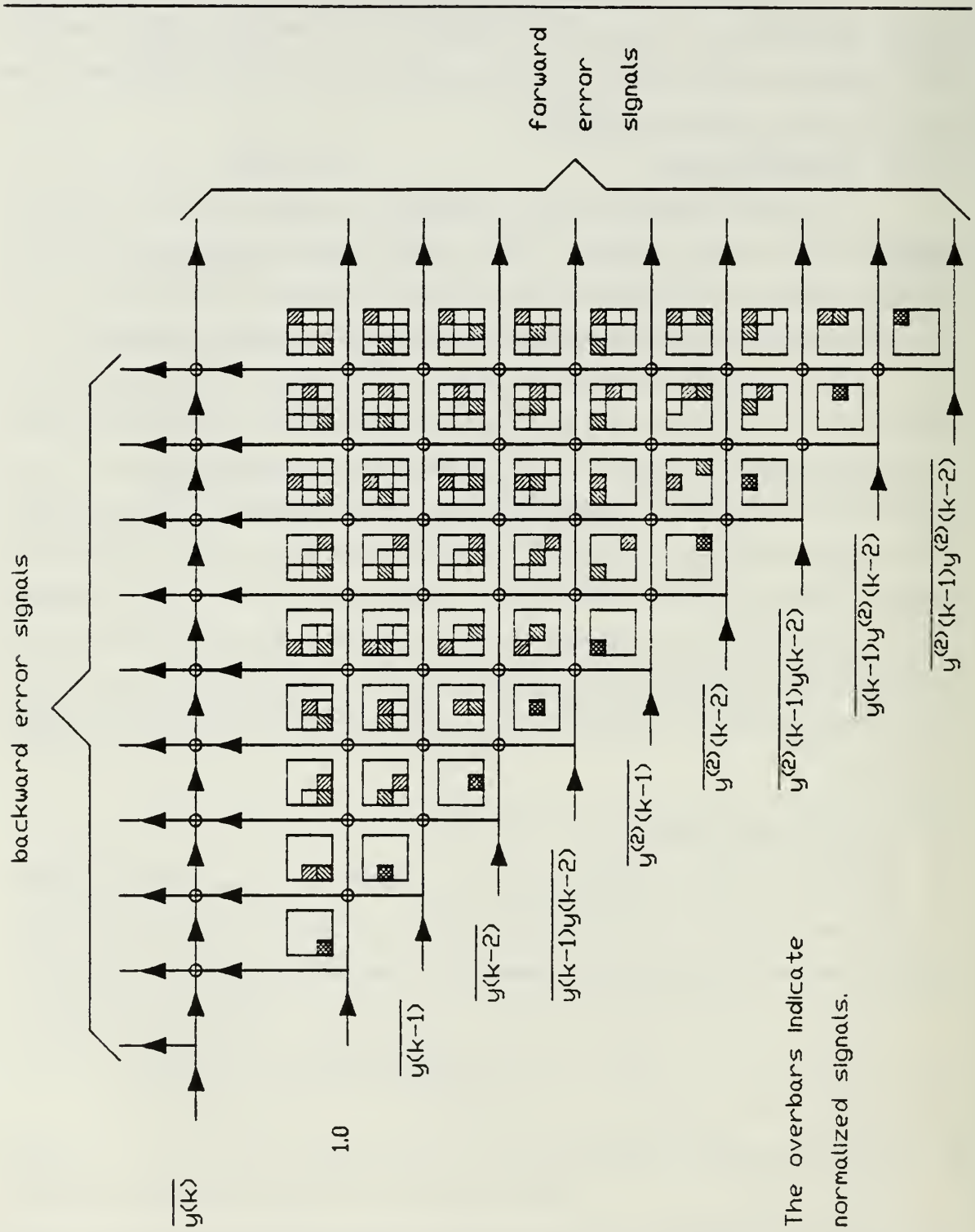


Figure 6.1: Quadratic Nonlinear Lattice Filter

bounded. This condition is extremely restrictive. however, it does provide a simple rule for building models which can be used for simulation purposes. (The issue of the inaccuracy introduced by using uniform noise to drive the system has been avoided.)

The model tested was

$$[H_{\lambda_1\lambda_2}] = \begin{bmatrix} -.1 & .22 & .02 \\ .02 & .09 & .001 \\ .2 & .05 & -.03 \end{bmatrix} \tag{6.22}$$

Using the nonlinear form of the Levinson algorithm, equation (6.11) the following model parameters were estimated for two repetitions of the experiment,

$$[H_{\lambda_1\lambda_2}]' = \begin{bmatrix} -.1003 & .26038 & .04568 \\ .01402 & .10198 & .03877 \\ .1998 & .03230 & -.03965 \end{bmatrix} \tag{6.23}$$

$$[H_{\lambda_1\lambda_2}]'' = \begin{bmatrix} -.1009 & .2612 & .06213 \\ .01151 & .09917 & .03125 \\ .19020 & .04199 & -.01432 \end{bmatrix} \tag{6.24}$$

VII. CONCLUSIONS AND DISCUSSION

A. SUMMARY OF NEW RESULTS

In this dissertation the use of tensor concepts in the field of signal processing was investigated. The research was successful in a number of areas, extending known results and introducing some new ones. In particular, tensors were used to study nonlinear and multidimensional systems.

The nonlinear modelling problem was formulated using tensors. By interchanging the roles played by the time order and the nonlinear order a new form, different from (although equivalent to) the traditional Volterra Series was developed. Using this new form, tensor equivalents of the discrete-time Wiener-Hopf and Yule-Walker equations were derived. These equations can be solved for the unknown system parameters. Conditions for the solution of the Wiener-Hopf equations, without requiring matrix inversion, were specified. This resulted in a computational saving of $O((p+1)^{3(N-1)})$ operations, where N is the largest time delay present and p is the highest exponent present in the system model. It was further shown that linear adaptive algorithms, such as LMS and RLS, can be applied to solve for the system parameters.

Existing Lattice filter algorithms were reformulated in a tensor framework. It was shown that they can be considered to be equivalent expansions in alternate coordinate systems. These results were then applied to the solution of the 2-D autoregressive modelling problem. Several new 2-D lattice structures were deduced. These models are not efficient in the sense that an AR model possessing $O(N^2)$ parameters would require $O(N^4)$ parameters when recast into a lattice form. It was shown that with proper assumptions of shift-invariance the complexity of the lattice models can be reduced to $O(N^3)$ parameters.

The 2-D lattice models developed were then used to deduce a nonlinear lattice model. This model differs from that of other researchers in that it allows updates to be performed in the nonlinear order as well as time order.

Finally, it was shown that these lattice models are amenable to systolic array implementations.

B. FUTURE DIRECTIONS FOR RESEARCH

The multidimensional and nonlinear lattice theories are by no means complete. So far, several new lattice models have been developed. It now remains to investigate the properties of the models.

For the 2-D lattice filters, conditions for model stability have yet to be established. The stability of multidimensional systems is a much more difficult problem than for the 1-D case because it is usually impossible to factor multivariate polynomials. However, necessary and sufficient conditions for AR 2-D model stability have been established. It is believed that these can be used to derive lattice stability conditions.

Another important topic is the synthesis of 2-D transfer functions using lattice structures. Stated differently, the problem is to design a 2-D lattice parameter filter that will implement a given 2-D frequency response characteristic.

For the nonlinear lattice structures similar issues need to be addressed. In this case, however, the problems are more complex since the stability and synthesis questions have not been solved for the nonlinear AR models.

Before the synthesis problem for the lattice implementation can be solved several more basic questions need to be answered. The first is the definition of a useful and meaningful specification of the desired filter characteristic. The nonlinear AR filter affects not only the frequency content of the driving signal but also its probability distribution. The filter will also have different frequency characteristics for different driving signals. All of these effects should be included in the specification.

It has been shown that for the continuous case, a nonlinear whitening filter will yield a signal with a gaussian probability distribution [Ref. 48: p. 357]. For the discrete case it has been shown that this is not true. The inaccuracy introduced by approximating the input to the synthesis model by a gaussian

introduced by approximating the input to the synthesis model by a gaussian signal is an important question.

Stability of the nonlinear lattices is also an important question. No simple stability tests exist for the nonlinear AR models. Their recursive nature makes stability difficult to analize. In general, stability will be a function of the input as well as the system which significantly complicates the problem.

LIST OF REFERENCES

1. Kron, G., *Tensor Analysis of Networks,* John Wiley and Sons, Inc., 1939.

2. Yaglom, A.M., *Introduction to the Theory of Stationary Random Functions,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1962.

