NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

PARAMETRIC MODELING OF LINEAR AND NONLINEAR SYSTEMS

by

Francis Anthony Perry

June 1980

Thesis Advisor; S. R. Parker

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A089501 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Parametric Modeling of Linear and Nonlinear Systems	5. TYPE OF REPORT & PERIOD COVERED Doctoral Dissertation June 1980
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Francis Anthony Perry	8. CONTRACT OR GRANT NUMBER(s) (9) Doctoral thesis
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940 (11)	12. REPORT DATE Jun[illegible]80
	13. NUMBER OF PAGES 285
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 286	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution 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)
Modeling, Parametric Modeling, Linear Systems, Nonlinear Systems, Autoregressive, Moving Average, Autoregressive Moving Average, Nonlinear Autoregressive Moving Average, AR, MA, ARMA, Volterra Series, Lattice Filter, Multichannel Filter, Adaptive Lattice Filter, Levinson Algorithim

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The problem of obtaining parametric models for linear and nonlinear systems based on observations of the input and output of the system is one of wide ranging interest. For linear systems, moving average (MA) and autoregressive (AR) models have received considerable attention and based on the Levinson algorithm, a number of very powerful methods involving lattice filter structures have been developed to obtain the model solutions. For nonlinear systems the Volterra series model which is a nonlinear extension of the moving

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601
(Page 1)

251450

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

average model is frequently used.

The purpose of this research is to extend these techniques to more general linear and nonlinear models. Using the equation error formulation, lattice solution methods in batch processing and adaptive form are developed for both single and multichannel autoregressive moving average (ARMA) models for linear systems and Volterra series models for nonlinear systems. A nonlinear extension of the ARMA model is also considered and is shown in some cases to remedy problems encountered in Volterra modeling of nonlinear systems. Lattice methods are also developed for the nonlinear ARMA model and it is shown that the methods obtained for linear ARMA modeling follow as a special case of the nonlinear results.

Experimental verification of the methods developed for single channel linear ARMA modeling is presented and used to explore the characteristics of the lattice solution techniques. The results clearly indicate that the lattice methods are extremely powerful, capable of producing highly accurate system models using short runs of data.

Approved for public release; distribution unlimited

Parametric Modeling of Linear and Nonlinear Systems

by

Francis Anthony Perry
Lieutenant, United States Navy
B.S., Pennsylvania State University, 1972
M.S. Naval Postgraduate School, 1978

Submitted in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

from the
NAVAL POSTGRADUATE SCHOOL
June 1980

Author Francis A. Perry

Approved by:

H. A. Titus
Professor of Electrical Engineering

R. W. Hamming
Professor of Computer Science

R. Panholzer
Professor of Electrical Engineering

P. C. Wang
Assoc. Professor of Mathematics
Assoc. Professor of National Security Affairs

S. R. Parker
Professor of Electrical Engineering
Thesis Advisor

Approved by
Chairman, Department of Electrical Engineering

Approved by
Academic Dean

ABSTRACT

The problem of obtaining parametric models for linear and nonlinear systems based on observations of the input and output of the system is one of wide ranging interest. For linear systems, moving average (MA) and autoregressive (AR) models have received considerable attention and, based on the Levinson algorithm, a number of very powerful methods involving lattice filter structures have been developed to obtain the model solutions. For nonlinear systems the Volterra series model which is a nonlinear extension of the moving average model is frequently used.

The purpose of this research is to extend these techniques to more general linear and nonlinear models. Using the equation error formulation, lattice solution methods in batch processing and adaptive form are developed for both single and multichannel autoregressive moving average (ARMA) models for linear systems and Volterra series models for nonlinear systems. A nonlinear extension of the ARMA model is also considered and is shown in some cases to remedy problems encountered in Volterra modeling of nonlinear systems. Lattice methods are also developed for the nonlinear ARMA model and it is shown that the methods obtained for linear ARMA modeling follow as a special case of the nonlinear results.

Experimental verification of the methods developed for single channel linear ARMA modeling is presented and used

to explore the characteristics of the lattice solution techniques. The results clearly indicate that the lattice methods are extremely powerful, capable of producing highly accurate system models using short runs of data.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

It is with deepest appreciation that I wish to acknowledge my advisor, Professor Sydney R. Parker for his countless hours of patient assistance and guidance over the course of this research. I would like to thank the members of my doctoral committee who have followed this effort, contributing their time and valuable suggestions. I also wish to acknowledge the contributions of Commander Stacey Holmes and Doctor Joel Morris of the Office of Naval Research, and Mr. Nate Butler of the Naval Electronic Systems Command for their support of this research. Finally, I wish to thank my wife, Beth, and daughter, Karen, for their forebearance and encouragement.

I. INTRODUCTION

Traditionally, man has attempted to create models of portions of his environment for two principal reasons:

1. To gain insight and understanding as to their functioning;
2. As a prelude to taking some action such as attempting to exercise control over them.

The field of physics for instance, is replete with examples where men have created models to study and explain phenomenon from planetary motion to the motion and even origin of sub-atomic particles. In designing machines, engineers routinely rely on models of the components they use to describe how they will function, and to obtain the desired results in the final product. Economics is another field where the use of models abounds for such purposes as identifying, forecasting or trying to direct trends.

The scope of the modeling problem is quite broad begining with a decision on the type of model to be used, what physical quantities to measure, how to estimate the parameters of the model from the measurement, and finally a verification of the model. In the chapters that follow, one facet of this problem, that of estimating model parameters, will be explored in detail for a number of linear and non-linear models.

A. THE PROBLEM STATEMENT

The primary concern of this work is the determination of discrete time models for both linear and nonlinear, time invariant systems from sampled observations of the system inputs and outputs. The general approach underlying all of the models examined here assumes that the system to be modeled is described by the equation

$$y(k) = F_{10}[u(k)]+F_{20}[y(k-1)]+F_{30}[u(k),y(k-1)] \qquad (1.1)$$

where F_{10}, F_{20} and F_{30} are functions of past and present values of their arguments, and u(k) and y(k) are the system input and output respectively. This is depicted in Figure (1.1). A possible method for modeling this type of system is to create a model of exactaly the same configuration with functions F_{10}, F_{20} and F_{30},[1] assume a form for these functions, operate the system and model in parallel with the same input and adjust the parameters of the model functions to minimize the mean square error (MSE) between the model output $\hat{y}(k)$ and the system output. The symbol "^" is used here to indicate that the signal is an estimate of y(k). This is depicted in Figure 1.2 and is often referred to as direct form modeling since the assumed topology of the system is directly copied

[1] Script notation will be used to refer to quantities associated with the system while nonscript notation will be used for their corresponding approximants in the model.

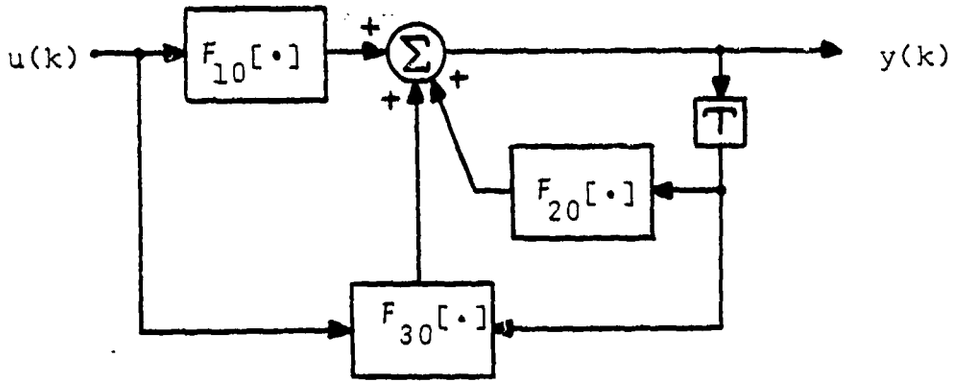


Figure 1.1. The assumed form for systems to be modeled. T represents a unit delay.

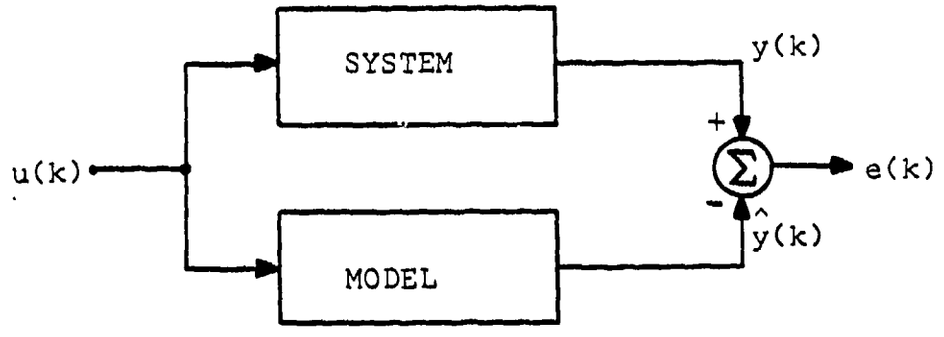


Figure 1.2. A direct approach to system modeling.

by the model. Here the model output is given by

$$\hat{y}(k)=F_{10}[u(k)]+F_{20}[\hat{y}(k-1)]+F_{30}[u(k),\hat{y}(k-1)] \qquad (1.2)$$

and the error signal is referred to as the "output error". As an example, if the system is linear an appropriate model is

$$F_{10}[u(k)] = \sum_{i=0}^{N} a(i)u(k-i) \tag{1.3a}$$

$$F_{20}[\hat{y}(k-1)] = \sum_{i=1}^{N} b(i)\ \hat{y}(k-i) \tag{1.3b}$$

$$F_{30}[u(k),\hat{y}(k-1)] = 0 \tag{1.3c}$$

(for linear models, $F_{30}[\cdot]$ will be zero). In general however, the direct form approach will have serious difficulties if either $F_{20}[\cdot]$ or $F_{30}[\cdot]$ are nonzero since the past values of $\hat{y}(k)$ used in these functions are themselves dependent on the model parameters. A mimimum mean square output error approach results in a system of highly nonlinear simultaneous equations which must be solved to obtain the model parameters.

To avoid these difficulties, the equation error approach [Refs. 34 and 23] to system modeling (which uses different model forms in the analysis and synthesis phases) will be applied to the problem. The analysis model is depicted in Figure 1.3 and differs from the direct form model in that F_{20} and F_{30} are functions of past and present values of the delayed system output rather than the analysis model output.

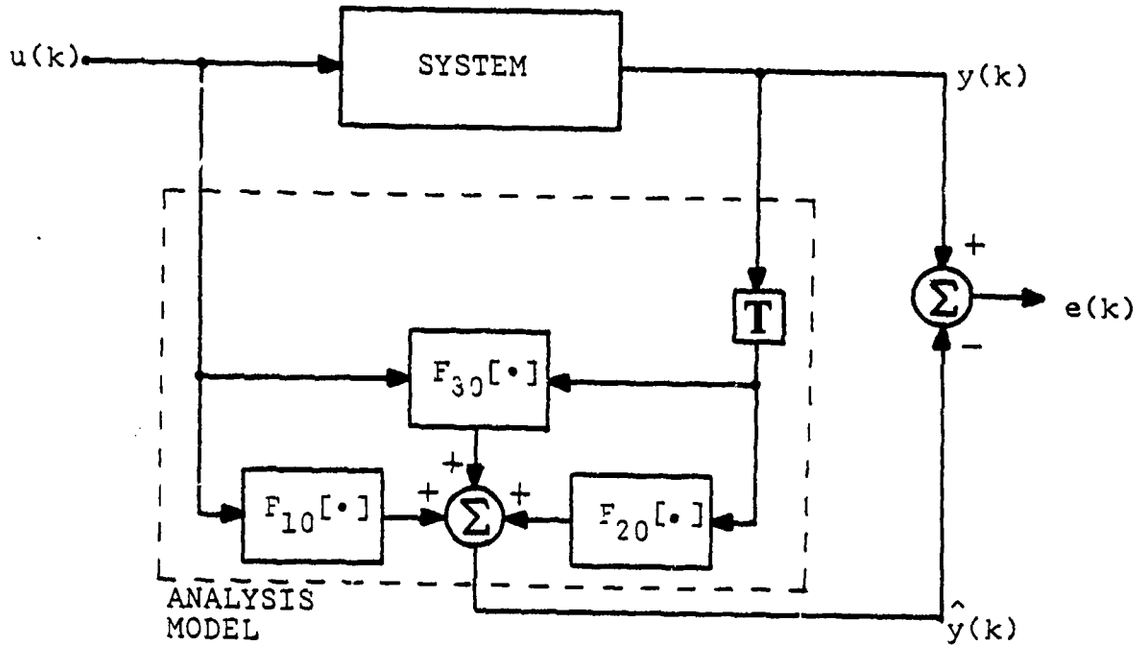


Figure 1.3. The equation error approach for system modeling.

For each of the models studied, a general form for the three functions is assumed and the parameters of the model (coefficients of the functions) are set to obtain a MMSE solution. In each case, the MSE cost function is a quadratic function of the model parameters (due to both the equation error formulation and to the form chosen for the functions) with a unique minimum and therefore the solution is given by a system of linear equations. The synthesis model is of the same form assumed for the system in Figure 1.1 and uses the functions F_{10}, F_{20} and F_{30} determined during the analysis phase.

As an alternative to the topology shown in Figure 1.3 it will occasionally be convenient to consider the error signal e(k) as the output of the analysis model rather than the prediction $\hat{y}(k)$. This can be accomplished in one of two ways by defining

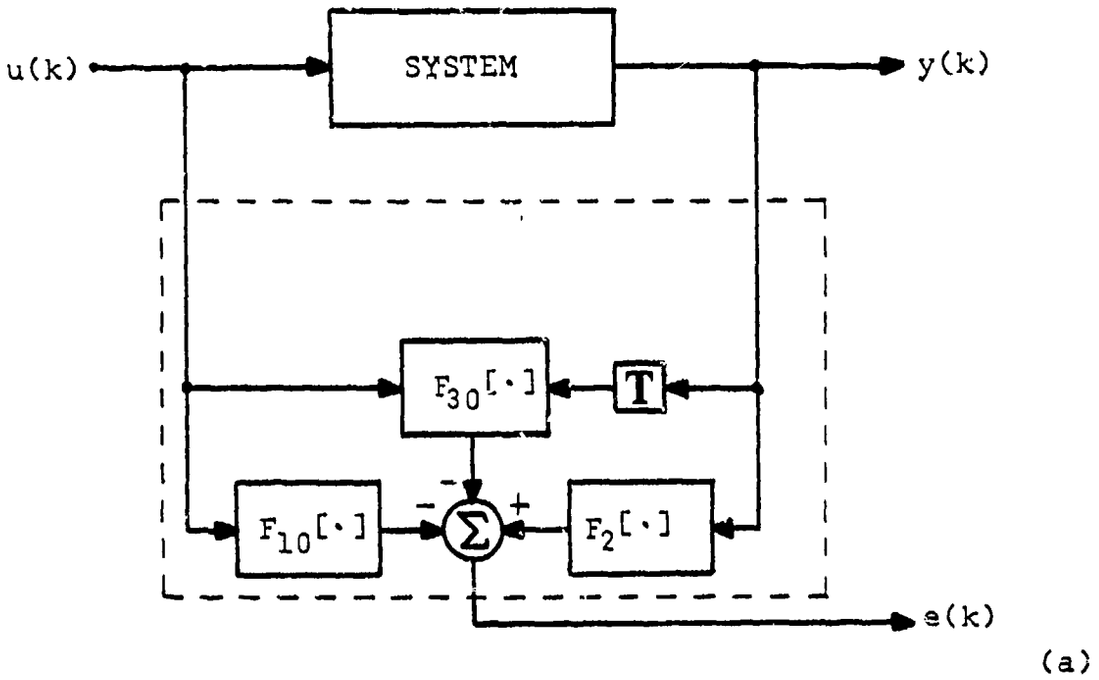
$$F_2[y(k)] = y(k) - F_{20}[y(k-1)] \tag{1.4a}$$

or

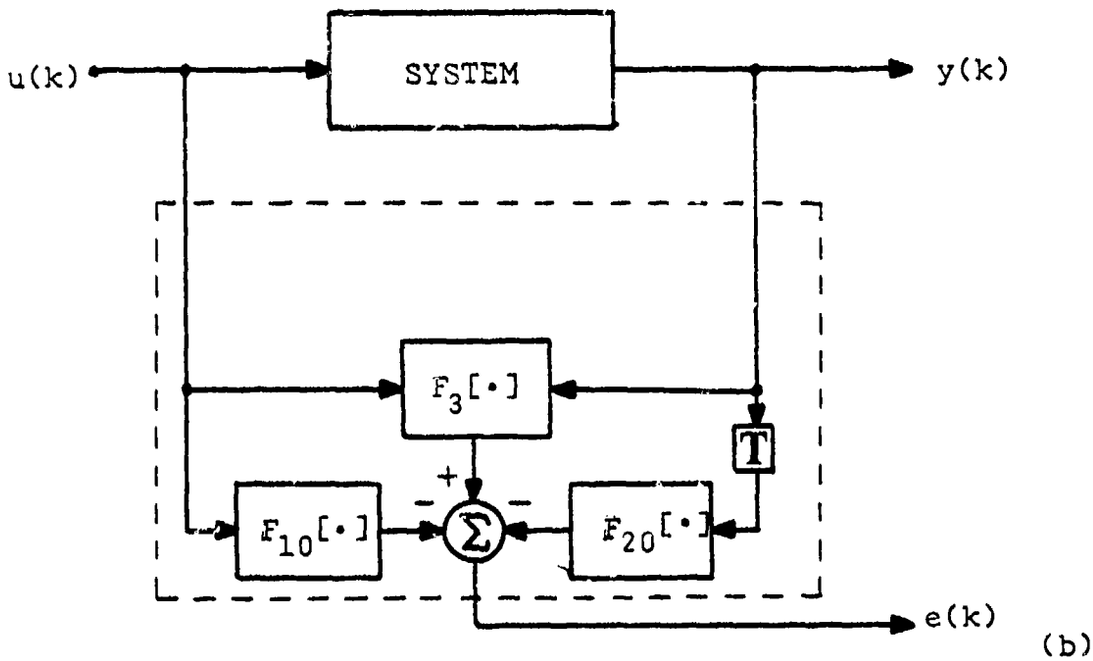
$$F_3[u(k),y(k)] = y(k)-F_{30}[u(k),y(k-1)] \tag{1.4b}$$

and reformulating the analysis model as shown in Figures 1.4a or 1.4b. These model forms are often referred to as prediction error models since their outputs are the errors in predicting y(k) rather than the predictions themselves. There are, however, no substantive differences between the modeling approaches depicted in Figure 1.3, 1.4a and 1.4b.

The equation error formulation can be generalized to multiple input multiple output systems as well (henceforth referred to as multichannel systems) by considering u(k) and y(k) as vectors of Q_i input signals and Q_0 output signals and F_{10}, F_{20}, F_{30}, F_2 and F_3 as vector functions. The prediction error signal e(k) becomes a Q_0-vector of signals and the model parameters can be set to minimize the trace of the prediction error covariance matrix $\underline{P} = \varepsilon\{\underline{e}(k)\underline{e}(k)^T\}$. Such generalizations have been developed to a degree in the multichannel filtering literature and will be investigated further here.



(a)



(b)

Figure 1.4. Prediction error model forms.

It is important to keep in mind however, that while the equation error formulation can be used to find a model solution, it is an indirect method as opposed to the direct form method which minimizes the mean square value of output error. The direct form model has been modified to obtain the equation error analysis model so that the parameters can be obtained via the solution of systems of linear equations. The price paid for this simplification in the model analysis problem is that additive noise on the measured system output will introduce a bias in the model coefficient estimates.

B. OVERVIEW

Chapter II along with appendices A through F provide a unified review of the existing background theory on minimum mean square equation error modeling of linear systems. The moving average (MA) and autoregressive (AR) models are presented and their relative merits compared. In Section II.C. the Levinson algorithm [Refs. 9, 10 and 27] for the AR and MA models is developed, greatly simplifying the solution process for these models. Section II.D. then shows that the Levinson algorithm defines the AR and MA models in terms of lattice filter structures.

These lattice structures have received widespread attention and have led to a host of new developments in modeling, spectral estimation, filter structures and adaptive filtering. Examination of the properties of these forms have suggested a number of new methods for calculating model coefficients

that offer increased accuracy, and in some cases guarantee model stability even in the presence of correlation estimates obtained by averaging over short time intervals [Refs. 5, 20, 29 and 36]. Applied by Burg [Ref. 5] to spectral estimation, these methods allow the determination of power spectra via AR modeling from very short runs of data without any need for the use of a window function. In finite precision arithemetic implementations, the lattice structures have been shown by Markel and Gray [Ref. 33] to be less sensitive to roundoff noise and coefficient quantization than direct structures and have led to the design of other structures that offer improved performance over conventional parallel realizations. Griffiths has shown that these lattices can be implemented adaptively [Refs. 16, 17 and 18] and that they offer the potential for more rapid convergence than conventional LMS adaptive filters. Recently Morf [Refs. 36, 37 and 38] has also used these lattice structures to implement a recursively updated deterministic least squares adaptive scheme. It is readily apparent therefore, that the original work of Levinson and the lattice structures that have evolved from it have had an important impact on the field of digital signal processing.

In Section II.E., the multichannel generalization of many of the single channel AR and MA modeling results is presented. After a discussion of the basic multichannel AR and MA models [Refs. 26 and 45], the multichannel version of the Levinson algorithm originally developed by Whittle [Ref. 56],

and Wiggins and Robinson [Ref. 61] is presented. A new form for the models is introduced and used here however, to facilitate the application of these results later in various other modeling problems. Multichannel lattice structures are then derived and from them alternative solution methods for the modeling problems are developed.

Finally, in Section II.F. the LMS adaptive algorithm [Ref. 58] is reviewed and the adaptive implementations of the lattice structures due to Griffiths are presented.

In Chapter III, the more general autoregressive moving average model is presented using the equation error formulation attributed to Kalman [Ref. 23]. After a brief discussion of the model, new model transition formulas are developed showing how the ARMA model is related to the simpler and less general AR and MA models. System input signal requirements for the ARMA modeling process are explored and it is shown that the power spectrum of the input signal can be considered as a frequency dependent weighting function in the model optimization. Then the main result of the chapter is presented. With suitable assumptions, a recursive in order solution method for ARMA modeling (the (n+1)=st order solution is obtained from the n-th order solution) is obtained based on the Levinson algorithm for multichannel AR models. From this, lattice solution methods for the ARMA model are developed in both batch processing and adaptive form. (Batch processing here refers to assuming ergodicity and estimating correlations with time averaging).

A similar result has recently been presented by Morf [Refs. 37 and 38] with the assumption of a white noise input signal to the system. The results presented here follow from a different approach without the assumption of a white noise input. Experimental results are also presented verifying the methods and theory, and showing their advantages (and disadvantages) over conventional ARMA modeling methods. The programs used in these simulations are listed in Appendix J. In Section III.F., and Appendix G, it is shown that these single channel methods readily extend to the multichannel ARMA model, and as one would expect, can be obtained as a special case.

In Chapter IV two types of nonlinear models, the Volterra series model and the new nonlinear ARMA model recently proposed by Parker [Ref. 64], are considered. After a brief discussion of the Volterra model, it is shown that the solution can be obtained using multichannel MA lattice methods if the regular form of the Volterra kernels is used in place of the conventional symmetric form. Then the nonlinear ARMA model is presented in Section IV.B. and it is shown that for many systems, this model can remedy the problem of the large number of terms (ideally infinite) required by the Volterra model to represent the system in much the same way that the ARMA model solved the problem arising in the MA model. In Section IV.B.2 it is also shown that by using the regular form, the solution for the nonlinear ARMA model can be obtained using multichannel AR lattice methods and that the linear ARMA model solutions developed in Chapter III follow

as a special case. Appendix I then presents two examples of nonlinear ARMA modeling. First a somewhat academic example of a cascade of linear and nonlinear subsystems is given then a nonlinear ARMA model is proposed for the tracking behavior of a phase locked loop.

Finally, in Chapter V, two applications for the linear and nonlinear ARMA modeling methods developed in Chapters III and IV are discussed briefly. (They are reduced order modeling of complex systems and modeling for fault detection and diagnosis.) Then in Section V.B. conclusions are drawn on the results of this work and a list of significant open questions (both old unanswered questions and new ones raised here) is compiled.

II. DISCRETE TIME LINEAR SYSTEM MODELING; BACKGROUND THEORY

While few physical systems are absolutely linear, linear models often suffice to accurately describe their behavior under normal operating conditions. A rich body of theory has therefore been developed for the analysis and modeling of linear systems [Ref. 22] and a thorough knowledge of this theory is vitally important to anyone interested in understanding the functioning of these systems. The continuing expansion in the availability of powerful, inexpensive digital computing capabilities has also made discrete time techniques take on a special prominence. With this as motivation, the portion of the background theory in discrete time linear modeling upon which much of the remainder of this work depends, is developed here from the unifying standpoint of a minimum mean square equation error model solution.

The moving average and the autoregressive models are developed first for single input single output systems. Their solution via the Levinson algorithm is presented and from this algorithm alternate solution methods based on lattice filter structures are derived. It is shown that almost all of these results can be generalized to the multiple input multiple output case and the corresponding multichannel modeling methods are developed. Finally, adaptive implementation of the modeling methods for both

the conventional filter structures and the lattice filter structures is presented as an alternative means of obtaining model solutions.

A. MOVING AVERAGE MODELS

The moving average (MA) model was among the earliest discrete models developed. [Refs. 4, 11 and 19] It estimates the current value of the output of a system as a weighted combination of the present value and N past values of the system input where N is the order of the model. The problem then is to estimate the weighting function or impulse response of the MA model in some fashion. Since the MA model characterizes a system in terms of a finite duration approximation of its impulse response and since any linear time invariant, single input single output system is completely specified by its impulse response, the MA model is quite general and can be used for a wide class of systems. Defining (N+1)-vectors of model weights and input data as

$$\underline{a}^{+T} = [a(0) \cdots a(N)] \quad [1,2] \qquad (2.1a)$$

---

[1] A superscript "+" is used to indicate that in spite of the fact that these vectors are used for a N-th order model, they are (N+1)-vectors with elements indexed from zero to N rather than from one to N. Superscript T demotes transpose.

[2] Superscripts in parenthesis will later be added to the model coefficient vectors to explicitly indicate their dependence on the order of the problem being solved. They are omitted for simplicity however whenever doing so does not result in ambiguous notation.

$$\underline{u}^+(K)^T = [u(k) \cdots u(k-N)] \qquad (2.1b)$$

the MA estimate of the system output becomes

$$\hat{y}(k) = \sum_{n=0}^{N} a(n)u(k-n) = \underline{u}^+(k)^T \underline{a}^+$$

In terms of the modeling approach of Figure 1.3, F_{20} and F_{30} are assumed to be zero. F_1 is a linear time invariant function of past and present values of u(k). Assuming stationarity, an expression for the mean square value of the error as a quadratic function of the weights {a(n)} is given by

$$E_2 = \underline{a}^{+T} \underline{R}_{u^+u^+} \underline{a}^+ - 2\, \underline{a}^{+T} \underline{r}_{u^+y} + R_{uu}(0) \qquad (2.2)$$

where in general $R_{vw}(n) = \varepsilon\{v(k)w(k+n)\}$, $\underline{r}_{vw} = \varepsilon\{\underline{v}(k)w(k+n)\}$, $\underline{R}_{vw} = \varepsilon\{\underline{v}(k)\underline{w}(k)^T\}$ and $\varepsilon\{\ \}$ denotes expectation.

$$\underline{R}_{u^+u^+} = \begin{bmatrix} R_{uu}(0) & \cdots & R_{uu}(-N) \\ \vdots & & \vdots \\ R_{uu}(N) & \cdots & R_{uu}(0) \end{bmatrix}$$

$$\underline{r}_{u^+y} = [R_{uy}(0) \quad \cdots \quad R_{uy}(N)]^T$$

The surface described by equation 2.2 can be pictured as a concave hyperparaboloid with a unique minimum and the

characteristics of such a surface are described in Appendix F. For example when N=1, the MSE as a function of a(0) and a(1) appears as shown in Figure 2.1.

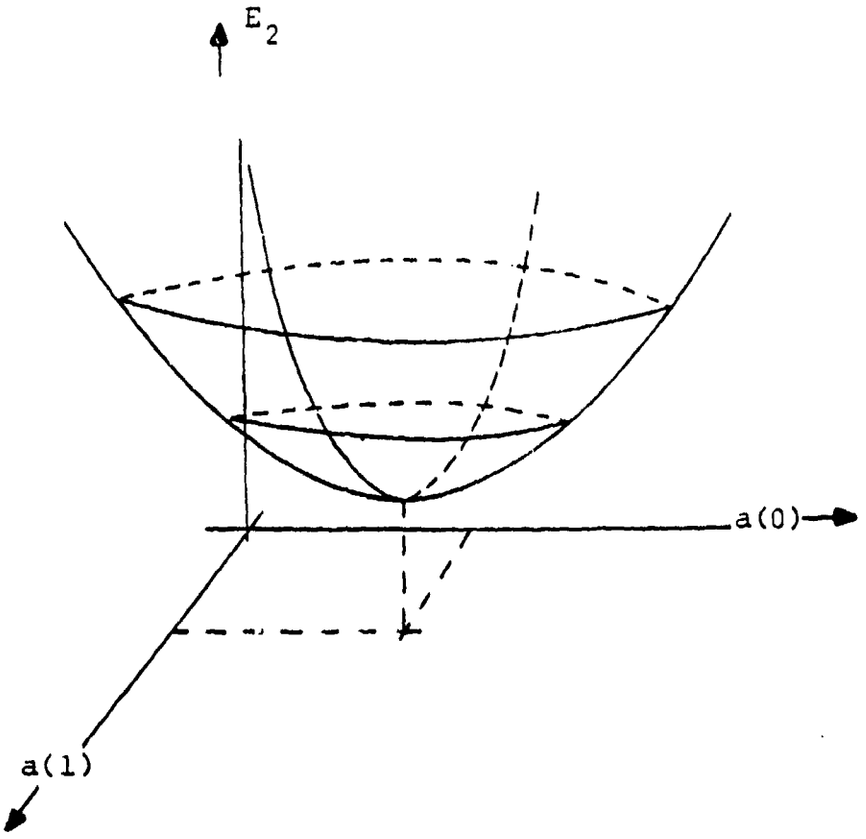


Figure 2.1. MSE as a function of model weights for a first order (N=1) MA model.

The minimum mean square error solution for the coefficients is given by

$$\underline{R}_{u^+u^+} \underline{a}^+_{OPT} = \underline{r}_{u^+y} \tag{2.3}$$

and the corresponding minimum value of the cost function E_2 is

$$E_{2_{min}} = R_{yy}(0) - \underline{a}^{+T}_{OPT} \; \underline{r}_{u^+y} \tag{2.4}$$

Equation (2.3) is a discrete time matrix form of the Wiener Hopf equation

$$R_{uy}(\tau) = \int_{-\infty}^{\infty} R_{uu}(\tau-\lambda)\, h(\lambda)\, d\lambda \tag{2.5}$$

where $u(\tau)$ and $y(\tau)$ are the continuous input and output signals and $h(\tau)$ is the system impulse response. The process of finding $\underline{a}^{+}_{OPT}$ in equation (2.3) is the discrete time equivalent of deconvolving the input autocorrelation function from the cross correlation of input and output to obtain the system impulse response in equation (2.5). Consequently the MA modeling process has been called discrete Wiener filtering or stochastic deconvolution.

This model constitutes a direct form approach as defined in section I.1 but does not encounter difficulty in obtaining the model weights since both F_{20} and F_{30} are assumed to be zero. As such, it possesses the advantage that the estimates of the model parameters will not be biased by the presence of additive noise on the output of the system as shown in Figure 2.2, as long as the noise is uncorrelated with the input signal. This can readily be seen by replacing v in equation 2.3 by y + v. Additive noise on the input signal

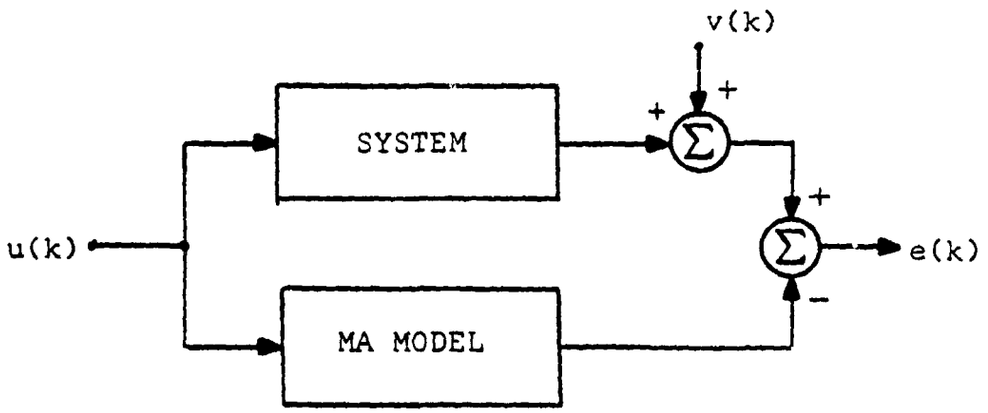


Figure 2.2. Moving Average Modeling

however will adversely affect the modeling process, and introduce a bias in the solution for the model coefficients.

In the transform domain, the model can be represented by a polynomial in powers of z^{-1} and has therefore been referred to as an <u>all zero model</u>

$$A(z) = \sum_{n=0}^{N} a(n)\, z^{-n} \qquad (2.6)$$

In terms of this transfer function relationship, any bias introduced in one or more of the model coefficients has the effect of shifting the zero locations of the model.

In summary a discussion of the advantages and disadvantages of MA modeling is instructive.

Advantages:

1) The solution for the model parameters involves only linear equations.

2) The solution is unbiased in the presence of additive noise on the system output as long as the noise and system input are uncorrelated.

3) Since the model is nonrecursive it is always stable.

Disadvantages:

1) The number of terms (N+1) needed for sufficient model accuracy may be quite large.

2) The solution of a large system of equations is required.

3) The required correlation terms are usually not known and must be estimated by assuming ergodicity and averaging in time. This requires the data to be windowed and set to zero outside the averaging interval in order to maintain the even symmetry of the autocorrelation functions.

4) The modeling process is restricted to linear time invariant systems.

B. AUTOREGRESSIVE MODELS

The autoregressive (AR) model attempts to deal with one of the difficulties (1) encountered in MA modeling; the need for a large number of coefficients to accurately describe the model. [Refs. 2, 4, 11, 19 and 28] In AR modeling, which is sometimes referred to as linear prediction, a prediction error approach is considered where

$$e(k) = y(k) - \sum_{n=1}^{N} b(n)\, y(k-n) \qquad (2.7a)$$

This can be written as

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

$$= y(k) - \underline{y}(k)^T \underline{b} \qquad (2.7b)$$

with

$$\underline{y}(k) = [y(k-1) \cdots y(k-N)]^T \qquad (2.7c)$$

and

$$\underline{b} = [b(1) \cdots b(N)]^T \qquad (2.7d)$$

Here F_{10} and F_{30} are assumed to be zero (this assumption will be modified later to allow a dependence on the input signal in the synthesis phase) and F_{20} provides an estimate of the current value of the system output as a weighted sum of N past outputs. The mean square value of prediction error as a quadratic function of the weights {b(n)} is given by

$$E_2 = \underline{b}^T \underline{R}_{yy} \underline{b} - 2\, \underline{b}^T \underline{r}_{yy} + R_{yy}(0) \qquad (2.8a)$$

and the corresponding MMSE solution for the weights is given by

$$\underline{R}_{yy} \underline{b}_{OPT} = \underline{r}_{yy} \qquad (2.8b)$$

with

$$E_{2_{min}} = R_{yy}(0) - \underline{b}_{OPT}^{T} r_{yy} \tag{2.8c}$$

Using equation 2.7a, an expression can be written in the transform domain for the prediction error model which accepts y(k) as its input and produces the error sequence e(k) as its output.

$$\frac{E(z)}{Y(z)} = 1 - \sum_{n=1}^{N} b(n) z^{-n} = B(z) \tag{2.9}$$

If it is assumed that the system input output relationship can be represented in transfer function form with

$$\frac{Y(z)}{U(z)} = H(z) = \frac{a(0)}{1 - \sum_{n=1}^{N} b(n)\ z^{-n}} = \frac{a(0)}{B(z)} \tag{2.10}$$

and that the model parameters can be determined so that B(z) = $\mathcal{B}$(z), then the prediction error output will be exactly e(k) = a(0) u(k). For this reason AR prediction error modeling has often been called inverse filtering since the prediction error filter essentially reverses the actions of the system (with the exception of a gain). Since the analysis model is in this inverse form rather than in the direct form, the presence of additive noise on the measurement of the system output as shown in Figure 2.3 will introduce a bias in the solution for the model parameters. This

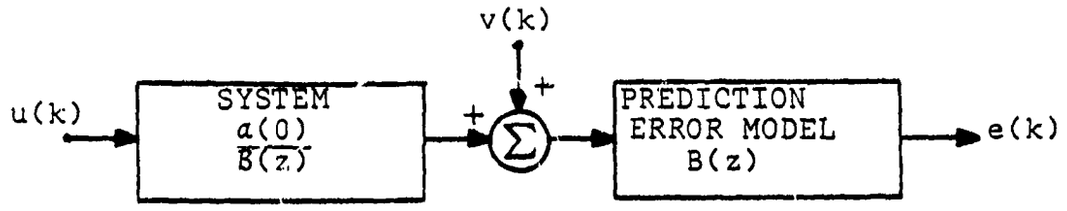


Figure 2.3. Autoregressive prediction error modeling as an inverse filtering process.

has the effect of shifting the roots of B(z) which are estimates of the poles of the system and is the price paid for the ability to obtain the model solution from a set of linear equations.

Thus far, only the analysis portion of the AR modeling process has been discussed. With the inverse filtering interpretation of the prediction error analysis model, a reasonable synthesis model is given in transfer function form as

$$H(z) = \frac{a(0)}{B(z)} \tag{2.11}$$

with the gain term set so that the mean square value of a(0) u(k) is the same as that of the prediction error signal. Thus it follows that

$$a(0)^2 = \frac{\mathcal{E}\{e(k)^2\}}{R_{uu}(0)} \tag{2.12}$$

Since the synthesis model is in the form of an all pole filter, an appropriate impulse response with infinite duration might be obtained using a low order model (small N), a result that is impossible to obtain in any finite order MA model. This is not to say however that a low or even finite order AR model will always be an appropriate model for any linear system. If the transfer function representation for a system contains both zeros and poles, no finite order AR or MA model can serve to exactly represent it. This fact can be understood by considering the form of a geometric series

$$\frac{1}{1-Cz^{-1}} = \sum_{n=0}^{\infty} (Cz^{-1})^n \qquad \text{for } |Cz^{-1}| < 1 \qquad (2.13)$$

which shows that a single pole can be represented by an infinite number of zeros and visa versa. Thus if the system has a single zero, a high order AR model may be required to represent it with sufficient accuracy.

In summary, the advantages and disadvantages of AR modeling may be listed as follows:

Advantages:

1) The solution for the model parameters involves only linear equations.
2) Sometimes an appropriate infinite impulse response can be obtained with a small number of parameters in the model.
3) Direct knowledge or measurement of the system input is not required for determing the system poles. Only a knowledge of its mean square

value is necessary for determining the gain factor.

Disadvantages:

1) The model is biased by the presence of additive noise on the measured system output signal.
2) The number of terms required for sufficient model accuracy may be quite large if zeros are present in the system. If this occurs, the inversion of a large matrix will be required.
3) The required correlation terms are usually not known and must be estimated by assuming ergodicity and using time averages.
4) The modeling process is restricted to linear time invariant systems.

This list of advantages and disadvantages is quite similar to the one compiled for MA models with two notable differences; the bias in the model and the absence of a requirement for input measurements. This second point is significant in that it has led to the application of AR modeling to many problems where an input signal is unmeasurable or indeed does not exist including speech modeling and spectral estimation. [Refs. 2, 5, 12, 15, 21, 32 and 44] The noise problem has restricted the process to applications where measurements with sufficiently high signal to noise ratio are available, making the effects of the bias minimal. [Refs. 24 and 43]

C. RECURSIVE IN ORDER SOLUTIONS FOR AR AND MA MODELS

The preceeding disucssions of the AR and MA modeling problems tacitly assummed an a priori knowledge of the correct model order. If this knowledge is not available a reasonable approach for determining the correct model order must be developed [Ref. 53]. A commonly employed strategy is to successively increment the model order while observing the MSE until further increases fail to substantially reduce the MSE. This requires solving for a number of different models and can be an arduous task if equations (2.8b) or (2.3) are employed directly.

The autocorrelation matrices appearing in the AR and MA model equations (2.8b) and (2.3) are highly structured matrices (both Toeplitz and symmetric) and this fact can be exploited to facilitate the solution of these equations. The Levinson algorithm [Refs. 9, 10 and 27] makes use of this structure to obtain model solutions recursively in order, that is, the solution for the n-th order model is assumed to be known and the solution for the (n+1)-st order model is then obtained from it. In this manner it is possible to start with a first order AR or a zeroth order MA solution given by a single equation and build up the desired order solution. The AR model will be treated first since it is a special case that simplifies the analysis. The simplifications arise due to the fact that the $\underline{r}_{yy}$ vector on the right hand side of equation (2.8b) is made up primarily of terms also appearing on the left hand side in $\underline{R}_{yy}$.

Superscripts in parenthesis are used to explicitly indicate the order of the problem when specifically needed.

1. The Levinson Algorithm For AR Modeling

The n-th order AR model solution of equation 2.8b is given by

$$\underline{R}_{yy}^{(n)}\ \underline{b}^{(n)} = \underline{r}_{yy}^{(n)} \tag{2.14}$$

The Levinson algorithm assumes a relationship between the n-th and (n+1)-st order solutions given by

$$\underline{b}^{(n+1)} = \begin{bmatrix} \underline{b}^{(n)} \\ --- \\ 0 \end{bmatrix} + \begin{bmatrix} \underline{\varepsilon}^{(n)} \\ --- \\ k^{(n+1)} \end{bmatrix} \tag{2.15}$$

and solves for the vector $\underline{\varepsilon}^{(n)}$ and the coefficient $k^{(n+1)}$. Define permuted versions of the vectors $\underline{b}^{(n)}$ and $\underline{r}_{yy}^{(n)}$ as $\underline{f}^{(n)}$ and $\underline{\rho}_{yy}^{(n)}$ by reversing the order of their elements.

$$\underline{f}^{(n)} = \begin{bmatrix} b(n)^{(n)} \\ \vdots \\ b(1)^{(n)} \end{bmatrix} \qquad \underline{\rho}_{yy}^{(n)} = \begin{bmatrix} R_{yy}(n)^{(n)} \\ \vdots \\ R_{yy}(1) \end{bmatrix} \tag{2.16}$$

Because of the Toeplitz symmetric structure of the auto-correlation matrix, equation (2.14) can also be written as

$$\underline{R}_{yy}^{n)}\ \underline{f}^{(n)} = \underline{\rho}_{yy}^{(n)} \tag{2.17}$$

and this relationship is essential in the development of the Levinson algorithm. (To apply the algorithm therefore when time averaged estimates of the needed correlations are used, the data must be windowed prior to averaging to maintain the even symmetry in the autocorrelation function estimates produce the required structure in the autocorrelation matrix.) Making use of equation (2.15), in the (n+1)-st order version of equation (2.14), and using the relationship of (2.17) to solve for $\underline{\varepsilon}^{(n)}$ and $k^{(n+1)}$ results in

$$\underline{\varepsilon}^{(n)} = -\underline{f}^{(n)} k^{(n+1)} \tag{2.18a}$$

and

$$k^{(n+1)} = \frac{R_{yy}(n+1) - \underline{\rho}^{(n)T} \underline{b}^{(n)}}{R_{yy}(0) - \underline{b}^{(n)T} \underline{r}_{yy}^{(n)}} \tag{2.18b}$$

Therefore, in using equations 2.15 and 2.18 to obtain $\underline{b}^{(n+1)}$ from $\underline{b}^{(n)}$ via the Levinson algorithm only one new piece of information, $k^{(n+1)}$, need be calculated. The denominator of equation (2.18b) can also be recognized from equation (2.8c) as the MMSE for the n-th order AR model $E_2^{(n)}$, and thus there is little concern over the possibility of it being zero. If the n-th order solution produces a perfect prediction (zero MSE) there is no point in trying to find a better prediction by increasing the order to n+1. The evaluation of equation (2.18b) can be further simplified by observing that the MSE also follows a recursion from one order to the next

given by

$$E_2^{(n+1)} = E_2^{(n)} \left[1 - k^{(n+1)^2}\right] \tag{2.19}$$

making it unnecessary to evaluate the denominator at each value of n. (Details of this derivation are omitted here but included in Appendix A in the derivation of the more general multichannel Levinson algorithm.) This relation for the propagation of mean square prediction error also leads to the restriction that $k^{(n+1)}$ must be bounded in magnitude by unity.

2. The Levinson Algorithm For MA Modeling

Next consider the n-th order MA model given by

$$\underline{R}_{u^+u^+}^{(n)} \; \underline{a}^{+(n)} = \underline{r}_{u^+y}^{(n)} \tag{2.20}$$

and again, assume a relationship between the (n+1)-st order and n-th order solutions given by

$$\underline{a}^{+(n+1)} = \begin{bmatrix} \underline{a}^{+(n)} \\ --- \\ 0 \end{bmatrix} + \begin{bmatrix} \underline{\gamma}^{(n+1)} \\ --- \\ g^{(n+1)} \end{bmatrix} \tag{2.21}$$

Notice that in this n-th order problem, $\underline{R}_{u^+u^+}^{(n)}$ is actually a n+1 by n+1 matrix and could be written as $\underline{R}_{uu}^{(n+1)}$. Define

$$\underline{r}_{uu}^{(n+1)} = \begin{bmatrix} R_{uu}(1) \\ \vdots \\ R_{uu}(n+1) \end{bmatrix} ; \quad \underline{\rho}_{uu}^{(n+1)} = \begin{bmatrix} R_{uu}(n+1) \\ \vdots \\ R_{uu}(1) \end{bmatrix} \tag{2.22}$$

Using equations (2,21) and (2,22) in the n+1 order MA model equation it follows that

$$\underline{\gamma}^{(n+1)} = - \underline{f}^{(n+1)} g^{(n+1)} \tag{2.23a}$$

where $\underline{f}^{(n+1)}$ is defined in a manner similar to (2.16) and is comprised of the coefficients that arise in an (n+1)-st order autoregression on the input signal u(k). Therefore to obtain a moving average model relating the system output signal y(k) to the input signal u(k), an autoregressive model for the system input must first be solved. Furthermore,

$$g^{(n+1)} = \frac{R_{uy}(n+1) - \underline{\rho}_{uu}^{(n+1)^T} \underline{a}^{+(n)}}{R_{uu}(0) - \underline{\rho}_{uu}^{(n+1)^T} \underline{f}^{(n+1)}} \tag{2.23b}$$

and the denominator of equation (2.23b) is the MMSE in the (n+1)-st order autoregression on the signal u(k).

It is significant to note that in applying the Levinson algorithm to find a given order AR or MA model, all lower order models along with their MSE's are obtained. Also, intermediate quantities emerge (the $\{k^{(n)}\}$ in the AR

model and the $\{k^{(n)}\}$ and $\{g^{(n)}\}$ in the MA model) which fully characterize the models and could be used as an alternative to the {a(n)} or {b(n)} coefficients. This point will be developed further in subsequent sections.

D. LATTICE FORM AR AND MA MODELS

The Levinson algorithm derived in the previous section can be used to derive lattice structures to implement the MA model and the AR analysis and synthesis models as alternatives to a tapped delay line type of implementation using the coefficients a(n) or b(n) directly. [Refs. 29, 30, 32 and 33]

1. The AR Modeling Lattice Structures

From the relationship between the (n+1)-st and n-th order solutions to the AR modeling problem determined in equations (2.15) and (2.18a) it follows that the transfer function of the prediction error model can be written recursively in order as

$$B^{(n+1)}(z) = B^{(n)}(z) - k^{(n+1)} z^{-n-1} B^{(n)}(z^{-1}) \qquad (2.24)$$

Defining a new transfer function

$$\overline{B}^{(n)}(z) = z^{-n} B^{(n)}(z^{-1}) \qquad (2.25a)$$

equation (2.24) can be written as

$$B^{(n+1)}(z) = B^{(n)}(z) - k^{(n+1)} z^{-1} \overline{B}^{(n)}(z) \qquad (2.25b)$$

and an expression can also be written for $\bar{B}^{(n+1)}(z)$ recursively in order by rewriting equation (2.25a) for order (n+1) and substituting equation (2.24) yielding.

$$\bar{B}^{(n+1)}(z) = z^{-1}\bar{B}^{(n)}(z) - k^{(n+1)}B^{(n)}(z) \qquad (2.25c)$$

As discussed earlier in connection with equation (2.9), $B^{(n)}(z)$ describes the n-th order prediction error model and when its input is the system output Y(z), it produces the n-th order prediction error signal.

$$E^{(n)}(z) = B^{(n)}(z)\, Y(z) \qquad (2.26)$$

In the time domain this signal can be interpreted as the error in predicting y(k) forward in time from a weighted combination of the n past values $\{y(k-1)\cdots y(k-n)\}$. To understand the significance of $\bar{B}^{(n)}(z)$ consider the output signal when this model is excited by Y(z).

$$\bar{E}^{(n)}(z) = \bar{B}^{(n)}(z)Y(z)$$

$$= z^{-n}\left[1 - \sum_{L=1}^{n} \bar{b}^{(n)}(i)z^{+i}\right]Y(z) \qquad (2.27)$$

In the time domain, $\bar{e}^{(n)}(k)$ can be interpreted as the error in predicting y(k-n) backward in time from a weighted combination of the future signals $\{y(k-n+1)\cdots y(k)\}$. These n-th order forward and backward prediction processes at time

k_o are illustrated in Figure 2.4. Henceforth, an overbar will always be used to denote quantaties associated with backward in time predictions.

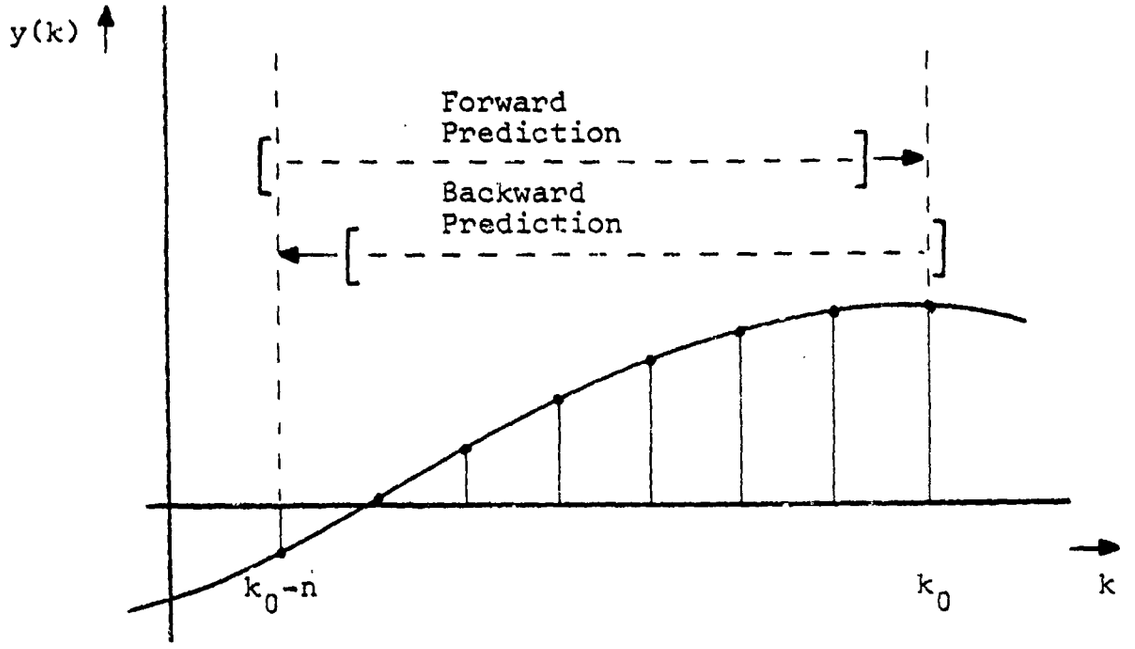


Figure 2.4. Forward and Backward Prediction Error Filtering.

From equations (2.25b) and (2.25c) equations can be written recursively in order for these forward and backward prediction error sequences as:

$$e^{(n+1)}(k) = e^{(n)}(k) - k^{(n+1)}\bar{e}^{(n)}(k-1) \qquad (2.28a)$$

$$\bar{e}^{(n+1)}(k) = \bar{e}^{(n)}(k-1) - k^{(n+1)}e^{(n)}(k) \qquad (2.28b)$$

Noting that the prediction error for a zero-order AR predictor of y(k) (or no predictor at all) is just the signal y(k) itself,

$$e^{(o)}(k) = \bar{e}^{(o)}(k) = y(k) \tag{2.28c}$$

the prediction error filter can be drawn in lattice form as shown in Figure 2.5 for a second order model.

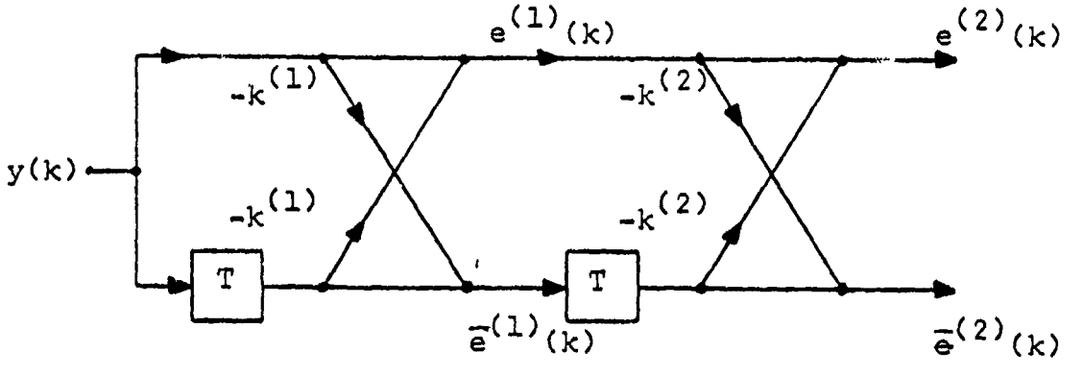


Figure 2.5. Lattice Form Of A Second Order Prediction Error Model.

This structure has many interesting properties, among the most important of which is the successive decoupling property. In going from one order AR model to the next, all of the previously determined transfer function coefficients {b(n)} will generally change. The Levinson algorithm shows however that only one new piece of information is needed to obtain the optimum (n+1)-st order solution from the optimum n-th order solution (see equation (2.24)). In terms of the lattice filter of Figure 2.4 this

means that given the optimum n-th order model in lattice form, one need only add another stage to the structure, setting the coefficient of that stage $k^{(n+1)}$ to minimize the mean square value of $e^{(n+1)}(k)$. Nothing in the first n stages need be changed. The overall high order minimization problem is in this fashion decomposed into a sequence of first order minimizations, one at each lattice stage.

Another important property of the lattice which can be proven and will be of use later is the orthonogalization of the backward prediction error sequence [Ref. 32] which states that

$$\varepsilon\{\bar{e}^{(i)}(k)\bar{e}^{(j)}(k)\} = \begin{cases} 0 & i \neq j \\ E_2^{(i)} & i=j \end{cases} \qquad (2.29)$$

Thus it is seen that a set of orthogonal signals (the backward prediction errors at the various stages) are generated as a by-product of the lattice model.

As a consequence of the successive decoupling property of the lattice, a number of alternatives to equation (2.18b) for determining the lattice coefficients can be found. The most obvious method is to set $k^{(n+1)}$ to explicitly minimize the mean square value of forward prediction error in equation (2.28a) at the (n+1)-st order stage given the best lattice of order n. This is termed the forward method and is denoted by a subscript F on the

lattice coefficients. The resulting solution is given by

$$k_F^{(n+1)} = \frac{\varepsilon\{e^{(n)}(k)\bar{e}^{(n)}(k-1)\}}{\varepsilon\{\bar{e}^{(n)}(k-1)^2\}} \tag{2.30}$$

Alternately, the mean square value of the backward prediction error signal in equation (2.28b) could be minimized to determine the coefficient resulting in the backward method solution given by

$$k_B^{(n+1)} = \frac{\varepsilon\{e^{(n)}(k)\bar{e}^{(n)}(k-1)\}}{\varepsilon\{e^{(n)}(k)^2\}} \tag{2.31}$$

Since, however, the forward and backward prediction error transfer functions are given by $B^{(n)}(z)$ and $z^{-n}B^{(n)}(z^{-1})$, it follows that

$$|B^{(n)}(z)| = |\bar{B}^{(n)}(z)| \tag{2.32}$$

and since they are both driven by the same input, Y(z), the mean square values of both the forward and backward prediction error signals at a given stage are the same making equations (2.30) and (2.31) equivalent. It is also possible to show that they are equivalent to equation (2.18b). Recognizing that the required expectations will eventually have to be estimated by using time averages, these two methods for calculating $k^{(n+1)}$ will not in general be exactly equivalent and it might be preferable to use the arithmetic mean of the mean square values of forward and

backward prediction error as a cost function

$$\tfrac{1}{2}[\varepsilon\{e^{(n+1)}(k)^2 + \bar{e}^{(n+1)}(k)^2\}] \tag{2.33a}$$

This leads to a third method derived by Burg [Ref. 5] in his work on maximum entropy spectral analysis and given by

$$k_{BG}^{(n+1)} = \frac{2\varepsilon\{e^{(n)}(k)\bar{e}^{(n)}(k-1)\}}{\varepsilon\{e^{(n)}(k)^2\}+\varepsilon\{\bar{e}^{(n)}(k-1)^2\}} \tag{2.33b}$$

Notice that $k_{BG}^{(n+1)}$ is the harmonic mean of $k_F^{(n+1)}$ and $k_B^{(n+1)}$. A fourth method due to Itakaura and Saito [Ref. 20] can also be derived which results in

$$k_{IS}^{(n+1)} = \frac{\varepsilon\{e^{(n)}(k)\bar{e}^{(n)}(k-1)\}}{\sqrt{\varepsilon\{e^{(n)}(k)^2\}\varepsilon\{\bar{e}^{(n)}(k-1)^2\}}} \tag{2.34}$$

and $k_{IS}^{(n+1)}$ is simply the geometric mean of the forward and backward coefficients.

Since equation (2.34) is of the form of a normalized correlation k_{IS} will always be bounded by unity in magnitude as required by equation (2.19). Furthermore since

$$|\text{ Harmonic Mean }| \leq |\text{ Geometric Mean }|$$

it follows that $k_{BG}^{(n+1)}$ will be similarly bounded. These bounds are significant since Markel [Ref. 32] has shown that $|k^{(n)}|<1$ is a necessary and sufficient condition to ensure

that the roots of $B^{(n)}(z)$ be within the unit circle guaranteeing the stability of the n-th order all pole model. No such guarantees of model stability exist when the forward or backward solution methods of equations (2.30) and (2.31) are used with the correlation estimates obtained by averaging for finite time intervals.

To determine the AR synthesis model in lattice form it is only necessary to rewrite equation (2.28a) as

$$e^{(n)}(k) = e^{(n+1)}(k) + k^{(n+1)}e^{(n)}(k-1) \qquad (2.35)$$

Together with equations (2.28b) and (2.28c), this describes the structure shown in Figure 2.6 for a second order case and when it is driven by the second order prediction error signal, it will reconstruct y(k) exactly. Thus it implements the transfer function $\frac{1}{B^{(2)}(z)}$ or, in general, when stages are used $\frac{1}{B^{(N)}(z)}$. Recognizing that if the prediction error model is an accurate model of the system denominator polynomial, $e^{(N)}(k)=a(o)u(k)$, this input signal is used in the synthesis model. Because of analogies with transmission lines and wave propagation models, the lattice coefficients have been referred to as reflection coefficients.

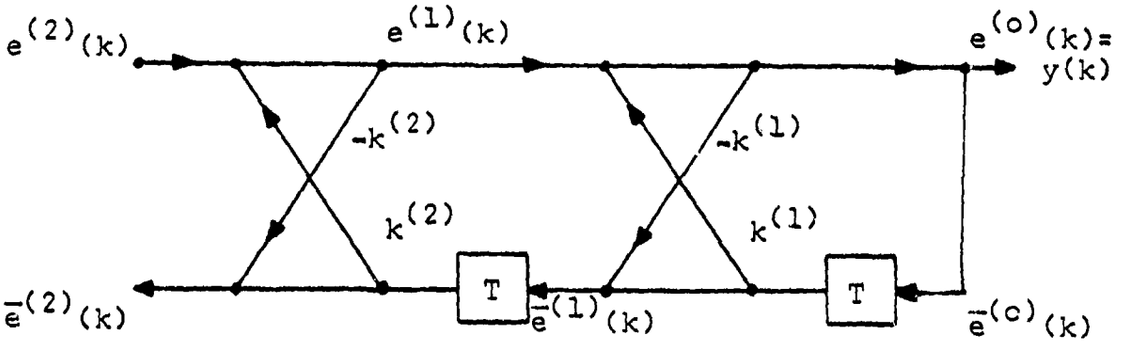


Figure 2.6. Lattice Form Of The All Pole Synthesis Model For The Second Order Case.

2. The MA Modeling Lattice Structure

A similar lattice form is applicable to the MA modeling problem. From equations (2.21) and (2.23a) the transfer function of the MA model can be written recursively in order as

$$A^{(n+1)}(z) = A^{(n)}(z) + g^{(n+1)}\bar{B}^{n+1}(z) \qquad (2.36)$$

where, as discussed in connection with equation (2.23a), $\bar{B}^{n+1}(z)$ is the backward prediction error transfer function for an autoregressive model of the input signal u(k). Multiplying both sides of equation (2.36) by U(z) and transforming into the time domain it follows that

$$\hat{y}^{(n+1)}(k) = \hat{y}^{(n)}(k) + g^{(n+1)}\bar{e}^{n+1}(k) \qquad (2.37)$$

where $\bar{e}^{n+1}(k)$ is the backward prediction error signal from the autoregression on the input signal u(k), and can be obtained by operating a prediction error lattice with u(k) as its input. Then with the additional term in equation (2.37) the lattice form of the MA model can be drawn as shown in Figure 2.7.

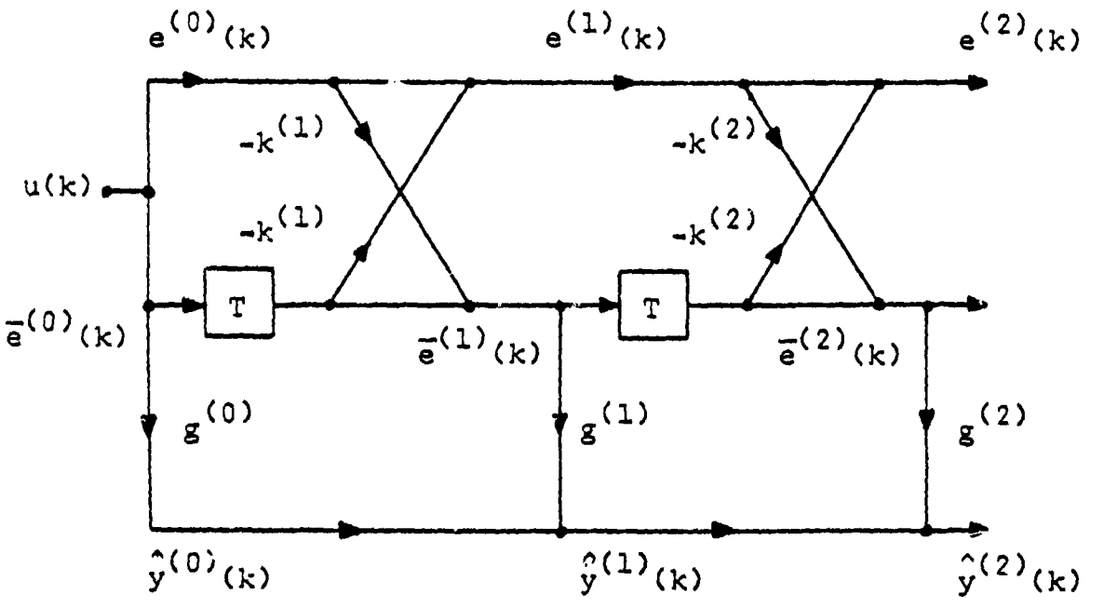


Figure 2.7. Lattice Form Of The MA Model (Second Order Case).

It was stated earlier that the AR prediction error lattice, as a by-product, forms a set of orthogonal or uncorrelated backward prediction error signals from its input. Here in the MA model, these orthogonal signals are linearly combined to form the MA estimate of the system output. If the input signal u(k) is a white process, an

examination of any of the solution methods previously discussed will show that all of the $\{k^{(n)}\}$ lattice coefficients will be zero since the delayed samples of a while process are already orthogonal. Otherwise the $\{k^{(n)}\}$ lattice coefficients will be set to orthogonalize the backward prediction error signals. As a consequence of this, the weighting coefficients $g^{(n)}$ can be set independently of each other; that is $g^{(n)}$ can be set to minimize

$$\varepsilon\{e_o^{(n)}(k)^2\} = \varepsilon\{[y(k) - \hat{y}^{(n)}(k)]^2\} \qquad (2.38)$$

given the best prediction of order n-1, $\hat{y}^{(n)}(k)$. This results in an alternative expression to equation (2.23b) for $g^{(n)}$ given by

$$g^{(n)} = \frac{\varepsilon\{e_o^{(n-1)}(k)\ \bar{e}^{(n)}(k)\}}{\varepsilon\{\bar{e}^{(n)}(k)^2\}} = \frac{\varepsilon\{y(k)\ \bar{e}^{(n)}(k)\}}{\varepsilon\{\bar{e}^{(n)}(k)^2\}} \qquad (2.39)$$

Here $e_o^{(n)}(k)$ is the error between the system output and its n-th order MA estimate.

