(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

SYSTEM IDENTIFICATION BY ARMA MODELING

by

Paul S. Dal Santo

September 1988

Thesis Advisor Murali Tummala

Approved for public release; distribution is unlimited.

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School — 6b Office Symbol *(if applicable)* 32	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization — 8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers Program Element No / Project No / Task No / Work Unit Accession No

11 Title *(include security classification)* SYSTEM IDENTIFICATION BY ARMA MODELING

12 Personal Author(s) Paul S. Dal Santo

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	September 1988	83

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	system identification, ARMA, multichannel, instrumental variable

19 Abstract *(continue on reverse if necessary and identify by block number)*

System identification concerns the mathematical modeling of a system based upon its input and output. It allows the development of a mathematical description when all that is available is the result of a process or the output of a system and not the process or system itself.

The purpose of this thesis is to develop algorithms for modeling systems as autoregressive-moving-average processes using the method of instrumental variables, a modification of the ordinary least-squares technique, and a multichannel method based upon processing the input and output data by separate infinite-impulse-response filters. The methods developed are tested by computer simulation using several second and third-order test cases and the results are presented.

20 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Murali Tummala	22b Telephone *(include Area code)* (408) 646-2645	22c Office Symbol 62Tu

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

SYSTEM IDENTIFICATION BY ARMA MODELING

by

Paul S. Dal Santo
Lieutenant, United States Coast Guard
B.S.E.E., United States Coast Guard Academy, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

Author: Paul S. Dal Santo

Paul S. Dal Santo

Approved by: Murali Tummala

Murali Tummala, Thesis Advisor

Lawrence J. Ziomek

Lawrence J. Ziomek, Second Reader

John P. Powers

John P. Powers, Chairman,
Department of Electrical and Computer Engineering

G E Schacher

Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

System identification concerns the mathematical modeling of a system based upon its input and output. It allows the development of a mathematical description when all that is available is the result of a process or the output of a system and not the process or system itself.

The purpose of this thesis is to develop algorithms for modeling systems as autoregressive-moving-average processes using the method of instrumental variables, a modification of the ordinary least-squares technique, and a multichannel method based upon processing the input and output data by separate infinite-impulse-response filters. The methods developed are tested by computer simulation using several second and third-order test cases and the results are presented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. SYSTEM IDENTIFICATION BASICS

System identification concerns the modeling of systems as sets of mathematical equations based upon the input and output of the system. [Ref. 1: pp. 3-6]. It allows a model to be developed when all that is available is the result of a process or the output of a system and not the process or the system itself. System identification is an important area of study. Solution of the modeling problem offers many alternatives for the continued study of the system. Among these are:

- Nondestructive analysis of the system.
- Simulation studies using the model.
- Easy adaptation of the model to changing system environment.
- Spectral analysis of the system.

Modeling can simulate the system's operation at a fraction of the cost of actual system operation. Complex operations not possible with the actual system for fear of damaging it or personal injury can be simulated. This can expose how the system will operate in adverse conditions not normally experienced. In speech processing, modeling the speech process has the potential for significantly reducing the amount of information necessary to store in order to reproduce the speech.

The modeling process shown in Figure 1 on page 2 assumes the unknown system's input and the output data are available for processing. In many cases, if the system's input is unknown or data is not available, a white noise input can be used in its place. The modeling process uses the input and output data to find a set of parameters which closely approximate the operation of the system. The better the identification technique, the more closely the model follows the performance of the actual system.

Many types of models are available. This thesis investigates a linear parametric model that can be described by difference equations. This type of model lends itself well to simulation on a digital computer. The frequency characteristics of the system determined from the parameters of these types of models are more accurate than what can be determined from classical means such as FFTs. This is because classical methods use windows which assume data beyond their extent is zero [Ref. 2: p. 173]. This is not a realistic assumption. Models in this category include the moving-average (MA) model,

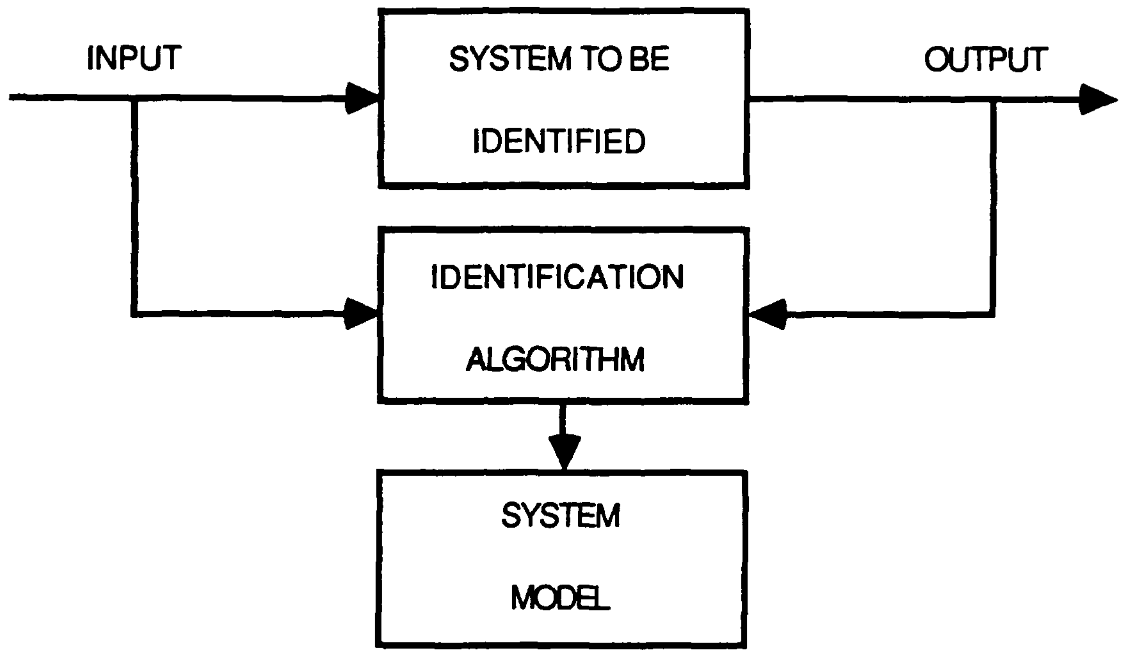


Figure 1. System identification problem

the autoregressive (AR) model, and the autoregressive-moving-average (ARMA) model. In the frequency domain, MA processes are characterized by sharp nulls and smooth peaks and AR processes are characterized by smooth nulls and sharp peaks. ARMA processes have sharp peaks and sharp nulls [Ref. 2: p. 173]. An advantage of the MA process is its inherent stability. An advantage of the AR process is the large number of algorithms already available for modeling systems. An advantage of the ARMA process is that it uses far fewer parameters than either the MA or AR process alone to model a system. This satisfies the general requirement to reduce the complexity of the model.

In addition to a large variety of models, there are two processing modes: block and sequential.

Block processing uses a fixed length block of data in the parameter estimation process. It ignores data before and after the block. This is not a real time processing method because all data must be available before processing can start. Block processing generally involves inversions of data matrices whose sizes are on the order of $(N + M) \times (N + M)$ where N is the order of the AR process and M is the order of the MA process.

Sequential processing uses new data to update the parameter estimations. It starts by initializing an estimate of the inverse of the data covariance as as a diagonal matrix. It uses each new data point to update this matrix. Then it updates the parameter estimates using the updated inverse data covariance matrix. It is a real time method. The algorithm to implement the sequential processing method is generally more complex than the block method but less computationally intensive because the matrix inversions are not required.

This thesis concerns only systems represented by discrete time data uniformly sampled at a sufficient rate to meet the Nyquist criteria.

The work in this thesis assumes that the input data is a wide-sense stationary random sequence. Tests of the algorithms used a pseudorandom Gaussian input with a mean of zero and a variance of one.

B. PROBLEM STATEMENT

The purpose of this thesis is to develop algorithms for modeling systems as ARMA processes using the method of instrumental variables (IV) and a multichannel approach. Tests of the methods will be conducted to determine the accuracy of their results and the speed with which they converge.

The IV approach is a modification of the method of ordinary least squares. This approach is developed first as a block processing case and then converted to a sequential processing case. Tests are conducted of only the sequential processing case.

Using a multichannel scheme allows the input and output data of the unknown system to be processed separately. This reduces the sizes of the data matrices involved in the modeling process. Both block and sequential processing cases are formulated but only the block processing case is tested.

C. OVERVIEW OF THESIS

Chapter 2 is about ARMA modeling. It also presents a detailed derivation of the method of ordinary least squares because it forms the basis on which other modeling techniques depend.

Chapter 3 presents a modified least-squares approach called the method of instrumental variables. It is attractive due to its simplicity and good noise performance. Chapter 3 presents results of using this method on several second and third-order test systems.

Chapter 4 presents a new multichannel approach to ARMA modeling. This approach is presented in block and sequential processing forms. This chapter also presents several adaptations of the block form which improve its speed of convergence.

Chapter 5 contains a summary of the thesis and lists topics for further research.

The appendix contains the programs used to test the sequential IV algorithm and the block multichannel iterative algorithm. Subroutines common to both programs are grouped together and listed at the end of the appendix.

II. ARMA MODELING

A. ARMA PROCESSES

Modeling as an autoregressive-moving-average (ARMA) process has the potential for achieving a close fit to the system using a reduced order over that which a moving average or an autoregressive model alone could achieve. ARMA modeling is concerned with finding a set of AR parameters and MA parameters which combined describe an ARMA process that approximates the characteristics of a target system.

The general form of the ARMA model is shown in Figure 2 on page 6. The output at time n, $y(n)$, is a linear combination of past outputs and past and present inputs. The a_i and b_i are constants referred to as tap weights. The a_i parameters form the MA part of the ARMA model. The b_i parameters form the AR part. In equation form the output of the ARMA system is represented by the following difference equation:

$$y(n) = -\sum_{i=1}^{N} b_i y(n-i) + \sum_{i=0}^{M} a_i u(n-i) \tag{2.1}$$

where N is the order of the AR part of the ARMA model and M is the order of the MA part of the ARMA model. This means the ARMA output at the current time depends on the last N values of the ARMA output. The N b_i weighting parameters determine exactly how the new output depends on the past outputs. The M a_i weighting parameters determine how the new output depends on the current and $M-1$ past inputs.

Equation (2.1) in vector form becomes:

$$y = \mathbf{x}^T \boldsymbol{\theta} \tag{2.2}$$

where $\mathbf{x}$ is a $(N+M+1) \times 1$ vector of input and output data values given by:

$$\mathbf{x} = [\, -y(n-1) \quad -y(n-2) \quad \dots \quad -y(n-N) \quad x(n) \quad x(n-1) \quad \dots \quad x(n-M)]^T \tag{2.3}$$

and $\boldsymbol{\theta}$ is a $(N+M+1) \times 1$ vector of the AR and MA tap weights given by:

$$\boldsymbol{\theta} = [b_1 \quad b_2 \quad \dots \quad b_N \quad a_0 \quad a_1 \quad \dots \quad a_M]^T \tag{2.4}$$

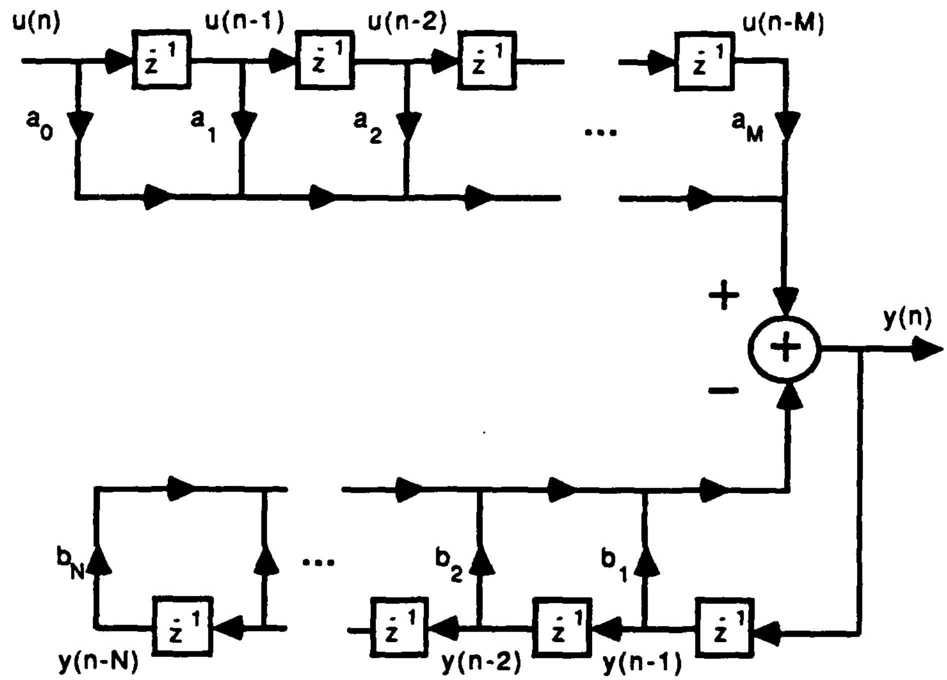


Figure 2. Structure of the ARMA model

For $N + L - 1$ data points available for y and $M + L$ data points available for u, we can write a block equation which gives the value of the output at progressive sampling times:

$$\begin{bmatrix} y(n-L+1) \\ y(n-L+2) \\ \cdot \\ \cdot \\ \cdot \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^T(n+1) \\ \mathbf{x}^T(n+2) \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{x}^T(n+L) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_N \\ a_0 \\ a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_M \end{bmatrix} \tag{2.5}$$

The i^{th} row in equation (2.5) is the value of y at time $n+i$ based on output data available through time $n+i-1$ and input data available through $n+i$. The i^{th} row is identically equation (2.2) at time $n+i$. In vector form equation (2.5) becomes:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} \tag{2.6}$$

where $\boldsymbol{\theta}$ is defined in equation (2.4); y, the vector of output values, is given by:

$$\mathbf{y} = [y(n-L+1) \;\; y(n-L+2) \;\; \ldots \;\; y(n)]^T \tag{2.7}$$

and **X** is a partitioned matrix with rows comprised of data vectors exactly like equation (2.3) only shifted in time. At successive sampling times, when new data is obtained, data used to calculate the previous output shifts one column to the right. The new data fills in the left most y and u columns. The matrix **X** is given by:

$$\mathbf{X} = \begin{bmatrix} -y(n-L) & \ldots & -y(n-\eta+1) & u(n-L+1) & \ldots & u(n-\mu+1) \\ -y(n-L+1) & \ldots & -y(n-\eta+2) & u(n-L+2) & \ldots & u(n-\mu+2) \\ \cdot & \ldots & \cdot & \cdot & \ldots & \cdot \\ \cdot & \ldots & \cdot & \cdot & \ldots & \cdot \\ \cdot & \ldots & \cdot & \cdot & \ldots & \cdot \\ -y(n-1) & \ldots & -y(n-N) & u(n) & \ldots & u(n-M) \end{bmatrix} \tag{2.8}$$

where η is defined as $N+L$ and μ is defined as $M+L$.