3. Lev-Ari, H., Kailath, T., Cioffi, J., "Least-Squares Adaptive Lattice and Transversal Filters: A Unified Geometric Theory," *IEEE Trans. on Info. Theory,* Vol. IT- 30, No. 2, March 1984, pp. 222-234.

4. MacLennan, B.J., *Principles of Programming Languages: Design, Evaluation, and Implementation,* Holt, Rinehart and Winston, 1983, pp. xiii.

5. Golab, S., *Tensor Calculus,* Elsevier Scientific Publishing Company, Third Edition, 1974.

6. Synge, J.L. Schild, A., *Tensor Calculus,* Dover Publishing Inc., 1978.

7. Young, E.C., *Vector and Tensor Analysis,* Marcel Dekker, Inc., Pure and Applied Mathematics Series, 1978.

8. Volterra, V., *Theory of Functionals and of Integral and Integro-Differential Equations,* Blackie & Son Limited, 1931.

9. Schetzen, M., *The Volterra and Wiener Theories of Nonlinear Systems,* John Wiley and Sons Inc., 1980.

10. Wiener, N., *Nonlinear Problems in Random Theory,* Technology Press of MIT and John Wiley & Sons, Inc., 1958.

11. Lee, Y.W., Schetzen, M., "Measurement of the Wiener Kernels of a Nonlinear System by Cross-Correlation," *Int. J. Control,* Vol. 1, pp 237-254, 1965.

12. Alper, P., "A Consideration of the Discrete Volterra Series," *IEEE Trans. on Automatic Control,* Vol. AC-10, July 1965, pp 322-327.

13. Sandor, J., Williamson. D.. "Identification and analysis of Non-linear Systems by Tensor Techniques." *Int. J. Control.* Vol. 27, No. 6. 1978. pp 853-878.

14. Parker, S.R., Thomas. J.J., *The Modeling and Identification of Discrete Nonlinear Moving Average Systems by Means of Tensor Convolutions,* The Third Internatinal Symposium on Forecasting, Philadelphia, PA, June 1983.

15. Parker, S.R., Perry, F.A., "A Discrete ARMA Model for Nonlinear System Identification", *IEEE Trans. on CAS,* Vol. CAS-28, No. 3, March 1981.

16. Parker, S.R., Mayoral, L.M., Thomas, J.J., *An Adaptive Kalman Identifier and Its Application to Linear and Nonlinear ARMA Modeling,* Proc. of the 16th Annual Conf. on Inform. Sciences and Systems, March 17-19 1982.

17. Zarzycki. J., Dewilde, P., *Nonlinear Least-Squares Prediction of Higher-Order Random Sequences,* Technical Univ. of Wroclaw Report No. I28/PRE/020/83. 1983.

18. Zarzycki. J., *Fast Algorithms For Least-Squares Nonlinear Prediction.* Technical Univ. of Wroclaw Report No.I28/P-029/83, June 1984.

19. Zarzycki. J.. *Adaptive Properties of Nonlinear Ladder-Filters.* Technical Univ of Wroclaw Report No. I28/P- 013/84. June 1984.

20. Zarzycki, J.. *Generalized Ladder-Filters For Nonlinear Prediction of Higher Order Random Sequences,* Technical Univ. of Wroclaw Report No. I28/P-028/83. June 1984.

21. Zarzycki. J.. *Nonlinear Prediction of Higher Order Random Sequences: Geometric Approach.* Technical Univ. of Wroclaw Report No. I28/P-012/84. June 1984.

22. Rugh. W.J.. *Nonlinear Systems Theory.* The Johns Hopkins University Press. Baltimore. Maryland. 1981.

23. Parker. S.R.. Lenk. P.J.. *Discrete Time Tensor Formulations For the Modeling of Nonlinear Systems.* IEEE Int. Symp. on CAS. Kyoto Japan. 1985.

24. Nering. E.D., *Linear Algebra and Matrix Theory.* Second Edition. John Wiley and Sons, Inc., pp. 7-8. 1970.

25. Chen C-T., *Linear System Theory and Design,* Holt, Rinehart and Winston, pp. 9-10, 1984.

26. Schetzen, M., "Nonlinear System Modeling Based on the Wiener Theory", *Proceedings of the IEEE,* Vol. 69, No. 12, December 1981.

27. Golub, G.H., Van Loan, C.F., *Matrix Computations,* Johns Hopkins University Press, 1983.

28. McCool. J.M., Widrow, B., *Principles and Applications of Adaptive Filters: A Tutorial Review,* Naval Undersea Centre. NUC TP 530, March 1977.

29. Widrow, B., *Adaptive Filters from Aspects of Network and Systems Theory.* Edited by R.E. Kalman and N. DeClaris, Holt, Rinehart and Winston. 1970.

30. Graham, A.. *Kronecker Products and Matrix Calculus: with Applications,* Ellis Horwood Ltd.. Chichester, West Sussex, England, p. 21, 1981.

31. Brewer, J.W., "Kronecker Products and Matrix Calculus in Systems Theory". *IEEE Transactions on CAS,* Vol. CAS-25. No. 9, p. 772. September 1978.

32. Goodwin. G.C.. Payne, R.L.. *Dynamic System Identification: Experimental Design and Data Analysis.* Academic Press, Vol. 136. Mathematics in Science and Engineering. 1977.

33. Melsa. J.L.. Cohn. D.L.. *Decision and Estimation Theory.* McGraw-Hill Book Company. 1978.

34. Strang. G.. *Linear Algebra.* 2d Ed.. Prentice-Hall Inc.. Englewood Cliffs. NJ. 1983.

35. Markel. J.D.. Gray. A.H.. *Linear Prediction of Speech.* Springer-Verlag. Berlin. 1976.

36. Itakura, F., Saito. S., *Digital Filtering Techniques for Speech Analysis and Synthesis,* 7th Int. Congress Acoustics. Budapest. Hungary, pp. 261-264, 1971.

37. Makhoul, J., "A Class of All-Zero Lattice Digital Filters: Properties and Applications," *IEEE Trans. on ASSP,* Vol. ASSP-26, No. 4, 1978 August.

38. Honig, M.L., Messerschmitt, D.G., *Adaptive Filters: Structures, Algorithms, and Applications,* Kluwer Academic Press, 1984.

39. Ziemer, R.E., Tranter, W.H., *Principles of Communications: Systems, Modulation, and Noise,* Houghton Mifflin Company, Boston, MA, 1976.

40. Levinson, N., "The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction," *J. Math. Phys.,* Vol. 25. No. 4. pp. 261-278. Jan 1947.

41. Yule, G.U.. "On a Method of Investigating Periodicities in Disturbed Series, With Special Reference to Wolfer's Sunspot Numbers." *Phil. Trans. Roy. Soc.*. Vol. 226-A. 1927, pp. 267-298.

42. Durbin. J.. "The Fitting of Time Series Models," *Rev. Inst. Int. Statist.,* Vol. 28. No. 3. pp 233-243. 1960.

43. Marzetta. T.M.. *A Linear Prediction Approach to Two-Dimensional Spectral Factorization and Spectral Estimation,* Ph.D. Dissertation, Massachusetts Inst. Technol., Cambridge. MA, 1978.

44. Marzetta, T.M.. "Two-dimensional Linear Prediction: Autocorrelation Arrays. Minimum Phase Filters, and Reflection Coefficient Arrays". *IEEE Trans. Acoust.. Speech. Sig.Proc.*, Vol. ASSP-28. Dec. 1980. pp. 725-733.

45. Parker. S.R.. Kayran. A.H.. "Lattice Parameter Autoregressive Modeling of Two-Dimensional Fields - Part 1: The Quarter-Plane Case." *IEEE Transactions on ASSP.* Vol. ASSP-32. No. 4. August 1984.

46. Shawcross. C.B.A.. *The Use of Two Dimensional Lattice Models in Isolated Word Recognition.* Engineer's Thesis. Naval Postgraduate School. Monterey. CA. December 1983.

47. Naylor, A.W., Sell, G.R., *Linear Operator Theory in Engineering and Science*, Holt Rinehart and Winston, Inc., 1971.

48. Kailath, T., "A General Likelihood-Ratio Formula for Random Signals in Gaussian Noise," *IEEE Trans. on Info. Theory*, Vol. IT-15, No. 3, May 1969.

49. Kung, H.T., Leiserson, C.E., *Systolic Arrays (for VLSI)*, Proc., Sparse Matrix Symp (SIAM), pp. 256-282, April 1978.