E. MULTICHANNEL AR AND MA MODELING

Both the AR and MA modeling problems previously discussed, as well as their solution via the Levinson recursion and lattice filter methods, can be generalized to the multichannel case by replacing the various signals with signal vectors and replacing the weighting coefficients with appropriately dimensioned matrices of coefficients.

A discussion of this appears in Robinson [Ref. 45]. The equations that describe the AR and MA models and their MMSE solutions are repeated here for convenience.

$$\underline{\text{AR}} \qquad \hat{y}(k) = \sum_{i=1}^{N} b(i)\, y(k-i) \tag{2.40a}$$

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(-1) & \cdots & R_{yy}(1-N) \\ R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(2-N) \\ \vdots & \vdots & & \vdots \\ R_{yy}(N-1) & R_{yy}(N-2) & \cdots & R_{yy}(0) \end{bmatrix} \begin{bmatrix} b(1) \\ b(2) \\ \vdots \\ b(N) \end{bmatrix} = \begin{bmatrix} R_{yy}(1) \\ R_{yy}(2) \\ \vdots \\ R_{yy}(N) \end{bmatrix} \tag{2.40b}$$

$$\underline{\text{MA}} \qquad \hat{y}(k) = \sum_{i=0}^{N} a(i)\, u(k-i) \tag{2.41a}$$

$$\begin{bmatrix} R_{uu}(0) & R_{uu}(-1) & \cdots & R_{uu}(-N) \\ R_{uu}(1) & R_{uu}(0) & \cdots & R_{uu}(1-N) \\ \vdots & \vdots & & \vdots \\ R_{uu}(N) & R_{uu}(N-1) & \cdots & R_{uu}(0) \end{bmatrix} \begin{bmatrix} a(0) \\ a(1) \\ \vdots \\ a(N) \end{bmatrix} = \begin{bmatrix} R_{uy}(0) \\ R_{uy}(1) \\ \vdots \\ R_{uy}(N) \end{bmatrix} \tag{2.41b}$$

In a multichannel generalization, y(k) becomes a vector of Q_0 output signals, u(k) becomes a vector of Q_i input signals, b(i) becomes a square matrix of $Q_0 \times Q_0$ coefficients and a(i) becomes a $Q_0 \times Q_i$ matrix of coefficients so that the N-th

order multichannel model equations can be written as

$$\underline{AR} \qquad \hat{\underline{y}}(k) = \sum_{n=1}^{N} \underline{b}(n)\, \underline{y}(k-n) \tag{2.42a}$$

$$\underline{MA} \qquad \hat{\underline{y}}(k) = \sum_{n=0}^{N} \underline{a}(n)\, \underline{u}(k-n) \tag{2.42b}$$

The equations for the MMSE solutions (2.40b) and (2.41b) generalize directly as well by replacing each correlation coefficient $R_{vw}(n)$ by matrices of correlation coefficients given by

$$\underline{R}_{vw}(n) = \mathcal{E}\{\underline{v}(k)\, \underline{w}(k+n)^T\} \tag{2.43}$$

where $\underline{v}(k)$ and $\underline{w}(k)$ are signal vectors. This causes the overall correlation matrices to take on a block Toeplitz structure. The transfer function relationships of the AR prediction error model and the MA model take the form of matrix polynomials

$$\underline{B}(z) = \underline{b}(1)\, z^{-1} + \cdots + \underline{b}(N)\, z^{-N} \tag{2.44a}$$

$$\underline{A}(z) = \underline{a}(0) + \underline{a}(1)\, z^{-1} + \cdots + \underline{a}(N)\, z^{-N} \tag{2.44b}$$

so that

$$\underline{E}(z) = [\underline{I} - \underline{B}(z)]\, \underline{Y}(z) \tag{2.44c}$$

$$\hat{\underline{Y}}(z) = \underline{A}(z)\ \underline{U}(z) \qquad (2.44d)$$

where $\underline{E}(z)$ is the transform of the multichannel AR prediction error vector $\underline{e}(k) = \underline{y}(k) - \hat{\underline{y}}(k)$. Alternately, equations (2.44a) and (2.44b) can be written as single matrices whose entries are polynomials in z rather than scalars. $\underline{B}(z)$ is of necessity a $Q_0 \times Q_0$ square matrix polynomial while the dimensions of $\underline{A}(z)$ ($Q_0 \times Q_i$) depend upon the number of inputs and outputs which need not be the same.

In the single channel AR problem, B(z) provides the transfer function of the prediction error, or inverse filter, and must be inverted to obtain the all pole synthesis filter. The stability of the synthesis model therefore depends upon the roots of this polynomial. The matrix polynomial $[\underline{I}-\underline{B}(z)]$ in the multichannel AR problem is, in like fashion, an inverse filter representation and must be inverted to obtain the synthesis model. This inversion of a matrix with polynomial entries is defined in the same manner as the inversion of a square matrix with scaler entries. To see what this inverse matrix polynomial looks like consider as an example a two channel autoregression with a prediction error filter given by

$$[\underline{I}-\underline{B}(z)] = \begin{bmatrix} 1 - b_{11}(z) & -b_{12}(z) \\ -b_{21}(z) & 1 - b_{22}(z) \end{bmatrix}$$

Applying Cramer's rule, the inverse matrix polynomial is written as

$$[\underline{I}-\underline{B}(z)]^{-1} = \frac{1}{\det[\underline{I}-\underline{B}(z)]} \begin{bmatrix} 1-b_{22}(z) & b_{21}(z) \\ b_{12}(z) & 1-b_{11}(z) \end{bmatrix}$$

and it is apparent that the stability of the multichannel synthesis model is dependent upon the locations of the zeros of the polynomial $\det[\underline{I}-\underline{B}(z)]$.

This straightforward generalization of the AR and MA models is what has customarily been used in the literature to develop the multichannel models and similarly derived generalizations of the Levinson algorithm to solve them recursively in order are available as well. [Refs. 57 and 61] The multichannel AR and MA modeling problems and their solutions via the Levinson algorithm can however be recast as shown in Appendix A to make them compatible with the form of other linear and nonlinear modeling problems. To avoid confusion later in the application of the results, the derivations in Appendix A have been carried out in a generic form with x and d used to represent some of the signals and coefficients respectively. The symbols u, y, a and b have been reserved to denote system input, output and weighting coefficients.

Equations (A.7) and (A.26) provide the MMSE solutions to the multichannel AR and MA modeling problems in forms

different than (although entirely equivalent to) those resulting from the straightforward generalizations of equations (2.40b) and (2.41b). The multichannel generalization of the Levinson algorithm derived in Appendix A can, with one exception, be seen as a matrix algebra generalization of the single channel algorithm and as, one would suspect, the single channel algorithm results as a special case of Appendix A. The one exception is that in the multichannel case, the n-th order forward and backward prediction error models are not simply related to one another. The single channel AR backward predictor is given by $z^{-n}B^{(n)}(z^{-1})$ but in the multichannel case the backward prediction is not $z^{-n}[\underline{I}-\underline{\mathbf{D}}^{(n)}(z^{-1})]$.

Because of this, two reflection coefficient matrices $\underline{K}$ and $\overline{\underline{K}}$ are required at each stage in the recursion to relate the n-th and (n+1)-st order solutions rather than just one as in the single channel problem. Also, in the single channel case, the fact that $|B^{(n)}(z)| = |\overline{B}^{(n)}(z)|$ and therefore $\varepsilon\{e^{(n)}(k)^2\}=\varepsilon\{\overline{e}^{(n)}(k)^2\}$ made possible the derivation of the Burg algorithm and the Itakaura-Saito algorithm, which ensured the magnitude of the reflection coefficient was bounded by unity and that equation (2.19) would result in nonnegative values of MSE. In the multichannel algorithm however, the forward and backward prediction error covariance matrices and their traces are not the same (except for $\underline{P}^{(0)}$ and $\overline{\underline{P}}^{(0)}$ and as a result, straightforward generalizations of the Burg and Itakaura-Saito algorithms are not possible.

Consequently with correlation estimates obtained by averaging over finite time intervals, there are no guarantees that equations (A.18a) and (A.19b) maintain the positive definiteness of the prediction error covariance matrices.

Multichannel generalizations of Burg's algorithm due to Nuttal [Refs. 40 and 41], Morf [Ref. 39], and Strand [Ref. 50] which guarantee the positivity of the covariance matrices are available but are not explored here because of their complexity and because they would take the discussion too far afield.

The relations of equations (A.20) and (A.30) which are repeated here for convenience permit the construction of the multichannel AR analysis and synthesis lattice structures and the MA lattice structure. For the multichannel AR model,

$$\underline{e}^{(n+1)}(k) = \underline{e}^{(n)}(k) - \underline{K}^{(n+1)^T}\underline{\bar{e}}^{(n)}(k-1) \qquad (2.45a)$$

$$\underline{\bar{e}}^{(n+1)}(k) = \underline{\bar{e}}^{(n)}(k-1) - \underline{\bar{K}}^{(n+1)^T}\underline{e}^{(n)}(k) \qquad (2.45b)$$

$$\underline{e}^{(0)}(k) = \underline{\bar{e}}^{(0)}(k) = \underline{x}(k) \qquad (2.45c)$$

and the corresponding prediction error lattice is shown in Figure 2.8.

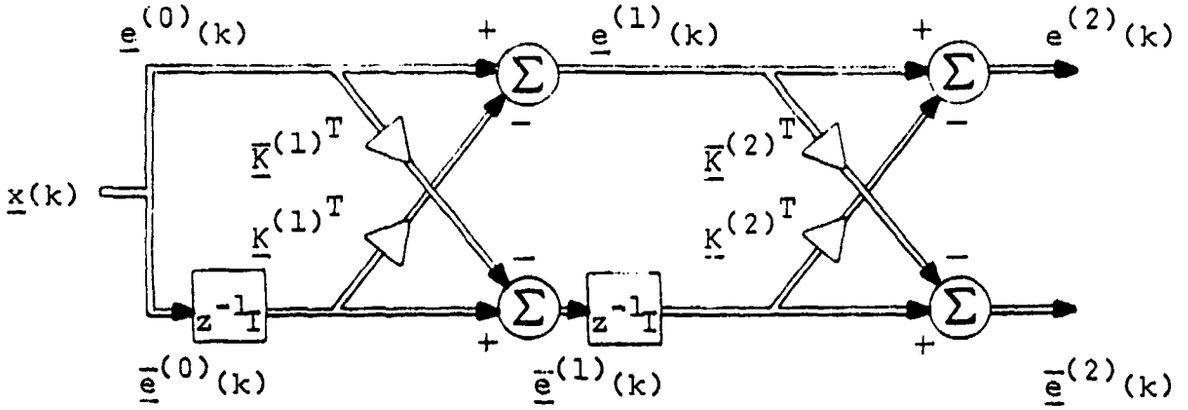


Figure 2.8. Multichannel AR prediction error lattice structure for a second order model. All signal paths are vector paths and summations are vector summations. The multiplications indicate premultiplication of the signal vector by the specified coefficient matrix.

To obatin the multichannel AR synthesis lattice, equation (2.45a) can simply be rewritten as

$$\underline{e}^{(n)}(k) = \underline{e}^{(n+1)}(k) + \underline{K}^{(n+1)T}\bar{\underline{e}}^{(n)}(k-1) \qquad (2.46)$$

resulting in the structure shown in Figure 2.9.

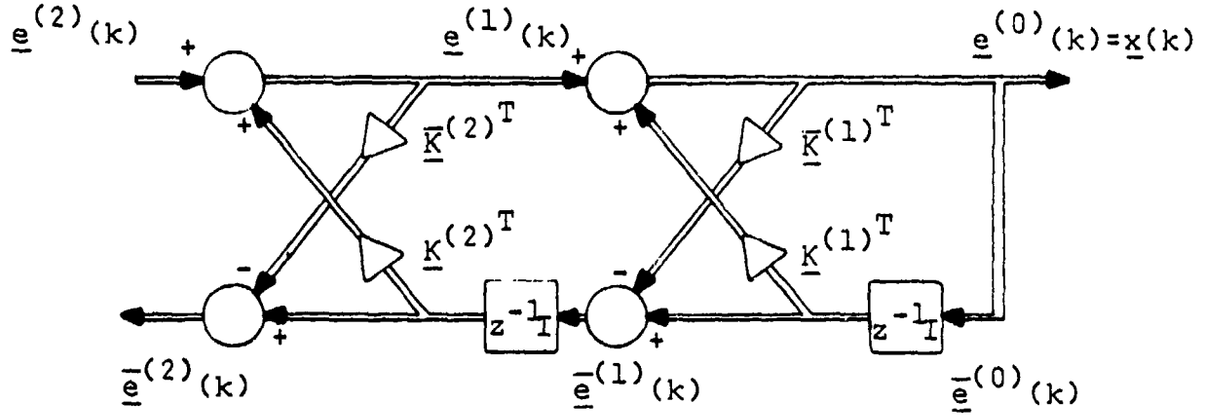


Figure 2.9. Multichannel AR synthesis lattice structure for a second order model.

For the multichannel MA model, equations (2.45) and

$$\hat{\underline{y}}^{(n+1)}(k) = \hat{\underline{y}}^{(n)}(k) + \underline{G}^{(n+1)^T}\, \underline{\bar{e}}^{(n+1)}(k) \tag{2.47}$$

describe the lattice structure shown in Figure 2.10.

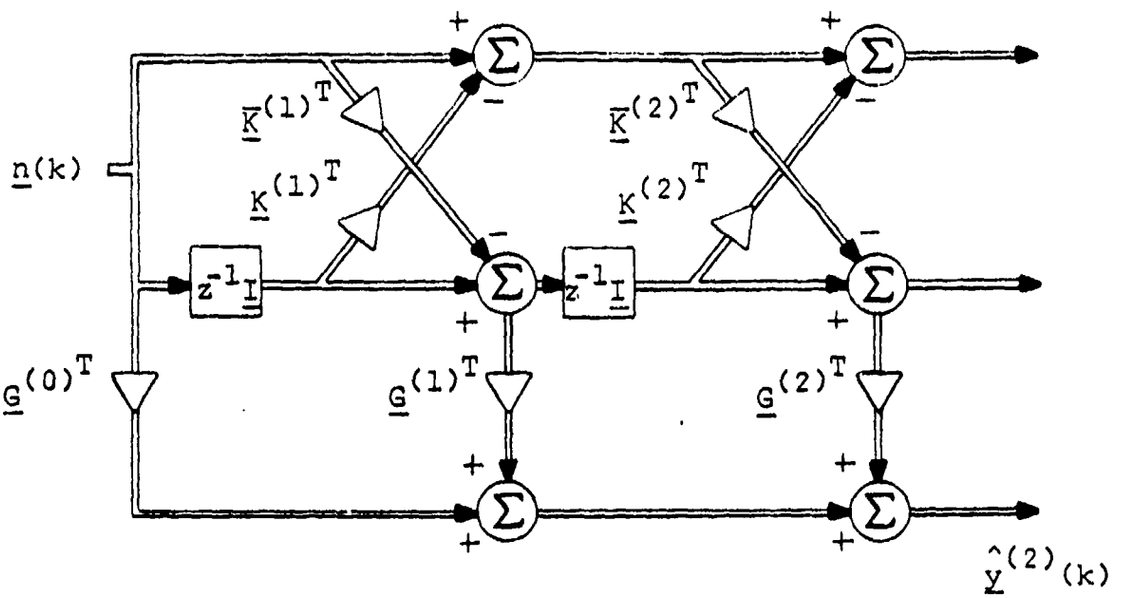


Figure 2.10. Multichannel MA lattice structure for a second order model.

As in the single channel case, the multichannel prediction error lattice exhibits the successive decoupling property and orthogonalizes the backward prediction errors at the various stages so that

$$\mathcal{E}\{\underline{\bar{e}}^{(i)}(k)\, \underline{\bar{e}}^{(j)}(k)^T\} = \begin{cases} \underline{0} & i \neq j \\ \underline{P}^{(i)} & i = j \end{cases} \tag{2.48}$$

As a consequence of the successive decoupling, the forward and backward reflection coefficient matrices at the (n+1)-st stage can be set to minimize the trace of the forward and backward prediction error covariance matrices respectively, given the best lattice of order n. This provides alternatives to equations (A.17) for calculating $\underline{K}$ and $\overline{\underline{K}}$ and also generalizes the forward and backward single channel solutions discussed previously, resulting in

$$\underline{K}^{(n+1)} = \overline{\underline{P}}^{(n)^{-1}} \underline{\Delta}^{(n)^T} \tag{2.49a}$$

$$\overline{\underline{K}}^{(n+1)} = \underline{P}^{(n)^{-1}} \underline{\Delta}^{(n)} \tag{2.49b}$$

where

$$\underline{\Delta}^{(n)} = \varepsilon\{\underline{e}^{(n)}(k)\ \overline{\underline{e}}^{(n)}(k-1)^T\} \tag{2.49c}$$

It is also possible to show that these relationships are entirely equivalent to equation (A.17). In the multichannel MA lattice, the orthogonality of the backward prediction error signals also allows the $\underline{G}$ matrices to be calculated in succession providing an alternative to equation (A.28b) and generalizing the single channel solution given by equation (2.39). Setting $G^{(n)}$ to minimize the trace of the error covariance matrix

$$\underline{P}_0^{(n)} = \varepsilon\{\underline{e}_0^{(n)}(k)\ \underline{e}_0^{(n)}(k)^T\} \tag{2.50a}$$

where

$$\underline{e}_0^{(n)}(k) = \underline{y}(k) - \hat{\underline{y}}^{(n)}(k) \tag{2.50b}$$

results in a solution given by

$$\underline{G}^{(n)} = \underline{P}^{(n)^{-1}} \varepsilon\{\underline{\bar{e}}^{(n)}(k)\ \underline{y}(k)^T\} \tag{2.50c}$$

Another important characteristic of the lattice solutions to the AR and MA modeling problems given by equations (2.49) and (2.50) and their single channel counterparts is that they do not impose any requirements to window the data when finite time averages are used to estimate correlations. The autocorrelation function of a signal is inherently an even function so that $R_{vv}(n) = R_{vv}(-n)$. This fact is responsible for much of the special structure of the correlation matrices that appear in the model solution equations, and was also used in the derivation of the Levinson algorithm. In estimating the autocorrelation function via time averaging over a finite interval, a window function that is nonzero over only a given interval must be applied to the data to retain this even symmetry property in the estimate. If the data is not set to zero outside a given interval, end effects will destroy the symmetry so that $\hat{R}_{vv}(n) \neq \hat{R}_{vv}(-n)$ as depicted in Figure 2.11.

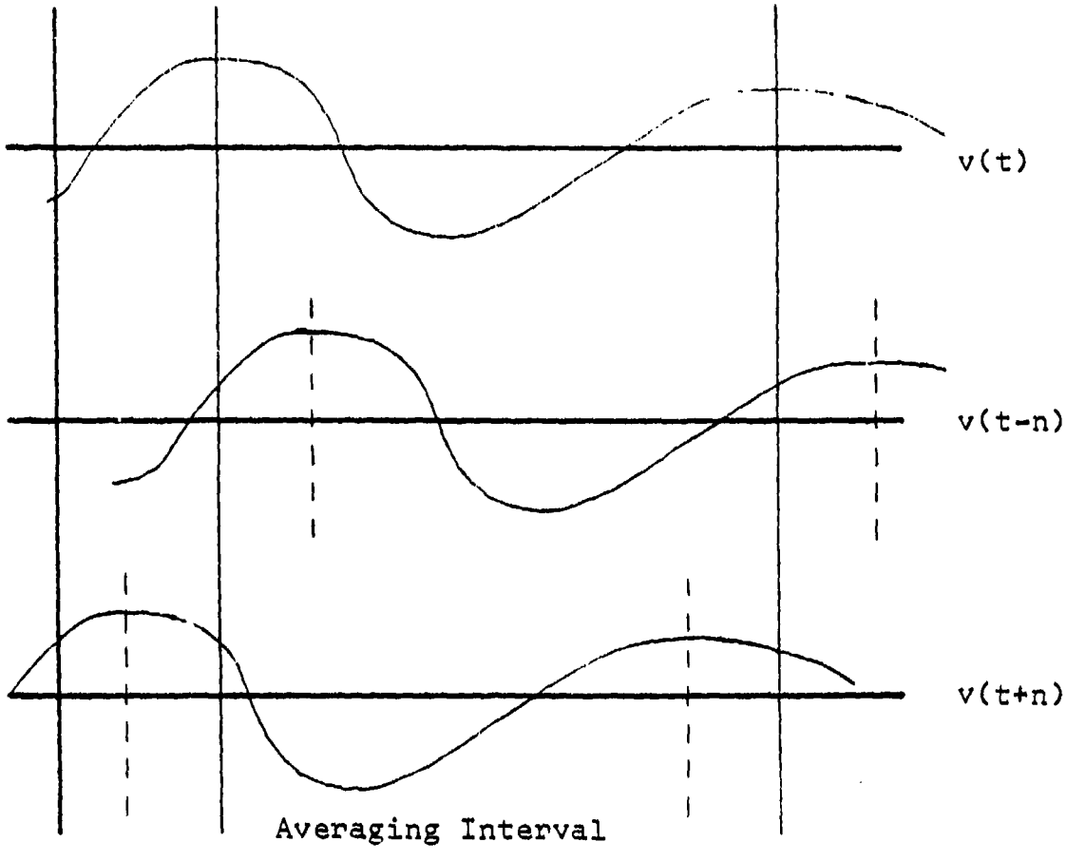


Figure 2.11. Time averaging to estimate correlations without windowing.

In the lattice solutions of equation (2.49) however, there is no requirement to make such an artificial assumption about the data (that it is zero outside some interval).

These properties of the lattice solution methods were responsible for their initial use by Burg in his work on maximum entropy spectral analysis [Ref. 5] and have continued to generate interest in the application of lattice methods to other types of modeling problems.

F. ADAPTIVE MODELING

The LMS adaptive algorithm provides a well known alternative method for obtaining the solution to the AR or MA modeling problems which does not require the estimation of correlations or the inversion of a matrix [Refs. 58, 59 and 60]. This algorithm updates an estimate of the model solution vector at each time instant by an amount proportional to the negative of the instantaneous gradient of the cost function; i.e., in a MA model,

$$\underline{a}^{+}(k+1) = \underline{a}^{+}(k) - \mu \, \underline{\nabla}^{+}(k) \tag{2.51a}$$

where μ is a proportionality constant or adaptive gain. Since the actual gradient is usually not known, it is approximated by using the square of a single sample of the error as an estimate of the MSE so that

$$\hat{\underline{\nabla}}^{+}(k) = \frac{\partial e(k)^2}{\partial \underline{a}^{+}} \bigg|_{\underline{a}^{+}=\underline{a}^{+}(k)} = -2 \, \underline{u}^{+}(k) \, e(k) \tag{2.51b}$$

and

$$\underline{a}^{+}(k+1) = \underline{a}^{+}(k) + 2\mu \, \underline{u}^{+}(k) \, e(k) \tag{2.51c}$$

In each of the models considered here, the cost function (MSE or trace $\underline{P}$) is a quadratic function of the model weights and defines a concave hyperparaboled with a unique minimum. The functioning of the LMS algorithm under these conditions

can easily be understood by considering the scalar case of equation (2.51) illustrated in Figure 2.12.

As this illustration shows, the algorithm can actually diverge for too large a value of adaptive gain. The rate of convergence is also dependent on the size of the adaptive gain. Widrow has shown that for stability, the gain must be set so that

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{2.52a}$$

While, in the mean, the weight vector converges with an exponential time constant of

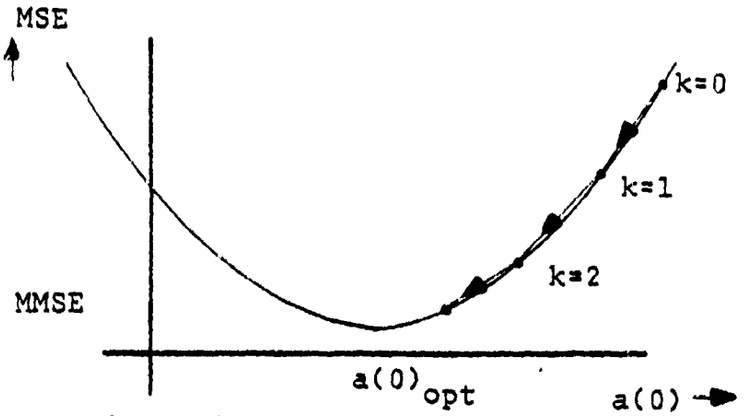
$$\tau \cong \frac{1}{2\mu\lambda_{min}} \tag{2.52b}$$

where λ_{min} and λ_{max} are the smallest and largest eigenvalues of the input autocorrelation matrix $\underline{R}_{u^+u^+}$. From the standpoint of stability, μ should be made relatively small but for rapid convergence, equation (2.52b) dictates that it should be large. Setting

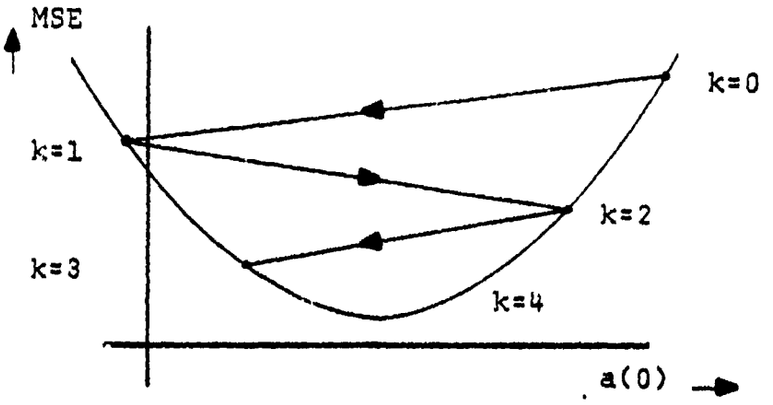
$$\mu = \frac{\alpha}{\lambda_{max}} \tag{2.53a}$$

where α is a normalized gain and $0 < \alpha < 1$, equation (2.52b) becomes

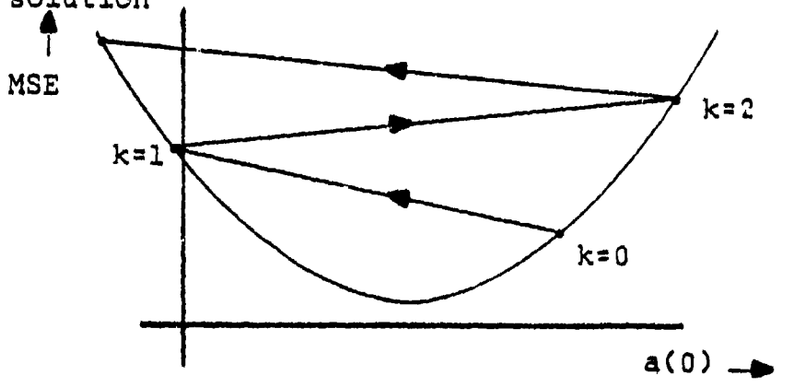
$$\tau \cong \frac{1}{2\alpha} \frac{\lambda_{max}}{\lambda_{min}} \tag{2.53b}$$



a) Small adaptive gain; steady convergence toward the solution.



b) Intermediate adaptive gain; oscilatory convergence toward the solution



c) Large adaptive gain; divergence away from the solution

Figure 2.12. Behavior of the LMS adaptive algorithm for various adaptive gains.

and for a wide disparity between the largest and smallest eigenvalues ($\lambda_{min} << \lambda_{max}$), convergence will be quite slow. This consideration becomes increasingly important when high order model solutions are obtained adaptively since the dimensonality of the input autocorrelation matrix will be high and the possibility of a wide eigenvalue disparity greater.

These same adaptive techniques have been applied by Griffiths [Refs. 16, 17, 18 and 31], to the AR and MA lattice filter structures derived in the last section. The key difference between the conventional adaptive filter and the adaptive lattice is that in the lattice, the adaptation is carried out on a stage by stage basis for each of the reflection coefficient matrices while in the more conventional approach, the entire weight vector is adapted. It has already been established that the lattice structure makes the model solutions recursive in order. Implementing the lattice adaptivity makes the solution recursive in time as well since the estimate of the solution at each instant is dependent upon prior estimates of the solution.

The conventional adaptive filter algorithm forms an error signal as the difference between some desired signal and its estimate; i.e.

$$e(k) = y(k) - \underline{a}^{+}(k)^{T}\underline{u}^{+}(k) \qquad (2.54)$$

where y(k) is the desired signal, $\underline{a}^+$ is the weight vector and $\underline{u}^+(k)$ is the input vector, and the time update for the weight vector is given by equation (2.51c). To derive the adaptive AR lattice consider equations (2.45) for a single stage. The lattice in general has vecto error and desired signals and coefficient matrices as opposed to scalar error and desired signals and a coefficient vector in equation (2.54) but such a generalization is straightforward. Comparing equation (2.45a) to (2.54) it is clear that:

1) $\underline{e}^{(n+1)}(k)$ is analogous to the error signal;

2) $\underline{e}^{(n)}(k)$ is analogous to the desired signal;

3) $\underline{\bar{e}}^{(n)}(k-1)$ is analogous to the input signal vector.

Using the trace of $\underline{P}^{(n+1)}$ as a cost function and applying a LMS adaptive algorithm to determine the forward reflection coefficient matrix it follows that

$$\underline{K}^{(n+1)}(k+1) = \underline{K}^{(n+1)}(k) + 2\mu^{(n+1)}\underline{\bar{e}}^{(n)}(k-1)\underline{e}^{(n+1)}(k)^T \tag{2.55a}$$

With these analogies, equations (2.51c) and (2.55a) are seen to be virtually identical with the exception that (2.51c) uses a scalar error to adapt a weight vector and (2.55a) uses a vector error signal to adapt a coefficient matrix.

Proceeding in a similar fashion with equation 2.45b it is clear that:

1) $\underline{\bar{e}}^{(n+1)}(k)$ is analogous to the error signal;

2) $\underline{\bar{e}}^{(n)}(k-1)$ is analogous to the desired signal;

3) $\underline{e}^{(n)}(k)$ is analogous to the input signal vector.

With the trace of $\underline{P}^{(n+1)}$ as a cost function, the time update relation for the backward reflection coefficient matrix is

$$\underline{K}^{(n+1)}(k+1) = \underline{K}^{(n+1)}(k) + 2\bar{\mu}^{(n+1)}\ \underline{e}^{(n)}(k)\ \underline{\bar{e}}^{(n+1)}(k)^T \tag{2.55b}$$

For a MA lattice, equation (2.47) must also be considered. Multiplying both sides of (2.47) by minus one and adding $\underline{y}(k)$ results in

$$\underline{e}_0^{(n+1)}(k) = \underline{e}_0^{(n)}(k) - \underline{G}^{(n+1)^T}\underline{\bar{e}}^{(n+1)}(k) \tag{2.56}$$

where $\underline{e}_0^{(n+1)}(k)$ is defined as in equation (2.50). It is evident that:

1) $\underline{e}_0^{(n+1)}(k)$ is analogous to the error signal;

2) $\underline{e}_0^{(n)}(k)$ is analogous to the desired signal;

3) $\underline{\bar{e}}^{(n+1)}(k)$ is analogous to the input signal vector.

With the trace of $\underline{P}_0^{(n+1)}$ as a cost function, the time update relation for $\underline{G}^{(n+1)}$ is given by

$$\underline{G}^{(n+1)}(k+1) = \underline{G}^{(n+1)}(k) + 2\mu_g^{(n+1)}\ \underline{\bar{e}}^{(n+1)}(k)\ \underline{e}_0^{(n+1)}(k)^T \tag{2.57}$$

It is significant to note that three different adaptive gains have been used in equations (2.55) and (2.57) and that the gains have been superscripted indicating that they vary from one lattice stage to the next. For stability considerations the adaptive gain used in the LMS algorithm must satisfy equation (2.52a) and therefore is related to the largest eigenvalue of the input autocorrelation matrix by equation (2.53a). In developing the time update relations for the lattice coefficients, three different input signals were used and these inputs also differ from one lattice stage to the next. Indeed, even for the case where the input $\underline{x}(k)$ to the lattice structure is stationary, the inputs to all lattice stages except the first are nonstationary since these inputs are outputs of other lattice stages. This fact indicates that time varying adaptive gains are appropriate as well in equations (2.55) and (2.57). Equation (2.53a) is of no direct usefulness however in setting the adaptive gains since the time varying eigenvalues are not known. Recognizing that the largest eigenvalue is always less than the trace of the input autocorrelation matrix (which is a measure of the power in the input signal vector) the gains can be set as

$$\mu^{(n+1)}(k) = \frac{\alpha}{\sigma_{n+1}(k)^2} \qquad (2.58a)$$

$$\bar{\mu}^{(n+1)}(k) = \frac{\alpha}{\bar{\sigma}_{n+1}(k)^2} \tag{2.58b}$$

$$\mu_g^{(n+1)}(k) = \frac{\alpha}{\gamma_{n+1}(k)^2} \tag{2.58c}$$

where $\sigma_{n+1}(k)^2$, $\bar{\sigma}_{n+1}(k)^2$ and $\gamma_{n+1}(k)^2$ are estimates of the power in the three input signal vectors and α is a normalized adaptive step size with $0 < \alpha < 1$. A method that has commonly been applied to obtain these estimates is to employ a first order low pass filter so that

$$\sigma_{n+1}(k+1)^2 = [1-\alpha]\sigma_{n+1}(k)^2 + \alpha \underline{e}^{(n)}(k-1)^T \underline{e}^{(n)}(k-1) \tag{2.59a}$$

$$\bar{\sigma}_{n+1}(k+1)^2 = [1-\alpha]\bar{\sigma}_{n+1}(k)^2 + \alpha \underline{e}^{(n)}(k)^T \underline{e}^{(n)}(k) \tag{2.59b}$$

$$\gamma_{n+1}(k+1)^2 = [1-\alpha]\gamma_{n+1}(k)^2 + \alpha \underline{e}_0^{(n)}(k)^T \underline{e}_0^{(n)}(k) \tag{2.59c}$$

Taken together, equations (2.55), (2.57), (2.58) and (2.59) define the adaptive solutions for the AR and MA multichannel lattice models.

To understand the potential advantage offered by the adaptive lattice form, recognize that while the conventional approach solves a high order minimization problem by adapting all the coefficients at once, the lattice breaks the problem down into a succession of lower order minimizations at each stage and solves these lower order problems adaptively. The

dimensionality of the input autocorrelation matrices at each lattice stage in general is significantly less than that of the large input autocorrelation matrix in the conventional adaptive algorithm and consequently it is hoped that the possibility of a large eigenvalue disparity with its attendant slow convergence is reduced.

This advantage is most evident in a single channel adaptive lattice where the inputs at each stage are single signals and their corresponding autocorrelation matrices are 1x1 in dimension. In this case the ratio of smallest to largest eigenvalues is unity and the convergence of each stage is quite rapid while the convergence of the overall model is independent of the eigenvalue ratio for the overall higher dimension input autocorrelation matrix. This has been demonstrated by Satorius [Refs. 46 and 47] who has shown that the single channel adaptive lattice converges much more rapidly than the corresponding conventional adaptive filter, and does so independently of the eigenvalue ratio on the overall input channel autocorrelation matrix.

Furthermore in a single channel adaptive lattice, the time update relations are simplified by the fact that the forward and backward reflection coefficients are the same. Using the average of the mean square values of backward and forward prediction errors as a cost function and applying an adaptive algorithm it follows that

$$K^{(n+1)}(k+1)=K^{(n+1)}(k)+\frac{\alpha}{\sigma_{n+1}(k)^2}[e^{(n)}(k)\bar{e}^{(n+1)}(k)$$

$$+\bar{e}^{(n)}(k-1)e^{(n+1)}(k)] \qquad (2.60a)$$

where

$$\sigma_{n+1}(k+1)^2=[1-\alpha]\sigma_{n+1}(k)^2+\frac{\alpha}{2}[e^{(n)}(k)^2+\bar{e}^{(n)}(k-1)^2] \qquad (2.60b)$$

The very nature of the lattice structure however with the output of one stage providing an input to the next stage, greatly complicates the analysis of the convergence properties of the adaptive lattice model. Even when $\underline{x}(k)$, the input to the lattice, is stationary, inputs to all stages except the first are nonstationary. An approximate analysis of convergence and stability on a stage by stage basis is possible if it is assumed that all prior stages have converged and are providing stationary inputs to the stage under investigation. With this assumption, the adaptive solution for the $\underline{K}^{(n+1)}$, $\underline{\bar{K}}^{(n+1)}$ and $\underline{G}^{(n+1)}$ matrices are obtained from the operation of three independent LMS algorithms as shown earlier, with inputs given by $\underline{\bar{e}}^{(n)}(k-1)$, $\underline{e}^{(n)}(k)$ and $\underline{\bar{e}}^{(n+1)}(k)$, respectively. Stability limits on the adaptive gains used in the stage and the convergence properties of the stage are then determined by the eigenvalues of the $\underline{\bar{P}}^{(n)}$, $\underline{P}^{(n)}$ and $\underline{\bar{P}}^{(n+1)}$ matrices. A more exact analysis of the properties of the adaptive lattice that

considers the nonstationary character of the inputs to the second and subsequent stages is not currently available.

III. ARMA MODELING

One of the most serious disadvantages of either AR or MA modeling is the fact that to adequately represent even simple linear systems, both methods may require a large number of parameters (a high order model). This problem arises since, from a transfer function standpoint, AR and MA models attempt to model the system using only poles or only zeros, in spite of the fact that the physical system may have both zeros and poles. While modeling the effects of a zero with a number of poles and visa versa can be analytically justified as shown in the previous chapter, it makes far more sense (both from the viewpoint of model accuracy and efficient use of model parameters) to let the model represent the system as it really is with both zeros and poles if this is at all possible. The ARMA model is a generalization of the AR and MA models and accomplishes exactly this, representing the system in rational transfer function form.

It is worth noting that the titles of all pole and all zero modeling that have been associated with AR and MA modeling are misnomers. Both have equal numbers of zeros and poles. In the AR model however, all the zeros occur at the origin of the z-plane as do the poles of a MA model. The ARMA model removes these constraints.

After a brief discussion of two alternate ARMA modeling methods due to Shanks and Prony, the equation error formulation for ARMA modeling is developed and the new results presented. Model transition formulas relating the ARMA model to the MA and AR models are developed and the input signal requirements of the modeling process explored. It is shown that after suitable modification, the Levinson algorithm can be applied to solve the ARMA modeling problem recursively in order and lattice solution methods are also developed for both a batch processing and an adaptive model solution. The results of experimental simulations of both of these modeling solution methods are presented and discussed, and comparisons are made with conventional means of ARMA modeling using the equation error formulation. Finally it is also shown that the lattice solution methods can be generalized to solve for the multichannel ARMA model with arbitrary numbers of inputs and outputs.

A. LINEAR ARMA MODELING AND ITS RELATION TO AR AND MA MODELING

The ARMA model for linear systems assumes the current value of the output of the system is given by a weighted combination of present and past values of the input and past values of the output. In terms of the discussions of Chapter I, F_{30} is assumed to be zero and F_{10} and F_{20} take on the following forms

$$F_{10}[u(k)] = \sum_{n=0}^{M} a(n)\, u(k-n) \qquad (3.1a)$$

$$F_{20}[y(k-1)] = \sum_{n=1}^{N} b(n)\, y(k-n) \qquad (3.1b)$$

leading to a transfer function representation for the system given by

$$H(z) = \frac{\sum_{n=0}^{M} a(n) z^{-n}}{1 - \sum_{n=1}^{N} b(n) z^{-n}} = \frac{A(z)}{B(z)} \qquad (3.1c)$$

A number of methods exist for finding the model coefficients $\{a(n)\}$ and $\{b(n)\}$. As stated in Chapter I, a MMSE solution via the direct form modeling approach requires the solution of a system of highly nonlinear equations and in general is untractable. An alternative is to first obtain an estimate of the denominator polynomial $B(z)$ by some means such as AR modeling and then using this in the system shown in Figure 3.1, estimate the numerator polynomial $A(z)$ by setting its coefficients to minimize the mean square value of the error. This method was first explored by Shanks. [Ref. 49]

Another alternative is to apply the Prony method [Refs. 8, 52 and 56] derived in Appendix E which obtains the model parameters by matching the impulse response of the system and model over the first $N+M+1$ sample intervals. Both of these techniques share a common characteristic. They both start by independently estimating the denominator coefficients (or model poles) and then, given this estimate, solve for

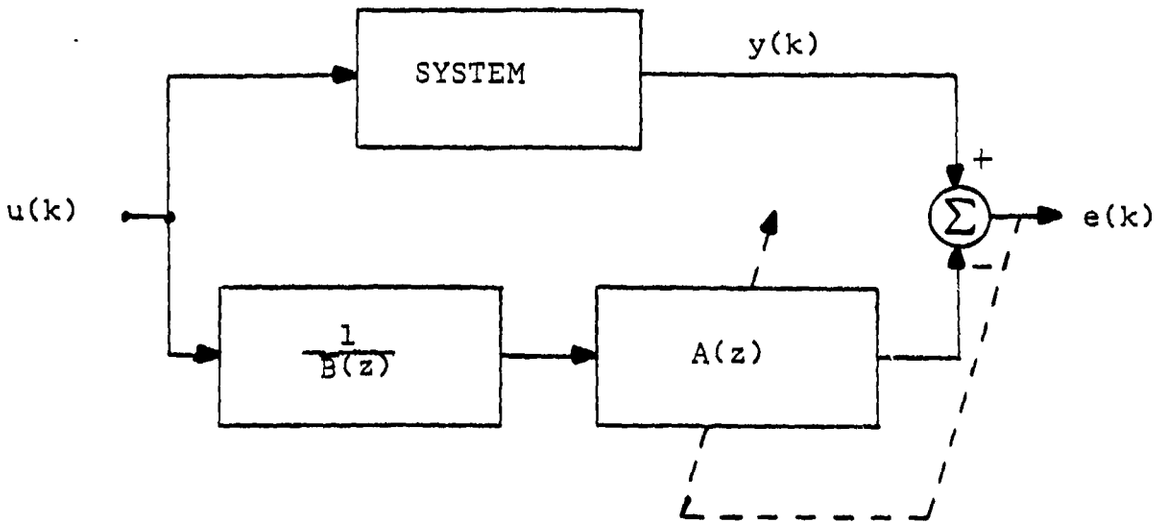


Figure 3.1. Shanks method for ARMA modeling

the numerator coefficients (or model zeros). This is intuitively unappealing in that one would expect these two estimation problems to be more closely coupled with the zero estimates also affecting the estimates of the poles.

The application of the equation error formulation to the ARMA modeling problem permits simultaneous estimation of the model zeros and poles and was first used by Kalman [Ref. 23] in work on self-optimizing control systems. The prediction error form of the model is considered here where f_{30} is set to zero while

$$F_{10}[u(k)] = \sum_{n=0}^{M} a(n)\, u(k-n) \qquad (3.2a)$$

$$F_{2}[y(k)] = y(k) - \sum_{n=1}^{N} b(n)\, y(k-n) \qquad (3.2b)$$

The analysis model is depicted in Figure 3.2 where

$$A(z) = \sum_{n=0}^{M} a(n)z^{-n} \tag{3.3a}$$

and

$$B(n) = 1 - \sum_{n=1}^{N} b(n)z^{-n} \tag{3.3b}$$

and these polynomials are the estimates of the system transfer function numerator and denominator.

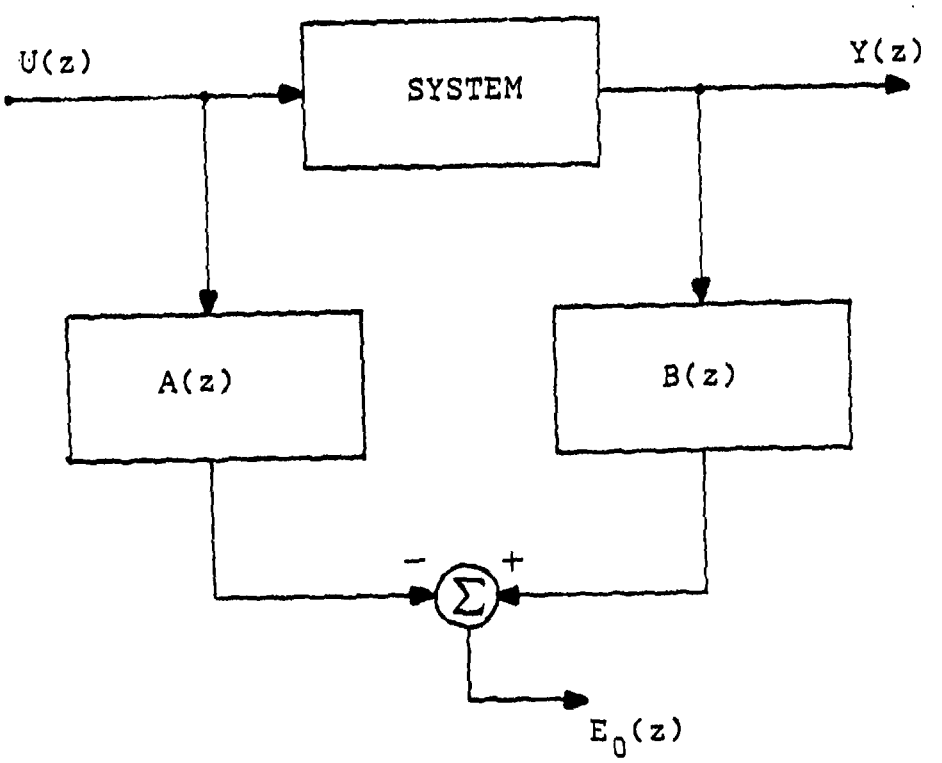


Figure 3.2. The equation error formulation for ARMA modeling.

The expression for the model error can be written as

$$e_0(k) = y(k) - [\underline{y}(k)^T \mid \underline{u}^+(k)^T] \begin{bmatrix} \underline{b} \\ \hline \underline{a}^+ \end{bmatrix} \tag{3.4a}$$

where $\underline{y}(k)$ and $\underline{b}$ are defined as in equation (2.7) and

$$\underline{u}^+(k) = [y(k) \cdots u(k-M)]^T \tag{3.4b}$$

$$\underline{a}^+ = [a(0) \cdots a(M)]^T \tag{3.4c}$$

This results in an expression for the mean square error which is a quadratic function of the model coefficients, with the MMSE solution for those coefficients given by

$$\begin{bmatrix} \underline{R}_{yy} & \underline{R}_{yu^+} \\ \hline \underline{R}_{u^+y} & \underline{R}_{u^+u^+} \end{bmatrix} \begin{bmatrix} \underline{b} \\ \hline \underline{a}^+ \end{bmatrix} = \begin{bmatrix} \underline{r}_{yy} \\ \hline \underline{r}_{u^+y} \end{bmatrix} \tag{3.5a}$$

and

$$E_{2_{min}} = R_{yy}(0) - [\underline{b}^T \mid \underline{a}^{+T}] \begin{bmatrix} \underline{r}_{yy} \\ \hline \underline{r}_{u^+y} \end{bmatrix} \tag{3.5b}$$

1. Model Transition Relationships

Comparing equations (3.4) and (3.5) with their AR and MA counterparts, it is clear that the ARMA model provides a generalization of these other models since they can be obtained from the ARMA model by assuming that either $\underline{a}^+$ or $\underline{b}$ is zero. Consequently it is susceptible to the same type of

bias introduced in the AR and MA models by the presence of additive noise on either the system input or output signals. To develop the relationships between these models further, consider the inversion of the correlation matrix in equation (3.5a) in terms of its component matricies. Since the left and right inverses of a nonsingular square matrix are the same, either

$$\begin{bmatrix} \underline{A} & \underline{B} \\ \underline{C} & \underline{D} \end{bmatrix} \begin{bmatrix} \underline{R}_{yy} & \underline{R}_{yu^+} \\ \underline{R}_{u^+y} & \underline{R}_{u^+u^+} \end{bmatrix} = \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{0} & \underline{I} \end{bmatrix} \tag{3.6a}$$

or

$$\begin{bmatrix} \underline{R}_{yy} & \underline{R}_{yu^+} \\ \underline{R}_{u^+y} & \underline{R}_{u^+u^+} \end{bmatrix} \begin{bmatrix} \underline{A} & \underline{B} \\ \underline{C} & \underline{D} \end{bmatrix} = \begin{bmatrix} \underline{I} & \underline{0} \\ \underline{0} & \underline{I} \end{bmatrix} \tag{3.6b}$$

can be used to find the required inverse in partitioned form. Solving for the right inverse of equation (3.6b) yields

$$\underline{A} = (\underline{R}_{yy} - \underline{R}_{yu^+}\underline{R}_{u^+u^+}^{-1}\underline{R}_{u^+y})^{-1} \tag{3.7a}$$

$$\underline{B} = -\underline{R}_{yy}^{-1}\underline{R}_{yu^+}(\underline{R}_{u^+u^+} - \underline{R}_{u^+y}\underline{R}_{yy}^{-1}\underline{R}_{yu^+})^{-1} \tag{3.7b}$$

$$\underline{C} = -\underline{R}_{u^+u^+}^{-1}\underline{R}_{u^+y}(\underline{R}_{yy} - \underline{R}_{yu^+}\underline{R}_{u^+u^+}^{-1}\underline{R}_{u^+y})^{-1} \tag{3.7c}$$

$$\underline{D} = (\underline{R}_{u^+u^+} - \underline{R}_{u^+y}\underline{R}_{yy}^{-1}\underline{R}_{yu^+})^{-1} \tag{3.7d}$$

Solving for the left inverse of equations (3.6a) gives identical results for $\underline{A}$ and $\underline{D}$ while equivalent but different forms are obtained for $\underline{B}$ and $\underline{C}$ given by

$$\underline{B} = -(\underline{R}_{yy} - \underline{R}_{yu^+} \underline{R}_{u^+u^+}^{-1} \underline{R}_{u^+y})^{-1} \underline{R}_{yu^+} \underline{R}_{u^+u^+}^{-1} \tag{3.8a}$$

$$\underline{C} = -(\underline{R}_{u^+u^+} - \underline{R}_{u^+y} \underline{R}_{yy} \underline{R}_{yu^+})^{-1} \underline{R}_{u^+y} \underline{R}_{yy}^{-1} \tag{3.8b}$$

Using equations (3.7) in equation (3.5a), the solutions for the ARMA coefficient vectors are given by

$$\underline{b} = (\underline{R}_{yy} - \underline{R}_{yu^+} \underline{R}_{u^+u^+}^{-1} \underline{R}_{u^+y})^{-1} \underline{r}_{yy}$$

$$- \underline{R}_{yy}^{-1} \underline{R}_{yu^+} (\underline{R}_{u^+u^+} - \underline{R}_{u^+y} \underline{R}_{yy}^{-1} \underline{R}_{yu^+})^{-1} \underline{r}_{u^+y} \tag{3.9a}$$

and

$$\underline{a}^+ = -\underline{R}_{u^+u^+}^{-1} \underline{R}_{u^+y} (\underline{R}_{yy} - \underline{R}_{yu^+} \underline{R}_{u^+u^+} \underline{R}_{u^+y})^{-1} \underline{r}_{yy}$$

$$+ (\underline{R}_{u^+u^+} - \underline{R}_{u^+y} \underline{R}_{yy}^{-1} \underline{R}_{yu^+})^{-1} \underline{r}_{u^+y} \tag{3.9b}$$

The matrix inversion lemma [Refs. 11 and 19] which states that

$$(E+FGH)^{-1} = E^{-1} - E^{-1}F(G^{-1}+HE^{-1}F)^{-1} HE^{-1} \tag{3.10}$$

for nonsingular square matrices E, G and E+FGH, can be used in equation (3.9) to rewrite them as

$$\underline{b} = \underline{R}_{yy}^{-1}\,\underline{r}_{yy} + \underline{R}_{yy}^{-1}\,\underline{R}_{yu^+}(\underline{R}_{u^+u^+} - \underline{R}_{u^+y}\,\underline{R}_{yy}^{-1}\,\underline{R}_{yu^+})^{-1}$$

$$[\underline{R}_{u^+y}\,\underline{R}_{yy}^{-1}\,\underline{r}_{yy} - \underline{r}_{u^+y}] \qquad (3.11a)$$

$$\underline{a}^{+} = \underline{R}_{u^+u^+}^{-1}\,\underline{r}_{u^+y} + \underline{R}_{u^+u^+}^{-1}\,\underline{R}_{u^+y}\,(\underline{R}_{yy} - \underline{R}_{yu^+}\,\underline{R}_{u^+u^+}^{-1}\,\underline{R}_{u^+y})^{-1}$$

$$[\underline{R}_{yu^+}\,\underline{R}_{u^+u^+}^{-1}\,\underline{r}_{u^+y} - \underline{r}_{yy}] \qquad (3.11b)$$

The all zero and all pole model solutions of corresponding orders however are

$$\underline{a}^{+}_{AZ} = \underline{R}_{u^+u^+}^{-1}\,\underline{r}_{u^+y} \quad \text{and} \quad \underline{b}_{AP} = \underline{R}_{yy}^{-1}\underline{r}_{yy}$$

where subscripts are used to distinguish these solutions from their counterparts in the ARMA zero pole model. From equations (3.11) it follows that

$$\underline{b}_{ZP} = \underline{b}_{AP} + \underline{R}_{yy}^{-1}\,R_{yu^+}\,(\underline{R}_{u^+u^+} - \underline{R}_{u^+y}\,\underline{R}_{yy}^{-1}\,\underline{R}_{yu^+})^{-1}$$

$$[\underline{R}_{u^+y}\,\underline{b}_{AP} - \underline{r}_{u^+y}] \qquad (3.12a)$$

$$\underline{a}_{ZP} = \underline{a}_{AZ} + \underline{R}^{-1}_{u^+u^+} \underline{R}_{u^+y} \left(\underline{R}_{yy} - \underline{R}_{yu^+} \underline{R}^{-1}_{u^+u^+} \underline{R}_{u^+y}\right)^{-1}$$

$$\left[\underline{R}_{yu^+} \underline{a}_{AZ} - \underline{r}_{yy}\right] \qquad (3.12b)$$

Following a similar development, the left inverse relationships of equations (3.8) can be used to write

$$\underline{b}_{ZP} = \left(\underline{R}_{yy} - \underline{R}_{yu^+} \underline{R}^{-1}_{u^+u^+} \underline{R}_{u^+y}\right)^{-1} \left[\underline{r}_{yy} - \underline{R}_{yu^+} \underline{a}_{AZ}\right] \qquad (3.12c)$$

$$\underline{a}_{ZP} = \left(\underline{R}_{u^+u^+} - \underline{R}_{u^+y} \underline{R}^{-1}_{yy} \underline{R}_{yu^+}\right)^{-1} \left[\underline{r}_{u^+y} - \underline{R}_{u^+y} \underline{b}_{AP}\right] \qquad (3.12d)$$

Equations (3.12) are termed the "Zero Pole Model Transition Formulas" and specify the relationships between the various models. It is interesting to note that equations (3.12a) and (3.12b) take the form of a linear observer with a new estimate of the solution given by the old estimate plus a gain times an error term. To gain some insight into the functioning of these formulas, consider the form of $R_{yy}(n)$ and $R_{uy}(n)$ for the linear system described by the transfer function of equation (3.1c)

$$R_{yy}(-n) = \sum_{L=1}^{N} b(i)\, R_{yy}(-i-n) + \sum_{L=0}^{M} a(i)\, R_{yu}(-i-n)$$

(3.13a)

$$R_{uy}(-n) = \sum_{L=1}^{N} b(i)\, R_{uy}(-i-n) + \sum_{i=0}^{M} a(i)\, R_{uu}(-i-n) \qquad (3.13b)$$

Assuming that N=N and M=M, and writing equation (3.13a) for $-N \leq n \leq -1$ and equation (3.13b) for $-M \leq n \leq 0$ results in

$$\underline{R}_{yy}\, \underline{b} + \underline{R}_{yu^+}\, \underline{a}^+ = \underline{r}_{yy} \qquad (3.14a)$$

$$\underline{R}_{u^+y}\, \underline{b} + \underline{R}_{u^+u^+}\, \underline{a}^+ = \underline{r}_{u^+y} \qquad (3.14b)$$

These constraints on the system input and output auto and cross correlation coefficients are the ARMA modeling equations of (3.5a) with the model coefficients replaced by the system parameters. In AR modeling, $\underline{b}_{AP}$ is set to satisfy the constraints of equation (3.14a) with the assumption that $\underline{a}^+$ is zero. The error term in the model transition formula (3.12a) then checks this solution to see if it also satisfies the constraints of equation (3.14b) still assuming that $\underline{a}^+$ is zero.

$$\text{error} = \underline{R}_{u^+y}\ \underline{b}_{AP} - \underline{r}_{u^+y} \qquad (3.15)$$

If this error is zero and the constraints of equation (3.14b) are satisfied, equation (3.12a) sets $\underline{b}_{ZP} = \underline{b}_{AP}$ and equation (3.12d) sets $\underline{a}^{+}$ to zero. If however the error is nonzero, (3.12a) adjusts $\underline{b}_{AP}$ in proportion to the error to obtain $\underline{b}_{ZP}$ and (3.12d) then provides a nonzero $\underline{a}^{+}_{ZP}$. Thus equations (3.12a) and (3.12d) are complementary, specifying the ARMA or zero pole model solution of order M over N when given the N-th order AR or all pole model solution.

In like manner, equations (3.12b) and (3.12c) give the ARMA model solution of order M over N when given the M-th order MA or all zero model solution. The all zero solution is obtained from equation (3.14b) assuming that $\underline{b}$ is zero. Equation (3.12b) checks this solution against the constraints imposed by equation (3.14a) with the same assumptions and adjusts $\underline{a}^{+}_{AZ}$ appropriately to determine $\underline{a}^{+}_{ZP}$. Equation (3.12c) then sets $\underline{b}_{ZP}$, completing the zero pole model solution.

2. Modeling Input Signal Requirements

Another aspect of the modeling problem that must be considered is that of system identifiability. If, from the available measurements of signals, a model can be obtained that accurately represents the system's operation, the system is considered identifiable. The two issues that arise therefore are the measurement requirements (which signals must be measured) and requirements on the input signal used to excite the system during the modeling process. In the equation error formulation of the ARMA

model, both existing signals (system input and output) must be observed (or at least a knowledge of their auto and cross correlation functions must be available). Most discussions of input signal requirements for identifiability simply state that the system can be identified if the input signal is sufficiently rich, persistent or exciting. eg [Ref. 19] To explore the question of input signal requirements further, consider the mean square equation error cost function being minimized. Assuming that the equation error signal is ergodic and has finite energy its mean square value is obtained via time averaging as

$$E_2 = \varepsilon\{e(k)^2\} = \sum_{n=-\infty}^{\infty} e(n)^2 \qquad (3.16)$$

Applying Parsevals relation this becomes

$$E_2 = \sum_{n=-\infty}^{\infty} e(n)^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} E(e^{j\theta})E^*(e^{j\theta})\, d\theta \qquad (3.17)$$

where $\theta = \omega T$ and * indicates the complex conjugate. The equation error is represented in the transform domain as

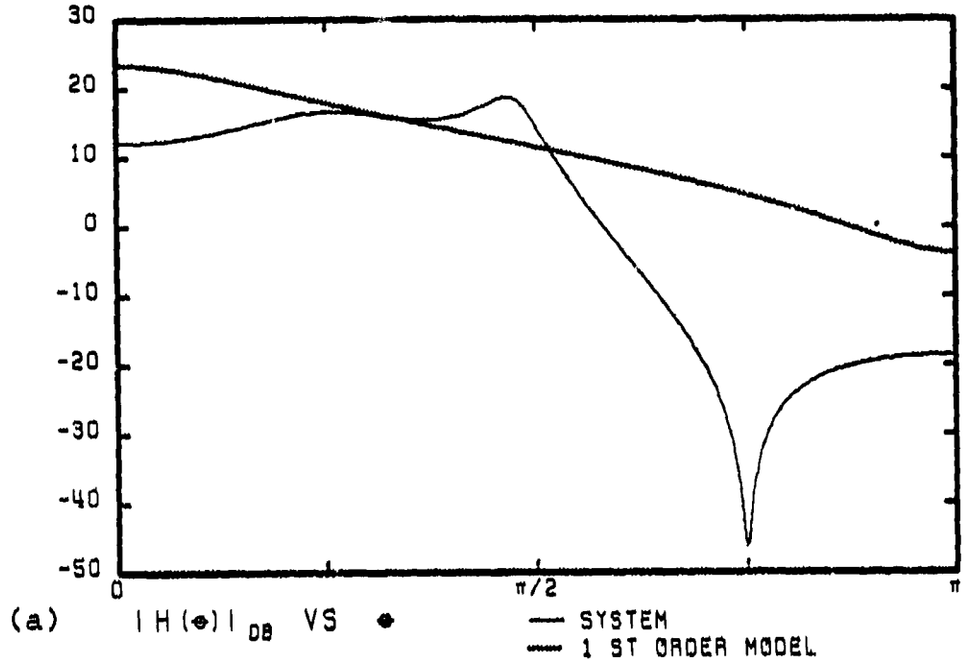
$$E(z) = [B(z)H(z) - A(z)]U(z) \qquad (3.18)$$

and using this in equation (3.17), the cost function becomes

$$E_2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |[B(e^{j\theta})H(e^{j\theta}) - A(e^{j\theta})|^2 \cdot |U(e^{j\theta})|^2 d\theta \qquad (3.19)$$

showing that the power spectrum of the input signal acts as a frequency dependent weighting function on a transfer function error term. Therefore to identify the system equally well at all frequencies, the input must have a flat spectrum as will be the case for a white noise input or an impulse function input. Otherwise, the model transfer function will only be matched to the system transfer function over the range of frequencies where the input signal has significant power.

As an example consider the equation error ARMA model for a fourth order system driven by a single sine wave input at a frequency of $\pi/3$. According to equation (3.19), the model transfer function will only be required to match the system at this single frequency, and to accomplish this, only a first order model is needed. Any increase in model order above first order therefore should have no effect. Figure 3.3a shows a comparison of the magnitude spectrum of the fourth order system and its first order ARMA model obtained using the sinusoidal input and as anticipated they match at the frequency $\pi/3$ (coincidentally they also match at one other frequency as well). Figure 3.3b shows the same comparison but with a fourth order ARMA model. It is clear that increasing the model order failed to improve its accuracy and that the model accurately represents the system only at the frequency of the input signal and, by coincidence, at one other frequency.



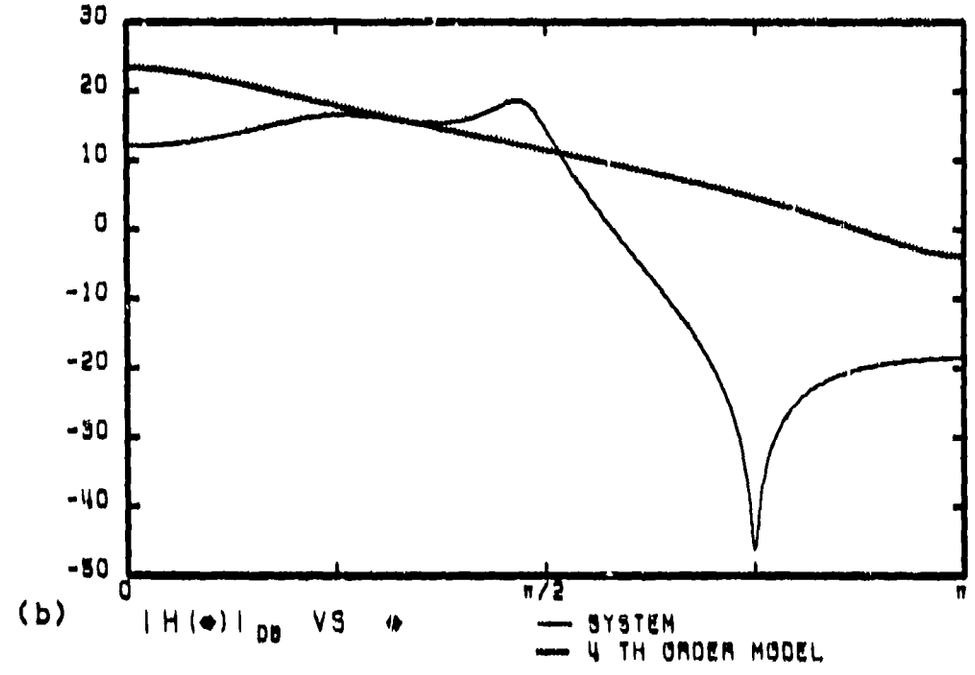


Figure 3.3. ARMA models for a system driven by a single sinusoid.

It should be noted that this type of analysis to determine input signal requirements could be applied to the AR and MA models as well resulting in the same conclusions.