If the a_i and b_i are estimates of the true values of the AR and MA parameters, then the filter output will be an estimate of the true output. We use a hat over a variable (for example, $\hat{y}$) to indicate an estimated value. Rewriting equation (2.6) using the estimated ARMA parameters results in:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\theta}} \tag{2.9}$$

where $\hat{\boldsymbol{\theta}}$ is defined as:

$$\hat{\boldsymbol{\theta}} = [\hat{b}_1 \;\; \hat{b}_2 \;\; \dots \;\; \hat{b}_N \;\; \hat{a}_0 \;\; \hat{a}_1 \;\; \dots \;\; \hat{a}_M]^T \tag{2.10}$$

and $\hat{\mathbf{y}}$ is now the vector of estimated output values and is given by:

$$\hat{\mathbf{y}} = [\hat{y}(n-L+1) \;\; \hat{y}(n-L+2) \;\; \dots \;\; \hat{y}(n)]^T \tag{2.11}$$

Up until now, we have discussed estimating the output of a system given its input, past output, and an estimate of the parameters which describe it. If, however, we know the output and input of the system, based on these equations, we can use them to generate a set of $\hat{a}_i$ and $\hat{b}_i$ which produces an ARMA output which is the best possible estimate of the system output. Then the $\hat{a}_i$ and $\hat{b}_i$ will be optimal parameters for describing the operation of the unknown system as an ARMA process.

B. METHOD OF ORDINARY LEAST-SQUARES

In this thesis we use the method of ordinary least- squares as the means of finding the optimum set of ARMA parameters. It is a well known modeling technique. It offers the advantage of being widely used in the scientific community for a variety of modeling problems. It has been applied successfully to a large number of modeling problems with good results and has been successfully applied to classes of problems for which other methods have failed. [Ref. 3: p. 4]

To apply the method of ordinary least-squares to system identification we form the error between the actual system output and the estimated output generated by the ARMA model. This error is given by:

$$\boldsymbol{\varepsilon} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\theta}} \tag{2.12}$$

where $\mathbf{y}$ is the vector of the actual system outputs given by equation (2.7) and $\hat{\mathbf{y}}$ is the vector of ARMA outputs given by equation (2.11). [Ref. 1: p. 176]

We then let the sum of the squares of the errors at the instances of time the measurements of the data were taken become a measure of how well the estimates approximate the true system outputs. This measure of performance, or cost function, is denoted J. It is written in equation form as:

$$J = \sum_{i=n+1}^{n+L} \varepsilon_i^2 = \varepsilon^T \varepsilon \tag{2.13}$$

Replacing the error in equation (2.13) with its equivalent expression from equation (2.12) results in:

$$J = \mathbf{y}^T\mathbf{y} + \hat{\theta}^T\mathbf{X}\mathbf{X}^T\hat{\theta} - 2\hat{\theta}^T\mathbf{X}\mathbf{y} \tag{2.14}$$

Equation (2.14) shows that the performance measure is a function of the estimated parameters. The criterion is to minimize the measure of performance by taking its derivative with respect to the parameter estimates and setting it equal to zero. Then equation (2.14) becomes:

$$\frac{\partial J}{\partial \hat{\theta}} = 0 = 0 + 2\mathbf{X}\mathbf{X}^T\hat{\theta} - 2\mathbf{X}^T\mathbf{y} \tag{2.15}$$

Solving for $\hat{\theta}$, the parameters, gives us the result:

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.16}$$

Equation (2.16) is the ordinary least-squares solution for the optimum ARMA parameters. It provides the best possible description, in a least-squares sense, of the data source. The resulting parameters provide the closest fit to the actual input and output data of the system in the sense of least-squares errors.

Equation (2.16) uses a block processing approach. The product of $\mathbf{X}^T\mathbf{X}$ must be formed and then inverted in order to calculate $\hat{\theta}$. In addition to being computationally intensive, the estimate cannot be updated when new data becomes available without recalculating $(\mathbf{X}^T\mathbf{X})^{-1}$. A sequential update which does not require $(\mathbf{X}^T\mathbf{X})^{-1}$ to be recalculated is presented in the next chapter in the context of the instrumental variable method of least-squares.

III. INSTRUMENTAL VARIABLE METHOD OF SYSTEM IDENTIFICATION

A. INTRODUCTION

The instrumental variable (IV) method of system identification is a variation of the method of ordinary least-squares. Its attraction over ordinary least-squares is that there is no bias in estimating the parameters when dealing with noise [Ref. 4: p. 406]. Also, this method is known to yield consistent estimates and remains as easy to use as the method of ordinary least-squares [Ref. 3: p. 119].

When an additive noise term is present in the observable output, $y(n)$, the output is given by:

$$y(n) = w(n) + v(n) \tag{3.1}$$

Here $w(n)$ represents the actual output of the system and $v(n)$ represents the noise. When this noise has a non-zero mean, using the noise corrupted output to model the unknown system by the ordinary least-squares approach leads to inaccurate estimates of its parameters. The parameters are referred to as biased estimates. [Ref. 3: p. 119, Ref. 1: pp. 192-193, and Ref. 5: p. 704].

The IV method shown in Figure 3 on page 11 generates an estimate of the unknown system's output by processing the input data through an auxiliary model which closely approximates the unknown system. In our implementation of the IV method, the auxiliary model is an ARMA model. Its output is free of the noise affecting the unknown system. The IV method uses the auxiliary model output (estimate), $\hat{w}$, to calculate the parameters of the unknown system. Therefore the IV parameter estimates are not biased like those generated by the method of ordinary least-squares.

The IV method assumes the existence of a matrix $\mathbf{Z}$ composed of the auxiliary model's input and output data which has the following two properties [Ref. 4: p. 406]:

$$\lim_{N\to\infty} \frac{1}{N} \mathbf{Z}^T \boldsymbol{\varepsilon} = \mathbf{0} \tag{3.2}$$

$$\lim_{N\to\infty} \frac{1}{N} \mathbf{Z}^T \mathbf{X} = \mathbf{Q} \tag{3.3}$$

where ε is the error in fitting the parameter estimates to the data and is given by:

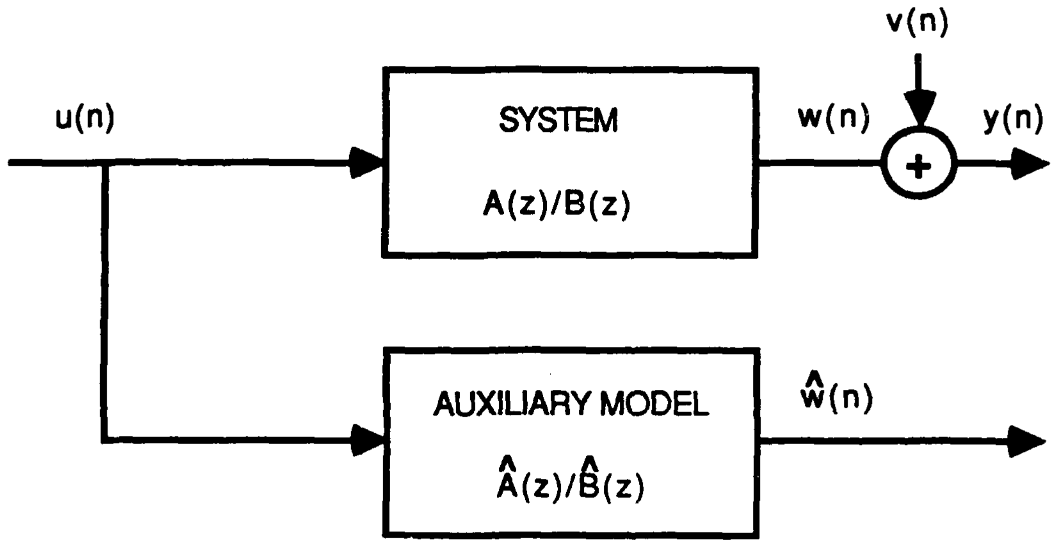


Figure 3. Modeling by the instrumental variable method

$$\varepsilon = \mathbf{y} - \mathbf{X}\hat{\theta}_{IV} \tag{3.4}$$

and **Q** is a nonsingular square matrix.

The first property means **Z** is orthogonal to the error. This leads to the cancellation of the bias term inherent in ordinary least-squares techniques [Ref. 4: p. 406]. The second property ensures the inverse of $\mathbf{Z}^T\mathbf{X}$ exists. **Z** is assumed to have the same structure and size as the data matrix **X** in equation (2.8). Its contents differ in that the noise corrupted system output $y(n)$ in **X** is replaced by the output of the auxiliary model $\hat{w}(n)$ in **Z**. The new data matrix **Z** is given by:

$$\mathbf{Z} = \begin{bmatrix} -\hat{w}(n-L) & \dots & -\hat{w}(n-\eta+1) & u(n-L+1) & \dots & u(n-\mu+1) \\ -\hat{w}(n-L+1) & \dots & -\hat{w}(n-\eta+2) & u(n-L+2) & \dots & u(n-\mu+2) \\ . & \dots & . & . & \dots & . \\ . & \dots & . & . & \dots & . \\ . & \dots & . & . & \dots & . \\ -\hat{w}(n-1) & \dots & -\hat{w}(n-N) & u(n) & \dots & u(n-M) \end{bmatrix} \tag{3.5}$$

where η is defined as $N + L$ and μ is defined as $M + L$. Comparing **X** in equation (2.8) and **Z** in equation (3.5), we note the substitution of $\hat{w}(n)$ for $y(n)$. Thus, we are now using estimates of the true output $\hat{w}(n)$ instead of noise corrupted samples $y(n)$.

To incorporate **Z** into the parameter estimation process we begin with equation (2.12), which we rewrite as:

$$\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\theta}} + \boldsymbol{\varepsilon} \tag{3.6}$$

This equation says that the estimates of the output, given by $\mathbf{X}\hat{\boldsymbol{\theta}}$, differ from the actual outputs, **y**, by some fitting error $\boldsymbol{\varepsilon}$. Multiplying equation (3.6) by $\mathbf{Z}^T$ yields:

$$\mathbf{Z}^T\mathbf{y} = \mathbf{Z}^T\mathbf{X}\hat{\boldsymbol{\theta}} + \mathbf{Z}^T\boldsymbol{\varepsilon} \tag{3.7}$$

Equation (3.3) ensures that $\mathbf{Z}^T\mathbf{X}$ can be inverted. Solving for $\hat{\boldsymbol{\theta}}$ results in:

$$\hat{\boldsymbol{\theta}} = (\mathbf{Z}^T\mathbf{X})^{-1}\mathbf{Z}^T\mathbf{y} - (\mathbf{Z}^T\mathbf{X})^{-1}\mathbf{Z}^T\boldsymbol{\varepsilon} \tag{3.8}$$

The $(\mathbf{Z}^T\mathbf{X})^{-1}\mathbf{Z}^T\mathbf{y}$ term in equation (3.8) is the IV estimate of the parameters. It is written as:

$$\hat{\boldsymbol{\theta}}_{IV} = (\mathbf{Z}^T\mathbf{X})^{-1}\mathbf{Z}^T\mathbf{y} \tag{3.9}$$

The $(\mathbf{Z}^T\mathbf{X})^{-1}\mathbf{Z}^T\boldsymbol{\varepsilon}$ term in equation (3.8) represents a potential bias in the estimate. The first property of the **Z** matrix, given in equation (3.2), ensures this bias goes to zero, asymptotically. Applying this property, equation (3.8) can be rewritten as:

$$\hat{\boldsymbol{\theta}} = (\mathbf{Z}^T\mathbf{X})^{-1}\mathbf{Z}^T\mathbf{y} = \hat{\boldsymbol{\theta}}_{IV} \tag{3.10}$$

Equation (3.10) gives an unbiased estimate of the ARMA parameters. [Ref. 1: pp. 192-193]:

Other least-squares methods avoid the bias inherent in ordinary least-squares but they are more complicated than the IV method to implement [Ref. 3: p. 119]. Although this thesis does not attempt an analysis of the IV method in the presence of noise, any practical system identification technique must deal with noise. Hence, the attraction of and the desire to use the IV method.