50. Kung, S.Y., "On Supercomputing with Systolic/Wavefront Array Processors," *Proc. of the IEEE*, Vol. 72, No. 7, July 1984.

51. Moldovan, "On the Design of Algorithms for VLSI Systolic Arrays," *Proc. of the IEEE*. Vol. 71, No. 1, Jan 1983.

52. Franks, L.E., *Signal Theory*, Prentice-Hall, Inc., Englewood Cliffs, NJ. 1968.

APPENDIX A: ALTERNATE PROOF OF THEOREM 4.2

In this appendix an alternate proof of the generalized error order update equations (4.40) is provided. The proof presented here relies on geometric arguments which are possible if an abstract mathematical framework is adopted. In this appendix, a random process will be considered to consist of a time-series of random variables. Each of these random variables is thought of as a vector in an infinite dimensional Hilbert space. For a rigorous discussion of these concepts see for example [Ref. 51,52].

A. DEFINITIONS AND FORMULATION

A discrete random process $\mathbf{Y} = \left\{ y(i): 0 < i < K \right\}$ can be considered to span a K+1 dimensional subspace $\mathbf{S}^{k+1}$ of an infinite dimensional Hilbert space **S** (assuming completeness and the linear independence of the y(i)). The inner product on this space is defined to be

$$\langle u,v \rangle = E\left\{ uv \right\} \tag{A.1}$$

where u and v are elements of **S**. This inner product induces a norm

$$\| u \| = \langle u,u \rangle^{1/2} \tag{A.2}$$

The error, e_{k-1}^{N}, in predicting the element y(k+1) from the previous N elements of **Y** is given by

$$e_{k-1}^{N} = \sum_{\lambda=0}^{K} h_{\lambda}^{N}(k-1) y(\lambda) \tag{A.3}$$

where

$$h_{\lambda}^{N}(k-1) = 0 \cdots 0 \quad h_{k}^{N}{}_{N-1} \quad h_{k}^{N}{}_{N-2} \quad \cdots \quad h_{k}^{N}{}_{1} \; 0 \cdots 0 \tag{A.4}$$

A normalized version of the forward error is given by

$$e_{k-1}^{N} = \frac{e_{k-1}^{N}}{e_{k+1}^{N}} \tag{A.5a}$$

$$= \sum_{\lambda=0}^{K} a_\lambda^N(k-1)y(\lambda) \tag{A.5b}$$

where

$$a_\lambda^N(k+1) = \frac{1}{\| e_{k+1}^N \|} h_\lambda^N(k+1) \tag{A.6}$$

The backwards prediction error, r_{k-N}^N, is the error associated with the prediction of y(i) given the next N elements of **Y**. It is given by,

$$r_{k-N}^N = \sum_{\lambda=0}^{K} \tilde{h}_\lambda^N(k-N)y(\lambda) \tag{A.7}$$

where

$$\tilde{h}_\lambda^N(k-N) = [0 \cdots 0 \ 1 \ -\tilde{h}_{k-N+1}^N \ -\tilde{h}_{k-N+2}^N \cdots -\tilde{h}_k^N \ 0 \cdots 0] \tag{A.8}$$

A normalized version of r_{k-n}^N can be defined as

$$\bar{r}_{k-N}^N = \frac{r_{k-N}^N}{\| r_{k-N}^N \|} \tag{A.9a}$$

$$= \sum_{\lambda=0}^{K} b_\lambda^N(k-N)y(\lambda) \tag{A.9b}$$

where

$$b_\lambda^N(k-N) = \frac{\tilde{h}_\lambda^N(k-N)}{\| r_{k-N}^N \|} \tag{A.10}$$

By **orthogonal** we mean that given two elements of the space **S**, u and v, say, then

$$\langle u,v \rangle = \begin{cases} 0 & \text{for } u \neq v \\ \| u \|^2 & \text{for } u = v \end{cases} \tag{A.11}$$

This will be indicated symbolically as

$$u \perp v \tag{A.12}$$

The expression

$$u \perp S^M \tag{A.13}$$

implies that u is orthogonal to all elements of the subspace S^M.

The symbol '$\oplus$' is used to indicate direct sum. For example

$$S^3 = S^1 \oplus S^2 \tag{A.14}$$

means that the space spanned by S^3 is the the space spanned by the Cartesian product of the underlying sets of the spaces S^1 and S^2 [Ref. 47: p. 196].

Finally, the symbol 'v ' will be used to mean **the span of.**

These definitions are sufficient to state and prove the generalized error order update theorem.

B. ERROR ORDER UPDATE RECURSIONS

1. Theorem 4.2

The (N+1) order errors can be calculated from the N- th order errors through the recursion

$$\begin{bmatrix} \bar{e}_{k+1}^{N+1} \\ \bar{r}_{k-N}^{N+1} \end{bmatrix} = \frac{1}{\sqrt{1-(\rho_{N+1}^{k})^2}} \begin{bmatrix} 1 & -\rho_{N+1}^{k} \\ -\rho_{N+1}^{k} & 1 \end{bmatrix} \begin{bmatrix} \bar{e}_{k+1}^{N} \\ \bar{r}_{k-N}^{N} \end{bmatrix} \tag{A.15}$$

where

$$\rho_{N+1}^{k} = \langle \bar{e}_{k+1}^{N}, \bar{r}_{k-N}^{N} \rangle \tag{A.16}$$

is called the **partial correlation coefficient** or **reflection factor** [Ref. 17: p. 37].

2. Proof

We have that

$$r_k^N \perp S = v\{y(k-N+1)\ y(k-N+2) \cdots y(k)\} \tag{A.17}$$

but

$$\bar{r}_{k-N}^{N} \perp S_k^{N+1} = v\{y(k-N)\ y(k-N+1) \cdots y(k)\} \tag{A.18}$$

Therefore,

$$S_k^{N+1} = v\left\{\bar{r}_{k-N}^N \;\; y(k-N-1) \;\cdots\; y(k)\right\} \tag{A.19}$$

which can be written as

$$S_k^{N+1} = S_k^N \oplus v\left\{\bar{r}_{k-N}^N\right\} \tag{A.20}$$

For the updated forward error, e_{k+1}^{N+1}, the following is true

$$e_{k+1}^{N+1} \perp S_k^N \perp v\left\{\bar{r}_{k-N}^N\right\} \tag{A.21}$$

Since $e_{k+1}^N \perp S_k^N$ we need only orthogonalize it (by a Gram-Schmidt procedure) with respect to $v\left\{\bar{r}_{k-N}^N\right\}$ to obtain e_{k+1}^{N+1}. Thus.

$$e_{k+1}^{N+1} = e_{k+1}^N - C\bar{r}_{k-N}^N \tag{A.22}$$

where

$$C = \frac{\langle e_{k+1}^N, \bar{r}_{k-N}^N \rangle}{\bar{r}_{k-N}^N} \tag{A.23a}$$

$$= \rho_{N+1}^k \quad e_{k-1}^N \tag{A.23b}$$

Therefore.

$$\bar{e}_{k+1}^{N+1} = \frac{e_{k+1}^N}{e_{k+1}^{N-1}} \bar{e}_{k-1}^N - \rho_{N+1}^k \bar{r}_{k-N}^N \tag{A.24}$$

Proof of the fact that

$$\frac{e_{k-1}^N}{e_{k-1}^{N-1}} = \frac{1}{\sqrt{1 - (\rho_{N-1}^k)^2}} \tag{A.25}$$

will not be repeated here (see Chapter 4. Section B).