B. A RECURSIVE IN ORDER SOLUTION FOR THE ARMA MODEL

Since the equation error formulation for the ARMA model is a generalization of similar formulations for AR and MA modeling, it is reasonable to assume that a Levinson-type algorithm could be devised to obtain the ARMA solution recursively in order, and that from that algorithm, lattice filter methods applicable to the ARMA modeling problem could be derived. Attempts to develop such an algorithm directly for the ARMA modeling equation (3.5a), however, fail to provide useful results. The first problem that arises is in deciding which model order to make recursive; the order of the numerator polynomial, the order of the denominator polynomial or both. If it is assumed that the numerator and denominator are of equal orders (M=N), the ARMA modeling equations become as shown in equation (3.20). However, efforts to develop a Levinson-type algorithm for this system of equations, where the numerator and denominator polynomials of the model are incremented simultaneously to obtain recursive in order solutions, are still frustrated by the presence of the (N+1)-st row and column in equation (3.20). This arises because the numerator coefficient vector $\underline{a}^+$ is a (N+1)-vector while the denominator coefficient vector $\underline{b}$ is a N-vector. If it is further assumed that the coefficient

$$
\left[\begin{array}{cccc|cccc}
R_{yy}(0) & R_{yy}(-1) & \cdots & R_{yy}(1-N) & R_{yu}(1) & R_{yu}(0) & \cdots & R_{yu}(1-N) \\
R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(2-N) & R_{yu}(2) & R_{yu}(1) & \cdots & R_{yu}(2-N) \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
R_{yy}(N-1) & R_{yy}(N-2) & \cdots & R_{yy}(-N) & R_{yu}(0) & R_{yu}(-1) & \cdots & R_{yu}(0) \\
\hline
R_{uy}(-1) & R_{uy}(-2) & \cdots & R_{uy}(-N) & R_{uu}(0) & R_{uu}(-1) & \cdots & R_{uu}(-N) \\
R_{uy}(0) & R_{uy}(-1) & \cdots & R_{uy}(1-N) & R_{uu}(1) & R_{uu}(0) & \cdots & R_{uu}(1-N) \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
R_{uy}(N-1) & R_{uy}(N-2) & \cdots & R_{uy}(0) & R_{uu}(N) & R_{uu}(N-1) & \cdots & R_{uu}(0)
\end{array}\right]
\left[\begin{array}{c} b(1) \\ b(2) \\ \vdots \\ b(N) \\ \hline a(0) \\ a(1) \\ \vdots \\ a(N) \end{array}\right]
=
\left[\begin{array}{c} R_{yy}(1) \\ R_{yy}(2) \\ \vdots \\ R_{yy}(N) \\ \hline R_{uy}(0) \\ R_{uy}(1) \\ \vdots \\ R_{uy}(N) \end{array}\right]
\qquad (3,20)
$$

a(0) is known in advance or can be estimated in some other fashion, equation (3.20) for the solution for the remaining 2N coefficients becomes as written in equation (3.21).

$$\left[\begin{array}{ccc|ccc} R_{yy}(0) & \cdots & R_{yy}(1-N) & R_{yu}(0) & \cdots & R_{yu}(1-N) \\ \vdots & & & \vdots & & \vdots \\ R_{yy}(N-1) & \cdots & R_{yy}(0) & R_{yu}(N-1) & \cdots & R_{yu}(0) \\ \hline R_{uy}(0) & \cdots & R_{uy}(1-N) & R_{uu}(0) & \cdots & R_{uu}(1-N) \\ \vdots & & & \vdots & & \vdots \\ R_{uy}(N-1) & \cdots & R_{uy}(0) & R_{uu}(N-1) & \cdots & R_{uu}(0) \end{array}\right] \left[\begin{array}{c} b(1) \\ \vdots \\ b(N) \\ \hline a(1) \\ \vdots \\ a(N) \end{array}\right] = \left[\begin{array}{c} R_{yy}(1) \\ \vdots \\ R_{yy}(N) \\ \hline R_{uy}(1) \\ \vdots \\ R_{uy}(N) \end{array}\right] - a(0) \left[\begin{array}{c} R_{yu}(1) \\ \vdots \\ R_{yu}(N) \\ \hline R_{uu}(1) \\ \vdots \\ R_{uu}(N) \end{array}\right]$$

(3.21)

The coefficient a(0) essentially has the role of a gain for the model, and a method for estimating it after all the other coefficients are obtained (as was done in AR modeling) will be discussed later. Equation (3.21) can be written as

$$\begin{bmatrix} \underline{R}_{yy} & \underline{R}_{yu} \\ \underline{R}_{uy} & \underline{R}_{uu} \end{bmatrix} \begin{bmatrix} \underline{b} \\ \underline{a} \end{bmatrix} = \begin{bmatrix} \underline{r}_{yy} & \underline{r}_{yu} \\ \underline{r}_{uy} & \underline{r}_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ -a(0) \end{bmatrix} \qquad (3.22)$$

Now consider the form of a two channel autoregressive model where the two input channels are y(k) and u(k); that is

$$x_1(k) = y(k)$$

$$x_2(k) = u(k)$$

Using equations (A.4a) and (A.7b) the two channel AR modeling equations are

$$\begin{bmatrix} \underline{R}_{yy} & \underline{R}_{yu} \\ \underline{R}_{uy} & \underline{R}_{uu} \end{bmatrix} \begin{bmatrix} \underline{d}_{11} & \underline{d}_{12} \\ \underline{d}_{21} & \underline{d}_{22} \end{bmatrix} = \begin{bmatrix} \underline{r}_{yy} & \underline{r}_{yu} \\ \underline{r}_{uy} & \underline{r}_{uu} \end{bmatrix} \qquad (3.23)$$

and comparing equations (3.22) and (3.23) it is clear that with the exception of the gain term a(0), the ARMA modeling solution can be obtained from the two channel AR solution of (3.23). Furthermore equation (3.23) can be solved independently of the gain term via the Levinson algorithm as shown in Appendix A or via the multichannel lattice methods developed in the previous chapter. Then all that remains to complete the solution for the ARMA model is to estimate the gain term a(0) and solve for the other model coefficients using

$$\begin{bmatrix} \underline{b} \\ \underline{a} \end{bmatrix} = \begin{bmatrix} \underline{d}_{11} & \underline{d}_{12} \\ \underline{d}_{21} & \underline{d}_{22} \end{bmatrix} \begin{bmatrix} 1 \\ -a(0) \end{bmatrix} \qquad (3.24)$$

From this it follows that the transfer functions A(z) and B(z) for the ARMA prediction error analysis model can be related to the two channel AR prediction error matrix polynomial transfer function by

$$\begin{bmatrix} B(z) \\ \\ -A(z) \end{bmatrix} = [\underline{I} - \mathbf{D}(z)] \begin{bmatrix} 1 \\ \\ -a(0) \end{bmatrix} \tag{3.25}$$

It is also easy to relate the ARMA equation error to the two channel AR prediction error vector. Using equations (A.3) and (A.5), the two channel AR prediction error vector is written as

$$\underline{e}(k)^T = \begin{bmatrix} y(k) \\ u(k) \end{bmatrix}^T - [\underline{y}(k)^T \mid \underline{u}(k)^T] \begin{bmatrix} \underline{d}_{11} & \underline{d}_{12} \\ \underline{d}_{21} & \underline{d}_{22} \end{bmatrix} \tag{3.26}$$

$$= [e_y(k) \quad e_u(k)]$$

Defining

$$\underline{\Psi} = \begin{bmatrix} 1 \\ -a(0) \end{bmatrix} \tag{3.27a}$$

and postmultiplying equation (3.26) by $\underline{\psi}$ and using (3.24) it follows that

$$\underline{e}(k)^T \underline{\psi} = y(k) - a(0)u(k) - [\underline{y}(k)^T \vdots \underline{u}(k)^T] \begin{bmatrix} \underline{b} \\ \cdots \\ \underline{a} \end{bmatrix} \qquad (3.27b)$$

But from equation (3.4a) this is exactly the ARMA model equation error so that

$$e_0(k) = \underline{e}(k)^T \underline{\psi} \qquad (3.27c)$$

and

$$\mathcal{E}\{e_0(k)^2\} = \underline{\psi}^T \underline{P} \underline{\psi} \qquad (3.27d)$$

where $\underline{P}$ is the forward prediction error covariance matrix for the 2 channel AR model. Equation (3.27d) also provides a means of estimating the gain term a(0) after the two channel AR solution has been obtained by setting it to minimize the mean square value of equation error resulting in

$$a(0) = \frac{\mathcal{E}\{e_u(k)e_y(k)\}}{\mathcal{E}\{e_u(k)^2\}} \qquad (3.28)$$

and completing the ARMA model solution.

The portion of the ARMA model solution in equation (3.24) given by the two channel AR solution can be found recursively in order using either the Levinson algorithm or the lattice filter techniques. If the desired ARMA model order is not known in advance, a model gain term a(0) can be estimated for each order two channel AR solution to find the ARMA model of corresponding order along with its MSE. In this fashion, the entire family of ARMA models for the system from order zero to order N, along with their mean square errors are obtained. If, on the other hand, the desired ARMA model order is known apriori, the gain term need not be calculated at each stage. Only one gain term must be calculated to obtain the ARMA solution after the appropriate order two channel AR solution has been found.

It has already been shown that to fully identify the system using the equation error ARMA formulation, the input signal must have a flat spectrum as in the case of white noise. When white noise is used as the system input u(k), simplifications emerge in the solution of the two channel AR model via the Levinson algorithm or lattice methods. For a white sequence with variance σ_u^2,

$$R_{uu}(n) = \begin{cases} 0 & , \ n \neq 0 \\ \sigma_u^2 & , \ n = 0 \end{cases} \qquad (3.29a)$$

and

$$R_{uy}(n) = \begin{cases} 0 & ; n < 0 \\ \sigma_u^2 h(n) & ; n \geq 0 \end{cases} \qquad (3.29b)$$

where h(n) is the sampled impulse response of the system. Consider equation (A.16c) for $\underline{K}^{(1)}$.

$$\underline{K}^{(1)} = \begin{bmatrix} R_{yy}(0) & R_{yu}(0) \\ R_{uy}(0) & R_{uu}(0) \end{bmatrix}^{-1} \begin{bmatrix} R_{yy}(1) & 0 \\ R_{uy}(0) & 0 \end{bmatrix} \qquad (3.29c)$$

This shows that $k_{12}^{(1)}$ and $k_{22}^{(1)}$ are zero and furthermore since $\underline{r}_{yu}^{(n)}$ and $\underline{r}_{uu}^{(n)}$ are zero for all n it is seen that

$$k_{12}^{(n)} = k_{22}^{(n)} = 0 \qquad (3.29d)$$

for all n as well. This can readily be understood by considering the role of these two coefficients at each stage in the AR prediction of y(k) and u(k). k_{12} and k_{22} are the coefficients used in trying to predict u(k) from past values of y(k) and u(k), and when u(k) is a white sequence it cannot be predicted, forcing these coefficients to be zero. No such simplifications occur in the backward prediction problem (and therefore in the $\bar{K}$ matrices) since even for a white u(k), a backward prediction of u(k-n) from subsequent values of u and y is possible. This is because in

general, a linear dependence of y(k) upon past and present values of u(k) can occur (and certainly will occur when the relationship between y(k) and u(k) is described by an ARMA model).

As a result of these simplifications, it is seen from equation (A.19a) that the polynomials $d_{12}(z)$ and $d_{22}(z)$ are zero when u(k) is white and the ARMA model is given by

$$\begin{bmatrix} B(z) \\ -A(z) \end{bmatrix} = \begin{bmatrix} 1-d_{11}(z) & 0 \\ -d_{21}(z) & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -a(0) \end{bmatrix} \qquad (3.30)$$

In this special case it follows that

$$B(z) = \det[\underline{I} - \underline{D}(z)] \qquad (3.31)$$

and therefore stability of the ARMA model and of the two channel AR model are equivalent. In general, however, no such connection exists for arbitrary input signals. Furthermore, even when the system input u(k) is white, solutions for k_{12} and k_{22} will not in general be exactly zero since the required correlations are usually not known and must be estimated.

This development showing that the ARMA model solution can be obtained from a two channel autoregressive model depends on two assumptions:

1) The numerator and denominator polynomial orders in the model transfer function are assumed to be the

same and are incremented simultaneously to build up the desired solution recursively in order;

2) It is assumed that the coefficient of z to the zero power (a(0)) in the numerator polynomial of the model is either known in advance, or that another means of estimating it can be found so that it need not be estimated directly in the modeling equations of (3.20).

The second assumption causes no concern since the two channel autoregressive solution is obtained independently of a(0), and given that solution, it has been shown that a(0) can indeed be estimated in another fashion in equation (3.28).

The first assumption however, warrants further consideration since it seems somewhat restrictive (at first) to require that the numerator and denominator polynomials of the model have the same order when in fact, the system being modeled may have different order numerator and denominator polynomials. To see that this assumption is not restrictive in general, consider what is occuring as the model is built up recursively in order. At each model order n, the procedure finds the best n-th order model (with n zeros and n poles) in a minimum mean square equation error sense. Making the model numerator and denominator orders different (or equivalently, forcing some of the coefficients to zero in the model where the orders are the same) places a priori constraints on the model, forcing some of the poles or zeros to the origin in the z-plane, rather than allowing the

model to place them at will to minimize the cost function. As an example, consider the process of obtaining an ARMA model for a system given by

$$H(z) = \frac{\sum_{n=0}^{2} a(n)\, z^{-n}}{1 - \sum_{n=1}^{4} b(n)\, z^{-n}}$$

where two of the systems four zeros actually occur at the origin of the z plane. Constraining any of the model zeros to the origin at orders one, two or three will result in a model with higher cost (MSE) than if they were not constrained. Even at order three, a model without constraints can be expected to use the "extra" zero to help in approximating the effects of the system's fourth pole as shown in equation (2.13) yielding a lower cost and more accurate model than would result if one zero were forced to the origin. Only at order four are such constraints reasonable but even then, they are not necessary since the modeling procedure itself should recognize that the best fourth order model will have two zeros at z=0.

Therefore, it is seen that assuming equal orders for the model numerator and denominator is entirely reasonable as a general approach in obtaining MMSE models for unknown systems or even reduced order models for known systems. When it is known in advance that the best MMSE model for a system has zeros at z=0, imposing such a constraint on the model can

reduce the computational complexity of obtaining the solution, but even here the assumption of equal orders is not restrictive.

C. LATTICE FORM ARMA MODELING

In chapter two it was shown that the lattice structure of Figure 2.7 could be applied to solve the multichannel AR modeling problem in terms of the reflection coefficient matrices given by equations (2.49). Since the ARMA model solution can be obtained from a two channel AR solution with y(k) and u(k) as the input channels, only the structure described by equation (3.27c) need be added to a two channel AR lattice to obtain the lattice form of the ARMA analysis model. It is interesting to look at the exact structure of this lattice model as shown in Figure 3.4 for a second order case. This structure is seen as a lattice interconnection of two single channel AR lattices operating on the input signals y(k) and u(k). The coefficients on the main diagonals of the K and $\bar{K}$ matrices specify the single channel lattices while the off diagonal elements specify the interconnections. (This will also be the case for extensions to lattices with any number of channels.)

The ARMA synthesis model implementing the transfer function A(z)/B(z) can also be put in lattice form. The forward prediction in the two channel AR analysis lattice is described by

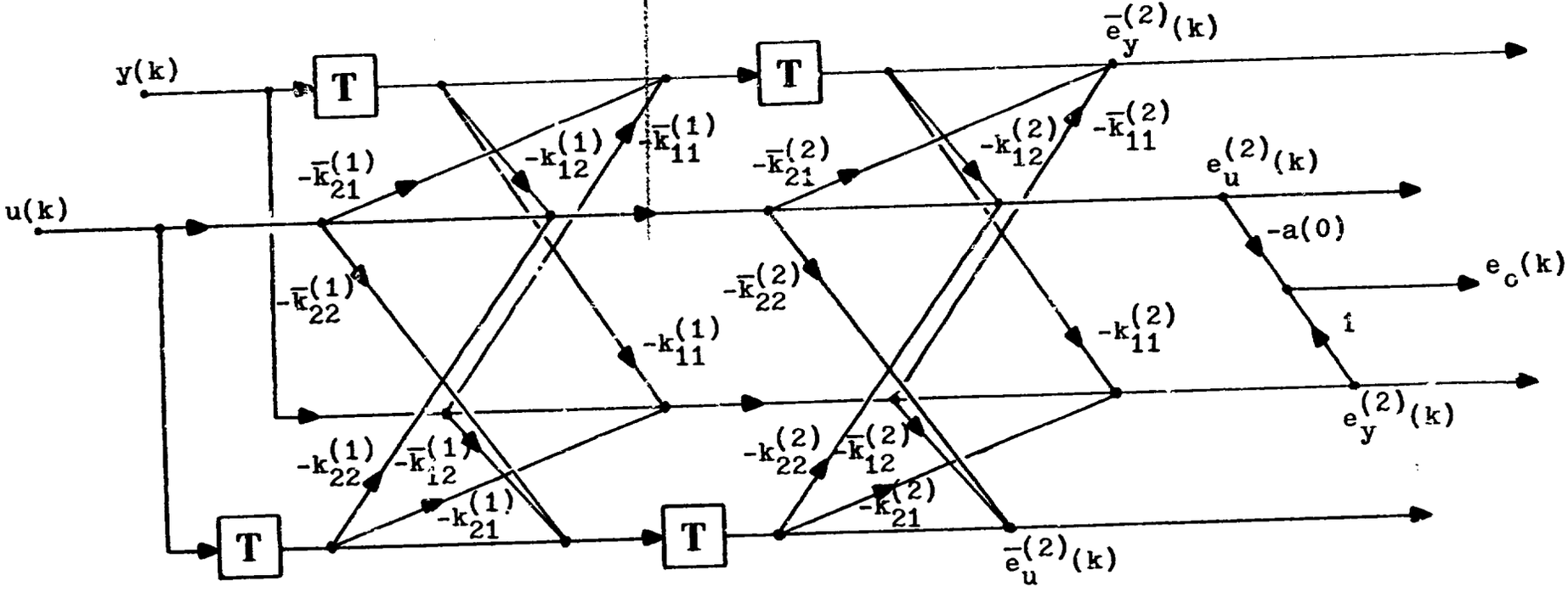


Figure 3.4. Lattice analysis structure for ARMA modeling.

$$\begin{bmatrix} e_y(k) \\ e_u(k) \end{bmatrix}^{(n+1)} = \begin{bmatrix} e_y(k) \\ e_u(k) \end{bmatrix}^{(n)} - \begin{bmatrix} k_{11} & k_{21} \\ k_{12} & k_{22} \end{bmatrix}^{(n+1)} \begin{bmatrix} \bar{e}_y(k-1) \\ \bar{e}_u(k-1) \end{bmatrix}^{(n)} \tag{3.32}$$

and the equation for $e_k(k)^{(n+1)}$ can be rewritten as

$$e_y(k)^{(n)} = e_y(k)^{(n+1)} + k_{11}^{(n+1)} \bar{e}_y(k-1)^{(n)} + k_{21}^{(n+1)} \bar{e}_u(k-1)^{(n)} \tag{3.33}$$

Equation (3.33) along with the equation of e_u in (3.32), and the equations for the backward prediction (2.45b) describe the structure shown in Figure 3.5 for a second order case. To provide the required input at $e_y(k)^{(2)}$, recognize from equation (3.27c) that

$$e_y(k)^{(2)} = e_0(k) + a(0) e_u(k)^{(2)} \tag{3.34a}$$

If the ARMA model is an accurate representation of the system, the equation error $e_0(k)$ will be quite small (ideally zero) so that in general for the N-th order case

$$e_y(k)^{(N)} \simeq a(0)\ e_u(k)^{(N)} \tag{3.34b}$$

This is indicated by the dashed feedback path in Figure 3.5.

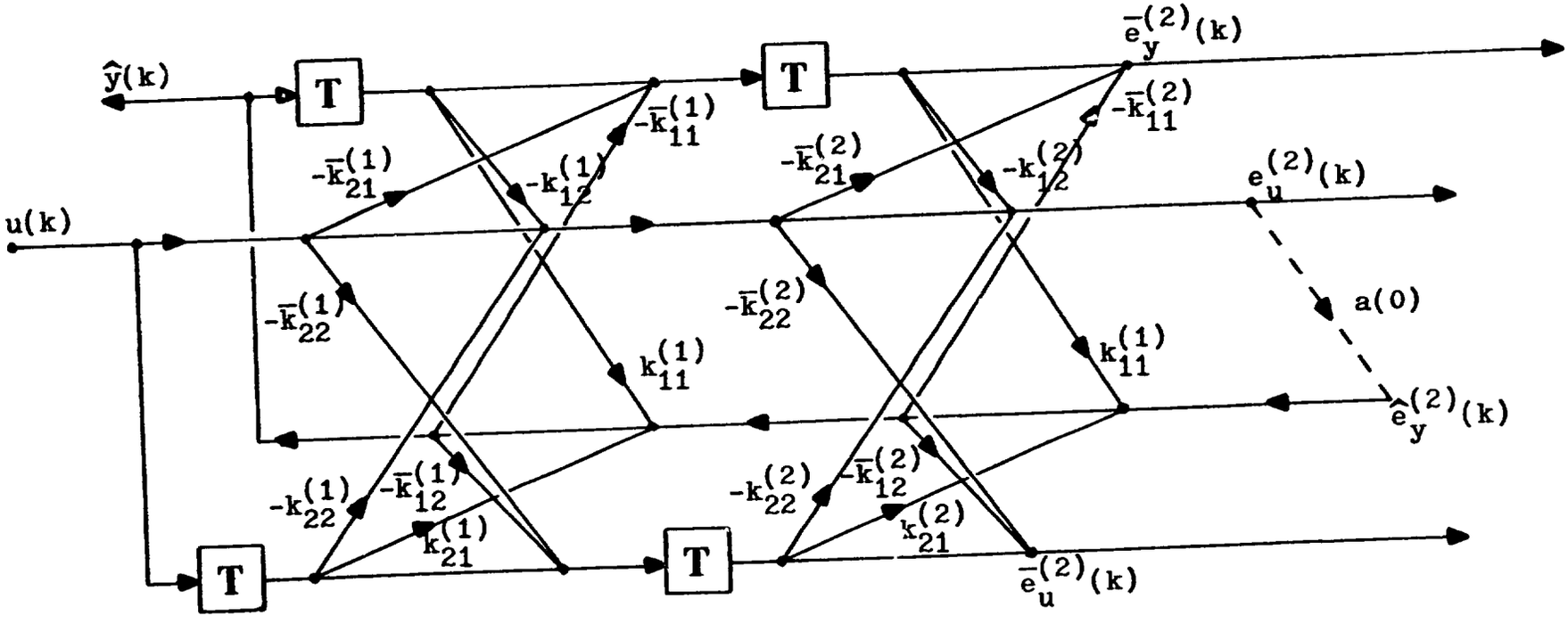


Figure 3.5. Lattice synthesis structure for ARMA modeling.

D. ARMA MODEL SIMULATIONS; BATCH PROCESSING

ARMA modeling procedures for linear systems using both the lattice filter method and a brute force matrix inversion with equation (3.20) have been implemented and the results of these two approaches have been compared in over a thousand model simulations of more than thirty different linear systems. The experimental results which follow are a representative sampling of these simulations.

In the lattice filter method, equations (2.49) were used to calculate the forward and backward reflection coefficient matrices, with time averages over a specified interval used to estimate the required correlations in $\underline{P}^{(0)} = \underline{\bar{P}}^{(0)}$ and $\underline{\Delta}^{(n)}$ for $0 \leq n \leq N-1$. Equations (A.15), (A.16a) and (A.16b) were used to obtain the two channel AR model coefficients from the K and $\bar{K}$ matrices and with the gain calculated in equation (3.28), equation (3.24) was used to obtain the desired ARMA model coefficients. Equations (A.18) were used to update the forward and backward prediction error covariance matrices from one lattice stage to the next. Figure 3.6a provides a flow diagram of the procedure.

In the brute force matrix inversion method with equation (3.20) time averaging was again employed to estimate the required correlation coefficients. A rectangular window was applied to the data, however, to retain the even symmetry of the autocorrelation function in these estimates. The ARMA model coefficients were then obtained using a general purpose library subroutine (which employed gaussian

elimination) to solve equation (3.20). Figure 3.6b provides a flow diagram of this procedure.

In both cases zero mean, unit variance gaussian white noise was used as the system input. In the simulation results that follow, a description of each system discussed (transfer function coefficients, zero locations and pole locations) is listed in tabular form and root locations in the z plane as well as transfer function magnitudes are plotted for the system, and various models obtained for it. In each case, models were obtained for averaging intervals of 200, 500, 1000, 2000 and 4000 data points. Only the results for the two extremes of 200 and 4000 points are included here.

The first system considered has a second order numerator in z inverse and a fourth order denominator, and its characteristics are listed in Table 3.1. Figure 3.7 shows a comparison of the root locations and transfer function magnitude of this system with those of fourth order lattice filter and brute force models obtained when correlations were estimated by averaging over 200 samples. Figure 3.8 provides the same comparison for a longer averaging interval of 4000 samples of data. While both methods perform comparably with the longer averaging interval, the lattice method produces a far more accurate model with the short averaging interval. It is also interesting to compare the performance of the two methods when the system is overmodeled; that is, when the model order is increased beyond that of

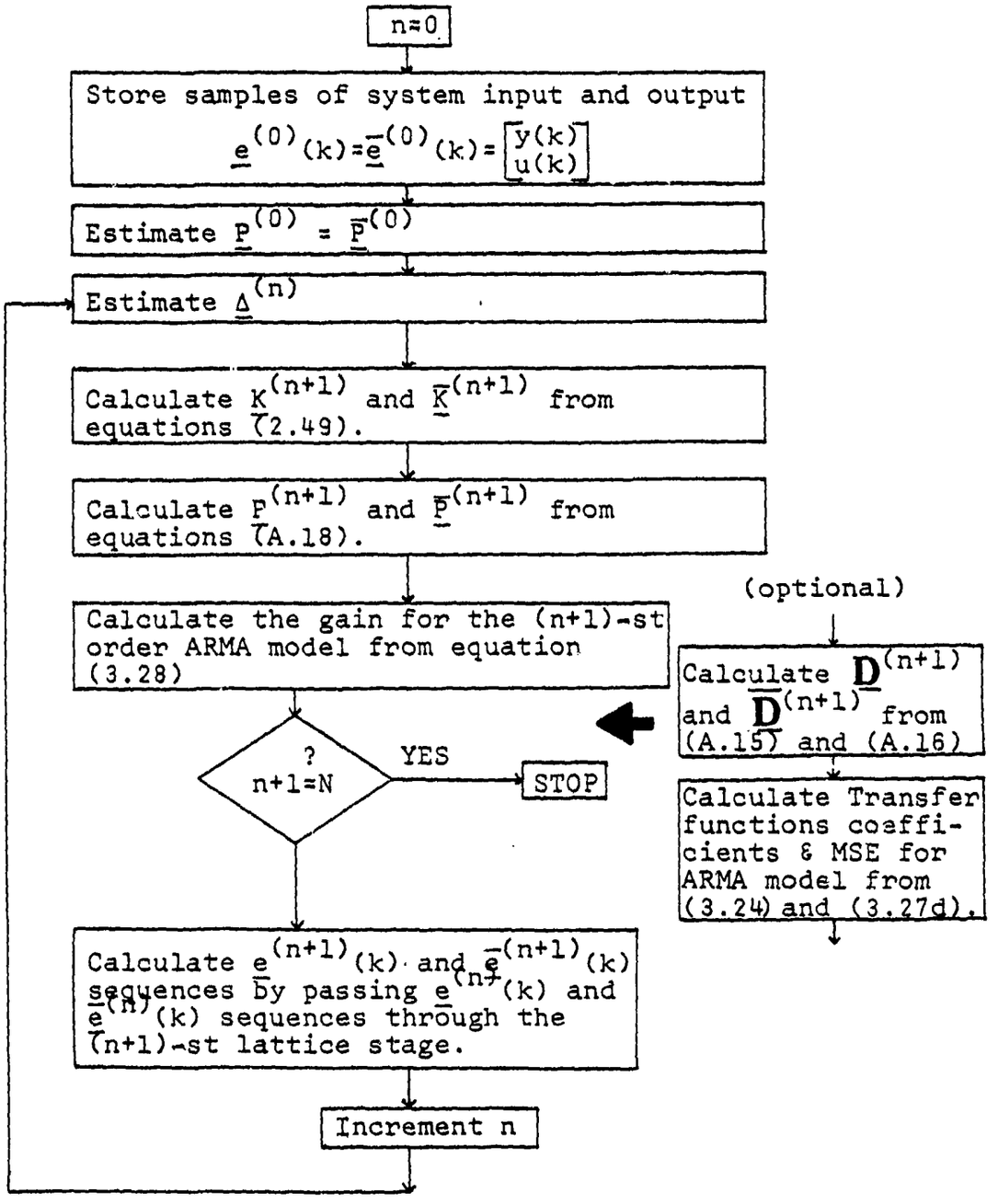


Figure 3.6a. Flow diagram of batch processing lattice solution for all ARMA models orders 1 to N.

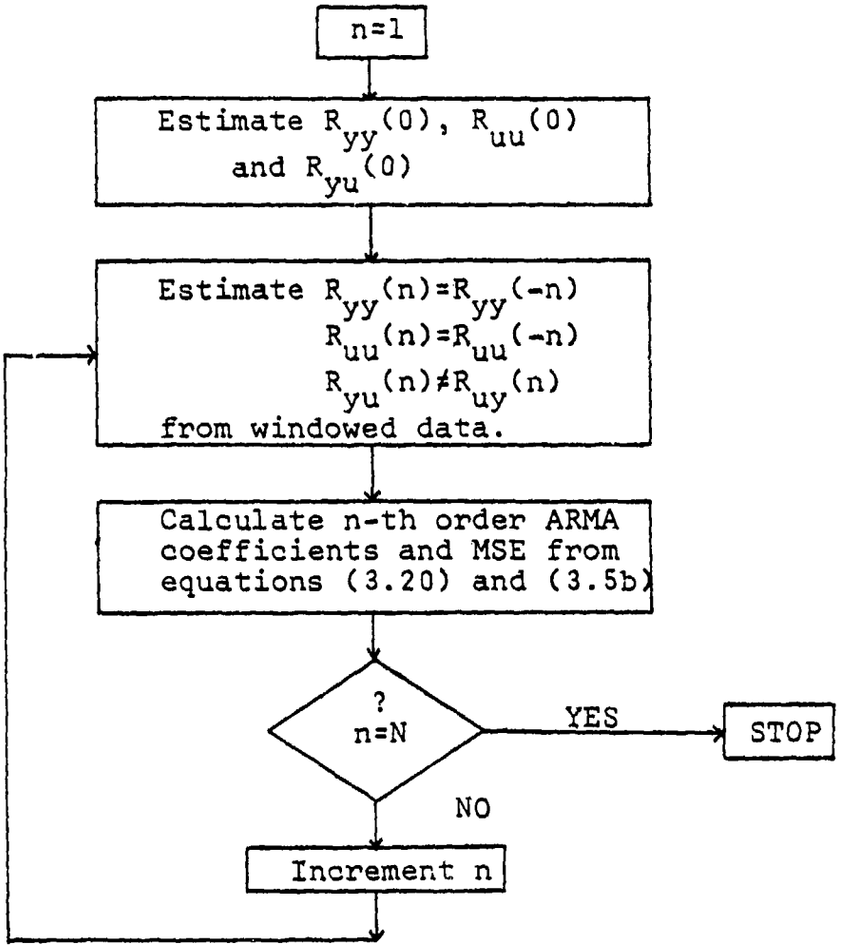


Figure 3.6b. Flow diagram of batch processing brute force solution for all ARMA models of order 1 to N.

TABLE 3.1

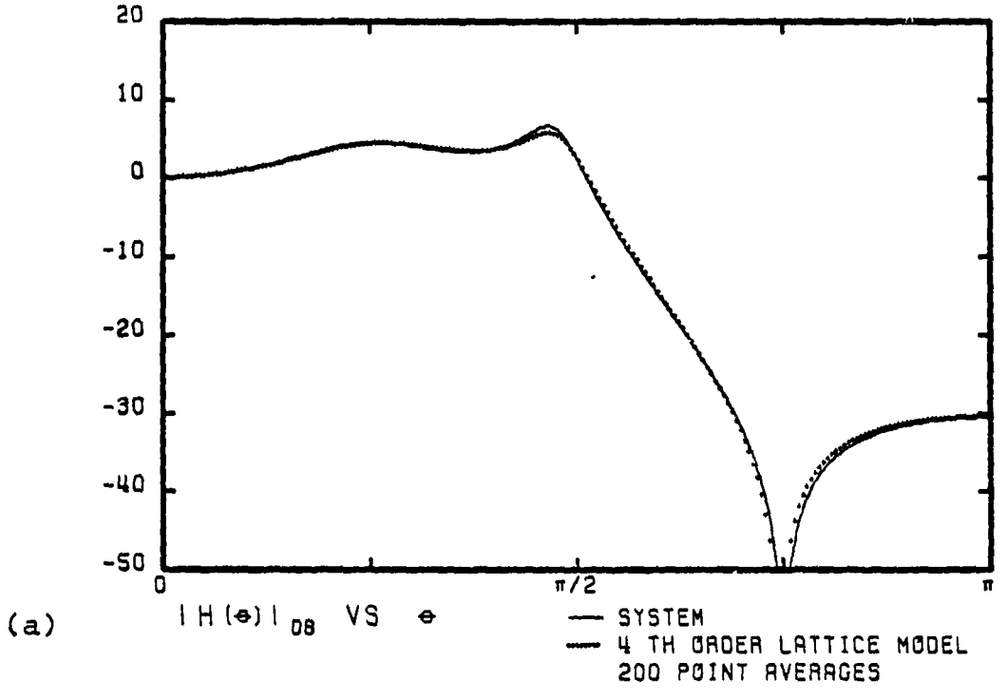
SYSTEM A

TRANSFER FUNCTION COEFFICIENTS

NUMERATOR		DENOMINATOR	
A(0)=	0.25000		
A(1)=	0.35000	B(1)=	1.14000
A(2)=	0.24500	B(2)=	-1.45490
A(3)=	0.0	B(3)=	0.88490
A(4)=	0.0	B(4)=	-0.40745

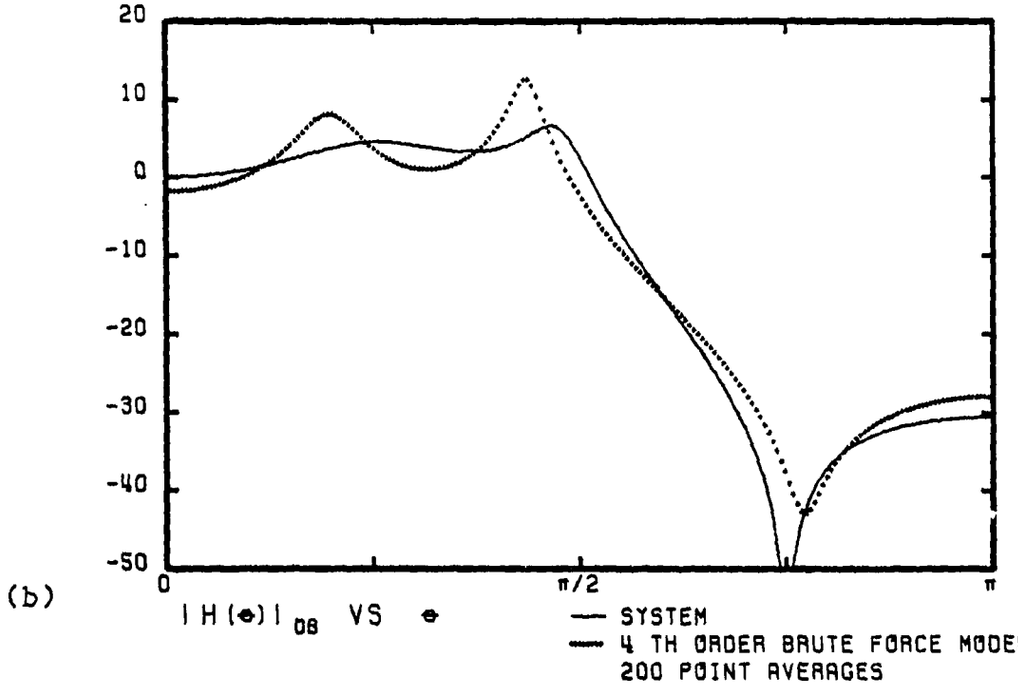
ROOT LOCATIONS

ZEROS		POLES	
RE	IM	RE	IM
-0.70000	0.70000	0.50000	0.50000
-0.70000	-0.70000	0.50000	-0.50000
0.0	0.0	0.07000	0.90000
0.0	0.0	0.07000	-0.90000



(a) $|H(\theta)|_{dB}$ VS θ

— SYSTEM
— 4 TH ORDER LATTICE MODEL
200 POINT AVERAGES



(b) $|H(\theta)|_{dB}$ VS θ

— SYSTEM
— 4 TH ORDER BRUTE FORCE MODEL
200 POINT AVERAGES

Figure 3.7

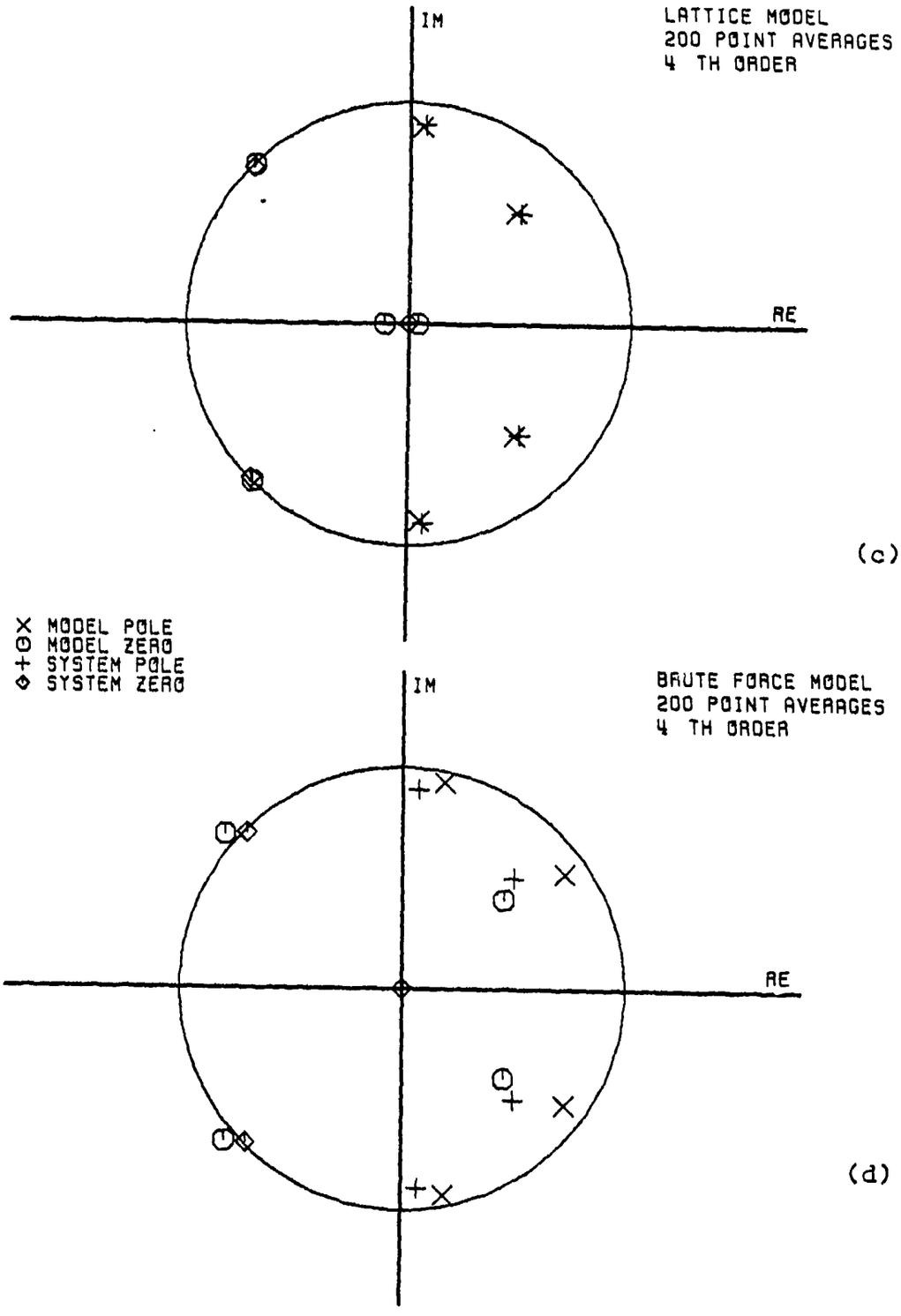
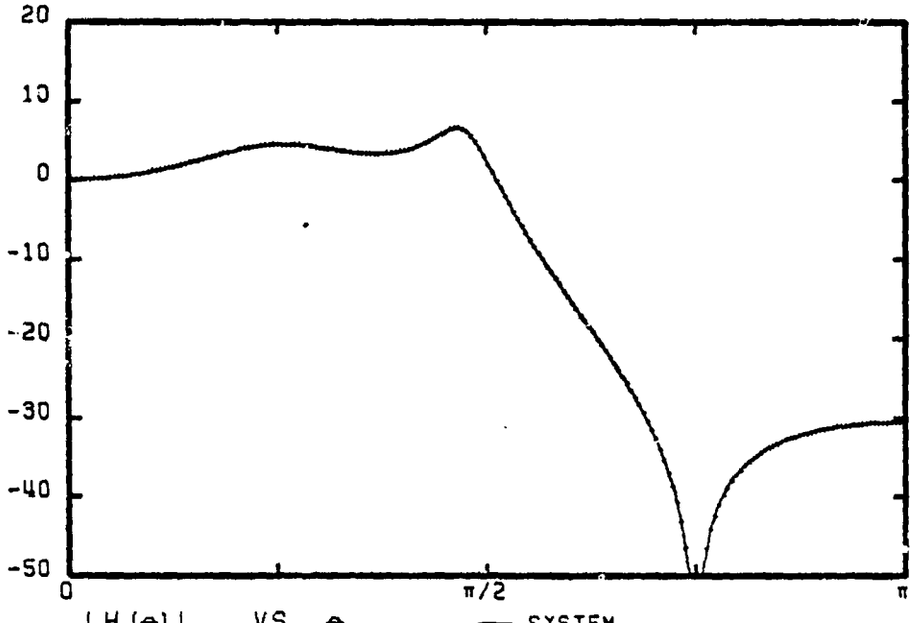
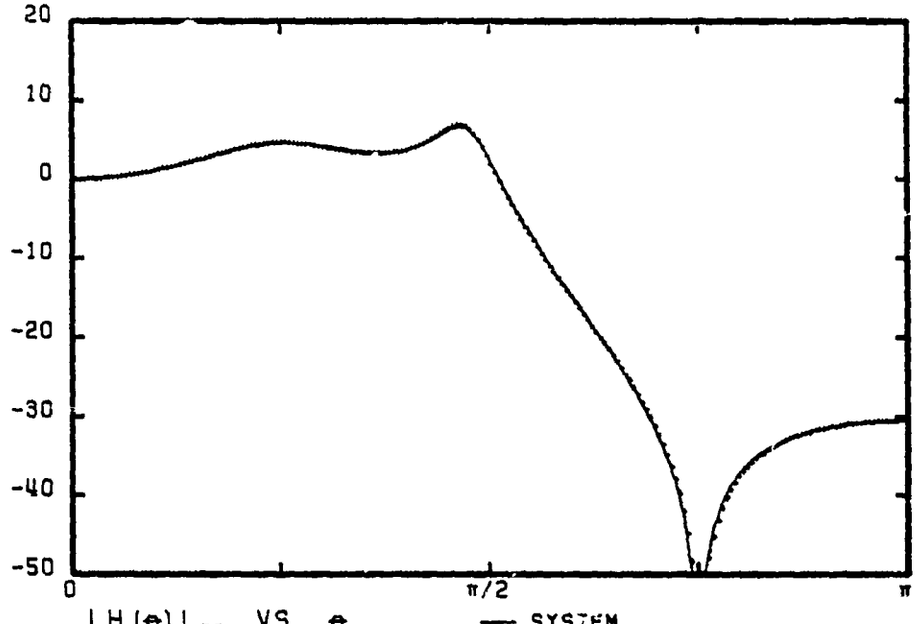


Figure 3.7 con't


20
10
0
-10
-20
-30
-40
-50
0
π/2
π
|H(e)|dB VS e
SYSTEM
4 TH ORDER LATTICE MODEL
4000 POINT AVERAGES

(a)


20
10
0
-10
-20
-30
-40
-50
0
π/2
π
|H(e)|dB VS e
SYSTEM
4 TH ORDER BRUTE FORCE MODEL
4000 POINT AVERAGES

(b)

Figure 3.8

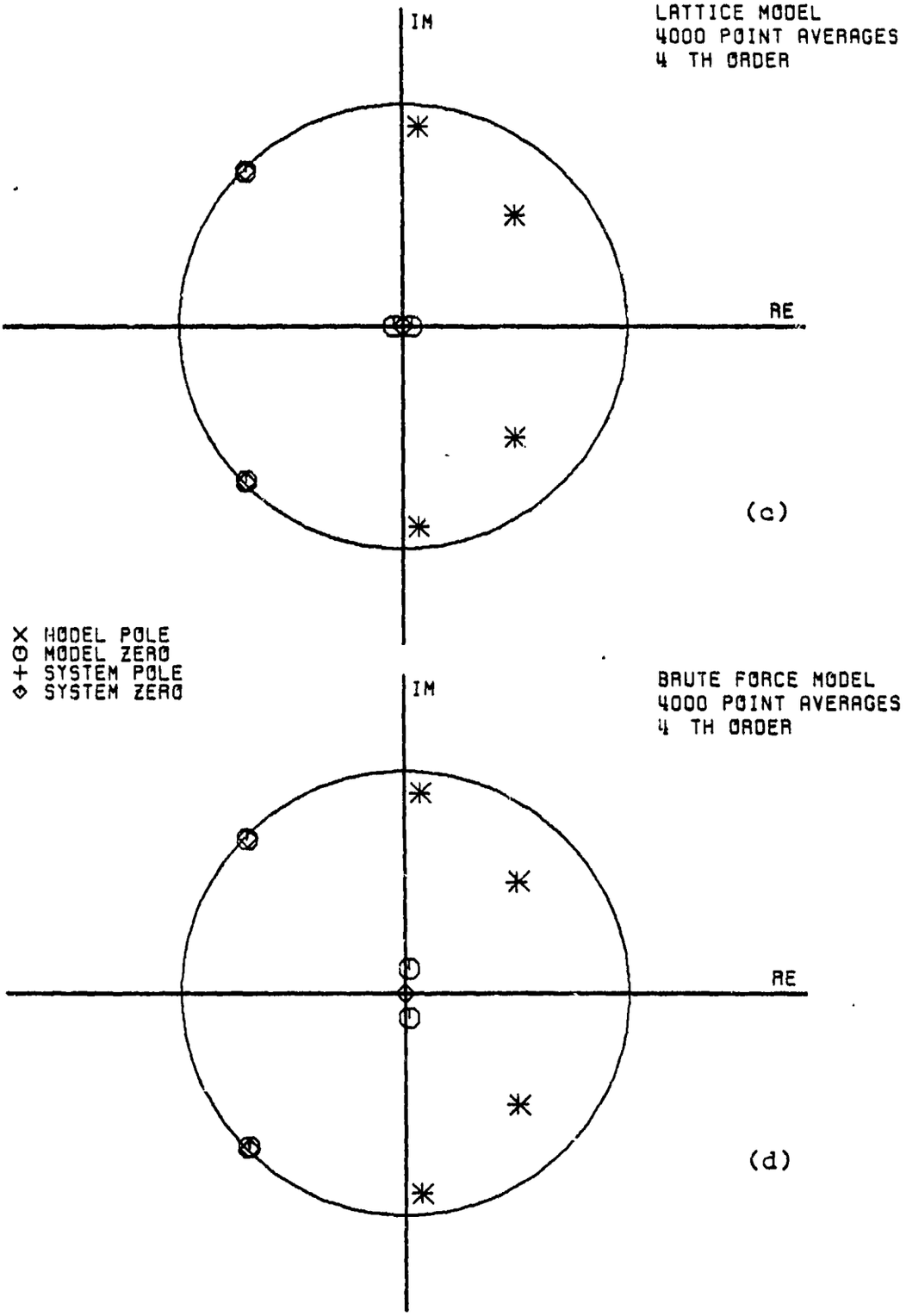
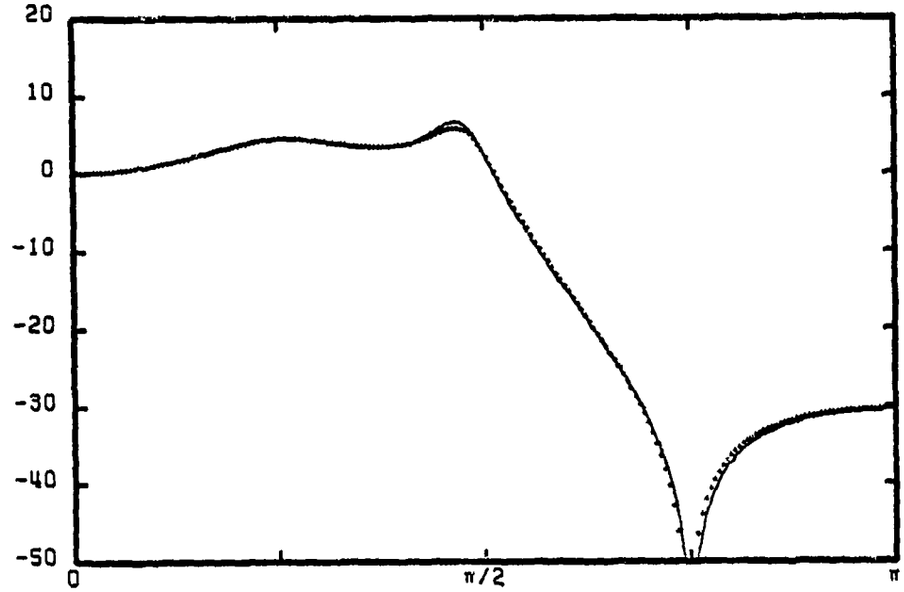


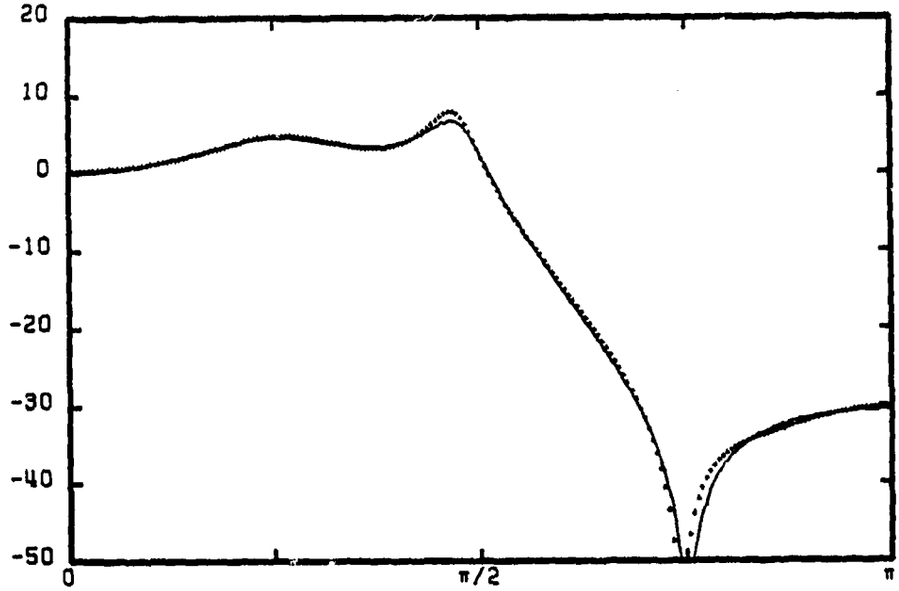
Figure 3.8 con't

the system. Figures 3.9 and 3.10 show such a comparison when sixth order models are obtained for this fourth order system. Ideally, the extra zeros and poles should lie at the origin in the z plane making the additional transfer function coefficients zero. Figure 3.9 shows that for a 200 point averaging interval, the lattice locates the extra roots in the vicinity of the origin while the brute force model does not. When the averaging interval is increased to 4000 points, the extra roots of the lattice model move in toward the origin while those of the brute force model do not. Instead, a zero pole cancellation at some arbitrary location occurs in the brute force model. In all cases investigated during this effort, the lattice method clustered the extra roots in the vicinity of the origin and as the averaging interval was increased to take in more data, these roots were consistently moved in closer to the origin. This property is further evidenced by the plot of mean square equation error as a function of model order shown in Figure 3.11 for a 200 point averaging interval. The MSE for the lattice model flattens out at order four indicating that further increases in model order fail to increase its accuracy. Meanwhile the MSE for the brute force model continues to decrease beyond fourth order as it uses the additional roots to reduce modeling errors caused by innaccuracies in the fourth order model.



(a) $|H(\theta)|_{DB}$ VS θ

— SYSTEM
— 6 TH ORDER LATTICE MODEL
200 POINT AVERAGES



(b) $|H(\theta)|_{DB}$ VS θ

— SYSTEM
— 6 TH ORDER BRUTE FORCE MODEL
200 POINT AVERAGES

Figure 3.9

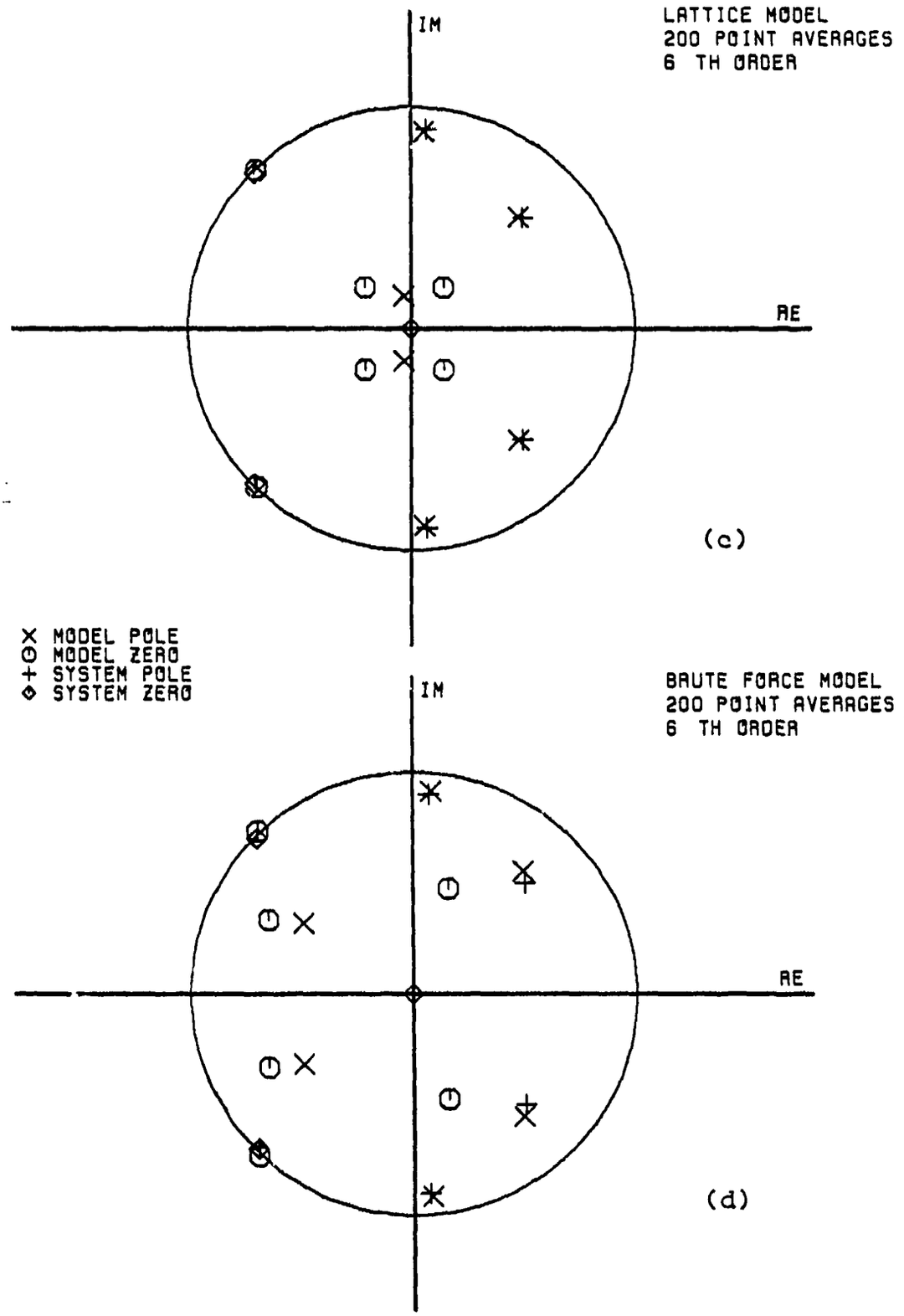
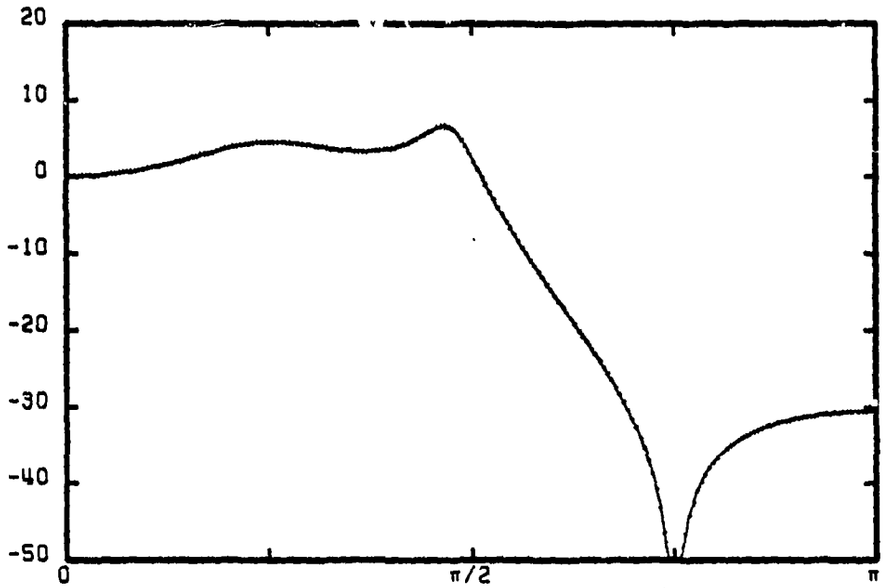
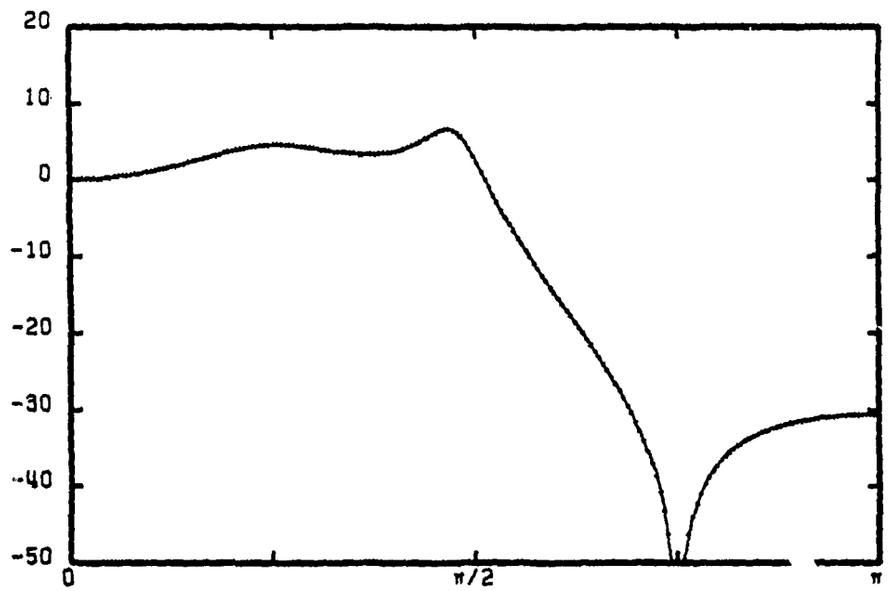


Figure 3.9 con't



(a) $|H(e)|_{DB}$ VS θ

— SYSTEM
— 6 TH ORDER LATTICE MODEL
4000 POINT AVERAGES



(b) $|H(e)|_{DB}$ VS θ

— SYSTEM
— 6 TH ORDER BRUTE FORCE MODEL
4000 POINT AVERAGES

Figure 3.10

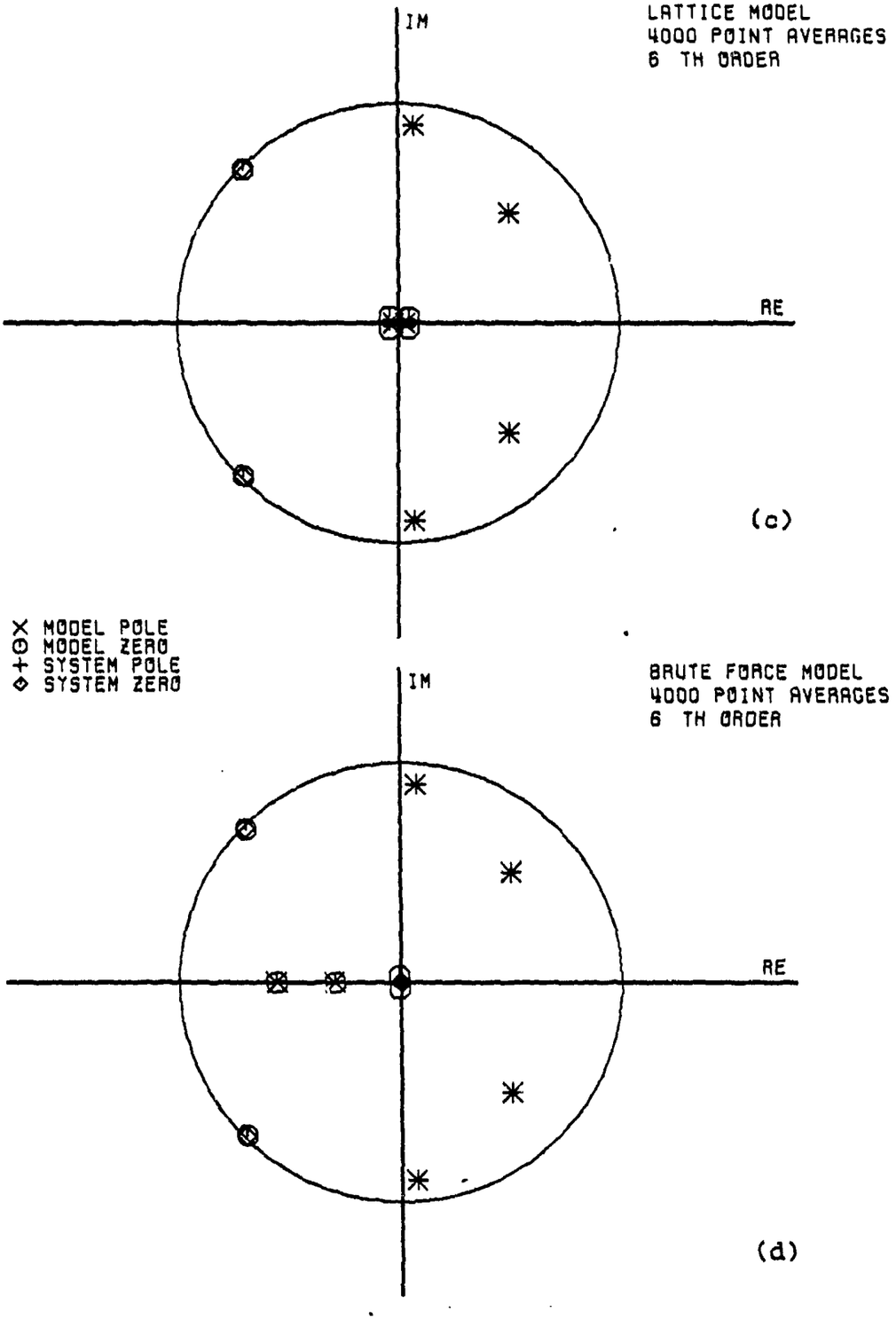


Figure 3.10 con't

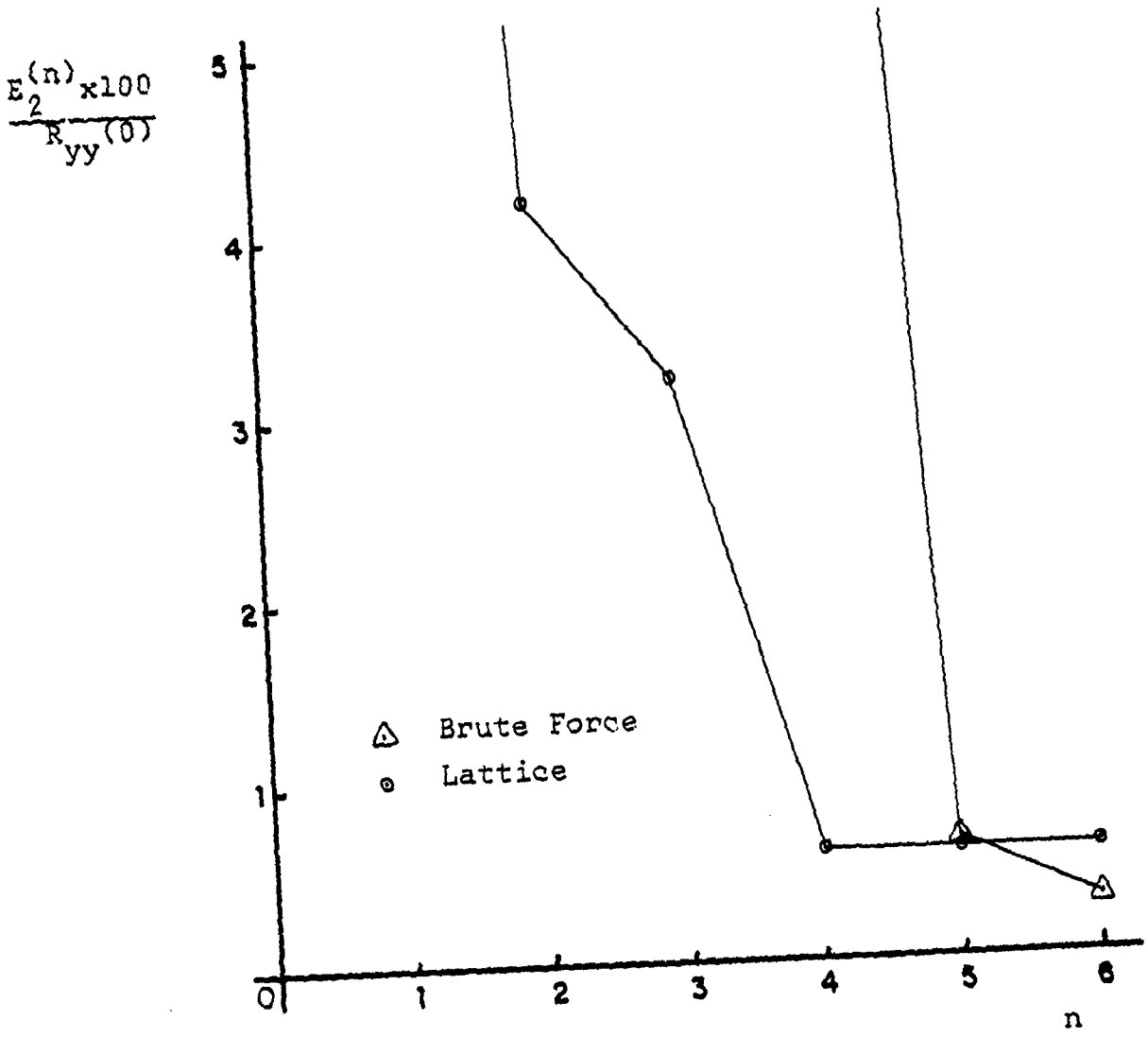


Figure 3.11. Mean square value of equation error (as a percentage of the mean square value of system output) vs model order for lattice and brute force models of system A with 200 point averages.