Equation (3.10) represents the block processing case. It assumes $N + L - 1$ output samples and $M + L$ input samples are available. These samples are used to calculate an estimate of the parameters. Samples beyond this range are not included in the estimation process. Block processing involves multiplication of two $L \times (N + M + 1)$ matrices to form a third matrix. Then this third matrix must be inverted. This is a computationally intensive process. In what follows, we present a sequential algorithm to compute $\hat{\theta}_{IV}$ which avoids matrix inversions.

B. SEQUENTIAL LEAST-SQUARES ESTIMATION USING INSTRUMENTAL VARIABLES

A sequential process for estimating the parameters of an unknown system requires fewer computations than a block process. In a manner similar to that presented in Hsia [Ref. 3: pp. 22-25] for the general least-squares case, the block IV estimation process described above can be converted into a sequential IV estimation process. Using the sequential process also allows the coefficients to be updated based on the new data that becomes available.

The derivation of the sequential estimation procedure consists of two parts. The first part is the derivation of an equation to update the data matrix, $\mathbf{Q}(m+1)$, based on the previous data matrix, $\mathbf{Q}(m)$, and the new data: $\hat{w}(m)$, $y(m)$, and $u(m+1)$ where m represents the iteration. The second part involves developing an equation for updating the estimate of the parameters, $\hat{\theta}_{IV}(m+1)$, based on the previous estimate, $\hat{\theta}_{IV}(m)$, the previous data matrix, $\mathbf{Q}(m)$, and the new data: $\hat{w}(m)$, $y(m)$, and $u(m+1)$.

Define the data matrix $\mathbf{Q}(m)$ to be:

$$\mathbf{Q}(m) = [\mathbf{Z}_m^T \mathbf{X}_m]^{-1} \tag{3.11}$$

where $\mathbf{Z}_m$ is given by equation (3.5) and $\mathbf{X}_m$ is given by equation (2.8). The property of equation (3.3) assures that $\mathbf{Q}$ exists. Note that $\mathbf{Q}(m)$ includes output data available through m and input data available through $m+1$. Since both $\mathbf{Z}_m$ and $\mathbf{X}_m$ are $m \times (N+M)$ matrices, $\mathbf{Q}(m)$ will be a $(N+M) \times (N+M)$ matrix. As the number of rows of $\mathbf{Z}$ and $\mathbf{X}$ increase to accommodate the increasing numbers of data points, the size of $\mathbf{Q}$ will remain the same. At the next sample time, i.e., at $m+1$, the data matrix becomes:

$$\mathbf{Q}(m+1) = [\mathbf{Z}_{m+1}^T \mathbf{X}_{m+1}]^{-1} \tag{3.12}$$

where the data matrices at $m+1$ are given by:

$$\mathbf{Z}_{m+1} = \begin{bmatrix} \mathbf{Z}_m \\ \cdots \\ \mathbf{z}^T(m+1) \end{bmatrix} \tag{3.13}$$

$$\mathbf{X}_{m+1} = \begin{bmatrix} \mathbf{X}_m \\ \cdots \\ \mathbf{x}^T(m+1) \end{bmatrix} \tag{3.14}$$

and $\mathbf{z}^T(m+1)$ and $\mathbf{x}^T(m+1)$ are vectors which contain the most recent data values. Substituting equations (3.13) and (3.14) into equation (3.12) and expanding, results in:

$$\mathbf{Q}(m+1) = \left[[\mathbf{Z}_m^T \quad \mathbf{z}(m+1)] \begin{bmatrix} \mathbf{X}_m \\ \cdots \\ \mathbf{x}^T(m+1) \end{bmatrix} \right]^{-1} \tag{3.15}$$

Expanding further yields:

$$\mathbf{Q}(m+1) = [\mathbf{Z}_m^T\mathbf{X}_m + \mathbf{z}(m+1)\mathbf{x}^T(m+1)]^{-1} \tag{3.16}$$

In equation (3.16) we see that two terms make up the new data matrix. The $\mathbf{Z}_m^T\mathbf{X}_m$ term is all the data that was available through time m. The $\mathbf{z}(m+1)\mathbf{x}^T(m+1)$ term contains the new data. To perform the inversion, let $\mathbf{A} = \mathbf{Z}_m^T\mathbf{X}_m$, $\mathbf{B} = \mathbf{z}(m+1)$, $\mathbf{C} = 1$ and $\mathbf{D} = \mathbf{x}^T(m+1)$. Then by the matrix inversion lemma:

$$\mathbf{Q}(m+1) = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1} \tag{3.17}$$

Substituting the appropriate expressions for **A**, **B**, **C**, and **D** back into equation (3.17) yields the equation:

$$\begin{aligned} \mathbf{Q}(m+1) = (\mathbf{Z}_m^T\mathbf{X}_m)^{-1} &- (\mathbf{Z}_m^T\mathbf{X}_m)^{-1}\mathbf{z}(m+1) \\ &\bullet\ [1 + \mathbf{x}^T(m+1)(\mathbf{Z}_m^T\mathbf{X}_m)^{-1}\mathbf{z}(m+1)]^{-1} \\ &\bullet\ \mathbf{x}^T(m+1)(\mathbf{Z}_m^T\mathbf{X}_m)^{-1} \end{aligned} \tag{3.18}$$

Substituting $\mathbf{Q}(m)$ for $(\mathbf{Z}_m^T\mathbf{X}_m)^{-1}$ reduces equation (3.18) to:

$$\begin{aligned} \mathbf{Q}(m+1) = \mathbf{Q}(m) - \mathbf{Q}(m)\mathbf{z}(m+1)[1 + \mathbf{x}^T(m+1)\mathbf{Q}(m)\mathbf{z}(m+1)]^{-1} \\ \bullet\ \mathbf{x}^T(m+1)\mathbf{Q}(m) \end{aligned} \tag{3.19}$$

This completes the first step of the derivation. Equation (3.19) expresses $\mathbf{Q}$ at time $m+1$ in terms of the old $\mathbf{Q}$ and the new data. The term in the brackets is a scalar. Computational intensity has been reduced because a large matrix does not have to be generated and its inverse does not have to be calculated.

Continuing with the derivation, the estimate $\hat{\theta}_{IV}$ for data available through m can be written as:

$$\hat{\theta}_{IV}(m) = (\mathbf{Z}_m^T \mathbf{X}_m)^{-1} \mathbf{Z}_m^T \mathbf{y}_m \tag{3.20}$$

The estimate $\hat{\theta}_{IV}$ for data available through $m+1$ can be written as:

$$\hat{\theta}_{IV}(m+1) = (\mathbf{Z}_{m+1}^T \mathbf{X}_{m+1})^{-1} \mathbf{Z}_{m+1}^T \mathbf{y}_{m+1} \tag{3.21}$$

Substituting equation (3.12) into equation (3.21) results in an expression for the estimate of the parameters in terms of the new data matrix and all the available data given by:

$$\hat{\theta}_{IV}(m+1) = \mathbf{Q}(m+1)\mathbf{Z}_{m+1}^T \mathbf{y}_{m+1} \tag{3.22}$$

$$\hat{\theta}_{IV}(m+1) = \mathbf{Q}(m+1)[\mathbf{Z}_m^T \quad \mathbf{z}(m+1)] \begin{bmatrix} \mathbf{y}_m \\ \cdots \\ y(m+1) \end{bmatrix} \tag{3.23}$$

$$\hat{\theta}_{IV}(m+1) = \mathbf{Q}(m+1)[\mathbf{Z}_m^T \mathbf{y}_m + \mathbf{z}(m+1)y(m+1)] \tag{3.24}$$

Substituting for $\mathbf{Q}(m+1)$ from equation (3.19) and expanding results in:

$$\begin{aligned} \hat{\theta}_{IV}(m+1) = {} & \mathbf{Q}(m)\mathbf{Z}_m^T \mathbf{y}_m \\ & - \mathbf{Q}(m)\mathbf{z}(m+1)[1 + \mathbf{x}^T(m+1)\mathbf{Q}(m)\mathbf{z}(m+1)]^{-1} x^T(m+1)\mathbf{Q}(m)\mathbf{Z}_m^T \mathbf{y}_m \\ & + \mathbf{Q}(m)\mathbf{z}(m+1)y(m+1) \\ & - \mathbf{Q}(m)\mathbf{z}(m+1)[1 + x^T(m+1)\mathbf{Q}(m)\mathbf{z}(m+1)]^{-1} \\ & \bullet\ \mathbf{x}^T(m+1)\mathbf{Q}(m)\mathbf{z}(m+1)y(m+1) \end{aligned} \tag{3.25}$$

Although somewhat lengthy, this equation has the desired form. To simplify it, its last two terms can be arranged into the form:

$$\begin{aligned} & \mathbf{Q}(m)\mathbf{z}(m+1)\{1 - [1 + \mathbf{x}^T(m+1)\mathbf{Q}(m)\mathbf{z}(m+1)]^{-1}\mathbf{x}^T(m+1)\mathbf{Q}(m)\mathbf{z}(m+1)\} \\ & \bullet\ y(m+1) \end{aligned} \tag{3.26}$$

The terms within the braces can be thought of as the result of a previous application of the matrix inversion lemma with $A^{-1} = 1$, $B = 1$, $C^{-1} = 1$ and $D = x^T(m+1)Q(m)z(m+1)$. Reversing the lemma results in:

$$Q(m)z(m+1)[1 + x^T(m+1)Q(m)z(m+1)]^{-1}y(m+1) \tag{3.27}$$

Replacing the last two terms in equation (3.25) with this result gives us:

$$\begin{aligned}\hat{\theta}_{IV}(m+1) = Q(m)Z_m^T y_m - Q(m)z(m+1)[1 + x^T(m+1)Q(m)z(m+1)]^{-1} \\ \bullet\ x^T(m+1)Q(m)Z_m^T y_m + Q(m)z(m+1) \\ \bullet\ [1 + x^T(m+1)Q(m)z(m+1)]^{-1}y(m+1)\end{aligned} \tag{3.28}$$

Factoring $Q(m)z(m+1)$ and $[1 + x^T(m+1)Q(m)z(m+1)]^{-1}$ from the last two terms reduces equation (3.28) further to:

$$\begin{aligned}\hat{\theta}_{IV}(m+1) = Q(m)Z_m^T y_m + Q(m)z(m+1)[1 + x^T(m+1)Q(m)z(m+1)]^{-1} \\ \bullet\ [y(m+1) - x^T(m+1)Q(m)Z_m^T y_m]\end{aligned} \tag{3.29}$$

Substituting equation (3.11) into equation (3.20) and then equation (3.20) into equation (3.29) yields the final form for the update of the estimate of the parameters:

$$\begin{aligned}\hat{\theta}_{IV}(m+1) = \hat{\theta}_{IV}(m) + Q(m)z(m+1) \\ \bullet\ [1 + x^T(m+1)Q(m)z(m+1)]^{-1}[y(m+1) - x^T(m+1)\hat{\theta}_{IV}(m)]\end{aligned} \tag{3.30}$$

This is the desired result for updating the estimate of the parameters. Note that like equation (3.19), the matrix inversion of equation (3.21) has been reduced to inversion of a scalar. Equation (3.30) describes the update of $\hat{\theta}_{IV}(m+1)$ in terms of the previous estimate of the parameters, $\hat{\theta}_{IV}(m)$, the previous data matrix, $Q(m)$, and the new data: $\hat{w}(m)$, $y(m)$, and $u(m+1)$.

C. TESTING THE SEQUENTIAL INSTRUMENTAL VARIABLE ALGORITHM

Equations (3.19) and (3.30) above comprise the sequential IV algorithm. Several tests of this algorithm were made using second and third-order filters as unknown systems. Tests were run via computer simulation using the filters to generate the output data. A Gaussian random process with zero mean and unit variance was used as the input. The input was produced by IMSL subroutine GGNML. Graphs were created using DISSPLA. Table 1 on page 17 shows pole and zero locations as well as numerator and denominator parameters for the test filters.

Table 1. TEST SYSTEMS FOR THE IV MODELING METHOD

TEST FILTER	LOCATIONS OF POLES	LOCATIONS OF ZEROS	AR PARAMETERS	MA PARAMETERS
T2	0.445+j0.228 0.445 - j0.228	0.4+j1.273 0.4 - j1.273	1.0 -0.89 0.25	0.5 -0.4 0.89
T2N	0.445+j0.228 0.445 -j0.228	0.4+j0.8 0.4 -j0.8	1.0 -0.89 0.25	1.0 -0.80 0.80
T3	0.6605 0.6647+j0.502 0.6647-j0.502	-1.0 -1.0 -1.0	1.0 -1.99 1.57 -0.458	0.0154 0.0462 0.0462 0.0154

Results of the tests are shown in graphical form in Figure 4 on page 18, Figure 5 on page 19, and Figure 6 on page 20. Dashed lines indicate the true values of the parameters. Solid lines are the IV method's estimates.