Similar arguments can be used to verify the backward error order update equation given in (A.15).

```
C*********************************************************************
C
C    2DLAT MAIN PROGRAM
C
C    PROGRAM TO TEST 2D LATTICE SUBROUTINES
C
C    WRITTEN 29 APRIL 1985
C
C*********************************************************************
C
     REAL*8 ALPHA(26,26),BETA(26,26),A(26,26),B(26,26),RHO(26,26)
     REAL*8 HZ(25),Y(256,256),R(26,26)
     REAL*4 GAIN(40,40)
C
C    DEFINE THE AR FILTER COEFFICIENTS
C
C    DATA HZ/1.0,-.23,.12,.111,21*0.0/
C    DATA HZ/1.0,-.470,.007,.295,0.0,.015,.055,.003,.022,16*0.0/
     DATA HZ 1.0,.03,-.015,-.011,.033,-.47,.195,.055,0.0,-.085,0.0,0.0,
    *.003,.022,-.0001,.0067,0.0,0.0,0.0,0.0,0.0,0.0,-.002,0.0,.0001/
C
C    DEFINE OTHER PARAMETERS
C
C    MODEL SIZE
C
     N = 5
     MN = N*N
C
C    ACTUAL SYSTEM SIZE
C
     NA = 5
     MNA = NA*NA
C
C    NUMBER OF POINTS IN THE PLOT IS IP X IP.
C
     IP = 40
C
C    NUMBER OF DATA POINTS TO ESTIMATE CORRELATIONS IS IYS X IYS.
C
     IYS = 128
C
C    CALL APPROPRIATE SUBROUTINES
C
     CALL TXFCN(HZ,NA,GAIN,IP,0)
     CALL PLOT3(GAIN,IP,IP,'ORIGINAL SPECTRUM$')
     CALL GENRAT(HZ,NA,Y,IYS)
     CALL CORLAT(Y,IYS,R,N)
     CALL SCHUR(RHO,R,ALPHA,BETA,MN)
     CALL LEVSON(A,B,RHO,R,MN)
```

```
      DO 10 I = 1, MN
        HZ(I) = A(1.I)/A(I,I)
10    CONTINUE
      CALL TXFCN(HZ,N,GAIN,IP,1)
      CALL PLOT3(GAIN,IP,IP,'4-TH ORDER LATTICE APPROXIMATIONS')
      STOP
      END
```

```
C*******************************************************************
C
C    SUBROUTINE GENRAT
C
C    THIS SUBROUTINE GENERATES A 2D DATA FIELD FROM THE AR
C    FILTER COEFFICIENTS.  A WHITE NOISE INPUT AND WHITE NOISE
C    INITIAL CONDITIONS ARE USED.
C
C    WRITTEN 29 APRIL 1985
C
C*******************************************************************
C
     SUBROUTINE GENRAT(HZ,N,Y,IYS)
     REAL*8 HZ(25),Y(256,256),ADD
C
C    FETCH THE RANDOM NUMBER GENERATOR SEED
C    UNDER FORTHX, STORE SEED IN FILE 13.
C    UNDER DISSPLA ENTER SEED EXPLICITLY.
C
C     READ(13,10) IY
C10   FORMAT(I10)
C     REWIND 13
     IY = 817346
C
C    GENERATE THE RANDOM FIELD
C
     MN = N*N
     MNM1 = MN-1
     DO 40 I = 1,IYS
       DO 30 J = 1,IYS
         Y(I,J) = URAND(IY) - .5
         KI = 0
         KJ = 0
         DO 20 K = 1,MNM1
           IF (MOD(K,N).NE.0) GOTO 15
           KI = KI + 1
           KJ = 0
           GO TO 16
15          KJ = KJ + 1
16         IF (((I-KI).LE.0) OR.((J-KJ).LE.0)) GOTO 17
           ADD = Y(I-KI,J-KJ)
           GO TO 18
17          ADD = URAND(IY) - .5
18         Y(I,J) = Y(I,J) - HZ(K+1)*ADD
20       CONTINUE
30     CONTINUE
40   CONTINUE
C
C    STORE THE RANDOM NUMBER SEED
C
C     WRITE(13,50) IY
C50   FORMAT(I10)
C
```

```
C
      RETURN
      END
```

```
C*****************************************************************
C
C     SUBROUTINE CORLAT
C
C     THIS SUBROUTINE PRODUCES A CORRELATION MATRIX FROM
C     A 2-D DATA FIELD IN AN ORDER
C     WHICH IS COMPATIBLE WITH SUBROUTINE SCHUR
C
C     WRITTEN 30 APRIL 1985
C
C*****************************************************************
C
      SUBROUTINE CORLAT (Y,IYS,R,N)
      REAL*8 Y(256,256),R(26,26),SUM
C
C     DEFINE CONSTANTS
C
      MN = N*N
      IR = 0
C
C     BEGIN OUTER LOOP
C
      DO 100 MP1 = 1,N
        M0 = MP1 - 1
        LLIM = 2*MP1 - 1
        DO 90 L = 1,LLIM
          L0 = L - 1
          I1 = M0
          J1 = L0/2
          IF (MOD(L0,2).EQ.0) GO TO 10
          I1 = J1
          J1 = M0
10        JR = IR
          IR = IR + 1
          KBOT = L
          DO 80 NP1 = MP1,N
            N0 = NP1 - 1
            KLIM = 2*NP1 - 1
            DO 70 K = KBOT,KLIM
              K0 = K-1
              I = N0
              J = K0/2
              IF (MOD(K0,2).EQ.0) GO TO 20
              I = J
              J = N0
20            IOFF = I - I1
              JOFF = J - J1
              K1MAX = IYS
              K1MIN = IOFF + 1
              IF (IOFF.GT.0) GO TO 30
              K1MAX = IYS + IOFF
              K1MIN = 1
30            K2MAX = IYS
```

```
            K2MIN = JOFF + 1
            IF (JOFF.GT.0) GO TO 40
            K2MAX = IYS - JOFF
            K2MIN = 1
40           SUM = 0.0
            JR = JR + 1
C             WRITE(6,44)IR,JR,I1,J1,I,J,IOFF,JOFF
C44           FORMAT(8(2X,I4))
            DO 60 K1 = K1MIN,K1MAX
              DO 50 K2 = K2MIN,K2MAX
                SUM = SUM + Y(K1,K2)*Y(K1-IOFF,K2-JOFF)
50             CONTINUE
60           CONTINUE
            R(IR,JR) = SUM/(K1MAX-K1MIN+1)/(K2MAX-K2MIN+1)
70         CONTINUE
          KBOT = 1
80       CONTINUE
90     CONTINUE
100  CONTINUE
C
C    FILL IN THE SYMMETRIC HALF OF CORRELATION MATRIX
C
     DO 120 I = 2,MN
       IM1 = I - 1
       DO 110 J = 1,IM1
         R(I,J) = R(J,I)
110    CONTINUE
120  CONTINUE
C
C
     RETURN
     END
```

```
C*****************************************************************
C
C    SUBROUTINE TXFCN
C
C    GENERATES A 2-D FREQUENCY RESPONSE GIVEN THE TRANSFER
C    FUNCTION COEFFICIENTS
C
C    WRITTEN BY DR. B. MADAN
C    MODIFIED BY P.J. LENK  29 APRIL 1985
C
C    PARAMETER K0 INDICATES THE ORDERING OF THE COEFFICIENTS
C          - K0 = 0: PARAMETERS IN ROW-MAJOR ORDER
C          - K0 = 1: PARAMETERS IN TWIDDLED ORDER
C
C*****************************************************************
C
      SUBROUTINE TXFCN(HZ,N,GAIN,IP,K0)
      REAL*8 HZ(25),CONVRT(5,5)
      REAL*4 GAIN(40,40)
      COMPLEX CSUM
      COMPLEX*8 ARG1,ARG2
C
C     DEFINE CONSTANTS
C
      PI = 3.14159
C
C     DETERMINE ORDER AND REORDER IF NECESSARY THE COEFFICIENTS
C
      IF (K0.EQ.0) GO TO 60
      JR = 0
      DO 30 MP1 = 1,N
        M0 = MP1 - 1
        LLIM = 2*MP1 - 1
        DO 20 L = 1,LLIM
          L0 = L - 1
          I = M0
          J = L0 - 2
          IF (MOD(L0,2).EQ.0) GO TO 10
          I = J
          J = M0
10        JR = JR + 1
          CONVRT(I+1,J+1) = HZ(JR)
20      CONTINUE