To investigate the ability of these modeling methods to distinguish roots located near one another in the z plane, a pair of zeros were added to the previous system in close proximity to one of the pole pairs. The characteristics of this system are listed in Table 3.2. Figures 3.12 and 3.13 show the lattice method and brute force modeling results for 200 and 4000 point averaging intervals. With data over only 200 sampling instants, neither method is able to accurately model the effects of the adjacent roots. When the averaging interval is increased, the lattice correctly models the plant while the brute force method does not, and even results in an unstable model. (Figure 3.13b comparing the transfer function magnitudes has been plotted in spite of the model instability.)

To investigate the ability of these zero pole modeling methods to model systems that are actually all zero or all pole, the systems listed in Tables 3.3 and 3.4 were used. The modeling results are shown in Figures 3.14, 3.15, 3.16 and 3.17.

The conclusions drawn from the results of this experimental study are as follows:

1) For short data lengths, the lattice filter method provides more accurate models than does the brute force modeling method.
2) When the system is overmodeled using the lattice method, the excess roots are clustered about the origin in the z plane and as the averaging interval is increased and more data taken in, these roots move

TABLE 3.2

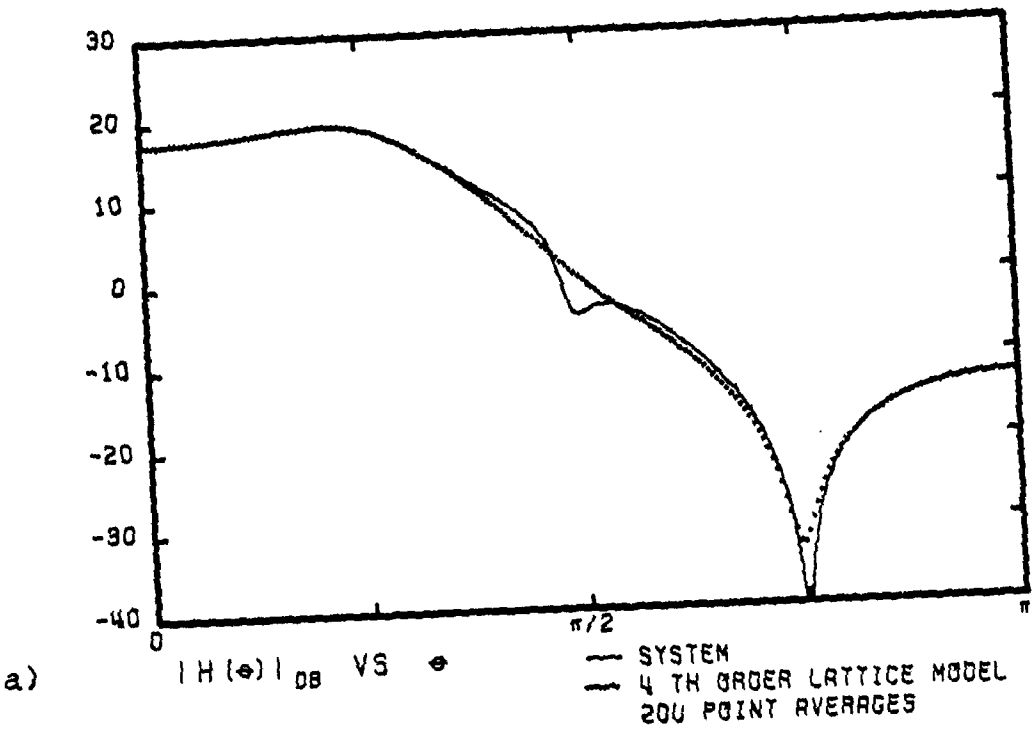
SYSTEM B

TRANSFER FUNCTION COEFFICIENTS

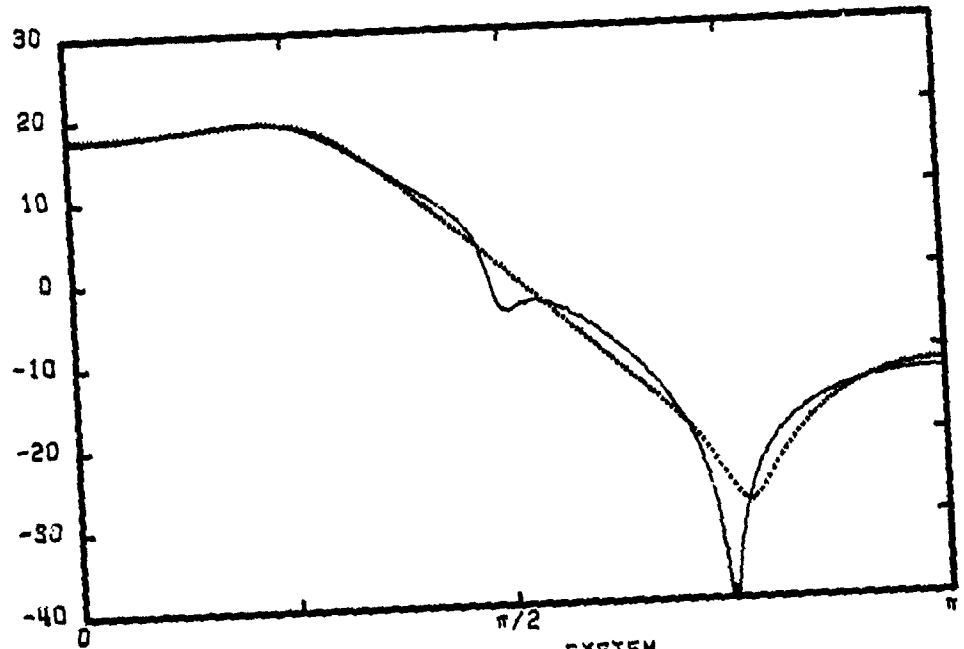
NUMERATOR		DENOMINATOR	
A(0)=	1.00000		
A(1)=	1.34000	B(1)=	1.14000
A(2)=	1.79940	B(2)=	-1.45490
A(3)=	1.20600	B(3)=	0.88490
A(4)=	0.88533	B(4)=	-0.40745

ROOT LOCATIONS

ZEROS		POLES	
RE	IM	RE	IM
-0.70000	0.70000	0.50000	0.50000
-0.70000	-0.70000	0.50000	-0.50000
0.03000	0.95000	0.07000	0.90000
0.03000	-0.95000	0.07000	-0.90000



(a) $|H(\theta)|_{DB}$ VS θ



(b) $|H(\theta)|_{DB}$ VS θ

Figure 3.12

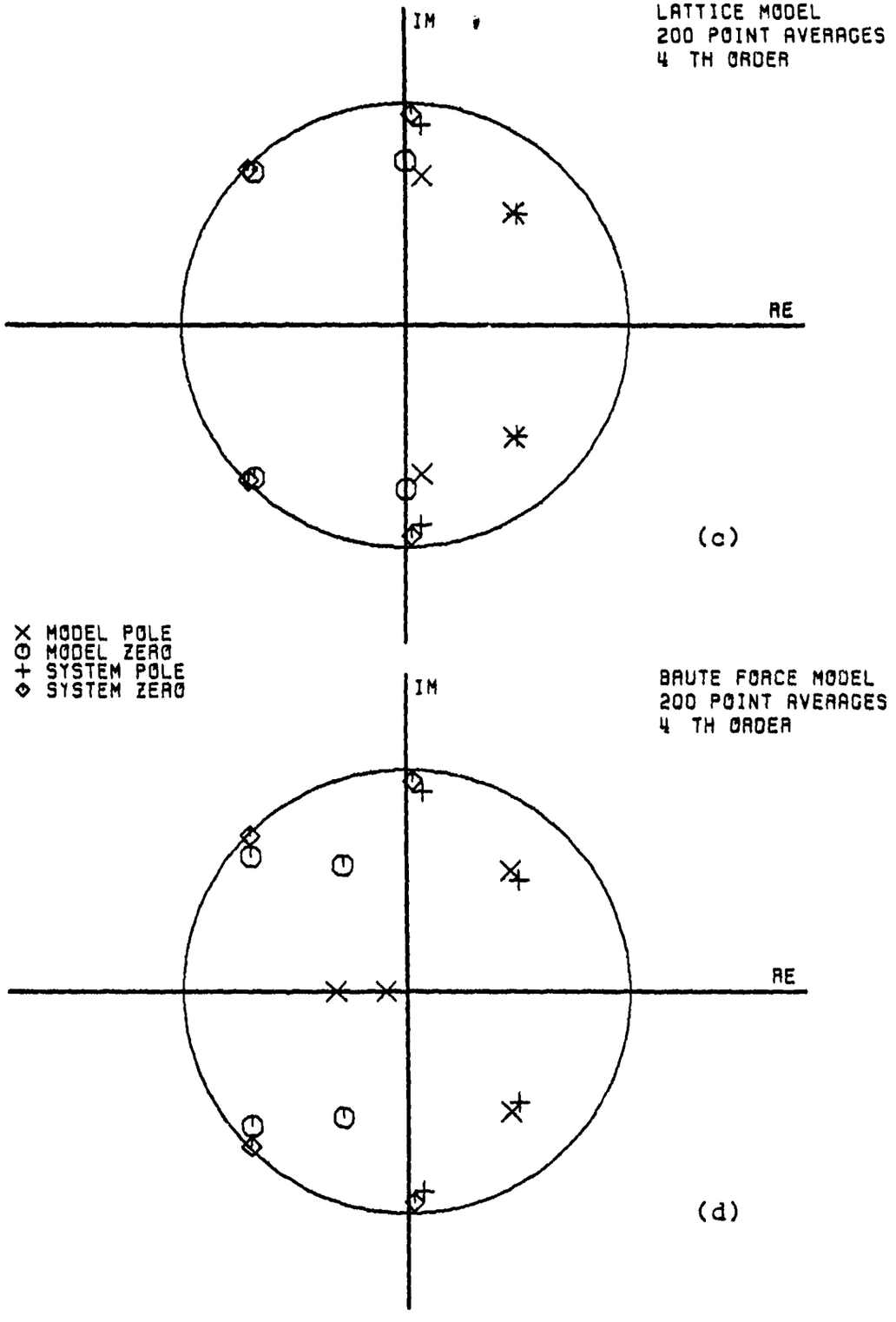
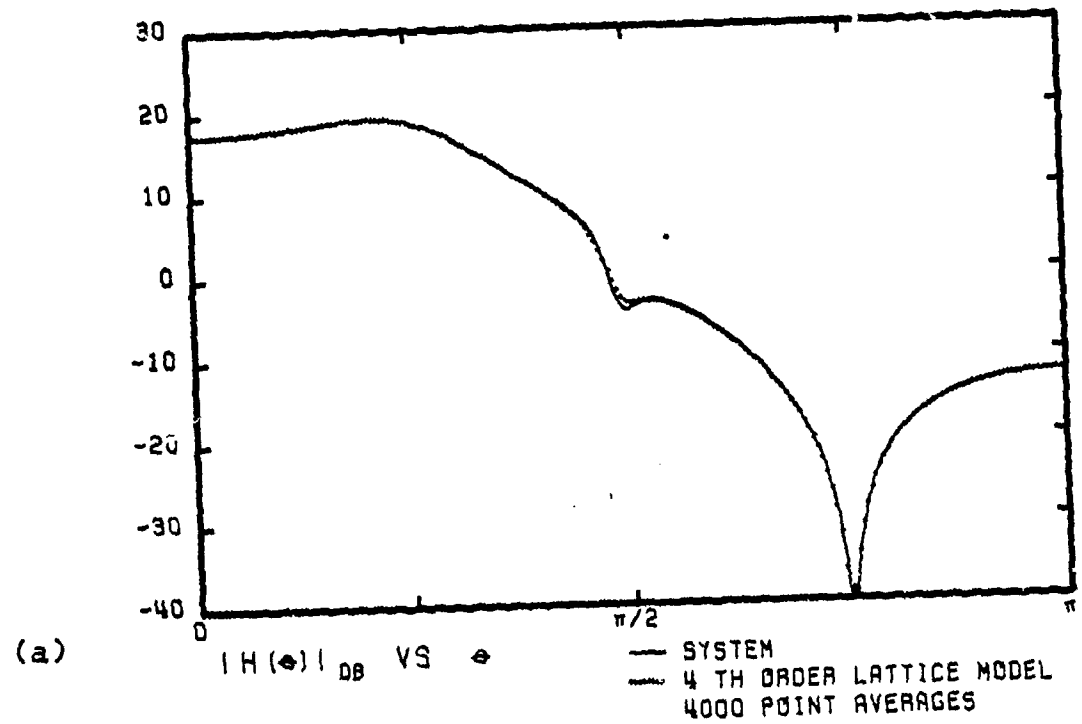


Figure 3.12 con't


30
20
10
0
-10
-20
-30
-40
0
π/2
π
(a)
| H (e) | DB VS e
SYSTEM
4 TH ORDER LATTICE MODEL
4000 POINT AVERAGES

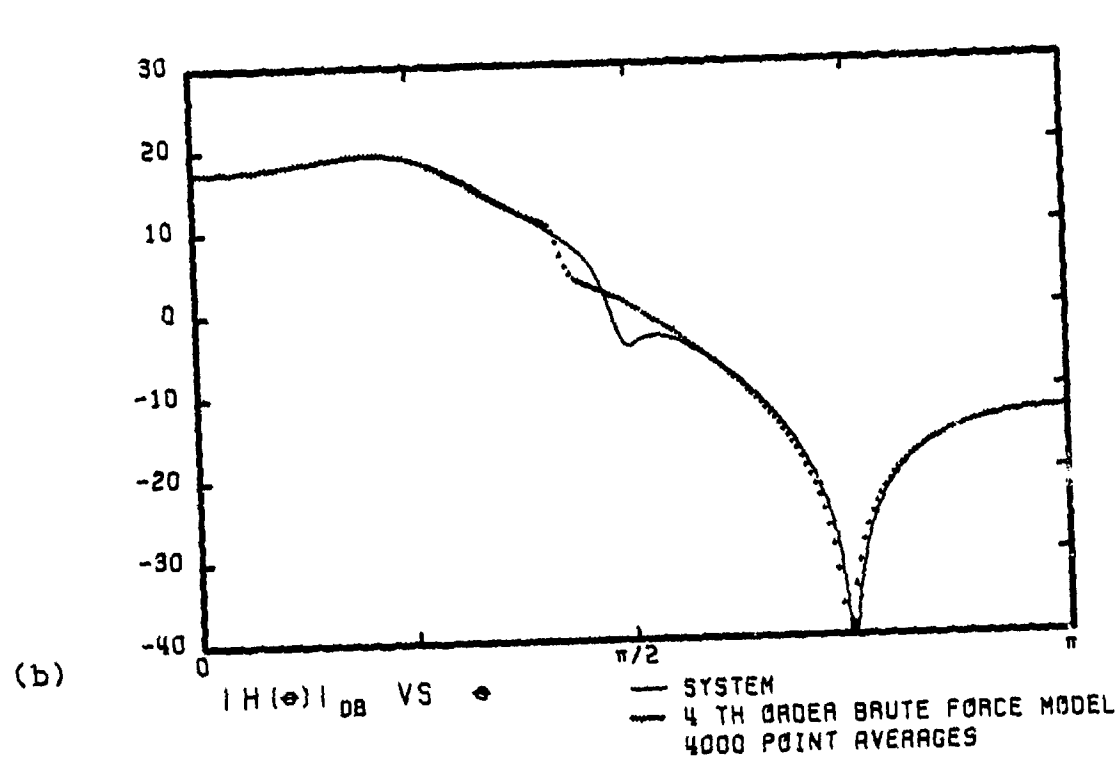

30
20
10
0
-10
-20
-30
-40
0
π/2
π
(b)
| H (e) | DB VS e
SYSTEM
4 TH ORDER BRUTE FORCE MODEL
4000 POINT AVERAGES

Figure 3.13

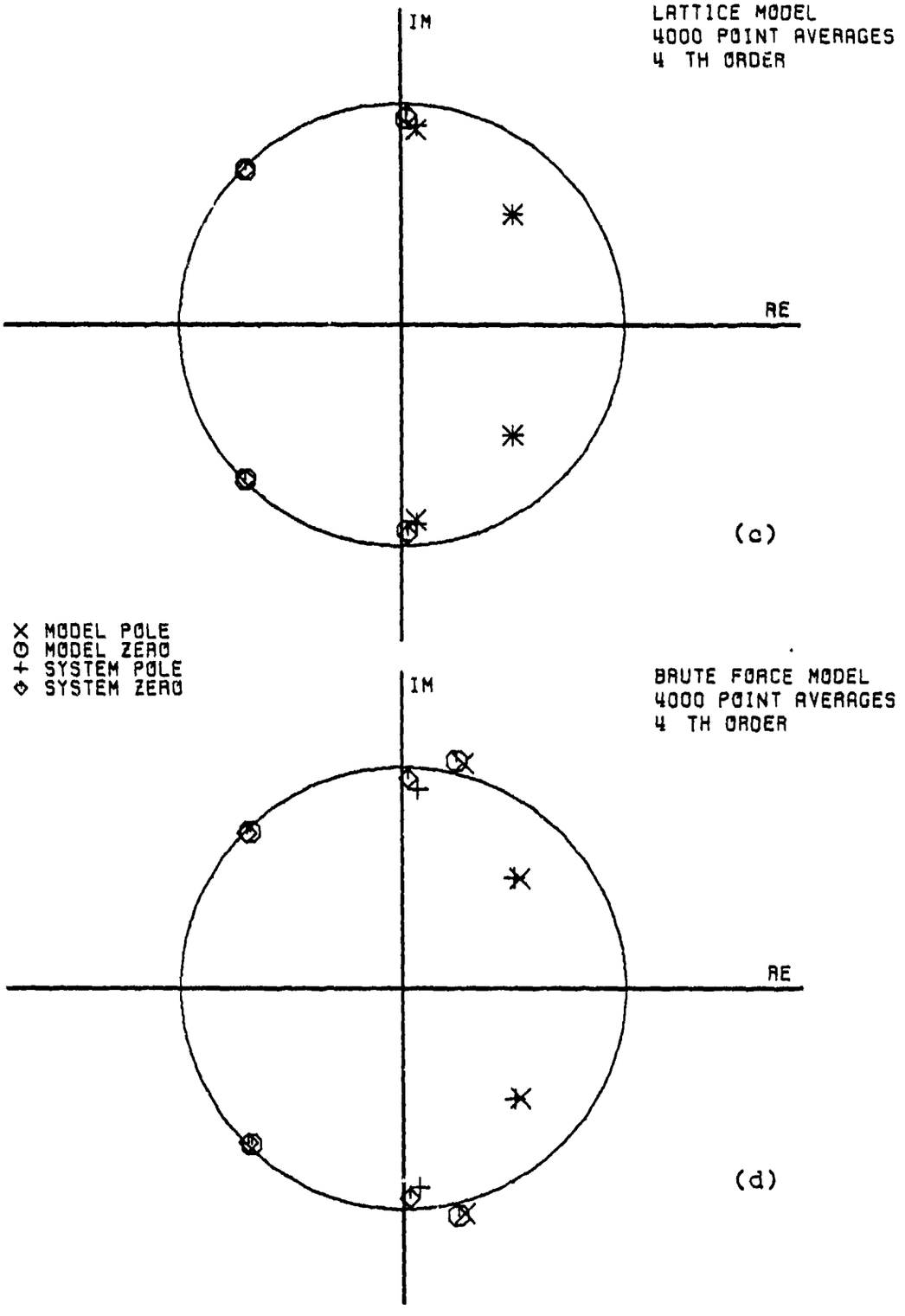


Figure 3.13 con't

TABLE 3.3

SYSTEM C

TRANSFER FUNCTION COEFFICIENTS

NUMERATOR		DENOMINATOR	
A(0) =	1.00000		
A(1) =	-1.50000	B(1) =	0.0
A(2) =	0.62500	B(2) =	0.0

ROOT LOCATIONS

ZEROS		POLES	
RE	IM	RE	IM
0.75000	0.25000	0.0	0.0
0.75000	-0.25000	0.0	0.0

TABLE 3.4

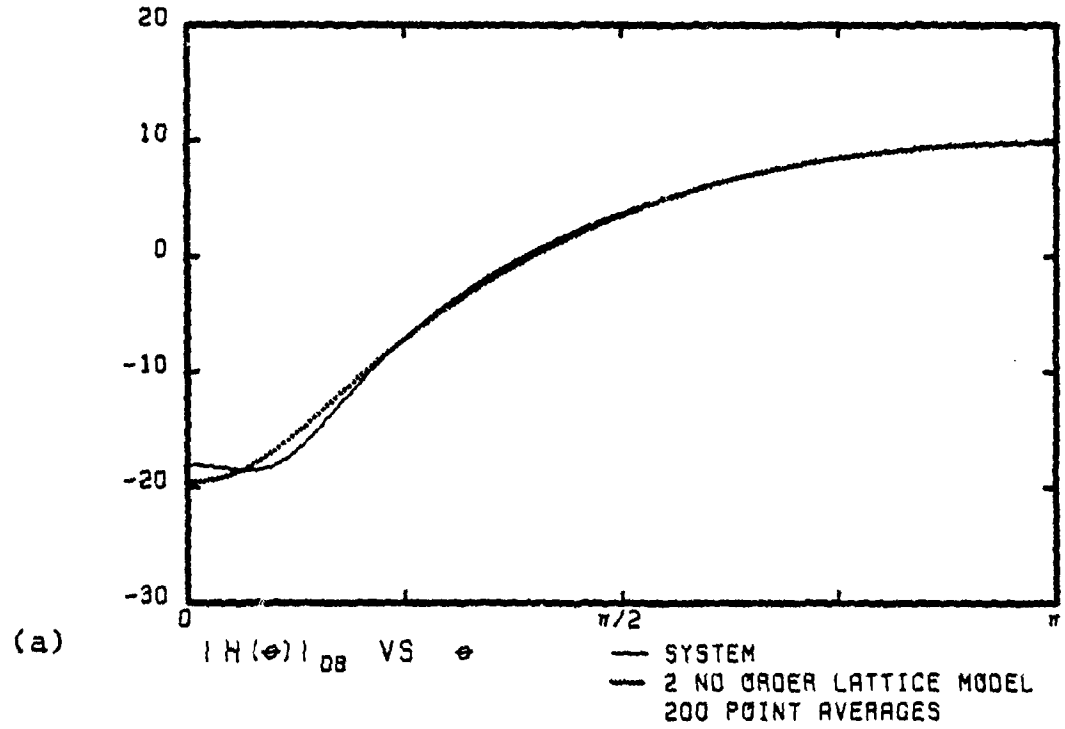
SYSTEM D

TRANSFER FUNCTION COEFFICIENTS

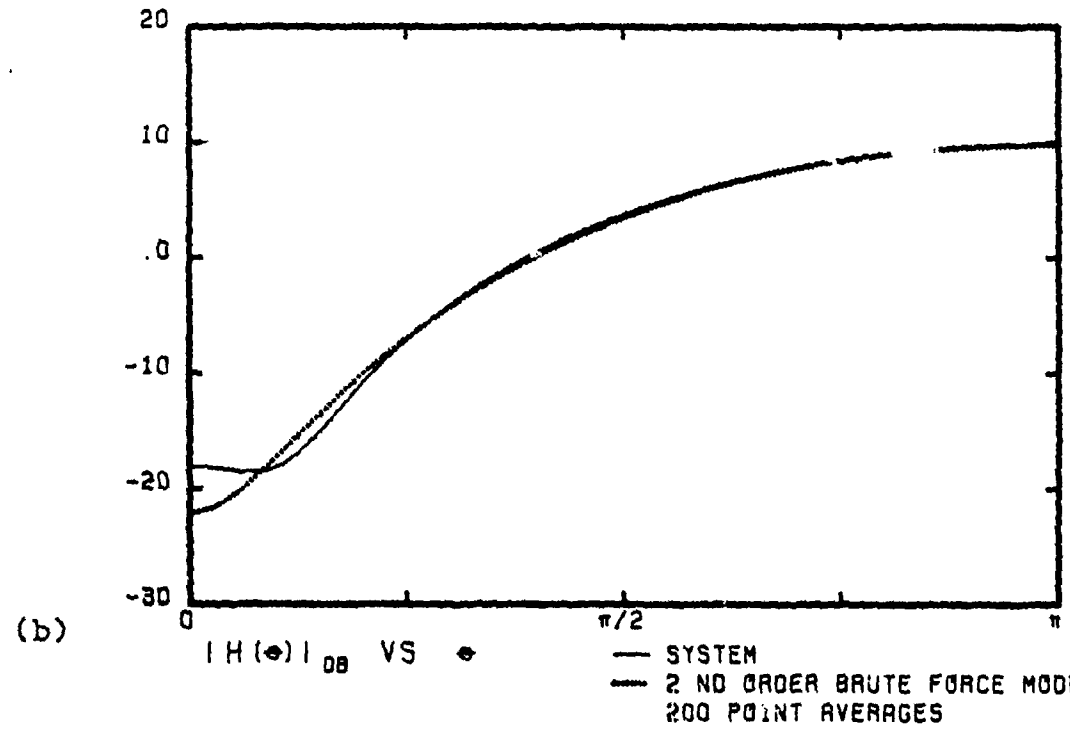
NUMERATOR		DENOMINATOR	
A(0) =	1.00000		
A(1) =	0.0	B(1) =	1.50000
A(2) =	0.0	B(2) =	-0.62500

ROOT LOCATIONS

ZEROS		POLES	
RE	IM	RE	IM
0.0	0.0	0.75000	0.25000
0.0	0.0	0.75000	-0.25000



(a) $|H(\omega)|_{dB}$ VS ω



(b) $|H(\omega)|_{dB}$ VS ω

Figure 3.14

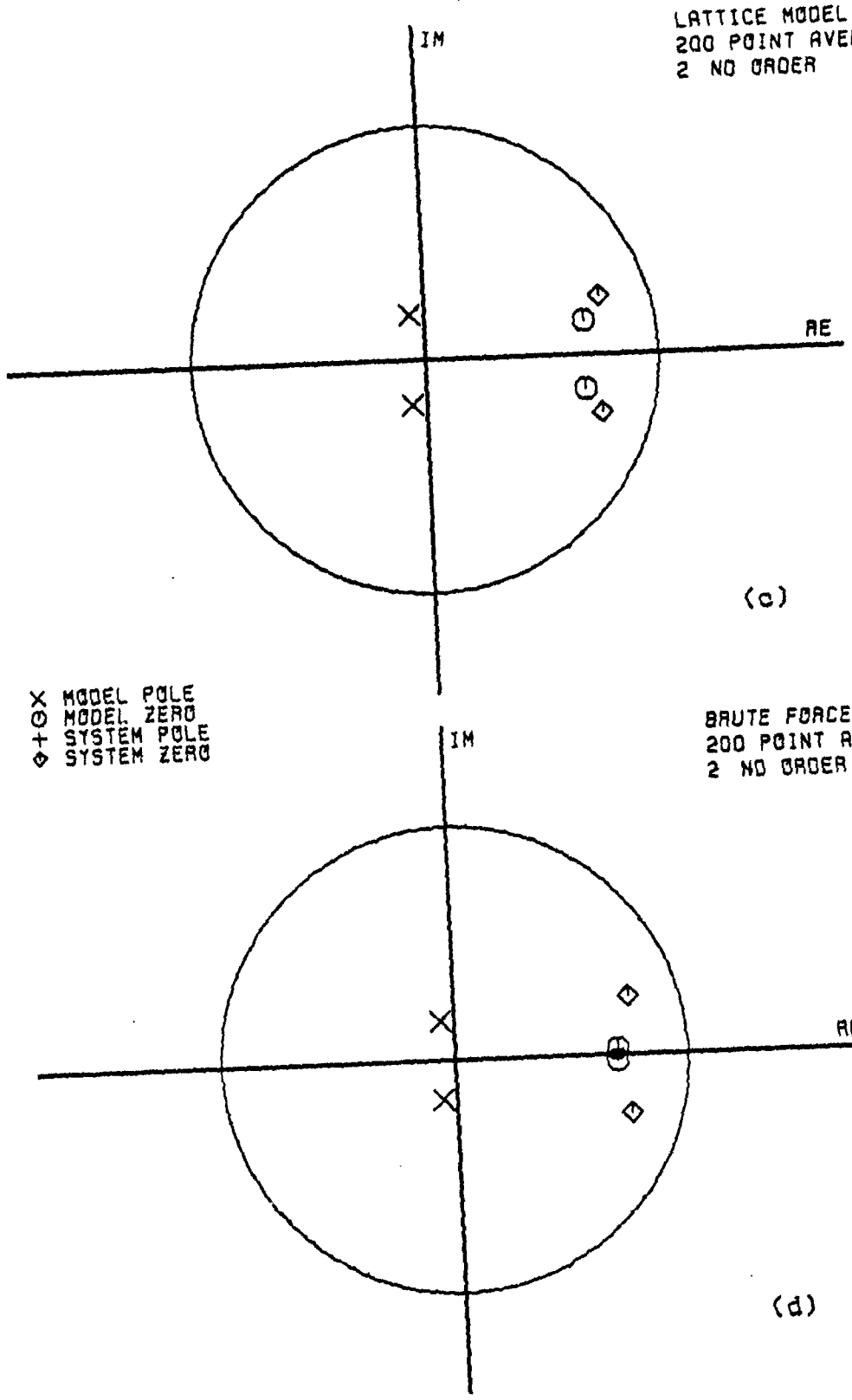
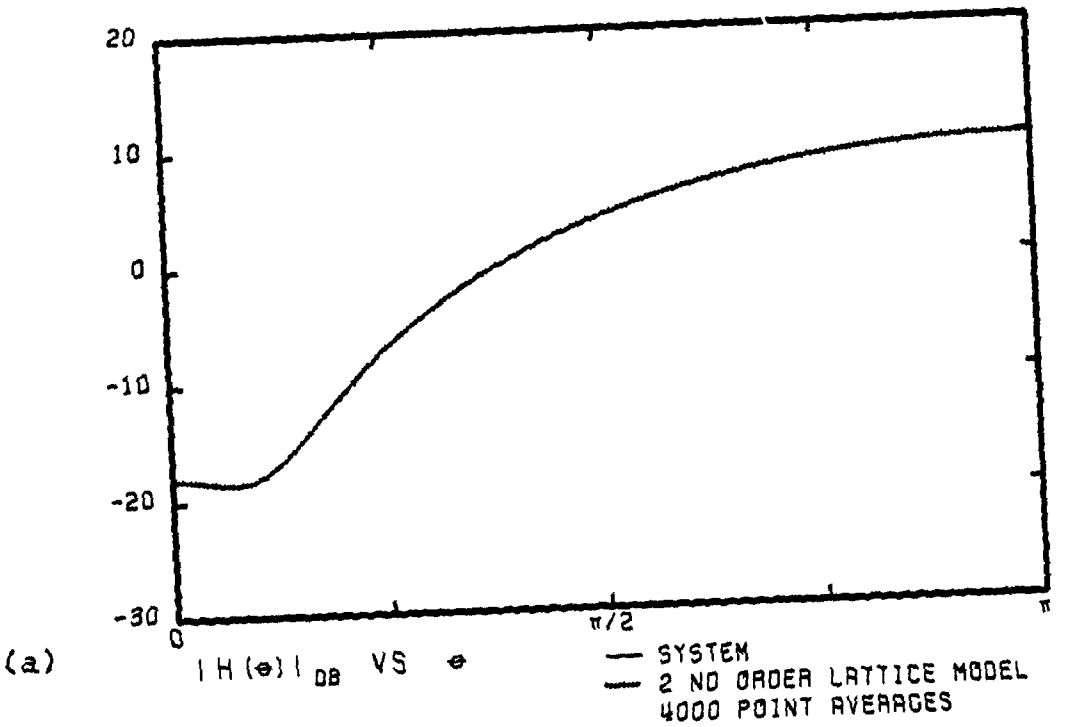
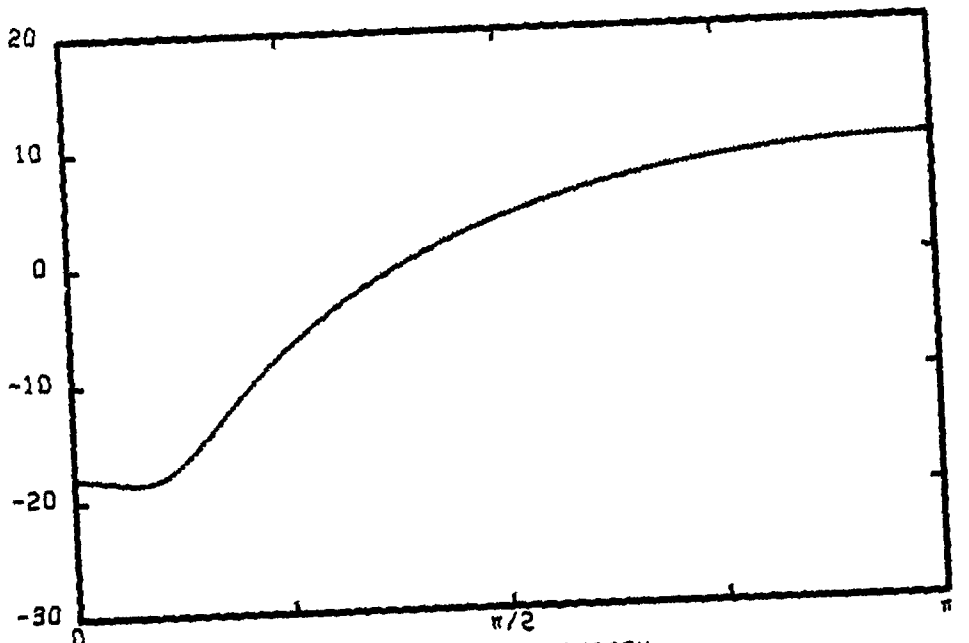


Figure 3.14 con't



(a) $|H(\theta)|_{DB}$ VS θ



(b) $|H(\theta)|_{DB}$ VS θ

Figure 3.15

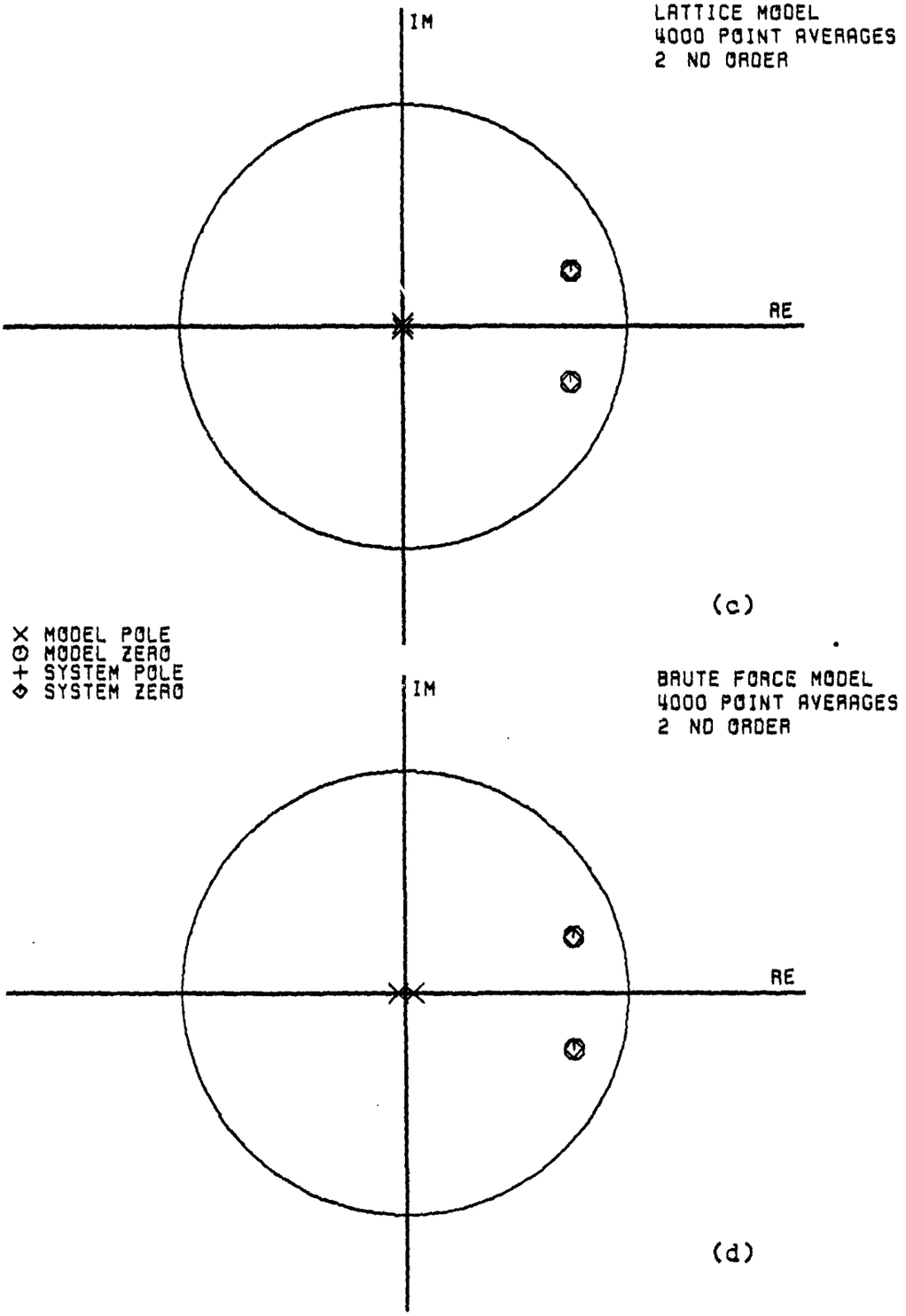
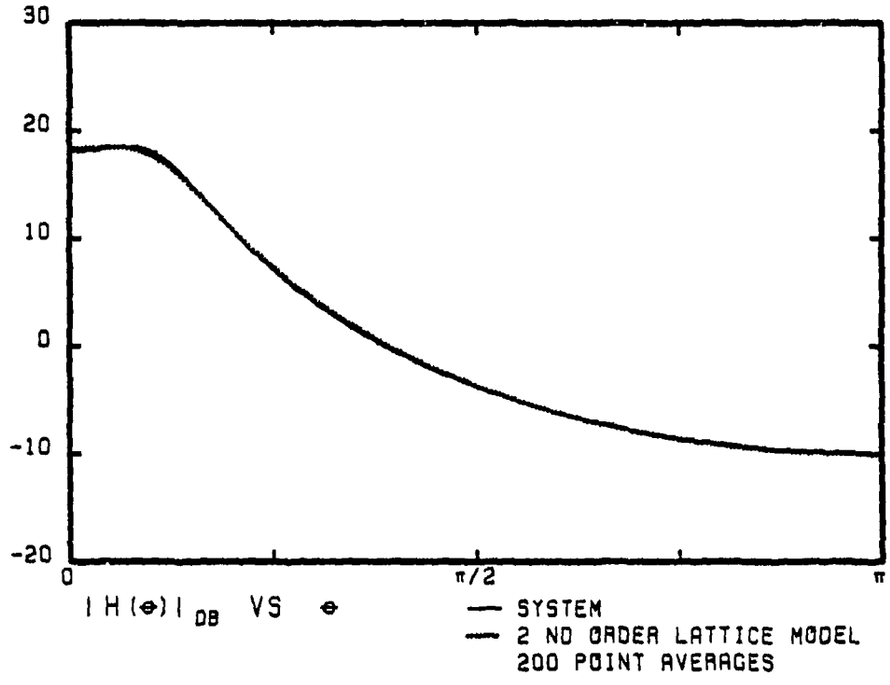


Figure 3.15 con't

(a)



(b)

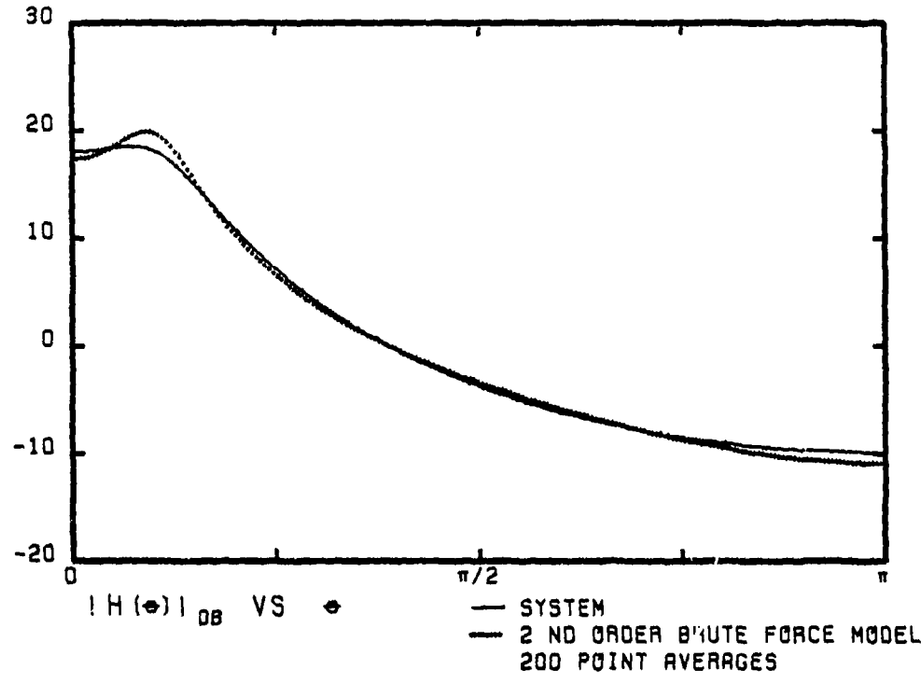


Figure 3.16

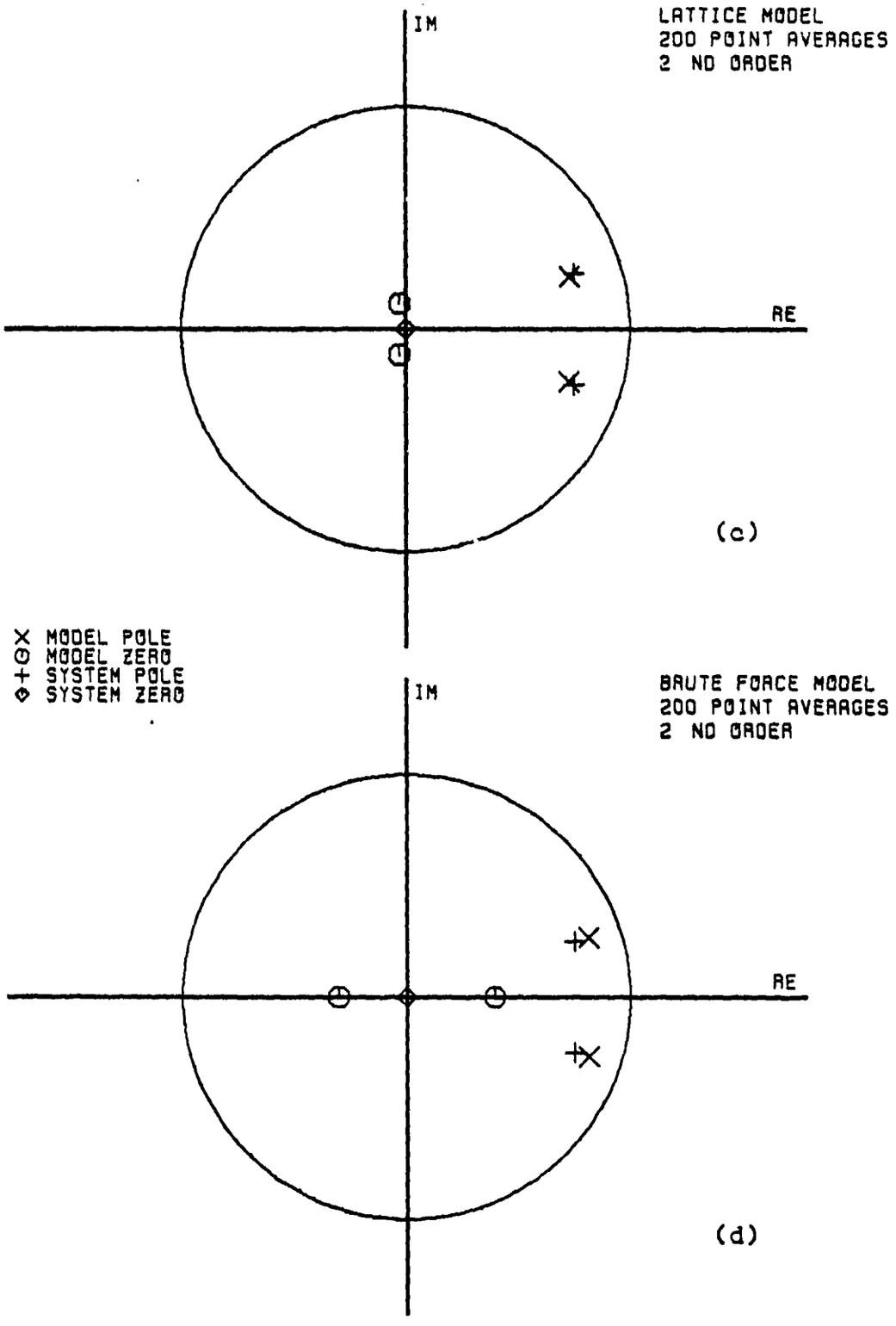
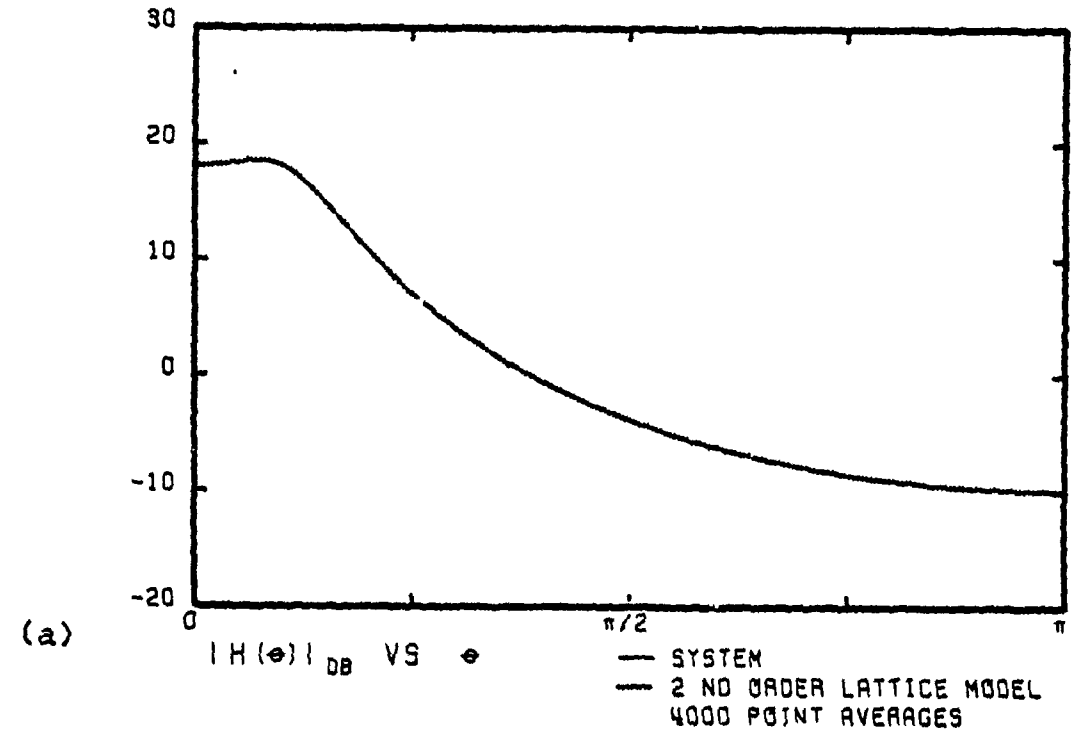


Figure 3.16 con't


30
20
10
0
-10
-20
0
π/2
π
(a)
| H (e) | DB VS e
— SYSTEM
— 2 ND ORDER LATTICE MODEL
4000 POINT AVERAGES

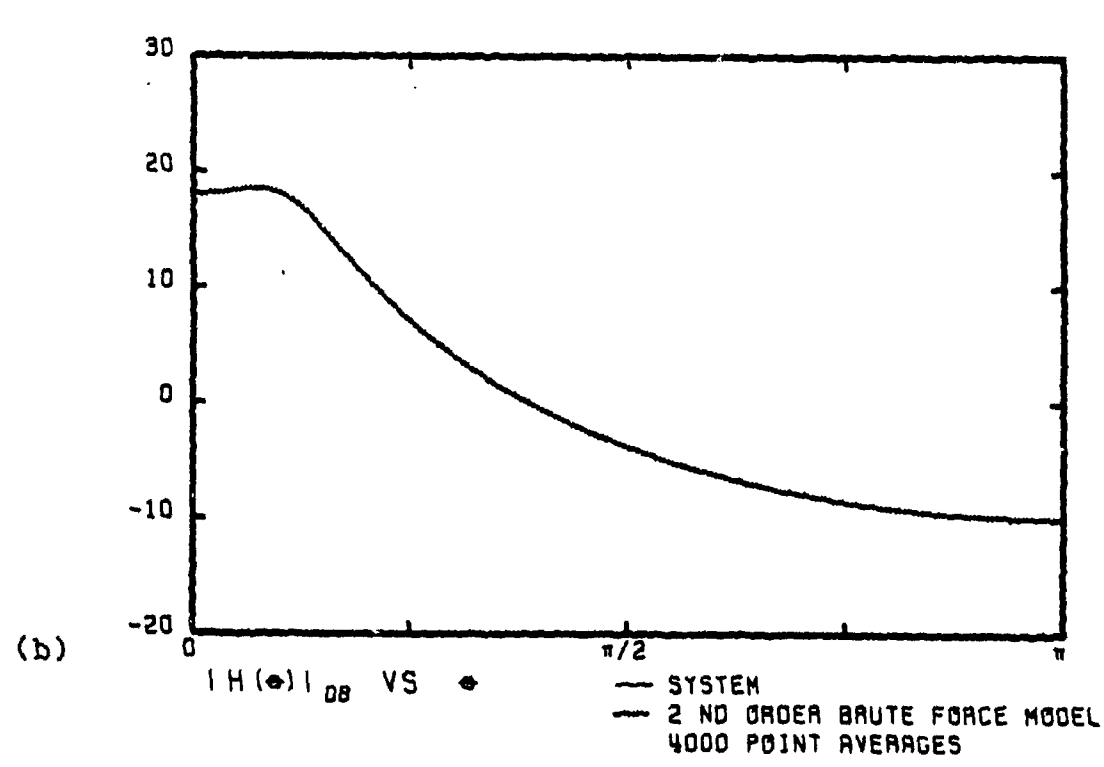

30
20
10
0
-10
-20
0
π/2
π
(b)
| H (e) | DB VS e
— SYSTEM
— 2 ND ORDER BRUTE FORCE MODEL
4000 POINT AVERAGES

Figure 3,17

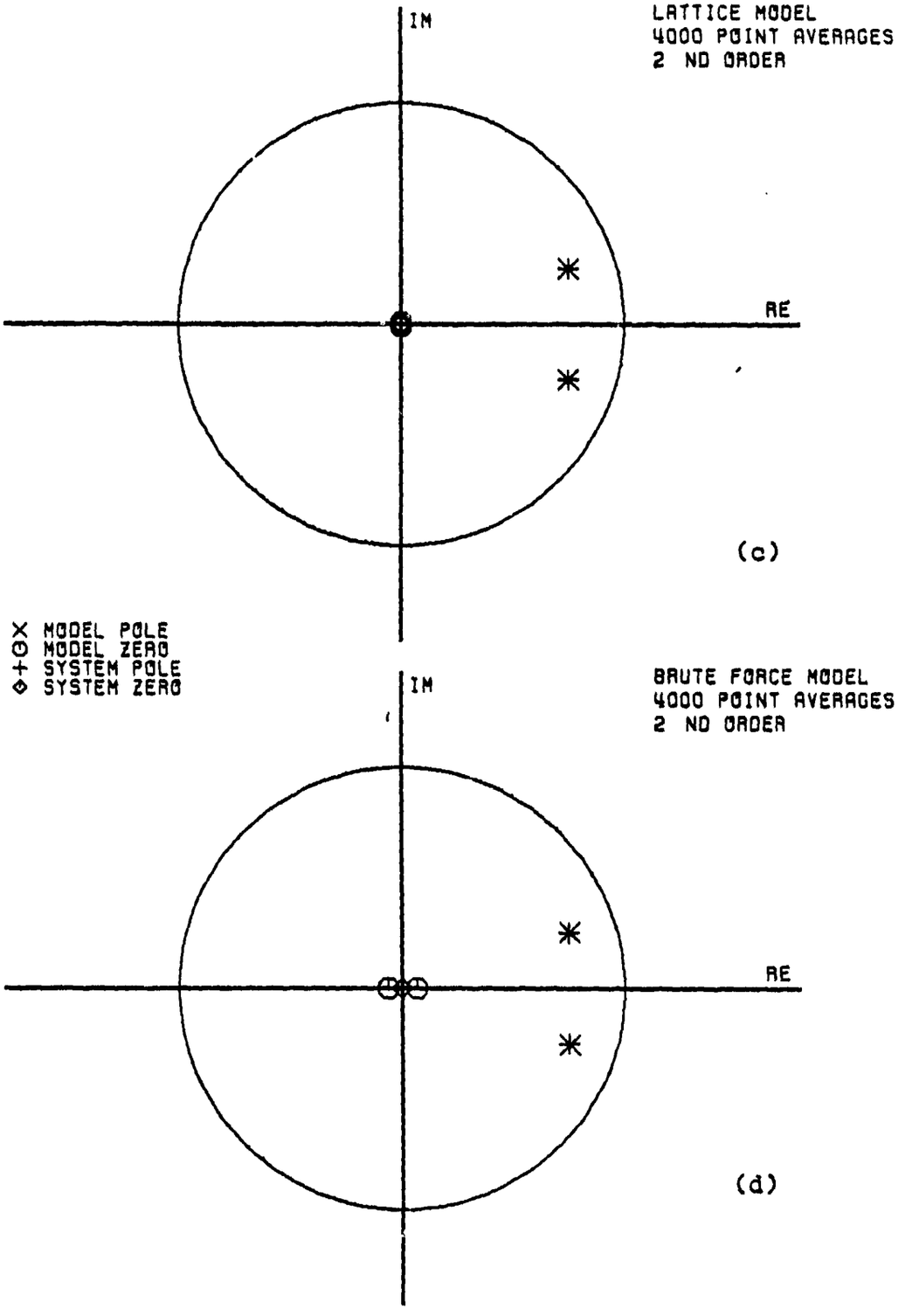


Figure 3.17 con't

consistently in toward the origin. This also forces the excess transfer function coefficients and reflection coefficients to be very small (ideally zero), clearly indicating that the model order is higher than necessary. The brute force method, on the other hand, scatters the excess roots throughout the z plane and produces cancellations of the excess zeros and poles. This results in nonzero values for the excess transfer function coefficients.

3) The MSE as a function of model order is generally better behaved for the lattice method than for the brute force method, decreasing rapidly until the correct model order is reached and then failing to decrease substantially as the model order is increased further.

Two qualitative explanations for the improved performance of the lattice filter modeling method are offered. The first is that the data is not windowed in the lattice method while a window that is nonzero only over the span of the averaging interval must be used in the brute force method to maintain even symmetry in the autocorrelation function estimates. The effects of this difference between the two modeling methods should be most noticeable for short data lengths, and become less evident as the length of the averaging interval is increased. The second possible cause for the lattice method's better performance is that the actual output sequences of the n-th order lattice are used to calculate the coefficients at the (n+1)-st order stage. In this manner,

modeling errors that have occurred in the n-th order lattice can be compensated for to some extent in the (n+1)-st order stage. No similar phenomenon is evident in the brute force modeling approach. Consider the differences between the Levinson algorithm (which is equivalent to the matrix inversion method) of equations (A.16) and the lattice method given by equations (2.49). Both methods calculate the corrections that must be made to the n-th order model to obtain the (n+1)-st order model in terms of $\underline{K}^{(n+1)}$ and $\underline{\bar{K}}^{(n+1)}$ and theoretically,

$$\varepsilon\{\underline{e}^{(n)}(k)\ \underline{\bar{e}}^{(n)}(k-1)^T\} = \underline{R}_{xx}(n+1)^T - \underline{\rho}^{(n)^T}\ \underline{\bar{D}}^{(n)}$$

When correlations are estimated by averaging over finite intervals however this equality will not in general be satisfied making the two methods different. The Levinson algorithm will estimate the correction terms to be added to the optimum n-th order model while the lattice method estimates the correction terms to be added to the estimated n-th order model actually obtained.

The improved performance of the lattice method is not achieved without cost, however. The method is made computationally expensive by the need to store the system input and output sequences and the lattice prediction error sequences and pass them through successive stages of the lattice as it is built up in order. The computational complexity of the two methods is compared in Table 3.5.

Table 3.5. Computational requirements for batch processing ARMA modeling of an N-th order system using P samples of the system input and output.

Number of Correlation Estimates Required	4N+3 averaged over P data points	4N+3
Matrix Inversions	1 - dimension 2N+1	2N-dimension 2 1-dimension 1
Data Storage Requirements	N.A.	2P samples
Computations To Pass Data Through The Filter	N.A.	8NP multiplications 4NP additions

E. ADAPTIVE LATTICE ARMA MODELING

In addition to the batch processing method described in the previous section, the lattice ARMA analysis model can be implemented adaptively as well. The adaptive lattice solution for the multichannel AR lattice, which solves most of the ARMA modeling problem, has already been described in chapter II. To make the lattice ARMA model adaptive, only an adaptive solution for the gain term a(0) need be added. To avoid ambiguity, the time varying adaptive estimate of this term at time k is denoted by $a_0(k)$. Applying an LMS adaptive algorithm it follows that

$$a_0(k+1) = a_0(k) - \mu_0 \nabla(k) \tag{3.35}$$

and using equation (3.27d) to form an instantaneous estimate of the gradient yields

$$\hat{\nabla}(k) = -2\, e_u(k)[e_y(k) - a_0(k) e_u(k)] \tag{3.36}$$

$$= -2\, e_u(k)\, e_o(k)$$

so that

$$a_0(k+1) = a_0(k) + 2\mu_0\, e_u(k)\, e_0(k) \tag{3.37}$$

Here it is clear that

1) $e_0(k)$ is analagous to the error signal.

2) $e_u(k)$ is analagous to the input signal.

3) $e_y(k)$ is analagous to the desired signal.

so that for stability μ_0 must satisfy

$$0 < \mu_0 < \frac{1}{\varepsilon\{e_u(k)^2\}} \tag{3.38}$$

Once again, however, the mean square value of $e_u(k)$ may vary from stage to stage and over time as the two channel AR lattice adapts making it appropriate to apply relations similar to equations (2.58) and (2.59)

$$\mu_0\ (k) = \frac{\alpha}{\sigma_0(k)} \tag{3.39a}$$

$$\sigma_0\ (k) = (1-\alpha)\ \sigma_0\ (k-1) + \alpha\ e_u(k)^2 \tag{3.39b}$$

where α is the normalized adaptive step size and the dependence on order is implicit (a superscript (n) could be used on a_0, σ_0, ∇, and all the error terms in equations (3.35) through (3.39) to explicitly denote their dependence on the order of the solution).

This adaptive lattice ARMA modeling scheme was implemented and the results of its use in modeling system A described in Table 3.1 are presented here. A normalized adaptive step size of $\alpha = .05$ was used in the following simulations and

the results represent an ensemble average of one hundred trials. Unity variance white noise was used as the system input. A flow diagram of the procedure is shown in Figure 3.18.

Figure 3.19 shows a plot of the mean square equation error as a function of time for a fourth order model while it adapts and Figure 3.20 shows the behavior of one term in the K matrix, $k_{11}^{(n)}$, and one term in the $\bar{K}$ matrix, $\bar{k}_{11}^{(n)}$, at the first five lattice stages, $1 \leq n \leq 5$. These graphs of the k_{11} terms clearly show the successive stage by stage manner in which the lattice model adapts. Figure 3.21 shows a comparison of the transfer function magnitude and root locations of the system with those of the fourth order adaptive model after 1000 and 2000 iterations. Figure 3.22 shows the same comparison in the overmodeled case when a sixth order model is obtained for this fourth order system.

While these results show that the adaptive solution performs well and is a viable alternative to the batch processing solution, Figures 3.22c and 3.22d show that one advantageous characteristic of the batch solution has not carried over. The excess roots in the overmodeled case are not tightly clustered in the vicinity of the origin indicating that the excess transfer function coefficients are not near zero. In examining Figure 3.20 it is also evident that the convergence of the reflection coefficients at the overmodeled lattice stage ($k_{11}^{(5)}$ and $\bar{k}_{11}^{(5)}$) towards their true value of zero is quite slow. This relatively slow tracking

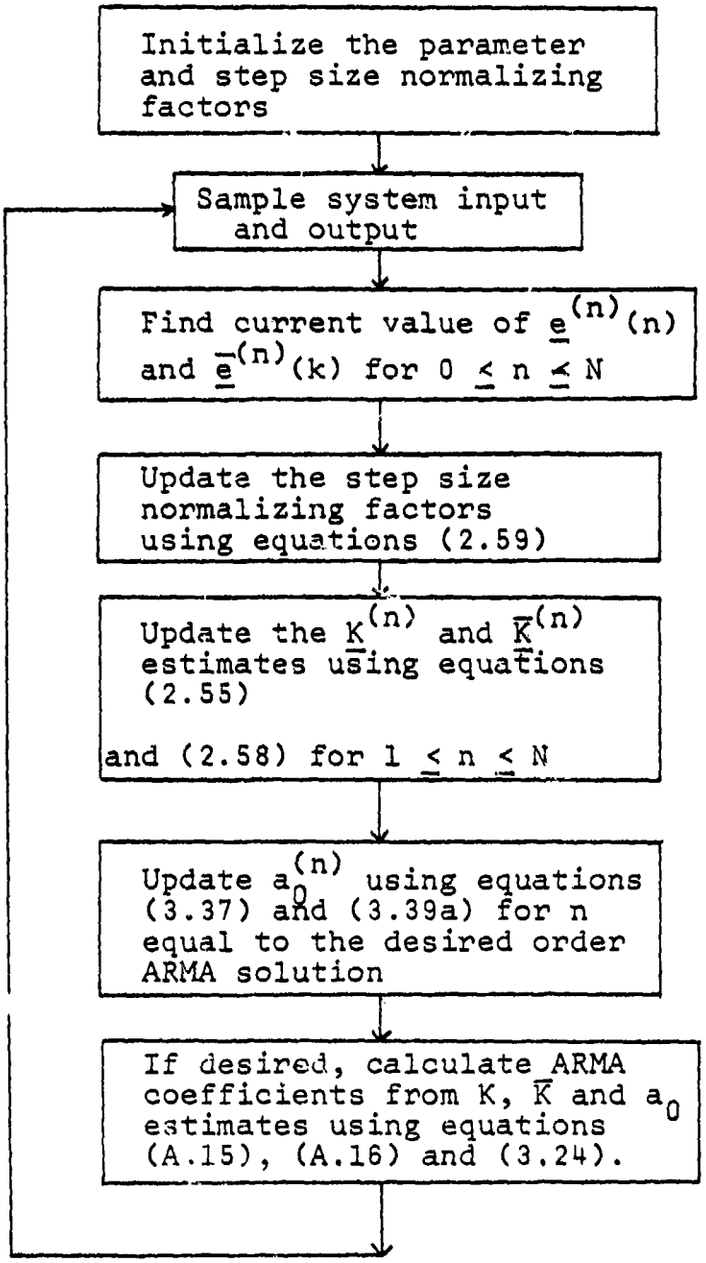


Figure 3.18. Flow diagram of adaptive lattice ARMA solution.

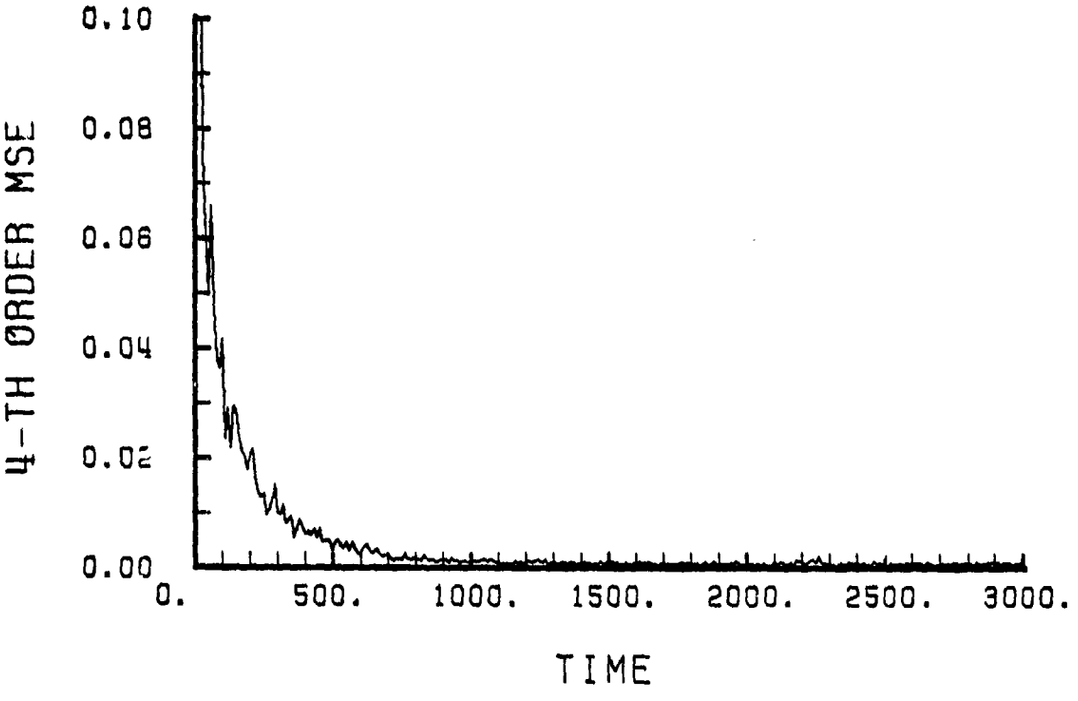


Figure 3.19

Mean square value of equation error for the fourth order adaptive lattice model as a function of time.