For both second-order test cases shown, the algorithm converged quickly and produced accurate results. For the third-order test case, convergence took longer but the values were accurate. A third-order system is more complex than a second-order system, so conceivably it would require more iterations to converge. The number of iterations required is of the same order as the method of ordinary least-squares.

Table 2 on page 21 contains the IV algorithm's best estimates of the parameters and the number of iterations required to converge to those estimates. It also shows the absolute and percent errors from the true parameters.

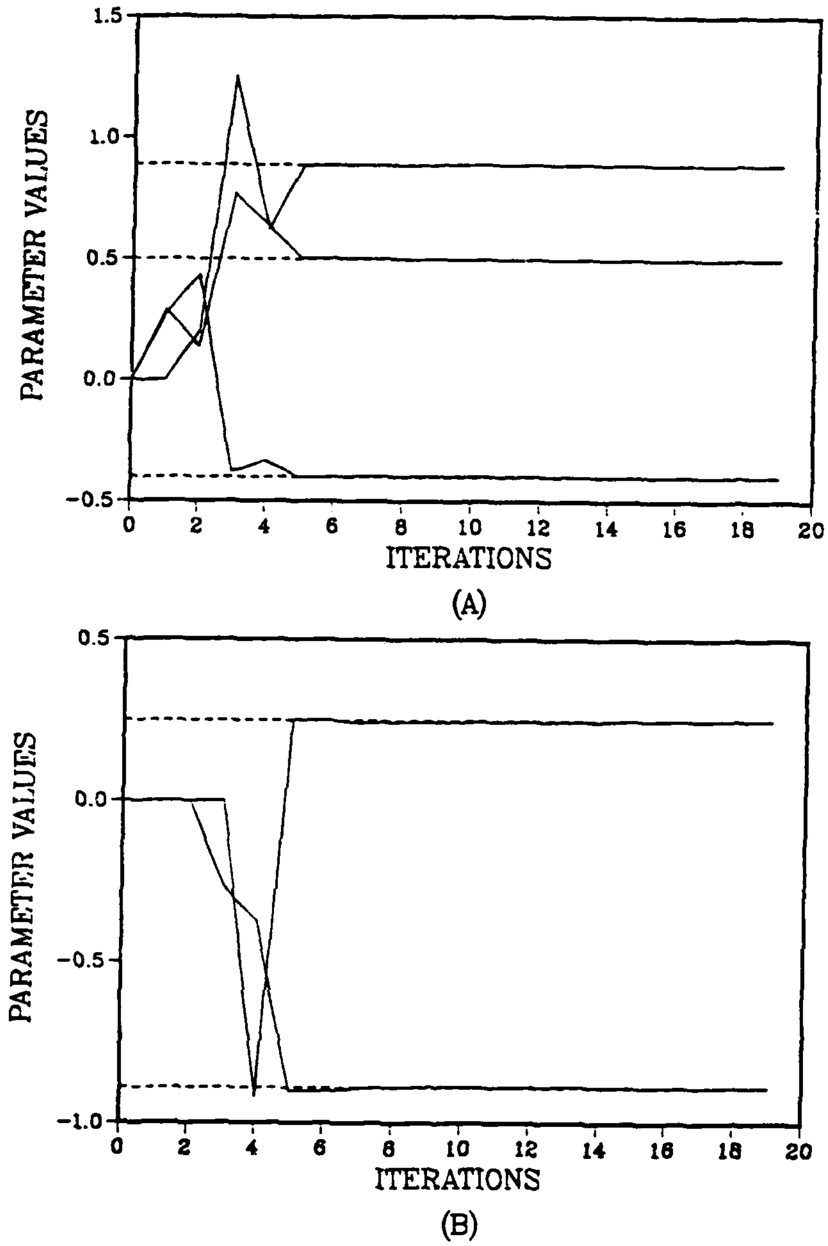
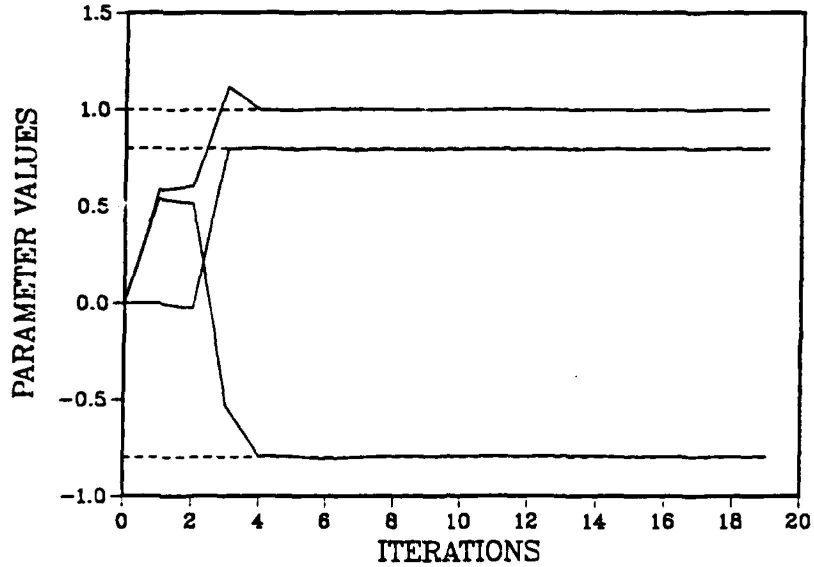
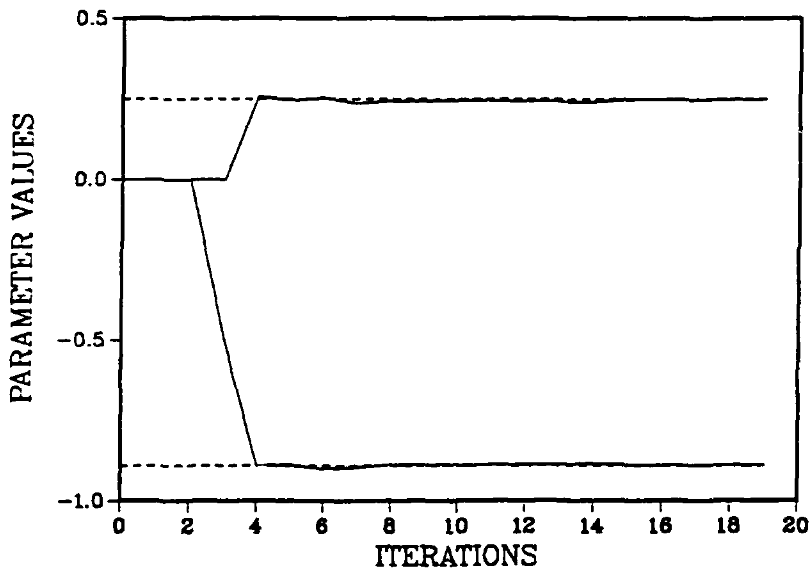


Figure 4. **Second-order test case T2. (A) MA parameters. (B) AR parameters.**



(A)



(B)

Figure 5. Second-order test case T2N. (A) MA parameters. (B) AR parameters.

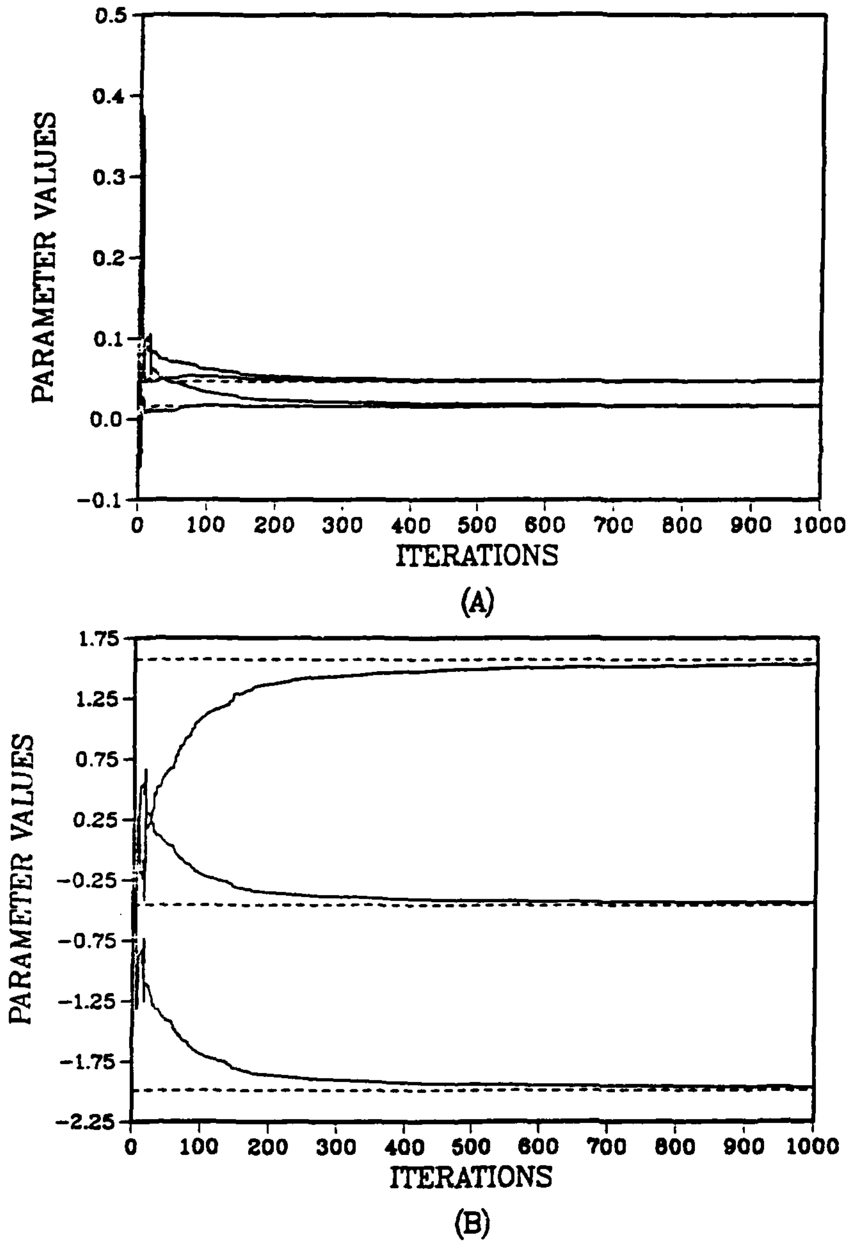


Figure 6. Third-order test case T3. (A) MA parameters. (B) AR parameters.

Table 2. COEFFICIENT ESTIMATES BY THE IV MODELING METHOD.

TEST FILTER	PARAMETER ESTIMATE	ABSOLUTE ERROR	PERCENT ERROR	ITERATIONS
T2	0.500	0.0	0.0	10
	-0.396	+0.004	0.10	
	0.888	-0.002	0.22	
	1.000	0.0	0.0	
	-0.888	+0.002	0.22	
	0.244	-0.006	2.40	
T2N	1.000	0.0	0.0	10
	-0.794	+0.006	0.750	
	0.794	-0.006	0.750	
	1.000	0.0	0.0	
	-0.887	+0.003	0.34	
	0.243	-0.007	2.80	
T3	0.0154	0.0	0.0	1000
	0.0466	+0.0004	0.87	
	0.0475	+0.0013	2.81	
	0.0169	+0.0015	9.74	
	1.000	0.0	0.0	
	-1.96	-0.03	3.0	
	1.532	-0.040	2.01	
	0.4379	-0.0204	4.45	

IV. SYSTEM IDENTIFICATION USING AN ITERATIVE MULTICHANNEL APPROACH

A. INTRODUCTION

This chapter presents an alternate system identification method, the iterative multichannel approach. This approach differs from the IV method and the method of ordinary least-squares presented in the preceding chapters in that it processes the input and output data from the unknown system in separate channels. In its block processing form one advantage over the IV and ordinary least-squares methods is a reduction in the sizes of the data matrices. As a result, the computational complexity of the multichannel algorithm is on the order of $M^2 + N^2$, where M is the order of the MA part and N is the order of the AR part. In contrast, the block IV and ordinary least-squares methods require computations on the order of $(M + N)^2$.

B. PREVIOUS MULTICHANNEL METHODS

Whittle [Ref. 6: pp. 129-130] was the first to develop a multichannel solution for the ARMA modeling problem. He sought to extend the recursive Durbin solution for estimating the parameters of a single variable autoregressive process to a multivariable autoregressive process. He discovered that to do this he would have to fit the data to two autoregressive processes simultaneously. One of the autoregressions would use present data samples to predict the value of the data one time step in the future. This is called forward prediction. The second autoregression would use present data samples to predict the value of the data at the previous time instant and is referred to as backward prediction. Sometime during this research, Whittle determined that if the input was derived from a MA scheme, making the process ARMA, then the solution would remain the same provided the correlations of the input used in the parameter estimation process had shifts greater than the MA scheme. Whittle's use of the two separate and simultaneous autoregressions to model an ARMA process can be thought of as a multichannel modeling approach.