30    CONTINUE
C
C     TRANSFER THE COEFFICIENTS BACK TO HZ
C
      JR = 0
      DO 50 I = 1,N
        DO 40 J = 1,N
          JR = JR + 1
          HZ(JR) = CONVRT(I,J)
          WRITE(6,35)HZ(JR)
```

```
35        FORMAT(2X,E12.5)
40      CONTINUE
50    CONTINUE
C
C     PROCEED WITH TRANSFER FUNCTION EVALUATION
C
60    DW = 2.0*PI/FLOAT(IP-1)
      DO 100 IW1 = 1,IP
        W1 = DW * (IW1 - 1)- PI
        DO 90 JW1 = 1,IP
          A1 = W1
          W2 = DW * (JW1 - 1)- PI
          IZ = 0
          CSUM = CMPLX(0.0,0.0)
          DO 80 I = 1,N
            ARG1 = CMPLX(0.0,-A1)
            A1 = A1 + W1
            A2 = W2
            DO 70 J = 1,N
              IZ = IZ + 1
              ARG2 = CMPLX(0.0,-A2)
              A2 = A2 + W2
C              WRITE(6,77)I,J,ARG1,ARG2,IZ,HZ(IZ),CSUM
C77              FORMAT(2(2X,I3),4(2X,E12.5),2X,I3,3(2X,E12.5))
              CSUM = CSUM + CMPLX(SNGL(HZ(IZ)),0.0)*CEXP(ARG1+ARG2)
70          CONTINUE
80        CONTINUE
        GAIN(IW1,JW1) = 1.0/CABS(CSUM)
        GAIN(IW1,JW1) = GAIN(IW1,JW1)*GAIN(IW1,JW1)
C       WRITE(6,78)IW1,JW1,CSUM,GAIN(IW1,JW1)
C78     FORMAT(2(2X,I3),3(2X,E12.5))
90    CONTINUE
C       WRITE(6,18)IW1,(GAIN(IW1,I),I=1,IP)
C18     FORMAT(1X,I3,5(2X,E12.5))
100   CONTINUE
      RETURN
      END
```

```
C*********************************************************************
C
C    SUBROUTINE PLOT3
C
C    THIS IS A ROUTINE TO USE THE DISSPLA PACKAGE TO
C    DRAW A THREE DIMENSIONAL PLOT OF A 2-D FILTER'S
C    FREQUENCY RESPONSE.
C
C    WRITTEN BY DR. B. MADAN
C    MODIFIED BY P.J. LENK  29 APRIL 1985
C
C*********************************************************************
C
     SUBROUTINE PLOT3(A,IM,JN,LABEL)
     DIMENSION A(IM,JN)
     INTEGER LABEL(100)
     CALL TEK618
C
C    INITIALIZE THE PLOTTING DEVICE
C
     WRITE(6,51)
51   FORMAT(' 1 CALL TEK618 OK')
C         CALL THE DEVICE
     CALL RESET('ALL')
     WRITE(6,52)
52   FORMAT(' 2 RESET ALL OK')
C         CALL SETUP FOR CUBE
     A1=FLOAT(IM)
     A2=FLOAT(JN)
     CALL PAGE(11.,9.5)
     WRITE(6,53)
53   FORMAT(' 3 CALL PAGE OK')
C    CALL PAGE(A1,A2)
C    CALL PHYSOR(0.,0.)
     CALL AREA2D(7.0,7.0)
     WRITE(6,54)
54   FORMAT(' 4 CALL AREA 2-D OK')
     CALL SIMPLX
     CALL HEIGHT(.2)
     CALL HEADIN(LABEL,100,4,1)
     CALL HEIGHT(0.14)
     CALL VIEW(-10.0,-5.0,15.0)
     CALL VOLM3D(12.0,12.0,12.0)
C         CALL X AXIS LABELLING ROUTINE
     CALL X3NAME(' W2 $',3)
C         CALL Y AXIS LABELLING ROUTINE
     CALL Y3NAME(' W1 $',3)
C         CALL Z AXIS LABELLING ROUTINE
     CALL Z3NAME(',2)
C         CALL THE SURFACE PLOT ROUTINE
C
     CALL GRAF3D(-1.0,0.2,1.0,-1.0,0.2,1.0,0.0,0.2,0.16.)
1500 CALL SURMAT(A,1,IM,1,JN,0)
```

```
CALL ENDPL (1)
CALL DONEPL
RETURN
END
```

```
C*********************************************************************
C
C     SUBROUTINE SCHUR
C
C     CALCULATES THE REFLECTION FACTORS FROM THE CORRELATION MATRIX
C
C     WRITTEN 29 APRIL 1985
C
C*********************************************************************
C
      SUBROUTINE SCHUR(RHO,R,ALPHA,BETA,N)
      REAL*8 RHO(26,26),R(26,26),ALPHA(26,26),BETA(26,26),RNORM,T
C
C     INITIALIZE THE ALPHA AND BETA ARRAYS
C
      DO 10 I = 1,N
        DO 5 J = 1,N
          ALPHA(I,J) = R(I,J)/DSQRT(R(I,I))
          BETA(I,J) = ALPHA(I,J)
          RHO(I,J) = 0.0
5       CONTINUE
C         WRITE(16,7)(ALPHA(I,J),J=1,N)
C7        FORMAT(5(2X,E12.5))
10    CONTINUE
C
C     BEGIN CALCULATING THE REFLECTION FACTORS
C
      DO 50 J = 2,N
        NJ1 = N - J + 1
        DO 40 I = 1,NJ1
          JI1 = J - I - 1
          IP1 = I + 1
          RHO(I,JI1) = ALPHA(I,JI1)/BETA(IP1,JI1)
          RNORM = DSQRT(1.0 - RHO(I,JI1)*RHO(I,JI1))
          DO 30 K = 1,N
            T = ALPHA(I,K)
            ALPHA(I,K) = (ALPHA(I,K)-RHO(I,JI1)*BETA(IP1,K))/RNORM
            BETA(I,K) = (BETA(IP1,K)-RHO(I,JI1)*T)/RNORM
30        CONTINUE
40      CONTINUE
C         WRITE(16,42)J,((ALPHA(I,K),K=1,N),I=1,N)
C42       FORMAT( 2X,I3,4(2X,E12.5))
C         WRITE(16,42)J,((BETA(I,K),K=1,N),I=1,N)
50    CONTINUE
C
C
      RETURN
      END
```

```
C*****************************************************************
C
C     SUBROUTINE LEVINSON
C
C     GENERATES THE AUTOREGRESSIVE MODEL COEFFICIENTS GIVEN THE
C     REFLECTION FACTORS
C
C     WRITTEN 29 APRIL 1985
C
C*****************************************************************
C
      SUBROUTINE LEVSON (A,B,RHO,R,N)
      REAL*8 A(26,26),B(26,26),RHO(26,26),R(26,26),RNORM,T
C
C     INITIALIZE THE A AND B MATRICES
C
      DO 20 I = 1,N
        DO 10 J = 1,N
          A(I,J) = 0.0
          B(I,J) = 0.0
10      CONTINUE
        A(I,I) = 1.0/DSQRT(R(I,I))
        B(I,I) = A(I,I)
20    CONTINUE
C
C     CALCULATE THE AR PARAMETERS USING LEVINSON'S ALGORITHM
C
      DO 60 J = 2,N
        NJ1 = N - J + 1
        DO 50 I = 1,NJ1
          IJ1 = I - J - 1
          RNORM = DSQRT(1.0 - RHO(I,IJ1)*RHO(I,IJ1))
          DO 40 K = I,IJ1
            T = A(I,K)
            A(I,K) = (A(I,K) - RHO(I,IJ1)*B(I+1,K))/RNORM
            B(I,K) = (B(I-1,K) - RHO(I,IJ1)*T)/RNORM
40        CONTINUE
50      CONTINUE
C       WRITE(16,77)J,((A(I,K),K=1,N),I=1,N)
C       WRITE(16,77)J,((B(I,K),K=1,N),I=1,N)
C77     FORMAT(,2X,I3,4(2X,E12.5))
60    CONTINUE
      DO 66 I = 1,N
        RNORM = A(1,I)*A(1,I)
C       WRITE(6,65)RNORM
C       WRITE(17,65)RNORM
C65     FORMAT(2X,E12.5)
66    CONTINUE
C
C
      RETURN
      END
```

```
C*********************************************************************
C
C    FUNCTION URAND
C
C    TAKEN FROM "COMPUTER METHODS FOR MATHEMATICAL COMPUTATIONS" BY
C    G.E. FORSYTHE, M.A. MALCOLM, AND C.B. MOLER
C    PRENTICE-HALL, ENGLEWOOD CLIFFS, NJ., 1977, P. 246.
C
C*********************************************************************
C
     REAL FUNCTION URAND(IY)
     INTEGER IA,IC,ITWO,M2,M,MIC
C      URAND IS A UNIFORM RANDOM NUMBER GENERATOR BASED ON THEORY AND
C    SUGGESTIONS GIVEN IN D.E. KNUTH (1969), VOL. 2.  THE INTEGER IY