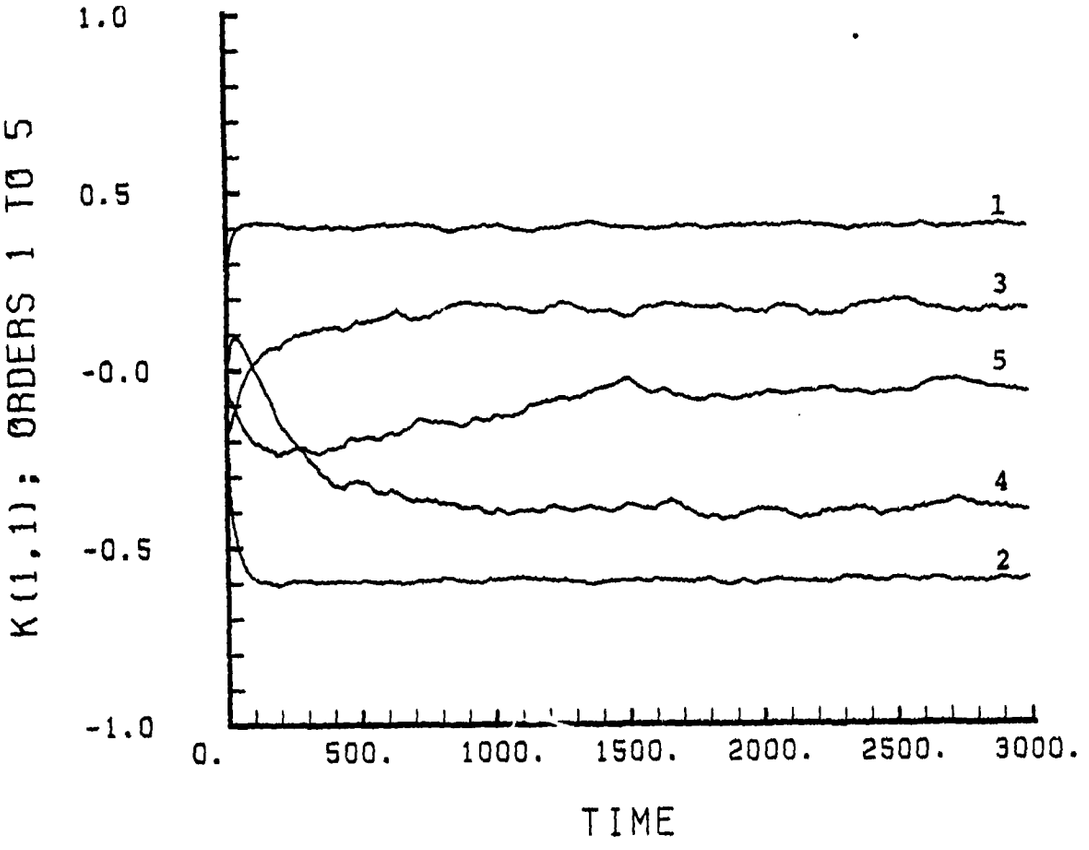


Figure 3.20a

K(1,1) term from the forward reflection coefficient matrices at the first five stages of the adaptive lattice model as a function of time.

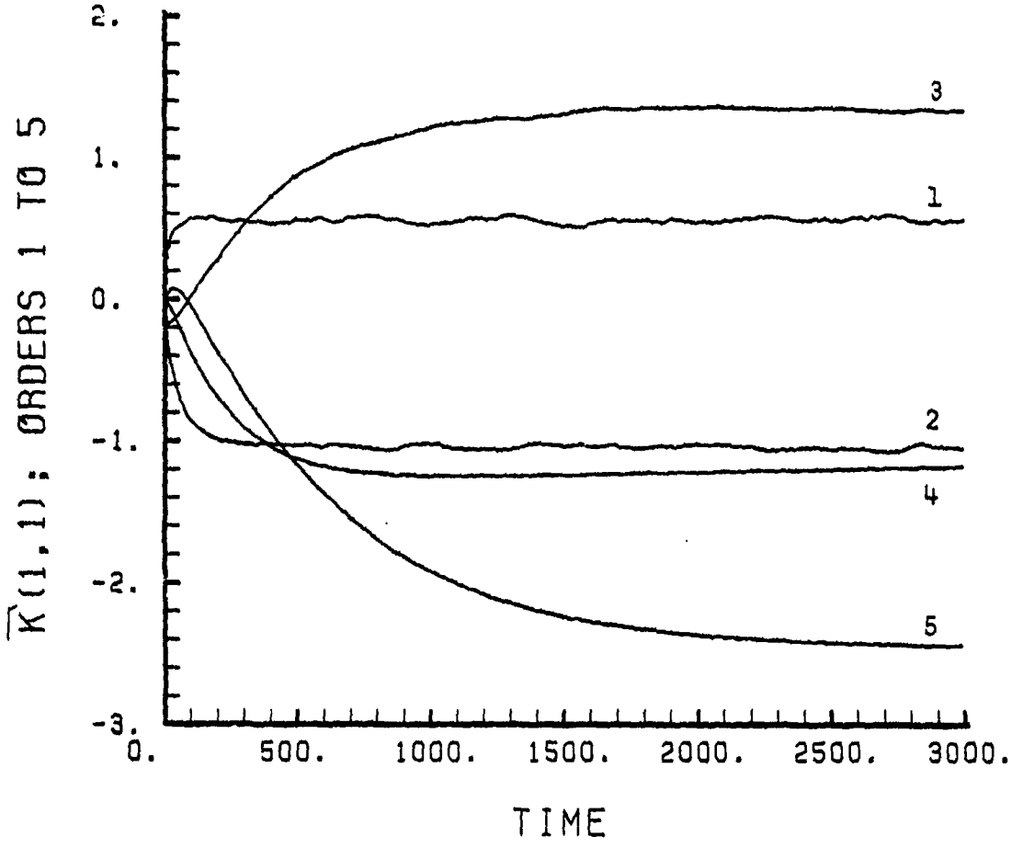
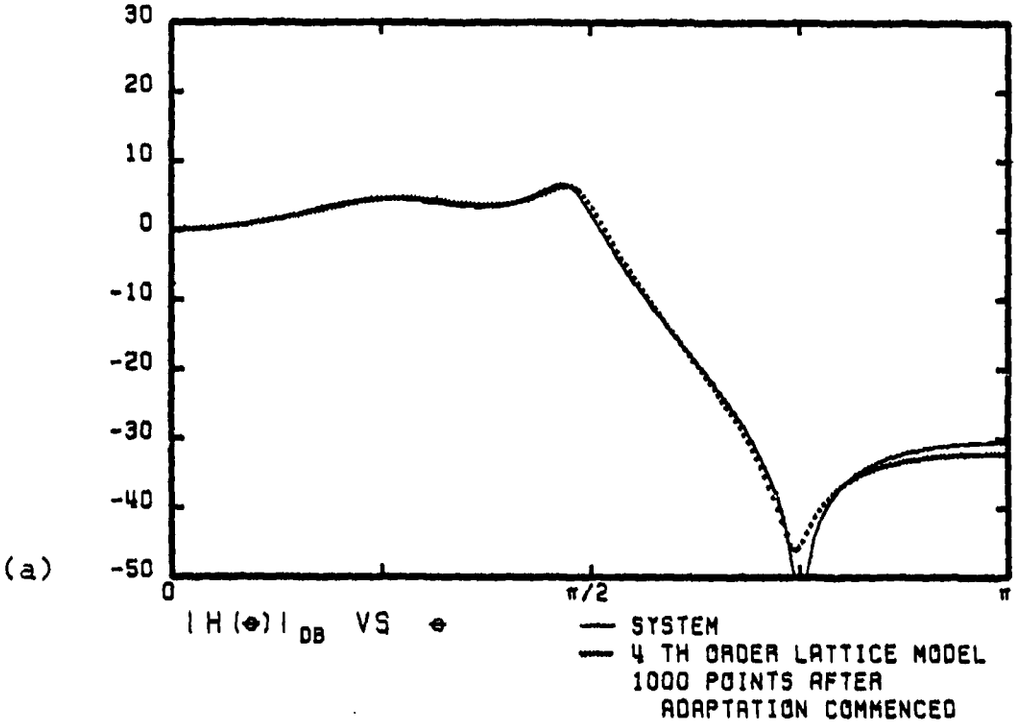


Figure 3.20b

K(1,1) term from the backward reflection coefficient matrices at the first five stages of the adaptive lattice model as a function of time.


30
20
10
0
-10
-20
-30
-40
-50
0
π/2
π
(a)
|H(ω)|DB VS ω
— SYSTEM
— 4 TH ORDER LATTICE MODEL
1000 POINTS AFTER
ADAPTATION COMMENCED

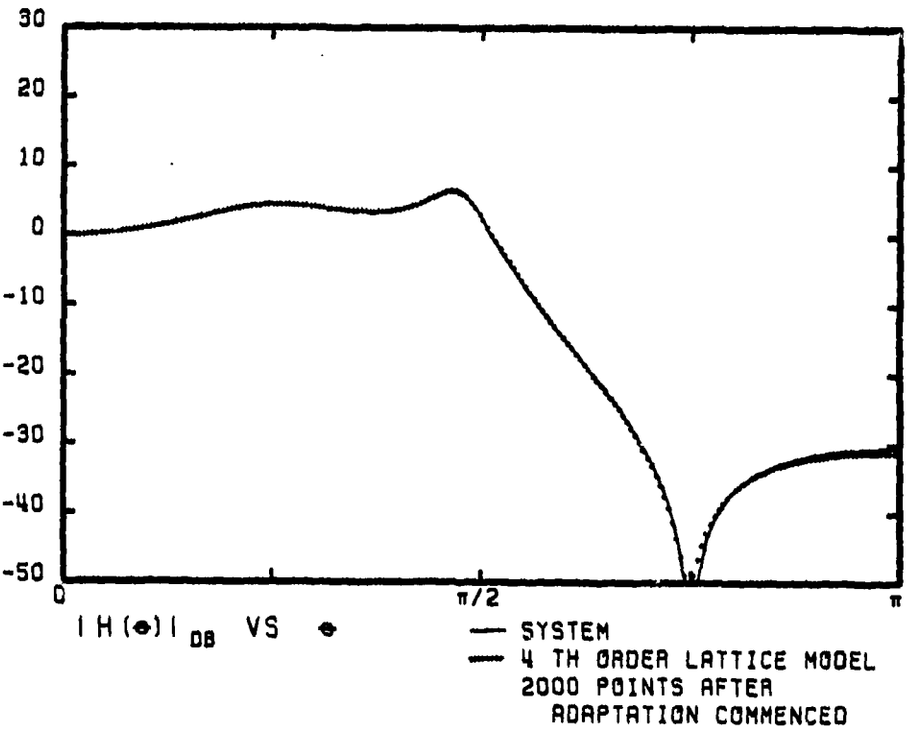

30
20
10
0
-10
-20
-30
-40
-50
0
π/2
π
(b)
|H(ω)|DB VS ω
— SYSTEM
— 4 TH ORDER LATTICE MODEL
2000 POINTS AFTER
ADAPTATION COMMENCED

Figure 3.21

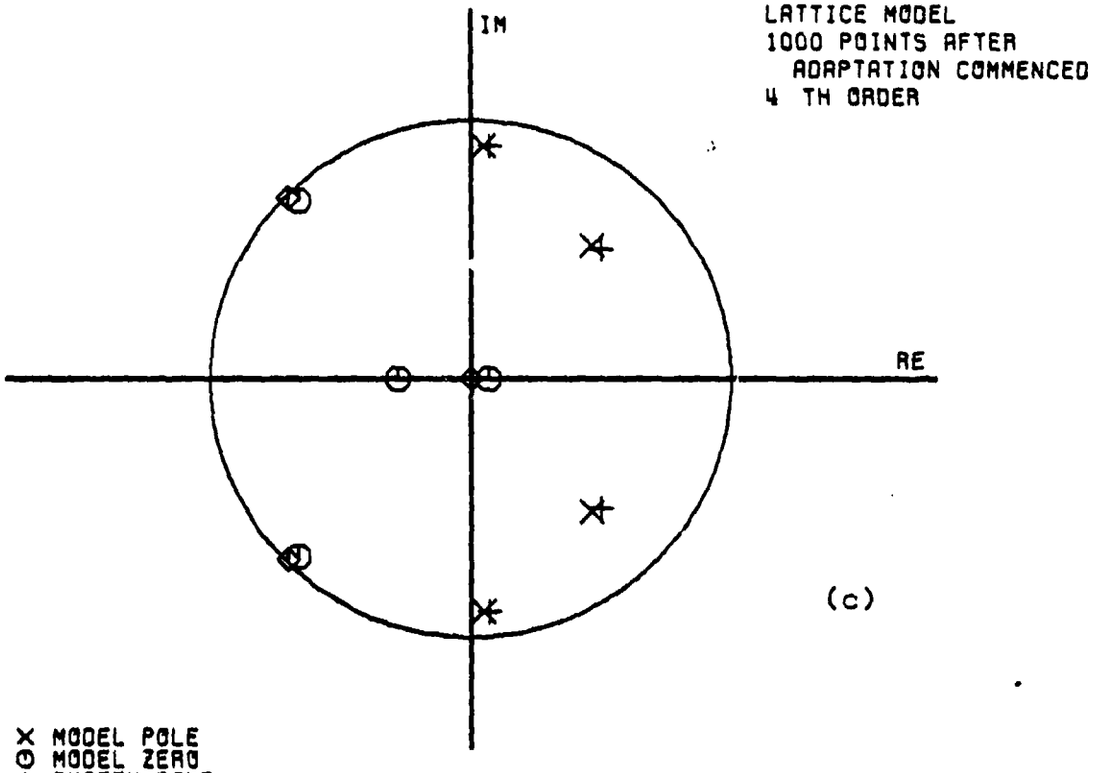
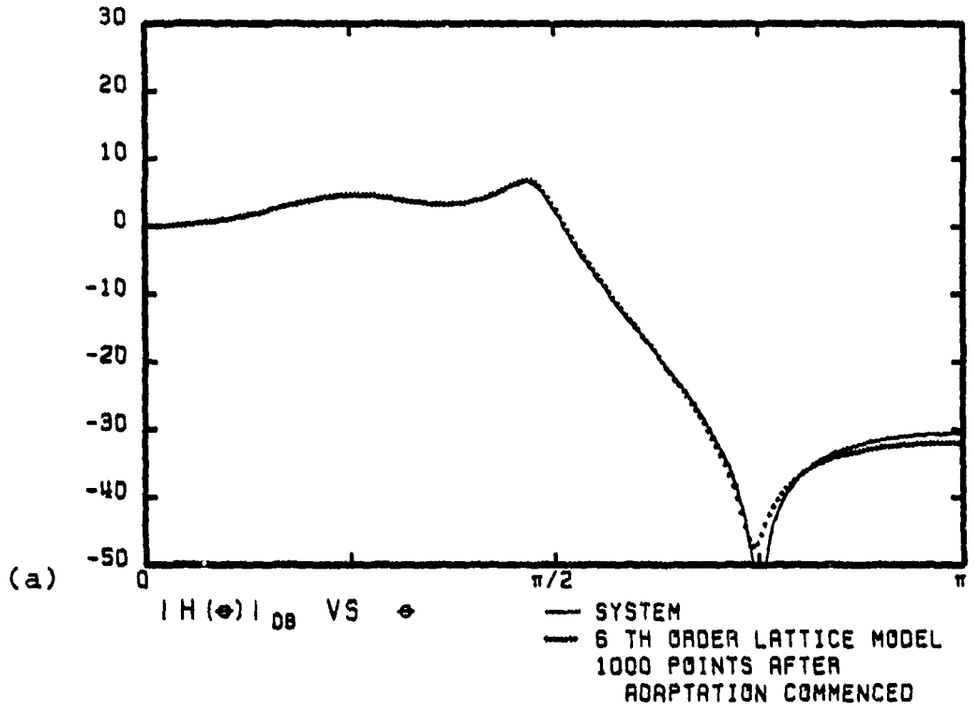


Figure 3.21 con't


30
20
10
0
-10
-20
-30
-40
-50
0
π/2
π
(a)
| H (ω) | DB VS ω
SYSTEM
6 TH ORDER LATTICE MODEL
1000 POINTS AFTER
ADAPTATION COMMENCED

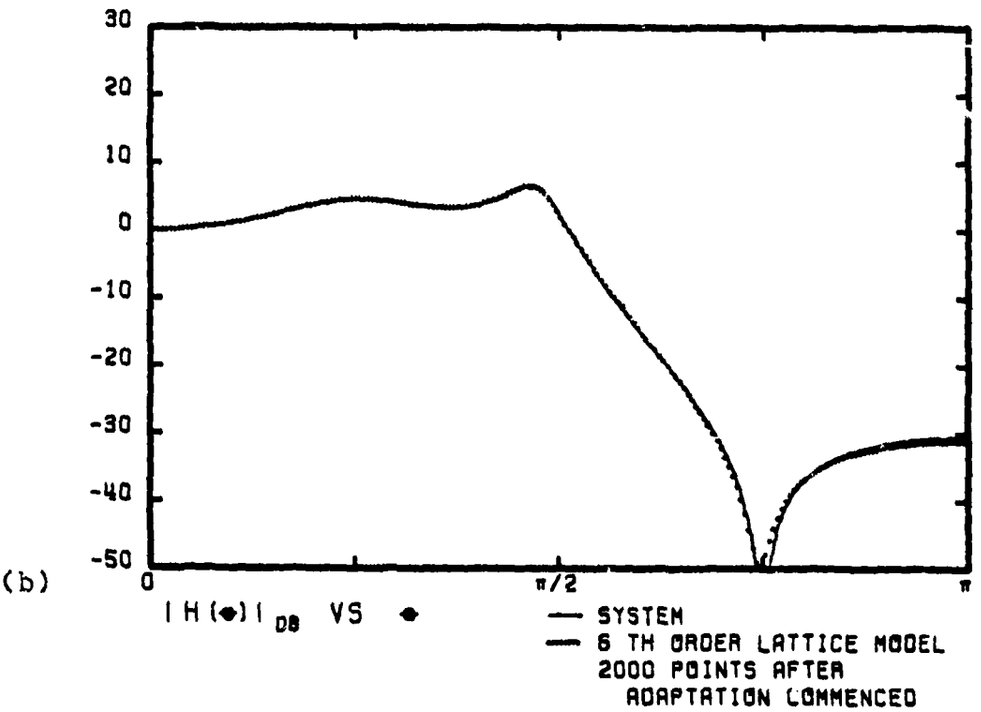

30
20
10
0
-10
-20
-30
-40
-50
0
π/2
π
(b)
| H (ω) | DB VS ω
SYSTEM
6 TH ORDER LATTICE MODEL
2000 POINTS AFTER
ADAPTATION COMMENCED

Figure 3.22

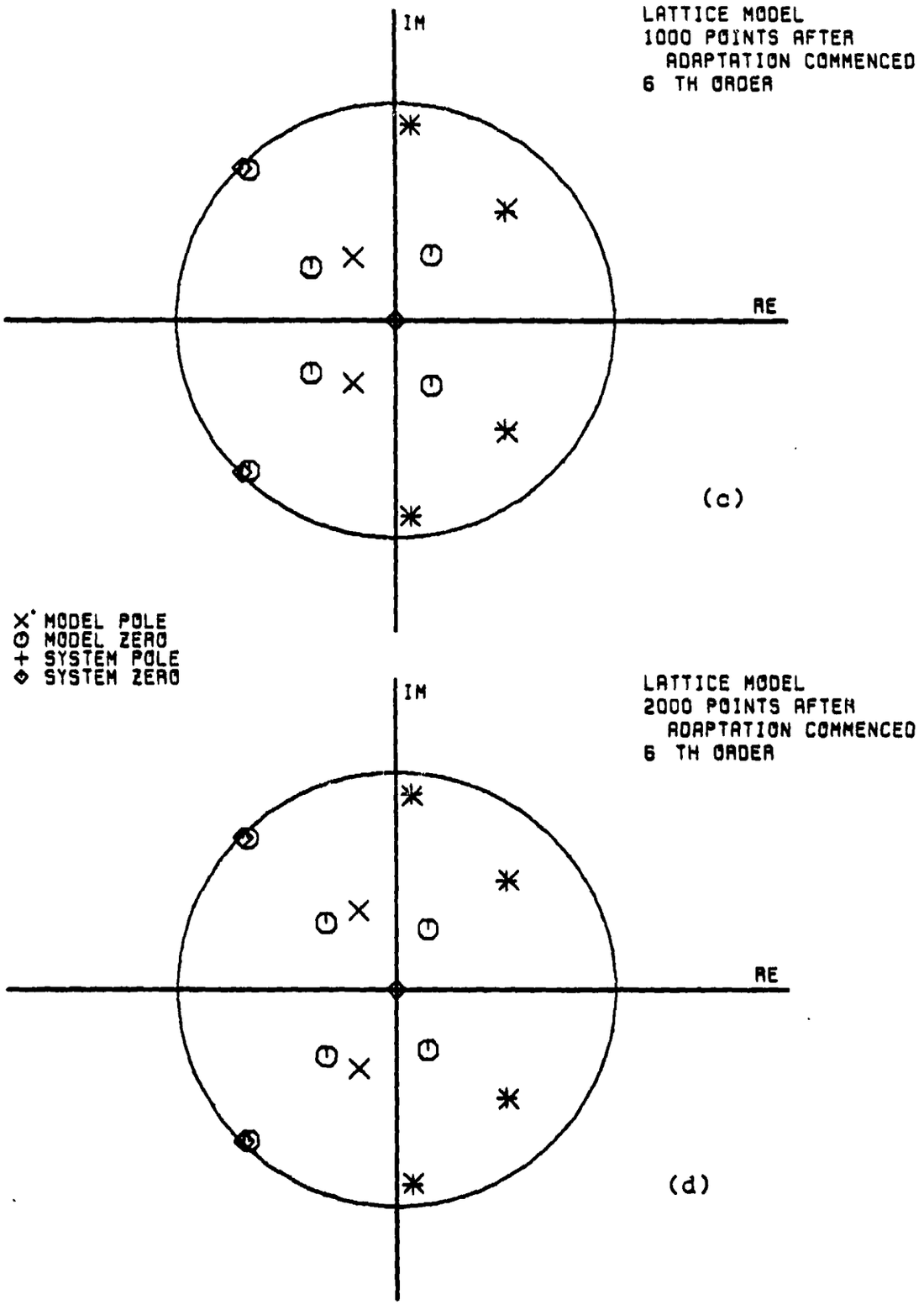


Figure 3.22 con't

of overmodeled, zero valued parameters has been found to be a general characteristic of the adaptive lattice algorithm and has also been noted briefly by Morf. [Ref. 38]

To understand the reason for this behavior, consider what occurs as the overmodeled fifth stage in the previous example adapts. Initially, before the coefficients of the first four lattice stages converge to their optimum values, the prediction error sequences out of the fourth stage are large and suboptimum. These signals provide incorrect inputs to the fifth stage driving its parameter estimates to some values other than their optimum zero values. As the first four stages converge, the prediction error sequences going into the fifth stage get small and since they drive the gradient estimates, convergence back toward zero is quite slow in these parameter estimates.

The cost functions being minimized at the overmodeled fifth stage are the trace of $\underline{P}^{(5)}$ and $\overline{P}^{(5)}$ given by

$$\underline{P}^{(5)} = \underline{P}^{(4)} - \underline{\Delta}^{(4)}\,\underline{K}^{(5)} - \underline{K}^{(5)^T}\underline{\Delta}^{(4)^T} + K^{(5)^T}\overline{P}^{(4)}\underline{K}^{(5)} \tag{3.40a}$$

$$\overline{P}^{(5)} = \overline{P}^{(4)} - \underline{\Delta}^{(4)^T}\overline{K}^{(5)} - \overline{K}^{(5)^T}\underline{\Delta}^{(4)} + \overline{K}^{(5)^T}\underline{P}^{(4)}\overline{K}^{(5)} \tag{3.40b}$$

Applying the results of Appendix F, the parabolic surfaces defined by these cost functions are described by the eigenvalues and eigenvectors of $\underline{P}^{(4)}$ and $\overline{P}^{(4)}$, the prediction error covariance matrices at the fourth stage. Consider the forward

predictions. The actual system output is given by

$$y(k) = \sum_{L=1}^{4} b(i)\, y(k-i) + \sum_{i=0}^{4} a(i)\, u(k-i) \qquad (3.41)$$

and for a white input signal, the minimum errors in the fourth order two channel autoregressive predictions of y and u are

$$e_y^{(4)}(k) = a(0)\, u(k) \qquad (3.42a)$$

$$e_u^{(4)}(k) = u(k) \qquad (3.42b)$$

This results in an optimal prediction error covariance matrix given by

$$\underline{P}^{(4)} = \begin{bmatrix} a(0)^2 & a(0) \\ a(0) & 1 \end{bmatrix} R_{uu}(0) \qquad (3.43)$$

with eigenvalues of 0 and $1+a(0)^2$. Also in the case of the system described by Table 3.1 where $a(3) = a(4) = 0$, it is seen from equation (3.41) that a perfect backward two channel AR prediction of $y(k-4)$ is possible resulting in an optimal backward prediction error covariance matrix of

$$\underline{P}^{(4)} = \begin{bmatrix} 0 & 0 \\ 0 & \varepsilon\{\bar{e}_u^{(4)}(k)\} \end{bmatrix} \qquad (3.44)$$

which also has a zero eigenvalue.

Since the ellipses obtained by passing a plane through the parabolic cost surface have axes whose half lengths given by $1/\sqrt{\lambda_i}$ and since $\underline{P}^{(4)}$ and $\overline{\underline{P}}^{(4)}$ each have one eigenvalue of zero, it is seen that the parabolic bowls along which $\underline{K}^{(5)}$ and $\overline{\underline{K}}^{(5)}$ adapt, degenerate toward infinitely long parabolic troughs as the first four stages converge toward their optimum values. This is responsible for the slow convergence of the overmodeled parameters back toward zero. To avoid this problem, some means of detecting this degeneration of the cost surface and then reseting the appropriate parameters to zero must be found and this certainly provides an interesting area for future study.

F. A LATTICE APPROACH FOR MULTICHANNEL ARMA MODELING

The lattice filter solution methods for the ARMA model can readily be generalized to multiple input multiple output ARMA models for linear systems. The equations for the multichannel ARMA model of the system shown in Figure 3.21 are developed in Appendix G with the solution for the model coefficients given by equation (G.7) repeated here for convenience.

$$\begin{bmatrix} \underline{R}_{YY} & \underline{R}_{Y^+U^+} \\ \underline{R}_{U^+Y} & \underline{R}_{U^+U^+} \end{bmatrix} \begin{bmatrix} \underline{B} \\ \underline{A}^+ \end{bmatrix} = \begin{bmatrix} \underline{R}_{Yy} \\ \underline{R}_{U^+y} \end{bmatrix} \qquad (3.45)$$

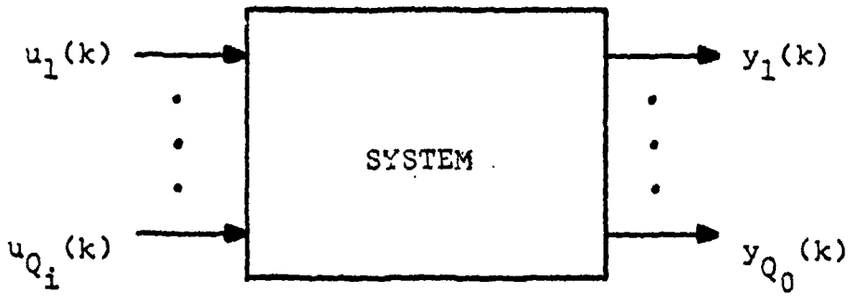


Figure 3.21. A general multichannel ARMA system

This is clearly a generalization of the single channel ARMA equation error solution given by equation (3.20) and just as in the single channel case some assumptions are required to apply a Levinson type algorithm.

If it is assumed that all the $a_{ij}(0)$ are known or can be estimated in another manner, they can be incorporated into a matrix given by

$$\underline{A}_0 = \begin{bmatrix} a_{11}(0) & \cdots & a_{1Q_0}(0) \\ & & \\ a_{Q_i1}(0) & \cdots & a_{Q_iQ_0}(0) \end{bmatrix} \tag{3.46}$$

and equation (3.45) becomes

$$\begin{bmatrix} \underline{R}_{YY} & \underline{R}_{YU} \\ \underline{R}_{UY} & \underline{R}_{UU} \end{bmatrix} \begin{bmatrix} \underline{B} \\ \underline{A} \end{bmatrix} = \begin{bmatrix} \underline{R}_{Yy} & \underline{R}_{Yu} \\ \underline{R}_{Uy} & \underline{R}_{Uu} \end{bmatrix} \begin{bmatrix} \underline{I} \\ -\underline{A}_0 \end{bmatrix} \tag{3.47}$$

This is similar in form to the equations for the solution of a $(Q_i + Q_0)$ channel autoregression with input channels $y_1(k)$, ..., $y_{Q_0}(k)$, $u_1(k)$, ..., $u_{Q_0}(k)$ so that the multichannel ARMA solution is related to the multichannel AR solution by

$$\begin{bmatrix} \underline{B} \\ \\ \underline{A} \end{bmatrix} = \underline{\mathbf{D}} \begin{bmatrix} \underline{I} \\ \\ -\underline{A}_0 \end{bmatrix} \tag{3.48}$$

The multichannel AR prediction error vector is given by

$$\underline{e}(k)^T = \begin{bmatrix} \underline{y}(k) \\ \hline \underline{u}(k) \end{bmatrix}^T - [\underline{Y}^T \mid \underline{U}^T]\,\underline{\mathbf{D}} = \begin{bmatrix} \underline{e}_y(k) \\ \hline \underline{e}_u(k) \end{bmatrix}^T \tag{3.49}$$

and defining

$$\underline{\Psi} = \begin{bmatrix} \underline{I} \\ \\ -\underline{A}_0 \end{bmatrix} \tag{3.50}$$

it follows that

$$\underline{e}(k)^T\underline{\Psi} = \underline{y}(k)^T - \underline{u}(k)^T\underline{A}_0 - [\underline{Y}^T \mid \underline{U}^T] \begin{bmatrix} \underline{B} \\ \\ \underline{A} \end{bmatrix} \tag{3.51}$$

$$= \underline{e}_0(k)^T$$

establishing the generalization of equation 3.27c relating the multichannel AR prediction error vector to the multichannel ARMA error vector $\underline{e}_0(k)$. The ARMA prediction error covariance then is given by

$$\underline{P}_0 = \underline{\psi}^T \underline{P} \underline{\psi} \tag{3.52}$$

and the coefficient matrix $\underline{A}_0$ can be set to minimize the trace of $\underline{P}_0$ resulting in a solution given by

$$\underline{A}_0 = \varepsilon\{\underline{e}_u(k)\ \underline{e}_u(k)^T\}^{-1}\ \varepsilon\{\underline{e}_u(k)\ \underline{e}_y(k)^T\} \tag{3.53}$$

completing the multichannel generalization of the single channel results. The portion of the solution given by the multichannel autoregression can be solved as before using the lattice methods in either batch or adaptive fashion. Then the matrix of gains can be obtained from equation (3.53) by batch processing or equations (3.37) and (3.39) can be generalized to yield an adaptive solution given by

$$\underline{A}_0(k+1) = \underline{A}_0(k) + 2\,\frac{\alpha}{\sigma_0^2(k)}\ \underline{e}_u(k)\ \underline{e}_0(k)^T \tag{3.54a}$$

where

$$\sigma_0^2(k) = (1-\alpha)\ \sigma_0^2(k-1) + \alpha\ \underline{e}_u(k)^T\ \underline{e}_u(k) \tag{3,54b}$$

It is clear that the equations and methods developed earlier for the single channel ARMA model are a special case of these results with $Q_i = Q_0 = 1$.

IV. NONLINEAR SYSTEM MODELING

The modeling of nonlinear systems is a far more complex problem than linear systems modeling. No attempt is made here to provide a comprehensive treatment of the problem. Rather, two specific models for systems comprised of the interconnection of linear and memoryless nonlinear subsystems are considered. Both of these models, the Volterra model and the new nonlinear ARMA model, are shown to be generalizations of the MA and ARMA modeling problems explored previously so that with appropriate modifications, the Levinson algorithm and lattice methods can be used to solve for the model parameters.

A. VOLTERRA NONLINEAR MODELING

The Volterra series model characterizes nonlinear systems using a generalization of convolution where the system output is approximated as a summation (possibly infinite) of outputs of degree m systems.

$$\hat{y}(k) = \sum_{m=1}^{M} y_m(k) \qquad (4.1a)$$

This is shown pictorially in Figure 4.1.

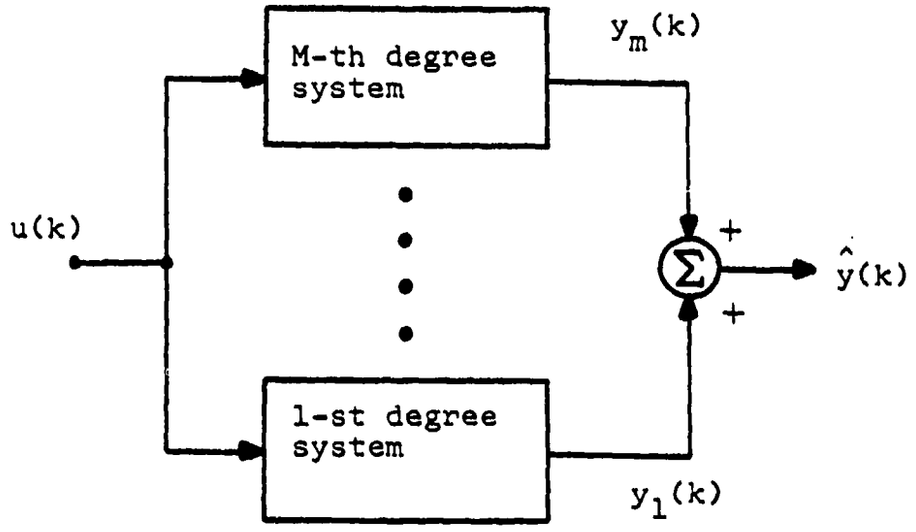


Figure 4.1. The Volterra model for nonlinear systems

A number of representations for these m-th degree systems are possible. The most commonly used representation is in terms of symmetric Volterra kernels and is given by

$$y_m(k) = \sum_{n_m=0}^{\infty} \cdots \sum_{n_1=0}^{\infty} a_s(n_1 \ldots n_m) u(k-n_1) \ldots u(k-n_m) \quad (4.1b)$$

and $a_s(n_1 \ldots n_m)$ is the m-th degree symmetric Volterra kernel. (Any permutation of the indicies results in the same value for this kernel giving a high degree of symmetry.)

This model arises quite naturally for a linear system in cascade with a power series nonlinearity as shown in Figure 4.2 for a quadratic nonlinearity where the output can be written as

$$y(k) = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} a(n_1)a(n_2)u(k-n_1)u(k-n_2) \qquad (4.2a)$$

and

$$a_s(n_1n_2) = a(n_1)a(n_2) \qquad (4.2b)$$

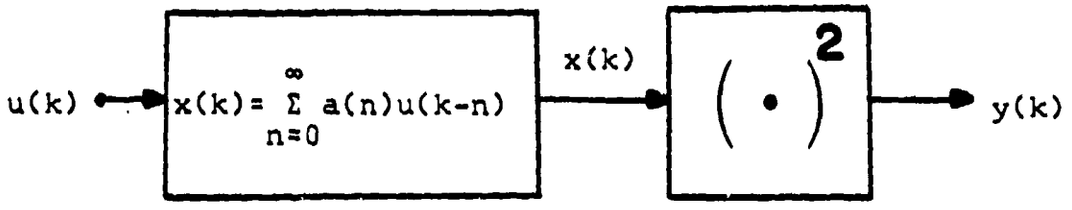


Figure 4.2. A quadratic nonlinear system.

The Volterra series model in this form has been widely discussed in the literature [e.g. Ref. 1, 6, 13, 14, 25, 26, 48 and 54] since Wiener [Ref. 62] first applied it to systems analysis and modeling. It has the added benefit of treating linear systems as a special case of the model since the first degree kernel is exactly the convolutional representation of the MA model.

The number of kernels (M) required for an accurate model depends on the nature of the nonlinearity in the system and as long as the nonlinearity is soft, a relatively low degree

model will suffice, keeping the problem manageable in that regard. The primary difficulty associated with Volterra nonlinear modeling arises from the fact that it uses a nonrecursive MA representation for the linear portion of the system, requiring in general, an infinite memory as indicated by the upper limits on the summations in equation (4.1b). In practice, these summations need to be truncated as shown in equation (4.3)

$$y_m(k) = \sum_{n_1=0}^{N} \cdots \sum_{n_m=0}^{N} a_s(n_1 \ldots n_m)u(k-n_1)\ldots u(k-n_m) \quad (4.3)$$

but a large number of terms may still be required to accurately model the system. One method of solving for the model is to set the parameters (terms of the Volterra kernels) to obtain a minimum mean square equation error where the equation error is defined as the difference between the system output and $\hat{y}(k)$.

To simplify the solution and reduce the number of parameters that must be obtained, the symmetry of the kernels can be exploited by rewriting equation (4.1b) as

$$y_m(k) = \sum_{n_1=0}^{\infty} \sum_{n_2=n_1}^{\infty} \cdots \sum_{n_m=n_{m-1}}^{\infty} a_t(n_1 \ldots n_m)u(k-n_1)\ldots u(k-n_m) \quad (4.4)$$

where $a_t(n_1 \ldots n_m)$ is the m-th degree triangular Volterra kernel. For a finite upper limit of N on the summations the m-th degree symmetric kernel will contain $(N+1)^m$ terms

but only $\frac{(N+m)!}{N!\ m!}$ of them are distinct with the remainder determined by symmetry considerations.

Still another representation for $y_m(k)$ uses the regular form of the Volterra kernel (this terminology has recently been introduced by Mitzel, Clancy and Rugh [Refs. 7 and 35]). It is given by

$$y_m(k) = \sum_{h_1=0}^{\infty} \cdots \sum_{h_m=0}^{\infty} a_r(h_1 \ldots h_m) u(k-h_1) u(k-h_1-h_2)$$

$$\ldots\ u(k-h_1-\ldots-h_m) \qquad (4.5)$$

where $a_r(h_1 \ldots h_m)$ is the m-th degree regular Volterra kernel. With infinite upper limits on the summations, the symmetric, triangular and regular forms of the Volterra kernels are equivalent, however, when finite upper limits are used they cover the field of the model kernels in different ways.

Because of its symmetry, it is reasonable to have equal upper limits on all the summations on the symmetric kernel as was done in equation (4.3). This is shown in Figure 4.3a for a second degree kernel. The equivalent triangular kernel is given by

$$y_m(k) = \sum_{n_1=0}^{N} \cdots \sum_{n_m=n_{m-1}}^{N} a_t(n_1 \ldots n_m) u(k-n_1) \ldots u(k-n_m) \qquad (4.6a)$$

and it covers the region shown in Figure 4.3b for a second degree case. The corresponding regular form expansion, however,

requires variable upper limits on the summations to cover the equivalent kernel space as shown in equation (4.6b), and Figure 4.3c for a second degree kernel.

$$y_m(k) = \sum_{h_1=0}^{N} \sum_{h_2=0}^{N-h_1} \cdots \sum_{h_m=0}^{N-h_{m-1}} a_r(h_1 \ldots h_m) u(k-h_1) u(k-h_1-h_2)$$

$$\ldots u(k-h_1-\ldots-h_m) \qquad (4.6b)$$

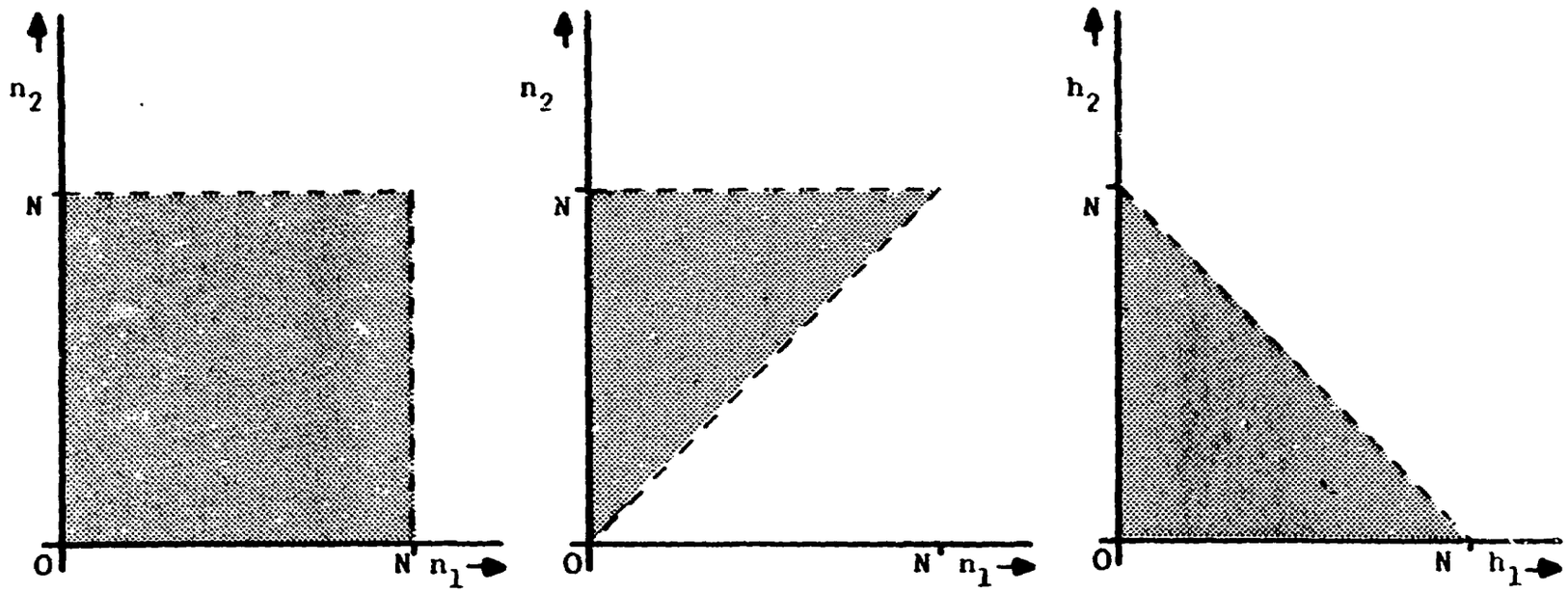
Thus it is seen that only half the field needs to be covered by the triangular and regular expansions to identify the kernel associated with the square field of the regular expansion. When the regular form expansion is used with constant finite upper limits there is no inherent reason to make all the upper limits equal since there is no symmetry in the kernel. Considering the regular expansion

$$y_m(k) = \sum_{h_1=0}^{N_1} \cdots \sum_{h_m=0}^{N_m} a_r(h_1 \ldots h_m) u(k-h_1) \ldots (u(k-h_1-\ldots-h_m) \qquad (4.7a)$$

a triangular expansion given by

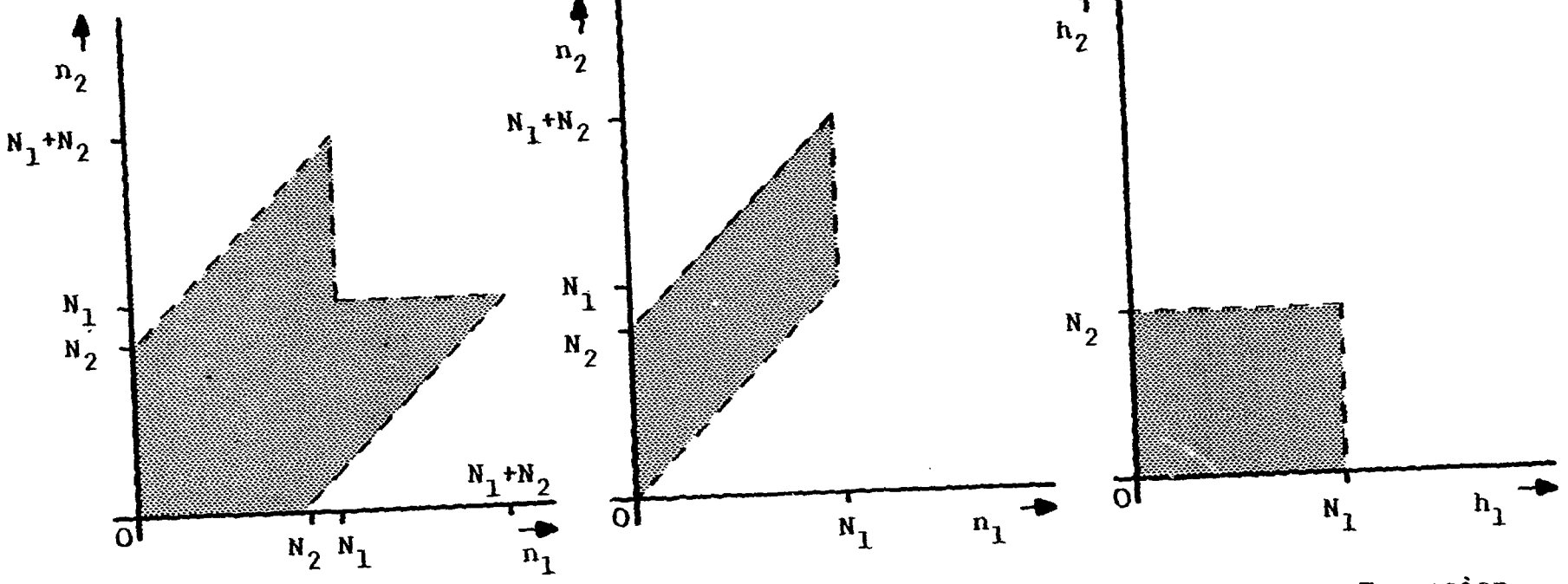
$$y_2(k) = \sum_{n_1=0}^{N_1} \sum_{n_2=n_1}^{n_1+N_2} \cdots \sum_{n_m=n_{m-1}}^{n_{m-1}+N_m} a_t(n_1 \ldots n_m) u(k-n_1) \ldots u(k-n_m) \qquad (4.7b)$$

is required to cover the corresponding field. For a quadratic expansion this is shown in Figures 4.4c and 4.4b. The equivalent region in the symmetric field is shown in Figure 4.2a.



(a) Symmetric Expansion (b) Triangular Expansion (c) Regular Expansion

Figure 4.3. Range of variables corresponding to a square field in the symmetric kernel space.



(a) Symmetric Expansion (b) Triangular Expansion (c) Regular Expansion

Figure 4.4. Range of variables corresponding to a rectangular field in the regular kernel space.

Therefore, identifying a rectangular kernel in the regular kernel space is equivalent to obtaining a symmetric kernel in the arrow shaped region of Figure 4.4a.

Which type of expansion is more appropriate for a given system depends on the shape of the kernels for that system. For example, if a quadratic nonlinear system has a kernel with a relatively square shape in the symmetric kernel space, a regular form expansion with constant upper summation limits will have to estimate many zero valued terms and is inefficient. On the other hand if the system has a kernel similar to the arrow shaped region of Figure 4.4a in the symmetric space, a regular expansion will be efficient while a symmetric expansion over a larger field would be required with many zero valued parameters. Not enough is known about how to relate types of nonlinear systems with various shaped kernels however, and the question of kernel shape and the best type of expansion is not pursued further here.

1. Lattice Filter Methods For Volterra Modeling

Lattice filter methods derived earlier can be applied to obtain a minimum mean square equation error solution for the Volterra model if the regular form of the expansion is used. The Volterra model can be put in the form of a multiple input single output MA model by defining a new family of signals as nonlinear combinations of delayed values of the system input u(k).

$$u_{h_2 \cdots h_m}(k) = u(k)\, u(k-h_2)\, u(k-h_2-h_3) \cdots u(k-h_2-\cdots-h_m) \tag{4.8}$$

For finite summations the regular form of the expansion becomes

$$y_m(k) = \sum_{h_m=0}^{N_{mm}} \cdots \sum_{h_2=0}^{N_{m2}} \left\{ \sum_{h_1=0}^{N_{m1}} a_r(h_1 \cdots h_m) u_{h_2 \cdots h_m}(k-h_1) \right\} \tag{4.9}$$

and can be regarded as the sum of the outputs of a large number of linear filters whose inputs are given by $u_{h_2 \cdots h_m}(k)$. Equation (4.9) is exactly the form of a Q_i input, single output MA model where

$$Q_i = (N_{m2}+1)(N_{m3}+1)\cdots(N_{mm}+1) \tag{4.10}$$

Furthermore, if the same upper limit on the summations over h_1 is used in each of the various m-th degree systems, $(N_{11}=N_{21}=\cdots=N_{m1})$, the overall Volterra model given by equations (4.1a) and (4.9) is in a form suitable for solution via the multichannel Levinson algorithm or the multichannel MA lattice methods. The requirement to use the same upper limit on all summations over h_1 arises because the Levinson algorithm and lattice methods assume the same amount of memory in each channel of the model.

As a specific example consider a second degree expansion where $N_{11}=N_{21}=N_1$ and $N_{21}=N_2$.

$$\hat{y}(k) = \sum_{h_1=0}^{N_1} a_r(h_1)u(k-h_1) + \sum_{h_2=0}^{N_2}\sum_{h_1=0}^{N_1} a_r(h_1h_2)u_{h_2}(k-h_1) \tag{4.11a}$$

$$u_{h_2}(k) = u(k)u(k-h_2) \tag{4.11b}$$

Defining data and coefficient vectors for each channel given by

$$\underline{u}^+_{h_2}(k) = [u_{h_2}(k)\ldots u_{h_2}(k-N_1)]^T \tag{4.12a}$$

$$\underline{a}^+_{h_2} = [a_r(0,h_2)\ldots a_r(N_1, h_2)]^T \tag{4.12b}$$

and embedding them into single data and coefficient vectors written as

$$\underline{\mathbf{X}}^+(k) = [\underline{u}^+(k)^T \mid \underline{u}^+_0(k)^T \mid \ldots \mid \underline{u}^+_{N_2}(k)^T]^T \tag{4.12c}$$

$$\underline{d}^+ = [\underline{a}^{+T} \mid \underline{a}^{+T}_0 \mid \ldots \mid \underline{a}^{+T}_{N_2}]^T \tag{4.12d}$$

the equation for the Volterra model output becomes

$$\hat{y}(k) = \underline{\mathbf{X}}^+(k)^T \underline{d}^+ \tag{4.12e}$$

which is clearly of the same form as equation (A.24a) and represents a N_2+2 input, single output MA model. All the nonlinearities in the Volterra model are external to this MA model, in the formation of its various input signals from the system input u(k). This model is illustrated in Figure 4.5.

It is interesting to consider what the recursive in order nature of the Levinson algorithm or lattice methods mean to the nonlinear Volterra problem. In building up the MA model solution recursively in order, the upper limit of the summations over h_1 is increased until the desired value is reached. In terms of the Volterra model kernels, this means allowing each of the regular form kernels to grow in size in the h_1 dimension while holding their boundaries fixed at prespecified values in all the other dimensions. In the linear MA model, the kernel has only one dimension and therefore the recursive in order solution eliminates any requirement to prespecify its upper limit (the order of the model). For the higher degree nonlinear kernels, these methods reduce by one the number of kernel boundaries that must be prespecified for each kernel. Allowing the kernel to grow in any of the other M-1 dimensions (h_2 through h_m) corresponds to adding additional channels to an existing lattice and simple methods of accomplishing this are not available.

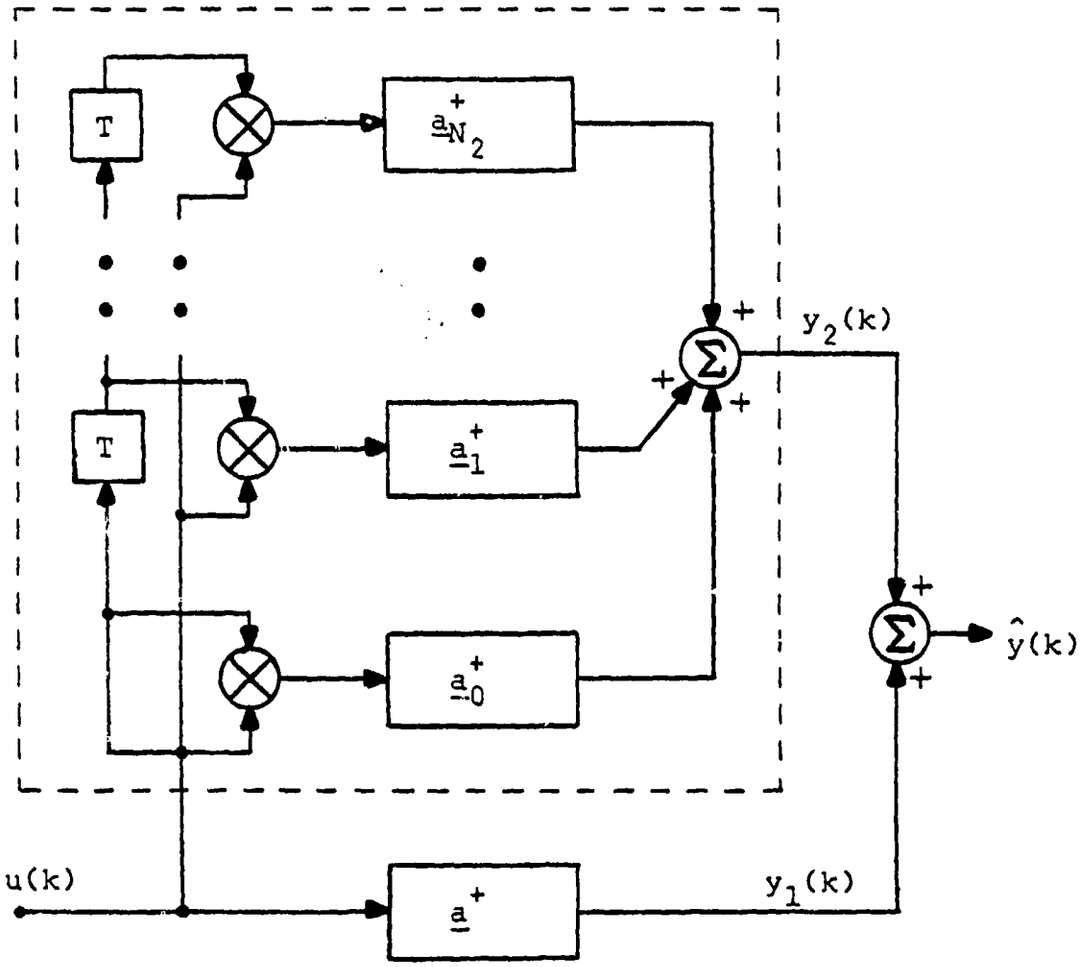


Figure 4.5. A second degree nonlinear Volterra model in multichannel MA form. The $\underline{a}^+_{h_1}$ represents are coefficients of the single input single output MA models shown.

3. NONLINEAR ARMA MODELING

As was previously mentioned, the primary difficulty associated with the Volterra series model arises from the fact that it is a nonlinear generalization of the MA model and as such, a large number of terms may be required to accurately represent even a mildly nonlinear system. In linear system modeling this difficulty was remedied by using the more general ARMA model. It is reasonable to assume therefore that a nonlinear generalization of the ARMA model could remedy the problem in the nonlinear modeling case (for at least certain types of nonlinear systems). Such a generalization called the nonlinear ARMA model has recently been proposed. [Ref. 64] This model forms an estimate of the current value of the system output as follows:

$$\hat{y}(k) = \sum_{n_1=0}^{\infty} a_s(n_1)u(k-n_1) + \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} a_s(n_1 n_2)u(k-n_1)u(k-n_2)$$

$$+\ldots+ \sum_{n_1=0}^{\infty}\ldots\sum_{n_p=0}^{\infty} a_s(n_1..n_p)u(k-n_1)\ldots u(k-n_p)$$

$$+ \sum_{m_1=1}^{\infty} b_s(m_1)y(k-m_1)+ \sum_{m_1=1}^{\infty}\sum_{m_2=1}^{\infty} b_s(m_1\ m_2)y(k-m_1)y(k-m_2)$$

$$+\ldots+ \sum_{m_1=1}^{\infty}..\sum_{m_q=1}^{\infty} b_s(m_1\ldots m_q)y(k-m_1)\ldots y(k-m_q)$$

$$+\sum_{n_1=0}^{\infty}\sum_{m_1=1}^{\infty} C(n_1 m_1)u(k-n_1)y(k-m_1)+\ldots+\sum_{n_1=0}\ldots\sum_{n_p=0}\sum_{m_1=1}\ldots$$

$$\ldots\sum_{m_q=1}^{\infty} C(n_1\ldots n_p m_1\ldots m_q)u(k-n_1)\ldots u(k-n_p)y(k-m_1)$$

$$\ldots y(k-M_q) \qquad (4.13)$$

The first three terms of equation (4.13) are a discrete Volterra expansion of the input signal u(k) and represent $F_{10}[u(k)]$ in terms of the discussion of Chapter I. The second three terms are a discrete Volterra expansion on the system output delayed one sample interval and represent $F_{20}[y(k-1)]$. The final two terms are bivariate expansions of the system input and delayed output and represent $F_{30}[u(k), y(k-1)]$. (This is the first and only model considered where $F_{30}[\cdot]$ is not assumed to be zero.) Equation (4.13) is clearly a nonlinear extension of the linear ARMA model contained in the first and fourth terms. As was the case in the Volterra model, the number of multiple summations required in (4.13) is dependent upon the nature of the nonlinearity in the system being modeled. The upper limits on the summations in (4.13) have been written as infinity indicating a requirement for infinite memory or model order. As is discussed subsequently, however, the required model order (memory) may in fact be finite due to the nature of the system being modeled, thereby alleviating the difficulty encountered in the Volterra nonlinear model previously presented.

The kernels of the input expansion and output expansion $a_s(\cdot)$ and $b_s(\cdot)$ are symmetric since any permutation of the indicies results in the same value for the kernel. They have therefore been labeled with a subscript "s" and can also be written in triangular and regular form.

$$\sum_{n_1=0}^{\infty}\cdots\sum_{n_p=0}^{\infty} a_s(n_1\ldots n_p)u(k-n_1)\ldots u(k-n_p)$$

$$= \sum_{n_1=0}^{\infty}\sum_{n_2=n_1}^{\infty}\cdots\sum_{n_p=n_{p-1}}^{\infty} a_t(n_1\ldots n_p)u(k-n_1)\ldots u(k-n_p)$$

$$= \sum_{h_1=0}^{\infty}\cdots\sum_{h_p=0}^{\infty} a_r(h_1\ldots h_p)u(k-h_1)u(k-h_1-h_2)\ldots$$

$$\ldots u(k-h_1-\ldots h_p) \qquad (4.14)$$

$$\sum_{m_1=1}^{\infty}\cdots\sum_{m_q=1}^{\infty} b_s(m_1\ldots m_q)y(k-m_1)\ldots y(k-m_q)$$

$$= \sum_{m_1=0}^{\infty}\sum_{m_2=m_1}^{\infty}\cdots\sum_{m_q=m_{q-1}}^{\infty} b_t(m_1\ldots m_q)y(k-1-m_1)\ldots(k-1-m_q)$$

$$= \sum_{h_1=0}^{\infty}\cdots\sum_{h_q=0}^{\infty} b_r(h_1\ldots h_q)y(k-1-h_1)y(k-1-h_1-h_2)$$

$$\ldots y(k-1-h_1-\ldots-h_q) \qquad (4.15)$$

In writing the triangular and regular forms in equation (4.15) the lower index on the summations has been shifted to zero.

In the case of the bivariate expansion terms in equation (4.13) the kernels do not possess symmetry so that triangular expansions are not possible but regular form expansions are possible.

$$\sum_{n_1=0}^{\infty} \cdots \sum_{n_p=0}^{\infty} \sum_{m_1=1}^{\infty} \cdots \sum_{m_q=1}^{\infty} C(n_1 \ldots n_p m_1 \ldots m_q) u(k-n_1)$$

$$\ldots u(k-n_p)\, y(k-m_1) \ldots y(k-m_q)$$

$$= \sum_{h_1=0}^{\infty} \cdots \sum_{h_{p+q}=0}^{\infty} c_{r1}(h_1 \ldots h_{p+q}) u(k-h_1) \ldots u(k-h_1-\ldots-h_p)$$

$$y(k-1-h_1-\ldots-h_{p+1}) \ldots y(k-1-h_1-\ldots-h_{p+q})$$

$$= \sum_{h_1=0}^{\infty} \cdots \sum_{h_{p+q}=0}^{\infty} c_{r2}(h_1 \ldots h_{p+q}) y(k-1-h_1) \ldots y(k-1-h_1-\ldots-h_q)$$

$$u(k-h_1-\ldots-h_{q+1}) \ldots u(k-h_1-\ldots-h_{p+q})$$

(4.16)

Thus two regular forms are possible for the bivariate expansions. Figures 4.6 and 4.7 illustrate the manner in which the regular form of the bivariate expansion covers the field

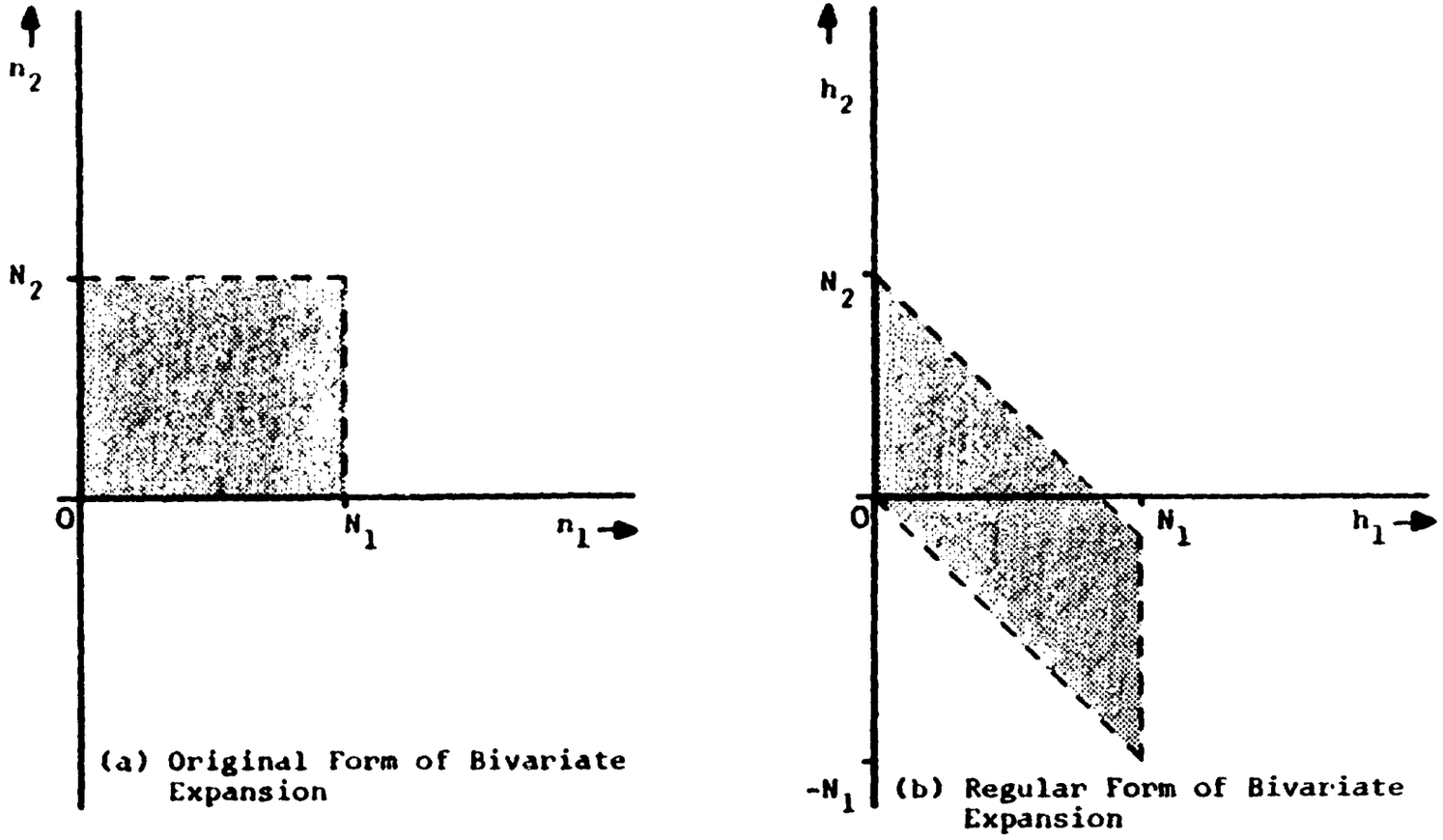
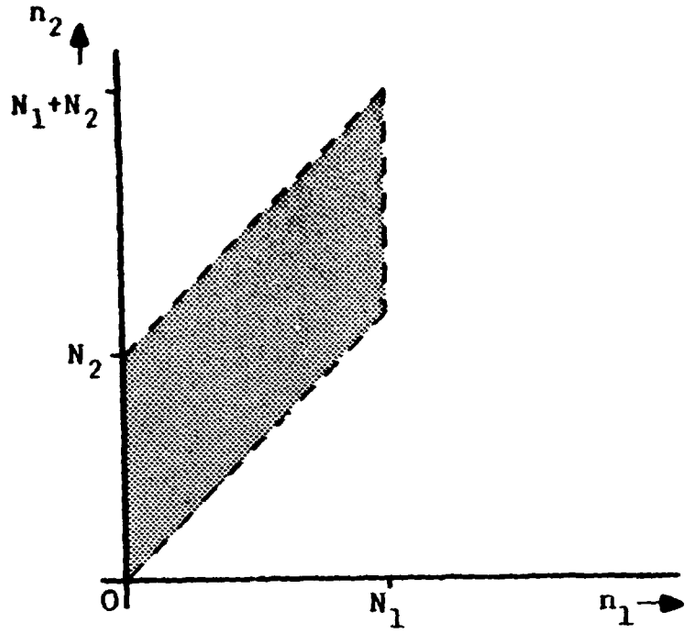
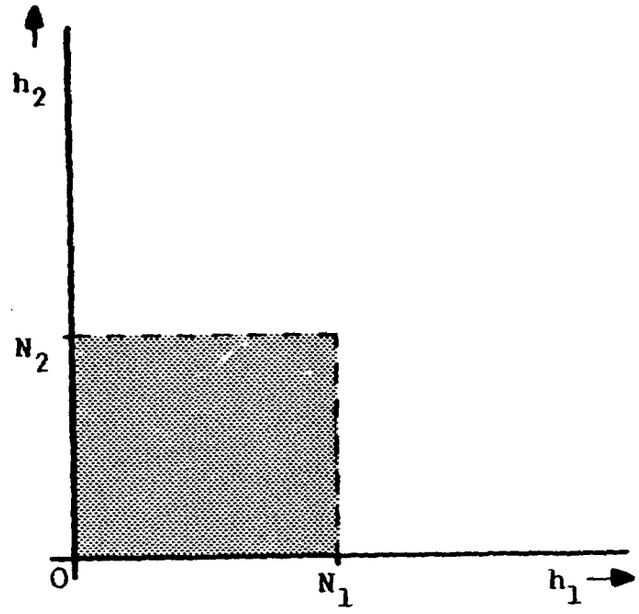


Figure 4.6. Range of variables for a rectangular field in the original form of the bivariate expansion of degree two.