Further work in the area of multichannel ARMA modeling was conducted by Perry and Parker [Ref. 7: pp. 509-510]. They started out with the ARMA problem formulation discussed in Chapter 2. Using the method of ordinary least-squares to minimize the mean square error, they found the solution for the estimate of the parameters to be the Wiener solution given by:

$$\begin{bmatrix} \mathbf{R}_{yy} & \mathbf{R}_{yu'} \\ \mathbf{R}_{u'y} & \mathbf{R}_{u'u'} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a}' \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{yy} \\ \mathbf{r}_{yu'} \end{bmatrix} \tag{4.1}$$

In equation (4.1) $\mathbf{R}_{yy}$ is a matrix of autocorrelations of the past outputs, $\mathbf{R}_{u'u'}$ is a matrix of autocorrelations of the inputs, $\mathbf{R}_{yu'}$ and $\mathbf{R}_{u'y}$ are crosscorrelations of the input and output data, $\mathbf{b}$ is a vector of AR parameters, and $\mathbf{a}'$ is a vector of MA parameters. In addition, $\mathbf{r}_{yy}$ is a vector of autocorrelations of past output data with the current output and $\mathbf{r}_{yu'}$ is a vector of crosscorrelations of input data with the current output. By assuming the first MA parameter, a'_0 was known, they were able to treat it as a gain and factor it out of all the other MA parameters. This allowed them [Ref. 7: pp. 509-510] to extract the $(N+1)^{st}$ row and column of equation (4.1) and rewrite the solution in the form:

$$\begin{bmatrix} \mathbf{R}_{yy} & \mathbf{R}_{yu} \\ \mathbf{R}_{uy} & \mathbf{R}_{uu} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{yy} \\ \mathbf{r}_{yu} \end{bmatrix} - \begin{bmatrix} \mathbf{r}_{yu} \\ \mathbf{r}_{uu} \end{bmatrix} \tag{4.2}$$

In equation (4.2), $\mathbf{a}'$ has been rewritten as $\mathbf{a}$ to indicate that the a'_0 term has been factored out of all of the MA parameters. Reasoning that equation (4.2), the ARMA solution, was a generalization of the AR solution, they figured that it must have a recursive solution consisting of some combination of the Levinson-Durbin algorithm, a recursive solution for the AR problem. They then determined equation (4.2) was in a form similar to Whittle's formulation of the problem. So they reasoned that they could use a form of Whittle's solution to solve the ARMA modeling problem. Like Whittle, their solution consists of a forward and a backward autoregression. It uses two coupled lattice filters to process the input and output data. Off diagonal elements of the lattice coefficient matrices specify the coupling points of the two lattices.

C. ITERATIVE APPROACH TO MULTICHANNEL ARMA MODELING

This thesis proposes another solution to the ARMA modeling problem using the multichannel approach. It is an iterative approach with no direct coupling of the two channels. However, note that there is an implicit coupling in the sense that the ARMA system output samples $y(n)$ are a function of the present and past input samples $u(n)$, and the past output samples. This is shown in equation (2.1). The approach proposed uses two separate finite-impulse-response (FIR) filters to estimate the unknown system. One filter estimates the AR part of the unknown system. The second estimates the MA part of the unknown system. Figure 7 on page 25 shows the structure of this approach.

The derivation of the equations for this approach follows the method of ordinary least-squares.

From Figure 7 on page 25, the value of the signal $Y(z)$ is seen to be $\frac{\mathbf{A}(z)}{\mathbf{B}(z)} U(z)$. When this signal passes through $\mathbf{C}(z)$, if $\mathbf{C}(z)$ is close to $\mathbf{B}(z)$, the resulting signal $X(z)$ is approximately $\mathbf{A}(z)U(z)$. Also from Figure 7 on page 25, the value of $Z(z)$ is seen to be $\mathbf{A}(z)U(z)$ provided $\mathbf{D}(z)$ is close to $\mathbf{A}(z)$. The difference of these two signals forms the error which we seek to minimize by the method of ordinary least-squares. In minimizing the error we seek to drive $\mathbf{D}(z)$ and $\mathbf{C}(z)$ as close as possible to $\mathbf{A}(z)$ and $\mathbf{B}(z)$, respectively.

Referring to Figure 7 on page 25, signals $z(n)$ and $x(n)$ are defined as the outputs from two FIR filters and are given by the equations:

$$z(n) = d_0u(n) + d_1u(n-1) + d_2u(n-2) + \cdots + d_Mu(n-M) = \mathbf{u}^T(n)\mathbf{d} \tag{4.3}$$

$$x(n) = y(n) + c_1y(n-1) + c_2y(n-2) + \cdots + c_Ny(n-N) = \mathbf{y}^T(n)\mathbf{c} \tag{4.4}$$

where the vectors $\mathbf{d}$ and $\mathbf{c}$ represent the systems $\mathbf{D}(z)$ and $\mathbf{C}(z)$, respectively. The vectors $\mathbf{d}$, $\mathbf{u}(n)$, $\mathbf{c}$, and $\mathbf{y}(n)$ are given by the following equations:

$$\mathbf{d} = [d_0 \quad d_1 \quad d_2 \quad \ldots \quad d_M]^T \tag{4.5}$$

$$\mathbf{u}(n) = [u(n) \quad u(n-1) \quad u(n-2) \quad \ldots \quad u(n-M)]^T \tag{4.6}$$

$$\mathbf{c} = [1 \quad c_1 \quad c_2 \quad \ldots c_N]^T \tag{4.7}$$

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

The $\mathbf{d}$ parameters are estimates of the MA portion of the ARMA process. The $\mathbf{c}$ parameters approximate its AR portion. The vector $\mathbf{u}(n)$ is the input data vector of length M, the order of the MA part, and $\mathbf{y}(n)$ is the output data vector equal of length N, the order of the AR part.

Forming the error between x and z results in:

$$e(n) = z(n) - x(n) = \mathbf{u}^T(n)\mathbf{d} - \mathbf{y}^T(n)\mathbf{c} \tag{4.9}$$

The least-squares performance criterion is:

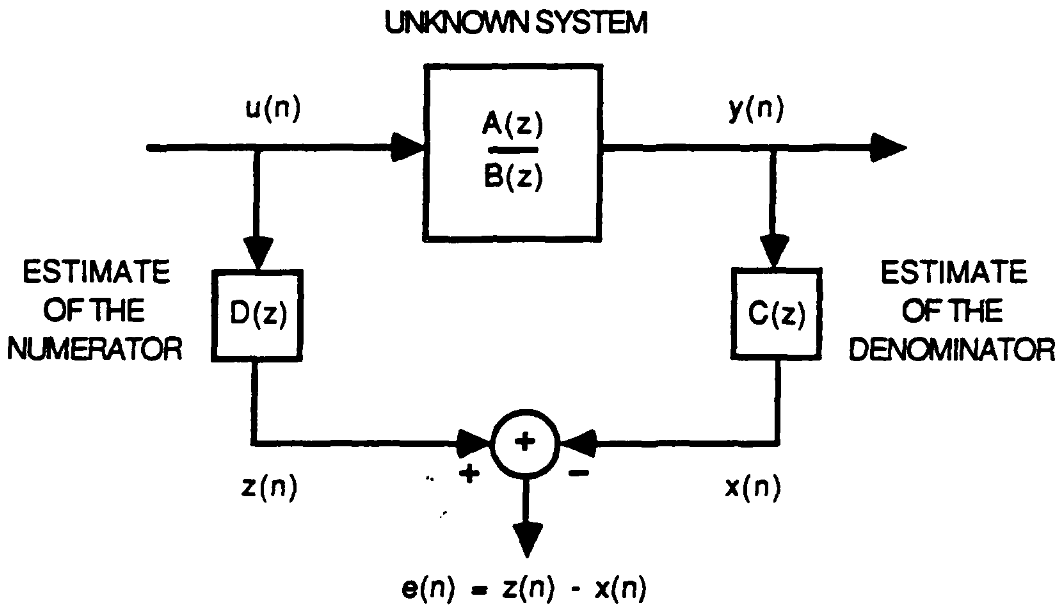


Figure 7. Multichannel modeling process

$$J = \sum_{n=1}^{L} e^2(n) = \sum_{n=1}^{L} [z(n) - x(n)]^2 \tag{4.10}$$

Substituting for $e(n)$ from equation (4.9), we can write equation (4.10) in vector form as:

$$J = \sum_{n=1}^{L} (\mathbf{u}^T\mathbf{d} - \mathbf{y}^T\mathbf{c})^2 \tag{4.11}$$

where we have dropped the time index, n, for convenience. Expanding equation (4.11) results in:

$$J = \sum_{n=1}^{L} \mathbf{d}^T\mathbf{u}\mathbf{u}^T\mathbf{d} + \mathbf{c}^T\mathbf{y}\mathbf{y}^T\mathbf{c} - 2\mathbf{d}^T\mathbf{u}\mathbf{y}^T\mathbf{c} \tag{4.12}$$

We notice that the performance criterion is a function of both **d** and **c**. Minimizing the performance criterion by differentiating with respect to the vector **c** and equating the results to zero yields:

$$\frac{\partial J}{\partial \mathbf{c}} = 0 = 0 + 2\sum_{n=1}^{L}(\mathbf{y}\mathbf{y}^T)\mathbf{c} - 2\sum_{n=1}^{L}(\mathbf{y}\mathbf{u}^T)\mathbf{d} \tag{4.13}$$

Similarly, differentiating the performance criterion with respect to the vector **d** and equating the result to zero yields:

$$\frac{\partial J}{\partial \mathbf{d}} = 0 = 0 + 2\sum_{n=1}^{L}(\mathbf{u}\mathbf{u}^T)\mathbf{d} - 2\sum_{n=1}^{L}(\mathbf{u}\mathbf{y}^T)\mathbf{c} \tag{4.14}$$

Solving equation (4.13) for **c** and equation (4.14) for **d** results in two equations for estimating the AR and MA parameters of the unknown system given by:

$$\mathbf{c} = \mathbf{R}_{yy}^{-1}\mathbf{R}_{yu}\mathbf{d} \tag{4.15}$$

and

$$\mathbf{d} = \mathbf{R}_{uu}^{-1}\mathbf{R}_{uy}\mathbf{c} \tag{4.16}$$

where

$$\mathbf{R}_{uu} = \sum_{n=1}^{L}\mathbf{u}\mathbf{u}^T = \begin{bmatrix} r_{uu}(0) & r_{uu}(1) & . & . & r_{uu}(N) \\ r_{uu}(1) & r_{uu}(0) & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ r_{uu}(N) & . & . & . & r_{uu}(0) \end{bmatrix} \tag{4.17}$$

is an estimate of the input autocorrelation matrix. The elements of $\mathbf{R}_{uu}$ are computed using the unbiased method as follows:

$$r_{uu}(l) = \frac{1}{L-l}\sum_{j=0}^{L-l}u(j)u(j-l) \quad \text{for } l = 0,\ 1,\ 2,\ \ldots\ , M \tag{4.18}$$

Matrices $\mathbf{R}_{yy}$, $\mathbf{R}_{uy}$, and $\mathbf{R}_{yu}$ appearing in equations (4.15) and (4.16) have structures identical to equation (4.17), where $\mathbf{R}_{yy}$ is the estimate of the output autocorrelation matrix, and $\mathbf{R}_{uy}$ and $\mathbf{R}_{yu}$ are estimates of the crosscorrelation matrices. Note that $\mathbf{R}_{yu} = \mathbf{R}_{uy}^T$. The elements of these matrices; r_{yy}, r_{uy} and r_{yu}, are computed as follows:

$$r_{yy}(l) = \frac{1}{L-l} \sum_{j=0}^{L-l} y(j)y(j-l) \text{ for } l = 0,\ 1,\ 2,\ \ldots,\ N \tag{4.19}$$

$$r_{uy}(l) = \frac{1}{L-l} \sum_{j=0}^{L-l} u(j)y(j-l) \text{ for } l = 0,\ 1,\ 2,\ \ldots,\ M \tag{4.20}$$

and

$$r_{yu}(l) = \frac{1}{L-l} \sum_{j=0}^{L-l} y(j)u(j-l) \text{ for } l = 0,\ 1,\ 2,\ \ldots,\ N \tag{4.21}$$

Up to this point, following the standard least-squares procedure has resulted in two dependent or coupled equations to solve for the parameters of an unknown system modeled as an ARMA process. How best to solve these equations? By iteration. The steps of the iterative process are to

- Calculate the correlation matrices and vectors from the available data.
- Initialize $\mathbf{c}$. Exploit the fact that the first parameter in $\mathbf{c}$ is, $c_0 = 1$.
- Solve for $\mathbf{d}$ from this initial $\mathbf{c}$.
- Solve for $\mathbf{c}$ from $\mathbf{d}$.
- Repeat beginning at the third step.