C    SHOULD BE INITIALIZED TO AN ARBITRARY INTEGER PRIOR TO THE FIRST
C    CALL TO URAND.  THE CALLING PROGRAM SHOULD NOT ALTER THE VALUE OF IY
C    BETWEEN SUBSEQUENT CALLS TO URAND.  VALUES OF URAND WILL BE RETURNED
C    IN THE INTERVAL (0,1).
C
     DOUBLE PRECISION HALFM
     REAL S
     DOUBLE PRECISION DATAN,DSQRT
     DATA M2/0/,ITWO/2/
     IF(M2.NE.0) GO TO 20
C
C    IF FIRST ENTRY, COMPUTE MACHINE INTEGER WORD LENGTH
C
     M=1
   10 M2=M
     M=ITWO*M2
     IF(M.GT.M2) GO TO 10
     HALFM=M2
C
C    COMPUTE MULTIPLIER AND INCREMENT FOR LINEAR CONGRUENTIAL METHOD
C
     IA=8*IDINT(HALFM*DATAN(1.D0)/8.D0)+5
     IC=2*IDINT(HALFM*(.5D0-DSQRT(3.D0)/6.D0))+1
     MIC=(M2-IC)+M2
C
C    S IS THE SCALE FACTOR FOR CONVERTING TO FLOATING POINT
C
     S=.5/HALFM
C
C    COMPUTE NEXT RANDOM NUMBER
C
   20 IY=IY*IA
C
C    THE FOLLOWING STATEMENT IS FOR COMPUTERS WHICH DO NOT ALLOW
C    INTEGER OVERFLOW ON ADDITION
C
     IF(IY.GT.MIC) IY=(IY-M2)-M2
C
     IY=IY+IC
```

```
C
C    THE FOLLOWING IS FOR COMPUTERS FOR WHICH THE WORD LENGTH
C    FOR ADDITION IS GREATER THAN FOR MULTIPLICATION
C
     IF(IY/2.GT.M2)IY=(IY-M2)-M2
C
C    THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE INTEGER OVERFLOW
C    AFFECTS THE SIGN BIT
C
     IF(IY.LT.0)IY=(IY+M2)+M2
     URAND=FLOAT(IY)*S
     RETURN
     END
```

```
C*********************************************************************
C
C    PROGRAM NLMAIN
C
C    THIS IS THE PROGRAM TO TEST THE NONLINEAR LATTICE MODEL
C
C    WRITTEN 7 MAY 1985
C
C*********************************************************************
C
     REAL*8 ALPHA(26,26),BETA(26,26),A(26,26),B(26,26),RHO(26,26)
     REAL*8 HZ(5,5),Y(10000),R(26,26)
C
C    DEFINE SYSTEM PARAMETERS
C
     DATA HZ/.2,.02,-.1,0.0,0.0,.05,0.09,.22,0.0,0.0,-.03,.001,0.02,0.0
    *,0.0,10*0.0/
C
C    DEFINE SYSTEM CONSTANTS
C
     DO 5 I = 1,5
       K = 6 - I
       WRITE(6,4)(HZ(K,J),J=1,5)
4      FORMAT(5(2X,E12.5))
5    CONTINUE
     IYS = 1000
     NA = 3
     MNA = NA * NA
     MNAP1 = MNA + 1
C
C    DEFINE MODEL CONSTANTS
C
     N = 3
     MN = N * N
     MNP1 = MN + 1
C
C    CALL SUBROUTINES
C
     CALL NLGEN(Y,N,HZ,IYS)
     CALL NLCLAT(Y,IYS,R,N)
     DO 30 I = 1,MNP1
       WRITE(16,20)(R(I,J),J=1,MNP1)
20      FORMAT(5(2X,E12.5))
30   CONTINUE
     CALL SCHUR(RHO,R,ALPHA,BETA,MNP1)
     DO 60 I = 1,MNP1
       WRITE(16,20)(RHO(I,J),J=1,MNP1)
60   CONTINUE
     CALL LEVSON(A,B,RHO,R,MNP1)
     DO 70 I = 1,MNP1
       WRITE(16,20)(A(I,J),J=1,MNP1)
70   CONTINUE
C
```

```
C
      STOP
      END
```

```
C*********************************************************************
C
C    SUBROUTINE NLGEN
C
C    THIS SUBROUTINE GENERATES AN OUTPUT SEQUENCE FROM THE SYSTEM
C    DESCRIBED BY THE MODEL PARAMETERS CONTAINED IN H(,).  IT
C    USES WHITE NOISE UNIFORM ON (-.5,.5) TO EXCITE THE SYSTEM.
C    THE INITIAL CONDITIONS ARE ALSO DRAWN FROM THIS DISTRIBUTION.
C
C    WRITTEN MAY 7 1985
C
C*********************************************************************
C
     SUBROUTINE NLGEN(Y,N,H,IYS)
     REAL*8 Y(10000),H(5,5),DSEED
C
C    FETCH THE RANDOM SEED
C
     READ(13,10) IY
10   FORMAT(I10)
     REWIND 13
C    DSEED = DFLOAT(IY)
C
C    SET UP THE INITIAL CONDITIONS
C
     Y(1) = 2.*(URAND(IY) - .5)
     Y(2) = 2.*(URAND(IY) - .5)
C    CALL GGNML(DSEED,IYS,Y)
C
C    CALCULATE THE REMAINING VALUES OF THE SEQUENCE
C
     DO 40 I = 3,IYS
C       Y(I) = 2.*(URAND(IY) - .5)
C       Y(I) = .2*Y(I)
        Y(I) = URAND(IY)
C       DO 30 J = 1,N
C          JM1 = J - 1
C          DO 20 K = 1,N
C             KM1 = K - 1
C             Y(I) = Y(I) - H(J,K)*COORD(Y(I-1),JM1)*COORD(Y(I-2),KM1)
C20        CONTINUE
C30     CONTINUE
40   CONTINUE
C
C    FINISH
C
C     IY = DINT(DSEED)
     WRITE(13,50)IY
50   FORMAT(I10)
     REWIND 13
     RETURN
     END
```

```
C*********************************************************************
C
C     SUBROUTINE NLCLAT
C
C     THIS SUBROUTINE PRODUCES A CORRELATION MATRIX FROM NONLINEAR
C     TIME SEQUENCE IN AN ORDER WHICH IS COMPATIBLE WITH SUBROUTINE
C     SCHUR.
C
C     WRITTEN 7 MAY 1985
C
C*********************************************************************
C
      SUBROUTINE NLCLAT (Y,IYS,R,N)
      REAL*8 Y(10000),R(26,26),SUM,VEC(26)
C
C     DEFINE CONSTANTS
C
      MN = N*N
      MNP1 = MN + 1
      IYSM2 = IYS - 2
      FIYSM2 = FLOAT(IYSM2)
C
C     INITIALIZE R MATRIX TO ZERO
C
      DO 20 I = 1,MNP1
        DO 10 J = 1,MNP1
          R(I,J) = 0.0
10      CONTINUE
20    CONTINUE
C
C     BEGIN OUTER LOOP
C
      DO 80 I = 3,IYS
        IR = 1
        VEC(IR) = Y(I)
        DO 50 MP1 = 1,N
          M0 = MP1 - 1
          LLIM = 2*MP1 - 1
          DO 40 L = 1,LLIM
            L0 = L - 1
            I1 = M0
            J1 = L0
            IF (MOD(L0,2).EQ.0) GO TO 30
            I1 = J1
            J1 = M0
30          IR = IR + 1
            VEC(IR) = COORD(Y(I-1),I1)*COORD(Y(I-2),J1)
40        CONTINUE
50      CONTINUE
C
C     CALCULATE THE CORRELATIONS
C
        DO 70 J = 1,MNP1
```

```
          DO 60 K = J,MNP1
            R(J,K) = R(J,K) + VEC(J)*VEC(K)
C             WRITE(6,12)VEC(J),VEC(K),R(J,K)
C12           FORMAT(3(2X,E12.5))
60        CONTINUE
70      CONTINUE
80    CONTINUE
C
C     DIVIDE BY THE NUMBER OF DATA ELEMENTS CONSIDERED
C
      DO 100 J = 1,MNP1
        DO 90 K = J,MNP1
          R(J,K) = R(J,K)/FIYSM2
90      CONTINUE
100   CONTINUE
C
C     FILL IN THE SYMMETRIC HALF OF CORRELATION MATRIX
C
      DO 120 I = 2,MNP1
        IM1 = I - 1
        DO 110 J = 1,IM1
          R(I,J) = R(J,I)
110     CONTINUE
120   CONTINUE
C
C
      RETURN
      END
```

```
C**********************************************************************
C
C      NONLINEAR WIENER MODELLING PROGRAM
C      THIS USES FUNCTIONS CONTAINED IN ROUTINE COORD
C
C      MODIFIED 1 JAN 85, 14 JAN 85
C
C**********************************************************************
C