(a) Original Form of Bivariate Expansion



(b) Regular Form of Bivariate Expansion

Figure 4.7. Range of variables for a rectangular field in the regular form of the bivariate expansion of degree two.

of model kernels for a second degree case for finite upper limits of N_1 and N_2 on the summations. It is interesting to note that because of a lack of symmetry in the original form of the bivariate expansion, the causal region in the regular form extends outside the quarter plane.

As was the case with the Volterra model, it will be shown that in regular form, a minimum mean square equation error solution can be obtained for the nonlinear ARMA model coefficients using either the Levinson algorithm or the lattice methods. This will provide the nonlinear generalization of the results presented in Chapter III on linear ARMA modeling. Before developing this method of solution, however, the applicability of the nonlinear ARMA model to various types of systems and its memory requirements will be considered.

1. Identifiability Conditions and Memory Requirements

In the previous chapter on linear ARMA modeling it was stated that there were two facets to the question of model identifiability; input signal requirements and measurement requirements. The question of measurement requirements was not discussed, however, since it was assumed that all signals (input and output) were observed. In the study of systems comprised of interconnected linear and nonlinear subsystems, however, various internal signals exist and the effect of either observing or not observing them on the modeling process must be explored. To do so, it will be assumed that the system under study can be put into the form of Figure 4.8 fulfilling the following equations.

$$\underline{x}_L(k) = \underline{\Gamma}_1 \underline{y}_N(k) + \underline{C}_1\underline{u}(k) \tag{4.17a}$$

$$\underline{x}_N(k) = \underline{\Gamma}_2 \underline{y}_L(k) + \underline{C}_2\underline{u}(k) \tag{4.17b}$$

$$\underline{y}_N(k) = \underline{F}\,[\underline{x}_n(k)] \tag{4.17c}$$

$$y_L(k) = \underline{T}\,[\underline{x}_L(x)] \tag{4.17d}$$

where

$$\underline{x}_L(k) = [x_{L1}(k) \quad \ldots \quad x_{LP}(k)]^T \tag{4.18a}$$

a vector of inputs to P linear subsystems;

$$\underline{x}_N(k) = [x_{N1}(k) \quad \ldots \quad x_{NQ}(k)]^T \tag{4.18b}$$

a vector of inputs to Q nonlinear memoryless subsystems;

$$\underline{y}_L(k) = [y_{L1}(k) \quad \ldots \quad y_{LP}(k)]^T \tag{4.18c}$$

a vector of outputs from P linear subsystems;

$$\underline{y}_N(k) = [y_{N1}(k) \quad \ldots \quad y_{NQ}(k)] \tag{4.18d}$$

a vector of outputs from Q nonlinear memoryless subsystems.

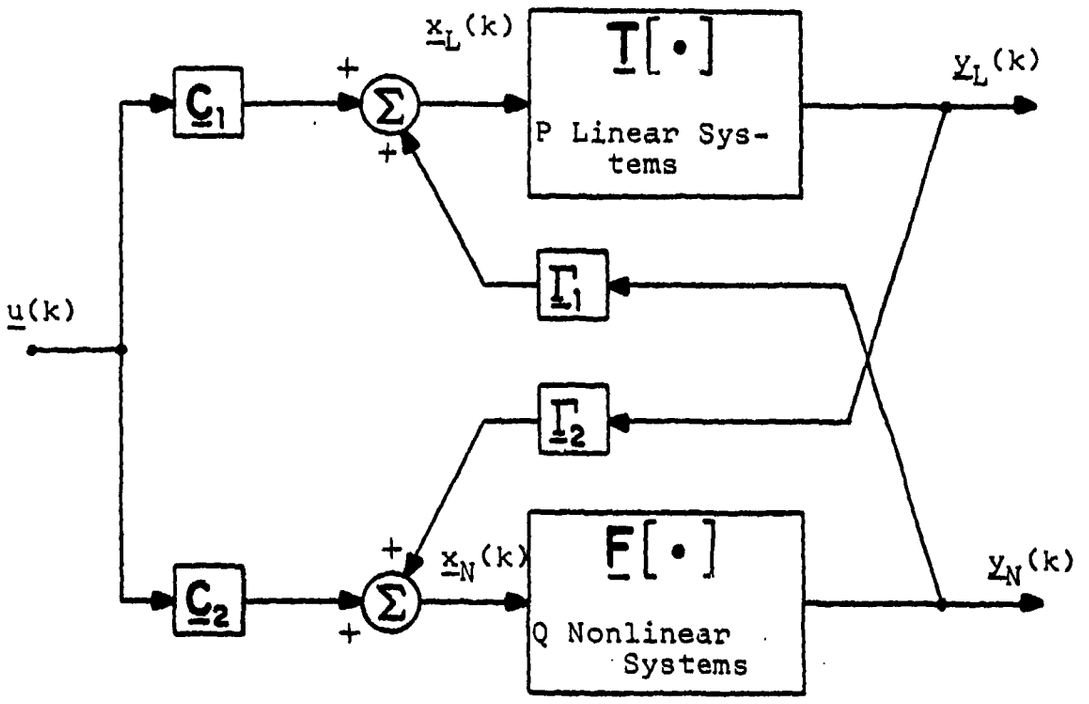


Figure 4.8. A General Nonlinear System

The z - transforms of these signals are also defined as $\underline{X}_L(z)$, $\underline{X}_N(z)$, $\underline{Y}_L(z)$, $\underline{Y}_N(z)$. $\underline{\Gamma}_1$, $\underline{\Gamma}_2$, $\underline{C}_1$ and $\underline{C}_2$ are matrices whose elements are either 0, -1 or +1 indicating the interconnections of the various subsystems. $\underline{T}[.]$ and $\underline{F}[.]$ are diagonal matrices specifing the linear and nonlinear subsystems as follows.

$$\underline{Y}_L(z) = \underline{T}(z)\ \underline{X}_L(z) \tag{4.19a}$$

where

$$\underline{T}(z) = [T_{ij}(z)] = \begin{cases} \dfrac{A_i(z)}{1-B_i(z)} & i=j \\ 0 & i\neq j \end{cases} \tag{4.19b}$$

and

$$A_i(z) = \sum_{n=0}^{N} a_i(n)\ z^{-n} \tag{4.19c}$$

$$B_i(z) = \sum_{n=1}^{N} b_i(n)\ z^{-n} \tag{4.19d}$$

$$\underline{Y}_N(k) = \underline{F}[\underline{x}_N(k)] = \begin{bmatrix} F_1[.] & & & 0 \\ & F_2[.] & & \\ & & \ddots & \\ 0 & & & F_Q[.] \end{bmatrix} \begin{bmatrix} x_{N_1}(k) \\ x_{N_2}(k) \\ \\ x_{NQ}(k) \end{bmatrix}$$

$$= \begin{bmatrix} F_1[x_{N1}(k)] \\ \vdots \\ F_Q[x_{NQ}(k)] \end{bmatrix} \tag{4.20}$$

This overall system given by Figure 4.8 is adequate to represent a broad class of systems comprised of interconnections of linear and nonlinear subsystems including cascades, parallel connections and feedback systems. Equations (4.19) and (4.20) assume all the subsystems are single input single output noninteracting systems. If desired, the collection of linear subsystems given by $\underline{T}(z)$ can readily be put into the general multichannel ARMA form of Section III.F to allow each output to be a function of past values of all outputs, and past and present values of all inputs.

Alternate representations of these linear and nonlinear subsystems are also useful. Equations (4.19) are equivalent to

$$\underline{Y}_L(z) = [A_i(z)]\underline{X}_L(z) + [B_i(z)]Y_L(z) \qquad (4.21a)$$

or in the time domain

$$\underline{y}_L(k) = [a_i(k)]*\underline{x}_L(k) + [b_i(k)]*\underline{y}_L(k) \qquad (4.21b)$$

Here * represents convolution and is carried out in the same sense as matrix multiplication and the matrices $[a_i(k)]$, $[b_i(k)]$, $[A_i(z)]$ and $[B_i(z)]$ are diagonal matrices whose i-th entries are the time domain functions $a_i(k)$ or $b_i(k)$ or the polynomials $A_i(z)$ or $B_i(z)$. The time domain functions $a_i(k)$ and $b_i(k)$ are the inverse transforms of the polynomials $A_i(z)$ and $B_i(z)$. This can also be represented

in nonrecursive form as

$$\underline{y}_L(k) = [h_i(k)] * \underline{x}_L(x) \tag{4.22a}$$

where $[h_i(k)]$ is a diagonal matrix of impulse responses defined by

$$h_i(k) = \sum_{n=0}^{\infty} h_i(n)\, \delta(k-n) \tag{4.22b}$$

The nonlinear systems can also be represented in terms of inverse functions assuming they exist over the necessary ranges of the variables so that

$$\underline{x}_N(k) = \underline{F}^{-1}[\underline{y}_N(k)] = \begin{bmatrix} F_1^{-1}[\cdot] & & 0 \\ & \ddots & \\ 0 & & F_Q^{-1}[\cdot] \end{bmatrix} \begin{bmatrix} y_{N1}(k) \\ \vdots \\ y_{NQ}(k) \end{bmatrix}$$

$$= \begin{bmatrix} F_1^{-1}[y_{N1}(k)] \\ \vdots \\ F_Q^{-1}[y_{NQ}(k)] \end{bmatrix} \tag{4.23}$$

So that equations (4.17) are iterative it is necessary that no delay free loops exist in the system of Figure 4.8. A necessary and sufficient condition for the absence of delay free loops is developed in Appendix H and requires that the terms of the determinant of the matrix

$$\underline{\alpha} = \lim_{z \to \infty} \underline{\Gamma}_2 \, \underline{T}_{(z)} \, \underline{\Gamma}_1 \tag{4.24}$$

and all its principal minors must be zero.

The various signal vectors and equations (4.17) can now be partitioned as

$$\begin{bmatrix} \underline{x}'_L(k) \\ \underline{x}''_L(k) \end{bmatrix} = \begin{bmatrix} \underline{\Gamma}_{1a} & \underline{\Gamma}_{1b} \\ \underline{\Gamma}_{1c} & \underline{\Gamma}_{1d} \end{bmatrix} \begin{bmatrix} \underline{y}'_N(k) \\ \underline{y}''_N(k) \end{bmatrix} + \begin{bmatrix} \underline{C}_{1a} \\ \underline{C}_{1b} \end{bmatrix} \underline{u}(k) \tag{4.25a}$$

$$\begin{bmatrix} \underline{x}'_N(k) \\ \underline{x}''_N(k) \end{bmatrix} = \begin{bmatrix} \underline{\Gamma}_{2a} & \underline{\Gamma}_{2b} \\ \underline{\Gamma}_{2c} & \underline{\Gamma}_{2d} \end{bmatrix} \begin{bmatrix} \underline{y}'_L(k) \\ \underline{y}''_L(k) \end{bmatrix} + \begin{bmatrix} \underline{C}_{2a} \\ \underline{C}_{2b} \end{bmatrix} \underline{u}(k) \tag{4.25b}$$

$$\begin{bmatrix} \underline{y}'_N(k) \\ \underline{y}''_N(k) \end{bmatrix} = \begin{bmatrix} \underline{F}_a[.] & \underline{F}_b[.] \\ \underline{F}_c[.] & \underline{F}_d[.] \end{bmatrix} \begin{bmatrix} \underline{x}'_N(k) \\ \underline{x}''_N(k) \end{bmatrix} \tag{4.25c}$$

$$\begin{bmatrix} \underline{y}'_L(k) \\ \underline{y}''_L(k) \end{bmatrix} = \begin{bmatrix} \underline{T}_a[.] & \underline{T}_b[.] \\ \underline{T}_c[.] & \underline{T}_d[.] \end{bmatrix} \begin{bmatrix} \underline{x}'_L(k) \\ \underline{x}''_L(k) \end{bmatrix} \tag{4.25d}$$

Equation (4.25c) can also be written in terms of inverse functions as in equation (4.23), and (4.25d) can be written using either recursive or nonrecursive representations of the linear systems as was done in (4.21) and (4.22). The single primed signal vectors in equations (4.25) represent those signals which are observed and the double primed signals are those which are not. It is assumed that all input signals

in $\underline{u}(k)$ are observed. It is possible to rewrite equations (4.25) as a single composite matrix equation as done in (4.26a). As written, equation (4.26a) is an infinite memory or model order representation because of the presence of the $\underline{h}(k)^*$ operators but a finite memory version can be written by expressing rows 4 and 8 as

$$\underline{y}_L'(k) = \underline{a}_a(k)^*\underline{x}_L'(k) + \underline{b}_a(k)^*\underline{y}_L'(k) + \underline{a}_b(k)^*\underline{x}_L''(k)$$

$$+ \underline{b}_b(k)^*\underline{y}_L''(k) \qquad (4.26b)$$

$$\underline{y}_L''(k) = \underline{a}_c(k)^*\underline{x}_L'(k) + \underline{b}_c(k)\underline{y}_L'(k) + \underline{a}_d(k)^*\underline{x}_L''(k)$$

$$+ \underline{b}_d'(k)^*\underline{y}_L''(k) \qquad (4.26c)$$

The third and seventh rows can also be written in terms of inverse functions if desired as

$$\underline{x}_N'(k) = \underline{F}_a^{-1}[\underline{y}_N'(k)] + \underline{F}_b^{-1}[\underline{y}_N''(k)] \qquad (4.26d)$$

$$\underline{x}_N''(k) = \underline{F}_c^{-1}[\underline{y}_N'(k)] + \underline{F}_d^{-1}[\underline{y}_N''(k)] \qquad (4.26e)$$

Now the problem of writing a system of iterative equations for the observed signals in terms of only observed signals consists of rewriting equation (4.26a) so that the upper right 4 by 5 partition is a null matrix. If this can be done, the general form of the nonlinear ARMA model can be

$$
\begin{bmatrix}
\underline{u}(k) \\
\underline{x}_L'(k) \\
\underline{x}_N'(k) \\
\underline{y}_L'(k) \\
\underline{y}_N'(k) \\
\hline
\underline{x}_L''(k) \\
\underline{x}_N''(k) \\
\underline{y}_L''(k) \\
\underline{y}_N''(k)
\end{bmatrix}
=
\left[\begin{array}{cccc|cccc}
\underline{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\underline{C}_{1a} & 0 & 0 & \underline{\Gamma}_{1a} & 0 & 0 & 0 & \underline{\Gamma}_{1b} \\
\underline{C}_{2a} & 0 & \underline{\Gamma}_{2a} & 0 & 0 & 0 & \underline{\Gamma}_{2b} & 0 \\
0 & \underline{h}_a(k)^* & 0 & 0 & \underline{h}_b(k)^* & 0 & 0 & 0 \\
0 & 0 & \underline{F}_a[\cdot] & 0 & 0 & \underline{F}_b[\cdot] & 0 & 0 \\
\hline
\underline{C}_{1b} & 0 & 0 & \underline{\Gamma}_{1c} & 0 & 0 & 0 & \underline{\Gamma}_{1d} \\
\underline{C}_{2b} & 0 & \underline{\Gamma}_{2c} & 0 & 0 & 0 & \underline{\Gamma}_{2d} & 0 \\
0 & \underline{h}_r(k)^* & 0 & 0 & \underline{h}_d(k)^* & 0 & 0 & 0 \\
0 & 0 & \underline{F}_c[\cdot] & 0 & 0 & \underline{F}_d[\cdot] & 0 & 0
\end{array}\right]
\begin{bmatrix}
\underline{u}(k) \\
\underline{x}_L'(k) \\
\underline{x}_N'(k) \\
\underline{y}_L'(k) \\
\underline{y}_N'(k) \\
\hline
\underline{x}_L''(k) \\
\underline{x}_N''(k) \\
\underline{y}_L''(k) \\
\underline{y}_N''(k)
\end{bmatrix}
\qquad (4.26a)
$$

used to identify the composite effects of the operators appearing in the upper left 5 by 5 partition. In some cases this will only be possible of the infinite memory version of the model (with h(k)* operators) is used and in other cases a finite memory model will suffice.

The process of determining whether infinite or finite memory nonlinear ARMA models can be used to identify a given system is illustrated for two examples in Appendix I. First a system consisting of a cascade of linear and nonlinear systems is considered. Then a model of the tracking behavior of a phase locked loop is put in the form of a nonlinear ARMA representation.

2. Lattice Filter Methods for the Nonlinear ARMA Model

As was the case with Volterra modeling, lattice filter methods can be applied to the nonlinear ARMA modeling problem if the regular form of the kernels is used. A family of signals is defined as nonlinear combinations of delayed values of y(k) and u(k) as follows.

$$u_{h_2 \ldots h_m}(k) = u(k)\, u(k-h_2)u(k-h_2-h_3)\ldots u(k-h_2-\ldots-h_m) \tag{4.27a}$$

$$y_{h_2 \ldots h_m}(k) = y(k-1)y(k-1-h_2)y(k-1-h_2-h_3) \ldots y(k-1-h_2-\ldots-h_m) \tag{4.27b}$$

$$u_{h_2 \ldots h_p}\, y_{h_{p+1} \ldots h_{p+q}}(k) = u(k)\, u(k-h_2) \ldots u(k-h_2-\ldots-h_p)$$

$$y(k-1-h_2-\ldots-h_{p+1})$$

$$\ldots y(k-1-h_1-\ldots-h_{p+q}) \quad (4.27c)$$

or alternately

$$y_{h_2 \ldots h_q}\, u_{h_{q+1} \ldots h_{q+p}} = y(k-1)y(k-1-h_2)\ldots y(k-1-h_2-\ldots-h_q)$$

$$u(k-h_2-\ldots-h_{q+1})\ldots u(k-h_2-\ldots-h_{q+p})$$

(4.27d)

With finite summations, the regular form of equation (4.14) becomes

$$\sum_{h_p=0}^{N_{p,p}} \cdots \sum_{h_2=0}^{N_{p,2}} \left\{ \sum_{h_1=0}^{N_{p,1}} a_r(h_1 \ldots h_p)\, u_{h_2 \ldots h_p}(k-h_1) \right\}$$

(4.28a)

Equation (4.15) becomes

$$\sum_{h_q=0}^{M_{q,q}} \cdots \sum_{h_2=0}^{M_{q,2}} \left\{ \sum_{h_1=0}^{M_{q,1}} b_r(h_1 \ldots h_q)\, y_{h_2 \ldots h_q}(k-h_1) \right\}$$

(4.28b)

and equations (4.16) become

$$\sum_{h_{p+q}=0}^{L_{p+q,p+q}} \cdots \sum_{h_2=0}^{L_{p+q,2}} \left\{ \sum_{h_1=0}^{L_{p+q,1}} C_{r1}(h_1 \cdots h_{p+q})\, u_{h_2 \cdots h_p} \right.$$

$$\left. y_{h_{p+1} \cdots h_{p+q}}(k-h_1) \right\} \qquad (4.28c)$$

or

$$\sum_{h_{q+p}=0}^{L_{q+p,q+p}} \cdots \sum_{h_2=0}^{L_{q+p,2}} \left\{ \sum_{h_1=0}^{L_{q+p,1}} C_{r2}(h_1 \cdots h_{q+p})\, y_{h_2 \cdots h_q} \right.$$

$$\left. u_{h_{q+1} \cdots h_{q+p}}(k-h_1) \right\} \qquad (4.28d)$$

These terms can be viewed as the summation of the outputs of a large number of linear filters whose inputs are the signals defined in equations (4.27). In the context of multichannel filtering, each of these filters can be considered as a single channel and each of the input signals can be associated with one of the channels. Since the lattice models use the same amount of memory in each channel, the upper limits on all summations over h_1 will be made the same ($N_{p,1} = N_{q,1} = L_{p+q,1} = N_1$). The upper limits on the summations over the other indicies determine the number of channels required.

For a quadratic nonlinear system, the present value of the system output is estimated as

$$\hat{y}(k) = \sum_{h_1=0}^{N_1} a_r(h_1)u(k-h_1) + \sum_{h_2=0}^{N_{22}} \sum_{h_1=0}^{N_1} a_r(h_1,h_2)u_{h_2}(k-h_1)$$

$$+ \sum_{h_1=1}^{N_1} b_r(h_1)y(k-h_1) + \sum_{h_2=0}^{M_{22}} \sum_{h_1=0}^{N_1} b_r(h_1,h_2)y_{h_2}(k-h_1)$$

$$+ \sum_{h_2=0}^{L_{22}} \sum_{h_1=0}^{N_1} C_{r_1}(h_1,h_2)\, u\, y_{h_2}(k-h_1) \qquad (4.29)$$

where $u_{h_2}(k) = u(k)u(k-h_2)$, $y_{h_2}(k) = y(k-1)y(k-1-h_2)$ and $u\, y_{h_2}(k) = u(k)y(k-1-h_2)$. Signal and coefficient vectors can be defined for the various quantities in equation (4.29) following the conventions established earlier; for example

$$\underline{y}(k) = [y(k-1) \ \ldots \ y(k-N_1)]^T \qquad (4.30a)$$

$$\underline{y}^+_{h_2}(k) = [y_{h_2}(k) \ \ldots \ y_{h_2}(k-N_1)]^T \qquad (4.30b)$$

and

$$\underline{b} = [b_r(1) \ \ldots \ b_r(N_1)]^T \qquad (4.30c)$$

$$\underline{b}^+_{h_2} = [b_r(0,k_2) \ \ldots \ b_r(N_1,h_2)]^T \qquad (4.30d)$$

Embedding all these vectors into single data and coefficient vectors, equation (4.29) becomes

$$\hat{y}(k) = \underline{\mathbf{X}}_1(k)^T \underline{d}_1 \tag{4.31a}$$

where

$$\underline{\mathbf{X}}_1(k) = [\underline{y}(k)^T \mid \underline{u}^+(k)^T \mid \underline{y}_0^+(k)^T \mid \cdots \underline{y}_{M_{22}}^+(k)^T \mid$$
$$\underline{u}_0^+(k)^T \mid \cdots \mid \underline{u}_{N_{22}}^+(k)^T \mid$$
$$\underline{uy}_0^+(k)^T \mid \cdots \mid \underline{uy}_{L_{22}}^+(k)^T]^T \tag{4.31b}$$

and

$$\underline{d}_1 = [\underline{b}^T \quad \underline{a}^{+T} \mid \underline{b}_0^{+T} \mid \cdots \mid \underline{b}_{M_{22}}^{+T} \mid$$
$$\underline{a}_0^{+T} \mid \cdots \mid \underline{a}_{N_{22}}^{+T} \mid$$
$$\underline{c}_0^{+T} \mid \cdots \mid \underline{c}_{L_{22}}^{+T}] \tag{4.31c}$$

The minimum mean square equation error solution for the model coefficient vector $\underline{d}_1$ is given by

$$\varepsilon\{ \underline{\mathbf{X}}_1(k) \underline{\mathbf{X}}_1(k)^T\} \underline{d}_1 = \varepsilon\{\underline{\mathbf{X}}_1(k) y(k)\} \tag{4.32}$$

where the equation error is defined as

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

While equation (4.32) is similar in form to equations (A.7a) and (A.26a) for AR and MA models, it lacks the necessary form for the application of the Levinson algorithm because of the fact that some component data and coefficient vectors are N_1-vectors while others are (N_1+1)-vectors. This is the same problem that arose in linear ARMA modeling and can be handled here in a similar manner. Defining

$$\underline{\psi} = [1 \quad -a_r(0) \quad -b_r(0,0) \quad \ldots \quad -b_r(0,M_{22}) \quad -a_r(0,0) \quad \ldots \quad -a_r(0,N_{22}) \quad -c_{r1}(0,0) \quad \ldots \quad -c_{r1}(0,L_{22})]^T \tag{4.33a}$$

and

$$\underline{x}(k) = [y(k) \quad u(k) \quad y_0(k) \quad \ldots \quad y_{M_{22}}(k) \quad u_0(k) \quad \ldots \quad u_{N_{22}}(k) \quad uy_0(k) \quad \ldots \quad uy_{L_{22}}(k)]^T \tag{4.33b}$$

and assuming that the coefficients in $\underline{\psi}$ can be estimated in some other manner, the MMSE solution for the remaining model coefficients becomes

$$\varepsilon\{\underline{\mathbf{X}}(k)\,\underline{\mathbf{X}}(k)^T\}\,\underline{d} = \varepsilon\{\underline{\mathbf{X}}(k)\,\underline{x}(k)^T\}\,\underline{\psi} \tag{4.34}$$

Here $\underline{\mathbf{X}}(k)$ and $\underline{d}$ are defined as $\underline{\mathbf{X}}_1(k)$ and $\underline{d}_1$ with all superscripts"+" removed from their component vectors (indicating they are all indexed from 1 to N_1 and are all N_1-vectors). Note that just as in the linear ARMA case, the coefficients in $\underline{\psi}$ essentially correspond to gains on their respective channels. Comparing equation (4.34) with (A.7a) for a multichannel autoregression it is clear that d can be obtained from the multichannel AR solution by

$$\underline{d} = \underline{\mathbf{D}}\,\underline{\psi} \tag{4.35}$$

where the signals in $\underline{x}(k)$ comprise the channels in the autoregression. Therefore, either the Levinson algorithm or the lattice methods can be used to solve for $\underline{\mathbf{D}}$ and from it and a knowledge of $\underline{\psi}$, the nonlinear ARMA model coefficients can be obtained.

By analogy with equation (3.27) it also follows that the nonlinear ARMA equation error is related to the AR prediction error vector by

$$e(k) = \underline{\psi}^T\,\underline{e}(k) \tag{4.36a}$$

so that

$$\varepsilon\{e(k)^2\} = \underline{\psi}^T\,\underline{\mathbf{F}}\,\underline{\psi} \tag{4.36b}$$

where $\underline{P}$ is the AR prediction error covariance matrix. The coefficients in ψ can therefore be set to minimize the mean square value of the nonlinear ARMA equation error in (4.36b) resulting in a solution given by

$$\begin{bmatrix} p_{22} & \cdots & p_{2N} \\ \vdots & & \\ p_{N2} & & p_{NN} \end{bmatrix} \begin{bmatrix} a_r(0) \\ \vdots \\ c_r(0,L_{22}) \end{bmatrix} = \begin{bmatrix} p_{21} \\ \vdots \\ p_{N1} \end{bmatrix} \tag{4.37a}$$

where N is the total number of channels in the model

$$N = 1 + 1 + (M_{22}+1) + (N_{22}+1) + (L_{22}+1)$$

and the p_{ij} are the elements of the prediction error covariance matrix. It is readily apparent that the linear ARMA model and its solution via the Levinson algorithm or lattice methods are a special case of this formulation of the nonlinear ARMA model just as one would expect.

V. APPLICATIONS, CONCLUSIONS, AND OPEN QUESTIONS

In the previous four chapters, existing methods for AR and MA modeling were reviewed and from them new methods for linear and nonlinear ARMA modeling were developed. Here, two applications for these new methods in reduced order modeling and modeling for fault detection and diagnosis are examined briefly. Then the results of this research are summarized, conclusions drawn and significant open questions for the continuation and extension of this work are listed.

A. APPLICATIONS

1. Reduced Order Modeling

Oftentimes, complex physical systems, both linear and nonlinear, can be approximated quite closely using simple models. The lattice solution methods developed here provide a very natural and efficient means of determining reduced order models for complex, high order systems especially in the case of linear systems. In Chapter III it was argued that for linear systems it is reasonable to build up ARMA models by simultaneously incrementing the order of the numerator and denominator polynomials as the lattice method does. When this method is used to build up a given order model, all lower order models and their mean square values of equation error are readily obtained as well (the only additional

calculations needed are for the MSE and the gain term or gain matrix in the multichannel case) making it easy to compare the various models and decide if reduced order models provide sufficient accuracy.

Consider for example the seventh order system whose characteristics are listed in Table 5.1. The magnitude spectra of second, third, sixth and seventh order lattice models obtained using batch processing with 4000 point averages are compared to that of the system in Figure 5.1. It is apparent that a second order model is unable to approximate the system well, however a third order model does provide a good approximation. Furthermore, increasing the model order to four, five and six fails to significantly improve its performance as evidenced by the sixth order plot in Figure 5.1c. The model accuracy is not significantly increased until seventh order (which corresponds to the order of the actual system) when a very good fit is achieved. This is further illustrated by the plot of the normalized mean square equation error as a function of model order shown in Figure 5.2. The cost drops rapidly going from second to third order but then fails to decrease significantly until seventh order is reached. The roots of the system and the various order models are also plotted in Figure 5.3.

The benefits of the lattice method for reduced order nonlinear ARMA modeling are not quite so pronounced however, since adding extra stages to the lattice corresponds to allowing the kernels to grow in only one of their multiple

TABLE 5.1

SYSTEM E

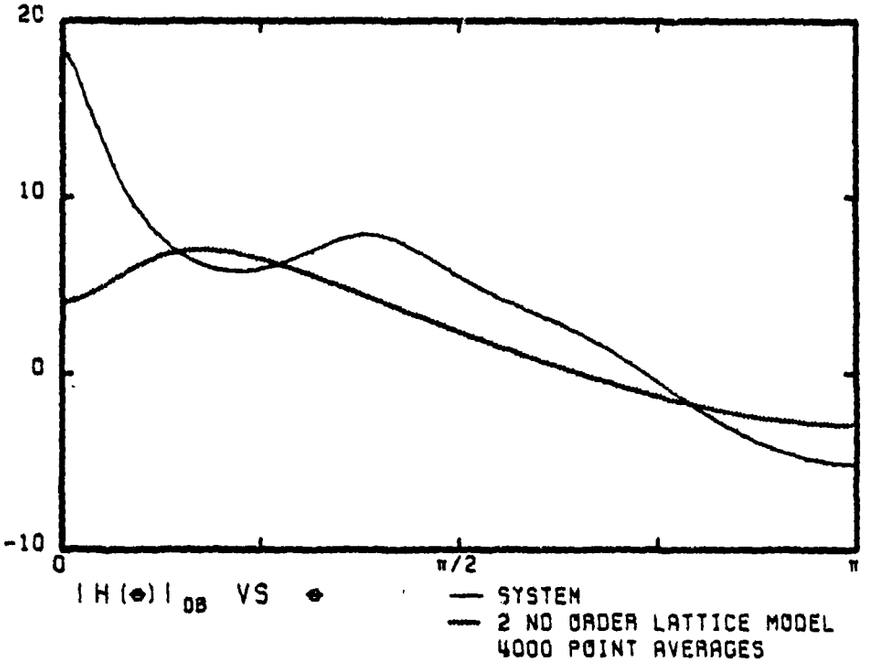
TRANSFER FUNCTION COEFFICIENTS

NUMERATOR		DENOMINATOR	
A(0)=	1.00000		
A(1)=	-2.16510	B(1)=	2.18950
A(2)=	1.56250	B(2)=	-2.21260
A(3)=	0.0	B(3)=	1.51740
A(4)=	0.0	B(4)=	-0.85350
A(5)=	0.0	B(5)=	0.45144
A(6)=	0.0	B(6)=	-0.23462
A(7)=	0.0	B(7)=	0.09331

ROOT LOCATIONS

ZEROS		POLES	
RE	IM	RE	IM
1.08250	0.62500	0.90000	0.0
1.08250	-0.62500	0.69282	0.40000
0.0	0.0	0.69282	-0.40000
0.0	0.0	0.23941	0.65778
0.0	0.0	0.23941	-0.65778
0.0	0.0	-0.28750	0.49796
0.0	0.0	-0.28750	-0.49796

(a)


20
10
0
-10
0
π/2
π
|H(ω)|dB VS ω
SYSTEM
2 ND ORDER LATTICE MODEL
4000 POINT AVERAGES

(b)

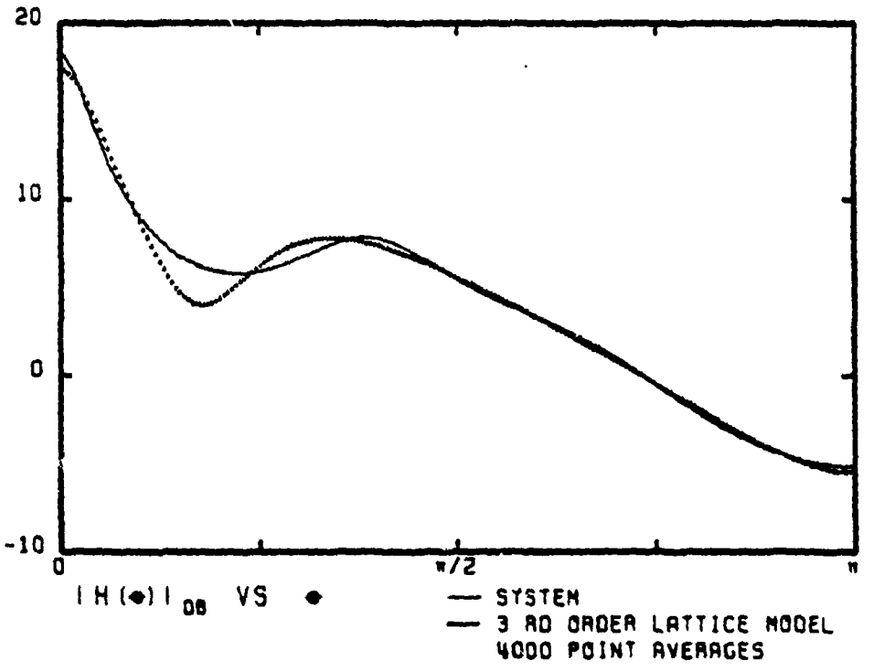
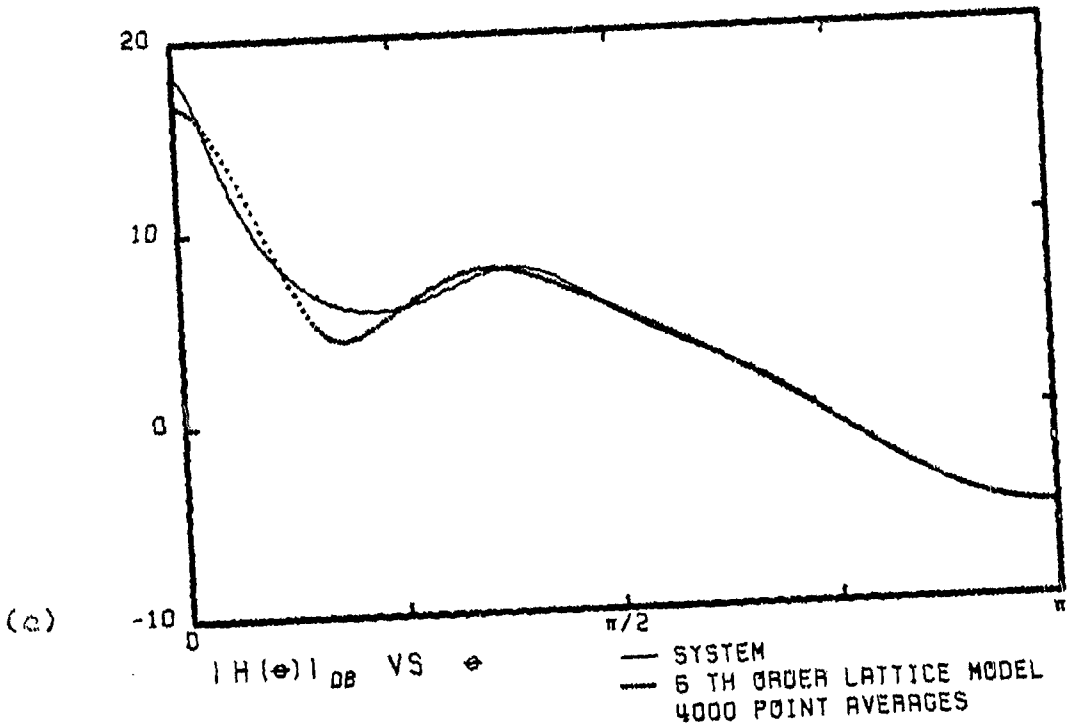
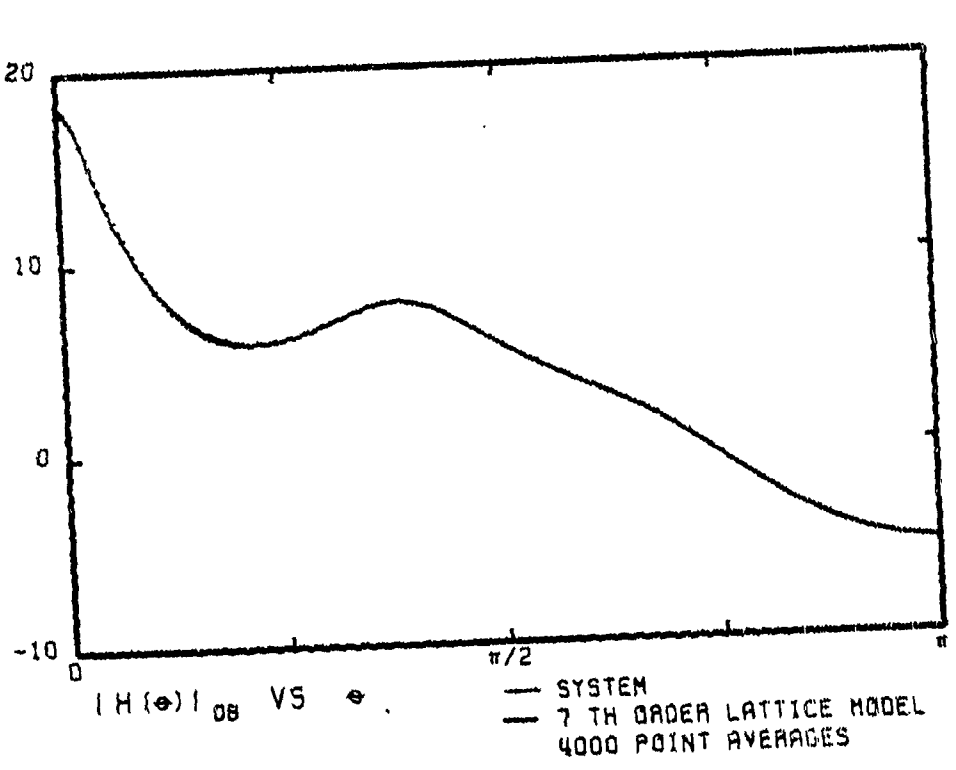

20
10
0
-10
0
π/2
π
|H(ω)|dB VS ω
SYSTEM
3 RD ORDER LATTICE MODEL
4000 POINT AVERAGES

Figure 5.1.



(c) $|H(e)|_{DB}$ VS θ



(d) $|H(e)|_{DB}$ VS θ.

Figure 5.1 con't

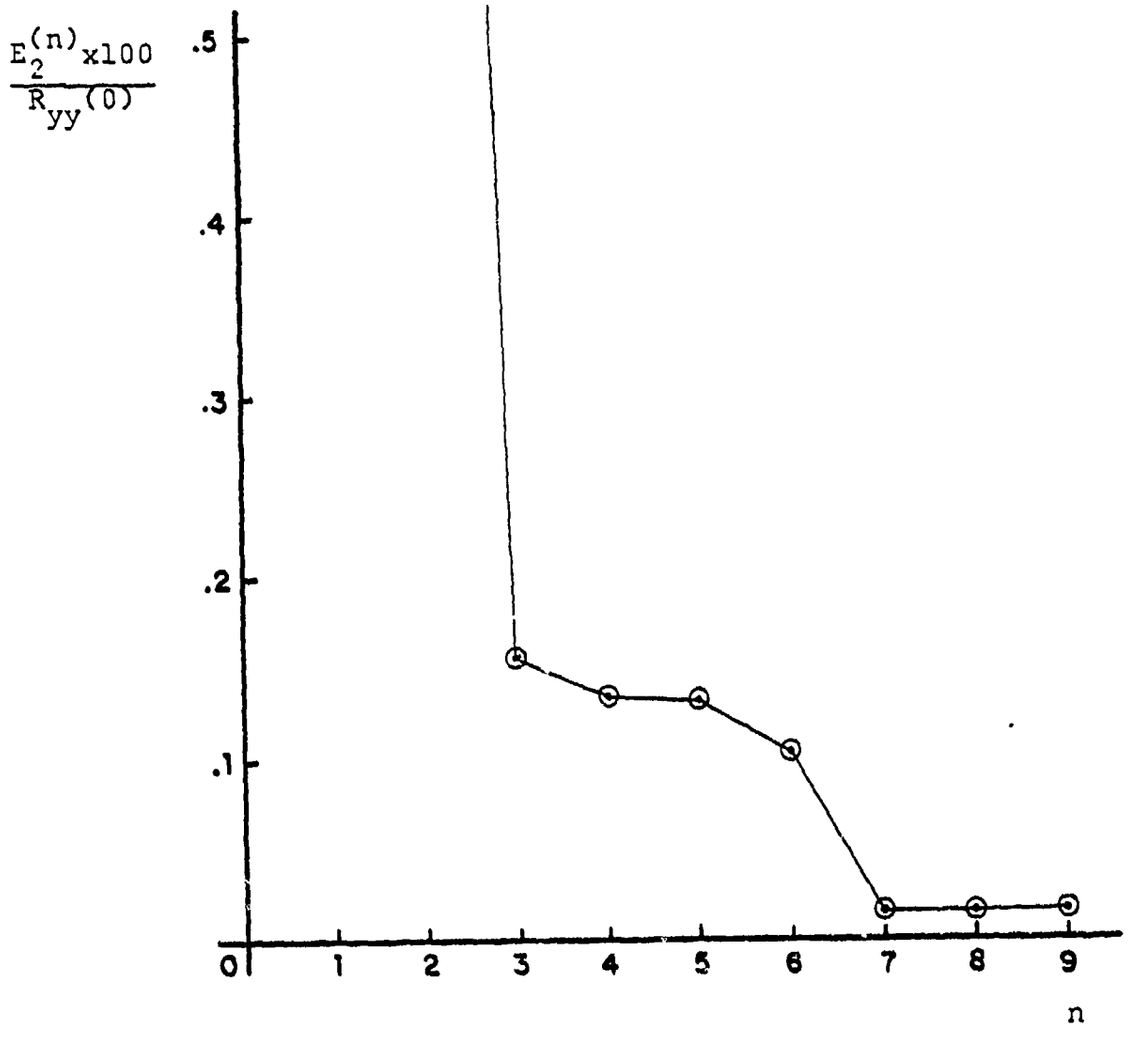


Figure 5.2. Mean square value of equation error (as a percentage of the mean square value of system output) vs. model order for lattice models of system E obtained using batch processing and 4000 point averages.

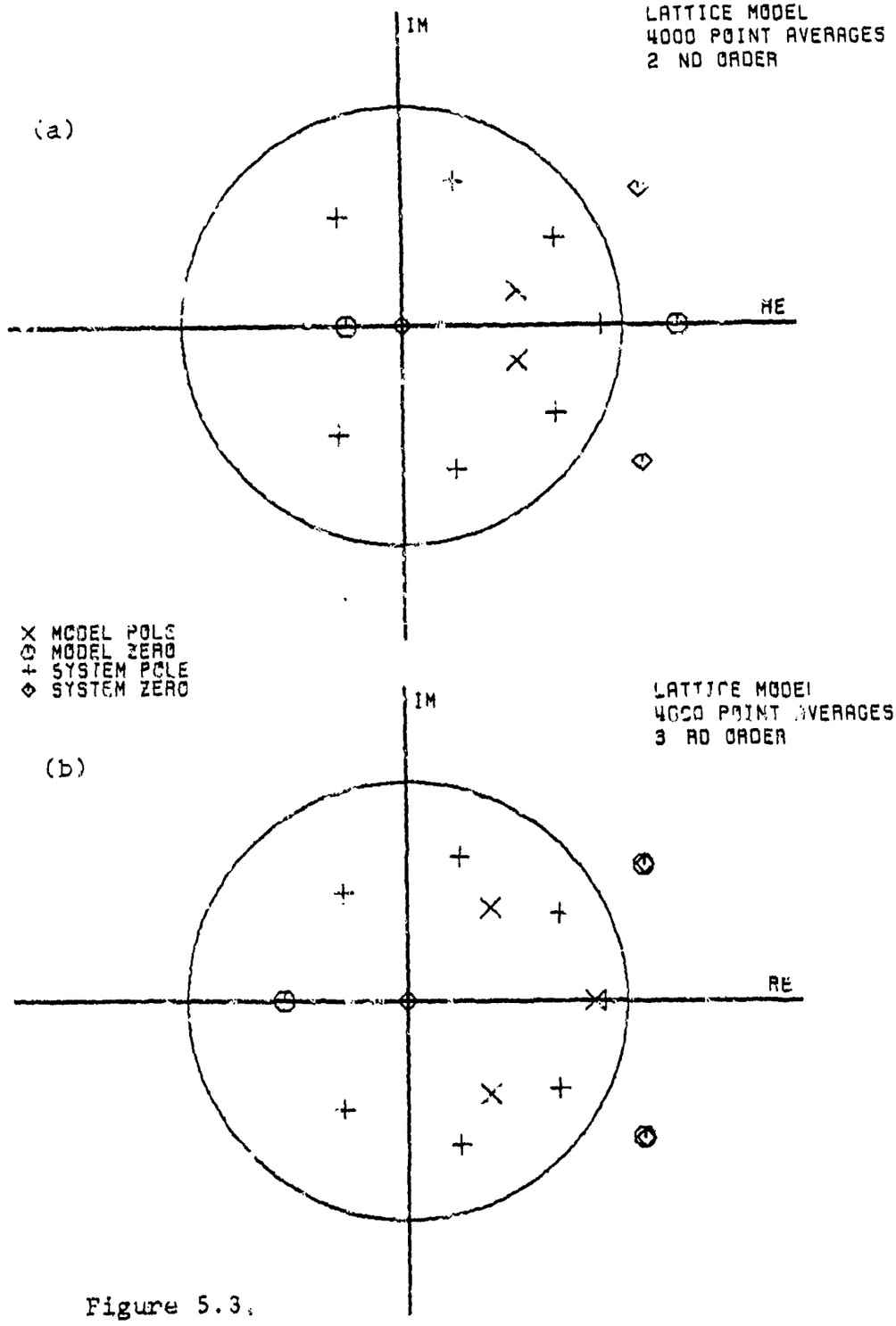

LATTICE MODEL
4000 POINT AVERAGES
2 ND ORDER
(a)
IM
RE
× MODEL POLE
⊖ MODEL ZERO
+ SYSTEM POLE
◇ SYSTEM ZERO
LATTICE MODEL
4000 POINT AVERAGES
3 RD ORDER
(b)
IM
RE

Figure 5.3.

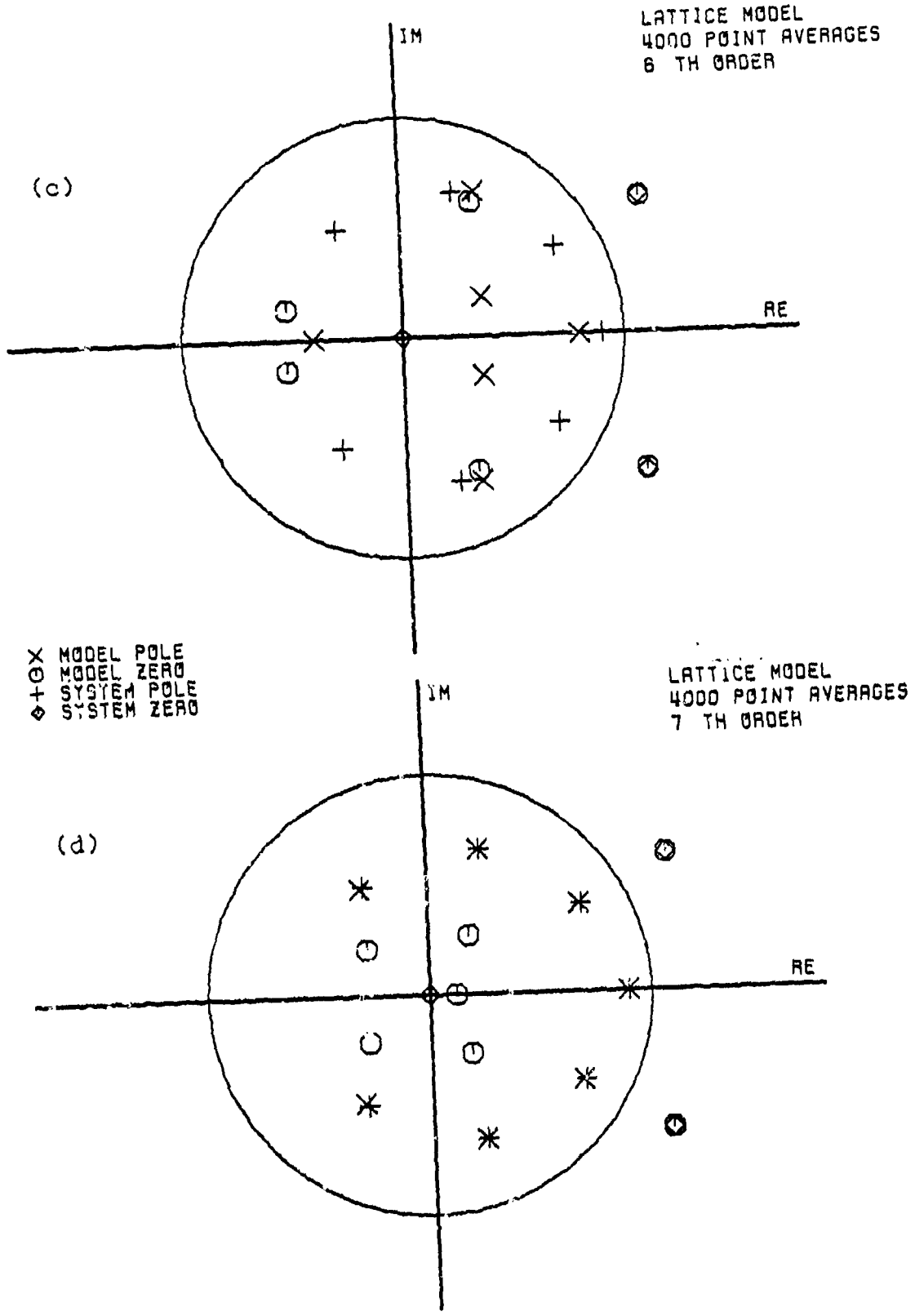


Figure 5.3 cont.

dimensions. (The linear "kernel" had only one dimension.) Here some a priori information would be useful in determining the values to prespecify for the boundaries of the kernels in the other dimensions. Still however, when compared to a brute force matrix inversion approach where all kernel boundaries must be prespecified, the lattice method provides a viable alternative in obtaining reduced order models especially for low degree systems.

2. Modeling For Fault Detection

The problem of fault detection and possible diagnosis can be formulated as follows.

a. Obtain a parameterization that describes the current functioning of the system under test.

b. From this parameterization, determine if the system is functioning normally or if a fault has occurred by comparison with a fault dictionary.

It is the first part of this problem that has been addressed in this work. The parameterization can be as simple as sampled measurements of the response to specific inputs however, the large volume of data that would generally be involved in such an approach would greatly complicate the second part of the problem. A more efficient approach in terms of utilization of parameters is to model the system and use the model parameters as a description of its current functioning.

For linear systems, ARMA models provide a very general framework with a number of possible parameter sets.

Three candidates are polynomial coefficients, root locations and lattice reflection coefficients with the latter offering many advantages. In addition to the advantages demonstrated by the experimental results of Chapter III, reflection coefficients provide a very effective and methodical way to build up knowledge of a system's characteristics. As model order is increased to more accurately represent a system, reflection coefficients already determined don't change making them ideal candidates for use in a dictionary lookup scheme (a characteristic not shared by the other parameterizations). This is made more important since reduced order models could be adequate to detect and perhaps diagnose some faults, especially catastrophic ones. While more parameters are required when reflection coefficients are used, these same coefficients also provide all reduced order models. For a single channel ARMA example, 8N reflection coefficients and N gains provide all models from order 1 to N while N^2+2N parameters would be required using either polynomial coefficients or roots to provide the same information. (6N reflection coefficients are needed if the input is white noise since $k_{12}=k_{22}=0$.)

A similar argument could be made for the use of lattice reflection coefficients with the nonlinear ARMA model for fault detection and diagnosis of nonlinear systems.

B. CONCLUSIONS AND OPEN QUESTIONS

The purpose of this research was to extend existing theories and methods in the modeling of linear and nonlinear systems to broader, more general types of models. After a discussion of available results in AR and MA modeling of linear systems with particular emphasis on the Levinson algorithm and lattice filter methods, model transition formulas were developed to relate the more general ARMA model for linear systems to the AR and MA models. It was shown that with suitable assumptions, the ARMA model solution could also be obtained recursively in order using either a modified Levinson algorithm or lattice filter methods. These results were developed extensively in both theory and practice for single channel linear ARMA modeling with experimental verification of both the batch processing and adaptive lattice methods presented. Portions of these results have already been published. [Refs. 65 and 66] The theory was also developed to generalize these results to the multichannel ARMA case.

Based on the simulation results it was concluded that the lattice methods offer the following advantages over a conventional brute force matrix inversion approach to ARMA modeling using windowed correlation estimates.

1. For short runs of data the batch lattice methods provide much more accurate results than the brute force method.

2. The batch lattice method performs much better than the brute force method when the system is over-modeled.
3. The MSE as a function of model order is well behaved for the lattice method.
4. The adaptive lattice method has difficulty tracking zero valued overmodeled parameters.

The cost of these advantages is the extra computational burden of passing the data through the lattice filter during the modeling process.

In the discussion of nonlinear system models the Volterra model was considered as a nonlinear extension of MA modeling and it was shown that lattice methods could be used to obtain the model solution if the problem was recast, using the regular form of the Volterra kernels. Then the new nonlinear ARMA model was considered and it was shown that this representation in some cases solves the problem of requiring a very large number of model parameters encountered in Volterra modeling. Then lattice methods were developed for the nonlinear ARMA problem and it was shown that the linear ARMA techniques presented earlier are a speical case of the nonlinear ARMA methods. For both types of nonlinear models, the recursive in order nature of the lattice methods was shown to allow the various model kernels to grow in one dimension while holding their boundaries fixed at prespecified values in the other dimensions. The use of the model nonlinear ARMA was also illustrated with two examples

and a nonlinear ARMA model was proposed for the tracking behavior of a phase locked loop.

Several significant questions remain for the continuation and extension of this work and are listed here.

1. Stability of the linear and nonlinear ARMA models must be considered. In the linear problem, stability is dependent on the roots of the demoninator polynomial of the synthesis model. The methods developed do not guarantee stability of the resulting model. (This was not found to be a problem in practice, however, unless extremely short runs of data in the range of 30 to 50 samples were used. Even then, model instability was not a frequent occurrence.) Stability for the nonlinear ARMA model remains to be clearly defined.
2. Input signal requirements in the nonlinear ARMA modeling process need to be investigated. In linear ARMA modeling the power spectrum of the input signal was found to play an important role. No requirements emerged however, on the probability density function (pdf) of the input. In nonlinear systems where the behavior is inherently level dependent, it is intuitively appealing to use an input signal with a flat power spectrum across the frequency range of interest and whose amplitude is uniformly distributed over the range of interest.

3. The inability of the adaptive lattice method to track zero valued overmodeled parameters is an interesting problem warranting further consideration. If some means of detecting the degeneration of the cost surface towards an infinite trough can be found, the problem could be remedied by simply reseting the appropriate parameters to zero.
4. Experimental experience needs to be gained with the nonlinear ARMA model itself and the lattice methods developed for it.
5. The characteritics of the lattice solution methods need to be further quantified to gain a comprehensive understanding of how and why it performs as it does. Also, further comparisons should be made between the lattice methods and conventional methods. Some comparisons were made here for batch processing methods but only a rectangular window was used on the data. Comparisons should be made using other types of window functions in the brute force method and the adaptive lattice method should also be compared with a conventional LMS adaptive algorithm applied to the equation error model in which all of the a(i) and b(i) coefficients are adapted simultaneously.
6. In the adaptive lattice method, scaling of the lattice input signals needs to be investigated. It was noted that the ratio of largest to smallest eigenvalue

of the input autocorrelation matrix was related to the speed of convergence of the adaptive algorithm. For the first lattice stage this matrix is

$$p^{(0)} = \varepsilon\{\begin{bmatrix} y^2(k) & y(k)\,u(k) \\ u(k)\,y(k) & u^2(k) \end{bmatrix}\}$$

If the system has a high gain such that the mean square value of the output y(k) is very much greater than that of the input u(k), convergence will be slow. This could be remedied by implementing an adaptive scaling scheme at the lattice input (perhaps similar to the first order low pass filter estimates used for adaptive step size normalization).

APPENDIX A

Alternate Multichannel Model Forms

Multichannel generalization of the MA and AR models were discussed in Chapter II along with their solution via the Levinson algorithm. Here the multichannel models and the Levinson algorithm are developed in an alternate form more compatible with other linear and nonlinear modeling problems to be explored later.

Consider first an N-th order, Q_0-channel, AR model where the current value of the signal vector $\underline{x}(k)$

$$\underline{x}(k) = [x_1(k) \ \ldots \ x_{Q_0}(k)]^T \tag{A.1a}$$

$$\underline{X}(z) = [X_1(z) \ \ldots \ X_{Q_0}(y)]^T \tag{A.1b}$$

is to be predicted from a weighted combination of N past values of each of the component signals. For each signal this estimate can be written as

$$x_j(k) = \sum_{i=1}^{Q_0} \sum_{n=1}^{N} d_{ij}(n) \, x_i(k-n) \tag{A,2a}$$

or

$$\hat{X}_j(k) = \sum_{i=1}^{Q_0} d_{ij}(z) \, X_i(z) \tag{A2.b}$$

where

$$d_{ij}(z) = \sum_{n=1}^{N} d_{ij}(n) \, z^{-n} \tag{A.2c}$$

Define an N-vector for each of the Q_0 channels to contain their required time histories as

$$\underline{x}_i(k) = [x_i(k-1) \ \ldots \ x_i(k-N)]^T \tag{A.3a}$$

for $1 \leq i \leq Q_0$ and embed all of these vectors into a single NQ_0-data vector given by

$$\underline{\mathbf{X}}(k) = [\underline{x}_1(k)^T \quad \ldots \quad \underline{x}_{Q_0}(k)^T]^T \tag{A.3b}$$

Define a $NQ_0 \times Q_0$ matris of weights as

$$\underline{\mathbf{D}} = \begin{bmatrix} \underline{d}_{11} & \cdots & \underline{d}_{1Q_0} \\ \vdots & & \vdots \\ \underline{d}_{Q_0 1} & & \underline{d}_{Q_0 Q_0} \end{bmatrix} \tag{A.4a}$$

where the N-vectors $\underline{d}_{ij}$ are given by

$$\underline{d}_{ij} = [d_{ij}(1) \ \ldots \ d_{ij}(N)]^T \tag{A.4b}$$

and contain the coefficients of the polynomials $d_{ij}(z)$. These polynomials can be combined in a matrix polynomial defined by

$$\underline{\mathbf{D}}(z) = \begin{bmatrix} d_{11}(z) & \ldots & d_{1Q_0}(z) \\ \vdots & & \vdots \\ d_{Q_0 1}(z) & \ldots & d_{Q_0 Q_0}(z) \end{bmatrix} \tag{A.4c}$$

With these definitions, the N-th order prediction of $\underline{x}(k)$ and its associated prediction error vector becomes

$$\hat{\underline{x}}(k)^T = \underline{\mathbf{X}}(k)^T \underline{\mathbf{D}} \tag{A.5a}$$

$$\underline{e}(k)^T = \underline{x}(k)^T - \hat{\underline{x}}(k)^T \tag{A.5b}$$

or in the transform domain

$$\underline{E}(z)^T = \underline{X}(z)^T [\underline{I} - \underline{\mathbf{D}}(z)] \tag{A.5c}$$

Comparison of equation (A.5c) and (2.44c) show that this matrix polynomial differs from the more generally used form of (2.44c) by a transposition. The coefficient matrix $\underline{\mathbf{D}}$ can be found by minimizing the trace of the prediction error covariance matrix

$$\underline{P} = \varepsilon\{\underline{e}(k)\ \underline{e}(k)^T\} \tag{A.6}$$

leading to a system of linear equations given by

$$\varepsilon\{\underline{\mathbf{X}}(k)\ \underline{\mathbf{X}}(k)^T\}\ \underline{\mathbf{D}} = \varepsilon\{\underline{\mathbf{X}}(k)\ \underline{x}(k)^T\} \tag{A.7a}$$

or

$$\begin{bmatrix} \underline{R}_{x_1x_1} & \cdots & \underline{R}_{x_1x_{Q_0}} \\ \vdots & & \vdots \\ \underline{R}_{x_{Q_0}x_1} & \cdots & \underline{R}_{x_{Q_0}x_{Q_0}} \end{bmatrix} \underline{\mathbf{D}} = \begin{bmatrix} \underline{r}_{x_1x_1} & \cdots & \underline{r}_{x_1x_{Q_0}} \\ \vdots & & \vdots \\ \underline{r}_{x_{Q_0}x_1} & & \underline{r}_{x_{Q_0}x_{Q_0}} \end{bmatrix} \tag{A.7b}$$

Adopting a shorthand notation, this becomes

$$\underline{\mathbf{R}}\,\underline{\mathbf{D}} = \underline{\mathbf{r}} \tag{A.7c}$$

As in the case of a single channel autoregression, the multichannel generalization of the Levinson algorithm also requires that the backward prediction problem of estimating $\underline{x}(k-N)$ backward in time from the values of $\underline{x}(k-N+1)$ through $\underline{x}(k)$, be solved simultaneously. Using an overscore to indicate quantities associated with the backward problem it follows that

$$\overline{\underline{\mathbf{X}}}(k)^T\,\overline{\underline{\mathbf{D}}} = \hat{\underline{x}}(k)^T \tag{A.8a}$$

$$\overline{\underline{e}}(k)^T = \underline{x}(k-N)^T - \hat{\underline{x}}(k)^T \tag{A.8b}$$

or

$$\overline{\underline{E}}(z)^T = \underline{X}(z)^T z^{-N}[\underline{I} - \overline{\underline{\mathbf{D}}}(z)] \tag{A.8c}$$

where $$\overline{\underline{\mathbf{X}}}(k) = [\overline{\underline{x}}_1(k)^T \quad \ldots \quad \overline{\underline{x}}_{Q_0}(k)^T]^T \tag{A.8d}$$

$$\overline{\underline{X}}_i(k) = [x_i(k-N+1) \ \ldots \ x_i(k)]^T \tag{A.8e}$$

and $$\overline{\underline{\mathbf{D}}} \quad \begin{bmatrix} \overline{\underline{d}}_{11} & \cdots & \overline{\underline{d}}_{1Q_0} \\ \vdots & & \vdots \\ \overline{\underline{d}}_{Q_0 1} & \cdots & \overline{\underline{d}}_{Q_0 Q_0} \end{bmatrix} \tag{A.8f}$$

$$\underline{\bar{d}}_{ij} = [\bar{d}_{ij}(1) \quad \ldots \quad \bar{d}_{ij}(N)]^T \tag{A.8g}$$

with $$\overline{\mathbf{D}}(z) = \begin{bmatrix} \bar{d}_{11}(z) & & \bar{d}_{1Q_0}(z) \\ \vdots & & \\ \bar{d}_{Q_0 1}(z) & \ldots & \bar{d}_{Q_0 Q_0} \end{bmatrix} \tag{A.8h}$$

and $$\bar{d}_{ij}(z) = \sum_{n=1}^{N} \bar{d}_{ij}(n)\, z^n \tag{A.8i}$$

Setting the coefficients of this backward predictor to minimize the trace of the backward prediction error covariance matrix

$$\underline{\bar{P}} = \varepsilon\{\underline{\bar{e}}(k)\, \underline{\bar{e}}(k)^T\} \tag{A.9}$$

leads to a MMSE solution given by

$$\varepsilon\{\overline{\mathbf{X}}(k)\, \overline{\mathbf{X}}(k)^T\}\, \overline{\mathbf{D}} = \varepsilon\{\overline{\mathbf{X}}(k)\, \underline{x}(k-N)^T\} \tag{A.10a}$$

and since $\varepsilon\{\underline{\bar{x}}_i(k)\, \underline{\bar{x}}_j(k)^T\} = \underline{R}_{x_i x_j}^T$

and $\varepsilon\{\underline{\bar{x}}_i(k)\, \underline{x}_j(k-N)\} = \underline{r}_{x_j x_i}$

this can be written as

$$\begin{bmatrix} \underline{R}_{x_1x_1}^T & \cdots & \underline{R}_{x_1x_{Q_0}}^T \\ \vdots & & \vdots \\ \underline{R}_{x_{Q_0}x_1}^T & \cdots & \underline{R}_{x_{Q_0}x_{Q_0}}^T \end{bmatrix} \overline{\underline{D}} = \begin{bmatrix} \underline{r}_{x_1x_1} & \cdots & \underline{r}_{x_{Q_0}x_1} \\ \vdots & & \vdots \\ \underline{r}_{x_1x_{Q_0}} & & \underline{r}_{x_{Q_0}x_{Q_0}} \end{bmatrix} \tag{A.10b}$$

Adopting a shorthand notation, this becomes

$$\overline{\underline{R}}\ \overline{\underline{D}} = \overline{\underline{r}} \tag{A.11}$$

At this point it is important to take note of a subtle difference between the single and multichannel problems that has arisen. While the single channel equivalent of equation (A.10b) for the backward predictor was not written previously, it is obviously

$$\underline{R}_{x_1x_1}^T \underline{d} = \underline{r}_{x_1x_1} \tag{A.12a}$$

and since

$$\underline{R}_{x_1x_1}^T = \underline{R}_{x_1x_1} \tag{A.12b}$$

it follows that the single channel forward and backward prediction problems have exactly the same solution (as was found to be the case in Section II.D. when the AR prediction error lattice was developed). This fact is responsible for

a number of simplifications in the single channel case:

a.) Only a single reflection coefficient $k^{(n+1)}$ was needed to link the n-th and (n+1)-st order solutions for both the backward and forward problems (see equations (2.25b) and (2.25c)).

b.) $|B(z)| = |\overline{B}(z)|$

c.) $\varepsilon\{e(k)^2\} = \varepsilon\{\overline{e}(k)^2\}$

d.) The development of the Burg method and the Itakaura-Saito method for calculating the reflection coefficients are a direct consequence of c above.