Here is a summary of the equations in proper order for implementing the algorithm:

- Compute $\mathbf{R}_{uu}$, $\mathbf{R}_{yy}$ and $\mathbf{R}_{uy}$ from equations (4.17) to (4.21). Note that $\mathbf{R}_{yu} = \mathbf{R}_{uy}^T$.
- Initialization:

$$\mathbf{c}^{(0)} = [1 \quad 0 \quad \ldots \quad 0]^T \tag{4.22}$$

- For $k = 0$ to K

$$\mathbf{d}^{(k+1)} = \mathbf{R}_{uu}^{-1}\mathbf{R}_{uy}\mathbf{c}^{(k)} \tag{4.23}$$

$$\mathbf{c}^{(k+1)} = \mathbf{R}_{yy}^{-1}\mathbf{R}_{yu}\mathbf{d}^{(k+1)} \tag{4.24}$$

where k is the iteration number.

This is a very simple algorithm. For the case where the AR and MA orders are equal, the correlation matrices are half the size of the block data matrices which must be generated and inverted in the IV algorithm.

1. Testing the Multichannel Iterative Algorithm

We tested the algorithm by computer simulation using second and third-order filters as unknown systems. Table 3 shows pole and zero locations as well as MA and AR parameters for the test filters. Data lengths of 500 data points were used to calculate the autocorrelation and crosscorrelation matrices. Besides the reported cases, we tested the algorithm on several other second, third and mixed-order cases. The performance of the algorithm in all cases that we tested was fairly uniform.

Table 3. TEST SYSTEMS FOR ITERATIVE MULTICHANNEL MODELING METHOD

TEST FILTER	LOCATION OF POLES	LOCATION OF ZEROS	AR PARAMETERS	MA PARAMETERS
T2	0.445 + j0.228 0.445 - j0.228	0.4 + j1.273 0.4 -j1.273	1.0 -0.89 0.25	0.5 -0.4 0.89
T2N	0.445 + j0.228 0.445 -j0.228	0.4 + j0.8 0.4 -j0.8	1.0 -0.89 0.25	1.0 -0.80 0.80
T3	0.6605 0.6647 + j0.5020 0.6647 -j0.5020	-1.0 -1.0 -1.0	1.0 -1.99 1.572 -0.4583	0.0154 0.0462 0.0462 0.0154

Results of the tests are shown in graphical form in Figure 8 on page 29, Figure 9 on page 30, and Figure 10 on page 31. Dashed lines indicate the true values of the parameters. Solid lines show the values the algorithm calculated.

Table 4 on page 32 contains the algorithm's best estimates of the parameters, along with the number of iterations required to converge to those estimates. It also shows the absolute and percent errors from the true parameters. For the second-order test cases, convergence to the true parameter values occurred within 20 iterations. The third-order test case took 14 iterations to converge to its steady-state values; however, these values were not the true parameter values.

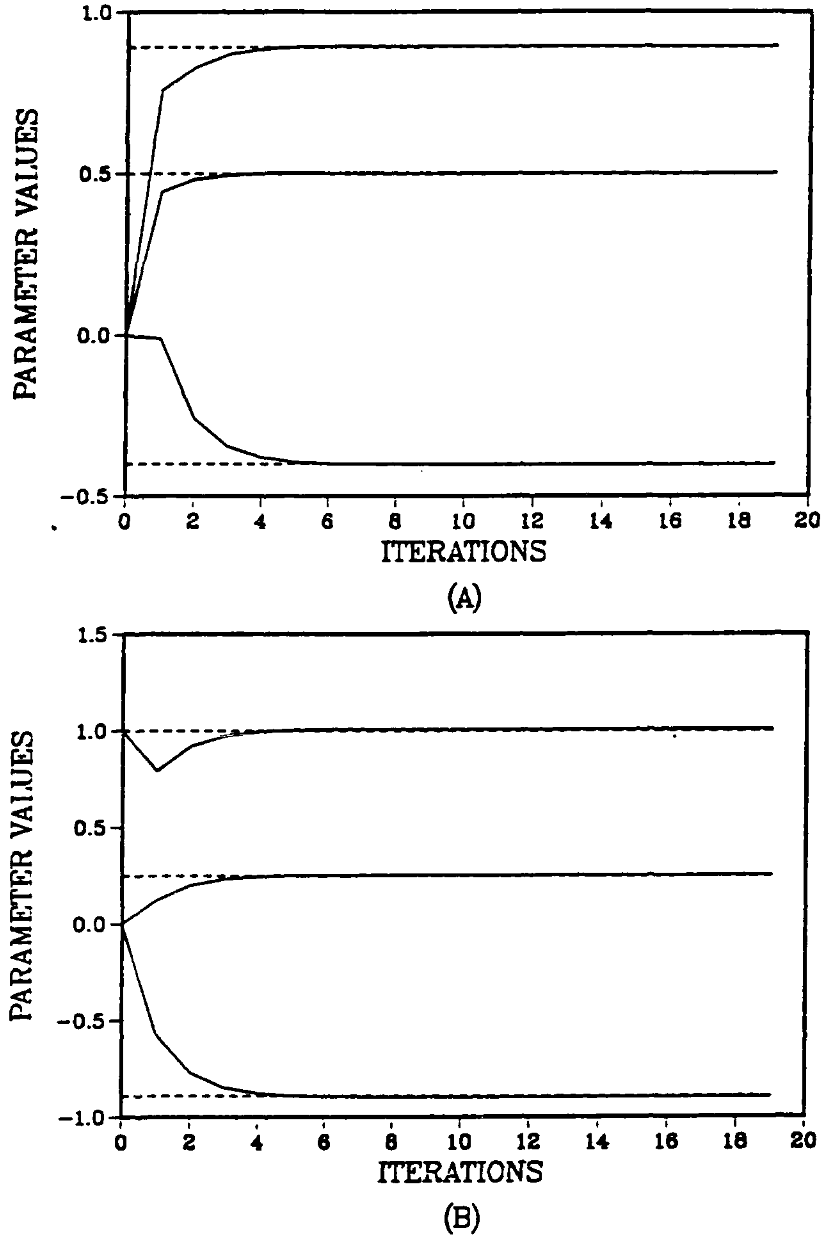
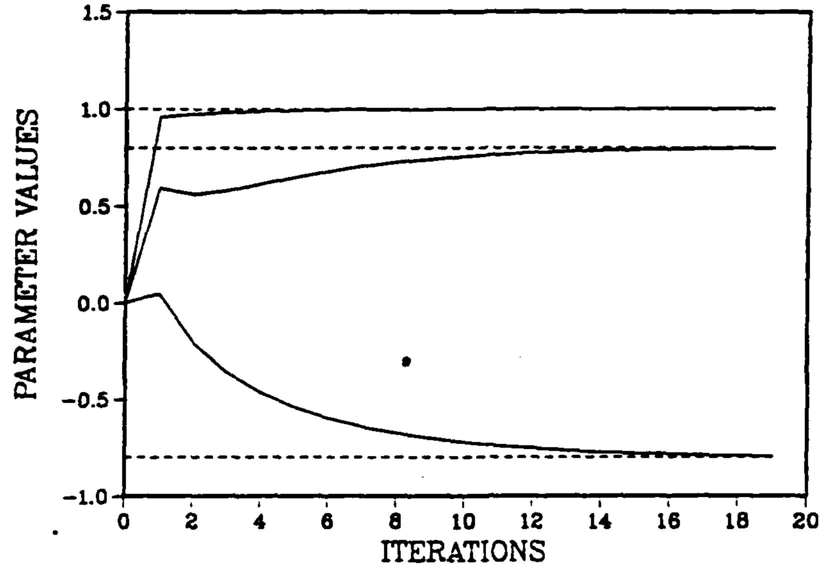
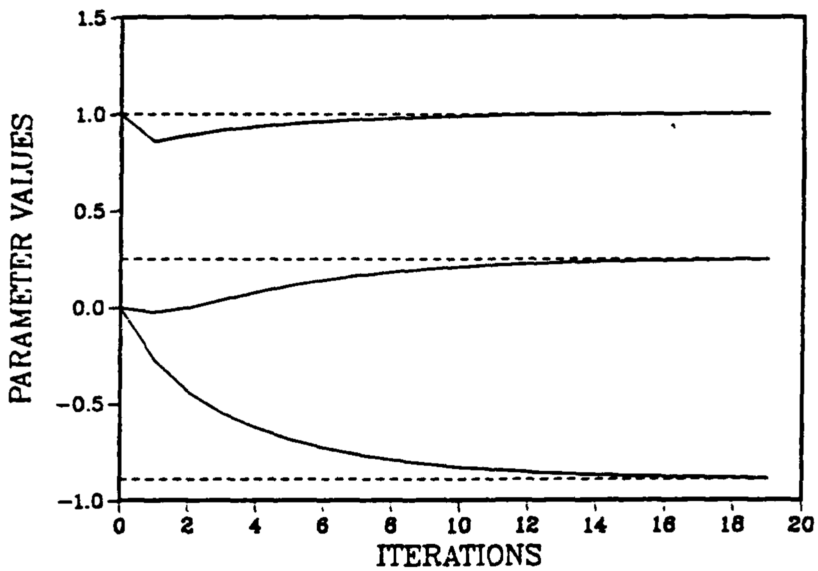


Figure 8. Second-order test case T2. (A) MA parameters. (B) AR parameters.



(A)



(B)

Figure 9. Second-order test case T2N. (A) MA parameters. (B) AR parameters.

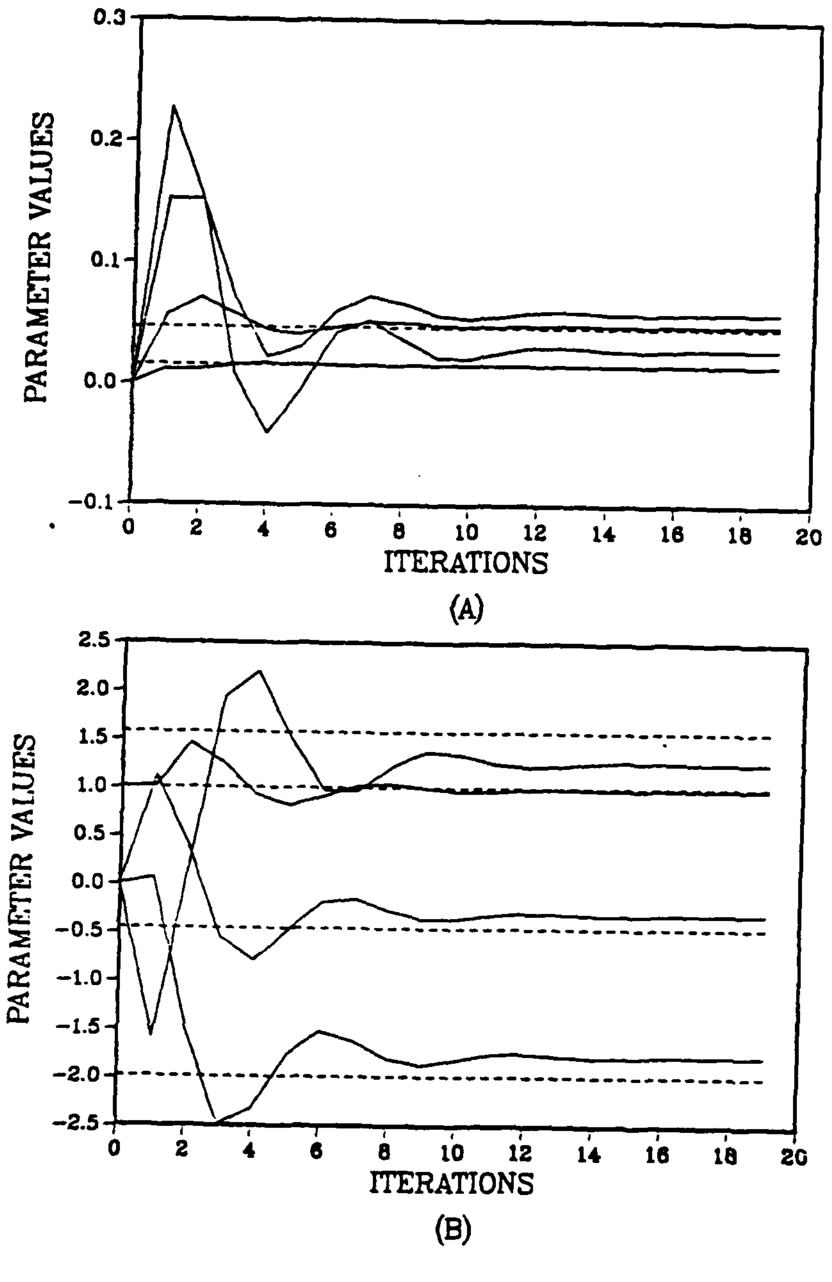


Figure 10. Third-order test case T3. (A) MA parameters. (B) AR parameters.