      DOUBLE PRECISION A(25,26),Z(25),X(15000),Y(15000),VAR,XA,YA,YHAT
      DOUBLE PRECISION ZZ(25),XM1,SEED,STORE(25)
C
C     INPUT SIMULATION PARAMETERS
C
      READ(13,77)IY
      REWIND 13
      WRITE(6,40)
40    FORMAT(2X,'MAGNITUDE OF NOISE')
      READ(5,41)VAR
41    FORMAT(F12.5)
      WRITE(6,42)
42    FORMAT(2X,'NUMBER OF POINTS')
      READ(5,43)N
43    FORMAT(I5)
      WRITE(6,44)
44    FORMAT(2X,'MAXIMUM POWER OF X ')
      READ(5,45)LW
45    FORMAT(I1)
      WRITE(12,46)N
      WRITE(6,46)N
46    FORMAT(2X,'THE NUMBER OF POINTS WAS ',I5)
      WRITE(12,47)LW
      WRITE(6,47)LW
47    FORMAT(2X,'THE MAXIMUM POWER OF X IS ',I1)
      WRITE(6,48)VAR,VAR
      WRITE(12,48)VAR,VAR
48    FORMAT(2X,'THE NOISE IS UNIFORM FROM -',F9.5,' TO +',F9.5)
      LW = LW - 1
      VAR = VAR*2.0
      DO 132 I = 1,25
        STORE(I) = 0.0
132   CONTINUE
      X(1) = VAR*(URAND(IY) - .5)
      Y(1) = 1.0
      DO 2 I=2,N
        X(I) = VAR*(URAND(IY) - .5)
        Y(I) = UNKNOW(X(I),X(I-1))
2     CONTINUE
      CALL NCRLAT(X,Y,A,LSQW,LW,N)
C       WRITE(6,134)
C       WRITE(12,134)
C134   FORMAT(/2X,'NO INVERSION SOLUTION')
      DO 142 I = 1, LSQW
```

```
      ZZ(J) = A(J,LSQW + 1)/A(J,J)
      STORE(J) = STORE(J) + ZZ(J)/25.0
142   CONTINUE
C     DO 133 J = 1,LW
C       IMIN = (J-1)*LW + 1
C       IMAX = J*LW
C       WRITE(6,173) (ZZ(I),I = IMIN,IMAX)
C       WRITE(12,173) (ZZ(I),I = IMIN,IMAX)
C173     FORMAT(5(2X,E12.5))
C133   CONTINUE
C     CALL SOLVE(A,Z,LSQW)
C     WRITE(6,135)
C     WRITE(12,135)
C135   FORMAT(/2X,'USING FULL MATRIX INVERSION')
C     DO 27 J = 1,LW
C       IMIN = (J-1)*LW + 1
C       IMAX = J*LW
C       WRITE(6,11) (Z(I),I = IMIN,IMAX)
C       WRITE(12,11) (Z(I),I = IMIN,IMAX)
11     FORMAT(5(2X,E12.5))
C27   CONTINUE
131   CONTINUE
      WRITE(6,201)
      WRITE(12,201)
201   FORMAT(/2X,'NO INVERSION SOLUTION AVERAGED OVER 25 RUNS')
      DO 200 J = 1,LW
        IMIN = (J - 1)*LW + 1
        IMAX = J * LW
        WRITE(6,11) (STORE(I),I = IMIN,IMAX)
        WRITE(12,11) (STORE(I),I = IMIN,IMAX)
200   CONTINUE
      WRITE(6,22)
      WRITE(12,22)
      XM1 = VAR*(URAND(IY) - .5)
      DO 73 I = 1,10
        XA = VAR*(URAND(IY) - .5)
        YA = UNKNOW(XA,XM1)
        YHAT = 0.0
        YYHAT = 0.0
        DO 14 J = 1,LW
          JM1 = J - 1
          X2 = COORD(XM1,JM1)
          DO 15 K = 1,LW
            KM1 = K - 1
            X1 = COORD(XA,KM1)
C             WRITE(6,52)X1,X2
C52           FORMAT(2(2X,E12.5))
            YHAT = YHAT + X1 * X2 * Z((J-1)*LW + K)
            YYHAT = YYHAT + X1 * X2 * ZZ((J-1)*LW + K)
15        CONTINUE
14      CONTINUE
        ERROR = (YA - YHAT)/YA
        ZERROR = (YA - YYHAT)/YA
```

```
      XM1 = XA
      WRITE(6,17) YA,YHAT,YYHAT,ERROR.ZERROR
17     FORMAT(5(2X,E12.5))
22     FORMAT(/6X,'YA',12X,'YHAT'10X,'YYHAT',9X,'ERROR',9X,'ZERROR')
      WRITE(12,17) YA,YHAT,YYHAT,ERROR,ZERROR
73   CONTINUE
    WRITE(13,77)IY
77   FORMAT(I10)
    STOP
    END
```

```
C*******************************************************************
C
C    SUBROUTINE UNKNOW
C
C    THIS ROUTINE DEFINES THE UNKNOWN SYSTEM
C
C*******************************************************************
C
      DOUBLE PRECISION FUNCTION UNKNOW(X,X1)
      DOUBLE PRECISION H(25),YHAT,XT1,XT2
      LW = 5
      H(1) = .2
      H(2) = -.4
      H(3) = .03
      H(4) = -.7
      H(5) = 0.0
      H(6) = .5
      H(7) = .35
      H(8) = .11
      H(9) = .9
      H(10) = 0.0
      H(11) = .01
      H(12) = 1.3
      H(13) = -.33
      H(14) = .7
      H(15) = 0.0
      H(16) = .43
      H(17) = .81
      H(18) = -.05
      H(19) = .4
      H(20) = 0.0
      H(21) = 0.0
      H(22) = 0.0
      H(23) = 0.0
      H(24) = 0.0
      H(25) = 0.0
      YHAT = 0.0
      DO 14 J = 1,LW
        JM1 = J - 1
        XT2 = COORD(X1,JM1)
        DO 15 K = 1,LW
          KM1 = K - 1
          XT1 = COORD(X,KM1)
C           WRITE(6,52)XT1,XT2
C52         FORMAT(2(2X,E12.5))
          YHAT = YHAT + XT1 * XT2 * H(JM1*LW + K)
15      CONTINUE
14    CONTINUE
      UNKNOW = YHAT
      RETURN
      END
```

```
C*********************************************************************
C
C    FUNCTION COORD
C
C    GENERATES OUTPUT OF THE FUNCTIONS BEING USED AS COORDINATES
C
C    CREATED 23 AUG 84
C
C*********************************************************************
C
     DOUBLE PRECISION FUNCTION COORD(X,I)
C
C    USE LEGENDRE POLYNOMIALS
C
C     IF (I.NE.0) GO TO 1
C       Y = 1.0
C       GO TO 30
C1    IF (I.NE.1) GO TO 2
C       Y = 1.732051*X
C       GO TO 30
C2    IF (I.NE.2) GO TO 3
C       Y = 3.354102*(X*X - 1./3.)
C       GO TO 30
C3    Y = 6.61438*(X*X*X - 3./5.*X)
C
C    USE SIMPLE POWER SERIES TYPE POLINOMIALS
C
     Y = 1.0
     IF (I.EQ.0) GO TO 30
     Y = X**I
30   COORD = DBLE(Y)
     RETURN
     END
```

```
C*********************************************************************
C
C    SUBROUTINE SOLVE
C
C    SUBROUTINE TO SOLVE A SYSTEM OF LINEAR EQUATIONS
C    USES GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
C*********************************************************************
C
     SUBROUTINE SOLVE(A,Z,K)
     DOUBLE PRECISION A(25,26),Z(25),Y,TEMP,LARGE
C
     KM1 = K - 1
     KP1 = K + 1
C
C
     DO 10 L = 1,KM1
       LP1 = L + 1
       DO 11 I = LP1,K
         LARGE = DABS(A(L,L))
         LROW = L
         DO 12 M = LP1,K
           IF (DABS(A(M,L)).LE.LARGE) GO TO 12
             LARGE = DABS(A(M,L))
             LROW = M
12        CONTINUE
C
         IF (L.EQ.LROW) GO TO 13
           DO 14 M = L,KP1
             TEMP = A(L,M)
             A(L,M) = A(LROW,M)
             A(LROW,M) = TEMP
14         CONTINUE
C
13       Y = A(I,L)/A(L,L)
C
         DO 15 M = 1,KP1
           A(I,M) = A(I,M) - A(L,M)*Y
15       CONTINUE
11     CONTINUE
10   CONTINUE
C
C
     Z(K) = A(K,KP1)/A(K,K)
     DO 16 L = 1,KM1
       I = K - L
       Z(I) = A(I,KP1)
       KMI = K - 1
       DO 17 M = 1,KMI
         J = K - M + 1
         Z(I) = Z(I) - A(I,J)*Z(J)