Unfortunately however, none of these simplifications carry over into the more general multichannel AR problem because by comparing equations (A.7b) and (A.10b) it is evident that in general $\overline{\underline{\mathbf{D}}} \neq \underline{\mathbf{D}}$.

Proceeding as in the single channel case, it is shown in Appendix B that because of the structure in the correlation matricies $\underline{\mathbf{R}}$ and $\overline{\underline{\mathbf{R}}}$, equations (A.7c) and (A.11) can be re-written as

$$\underline{\mathbf{R}}\ \underline{\mathbf{F}} = \underline{\rho} \qquad \text{(A.13a)}$$

and

$$\underline{\mathbf{R}}\ \overline{\underline{\mathbf{F}}} = \overline{\underline{\rho}} \qquad \text{(A.13b)}$$

where $\underline{\mathbf{F}}$, $\overline{\underline{\mathbf{F}}}$, $\underline{\rho}$ and $\overline{\underline{\rho}}$ are obtained from $\underline{\mathbf{D}}$, $\overline{\underline{\mathbf{D}}}$, $\underline{\mathbf{r}}$ and $\overline{\underline{\mathbf{r}}}$ respectively by taking their component vectors and turning them upside down in place; i.e.

$$\underline{\mathbf{F}} = \begin{bmatrix} \underline{f}_{11} & \cdots & \underline{f}_{1Q_0} \\ \vdots & & \vdots \\ \underline{f}_{Q_0 1} & \cdots & \underline{f}_{Q_0 Q_0} \end{bmatrix} \tag{A.14a}$$

where

$$\underline{f}_{ij} = [d_{ij}(N) \ \ldots \ d_{ij}(1)]^T \tag{A.14b}$$

and

$$\underline{\boldsymbol{\rho}} = \begin{bmatrix} \underline{\rho}_{x_1 x_1} & \cdots & \underline{\rho}_{x_1 x_{Q_0}} \\ \vdots & & \vdots \\ \underline{\rho}_{x_{Q_0} x_1} & \cdots & \underline{\rho}_{x_{Q_0} x_{Q_0}} \end{bmatrix} \tag{A.14c}$$

with

$$\underline{\rho}_{x_i x_j} = [R_{x_i x_j}(N) \ \ldots \ R_{x_i x_j}(1)]^T \tag{A.14d}$$

As will soon be evident, the relationships of equations (A.13) form the cornerstone for the Levinson algorithm and are made possible because the blocks comprising the $\underline{\mathbf{R}}$ and $\overline{\underline{\mathbf{R}}}$ matricies are themselves Toeplitz and because $\underline{R}_{x_i x_j} = \underline{R}^T_{x_j x_i}$.

Assume that the (n+1)-st order solutions can be related to the n-th order solutions by

$$\underline{d}_{ij}^{(n+1)} = \begin{bmatrix} \underline{d}_{ij}^{(n)} \\ \hline 0 \end{bmatrix} + \begin{bmatrix} \underline{\varepsilon}_{ij}^{(n)} \\ \hline k_{ij}^{(n+1)} \end{bmatrix} \tag{A.15a}$$

and

$$\underline{\bar{d}}_{ij}^{(n+1)} = \begin{bmatrix} \underline{\bar{d}}_{ij}^{(n)} \\ \hline 0 \end{bmatrix} + \begin{bmatrix} \underline{\bar{\varepsilon}}_{ij}^{(n)} \\ \hline \bar{k}_{ij}^{(n+1)} \end{bmatrix} \qquad \text{(A.15b)}$$

Embedding the vectors $\underline{\varepsilon}_{ij}^{(n)}$ and $\underline{\bar{\varepsilon}}_{ij}^{(n)}$ and the coefficients $k_{ij}^{(n+1)}$ and $\bar{k}_{ij}^{(n+1)}$ into matrices designated $\underline{\mathcal{E}}^{(n)}$ and $\underline{\bar{\mathcal{E}}}^{(n)}$, $\underline{K}^{(n+1)}$ and $\underline{\bar{K}}^{(n+1)}$ and solving for them in the (n_1)-st order problem it is shown in Appendix C that

$$\underline{\mathcal{E}}^{(n)} = - \underline{\bar{\mathbf{F}}}^{(n)} \underline{K}^{(n+1)} \qquad \text{(A.16a)}$$

$$\underline{\bar{\mathcal{E}}}^{(n)} = - \underline{\mathbf{F}}^{(n)} \underline{\bar{K}}^{(n+1)} \qquad \text{(A.16b)}$$

$$\underline{K}^{(n+1)} = [\underline{R}_{xx}(0) - \underline{\bar{\mathbf{r}}}^{(n)T} \underline{\bar{\mathbf{D}}}^{(n)}]^{-1} [\underline{R}_{xx}(n+1) - \underline{\bar{\rho}}^{(n)T} \underline{\mathbf{D}}^{(n)}] \qquad \text{(A.16c)}$$

$$\underline{\bar{K}}^{(n+1)} = [\underline{R}_{xx}(0) - \underline{\mathbf{r}}^{(n)T} \underline{\mathbf{D}}^{(n)}]^{-1} [R_{xx}(n+1)^{T} - \underline{\rho}^{(n)T} \underline{\bar{\mathbf{D}}}^{(n)}] \qquad \text{(A.16d)}$$

Also the inverted terms on the right hand side of equations (A.16c) and (A.16d) are just the backward and forward prediction error covariance matrices for the optimum n-th order models so that (A.16c) and (A.16d) become

$$\underline{K}^{(n+1)} = \underline{\bar{P}}^{(n)^{-1}}[\underline{R}_{xx}(n+1)-\underline{\bar{\rho}}^{(n)^T}\underline{\mathbf{D}}^{(n)}] \tag{A.17a}$$

$$\underline{\bar{K}}^{(n+1)} = \underline{P}^{(n)^{-1}}[\underline{R}_{xx}(n+1)^T-\underline{\rho}^{(n)^T}\underline{\bar{\mathbf{D}}}^{(n)}] \tag{A.17b}$$

As in the case of the single channel problem, the prediction error covariance matrices obey the recursions

$$\underline{P}^{(n+1)} = \underline{P}^{(n)}[I-\underline{\bar{K}}^{(n+1)}\underline{K}^{(n+1)}] \tag{A.18a}$$

$$\underline{\bar{P}}^{(n+1)} = \underline{\bar{P}}^{(n)}[I-\underline{K}^{(n+1)}\underline{\bar{K}}^{(n+1)}] \tag{A.18b}$$

completing the multichannel generalization of the Levinson algorithm for AR models.

Comparing equations (A.16), (A.17) and (A.18) with their single channel counterparts (2.18) and (2.19) it is clear that the multichannel Levinson algorithm simply represents a matrix algebra generalization of the single channel algorithm. Once again, predictors of all orders $0 \leq n \leq N$ are obtained in the process of finding the N-th order predictor along with all their prediction error covariance matrices and the overall minimization requiring the inversion of a $Q_0N \times Q_0N$ matrix is replaced by a sequence of N minimizations, each requireing the inversion of two $Q_0 \times Q_0$ matrices ($\underline{P}^{(n)}$ and $\underline{\bar{P}}^{(n)}$).

Using the relationships given in equations (A.15) and (A.16), successive orders of matrix polynomials can also be related to one another by

$$[\underline{I}-\underline{\mathbf{D}}^{(n+1)}(z)] = [\underline{I}-\underline{\mathbf{D}}^{(n)}(z)]-z^{-1}z^{-n}[\underline{I}-\overline{\underline{\mathbf{D}}}^{(n)}(z)]\underline{K}^{(n+1)} \tag{A.19a}$$

$$z^{-(n+1)}[\underline{I}-\overline{\underline{\mathbf{D}}}^{(n+1)}(z)]=z^{-1}z^{-n}[\underline{I}-\overline{\underline{\mathbf{D}}}^{(n)}(z)]-[\underline{I}-\underline{\mathbf{D}}^{(n)}(z)]\overline{\underline{K}}^{(n+1)} \tag{A.19b}$$

Premultiplying both sides of these equations by $\underline{X}(z)^T$, transposing, and transforming into the time domain provides relationships between the prediction error signals at each stage.

$$\underline{e}^{(n+1)}(k) = \underline{e}^{(n)}(k) - \underline{K}^{(n+1)^T}\,\overline{\underline{e}}^{(n)}(k-1) \tag{A.20a}$$

$$\overline{\underline{e}}^{(n+1)}(k) = \overline{\underline{e}}^{(n)}(k-1) - \overline{\underline{K}}^{(n+1)^T}\,\underline{e}^{(n)}(k) \tag{A.20b}$$

Recognizing that for a zeroth order prediction, the forward and backward prediction error vectors are just the input vector itself, it follows that

$$\underline{e}^{(0)}(k) = \overline{\underline{e}}^{(0)}(k) = \underline{x}(k) \tag{A.20c}$$

and the multichannel equivalents of equations (2.28) have been obtained.

Next consider the N-th order multichannel MA model in which the current value of a Q_0-vector of output signals $\underline{y}(k)$ is to be predicted from the present value and N past values of a Q_i-vector of input signals $\underline{x}(k)$.

$$\hat{y}_j(k) = \sum_{i=1}^{Q_i} \sum_{n=0}^{N} d_{ij}(n)\, x_i(k-n) \qquad \text{(A.21a)}$$

or

$$\hat{Y}_j(z) = \sum_{i=1}^{Q_i} d_{ij}^{+}(z)\, X_i(z) \qquad \text{(A.21b)}$$

where

$$d_{ij}^{+}(z) = \sum_{n=0}^{N} d_{ij}(n) z^{-n} \qquad \text{(A.21c)}$$

Using a superscript "+" to indicate the fact that the vectors are indexed from 0 to N rather than from 1 to N, define (N+1)-vectors for each of the input channels to contain their required time histories as

$$\underline{x}_j^{+}(k) = [x_j(k) \quad \ldots \quad x_j(k-n)]^T \qquad \text{(A.22a)}$$

and embed these vectors into a single $Q_i(N+1)$-data vector given by

$$\underline{\mathbf{X}}^{+}(k) = [\underline{x}_1^{+}(k)^T \quad \ldots \quad \underline{x}_{Q_i}^{+})k)^T]^T \qquad \text{(A.22b)}$$

Define a $Q_i(N+1) \times Q_0$ matrix of weights as

$$\underline{\mathbf{D}}^+ = \begin{bmatrix} \underline{d}^+_{11} & \cdots & \underline{d}^+_{1Q_0} \\ \vdots & & \vdots \\ \underline{d}^+_{Q_i 1} & \cdots & \underline{d}^+_{Q_i Q_0} \end{bmatrix} \tag{A.23a}$$

where the (N+1)-vectors $\underline{d}^+_{ij}$ are given by

$$\underline{d}^+_{ij} = [d_{ij}(0) \ \ldots \ d_{ij}(N)]^T \tag{A.23b}$$

and contain the coefficients of the polynomial $d^+_{ij}(z)$. These polynomials can be combined into a single matrix polynomial given by

$$\underline{\mathbf{D}}^+(z) = \begin{bmatrix} d^+_{11}(z) & \cdots & d^+_{1Q_0}(z) \\ \vdots & & \vdots \\ d^+_{Q_i 1}(z) & \cdots & d^+_{Q_i Q_0}(z) \end{bmatrix} \tag{A.23c}$$

With these definitions, the N-th order MA prediction of $\underline{y}(k)$ is given by

$$\hat{\underline{y}}(k)^T = \underline{\mathbf{X}}^+(k)^T \ \underline{\mathbf{D}}^+ \tag{A.24a}$$

or in the transform domain,

$$\hat{\underline{Y}}(z)^T = \underline{X}(z)^T \, \underline{\mathbf{D}}^+(z) \tag{A.24b}$$

Defining a prediction error vector as

$$\underline{e}_0(k)^T = \underline{y}(k)^T - \hat{\underline{y}}(k)^T \tag{A.25a}$$

and setting the coefficients in $\underline{\mathbf{D}}^+$ to minimize the trace of the prediction error covariance matrix

$$\underline{P}_0 = \varepsilon\{\underline{e}_0(k)\underline{e}_0(k)^T\} \tag{A.25b}$$

results in a solution given by

$$\varepsilon\{\underline{\mathbf{X}}^+(k) \, \underline{\mathbf{X}}^+(k)^T\} \, \underline{\mathbf{D}}^+ = \varepsilon\{\underline{\mathbf{X}}^+(k) \, \underline{y}(k)^T\} \tag{A.26a}$$

or

$$\begin{bmatrix} \underline{R}_{x_1^+ x_1^+} & \cdots & \underline{R}_{x_1^+ x_{Q_1}^+} \\ \vdots & & \vdots \\ \underline{R}_{x_{Q_i}^+ x_1^+} & \cdots & \underline{R}_{x_{Q_i}^+ x_{Q_i}^+} \end{bmatrix} \underline{\mathbf{D}}^+ = \begin{bmatrix} \underline{r}_{x_1^+ y_1} & \cdots & \underline{r}_{x_1^+ y_{Q_0}} \\ \vdots & & \vdots \\ \underline{r}_{x_{Q_i}^+ y_1} & \cdots & \underline{r}_{x_{Q_i}^+ y_{Q_0}} \end{bmatrix} \tag{A.26b}$$

Adopting a shorthand notation this becomes

$$\underline{\mathbf{R}}^{+}\ \underline{\mathbf{D}}^{+} = \underline{\mathbf{r}}^{+} \tag{A.26c}$$

Assume a relationship between the components of the (n+1)-st and n-th order solutions given by

$$\underline{d}_{ij}^{+(n+1)} = \begin{bmatrix} \underline{d}_{ij}^{+(n)} \\ \hline 0 \end{bmatrix} + \begin{bmatrix} \gamma_{ij}^{(n+1)} \\ \hline g_{ij}^{(n+1)} \end{bmatrix} \tag{A.27}$$

Embedding the vectors $\gamma_{ij}^{(n+1)}$ and the coefficients $g_{ij}^{(n+1)}$ into matrices designated $\underline{\mathcal{J}}^{(n+1)}$ and $\underline{G}^{(n+1)}$, and solving for them in the (n+1)-st order MA problem it is shown in Appendix D that

$$\underline{\mathcal{J}}^{(n+1)} = -\overline{\underline{\mathbf{F}}}^{(n+1)}\ \underline{G}^{(n+1)} \tag{A.28a}$$

$$\underline{G}^{(n+1)} = \overline{\underline{P}}^{(n+1)^{-1}}[\underline{R}_{xy}(n+1) - \overline{\underline{\rho}}^{(n+1)^{T}}\underline{\mathbf{D}}^{+(n)}] \tag{A.28b}$$

where $\overline{\underline{\mathbf{F}}}^{(n+1)}$ $\overline{\underline{\rho}}^{(n+1)}$ amd $\overline{\underline{P}}^{(n+1)}$ are matrices that emerge from the backward prediction problem in a multichannel (n+1)-st order autoregression on the input signal vector $\underline{x}(k)$. Again, it is clear that the multichannel MA solutions given by equations (A.28) are a matrix algebra generalization of equations (2.23) for the single channel MA model.

To relate successive order MA matrix polynomials to one another, equations (A.27) and (A.28a) can be used to write

$$\underline{\mathbf{D}}^{+}(z)^{(n+1)} = \underline{\mathbf{D}}^{+}(z)^{(n)} + z^{-n}[\underline{I} - \overline{\underline{\mathbf{D}}}^{(n)}(z)]\underline{G}^{(n+1)} \qquad (A.29)$$

where the second term on the right hand side that premultiplies G is the backward prediction error matrix polynomial from the autoregression on the input signal $\underline{x}(k)$. Premultiplying by $\underline{X}^{T}(z)$, transposing, and transforming into the time domain results in the multichannel equivalent of equation (2.37)

$$\hat{\underline{y}}^{(n+1)}(k) = \hat{\underline{y}}^{(n)}(k) + \underline{G}^{(n+1)^{T}}\underline{e}^{(n+1)}(k) \qquad (A.30)$$

and completes the derivation of the recursive in order solutions for the multichannel MA model.

APPENDIX B

A Key Relationship For The Multichannel Levinson Algorithm

Equation (A.13) is a key relationship in the development of the Levinson algorithm responsible for much of the algorithm's simplicity. Without this relationship, more than just the $\underline{K}$ and $\overline{\underline{K}}$ matrices would be needed to obtain the new model from the previous model.

In equation (A.7), consider the multiplication of the i-th block row of $\mathbf{R}$ by the j-th block column of $\mathbf{D}$ to form $\underline{r}_{x_i x_j}$.

$$\underline{R}_{x_i x_1}\underline{d}_{1j} + \ldots + \underline{R}_{x_i x_{Q_c}}\underline{d}_{Q_0 j} = \underline{r}_{x_i x_j} \tag{B.1}$$

In particular, consider a general term on the left hand side of equation (B.1) in detail.

$$\ldots + \begin{bmatrix} R_{x_i x_m}(0) & \ldots & R_{x_i x_m}(1-N) \\ \vdots & & \vdots \\ R_{x_i x_m}(N-1) & & R_{x_i x_m}(0) \end{bmatrix} \begin{bmatrix} d_{mj}(1) \\ \vdots \\ d_{mj}(N) \end{bmatrix} + \ldots = \begin{bmatrix} R_{x_i x_j}(1) \\ \vdots \\ R_{x_i x_j}(N) \end{bmatrix} \tag{B.2}$$

Define upside down versions of the $\underline{d}_{ij}$ and $\underline{r}_{x_i x_j}$ vectors as

$$\underline{f}_{ij} = \begin{bmatrix} d_{ij}(N) \\ \vdots \\ d_{ij}(1) \end{bmatrix} \tag{B.3a}$$

and

$$\underline{\rho}_{x_i x_j} = \begin{bmatrix} R_{x_i x_j}(N) \\ \vdots \\ R_{x_i x_j}(1) \end{bmatrix} \qquad \text{(B.3b)}$$

Using these permuted vectors in equation (B.1) in place of the $\underline{d}$ and $\underline{r}$ vectors, the relationship is still satisfied if the $\underline{R}$ matrices are permuted as well. In particular, from (B.2) it is evident that

$$\ldots + \begin{bmatrix} R_{x_i x_m}(0) & \ldots & R_{x_i x_m}(N-1) \\ R_{x_i x_m}(1-N) & \ldots & R_{x_i x_m}(0) \end{bmatrix} \begin{bmatrix} d_{ij}(N) \\ d_{ij}(1) \end{bmatrix}$$

$$+ \ldots = \begin{bmatrix} R_{x_i x_j}(N) \\ R_{x_i x_j}(1) \end{bmatrix} \qquad \text{(B.4a)}$$

or equivalently

$$\underline{R}^{T}_{x_i x_1} \underline{f}_{1j} + \ldots + \underline{R}^{T}_{x_i x_{Q_0}} \underline{f}_{Q_0 j} = \underline{\rho}_{x_i x_j} \qquad \text{(B.4b)}$$

Embedding the $\underline{f}_{ij}$ and $\underline{\rho}_{x_i x_j}$ vectors into matrices designated $\underline{\mathbf{F}}$ and $\underline{\boldsymbol{\rho}}$ respectively and using the definition of $\overline{\mathbf{R}}$ it follows that

$$\overline{\mathbf{R}}\,\underline{\mathbf{F}} = \underline{\boldsymbol{\rho}} \qquad \text{(B.5)}$$

Defining $\underline{\underline{F}}_{ij}$ and $\underline{\underline{\rho}}_{x_i x_j}$ as upside down versions of their corresponding vectors in $\underline{\underline{\mathbf{D}}}$ and $\underline{\underline{\mathbf{r}}}$ in equation (A.11) and embedding them in matrices designated $\underline{\underline{\mathbf{F}}}$ and $\underline{\underline{\rho}}$, it also follows from a similar development that

$$\underline{\underline{\mathbf{R}}} \; \underline{\underline{\mathbf{F}}} = \underline{\underline{\rho}} \qquad \text{(B.6)}$$

Equations (B.5) and (B.6) are essential to the development of the Levinson algorithm and are made possible by the Toeplitz structure of the block components of the $\mathbf{R}$ and $\underline{\underline{\mathbf{R}}}$ matrices.

APPENDIX C

The Multichannel Levinson Algorithm Solution For AR Modeling

In the (n+1)-st order versions of equations (A.7) and (A.10), the component matrices that make up $\underline{\mathbf{R}}$ and $\overline{\underline{\mathbf{R}}}$ can be written as follows

$$\underline{R}_{x_i x_j}^{(n+1)} = \left[\begin{array}{c|c} \underline{R}_{x_i x_j}^{(n)} & \begin{matrix} R_{x_i x_j}(-N) \\ \vdots \\ R_{x_i x_j}(-1) \end{matrix} \\ \hline R_{x_i x_j}(N) \cdots R_{x_i x_j}(1) & R_{x_i x_j}(0) \end{array}\right]$$

$$= \left[\begin{array}{c|c} R_{x_i x_j}^{(n)} & \overline{\underline{\rho}}_{x_i x_j}^{(n)} \\ \hline \underline{\rho}_{x_j x_i}^{(n)T} & R_{x_i x_j}(0) \end{array}\right] \qquad \text{(C.1)}$$

and

$$\overline{\underline{R}}_{x_i x_j}^{(n+1)} = \underline{R}_{x_i x_j}^{(n+1)T} = \left[\begin{array}{c|c} \underline{R}_{x_i x_j}^{(n)T} & \underline{\rho}_{x_i x_j}^{(n)} \\ \hline \underline{\rho}_{x_i x_j}^{(n)T} & R_{x_i x_j}(0) \end{array}\right] \qquad \text{(C.2)}$$

Additionally,

$$\underline{r}_{x_i x_j}^{(n+1)} = \begin{bmatrix} \underline{r}_{x_i x_j}^{(n)} \\ \hline R_{x_i x_j}(n+1) \end{bmatrix} \qquad (C.3)$$

With these matrices and vectors written in this partitioned form, and with the relationships assumed in equations (A.15) between the n-th and (n+1)-st order solutions, the (n+1)-st order modeling equations become:

Forward Model:

$$\underline{\mathbf{R}}^{(n)}\ \underline{\mathbf{D}}^{(n)} + \underline{\mathbf{R}}^{(n)}\ \underline{\epsilon}^{(n)} + \overline{\underline{\rho}}^{(n)}\underline{K}^{(n+1)} = \underline{\mathbf{r}}^{(n)} \qquad (C.4a)$$

$$\overline{\underline{\rho}}^{(n)^T}\ \underline{\mathbf{D}}^{(n)} + \overline{\underline{\rho}}^{(n)^T}\ \underline{\epsilon}^{(n)} + \underline{R}_{xx}(0)\underline{K}^{(n+1)} = \underline{R}_{xx}(n+1) \qquad (C.4b)$$

Backward Model:

$$\overline{\underline{\mathbf{R}}}^{(n)}\ \overline{\underline{\mathbf{D}}}^{(n)} + \underline{\mathbf{R}}^{(n)^T}\ \overline{\underline{\epsilon}}^{(n)} + \underline{\rho}^{(n)}\ \overline{\underline{K}}^{(n+1)} = \overline{\underline{\mathbf{r}}}^{(n)} \qquad (C.5a)$$

$$\underline{\rho}^{(n)^T}\ \overline{\underline{\mathbf{D}}}^{(n)} + \underline{\rho}^{(n)^T}\ \overline{\underline{\epsilon}}^{(n)} + \underline{R}_{xx}(0)\overline{\underline{K}}^{(n+1)} = \underline{R}_{xx}(n+1)^T \qquad (C.5b)$$

Equations (C.4a) and (C.5a) contain the n-th order modeling equations within them however, and therefore can be written as

$$\underline{\mathbf{R}}^{(n)}\ \underline{\epsilon}^{(n)} = -\ \overline{\underline{\rho}}^{(n)}\ \underline{K}^{(n+1)} \tag{C.6a}$$

$$\overline{\underline{\mathbf{R}}}^{(n)}\ \overline{\underline{\epsilon}}^{(n)} = -\ \underline{\rho}^{(n)}\ \overline{\underline{K}}^{(n+1)} \tag{C.6b}$$

and applying the relationship developed in Appendix B (equations (B.5) and (B.6)) yields

$$\underline{\epsilon}^{(n)} = -\ \overline{\underline{\mathbf{F}}}^{(n)}\ \underline{K}^{(n+1)} \tag{C.7a}$$

$$\overline{\underline{\epsilon}}^{(n)} = -\ \underline{\mathbf{F}}^{(n)}\ \overline{\underline{K}}^{(n+1)} \tag{C.7b}$$

Thus the $\underline{\varepsilon}$ and $\overline{\underline{\varepsilon}}$ vectors have been found in terms of the known quanties $\underline{f}$ and $\overline{\underline{f}}$ and the unknown $\underline{K}$ and $\overline{\underline{K}}$ matrices. Substituting equations (C.7) into (C.4b) and (C.5b) completes the solution resulting in expressions for the $\underline{K}$ and $\overline{\underline{K}}$ matrices given by

$$\underline{K}^{(n+1)} = [\underline{R}_{xx}(0) - \overline{\underline{\mathbf{r}}}^{(n)T}\overline{\underline{\mathbf{D}}}^{(n)}]^{-1}[\underline{R}_{xx}(n+1) - \overline{\underline{\rho}}^{(n)T}\underline{\mathbf{D}}^{(n)}] \tag{C.8a}$$

$$\underline{\bar{K}}^{(n+1)} = [\underline{R}_{xx}(0) - \underline{\mathbf{r}}^{(n)T}\underline{\mathbf{D}}^{(n)}]^{-1}[\underline{R}_{xx}(n+1)^T - \underline{\rho}^{(n)T}\underline{\bar{\mathbf{D}}}^{(n)}] \tag{C.8b}$$

The terms on the right hand side of these equations that are inverted are just the backward and forward prediction error covariance matrices for the optimum n-th order models given by

$$\underline{P}^{(n)} = \underline{R}_{xx}(0) - \underline{\mathbf{r}}^{(n)T}\underline{\mathbf{D}}^{(n)} \tag{C.9a}$$

$$\underline{\bar{P}}^{(n)} = \underline{R}_{xx}(0) - \underline{\bar{\mathbf{r}}}^{(n)T}\underline{\bar{\mathbf{D}}}^{(n)} \tag{C.9b}$$

Using equations (C.3), (A.15) and (C.7), the forward prediction error covariance matrix for the optimum (n+1)-st order model can therefore be written as

$$\underline{P}^{(n+1)} = \underline{R}_{xx}(0) - \underline{\mathbf{r}}^{(n)T}\underline{\mathbf{D}}^{(n)} + \underline{\mathbf{r}}^{(n)T}\underline{\bar{\mathbf{F}}}^{(n)}\underline{K}^{(n+1)} - \underline{R}_{xx}(n+1)^T\underline{K}^{(n+1)} \tag{C.10a}$$

$$= \underline{P}^{(n)} + [\underline{\mathbf{r}}^{(n)T}\underline{\bar{\mathbf{F}}}^{(n)} - \underline{R}_{xx}(n+1)^T]\underline{K}^{(n+1)} \tag{C.10b}$$

$$= \underline{P}^{(n)}[\underline{I} - \underline{P}^{(n)-1}[\underline{R}_{xx}(n+1)^T - \underline{\rho}^{(n)T}\underline{\bar{\mathbf{D}}}^{(n)}]\underline{K}^{(n+1)}] \tag{C.10c}$$

$$\underline{P}^{(n+1)} = \underline{P}^{(n)}[\underline{I}-\overline{\underline{K}}^{(n+1)}\underline{K}^{(n+1)}] \qquad \text{(C.10d)}$$

and following a similar development for the backward prediction error covariance matrix results in

$$\overline{\underline{P}}^{(n+1)} = \overline{\underline{P}}^{(n)}[\underline{I} - \underline{K}^{(n+1)}\overline{\underline{K}}^{(n+1)}] \qquad \text{(C.11)}$$

APPENDIX D

The Multichannel Levinson Algorithm Solution For MA Modeling

First note that because of the definitions of the $\underline{x}_i^+(k)$ and $\underline{x}_i(k)$ vectors (indexed from 0 to n and 1 to n respectively) in the n-th order models, the matrices $\underline{R}^{(n)}_{x_i^+ x_j^+}$ in the n-th order MA problem could also be written in terms of $\underline{x}_i(k)$ and $\underline{x}_j(k)$ as (assuming stationarity)

$$\underline{R}^{(n)}_{x_i^+ x_j^+} = \underline{R}^{(n+1)}_{x_i x_j} \tag{D.1a}$$

so that

$$\underline{\mathbf{R}}^{+(n)} = \underline{\mathbf{R}}^{(n+1)} \tag{D.1b}$$

The components of $\underline{\mathbf{R}}^{+(n+1)}$ and $\underline{\mathbf{r}}^{+(n+1)}$ in the (n+1)-st order MA modeling problem can be written in partitioned form as

$$\underline{R}^{(n+1)}_{x_i^+ x_j^+} = \left[\begin{array}{c|c} \underline{R}^{(n)}_{x_i^+ x_j^+} & \underline{\rho}^{(n+1)}_{x_i x_j} \\ \hline \underline{\rho}^{(n+1)^T}_{x_j x_i} & R_{x_i x_j}(0) \end{array}\right] \tag{D.2a}$$

and

$$\underline{r}^{(n+1)}_{x_i^+ y_j} = \begin{bmatrix} \underline{r}^{(n)}_{x_i^+ y_j} \\ \hline R_{x_i y_j}(n+1) \end{bmatrix} \qquad \text{(D.2b)}$$

Using these partitioned forms, and the relationship in equation (A.27) between the n-th and (n+1)-st order solutions, the (n+1)=st order modeling equations become

$$\underline{\mathbf{R}}^{+(n)}\ \underline{\mathbf{D}}^{+(n)} + \underline{\mathbf{R}}^{+(n)}\ \underline{\mathcal{J}}^{(n+1)} + \overline{\underline{\rho}}^{(n+1)}\underline{G}^{(n+1)} = \underline{\mathbf{r}}^{+(n)} \qquad \text{(D.3a)}$$

$$\overline{\underline{\rho}}^{(n+1)^T}\underline{\mathbf{D}}^{+(n)} + \overline{\underline{\rho}}^{(n+1)^T}\ \underline{\mathcal{J}}^{(n+1)} + \underline{R}_{xx}(0)\underline{G}^{(n+1)}$$

$$= \underline{R}_{xy}(n+1) \qquad \text{(D.3b)}$$

Equation (D.3a) contains within it, the modeling equation for the n-th order MA model and using equation (D.1b) can be rewritten as

$$\underline{\mathbf{R}}^{(n+1)}\ \underline{\mathcal{J}}^{(n+1)} = -\ \overline{\underline{\rho}}^{(n+1)}\underline{G}^{(n+1)} \qquad \text{(D.4)}$$

A comparison of this result with equation (C.6a) shows that

$$\underline{\mathcal{J}}^{(n+1)} = -\ \overline{\underline{\mathbf{F}}}^{(n+1)}\underline{G}^{(n+1)} \qquad \text{(D.5)}$$

where $\overline{\underline{F}}^{(n+1)}$ is the permuted version of the backward solution in an (n+1)-st order AR model of the input signal vector $\underline{x}(k)$. Furthermore, substituting equation (D.5) into (D.3b) results in a solution for $\underline{G}^{(n+1)}$ given by

$$\underline{G}^{(n+1)} = [\underline{R}_{xx}(0) - \overline{\underline{\rho}}^{(n+1)^T}\overline{\underline{F}}^{(n+1)}]^{-1}$$

$$[\underline{R}_{xy}(n+1) - \overline{\underline{\rho}}^{(n+1)^T}\underline{D}^{+(n)}] \qquad (D.6)$$

Since

$$\overline{\underline{\rho}}^{(n+1)^T}\overline{\underline{F}}^{(n+1)} = \overline{\underline{r}}^{(n+1)^T}\overline{\underline{D}}^{(n+1)}$$

it follows that the inverted term on the right hand side of equation (D.6) is just the backward prediction error covariance matrix for the optimum (n+1)-st order AR model on the input signal vector $\underline{x}(k)$.

APPENDIX E

Prony's Method For ARMA Modeling

Prony's method [Refs. 8, 52 and 56] obtains a zero pole model for a system by matching the impulse responses of the system and model over the first M+N+1 sample intervals where M and N are the orders of the model transfer function numerator and denominator polynomials. Assume that a signal y(k) is available that represents the impulse response of a causal system and that a rational transfer function model for this systems is desired. Using a "^" to denote the model output and u(k) to denote the input to both the system and model, the model transfer function is given by

$$H(z) = \frac{\hat{Y}(z)}{u(z)} = \frac{\sum_{n=0}^{M} a(n)z^{-n}}{1+\sum_{n=1}^{N} b(n)z^{-n}} \qquad (E.1)$$

For a unit sample input it follows that

$$B(z)\,\hat{Y}(z) = A(z) \qquad (E.2)$$

Equating like powers of z in this relationship results in

$$\sum_{i=0}^{N} b(i)\,\hat{y}(n-i) = \begin{cases} a(n) ; & 0 \le n \le M \\ 0 & \text{else} \end{cases} \qquad (E.3)$$

where b(0)=1.

Equating $\hat{y}(k)$ and $y(k)$ over the interval $0 \leq k \leq M+N$ produces a set of M+N+1 equations which can be expressed in matrix form as

$$\left[\begin{array}{c|ccccc} y(0) & 0 & 0 & \cdots & 0 \\ y(1) & y(0) & 0 & \cdots & 0 \\ \vdots & \vdots & & & \\ y(M) & y(M-1) & y(M-2) & \cdots & \\ \hline y(M+1) & y(M) & y(M-1) & \cdots & \\ \vdots & & & & \\ y(M+N) & \cdots & & & y(M) \end{array}\right] \left[\begin{array}{c} 1 \\ \hline b(1) \\ \vdots \\ b(N) \end{array}\right] = \left[\begin{array}{c} a(0) \\ \vdots \\ \underline{a(M)} \\ 0 \\ \vdots \\ 0 \end{array}\right] \qquad \text{(E.4a)}$$

or adopting a shorthand notation

$$\begin{bmatrix} \underline{Y}_1^+ & \underline{Y}_1 \\ \underline{Y}_2 & \underline{Y}_2 \end{bmatrix} \begin{bmatrix} 1 \\ \underline{b} \end{bmatrix} = \begin{bmatrix} \underline{a}^+ \\ \underline{0} \end{bmatrix} \qquad \text{(E.4b)}$$

Assuming that the NxN matrix $\underline{Y}_2$ is not singular if follows that

$$\underline{b} = -Y_2^{-1}\, y_2 \qquad \text{(E.5a)}$$

$$\underline{a}^+ = \underline{y}_1^+ - \underline{Y}_1\, \underline{Y}_2^{-1} \underline{y}_2 \qquad \text{(E.5b)}$$

The original Prony method goes on to form a partial fraction expansion and inverse transformation on the model transfer function H(z) resulting in a model for the impulse response of the system given as a sum of complex exponentials. This in unnecessary here however, since a rational transfer function model is the form sought.

Prony's method inherently assumes that matching a sufficient portion of the impulse response of the system results in an accurate model but this is not necessarily the case unless the impulse response damps out quickly or unless the system can be represented exactly by a low order model. Otherwise a very high order model may be required to obtain sufficient accuracy. Other difficulties associated with this technique are:

1.) The system impulse response must be available;
2.) There are no built in mechanisms to test for or ensure stability of the model;
3.) There is no averaging of noise in the data;
4.) Only a small portion of the available data points (M+N+1) are actually used.

These difficulties can be partially overcome by modifying the procedure to obtain an approximate match of the system and model responses over their entire duration rather than an exact match over the first M+N+1 points. Consider equations (E.4) modified to include the entire signal y(k) for $0 \leq k \leq \infty$.

$$\left[\begin{array}{c|ccc} y(0) & 0 & \cdots & 0 \\ y(1) & y(0) & & \vdots \\ \vdots & \vdots & & \\ y(M) & y(M-1) & & \\ \hline y(M+1) & y(M) & \cdots & \\ \vdots & \vdots & & \vdots \\ \end{array}\right] \left[\begin{array}{c} 1 \\ \hline \underline{b} \end{array}\right] = \left[\begin{array}{c} \underline{a}^{+} \\ \hline 0 \\ \vdots \end{array}\right] \qquad \text{(E.6a)}$$

Adopting a shorthand notation this becomes

$$\left[\begin{array}{c|c} \underline{Y}_1^{+} & \underline{Y}_1 \\ \hline \underline{Y}_3 & \underline{Y}_3 \end{array}\right] \left[\begin{array}{c} 1 \\ \hline \underline{b} \end{array}\right] = \left[\begin{array}{c} \underline{a}^{+} \\ \hline \underline{0} \end{array}\right] \qquad \text{(E.6b)}$$

This yields two equations

$$\underline{Y}_1^{+} + \underline{Y}_1 \, \underline{b} = \underline{a}^{+} \qquad \text{(E.7a)}$$

$$\underline{Y}_3 + Y_3 \, \underline{b} = \underline{0} \qquad \text{(E.7b)}$$

but with $\underline{Y}_3$, $\underline{Y}_3$ and $\underline{0}$ having an infinite number of rows, equation (E.7b) will in general have no solution. In practice

only a finite portion of the system impulse response y(k) can be considered but equation (E.7b) will still be inconsistent in general. A least squares estimate of $\underline{b}$ can however, be obtained by minimizing $\underline{e}^T\underline{e}$ where

$$\underline{e} = \underline{Y}_3 \underline{b} - (-\underline{y}_3) \tag{E.8a}$$

resulting in

$$\underline{b} = (\underline{Y}_3^T\underline{Y}_3)^{-1} \underline{Y}_3^T \underline{y}_3 \tag{E.8b}$$

which in turn can be used in equation (E.7a) to find $\underline{a}^+$. This approximate version of Prony's method is the one most commonly applied.

APPENDIX F

Parabolic Surfaces In n-Dimensions

Multi-dimensional parabolic surfaces are described by an equation of the form

$$y = \underline{x}^T \underline{A} \underline{x} - 2\, \underline{x}^T\, \underline{b} + c \qquad \text{(F.1)}$$

where:

y is the independent variable;

$\underline{x}$ is a vector of dependent variables;

$\underline{A}$ is a symmetric constant matrix;

$\underline{b}$ is a constant vector;

c is a constant.

($\underline{x}$, $\underline{b}$ and c can also be considered as matricies with the trace of the right hand side set equal to the independent variable but the problem remains essentially unchanged.) Completing the square for nonsingular $\underline{A}$ this becomes

$$y = (\underline{x} - \underline{A}^{-1}\underline{b})^T\, \underline{A}(\underline{x} - \underline{A}^{-1}\underline{b}) + c - \underline{b}^T\underline{A}^{-1}\underline{b} \qquad \text{(F.2)}$$

so that for positive definite $\underline{A}$ it is clear that the minimum value of y is obtained for $\underline{x} = \underline{A}^{-1}\underline{b}$ and this minimum is $c - \underline{b}^T\underline{A}^{-1}\underline{b}$. It is also clear that nonzero values of $\underline{b}$ and c simply raise and lower the surface and move the minimum point away from $\underline{x} = \underline{0}$. The shape of the surface (its relative concavity or flatness) is determined by the matrix

$\underline{A}$ in the quadratic term of equation (F.1). Therefore, to study the shape of this surface, consider the simpler problem with $\underline{b}$ and c set to zero.

$$y = \underline{x}^T \underline{A} \underline{x} \tag{F.3}$$

One way to examine the relative flatness or concavity is to look at the locus of points on the surface for constant values of y; in particular when y=1. Recognize that $\underline{A}$ can be rewritten as $\underline{Q} \underline{\Lambda} \underline{Q}^T$ where $\underline{\Lambda}$ is a diagonal matrix of eigenvalues and $\underline{Q}$ is a matrix whose columns are the unit length eigenvectors of $\underline{A}$. Now (F.3) becomes

$$\underline{x}^T \underline{Q} \underline{\Lambda} \underline{Q}^T \underline{x} = 1 \tag{F.4}$$

and introducing a new set of variables $\underline{w}=\underline{Q}^T\underline{x}$ (which are simply a rotation of the variables in $\underline{x}$), equation (F.4) reduces to

$$\lambda_1 w_1^2 + \ldots + \lambda_n w_n^2 = 1 \tag{F.5}$$

This equation describes an ellipsoid in n dimensions whose axes half lengths are given by $1/\sqrt{\lambda_i}$ for $1 \leq i \leq n$. This follows from letting all but one of the w's equal to zero and solving for the nonzero variable so that one point on the surface is for example $w_1 = 1/\sqrt{\lambda_1}$ with $w_2 = \ldots = w_n = 0$.

This point is just a multiple of the first eigenvector of A so that in general, the axes of the ellipsoid point in the direction of the eigenvectors of A with half lengths given by the reciprocal of the square root of the eigenvalues.

APPENDIX G

Multichannel ARMA Modeling

Consider a system with Q_0 output signals $[y_1(k) \ldots y_{Q_0}(k)]$ and Q_i input signals $[u_1(k) \ldots u_{Q_i}(k)]$. The multichannel ARMA analysis model forms an estimate of the present value of each output as a weighted combination of past values of all output signals and past and present values of all input signals.

$$y_n(k) = \sum_{j=1}^{Q_0} \sum_{i=1}^{N} b_{jn}(i)\, y_j(k-i) + \sum_{j=1}^{Q_i} \sum_{i=0}^{N} a_{jn}(i)\, u_j(k-1) \qquad \text{(G.1)}$$

Define data vectors for all the input and output channels as

$$\underline{y}_n(k) = [y_n(k-1) \ldots y_n(k-N)]^T \qquad \text{(G.2a)}$$

$$\underline{u}_n^+(k) = [u_n(k) \ldots u_n(k-N)]^T \qquad \text{(G.2b)}$$

and embed them into Q_0N and $Q_i(N+1)$ vectors given by

$$\underline{Y} = [\underline{y}_1(k)^T \quad \ldots \quad \underline{y}_{Q_0}(k)^T]^T \qquad \text{(G.3a)}$$

$$\underline{U}^+ = [\underline{u}_1^+(k)^T \quad \ldots \quad \underline{u}_{Q_i}^+(k)^T]^T \qquad \text{(G.3b)}$$

Define $NQ_0 \times Q_0$ and $(N+1)Q_i \times Q_0$ matrices of coefficients given by

$$\underline{B} = \begin{bmatrix} \underline{b}_{11} & \cdots & \underline{b}_{1Q_0} \\ \vdots & & \vdots \\ \underline{b}_{Q_{01}} & \cdots & \underline{b}_{Q_0Q_0} \end{bmatrix} \tag{G.4a}$$

and

$$\underline{A}^+ = \begin{bmatrix} \underline{a}^+_{11} & \cdots & \underline{a}^+_{1Q_0} \\ \vdots & & \vdots \\ \underline{a}^+_{Q_i1} & & \underline{a}^+_{Q_iQ_0} \end{bmatrix} \tag{G.4b}$$

where

$$\underline{b}_{ij} = [b_{ij}(1) \quad \cdots \quad b_{ij}(N)]^T \tag{G.4c}$$

$$\underline{a}^+_{ij} = [a_{ij}(0) \quad \cdots \quad a_{ij}(N)] \tag{G.4d}$$

With these definitions, the multichannel ARMA estimate for the vector of output signals becomes

$$\hat{\underline{y}}(k)^T = [\underline{Y}^T \,\vdots\, \underline{U}^{+T}] \begin{bmatrix} \underline{B} \\ \hline \underline{A}^+ \end{bmatrix} \tag{G.5}$$

Forming a prediction error vector as

$$\underline{e}_0(k) = \underline{y}(k) - \hat{\underline{y}}(k) \tag{G.6}$$

and setting the model coefficients to minimize the trace of the prediction error covariance matrix results in a solution given by

$$\begin{bmatrix} \underline{R}_{YY} & \underline{R}_{YU^+} \\ \underline{R}_{U^+Y} & \underline{R}_{U^+U^+} \end{bmatrix} \begin{bmatrix} \underline{B} \\ \underline{A}^+ \end{bmatrix} = \begin{bmatrix} \underline{R}_{Yy} \\ \underline{R}_{U^+y} \end{bmatrix} \tag{G.7}$$

In the transform domain, the prediction error model is represented by

$$\underline{E}_0(z)^T = \underline{Y}(z)^T [\underline{I}-\underline{B}(z)]-\underline{U}(z)^T \underline{A}(z) \tag{G.8}$$

where $\underline{E}$, $\underline{Y}$ and $\underline{U}$ are the transforms of the error, output and input vectors and the coefficients of the i,j-th elements of the matrix polynomials $\underline{B}(z)$ and $\underline{A}(z)$ are the elements of the vectors $\underline{b}_{ij}$ and $\underline{a}^+_{ij}$. The multichannel ARMA synthesis model is then given by

$$\underline{Y}^T(z) = \underline{U}(z)^T \underline{A}(z)[\underline{I}-\underline{B}(z)]^{-1} \tag{G.9}$$

with the matrix polynomial fraction serving as the generalization of the zero pole transfer function.

APPENDIX H

Delay Free Loops

To develop equation (4.24) guaranteeing the absence of delay free loops in the system of Figure 4.8, consider the equation for $y_N(k)$.

$$\underline{y}_N(k) = \underline{F}[\underline{x}_N(k)] \tag{H.1}$$

Since $\underline{F}[.]$ is a memoryless nonlinear function, proving that $\underline{x}_N(k)$ is not a function of $\underline{y}_N(k)$ is equivalent to proving that $\underline{y}_N(k)$ is not a function of itself, and therefore no delay free loops exist. From equations (4.17b) and (4.17d) with $\underline{u}(k)=0$ it follows that

$$\underline{X}_N(z) = \underline{\Gamma}_2 \, \underline{T}(z) \, \underline{\Gamma}_1 \, \underline{Y}_N(z) \tag{H.2}$$

and the coefficient of y_N at time k on the right hand side is given by

$$\underline{\alpha} = \lim_{z \to \infty} \underline{\Gamma}_2 \, \underline{T}(z) \, \underline{\Gamma}_1 \tag{H.3}$$

A nonzero α_{ij} indicates a dependence of $x_{Ni}(k)$ upon $y_{Nj}(k)$ and clearly therefore all the main diagonal elements of $\underline{\alpha}$ must be zero to avoid delay free loops. These elements are the terms of the (Q-1)-st principal minors of $\underline{\alpha}$. While this is a necessary condition it is not sufficient to avoid delay free loops since loops may exist through two or more signals.

The condition that $\alpha_{ij}\ \alpha_{ji} = 0$ for all i,j such that $1 \leq i,j \leq Q$ and $i \neq j$, ensures that no delay free loop exist through 2 signals and is equivalent to requiring all terms of the (Q-2)-nd principal minors of $\underline{\alpha}$ are zero. A term of a determinant [Ref. 63] is defined as the product of elements of the matrix taken one from each row and one from each column and the determinent of the matrix is the sum of all possible terms. It is clear therefore that every term must contain a cycle such as $\alpha_{ij}\alpha_{jk} \ldots \alpha_{li}$ and must therefore be zero. Since the determinant consists of every possible term, requiring that all terms of the determinants of the (Q-i)-th principal minors are zero ensures that no delay free loops exist through any combination of i signals. Examining all terms of the determinant of $\underline{\alpha}$ and all its principal minors ensures that all possible loops through the Q signals in $x_N(k)$ are examined. If any delay free loop exists, then at least one of the terms of one of the determinants will be nonzero.

APPENDIX I

Nonlinear ARMA Modeling Examples

The determination of memory requirements for the nonlinear ARMA model as well as its applicability to systems consisting of interconnections of linear and memoryless nonlinear subsystems is illustrated here using two examples. First a cascade of linear and nonlinear subsystems is considered. Then a real world example is considered and a nonlinear ARMA model is proposed for the tracking behavior of a phase locked loop.

A. A CASCADE OF LINEAR AND NONLINEAR SUBSYSTEMS

Consider the system shown in Figure I.1 where the signals u(k) and $y_{LZ}(k)$ are observed. In terms of the topology of Figure 4.8, seven signals can be identified (x_{L1}, x_{L2}, y_{L1}, y_{L2}, x_{N1}, y_{N1} and u) however for convenience three of the seven equations in (4.26) which specify

$$x_{L1}(k) = u(k) \tag{I.1a}$$

$$x_{L2}(k) = y_{N1}(k) \tag{I.1b}$$

$$x_{N1}(k) = y_{L1}(k) \tag{I.1c}$$

will not be explicitly written. In this case, equation (4.26) becomes

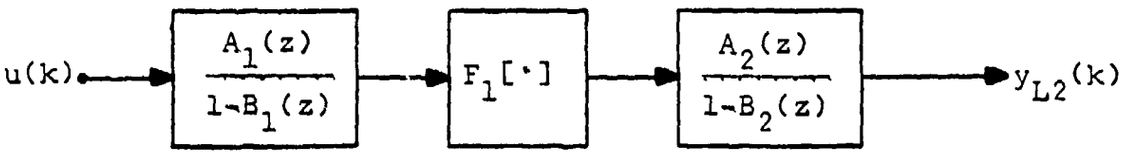


Figure I.1. A Nonlinear System

$$\begin{bmatrix} y_{L2}(k) \\ u(k) \\ \hline y_{L1}(k) \\ y_{N1}(k) \end{bmatrix} = \left[\begin{array}{cc|cc} b_2(k)^* & 0 & 0 & a_2(k)^* \\ 0 & 1 & 0 & 0 \\ \hline 0 & a_1(k)^* & b_1(k)^* & 0 \\ 0 & 0 & F_1[\cdot] & 0 \end{array}\right] \begin{bmatrix} y_{L2}(k) \\ u(k) \\ \hline y_{L1}(k) \\ y_{N1}(k) \end{bmatrix} \tag{I.2}$$

where the finite memory representations of $T_1(z)$ and $T_2(z)$ have been used. The objective now is to eliminate $a_2(k)^*$ from the upper right partition. Using the equations of rows three and four, the first row can be rewritten as

$$y_{L2}(k)=b_2(k)^*\ y_{L2}(k)+a_2(k)^*F_1[a_1(k)^*u(k)+b_1(k)^*y_{L1}(k)] \tag{I.3}$$

Notationally it is difficult to write equation (I.3) in the operator matrix form of equation (I.2) because of the nonlinear function however, the following representation is adopted.

$$\begin{bmatrix} y_{L2}(k) \\ u(k) \\ \hline y_{L1}(k) \\ y_{N1}(k) \end{bmatrix} = \left[\begin{array}{cc|cc} b_2(k)* & a_2(k)*F_1[a_1(k)*(.) & b_1(k)*(.)] & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & a_1(k)* & b_1(k)* & 0 \\ 0 & 0 & F_1[\cdot] & 0 \end{array}\right] \begin{bmatrix} y_{L2}(k) \\ u(k) \\ \hline y_{L1}(k) \\ y_{N1}(k) \end{bmatrix} \qquad (I.4)$$

No further reductions are possible to make the upper right partition a null matrix leading to the conclusion that for a finite memory nonlinear ARMA model to be appropriate, either $b_1(k)$ must be zero (the first linear system must be nonrecursive) or $y_{L1}(k)$ must also be observed. Alternately, if $b_1(k)$ is nonzero, an infinite memory representation can be used for the first linear system by replacing $a_1(k)*(.)$ with $h_1(k)*(.)$ and $b_1(k)*(.)$ with zero in equation (I.4) indicating that an infinite memory nonlinear ARMA model is appropriate when only $u(k)$ and $y_{L2}(k)$ are observed.

B. A NONLINEAR ARMA MODEL FOR A PHASE LOCKED LOOP

A continuous time model for the tracking behavior of a phase locked loop [Ref. 55] is shown in Figure I.2 where:

$\theta_1(t)$ is the phase of the incoming signal

$\theta_2(t)$ is the estimate of the phase of the incoming signal

$e(t)$ is the phase error signal

$F(s)$ is the transfer function of the loop filter

K_1 and K_2 are constants

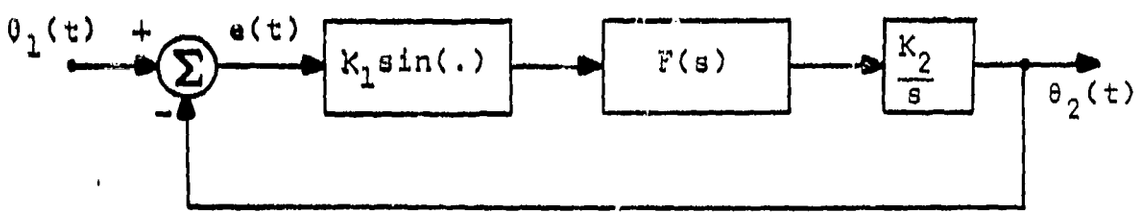


Figure I.2. A nonlinear model for the tracking behavior of a phase-locked loop.

The model is nonlinear because of the sin function in the loop. Often the assumption is made that $e(t) << \pi/2$ so that $\sin e(t) \approx e(t)$ in which case a linearized model is obtained as

$$\frac{\theta_2(s)}{\theta_1(s)} = \frac{K_1 K_2 F(s)}{K_1 K_2 F(s)+s} \tag{I.5}$$

A nonlinear ARMA model for the system can be obtained by first discretizing the model of Figure I.2 as shown in Figure I.3 where F(z) represents the discrete loop filter.

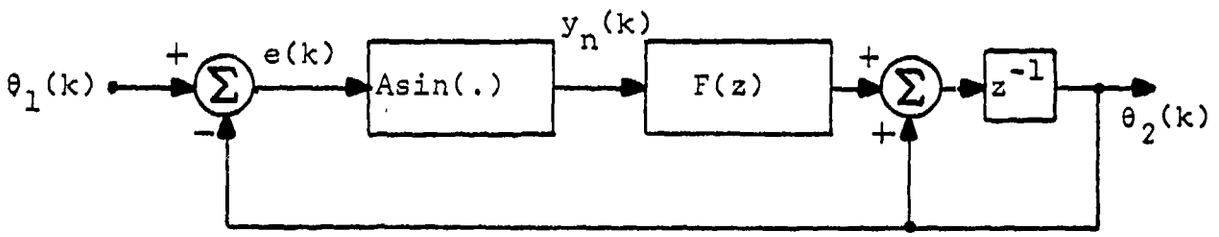


Figure I.3. A discrete version of the phase-locked loop model.

The integration has been approximated as $\frac{z^{-1}}{1-z^{-1}}$. (It is necessary to use this Euler forward approximation for integration rather than one such as the trapezoid rule to avoid a delay free loop). Defining a single linear system as the cascade of F(z) and the discrete integration

$$K_1 K_2 \frac{z^{-1}}{1-z^{-1}} F(z) = \frac{A(z)}{1-B(z)} \tag{I.6}$$

Equation (4.26) for the phase locked loop becomes

$$\begin{bmatrix} \theta_2(k) \\ \theta_1(k) \\ \hline y_N(k) \\ e(k) \end{bmatrix} = \left[\begin{array}{cc|cc} b_1(k)^* & 0 & a_1(k)^* & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & \sin(.) \\ -1 & 1 & 0 & 0 \end{array}\right] \begin{bmatrix} \theta_2(k) \\ \theta_1(k) \\ \hline y_N(k) \\ e(k) \end{bmatrix} \quad \text{(I.7)}$$

where it is assumed that $\theta_1(k)$ and $\theta_2(k)$ are observed. Using the relationships specified by rows 3 and 4, this can be written as

$$\begin{bmatrix} \theta_2(k) \\ \theta_1(k) \\ \hline y_N(k) \\ e(k) \end{bmatrix} = \left[\begin{array}{cc|cc} b_1(k)^* & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & \sin(.) \\ -1 & 1 & 0 & 0 \end{array}\right] \begin{bmatrix} \theta_2(k) \\ \theta_1(k) \\ \hline y_N(k) \\ e(k) \end{bmatrix}$$

$$+ \left[\begin{array}{ccc|cc} a_1(k)^* & \sin[-(.) & (.)] & 0 & 0 \\ \hline 0 & & 0 & 0 & 0 \\ 0 & & 0 & 0 & 0 \\ 0 & & 0 & 0 & \end{array}\right] \begin{bmatrix} \theta_2(k) \\ \theta_1(k) \\ \hline y_N(k) \\ e(k) \end{bmatrix} \quad \text{(I.8)}$$

from which it is clear that a finite memory nonlinear ARMA model is appropriate. The model can be obtained from the first row in equation I.8 by substituting a series expansion

for the sin function and truncating it at the degree desired resulting in

$$\theta_2(k)=b_1(k)*\theta_2(k) + a_1(k)* \sum_{m=0}^{M} \frac{(-1)^m}{(2m+1)!}(\theta_2(k)-\theta_1(k))^{2m+1} \tag{I.9}$$

where 2M+1 is the degree of the series approximation to sin(.). (Note that $\lim_{z\to\infty} A_1(z) = \lim_{z\to\infty} B_1(z) = 0$ so that the right hand side of equation (I.9) involves only past values of the output $\theta_2(k)$.) An infinite memory Volterra series model for this system has been considered by Van Trees [Ref. 55].