17     CONTINUE
       Z(I) = Z(I)/A(I,I)
```

```
16   CONTINUE
C
C
     RETURN
     END
```

```
C*****************************************************************
C
C    NONLINEAR CORRELATION MATRIX CALCULATOR
C
C    CREATED 23 AUG 84
C    USES FUNCTION UNKNOW TO DETERMINE FUNCTIONS FOR EXPANSION.
C    X - INPUT VECTOR
C    Y - OUTPUT VECTOR
C    PHI - CORRELATION MATRIX
C        - LAST COLUMN CONTAINS INPUT/OUTPUT CROSS-CORRELATION
C
C*****************************************************************
C
     SUBROUTINE NCRLAT(X,Y,PHI,LSQW,LW,N)
     DOUBLE PRECISION PHI(25,26),X(1),Y(1),TX(5,5)
C
C
     NM1 = N - 1
     LSQW = LW * LW
     LSQWP1 = LSQW + 1
C
C
     DO 40 I=1,LSQW
       DO 39 J=1,LSQWP1
         PHI(I,J) = 0.0
39     CONTINUE
40   CONTINUE
C
C
     DO 2 M=2,N
       MM1 = M-1
       DO 3 I=1,LW
         IM1 = I - 1
         X2 = COORD(X(MM1),IM1)
         DO 4 J=1,LW
           JM1 = J - 1
           X1 = COORD(X(M),JM1)
C           WRITE(6,81) X1,X2
C81          FORMAT(2(2X,E12.5))
           TX(I,J) = X1 * X2
C           WRITE(6,80) X(K),X(MM1),I,J,TX(I,J)
C80          FORMAT(2X,2(2X,E12.5),2(2X,I2),E12.5)
4        CONTINUE
3      CONTINUE
C
C
       DO 5 I=1,LW
         DO 6 II=1,LW
           K = (I-1)*LW + II
           PHI(K,LSQWP1) = PHI(K,LSQWP1) + Y(M)*TX(I,II)
           DO 7 J=1,LW
             DO 8 JJ=1,LW
               KK = (J-1)*LW + JJ
```

```
                PHI(K,KK) = PHI(K.KK) + TX(I,II)*TX(J,JJ)
8             CONTINUE
7           CONTINUE
6         CONTINUE
5       CONTINUE
2     CONTINUE
C
      DO 31 I=1,LSQW
        DO 32 J=1,LSQWP1
          PHI(I,J) = PHI(I,J)/FLOAT(NM1)
32      CONTINUE
31    CONTINUE
C
C
C     DO 17 I=1,LSQW
C       WRITE(11,16) (PHI(I,J), J = 1,LSQW)
C16      FORMAT(/,9(2X,E12.5))
C17   CONTINUE
      WRITE(11,19) (PHI(I,LSQWP1),I = 1.LSQW)
19    FORMAT(/,/,9(2X,E12.5))
C
C
      RETURN
      END
```

```
C*********************************************************************
C
C     NONLINEAR WIENER MODELLING PROGRAM
C     THIS USES THE LMS ADAPTIVE ALGORITHM 14 FEB 84 (VALENTINE'S)
C
C     UNKNOWN SYSTEM DEFINED IN FUNCTION UNKNOW
C
C*********************************************************************
C
      DOUBLE PRECISION TX(5,5),H(5,5),VAR,X,XM1,Y,YHAT,SEED
      READ(13,77)IY
      REWIND 13
      WRITE(6,40)
40    FORMAT(2X,'MAGNITUDE OF NOISE')
      READ(5,41)VAR
41    FORMAT(F12.5)
      WRITE(6,42)
42    FORMAT(2X,'NUMBER OF POINTS')
      READ(5,43)N
43    FORMAT(I5)
      WRITE(6,44)
44    FORMAT(2X,'MAXIMUM POWER OF X ')
      READ(5,45)LW
45    FORMAT(I1)
      WRITE(6,7)
7     FORMAT(2X,'CONVERGENCE FACTOR')
      READ(6,8)U
8     FORMAT(F12.5)
      WRITE(12,46)N
      WRITE(6,46)N
46    FORMAT(2X,'THE NUMBER OF POINTS WAS ',I5)
      WRITE(12,47)LW
      WRITE(6,47)LW
47    FORMAT(2X,'THE MAXIMUM POWER OF X IS ',I1)
      WRITE(6,48)VAR,VAR
      WRITE(12,48)VAR,VAR
48    FORMAT(2X,'THE NOISE IS UNIFORM FROM -',F9.5,' TO +',F9.5)
      WRITE(6,49)U
      WRITE(12,49)U
49    FORMAT(2X,'THE CONVERGENCE FACTOR WAS ',E12.5)
      LW = LW - 1
      VAR = VAR*2.0
      XM1 = VAR*(URAND(IY) - .5)
C
C  INITIALIZE THE H TENSOR
C
      DO 10 I = 1,LW
        DO 9 J = 1,LW
          H(I,J) = 0.0
9       CONTINUE
10    CONTINUE
C
C  BEGIN THE ITERATION LOOP
```

```
C
      DO 2 K=2,N
        X = VAR*(URAND(IY) - .5)
        Y = UNKNOW(X,XM1)
C         WRITE(6,97)I,X,Y
C97       FORMAT(2X,I5,2X,E12.5,2X,E12.5)
        YHAT = 0.0
C
C     CALCULATE THE MEASUREMENT TENSOR AND THE OUPUT ESTIMATE
C
        DO 14 I = 1,LW
          IM1 = I - 1
          X2 = COORD(XM1,IM1)
          DO 15 J = 1,LW
            JM1 = J - 1
            X1 = COORD(X,JM1)
C             WRITE(6,52)X1,X2
C52           FORMAT(2(2X,E12.5))
            TX(I,J) = X1*X2
            YHAT = YHAT + TX(I,J) * H(I,J)
15        CONTINUE
14      CONTINUE
C
C     CALCULATE THE NEW VALUE OF THE TENSOR
C
        ERROR = Y - YHAT
        DO 5 I = 1,LW
          DO 4 J = 1,LW
            H(I,J) = H(I,J) + 2.0*U*ERROR*TX(I,J)
4         CONTINUE
5       CONTINUE
        XM1 = X
C
C     PRINT THE NEW VALUE OF THE TENSOR
C
        KM1 = K - 1
        IF (KM1.NE.(KM1/100)*100)GO TO 2
        WRITE(6,13)KM1
        WRITE(12,13)KM1
13      FORMAT(/,2X,'ITERATION NUMBER ',I5)
        WRITE(6,135)
        WRITE(12,135)
135     FORMAT(2X,'THE NEW VALUE OF THE H TENSOR IS')
        DO 27 I = 1,LW
          WRITE(6,11) (H(I,J),J = 1,LW)
          WRITE(12,11) (H(I,J),J = 1,LW)
11        FORMAT(5(2X,E12.5))
27      CONTINUE
        WRITE(6,22)
        WRITE(12,22)
        WRITE(6,17) Y,YHAT,ERROR
17      FORMAT(5(2X,E12.5))
22      FORMAT(6X,'Y',12X,'YHAT',10X,'ERROR')
```

```
      WRITE(12,17) Y,YHAT,ERROR
2     CONTINUE
      WRITE(13,77)IY
77    FORMAT(I10)
      STOP
      END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, California, 93943	2
2.	Defence Technical Information Centre Cameron Station Alexandria, Virginia 22304-6145	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgradute School Monterey, California, 93943	1
4.	Dr. S.R. Parker, Code 62Px Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California, 93943	15
5.	Dr. P.H. Moose, Code 62Me Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California, 93943	1
6.	Dr. L.J. Ziomek, Code 62Zm Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California, 93943	1
7.	Dr. E.C. Crittenden, Code 61Ct Department of Physiscs Naval Postgraduate School Monterey, California, 93943	1
8.	Dr. U.R. Kodres, Code 52Kr Department of Computer Science Naval Postgraduate School Monterey, California, 93943	1
9.	Lt(N) P.J. Lenk DMCS 3-2-8 DGMEM, CEM, NDHQ 101 Colonel By Drive, Ottawa, Ontario Canada, K1V 0K2	6
10.	Dr. Ahmet H. Kayran Vanikoy, Cad. No. 17 Istanbul, Turkey	1

11.	DMCS DGMEM, CEM, NDHQ 101 Colonel By Drive, Ottawa, Ontario Canada, K1V 0K2	5
12.	Dr. Bharat Madan Department of Computer Science and Engineering IIT Delhi New Delhi, India 110029	1