APPENDIX J

Model Simulation Program Listings

This appendix provides a listing of the fortran model simulation programs used in the experimental study of the lattice characteristics. Included are the main programs for the batch processing ARMA lattice, the adaptive ARMA lattice and the brute force model solution. Each main program is followed by a collection of subroutines used only by that specific main program. Then a collection of common subroutines called from two or more locations in the batch lattice, adaptive lattice or brute force method is listed.

```
C*****************************************************************
C*   THIS IS THE MAIN PROGRAM FOR THE BATCH PROCESSING ARMA LATTICE.  *
C*   TOGETHER WITH THE APPROPRIATE SUBROUTINES, IT FINDS  ALL ARMA    *
C*   MODELS FROM ORDER 1 TO THE SYSTEM ORDER PLUS 2 USING FIRST 200   *
C*   POINT AVERAGES THEN 4000 POINT AVERAGES. RESULTS ARE OUTPUT ON THE *
C*   SYSTEM PRINTER AND A FILE CONTAINING MODEL COEFFICIENTS AND ROOTS  *
C*   IS CREATED FOR LATER PLOTTING ON THE VERSATEC PLOTTER.           *
C*****************************************************************
C
      DIMENSION U(5000),Y(5000),PA(10),PB(10),BMA(10),BME(10)
C
C   GENERATE THE SYSTEM INPUT SIGNAL
C
      WRITE (6,1000)
      CALL SNORM(28,0,5000)
C
C   GIVEN THE DATA IN SUBROUTINE PCOEF, SET UP THE PLANT
C
      CALL PCOEF(PA,PB,PG,IFD,N)
C
C   RUN THE PLANT AND GENERATE THE OUTPUT SIGNAL
C
      CALL PRUN(PA,PB,PG,IFD,U,Y,N,IFL,5000)
C
C   FIND THE LATTICE MODELS WITH 200 AND 4000 POINT AVERAGES
C
      DO 1200 IR=1,2
      IF(IR.EQ.1) NP=200
      IF(IR.EQ.2) NP=4000
      MAX=10
      MORD=MINO(MAX,N+2)
      CALL LATFM(L,Y,PA,PB,PG,MORD,NP)
1200  CONTINUE
1000  FORMAT('1')
100   STOP
      END
C
C*****************************************************************
C*   THIS SUBROUTINE CALCULATES ALL LATTICE ARMA MODELS FOR THE SYSTEM  *
C*   FROM FIRST ORDER TO ORDER MORD USING NP POINT AVERAGES WHEN GIVEN  *
C*   SYSTEM INPUT AND OUTPUT DATA.                                    *
C*****************************************************************
C
      SUBROUTINE LATFM(L,Y,PLTA,PLTB,PLTG,MORD,NP)
      DIMENSION U(1),Y(1),PLTA(1),PLTB(1)
      DIMENSION A(10),B(10),EYF(5000,2),EUF(5000,2),EYB(5000,2)
      DIMENSION ELB(5000,2),PF(2,2),PB(2,2),DEL(2,2)
      DIMENSION FK(2,2,10),BK(2,2,10),TFK(2,2),TBK(2,2)
```

```
      DIMENSION WK(10)
      DIMENSION ALF(10,2),AYF(10,2),BLF(10,2),BYF(10,2)
      DIMENSION ALB(10,2),AYB(10,2),BUB(10,2),BYB(10,2)
      WRITE (6,1310)
      WRITE (6,1300)
      WRITE (6,1330) NP
      WRITE (6,1320)
      INTVST=500
      DO 2 II=1,10
      A(II)=0.0
2     B(II)=0.0
      IDET=0
      DO 100 I=1,NORD
      IF(I.NE.1) GO TO 20
C
C   TIME AVERAGE TO FIND THE PREDICTION ERROR COVARIANCE MATRICIES
C   AT THE ZEROTH ORDER STAGE.
C
      DO 10 J=1,5000
      EYF(J,1)=Y(J)
      EYB(J,1)=Y(J)
      ELF(J,1)=L(J)
      ELB(J,1)=U(J)
10    CONTINUE
      PF(1,1)=0.0
      PF(1,2)=0.0
      PF(2,2)=0.0
      NPP=INTVST+NP
      DO 15 J=INTVST,NPP
      PF(1,1)=PF(1,1)+Y(J)**2
      PF(1,2)=PF(1,2)+Y(J)*L(J)
      PF(2,2)=PF(2,2)+U(J)**2
15    CONTINUE
      PF(1,1)=PF(1,1)/NP
      PF(1,2)=PF(1,2)/NP
      PF(2,2)=PF(2,2)/NP
      PF(2,1)=PF(1,2)
      PB(1,1)=PF(1,1)
      PB(1,2)=PF(1,2)
      PB(2,1)=PF(2,1)
      PB(2,2)=PF(2,2)
      IZ=0
C
C   LIST THE ZEROTH ORDER ERROR COVARIANCE MATRICIES ALONG WITH
C   EIGENVALUES, EIGENVECTORS AND ELIPSOIC AXIS HALF LENGTHS.
C
      WRITE (6,1000) IZ,PF(1,1),PF(1,2)
      WRITE (6,1010) PF(2,1),PF(2,2)
```

```
      CALL EIG(PF)
      WRITE (6,1020) IZ,PB(1,1),PB(1,2)
      WRITE (6,1010) PB(2,1),PB(2,2)
      CALL EIG(PB)
      DETF=PF(1,1)*PF(2,2)-PF(1,2)*PF(2,1)
      WRITE (6,1030) DETF,DETF
      GO TO  31
C
C   IF AT THE SECOND STAGE OR BEYOND, CALCULATE THE PREDICTION ERROR
C   SEQUENCES BY PASSING THE OLD ERROR SEQUENCES THROUGH THE LATEST
C   LATTICE STAGE.
C
20    DO 25 J=1,5000
      EYF(J,2)=EYF(J,1)
      EUF(J,2)=EUF(J,1)
      EYB(J,2)=EYB(J,1)
      EUB(J,2)=EUB(J,1)
25    CONTINUE
      DO 30 J=1,5000
      IF(J.NE.1) GO TO 27
      EYF(J,1)=EYF(J,2)
      EUF(J,1)=EUF(J,2)
      EYB(J,1)=EYB(J,2)
      EUB(J,1)=EUB(J,2)
      GO TO 30
27    EYF(J,1)=EYF(J,2)-FK(1,1,I-1)*EYB(J-1,2)-FK(2,1,I-1)*EUB(J-1,2)
      EUF(J,1)=EUF(J,2)-FK(1,2,I-1)*EYB(J-1,2)-FK(2,2,I-1)*EUB(J-1,2)
      EYB(J,1)=EYB(J-1,2)-BK(1,1,I-1)*EYF(J,2)-BK(2,1,I-1)*EUF(J,2)
      EUB(J,1)=EUB(J-1,2)-BK(1,2,I-1)*EYF(J,2)-BK(2,2,I-1)*EUF(J,2)
30    CONTINUE
C
C   TIME AVERAGE TO FIND THE DEL MATRIX.
C
31    NPP=NP+INTVST
      DEL(1,1)=0.0
      DEL(1,2)=0.0
      DEL(2,1)=0.0
      DEL(2,2)=0.0
      DO 40 J=INTVST,NPP
      DEL(1,1)=DEL(1,1)+EYF(J,1)*EYB(J-1,1)
      DEL(1,2)=DEL(1,2)+EYF(J,1)*EUB(J-1,1)
      DEL(2,1)=DEL(2,1)+EUF(J,1)*EYB(J-1,1)
      DEL(2,2)=DEL(2,2)+EUF(J,1)*EUB(J-1,1)
40    CONTINUE
      DEL(1,1)=DEL(1,1)/NP
      DEL(1,2)=DEL(1,2)/NP
      DEL(2,1)=DEL(2,1)/NP
      DEL(2,2)=DEL(2,2)/NP
```

```
      TFK(1,1)=DEL(1,1)
      TFK(1,2)=DEL(2,1)
      TFK(2,1)=DEL(1,2)
      TFK(2,2)=DEL(2,2)
      TEK(1,1)=DEL(1,1)
      TEK(1,2)=DEL(1,2)
      TEK(2,1)=DEL(2,1)
      TEK(2,2)=DEL(2,2)
C
CCC   CALCULATE THE NEXT REFLECTION COEFFICIENT MATRICIES FOR THE LATTICE.
C
      CALL SCLK(PE,FK,TFK,I)
      CALL SOLK(PF,BK,TBK,I)
C
CCC   LOCATE THE PREDICTION ERROR COVARIANCE MATRICIES USING THE OLD
CCC   COVARIANCE MATRICIES AND THE LATEST REFLECTION COEFFICIENT MATRICIES.
C
      CALL LOCCV(FF,PB,FK,EK,I)
      DETF=PF(1,1)*PF(2,2)-FF(2,1)*PF(1,2)
      DETB=PB(1,1)*PB(2,2)-PB(2,1)*PB(1,2)
C
CCC   CALCULATE THE ARMA MODEL GAIN AT THIS ORDER SOLUTION
C
      G=FF(1,2)/PF(2,2)
C
CCC   CALCULATE THE ARMA MODEL TRANSFER FUNCTION POLYNOMIALS AT THIS
CCC   ORDER SOLUTION.
C
      CALL UPPLY(A,B,FK,BK,AYF,AUF,AYB,AUB,BYF,BUF,BYB,BLB,I,G)
C
CCC   CALCULATE THE ARMA MEAN SQUARE EQUATION ERROR FOR THIS ORDER
C
      ESQ=PF(1,1)+PF(2,2)*G*G-(PF(1,2)+FF(2,1))*G
C
CCC   LIST LATEST LATTICE COEFFICIENTS, COVARIENCE MATRICIES, EIGENVALUES,
CCC   EIGENVECTORS AND AXIS HALF LENGTHS.
C
      WRITE (6,1100) I,FK(1,1),I,FK(1,2,I)
      WRITE (6,1010) FK(2,1,I),FK(2,2,I)
      WRITE (6,1100) I,EK(1,1,I),I,EK(1,2,I)
      WRITE (6,1010) BK(2,1,I),BK(2,2,I)
      WRITE (6,1000) I,FF(1,1),PF(1,2)
      WRITE (6,1010) PF(2,1),PF(2,2)
      CALL EIG(PF)
      WRITE (6,1020) I,PB(1,1),PB(1,2)
      WRITE (6,1010) PB(2,1),PB(2,2)
      CALL EIG(PB)
      WRITE (6,1030) DETF,DETB
```

```
      IF((DETF.LT.0.0).OR.(DETB.LT.0.0)) WRITE (6,1200)
      WRITE (6,1040) I,ESQ
      WRITE (6,105C)
C
CC   LIST ARMA MODEL TRANSFER FUNCTION COEFFICIENTS
C
      WRITE (6,107C) G
      DO 45 IND=1,I
45    WRITE (6,108C) IND,B(IND),IND,A(IND)
C
C   FIND AND LIST MODEL POLES AND PLOT MODEL AND SYSTEM TRANSFER
C   FUNCTION MAGNITUDES.
C
      CALL SPECTP(B,A,G,PLTB,PLTA,PLTG,MORD,I)
100   CONTINUE
100C  FORMAT('OPF(',I2,')= ',E14.5,5X,E14.5)
1010  FORMAT(' ',6X,E14.5,5X,E14.5)
102C  FORMAT('OPB(',I2,')= ',E14.5,5X,E14.5)
103C  FORMAT('DET PF = ',E14.5,5X,'DET PB = ',E14.5)
104C  FORMAT('0MSE FOR THE ',I2,' ORDER Z-P MODEL = ',E14.5)
105C  FORMAT(' COEFFICIENTS ARE:')
106C  FORMAT(' B(',I2,')=',E14.5,5X,'A(',I2,')=',E14.5)
107C  FORMAT(' ',25X,'A( 0)=',E14.5)
11CC  FORMAT('1KF(',I2,')=',E14.5,5X,'KF(',I2,')=',E14.5)
111C  FORMAT('0KB(',I2,')=',E14.5,5X,'KB(',I2,')=',E14.5)
112C  FORMAT(' ********** NEGATIVE DEFINITE COVARIANCE **********')
130C  FORMAT('0********** EXPLICIT LATTICE SOLUTIONS **********')
131C  FORMAT('1**************************************************')
132C  FORMAT('0**************************************************')
133C  FORMAT('0**********   ',I4,' POINT AVERAGES      **********')
      RETURN
      END
C
C*****************************************************************
C*   THIS SUBROUTINE CALCULATES THE REFLECTION COEFFICIENT MATRICIES    *
C*   WHEN GIVEN THE APPROPRIATE DEL AND F MATRICIES.                    *
C*****************************************************************
C
      SUBROUTINE SOLK(AF,AK,DEL,I)
      DIMENSION AF(2,2),AK(2,2,I),DEL(2,2)
      DIMENSION WK(10)
      IDGT=0
      CALL LEQT2F(AF,2,2,2,DEL,IDGT,WK,IER)
      DO 10 IR=1,2
      DO 5 IC=1,2
      AK(IR,IC,I)=DEL(IR,IC)
5     CONTINUE
10    CONTINUE
```

```
      RETURN
      END
C
C*****************************************************************************
C*   THIS IS THE MAIN ADAPTIVE ARMA LATTICE PROGRAM. TOGETHER WITH THE       *
C*   APPROPRIATE SUBROUTINES IT WILL RUN THE LATTICE 100 TIMES AND           *
C*   AVERAGE THE RESULTS OVER THE ENSEMBLE. RESULTS WILL BE PRINTED ON       *
C*   THE SYSTEM PRINTER AND TWO OUTPUT DATA FILES WILL ALSO BE CREATED       *
C*   CONTAINING MODEL COEFFICIENTS, ROOTS AND SPECIFIED TIME SEQUENCES       *
C*   FOR LATER PLOTTING ON THE VERSATEC PLOTTER.                             *
C*****************************************************************************
C
      DIMENSION X(6500,2),AER(5000,5)
      DIMENSION AFK(2,2,10,3),ABK(2,2,10,3),AAO(10,3)
      DIMENSION AEZPSQ(10,3),PA(10),PE(10),IS(100),XX(6500),AK(5000,5)
      NCH=2
      NES=100
      ENS=FLOAT(NES)
C
C   GIVEN THE DATA IN SUBROUTINE PCOEF, SET UP THE SYSTEM TO BE MODELED.
C
      CALL PCOEF(PA,PB,PG,ID,N)
      IORD=N+2
      CALL SINT(2E,IS,100)
      DO 50 I=1,6500
      IM=I-1
50    XX(I)=FLOAT(IM)
C
C   RUN THE ADAPTIVE LATTICE NES TIMES WITH INDEPENDENT INPUTS FOR
C   ENSEMBLE AVERAGING.
C
      DO 100 I=1,NES
C
C   GENERATE AN INPUT SIGNAL FOR THE I-TH RUN USING IS(I) AS A SEED.
C
      CALL SNORM(IS(I),X(1,2),6500)
C
C   RUN THE SYSTEM TO GENERATE THE OUTPUT FOR THE I-TH RUN.
C
      CALL PRUN(PA,PB,PG,ID,X(1,2),X(1,1),N,IFL,6500)
C
C   RUN THE ADAPTIVE ARMA LATTICE.
C
      CALL ADLAT(X,3000,IORD,AER,AFK,ABK,AAO,AEZPSQ,AK)
      WRITE (6,1000) I
100   CONTINUE
C
```

```
C  AVERAGE THE RESLLTS CVER THE ENSEMBLE AND OUTPLT THEM.
C
      DO 110 I=1,3000
      DO 105 INC=1,5
105   AER(I,INC)=AER(I,INC)/ENS
      DO 1C6 IRW=1,5
      IF(I.GT.3CC) GO TO 1C6
      AK(I,IRW)=AK(I,IRW)/ENS
1C6   CONTINUE
11C   CONTINUE
      DO 112 I=1,5
112   CALL FUNCK(AER(1,I))
      DO 115 I=1,3000
      DO 114 INC=1,5
114   IF(AER(I,INC).GT..1) AER(I,INC)=.1
115   CONTINUE
      DO 130 I=1,3
      DO 127 IJ=1,10
      AAC(IJ,II)=AAC(IJ,II)/ENS
      AEZPSQ(IJ,I)=AEZPSC(IJ,I)/ENS
      DO 125 IK=1,NCH
      DO 123 IL=1,NCH
      ABK(IK,IL,IJ,I)=ABK(IK,IL,IJ,I)/ENS
123   AFK(IK,IL,IJ,I)=AFK(IK,IL,IJ,I)/ENS
125   CONTINUE
127   CONTINUE
130   CONTINUE
      DO 135 I=1,3
      WRITE (6,1C1C) NES,I
135   CALL FPLT(XX,AER(1,I),3000,0)
      DO 140 I=1,5
      CALL FPLT(X),AK(1,I),3000,0)
      CALL FUNCK(AK(1,I))
140   CONTINUE
      CALL CPT2P(AFK,ABK,AAC,AEZPSQ,PA,PB,PG,IORC)
1000  FORMAT(' ',I2)
1C1C  FORMAT('1MSE ACRCSS THE ENSEMBLE CF ',I3,' RUNS AT',I2,' CRDER')
      STCP
      END
C
C*****************************************************************
C*  THIS SUBRCUTINE RUNS THE ADAPTIVE ARMA LATTICE CVER A NPT PCINT   *
C*  INTERVAL GIVEN RECORDS CF THE SYSTEM INPUT AND CUTPUT DATA.       *
C*****************************************************************
C
      SLBRCUTINE ADLAT(X,NPT,IORC,AER,AFK,ABK,AAC,AEZPSC,AK)
      DIMENSION X(6500,1),EF(2,10),EB(2,1C,2),FK(2,2,1C),BK(2,2,10)
      DIMENSICN SIGF(10),AER(5000,5),AFK(2,2,10,3),AEK(2,2,10,3)
```

```
      DIMENSION AAO(10,2),AO(10),SA(10),EZP(10),AEZPSQ(10,2),SIGB(10)
      DIMENSION AK(5000,5),GRAD(2,2,10),GRADB(2,2,10)
C
CC    INITIALIZE THE VARIABLES
C
      MAX=10
      NCH=2
      ALP=.05
      DEL=0.1
      IST=500
      ICM=ICRD-2
      DO 10 I=1,MAX
      DO 8 IJ=1,NCH
      DO 6 IK=1,NCH
      FK(IJ,IK,I)=0.0
6     BK(IJ,IK,I)=0.0
      EF(IJ,I)=0.0
      EE(IJ,I,1)=0.0
8     EE(IJ,I,2)=0.0
      SIGF(I)=DEL
      SIGB(I)=DEL
      SA(I)=0.0
      AO(I)=0.0
10    EZP(I)=0.0
C
C  RUN THE ADAPTIVE LATTICE OVER AN NPT POINT INTERVAL STARTING AT IST
C
      NPP=NPT+IST
      DO 200 K=IST,NPP
      KM=K-1
      KM1=K-IST+1
C
C    FIND THE ERROR SEQUENCES AT TIME K USING THE LATEST COEFFICIENTS
C
      DO 15 I=1,ICRD
      DO 13 J=1,NCH
13    EE(J,I,2)=EE(J,I,1)
15    CONTINUE
      DO 5C I=1,ICRD
      IM=I-1
      IF(I.EQ.1) GO TO 3C
      DO 25 N=1,NCH
      FA=0.0
      BA=0.0
      DO 20 L=1,NCH
      FA=FA+FK(L,N,I)*EE(L,IM,2)
2C    BA=BA+BK(L,N,I)*EF(L,IM)
      EF(N,I)=EF(N,IM)-FA
```

```
      EE(N,I,1)=EE(N,IM,2)-BA
      GO TO 50
   30 DO 40 N=1,NCH
      FA=0.0
      BA=0.0
      DO 35 L=1,NCH
      IF(K.EQ.IST) GO TO 35
      FA=FA+FK(L,N,1)*X(KM,L)
   35 BA=BA+BK(L,N,1)*X(K,L)
      EF(N,1)=X(K,N)-FA
      EE(N,1,1)=-BA
   40 IF(K.NE.IST) EB(N,1,1)=X(KM,N)+EB(N,1,1)
   50 EZF(I)=EF(2,I)-AC(I)*EF(2,I)
C
C  UPDATE THE STEP SIZE NORMALIZING FACTORS AT EACH STAGE
C
      DO 70 I=1,ICRD
      IM=I-1
      WSF=0.0
      WSB=0.0
      SA(I)=(1.0-ALP)*SA(I)+ALP*(EF(2,I)**2)
      IF(I.EQ.1) GO TO 60
      DO 55 J=1,NCH
      WSB=WSB+EF(J,IM)**2
   55 WSF=WSF+EB(J,IM,2)**2
      GO TO 67
   60 DO 65 J=1,NCH
      WSB=WSB+X(K,J)**2
      IF(K.EQ.IST) GO TO 65
      WSF=WSF+X(KM,J)**2
   65 CONTINUE
   67 SIGF(I)=(1.0-ALP)*SIGF(I)+ALP*WSF
   70 SIGB(I)=(1.0-ALP)*SIGB(I)+ALP*WSB
      DO 71 I=1,ICRD
      SIGF(I)=(SIGF(I)+SIGB(I))/2.0
   71 SIGB(I)=SIGF(I)
C
C  UPDATE THE LATTICE COEFFICIENTS
C
      DO 100 I=1,ICRD
      IM=I-1
      AC(I)=AC(I)+(ALP/SA(I))*EF(2,I)*EZF(I)
      IF(I.EQ.1)GO TO 85
      DO 80 IJ=1,NCH
      DO 75 IK=1,NCH
      EFAC(IJ,IK,I)=EB(IJ,IM,2)*EF(IK,I)
      EFZEE(IJ,IK,I)=EF(IJ,IM)*EB(IK,I,1)
   75 FK(IJ,IK,I)=FK(IJ,IK,I)+(ALP/SIGF(I))*EE(IJ,IM,2)*EF(IK,I)
```

```
      EK(IJ,IK,I)=EK(IJ,IK,I)+(ALP/SIGE(I))*EF(IJ,IP)*EB(IK,I,I)
   60 CONTINUE
      GO TO 100
   65 DO 95 IJ=1,NCH
      DO 90 IK=1,NCH
      IF(K.EQ.IST) GO TO 90
      CRAD(IJ,IK,I)=X(KP,IJ)*EF(IK,I)
      ERADS(IJ,IK,I)=X(K,IJ)*EB(IK,I,I)
      EK(IJ,IK,I)=EK(IJ,IK,I)+(ALP/SIGE(I))*X(KP,IJ)*EF(IK,I)
   90 EK(IJ,IK,I)=EK(IJ,IK,I)+(ALP/SIGE(I))*X(K,IJ)*EB(IK,I,I)
   95 CONTINUE
  100 CONTINUE
C
C  AVERAGE THE INSTANTANEOUS ZP MSE AC   - THE ENSEMBLE FOR THE CORRECT
C  ORDER AND AN OVERMODEL OF TWO ORDERS
C
      IF(KMI.GT.5000) GO TO 104
      DO 103 INDX=1,5
      AER(KMI,INDX)=AER(KMI,INDX)+EZP(INDX)**2
  103 AK(KMI,INDX)=AK(KMI,INDX)+BK(I,I,INDX)
  104 CONTINUE
C
C  IF AT THE 1000,2000 OR 3000 TIME POINTS SINCE ADAPTATION STARTED,
C  ENSEMBLE AVERAGE THE COEFFICIENTS AND THE INSTANTANEOUS ZP MSE
C  ESTIMATES AT EACH ORDER FROM 1 TO IORD
C
      IS1=IST+1000
      IS2=IST+2000
      IS3=IST+3000
      IF(K.EQ.IS1) GO TO 105
      IF(K.EQ.IS2) GO TO 110
      IF(K.EQ.IS3) GO TO 115
      GO TO 200
  105 L=1
      GO TO 120
  110 L=2
      GO TO 120
  115 L=3
  120 DO 135 I=1,IORD
      DO 130 IJ=1,NCH
      DO 125 IK=1,NCH
      AEK(IJ,IK,I,L)=AEK(IJ,IK,I,L)+EK(IJ,IK,I)
  125 AEK(IJ,IK,I,L)=AEK(IJ,IK,I,L)+EK(IJ,IK,I)
  130 CONTINUE
      AEZPSQ(I,L)=AEZPSQ(I,L)+EZP(I)**2
  135 AAC(I,L)=AAC(I,L)+AO(I)
  200 CONTINUE
      RETURN
```

```
      END
C
C*****************************************************************************
C*   THIS SUBROUTINE IS CALLED BY THE MAIN ADAPTIVE ARMA LATTICE             *
C*   PROGRAM AND CALCULATES TRANSFER FUNCTIONS FROM REFLECTION COEFFICI-*
C*   ENT MATRICIES AND PRODUCES THE OUTPUT ON THE SYSTEM PRINTER.            *
C*****************************************************************************
C
      SUBROUTINE OPTZP(AFK,ABK,AAO,AEZPSO,PA,PB,PG,ICRO)
      DIMENSION AFK(2,2,10,3),ABK(2,2,10,3),AAO(10,3),AEZPSO(10,3)
      DIMENSION PA(10),PB(10),FK(2,2,10),BK(2,2,10),A(10),B(10)
      DIMENSION AYF(10,2),AUF(10,2),BYF(10,2),BUF(10,2)
      DIMENSION AYB(10,2),AUB(10,2),BYB(10,2),BUB(10,2)
      ICP=ICRO-2
      DO 200 I=1,3
      DO 5 IJ=1,10
      A(IJ)=0.0
5     B(IJ)=0.0
      IF(I.EQ.1) NPA=1000
      IF(I.EQ.2) NPA=2000
      IF(I.EQ.3) NPA=3000
      WRITE (6,1000) NPA
      DO 10 IJ=1,2
      DO 9 IK=1,2
      DO 8 IL=1,10
      FK(IJ,IK,IL)=AFK(IJ,IK,IL,I)
8     BK(IJ,IK,IL)=ABK(IJ,IK,IL,I)
9     CONTINUE
10    CONTINUE
      DO 20 IC=1,ICRO
      G=AAO(IC,I)
      CALL UDPLY(A,B,FK,BK,AYF,AUF,AYB,AUB,BYF,BUF,BYB,BUB,IC,G)
      WRITE (6,1100) IC,FK(1,1,IC),FK(1,2,IC)
      WRITE (6,1110) FK(2,1,IC),FK(2,2,IC)
      WRITE (6,1120) IC,BK(1,1,IC),BK(1,2,IC)
      WRITE (6,1110) BK(2,1,IC),BK(2,2,IC)
      WRITE (6,1130) IC,AEZPSO(IC,I)
      WRITE (6,1140) AAO(IC,I)
      DO 12 IJ=1,IC
12    WRITE (6,1150) IJ,B(IJ),IJ,A(IJ)
      CALL SPECTP(B,A,AAO(IC,I),PB,PA,PG,ICRO,IC)
20    CONTINUE
200   CONTINUE
1000  FORMAT('1',15,' POINTS AFTER ADAPTATION BEGAN')
1100  FORMAT('0KF(',I2,')=',E14.5,5X,E14.5)
1110  FORMAT(' ',7X,E14.5,5X,E14.5)
1120  FORMAT('0KB(',I2,')=',E14.5,5X,E14.5)
1130  FORMAT('0INST. ESTIMATE OF MSE AT THE',I3,' STAGE=',E14.5)
```

```
114C  FORMAT('0',25X,'A( 0)=',E14.5)
115C  FORMAT(' B(',I2,')=',E14.5,5X,'A(',I2,')=',E14.5)
      RETURN
      END
C
C*****************************************************************
C*  THIS SUBROUTINE OUTPUTS DATA SEQUENCES TO A FILE FOR LATER   *
C*  PLOTTING.                                                    *
C*****************************************************************
C
      SUBROUTINE FUNOK(DAT)
      DIMENSION DAT(1)
      DO 10 I=1,3000,6
      I1=I+1
      I2=I+2
      I3=I+3
      I4=I+4
      I5=I+5
      WRITE (7,1000) DAT(I),DAT(I1),DAT(I2),DAT(I3),DAT(I4),DAT(I5)
10    CONTINUE
1000  FORMAT(6(E10.4,1X))
2000  RETURN
      END
C
C*****************************************************************
C*  THIS IS THE MAIN PROGRAM FOR THE BRUTE FORCE MODELING PROCEDURE  *
C*  USING WINDOWED DATA TO ESTIMATE CORRELATION FUNCTIONS. TOGETHER  *
C*  WITH THE APPROPRIATE SUBROUTINES, IT CALCULATES ALL ARMA MODELS  *
C*  OF ORDER ONE TO THE SYSTEM ORDER PLUS TWO USING FIRST 200 POINT  *
C*  AVERAGES THEN 4000 POINT AVERAGES. RESULTS ARE OUTPUT ON THE     *
C*  SYSTEM PRINTER AND A FILE OF MODEL COEFFICIENTS AND ROOTS IS ALSO *
C*  CREATED FOR LATER PLOTTING ON THE VERSATEC PLOTTER.              *
C*****************************************************************
C
      DIMENSION U(5000),Y(5000),PA(10),PE(10),BMA(10),BME(10)
      DIMENSION RLU(20),RYY(20),RYUM(20),RYUP(20)
C
C  GENERATE THE SYSTEM INPUT SIGNAL
C
      WRITE (6,1000)
      CALL SNORM(28,U,5000)
C
C  GIVEN THE DATA IN SUBROUTINE POLEF, SET UP THE SYSTEM
C
      CALL POLEF(PA,PB,PG,IPO,N)
C
C  RUN THE SYSTEM AND GENERATE THE OUTPUT SIGNAL
C
```

```
      CALL FRLN(PA,PB,PC,IFC,U,Y,N,IFL,SCOO)
C
C   CETAIN THE BRUTE FORCE MODELS FOR 2CC AND 400C POINT AVERAGES
C
      DO 1200 IR=1,2
      IF(IR.EQ.1) NP=20C
      IF(IR.EQ.2) NP=40CC
C
C   ESTIMATE THE CORRELATION FUNCTIONS USING WINDOWED DATA
C
      WRITE (6,10CO)
      CALL CORR(U,Y,RUU,RYY,RYUM,RYUP,RYC,RUC,RYUC,NP)
C
C   FIND THE ARMA MODELS WITH BRUTE FORCE MATRIX INVERSION
C
      MAX=10
      MORD=MIN0(MAX,N+2)
      DO 10 I=1,MORD
10    CALL BFMDL(RUU,RYY,RYUM,RYUP,RYC,RLO,RYUO,I,PB,PA,PC,MORD,Y,U,NP)
1200  CONTINUE
1000  FORMAT('1')
100   STOP
      END
C
C*****************************************************************
C*   THIS SUBROUTINE ESTIMATES THE CORRELATIONS NEEDED IN THE BRUTE  *
C*   FORCE MODELING PROCEDURE BY TIME AVERAGING. A RECTANGULAR WINDOW *
C*   IS USED ON THE DATA. THE ESTIMATES ARE ALSO PLOTTED AND LISTED ON *
C*   THE SYSTEM PRINTER.                                           *
C*****************************************************************
C
      SUBROUTINE CORR(U,Y,RUU,RYY,RYUM,RYUP,RYC,RUC,RYUC,NP)
      DIMENSION U(1),Y(1),RUU(1),RYY(1),RYUM(1),RYUP(1)
      DIMENSION XX(41),YY(41)
      INTVST=500
      WRITE (6,1020) NP
      DO 10 I=1,20
      RYY(I)=0.0
      RUU(I)=0.0
      RYUM(I)=0.0
10    RYUP(I)=0.0
      RYC=0.0
      RUC=0.0
      RYUC=0.0
      DO 50 N=1,20
      NPP=NP-N+INTVST
      DO 20 I=INTVST,NPP
      RYY(N)=RYY(N)+Y(I)*Y(I+N)
```

```
      RLL(N)=RLL(N)+U(I)*U(I+N)
      RYLP(N)=RYUF(N)+Y(I)*U(I+N)
      RYLP(N)=RYLP(N)+U(I)*Y(I+N)
   2C CONTINUE
      RYY(N)=RYY(N)/(NP-N)
      RLL(N)=RUU(N)/(NP-N)
      RYUP(N)=RYUP(N)/(NP-N)
   5C RYLP(N)=RYUF(N)/(NF-N)
      NFF=NP+INTVST
      DO 6C I=INTVST,NFF
      RYC=RYO+Y(I)*Y(I)
      RLC=RUC+U(I)*U(I)
   6C RYLO=RYLO+Y(I)*U(I)
      RYC=RYO/NF
      RLC=RUO/NF
      RYLC=RYLC/NF
      DO 70 I=1,41
      XX(I)=I-21
      YY(I)=RYO
   7C IF(I.NE.21) YY(I)=RYY(IABS(I-21))
      WRITE (6,100)
      CALL FPLT(XX,YY,41,0)
      DO 8C I=1,41
      YY(I)=RLO
   8C IF(I.NE.21) YY(I)=RUU(IABS(I-21))
      WRITE (6,101)
      CALL FPLT(XX,YY,41,0)
      DO 90 I=1,41
      YY(I)=RYUC
      IF(I.LT.21) YY(I)=RYUF(21-I)
   9C IF(I.GT.21) YY(I)=RYUF(I-21)
      WRITE (6,102)
      CALL FPLT(XX,YY,41,0)
      WRITE (6,104) RYC,RUO,RYLO
 104C FORMAT('0RYY(0)=',E14.5,' RUU(0)=',E14.5,' RYL(0)=',E14.5)
      DO 100 I=1,20
      WRITE (6,105) I,RYY(I),RUU(I),RYLP(I),RYUP(I)
  10C CONTINUE
  105C FORMAT(' I=',I2,' RYY RUU RYUP RYLP =',4(E14.5,2X))
  100C FORMAT('1 THE ACF OF THE OUTPUT Y IS')
  101C FORMAT(' THE ACF OF THE INPUT U IS')
  102C FORMAT(' THE XCF OF THE SIGNALS Y AND U IS')
  103C FORMAT('0',I5,' POINTS WERE AVERAGED TO ESTIMATE CORRELATIONS')
      RETURN
      END
C
```

```
C*****************************************************************
C*  THIS SUBROUTINE USES A ERLTE FORCE MATRIX INVERSION TO FIND THE     *
C*  APPL MODEL OF ORDER NC.                                            *
C*****************************************************************
C
      SUBROUTINE EFMDL(FLL,FYY,RYUM,RYLF,RYC,RUC,FYLC,N,FE,FA,FG,NC,Y,U,
     -NF)
      DIMENSION FLL(1),FYY(1),RYUM(1),RYLP(1)
      DIMENSION A(441),B(21),WKA(56)
      DIMENSION FE(10),FF(10)
      DIMENSION FA(1),FG(1)
      DIMENSION Y(1),U(1)
      ICGT=0
      DO 10 I=1,10
      FE(I)=0.0
10    FF(I)=0.0
      NCR=2*N+1
      DO 60 IR=1,N
      DO 50 IC=1,NCR
      IF(IC.GT.N) GO TO 30
      A(IR+(IC-1)*(2*N+1))=RYO
      IF(IC.NE.IR) A(IR+(IC-1)*NOR)=RYY(IABS(IC-IF))
      GO TO 50
30    A(IR+(IC-1)*NOR)=FYUO
      IF(IR.EQ.(IC-N-1)) GO TO 50
      I=IR-IC+N+1
      IF(I.GT.0) A(IR+(IC-1)*NCR)=RYUF(I)
      IF(I.LT.0) A(IR+(IC-1)*NCR)=RYLP(-I+1)
50    CONTINUE
60    F(IR)=RYY(IF)
      NF=N+1
      DO 100 IR=1,NP
      DO 95 IC=1,NCR
      IF(IC.GT.N) GO TO 80
      I=IC-IR+1
      A(IR+N+(IC-1)*NCR)=RYUO
      IF(I.GT.0) A(IR+N+(IC-1)*NOR)=RYLF(I)
      IF(I.LT.0) A(IR+N+(IC-1)*NOR)=RYLP(-I)
      GO TO 95
80    I=IR+N-IC
      A(IR+N+(IC-1)*NCR)=RUO
      IF(I.NE.0) A(IR+N+(IC-1)*NCR)=FUL(IABS(I))
95    CONTINUE
      F(IR+N)=RYUC
100   IF(IR.NE.1) B(IR+N)=RYUM(IR-1)
      CALL LEQT2F(A,1,NCF,NCR,B,ICGT,WKA,IER)
      WRITE (6,1000) N
      WRITE (6,1010) B(NF)
```

```
      DO 110 I=1,N
      IP=I+NP
      EE(I)=B(I)
      EA(I)=B(IP)
110   WRITE (6,1020) I,E(IP),I,B(I)
      CALL ESQ(E(NP),EA,EB,Y,U,N,NP)
      EESQ=RYQ-E(NP)*RYQ
      DO 120 I=1,N
120   EESQ=EESQ-EE(I)*RYY(I)-EA(I)*RYUP(I)
      WRITE (6,1110) EESQ
      CALL SPECTP(EB,EA,B(NP),PB,PA,PG,NC,N)
      ACR=ACR*ACR
1030  FORMAT(' R MATRIX BY DOLS WAS:')
1040  FORMAT(' R(',I3,')=',E14.5)
1050  FORMAT('0ERLTE FORCE INVERSION SOLUTION OF ORDER ',I2,' IS')
1010  FORMAT(' AC=',E14.5)
1020  FORMAT(' A',I2,'=',E14.5,5X,'B',I2,'=',E14.5)
1100  FORMAT(' MADE IT; EFNCL',I3)
1110  FORMAT(' MS EQUATION ERROR = ',E14.5)
      RETURN
      END
C
C*****************************************************************
C*  THIS SUBROUTINE ESTIMATES THE MEAN SQUARE VALUE OF OUTPUT ERROR  *
C*  FOR THE ARMA MODEL.                                              *
C*****************************************************************
C
      SUBROUTINE ESQ(AO,A,B,Y,U,N,NP)
      DIMENSION A(1),B(1),RYY(1),RYUP(1)
      DIMENSION Y(1),U(1),YH(5000),BB(10)
      INTVST=500
      DO 10 I=1,N
10    BB(I)=-B(I)
      CALL PRUN(A,BB,AO,C,U,YH,N,IFL,5000)
      IF(IFL.EQ.1) GO TO 70
      NFF=NP+INTVST
      ERR=0.0
      DO 20 I=INTVST,NFF
20    ERR=ERR+(Y(I)-YH(I))**2
      ERR=ERR/NF
      WRITE (6,1000) N,ERR
      GO TO 80
70    WRITE (6,1010) N
80    CONTINUE
1000  FORMAT('0MS OUTPUT ERROR FOR THE ',I2,' ORDER Z-P MODEL =',E14.5)
1010  FORMAT('0MS OUTPUT ERROR FOR ',I2,' ORDER NOT CALC; MODEL UNSTAB')
      RETURN
      END
```

```
C*****************************************************************
C*   THE FOLLOWING SUBROUTINES ARE COMMON ROUTINES CALLED FROM TWO OR   *
C*   MORE PLACES IN THE BATCH LATTICE, ADAPTIVE LATTICE OR BRUTE FORCE  *
C*   MODELING PROGRAMS.                                                 *
C*****************************************************************
C
C
C*****************************************************************
C*   THIS SUBROUTINE RUNS THE SYSTEM WHEN GIVEN ITS TRANSFER FUNCTION   *
C*   COEFFICIENTS AND THE INPUT SIGNAL AND PRODUCES THE SYSTEM OUTPUT    *
C*   SIGNAL.                                                            *
C*****************************************************************
C
      SUBROUTINE FRUN(A,B,G,ID,U,Y,N,IFL,NP)
      DIMENSION A(1),B(1),U(1),Y(1),X(10)
      DO 10 I=1,N
      X(I)=0.0
10    Y(I)=0.0
      JJ=NP-ID
      IFL=0
      DO 50 I=1,JJ
      IF(IFL.EQ.1) GO TO 50
      SMR=0.0
      SMN=0.0
      DO 20 II=1,N
      SMR=SMR-B(II)*X(II)
20    SMN=SMN+A(II)*X(II)
      SMR=SMR+G*U(I)
      Y(I+ID)=SMR+SMN
      IF(ABS(Y(I+ID)).GT.10000000000.0) IFL=1
      NM=N-1
      DO 30 II=1,NM
30    X(N+1-II)=X(N-II)
      X(1)=SMR
50    CONTINUE
      RETURN
      END
C
C*****************************************************************
C*   THIS SUBROUTINE CALCULATES POLYNOMIAL COEFFICIENTS WHEN GIVEN THE  *
C*   ROOTS.                                                             *
C*****************************************************************
C
      SUBROUTINE MAKPOL(N,R,C)
      COMPLEX*16 R(N),C(N)
      IF(N.LE.0) RETURN
      DO 10 I=1,N
```

```
      C(I)=R(I)
      K=A
      L=A-1
      DO 30 I=1,M
      DO 30 I=2,K
30    C(I)=C(I)+C(I-1)
      K=K-1
      DO 20 I=1,K
      J=I+L
      C(J)=R(J)+C(I)
20    CONTINUE
      K=A/2
      K=2*K/(A-K)
      DO 40 I=K,A,2
40    C(I)=-C(I)
      RETURN
      END
C
C*****************************************************************
C*  THIS SUBROUTINE UPDATES THE THC CHANNEL AR TRANSFER FUNCTION FROM *
C*  ONE STAGE OF THE LATTICE TO THE NEXT GIVEN THE OLD TRANSFER      *
C*  FUNCTION AND THE LATEST LATTICE REFLECTION COEFFICIENT MATRICIES. *
C*****************************************************************
C
      SUBROUTINE UPDFLY(A,B,FK,BK,AYF,ALF,AYB,AUB,EYF,BLF,EYB,EUB,I,C)
      DIMENSION FK(2,2,1),BK(2,2,1),AYF(10,1),ALF(10,1),AYB(10,1)
      DIMENSION ALB(10,1),EYF(10,1),BUF(10,1),BYB(10,1),ELB(10,1)
      DIMENSION A(1),B(1)
      AYF(I,2)=FK(1,1,I)
      BLF(I,2)=FK(1,2,I)
      AYF(I,2)=FK(2,1,I)
      ALF(I,2)=FK(2,2,I)
      BYB(I,2)=BK(1,1,I)
      BUB(I,2)=BK(1,2,I)
      AYB(I,2)=BK(2,1,I)
      ALB(I,2)=BK(2,2,I)
      IF(I.EQ.1) GO TO 25
      IA=I-1
      DO 24 J=1,IA
      BYF(J,2)=BYF(J,1)
      BLF(J,2)=BUF(J,1)
      AYF(J,2)=AYF(J,1)
      ALF(J,2)=ALF(J,1)
      BYB(J,2)=BYB(J,1)
      BUB(J,2)=BUB(J,1)
      AYB(J,2)=AYB(J,1)
      ALB(J,2)=AUB(J,1)
24    CONTINUE
      DO 25 J=1,IA
```

```
      K=J-1
      BYF(J,2)=BYF(J,2)-BYB(K,1)*FK(1,1,1)-BUB(K,1)*FK(2,1,1)
      BUF(J,2)=BUF(J,2)-BYB(K,1)*FK(1,2,1)-BUB(K,1)*FK(2,2,1)
      AYF(J,2)=AYF(J,2)-AYB(K,1)*FK(1,1,1)-AUB(K,1)*FK(2,1,1)
      AUF(J,2)=AUF(J,2)-AYB(K,1)*FK(1,2,1)-AUB(K,1)*FK(2,2,1)
      BYB(J,2)=BYB(J,2)-BYF(K,1)*BK(1,1,1)-BUF(K,1)*BK(2,1,1)
      BUB(J,2)=BUB(J,2)-BYF(K,1)*BK(1,2,1)-BUF(K,1)*BK(2,2,1)
      AYB(J,2)=AYB(J,2)-AYF(K,1)*BK(1,1,1)-AUF(K,1)*BK(2,1,1)
      AUB(J,2)=AUB(J,2)-AYF(K,1)*BK(1,2,1)-AUF(K,1)*BK(2,2,1)
   25 CONTINUE
C     DO 35 J=1,I
      BYF(J,1)=BYF(J,2)
      BUF(J,1)=BUF(J,2)
      AYF(J,1)=AYF(J,2)
      AUF(J,1)=AUF(J,2)
      BYB(J,1)=BYB(J,2)
      BUB(J,1)=BUB(J,2)
      AYB(J,1)=AYB(J,2)
      AUB(J,1)=AUB(J,2)
      B(J)=BYF(J,1)-G*BUF(J,1)
      A(J)=AYF(J,1)-G*AUF(J,1)
   35 CONTINUE
      RETURN
      END
C*****************************************************************
C*  THIS SUBROUTINE PLOTS THE MAGNITUDE SPECTRA OF THE SYSTEM AND *
C*  MODEL AND ALSO CALLS ROUTINES TO CALCULATE THE MODEL ROOTS AND*
C*  PREDICTION ERROR OR COVARIANCE MATRIX EIGENVALUES, EIGENVECTORS AND*
C*  ELLIPSOIDAL AXIS HALF LENGTHS. THIS DATA IS LISTED ON THE SYSTEM *
C*  PRINTER AND THE MODEL COEFFICIENTS AND ROOTS ARE ALSO WRITTEN TO *
C*  A FILE FOR LATER PLOTTING ON THE VERSATEC PLOTTER.               *
C*****************************************************************
C
      SUBROUTINE SPECTPLT(E,A,AO,PB,PA,PG,NORD,ICR)
      DIMENSION A(1),B(1),PA(1),PB(1)
      DIMENSION X(201),Y1(201),Y2(201)
      IPFL=0
      ICP=NORD-2
      IF((ICR.EQ.ICP).OR.(ICR.EQ.NORD)) IPFL=1
      IF((ICR.EQ.2).OR.(ICR.EQ.3)) IPFL=1
      IF(ICR.EQ.6) IPFL=1
      IF(IPFL.EQ.0) GO TO 11
      WRITE (2,1110) ICR
      WRITE (2,1120) AO
      DO 10 I=1,ICR
   10 WRITE (2,1130) A(I),B(I)
   11 PI=3.14159
```

```
      Y1MAX=0.0
      Y2MAX=0.0
      DO  50  I=1,201
       AIM=FLOAT(I-1)
       X(I)=(FI/200.0)*AIM
       RNP=PG
       RNM=AG
       AINP=0.0
       AINM=0.0
       RCP=1.0
       RCM=1.0
       AICP=0.0
       AICM=0.0
       DO  20  K=1,NCRD
        TH=K*X(I)
        RNP=RNP+PA(K)*PG*COS(TH)
        RNM=RNM+A(K)*COS(TH)
        AINP=AINP-PA(K)*PG*SIN(TH)
        AINM=AINM-A(K)*SIN(TH)
        RCP=RCP+PB(K)*COS(TH)
        RCM=RCM-B(K)*COS(TH)
        AICP=AICP-PB(K)*SIN(TH)
        AICM=AICM+B(K)*SIN(TH)
   20   CONTINUE
        Y1(I)=SQRT((RNP**2+AINP**2)/(RCP**2+AICP**2))
        Y2(I)=SQRT((RNM**2+AINM**2)/(RCM**2+AICM**2))
       IF(Y1(I).LT..00001) Y1(I)=.00001
       IF(Y2(I).LT..00001) Y2(I)=.00001
       Y1(I)=20.0*ALOG10(Y1(I))
       Y2(I)=20.0*ALOG10(Y2(I))
        IF(Y1(I).GT.Y1MAX) Y1MAX=Y1(I)
        IF(Y2(I).GT.Y2MAX) Y2MAX=Y2(I)
   50   CONTINUE
       WRITE (6,1020)
       CALL ZEROS(AO,A,ICR,0,IPFL)
       WRITE (6,1030)
       CALL ZEROS(1.0,B,ICR,1,IPFL)
       IF(ICR.EQ.NCRD) WRITE (2,1140)
       IF(Y2MAX.GT.Y1MAX) GO TO 60
       WRITE (6,1100)
       CALL FPLT(X,Y1,201,1)
       CALL FPLT(X,Y2,201,3)
       WRITE (6,1000)
       RETURN
   60  WRITE (6,1100)
       CALL FPLT(X,Y2,201,1)
       CALL FPLT(X,Y1,201,3)
       WRITE (6,1010)
```

```
      1CCC  FCFMAT(':0.. PLANT        ++ MCDEL')
      1CCC  FCFMAT(':0.. MODEL        ++ PLANT')
      1CCC  FCFMAT(':0MODEL ZERC LCCATIONS')
      1CCC  FCFMAT(':0MODEL PCLE LCCATIONS')
      1CCC  FCFMAT(':1')
      1CCC  FCFMAT(' ',12)
      1CCC  FCFMAT(' ',E14.5)
      1CCC  FCFMAT(' ',E14.5,3X,E14.5)
      1CCC  FCFMAT(' 55')
            RETURN
            END
C*****************************************************************************
C*****************************************************************************
C*    THIS SUBROUTINE CALCULATES THE ZEROS OF A POLYNOMIAL AND LISTS         *
C*    THEM IN A FILE AND ON THE SYSTEM PRINTER.                              *
C*****************************************************************************
C*****************************************************************************
C
      SUBROUTINE ZEROS(CC,C,N,IFL,IPFL)
      COMPLEX*8 Z(10)
      DIMENSION C(11),A(11)
      REAL*4 ZZ(2C)
      EQUIVALENCE (Z,ZZ)
      A(1)=CC
      DO 10 I=1,N
      IF=I+1
      IF((IFL.EQ.0) A(IF)=C(I)
      IF((IFL.EC.1) A(IP)=-C(I)
10    CONTINUE
      CALL ZRPOLY(A,N,Z,IER)
      DO 20 I=1,N
      RR=ZZ(I+(I-1)*2)
      RI=ZZ(2+(I-1)*2)
      R=SQRT(RR**2+RI**2)
      IF((RR.EC.C.0).AND.(RI.EQ.0.0)) RR=1.E-10
      THETA=ATAN2(RI,RR)
      THETA=THETA*180.0/3.141593
      IF((IPFL.EQ.1) WRITE (2,1100) RR,RI
      WRITE (6,1CCC) RR,RI,RAD,THETA
C
1CCC  FCFMAT(' ',E14.6,1CX,E14.6,10X,E14.6,10X,E14.6)
101C  FCFMAT(' ',E14.6)
110C  FCFMAT(' ',E14.5,3X,E14.5)
      RETURN
      END
C*****************************************************************************
C*****************************************************************************
C*    THIS SUBROUTINE UPDATES THE FORWARD AND BACKWARD PREDICTION ERROR      *
C*    COVARIANCE MATRICIES FROM ONE LATTICE STAGE TO THE NEXT GIVEN THE      *
C*    OLD COVARIANCE MATRICIES AND THE LATEST LATTICE REFLECTION             *
```

```
C*  COEFFICIENT MATRICIES.                                              *
C************************************************************************
C
      SUBROUTINE UPCOV(PF,PB,FK,BK,I)
      DIMENSION PF(2,2),PB(2,2),FK(2,2,1),BK(2,2,1)
      DIMENSION TEM(2,2)
      TEM(1,1)=1-FK(1,1,I)*FK(1,1,I)-FK(1,2,I)*FK(2,1,I)
      TEM(1,2)=-BK(1,1,I)*FK(1,2,I)-BK(1,2,I)*FK(2,2,I)
      TEM(2,1)=-FK(2,1,I)*FK(1,1,I)-BK(2,2,I)*FK(2,1,I)
      TEM(2,2)=1-FK(2,1,I)*FK(1,2,I)-FK(2,2,I)*FK(2,2,I)
      W1=PF(1,1)*TEM(1,1)+PF(1,2)*TEM(2,1)
      W2=PF(1,1)*TEM(1,2)+PF(1,2)*TEM(2,2)
      W3=PF(2,1)*TEM(1,1)+PF(2,2)*TEM(2,1)
      W4=PF(2,1)*TEM(1,2)+PF(2,2)*TEM(2,2)
      PF(1,1)=W1
      PF(1,2)=W2
      PF(2,1)=W3
      PF(2,2)=W4
      TEM(1,1)=1-FK(1,1,I)*BK(1,1,I)-FK(1,2,I)*BK(2,1,I)
      TEM(1,2)=-FK(1,1,I)*BK(1,2,I)-FK(1,2,I)*BK(2,2,I)
      TEM(2,1)=-FK(2,1,I)*BK(1,1,I)-FK(2,2,I)*BK(2,1,I)
      TEM(2,2)=1-FK(2,1,I)*BK(1,2,I)-FK(2,2,I)*BK(2,2,I)
      W1=PB(1,1)*TEM(1,1)+PB(1,2)*TEM(2,1)
      W2=PB(1,1)*TEM(1,2)+PB(1,2)*TEM(2,2)
      W3=PB(2,1)*TEM(1,1)+PB(2,2)*TEM(2,1)
      W4=PB(2,1)*TEM(1,2)+PB(2,2)*TEM(2,2)
      PB(1,1)=W1
      PB(1,2)=W2
      PB(2,1)=W3
      PB(2,2)=W4
      RETURN
      END
C
C************************************************************************
C*  THIS SUBROUTINE CALCULATES AND LISTS EIGENVALUES,EIGENVECTORS      *
C*  AND ELIPSOID AXIS HALF LENGTHS FOR THE PREDICTION ERROR COVARIANCE *
C*  MATRICIES.                                                         *
C************************************************************************
C
      SUBROUTINE EIG(P)
      DIMENSION P(2,2),PP(3),EVAL(2),EVEC(2,2),WK(5)
      PP(1)=P(1,1)
      PP(2)=P(2,1)
      PP(3)=P(2,2)
      CALL EIGRS(PP,2,1,EVAL,EVEC,2,WK,IER)
      A=1.0/SQRT(ABS(EVAL(1)))
      B=1.0/SQRT(ABS(EVAL(2)))
      WRITE (6,1000) EVAL(1),EVAL(2)
```

```
      WRITE (6,102C) A,B
      WRITE (6,101C) EVEC(1,1),EVEC(2,1),EVEC(1,2),EVEC(2,2)
100C  FORMAT(' ',10X,'EIGENVALUES ARE: ',2(E14.5,2X))
101C  FORMAT(' ',10X,'EIGENVECTORS ARE: ',2(' (',F14.5,',',E14.5,')'))
102C  FORMAT(' ',10X,' AXIS HALF LENGTHS ARE: ',2(E14.5,2))
      RETURN
      END
C
C*****************************************************************************
C*   THIS SUBROUTINE TAKES THE ROOTS, GAIN AND BULK DELAY DATA ENTERED IN    *
C*   IN IT AND CALCULATES THE SYSTEM TRANSFER FUNCTION. THE DATA IS          *
C*   ALSO LISTED ON THE SYSTEM PRINTER AND IN A FILE.                        *
C*****************************************************************************
C
      SUBROUTINE FCGEF(A,B,G,IC,N)
      COMPLEX*16 R(10),C(10)
      DIMENSION A(1),B(1),ZR(2,10),PR(2,10)
      REAL*8 RR(20),CC(20)
      EQUIVALENCE (R,RR),(C,CC)
C
C   SET THE PLANT ORDER
C
      N=7
C
C   SET THE GAIN
C
      G=1.0
C
C   SET THE PLANT DELAY
C
      IC=0
C
C   ENTER THE COMPLEX ZEROS IN R(1) THRU R(N)
C
      R(1)=(1.00253,.625)
      R(2)=(1.00253,-.625)
      R(3)=(0.0,0.0)
      R(4)=(0.0,0.0)
      R(5)=(0.0,0.0)
      R(6)=(0.0,0.0)
      R(7)=(0.0,0.0)
      R(8)=(0.0,0.0)
      R(9)=(0.0,0.0)
      R(10)=(0.0,0.0)
      WRITE (2,110C) N
      WRITE (6,100C) G,IC
      WRITE (6,101C)
      DO 10 I=1,N
```

```
      ZR(1,I)=RR(1+(I-1)*2)
      ZR(2,I)=RR(2+(I-1)*2)
      WRITE (6,1020) ZR(1,I),ZR(2,I)
10    CONTINUE
      CALL PIXPOL(A,N,C)
      DO 20 I=1,N
20    A(N+1-I)=CC(1+(I-1)*2)
C
C  ENTER THE N COMPLEX POLES IN P(1) THRU P(N)
C
      R(1)=(.5,0.0)
      R(2)=(.65262,.4)
      R(3)=(.65262,-.4)
      R(4)=(.2364140003,.6577845345)
      R(5)=(.2364140003,-.6577845345)
      R(6)=(-.2875,.4579646272)
      R(7)=(-.2875,-.4579646272)
      R(8)=(0.5,0.0)
      R(9)=(0.0,0.0)
      R(10)=(0.0,0.0)
      WRITE (6,1030)
      DO 30 I=1,N
      PP(1,I)=RR(1+(I-1)*2)
      PP(2,I)=RR(2+(I-1)*2)
      WRITE (6,1020) PP(1,I),PP(2,I)
30    CONTINUE
      CALL PIXPOL(D,N,C)
      DO 40 I=1,N
40    B(N+1-I)=CC(1+(I-1)*2)
      WRITE (6,1040)
      DO 50 I=1,N
50    WRITE (6,1050) I,A(I)
      WRITE (6,1060)
      DO 60 I=1,N
      BB=-B(I)
60    WRITE (6,1070) I,BB
      WRITE (7,1110) N
      DO 65 I=1,N
      A1=G*A(I)
      B1=-B(I)
65    WRITE (7,1120) A1,B1
      DO 66 I=1,N
66    WRITE (7,1120) ZR(1,I),ZR(2,I)
      DO 67 I=1,N
67    WRITE (7,1120) PP(1,I),PP(2,I)
      YCEN=-10.0
      NDEC=3
      WRITE (7,1130) YCEN,NDEC
```

```
      DIMENSION
      IF(NPLOT.EQ.1) GO TO 70
      DO 70 I=1,N
      X(I)=0.0
      Y(I)=0.0
70    CONTINUE
1000  FORMAT('1PLANT GAIN IS ',E14.5,' WITH ',I2,' PURE DELAYS')
1010  FORMAT('ZERO LOCATIONS FOR PLANT')
1020  FORMAT(' ',E14.6,1X,E14.6)
1030  FORMAT('POLE LOCATIONS FOR PLANT')
1040  FORMAT('PLANT NUMERATOR COEFFICIENTS')
1050  FORMAT(' B(',I2,')=',E14.5)
1060  FORMAT('DENOMINATOR COEFFICIENTS FOR PLANT')
1070  FORMAT(' E(',I2,')=',E14.5)
1080  FORMAT('  ',I2)
1090  FORMAT(' ',E14.5)
1100  FORMAT(' ',E14.5,3X,E14.5)
1110  FORMAT(' ',E14.5,3X,I2)
      RETURN
      END
C
C*****************************************************************
C*  THIS SUBROUTINE PRODUCES THE PLOTS ON THE SYSTEM PRINTER IF PLOTP  *
C*  IS CALLED OR ON THE TYPE SETTING TERMINAL IF PLOTT IS CALLED       *
C*****************************************************************
C
      SUBROUTINE PPLT(X,Y,N,P)
      DIMENSION X(1),Y(1)
C     WRITE (6,1000)
      CALL PLOTP(X,Y,N,P)
1000  FORMAT('1')
      RETURN
      END
```

LIST OF REFERENCES

1. Alper, P., "A Consideration of the Discrete Volterra Series", IEEE Trans. A.C., Vol. 10, pp. 322-327, July 1965.

2. Atal, B.S. and Hanauer, S.L., "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", J. Acoust. Soc. Am., Vol. 50, pp. 637-655, 1971.

3. Athans, M. and Schweppe, F.C., Gradient Matrices and and Matrix Calculations, Tech note 1965-53, MIT Lincoln Laboratory, 17 November 1965.

4. Box, G.E.P. and Jenkins, G.M., Time Series Analysis, Rev. ed., Holden-Day, 1967.

5. Burg, J.P., Maximum Entropy Spectral Analysis, Ph.D. Dissertation, Stanford University, May 1975.

6. Bussgang, J.J., Ehrman, L., Graham, J.W., "Analysis of Nonlinear Systems with Multiple Inputs", Proceedings of the IEEE, Vol. 62, No. 8, pp. 1088-1110, August 1974.

7. Clancy, D.J. and Rugh, W.J., "A Note on the Identification of Discrete-Time Polynomial Systems", IEEE Trans. A.C., Vol. AC-24, No. 6, December 1979.

8. Dudley, D.G., "Fitting Noisy Data With a Complex Exponential Series", Lawrence Livermore Lab Report #UCRL 52242, 7 March 1977.

9. Durbin, J., "Efficient Estimation of Parameters in Moving Average Models", Biometrika, Vol. 46, Parts 1 and 2, pp. 306-316, 1959.

10. Durbin, J., "The Fitting of Time Series Models", Rev. Inst. Int. Statist., Vol. 28, No. 3, pp. 233-244, 1960.

11. Eykhoff, P., System Identification, John Wiley and Sons, 1974.

12. Flanagan, J.L., Speech Analysis Synthesis and Perception, 2nd ed., Springer Verlag, 1972.

13. Fu, F.C., and Parlaon, J.B., "On the Volterra-Series Functional Identification of Non-Linear Discrete-Time Systems", Int. J. Control, Vol. 18, No. 6, pp. 1281-1289, 1973.

14. George, D.A., "Continuous Nonlinear Systems", MIT Research Laboratory of Electronics, Tech. Report 335, 24 July 1959.

15. Griffiths, L.J., "Rapid Measurement of Digital Instantaneous Frequency", IEEE Trans. on A.S.S.P., Vol. ASSP-23, No. 2, pp. 207-222, April 1975.

16. Griffiths, L.J., "A Continuously-Adaptive Filter Implemented as a Lattice Structures", Proc. of the 1977 IEEE Int. Conf. on A.S.S.P., pp. 87-90.

17. Griffiths, L.J., "An Adaptive Lattice Structure For Noise Cancelling Applications", Proc. of the IEEE 1978 Int. Conf. on A.S.S.P., pp. 87-90.

18. Griffiths, L.J., "Adaptive Structures For Multiple Input Noise Cancelling Applications", Proc. of IEEE 1979 Int. Conf. on A.S.S.P., pp. 925-928.

19. Hsia, T.C., System Identification, Lexington Books, D.C. Heath and Company, 1977.

20. Itakura, F., and Saito, S., "On the Optimum Quantization of Feature Parameters in the Parcor Speech Synthesizer", Proc. 1972 Conf. Speech Commun, Process, pp. 434-437, 1972.

21. Jackson, L.B., Tufts, D.W., Soong, F.K., and Rao, R.M., "Frequency Estimation by Linear Prediction", Proc. of the IEEE 1978 Int. Conf. on A.S.S.P.

22. Kailath, Thomas, "A View of Three Decades of Linear Filter Theory", IEEE Trans. on Information Theory, Vol IT-20, No. 2, pp. 146-181, March 1974.

23. Kalman, R.E., "Design of a Self-Optimizing Control System", Transactions A.S.M.E., Vol. 80, pp. 468-478, February 1958.

24. Kay, S.M., "The Effect of Noise on the AR Spectral Estimator", IEEE Trans. on A.S.S.P., Vol. ASSP-27, No. 5, pp. 478-485, October 1979.

25. Kim, Y.C., Tong, W.F., Powers, E.J. and Roth, J.R., "Extension of the Coherence Function to Quadratic Models", Proc. of IEEE, Vol. 67, No. 3, pp. 428-429, March 1979.

26. Koopmans, L.H., The Spectral Analysis of Time Series, Academic Press, 1974.

27. Levinson, N., "The Wiener RMS Error Criteria in Filter Design and Prediction", J. Math Phys., Vol. 25, pp. 261-278, January 1947.

28. Makhoul, J., "Linear Prediction: A Tutorial Review", IEEE Proc., Vol. 63, No. 4, pp. 561-580, April 1975.

29. Makhoul, J., "Stable and Efficient Lattice Methods for Linear Prediction", IEEE Trans. on A.S.S.P., Vol. A.S.S.P. 25. No. 5, October 1977.

30. Makhoul, J., "A Class of All Zero Lattice Digital Filters: Properties and Application", IEEE Trans. on A.S.S.P., Vol. A.S.S.P. 26, No. 4, pp. 304-314, August 1970.

31. Makhoul, J. and Viswanathan, R., "Adaptive Lattice Methods for Linear Prediction", Proc. of the IEEE 1978 Int. Conf. on A.S.S.P., pp. 83-86.

32. Markel, J.E., Linear Prediction of Speech, pp. 20-41, Springer Verlay, 1975.

33. Markel, J.E. and Gray, A.H., "Roundoff Characteristics of a Class of Orthogonal Polynomial Structures", IEEE Trans. on A.S.S.P., Vol. A.S.S.P.-23, pp. 473-486, October 1975.

34. Mendel, J.M., Discrete Techniques of Parameter Estimation, Marcel Dekker Inc., 1973.

35. Mitzel, G.E., Clancy, S.J., and Rugh, W.J., "On Transfer Function Representation for Homogeneous Nonlinear Systems", IEEE Trans. on Auto. Control, Vol. AC-24, No. 2, pp. 242-249, April 1979.

36. Morf, M., Lee, D., Nickolls, J. and Vieira, A., "A Classification of Algorithms for ARMA Models and Ladder Realizations", Proc. of the IEEE 1977 Int. Conf. on A.S.S.P., pp. 13-19.

37. Morf, M., and Lee, D., "Recursive Least Square Ladder Forms for Fast Parameter Tracking", Proc. 1978 IEEE Conf. on Decision and Control, January 12, 1979, San Diego, CA, pp. 1362-1367.

38. Morf, M., and Lee, D., "Fast Algorithms for Speech Modeling", Tech. Report #M308-1, Infor. Syst. Lab, Stanford University, Stanford, CA, DCA Contract #DCA 100-77-C-0008.

39. Morf, M. and others, "Recursive Multichannel Maximum Entrophy Spectral Estimation", IEEE Trans. on Geoscience Electronics, Vol. GE-16, April 1978.

40. Nuttall, A.H., "Multivariate Linear Predictive Spectral Analysis Employing Forward and Backward Averaging: A Generalization of Burg's Algorithm", NUSC Technical Report 5501, New London, CT, 13 Oct 1976.

41. Nuttall, A.H., "Positive Definite Spectral Estimate and Stable Correlation Recursion for Multivariate Linear Predictive Spectral Analysis", NUSC Tech. Doc. 5729, New London, CT, 14 Nov 1976.

42. Oppenheim, A.V. and Schafer, R.W., Digital Signal Processing, Prentice Hall Inc., 1975.

43. Pagano, M., "Estimation of Models of Autoregressive Signal Plus White Noise", Annals of Statistics, Vol. 2, No. 1, pp. 99-108, 1974.

44. Rabiner, L.R., and Schafer, R.W., Digital Processing of Speech Signals, Prentice-Hall, 1978.

45. Robinson, E.A., Multichannel Time Series Analysis with Digital Computer Programs, Rev-Ed., Holden-Day, 1967.

46. Satorius, E.H., Smith, J.D., and Reeves, P.M., "Adaptive Noise Cancelling of a Sinusoidal Interference Using A Lattice Structure", Proc. of the IEEE 1979 Int. Conf. on A.S.S.P., pp. 029-032.

47. Satorius, E.H. and Alexander, S.T., "Rapid Equalization of Communications Channels Using Adaptive Lattice Algorithims", Proc. of the IEEE 1979 Int. Conf. on A.S.S.P., pp. 294-298.

48. Schetzen, M., "Measurement of the Kernels of a Non-Linear System of Finite Order", Int. J. Control, Vol. 1, pp. 251-263, 1965.

49. Shanks, J.L., "Recursion Filters for Digital Signal Processing", Geophysics, Vol. 32, No. 1, pp. 33-51, February 1967.

50. Strand, O.N., "Multichannel Complex Maximum Entrophy (Autoregressive) Spectral Analysis", IEEE Trans. Auto. Cont., Vol. AC-22, pp. 634-640, 1977.

51. Strang, G., Linear Algebra and Its Applications, Academic Press, 1976.

52. Tuttle, D.F., Jr., "On Fluids, Networks, and Engineering Education", From Aspects of Network and System Theory, edited by: R.E. Kalman and N. DeClaris, pp. 591-612, Holt Rinehart and Winston Inc., 1970.

53. Unbehanen, H. and Bobring, B., "Tests For Determining Order in Parameter Estimation", Automatica, Vol. 10, pp. 233-244, May 1974.

54. Van Trees, H.L., Synthesis of Nonlinear Control Systems, the M.I.T. Press, 1962.

55. Van Trees, H.L., Functional Techniques for the Analysis of the Nonlinear Behavior of Phase Locked Loops, Proc. IEEE, 894-911, August 1964.

56. Weis, L. and McDonough, R.N., "Prony's Method, Z Transforms and Pade Approximation", SIAM Review, Vol. 5, pp. 145-149, April 1963.

57. Whittle, P., "On the Fitting of Multivariate Auto-regressions and the Approximate Factorization of a Spectral Density Matrix", Biometrika, Vol. 50, pp. 129-134.

58. Widrow, B., "Adaptive Filters" from Aspects of Network and Systems Theory, edited by R.E. Kalman and N. DeClaris, Holt, Reinhart and Winston Inc., 1970.

59. Widrow, B. and others, "Adaptive Noise Cancelling: Principles and Applications", Proc. of the IEEE, Vol. 63, No. 12, Dec. 1975.

60. Widrow, B. and others, "Stationary and Non Stationary Learning Characteristics of the LMS Adaptive Filter", Proc. of the IEEE, Vol. 64, No. 8, pp. 1151-1162, August 1976.

61. Wiggins, R.A. and Robinson, E.A., "Recursive Solution to the Multichannel Filtering Problem", J. Geophysics Research, Vol 70, pp. 1885-1891, MR 32#609, 1965.

62. Wiener, N., "Response of a Nonlinear Device to Noise", MIT Radiation Laboratory, Cambridge, Mass., Rept. No. V-186, April 1942.

63. Muir, T., A Treatise on the Theory of Determinants, Revised and enlarged by W.H. Metzler, Dover Publications Inc., 1960.

64. Parker, S.R., "An Autoregressive Moving Average (ARMA) Discrete Nonlinear Model", Proc. of the 1980 Int. Sympoisum on Circuits and Systems, pp. 918-920, April 1980.

65. Perry, F.A. and Parker, S.R., "Adaptive Solution of Multichannel Lattice Models for Linear and Nonlinear Systems, Proc. of the 1980 Int. Symposium on Circuits and Systems, pp. 744-747, April 1980.

66. Perry, F.A. and Parker, S.R., "Recursive Solutions for Zero Pole Model", Proc. of the Thirteenth Asilomar Conf. on Circuits, Systems and Computers, November 1979.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor S. R. Parker, Code 62Px Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	5
5.	Professor R. W. Hamming, Code 52Hg Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6.	Professor R. Panholzer, Code 62Pz Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
7.	Professor H. A. Titus, Code 62Ts Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
8.	Professor F. C. Wang, Code 53Wg Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
9.	Professor D. E. Kirk, Code 62Ki Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
10.	Professor G. Thaler, Code 62Tr Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1

11. Professor T. Tao, Code 6 — 1
Department of Electrical Engineering
Naval Postgraduate School
Monterey, California 93940

12. Lieutenant Francis A. Perry — 3
124 Belle Drive
Marina, California 93933

13. Captain H. Cox, USN — 1
Naval Electronic Systems Command
Naval Electronic Systems Command Headquarters
PME-124
Washington, D.C. 20360

14. Commander — 1
Naval Surface Weapons Center
White Oak Laboratory
ATTN: LCDR John Gauss, USN
Silver Springs, Maryland 20910

15. Mr. Martin Mandelberg — 1
U.S. Coast Guard Research and Development Center
Avery Point
Groton, Connecticut 06340

16. Lieutenant Commander John Martinos — 1
Hellenic Navy
Clafkio 22
Chalandri
Athens, Greece

17. Lieutenant Martin Romeo — 1
SMC 2280
Naval Postgraduate School
Monterey, California 93940

18. Lieutenant Vasilius Xiomas — 1
SMC 2690
Naval Postgraduate School
Monterey, California 93940

19. Dr. Joel Norris — 1
Code 427-ONR
Office of Naval Research
800 North Quincy
Arlington, Virginia 22217

20. Commander Stacey Holmes, USN — 1
Naval Electronic Systems Command
Code 310
2511 Jefferson Davis Highway
Arlington, Virginia 22202

21. Mr. Nate Butler — 1
Naval Electronic Systems Command
Naval Electronic Systems Command Headquarters
Washington, D.C. 20360

22. Dr. A. S. Milton — 1
Naval Research Laboratory
Code 6550
Washington, D.C. 20375

23. Professor Jannice Jenkins — 1
Northwestern University Medical School
Reingold ECG Center
310 E. Superior Street
Chicago, Illinois 60611

24. Professor Sammuel Bedrosean — 1
Department of Systems Engineering
Moore School DZ
University of Pennsylvania
Philadelphia, Pennsylvania 19174

25. Dr. Stephan Horvath — 1
E.T.H. Zurich
CH-8093 Zurich
Switzerland

26. Professor Jim McClellan — 1
M.I.T.
Department of E.E. and C.S.
Cambridge, Massachusetts 02139

27. Commander — 2
Naval Ocean Systems Center
ATTN: Lieutenant Frank Perry, USN
San Diego, California 92152

28. Professor Martin Morf — 1
Stanford University
Information Systems Laboratory
Durand 109
Stanford, California 94305

29. Professor Bernard Widrow — 1
Stanford University
800 Lathrop Drive
Stanford, California 94305

30. Professor Lloyd Griffiths — 1
M.I.T.
Department of E.E. and C.S.
Room 36-615
77 Massachusetts Avenue
Cambridge, Massachusetts 02139

31. Professor A. V. Oppenheim — 1
M.I.T.
Department of E.E. and C.S.
Room 36-393
Cambridge, Massachusetts 02139

32. Dr. Ben Friedlander — 1
Systems Control Inc.
1801 Page Mill Road
Palo Alto, California 94304

33. Dr. John Triechler — 1
Argo Systems
884 Hermosa Court
Sunnyvale, California 94086

34. Dr. Bill Hodgkiss — 1
University of California at San Diego
Marine Physical Laboratory
Scripts Institution of Oceanography
San Diego, California 92152

35. Commander — 1
Naval Ocean Systems Center
ATTN: Dr. Harper Whitehouse
San Diego, California 92152