Table 4. **PARAMETER ESTIMATES BY THE ITERATIVE MULTICHANNEL ALGORITHM.**

TEST FILTER	PARAMETER ESTIMATE	ABSOLUTE ERROR	PERCENT ERROR	ITERATIONS
T2	0.500 -0.393 0.890 1.000 -0.889 0.247	0.0 -0.007 0.0 0.0 +0.001 -0.003	0.0 1.75 0.0 0.0 0.11 1.20	5
T2N	1.000 -0.794 0.798 1.000 -0.886 0.245	0.0 +0.006 -0.002 0.0 +0.004 -0.005	0.0 0.75 0.25 0.0 0.45 2.00	20
T3	0.0153 0.0487 0.0590 0.0287 0.99 -1.79 1.267 -0.3086	-0.0001 +0.0025 +0.0128 +0.0133 -0.01 +0.20 -0.305 +0.1497	0.65 5.41 27.71 86.36 1.0 10.05 19.40 32.66	14

2. Stopping Parameter

In tests of third-order systems, the parameter estimates swung through or close to the true coefficient values and converged to values somewhat removed from the true values. We developed a stopping parameter to flag the iteration when the estimates were closest to the true values. This occurs when the error is smallest. Referring to Figure 7 on page 25, if $\mathbf{D}(z)$ is equal to $\mathbf{A}(z)$ and $\mathbf{C}(z)$ is equal to $\mathbf{B}(z)$, x and z will both equal $\mathbf{A}(z)U(z)$. The error will be zero. The farther removed $\mathbf{D}(z)$ and $\mathbf{C}(z)$ are from $\mathbf{A}(z)$ and $\mathbf{B}(z)$, respectively, the larger the error becomes.

The stopping parameter is calculated from the difference of the z and x values at every iteration. After the parameter vectors $\mathbf{d}$ and $\mathbf{c}$ are estimated for a particular iteration, the stopping parameter is calculated from:

$$e_k(n) = z_k(n) - x_k(n) = \mathbf{y}_k^T\mathbf{c}_k - \mathbf{u}_k^T\mathbf{d}_k \tag{4.25}$$

where k is the iteration number and **d** , **u**, **c** and **y** are defined in equations (4.5) to (4.8). The parameter vectors **d** and **c** represent the systems $\mathbf{D}(z)$ and $\mathbf{C}(z)$, respectively.

Figure 11 on page 34 and Figure 12 on page 35 graph the stopping parameter (dotted line) along with the estimated and true values of the parameters. They show that when the stopping parameter is smallest, the parameters are closest to their true values. Table 5 shows the improvement in the estimates of the parameters resulting from choosing those values when the stopping parameter is smallest.

Table 5. PARAMETER ESTIMATES WHEN STOPPING PARAMETER IS SMALLEST.

TEST FILTER	PARAMETER ESTIMATE	ABSOLUTE ERROR	PERCENT ERROR	ITERATIONS
T3	0.0156	+0.0002	1.30	10
	0.0478	+0.0016	3.46	
	0.0536	+0.0074	16.02	
	0.0196	+0.0042	27.27	
	0.97	-0.03	3.0	
	-1.84	+0.15	7.54	
	1.375	-0.197	12.53	
	-0.3672	+0.0911	19.88	

The stopping parameter can be used in a real modeling situation because it comes from the data and the estimates of the parameters. Another measure of how well the estimates of the parameters fit the actual system is the norm of the coefficient error. This cannot be used in a real modeling situation however, because the values of the true parameters are not known. We calculated it for the test cases as a check on the appropriateness of using the stopping parameter. Figure 13 on page 36 and Figure 14 on page 37 graph the stopping parameter (dotted line) and the norm of the coefficient error for test cases T2 and T3. On both graphs the two curves correspond well. Both reach their minimum value at the same point, the point where the estimates of the parameters are closest to their true values.

3. Linear-prediction of the Denominator Coefficients

The iterative algorithm detailed in equations (4.22) to (4.24) starts by initializing the AR parameter estimates to:

$$\mathbf{c}^{(0)} = [1 \quad 0 \quad \dots \quad 0]^T \tag{4.26}$$

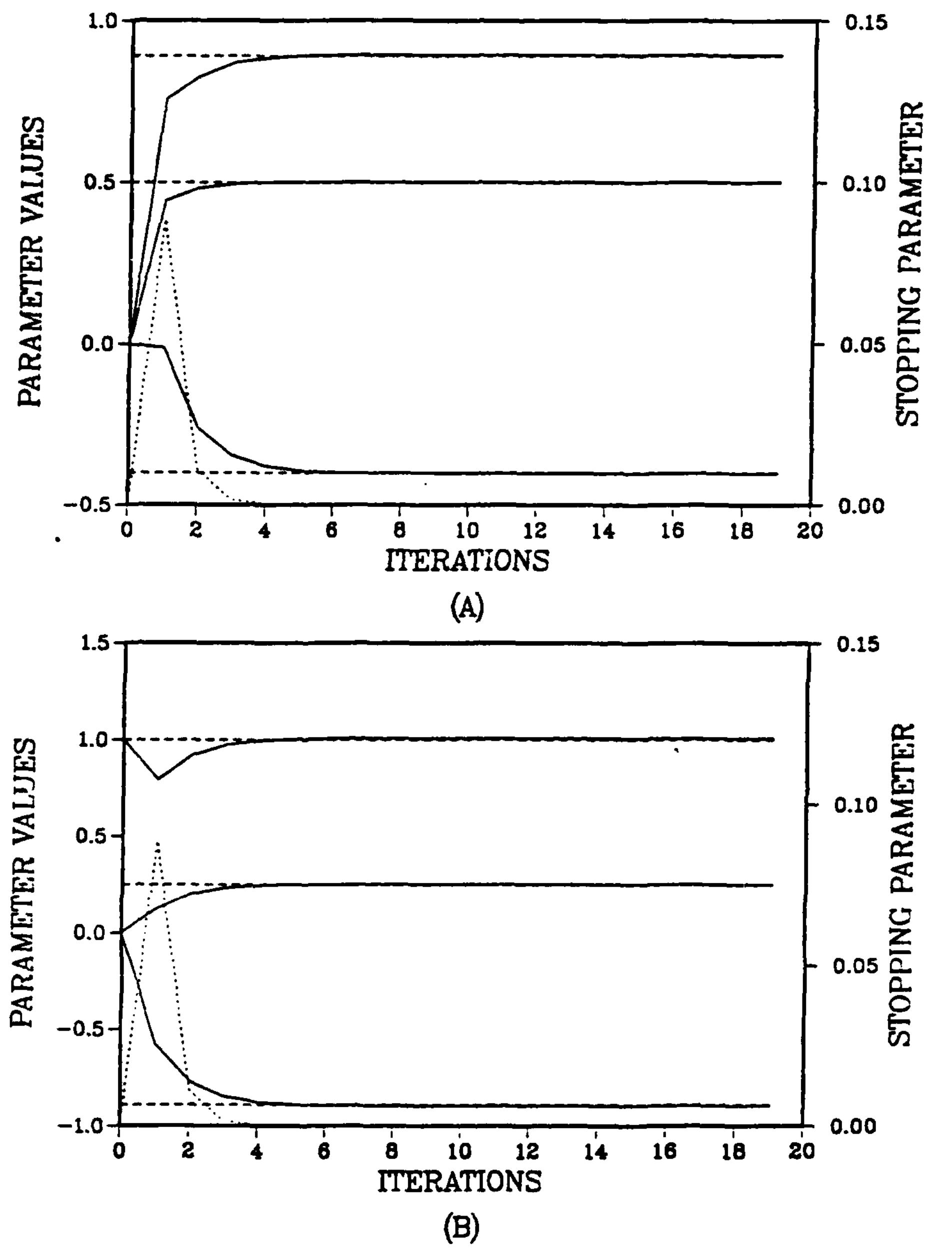
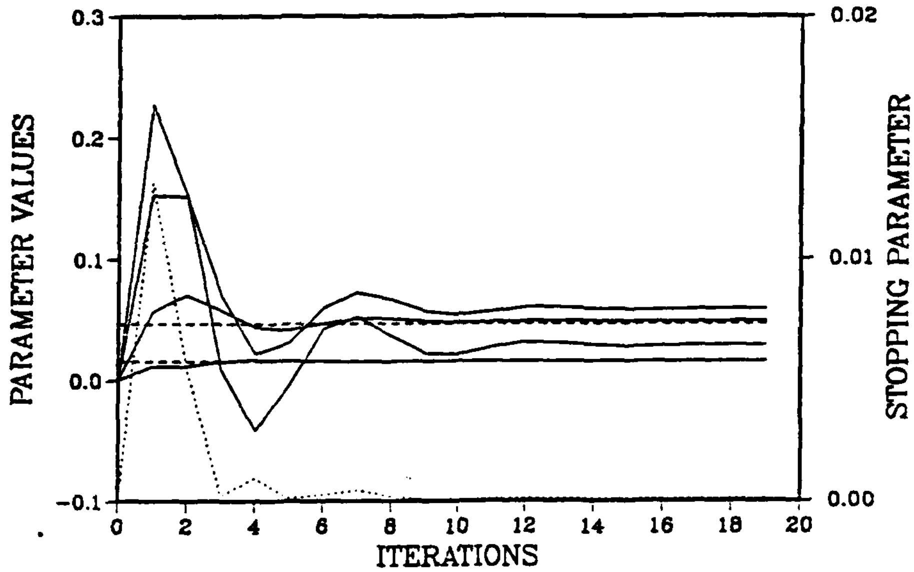
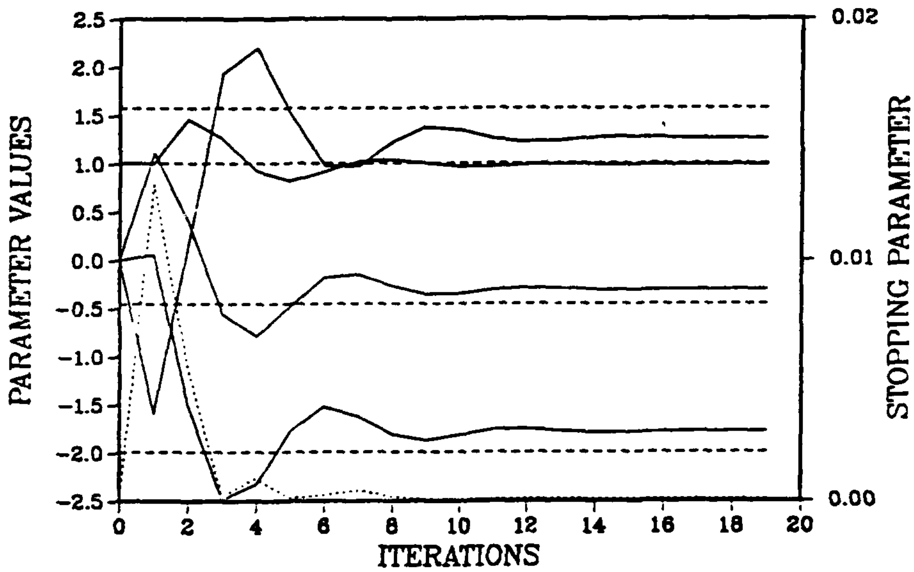


Figure 11. Stopping parameter example for test case T2.



(A)



(B)

Figure 12. Stopping parameter example for test case T3.

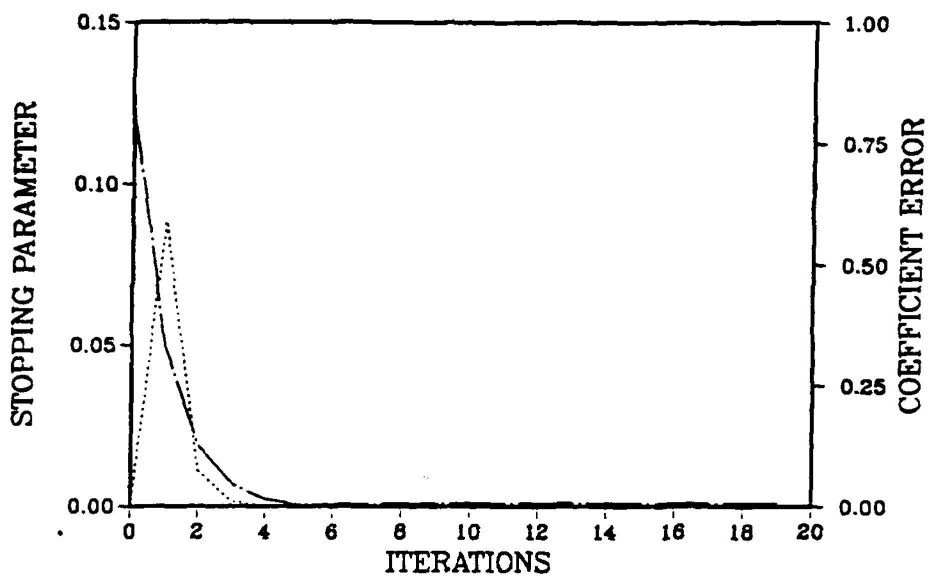


Figure 13. Stopping parameter and norm of the coefficient error for test case T2.

where c_0 is known to be 1 from the Z transform of the ARMA difference equation, equation (2.1). The Z transform is given by:

$$Y(z)[1 + c_1z^{-1} + c_2z^{-2} + \cdots] = U(z)[d_0 + d_1z^{-1} + d_2z^{-2} + \cdots] \tag{4.27}$$

$$\frac{Y(z)}{U(z)} = \frac{d_0 + d_1z^{-1} + d_2z^{-2} + \cdots}{1 + c_1z^{-1} + c_2z^{-2} + \cdots} \tag{4.28}$$

The initial estimates for the other AR parameters are zero. This can be far from their actual values. A closer estimate of the other AR parameters should result in quicker convergence for all parameters. A closer estimate of the AR parameters can be obtained by using linear-prediction techniques. Figure 15 on page 39 shows the approach used. In Figure 15 on page 39, $y(n)$ is the output from the unknown system. The system $C'(z)$, which is represented by the vector c', is the linear-prediction filter used to estimate $y(n)$. It uses the previous $n - N$ samples of the output to generate a current estimate. This is given by the equation:

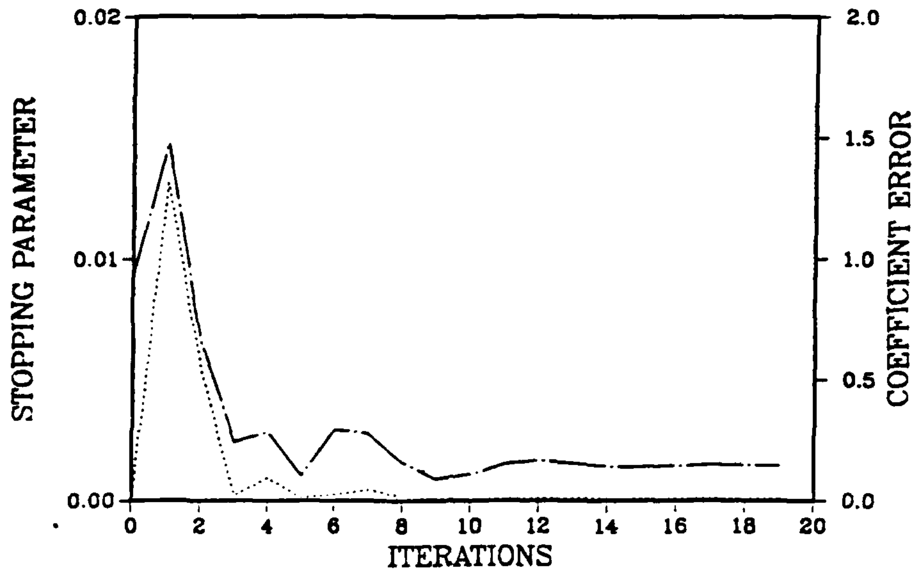


Figure 14. Stopping parameter and norm of the coefficient error for test case T3.

$$\hat{y}(n) = \mathbf{y}^T(n-1)\mathbf{c}' \tag{4.29}$$

where $\mathbf{c}'$ is a vector of the tap weights of the autoregressive process given by:

$$\mathbf{c}' = [c'_0 \quad c'_1 \quad \ldots c'_{n-N}]^T \tag{4.30}$$

and $\mathbf{y}(n-1)$ is a vector of the past output values given by:

$$\mathbf{y}(n-1) = [y(n-1) \quad y(n-2) \quad \ldots \quad y(n-N)]^T \tag{4.31}$$

Following least-squares techniques, we form the error between the estimate and the actual value of the output. The sum of the squares of the errors becomes the performance criterion. This is differentiated with respect to the tap weights and set equal to zero. Solving this for the tap weights results in the equation:

$$\mathbf{c}' = \mathbf{R}_{yy}^{-1}\mathbf{r}_{yy} \tag{4.32}$$

This is the standard Weiner filter solution [Ref. 8: p. 32]. It tells us that the best estimate of the AR parameters can be found from the correlations of the output data. The matrix $\mathbf{R}_{yy}$ is the autocorrelation matrix of the past outputs and $\mathbf{r}_{yy}$ is autocorrelation vector of the past outputs with the current output. In all cases tested, we did not achieve any significant improvement in the accuracy of the estimates of the parameters, or in the speed of convergence, using the straight linear prediction of equation (4.32).

A modification to this approach, which we refer to as modified linear-prediction, uses correlation lags beginning on the order of the MA portion of the ARMA process. For example, correlations for calculating $\mathbf{R}_{yy}$ for a third-order system would start at a lag of three and increase to a lag of five. Correlations for calculating $\mathbf{r}_{yy}$ would start at a lag of four and increase to a lag of six. This ensures that the effect of the MA part of the unknown system is removed from the linear-prediction of the AR part. This modified method of linear-prediction is given by the equation:

$$\begin{bmatrix} c'_0 \\ c'_1 \\ \cdot \\ \cdot \\ \cdot \\ c'_{n-N} \end{bmatrix} = \begin{bmatrix} r_{yy}(q) & r_{yy}(q-1) & \cdot\ \cdot & r_{yy}(q-p+1) \\ r_{yy}(q+1) & r_{yy}(q) & \cdot\ \cdot & r_{yy}(q-p+2) \\ \cdot & \cdot & \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot & \cdot \\ r_{yy}(q+p-1) & r_{yy}(q+p-2) & \cdot\ \cdot & r_{yy}(q) \end{bmatrix} \begin{bmatrix} r_{yy}(q+1) \\ r_{yy}(q+2) \\ \cdot \\ \cdot \\ \cdot \\ r_{yy}(q+p) \end{bmatrix} \tag{4.33}$$

where q is the order of the MA portion and p is the order of the AR portion. [Ref. 2: p. 182]

Figure 16 on page 40 shows the results of using modified linear-prediction with third-order test case T3. When comparing this graph to the estimates obtained without linear-prediction, shown in Figure 12 on page 35, note that the vertical axes have different scales. Table 6 on page 39 lists the values of the estimates at iteration 10 and compares them with the true values via the absolute and percent errors. A comparison of Table 6 on page 39 with Table 5 on page 33, the best estimates without the use of modified linear-prediction, shows that modified linear prediction has significantly increased the accuracy of the AR estimates at the tenth iteration. The accuracy of the MA estimates remains approximately the same. The tenth iteration was chosen as the point to select the parameter values because in both cases this was the iteration where the stopping parameter had the smallest value.

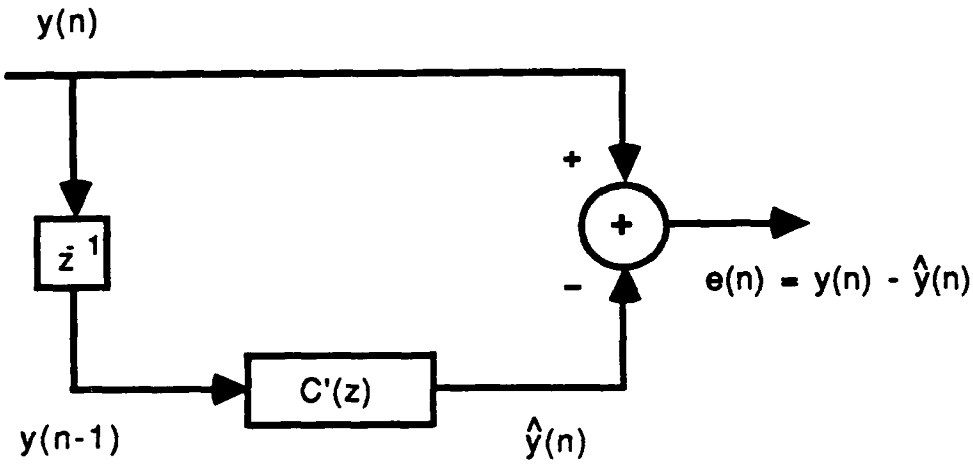


Figure 15. Linear prediction block diagram

Table 6. PARAMETER ESTIMATES USING MODIFIED LINEAR-PREDICTION FOR INITIAL ESTIMATE OF AR PARAMETERS.

TEST FILTER	PARAMETER ESTIMATE	ABSOLUTE ERROR	PERCENT ERROR	ITERATIONS
T3	0.0156	+0.0002	1.30	10
	0.0485	+0.0023	4.98	
	0.0512	+0.0050	10.80	
	0.0101	+0.0053	34.42	
	1.00	0.0	0.0	
	-1.98	+0.01	0.50	
	1.553	-0.019	1.21	
	-0.4458	+0.0125	2.73	

D. FORMULATION OF THE SEQUENTIAL MULTICHANNEL APPROACH

To decrease the computational intensity of updating the estimates of the AR and MA parameters due to new data, an algorithm to sequentially process the data has been

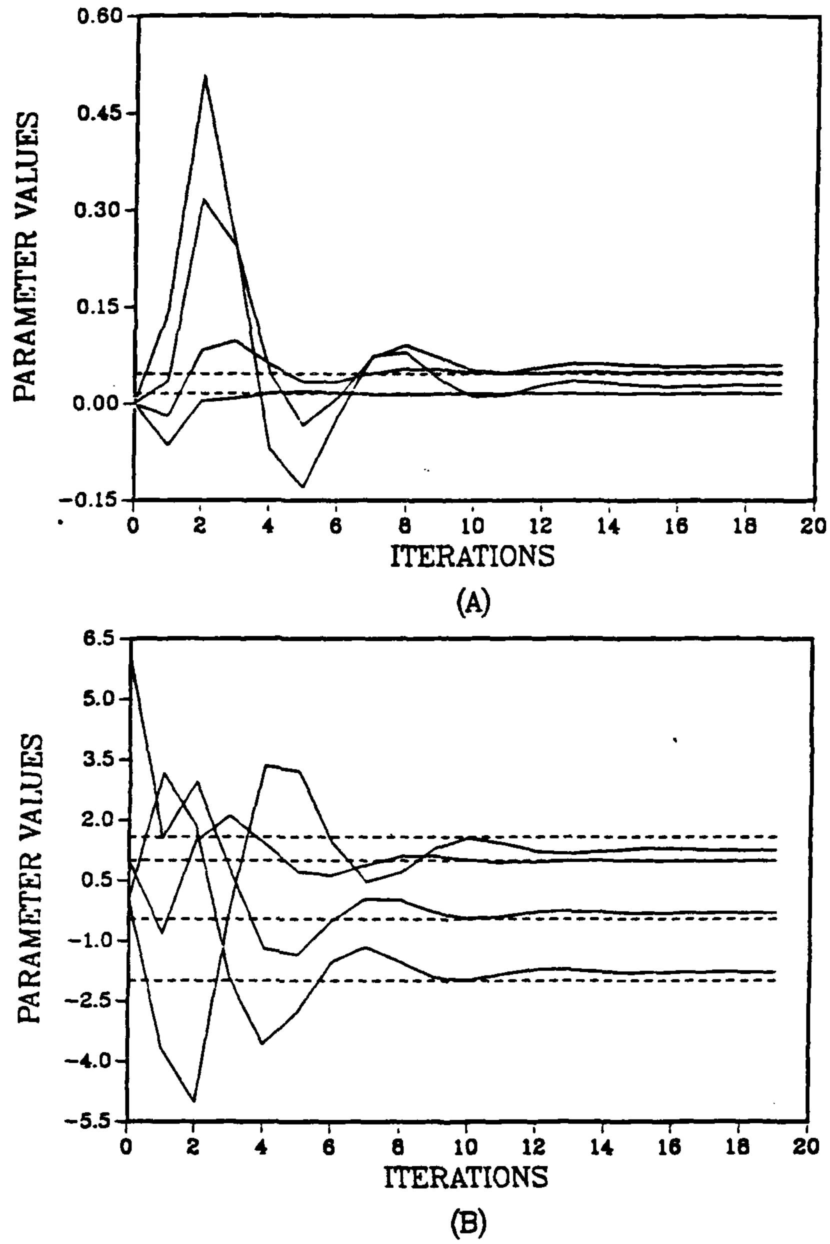


Figure 16. Parameter estimates for test case T3 using modified linear-prediction for initial estimate of AR parameters.

developed. Development begins with the performance criterion seen previously for the block data case in equation (4.10). From that starting point, equations are developed which relate new estimates of the parameters to the previous estimates and the new data. Separate but similar equations are developed for the MA and the AR coefficients. Due to the similar nature of the development of these equations, only the development of the equations for the AR coefficients is presented here. However, the final results for both the AR and MA coefficients are given.

The performance criterion can be written as:

$$J = \sum_{i=0}^{n} e(i)^2 = \sum_{l=0}^{n} [z_l - x_l]^2 = \sum_{l=0}^{n} [z_l - \mathbf{y}_l^T \mathbf{c}]^2 \tag{4.34}$$

Expanding this results in:

$$J = \sum_{l=0}^{n} z_l^T z_l - 2 z_l^T \mathbf{y}_l^T \mathbf{c} + \mathbf{c}^T \mathbf{y}_l \mathbf{y}_l^T \mathbf{c} \tag{4.35}$$

where **c** and **y** are defined in equations (4.7) and (4.8) and z is the scalar signal at the output of **D**. Differentiating the performance criterion with respect to **c** and setting the result equal to zero yields:

$$\frac{\partial J}{\partial \mathbf{c}} = 0 = \left(\sum_{l=0}^{n} \mathbf{y}_l \mathbf{y}_l^T\right)\mathbf{c} - \sum_{l=0}^{n} z_l \mathbf{y}_l \tag{4.36}$$

Equation (4.36) can also be written as

$$\sum_{l=0}^{n} z_l \mathbf{y}_l = \left(\sum_{l=0}^{n} \mathbf{y}_l \mathbf{y}_l^T\right)\mathbf{c} \tag{4.37}$$

Solving for the AR parameter vector results in:

$$\mathbf{c} = \left(\sum_{l=0}^{n} \mathbf{y}_l \mathbf{y}_l^T\right)^{-1} \sum_{l=0}^{n} z_l \mathbf{y}_l \tag{4.38}$$

Because $\mathbf{c}$ is an estimate based on data available through time n we signify this by introducing the index n on $\mathbf{c}$ to yield:

$$\mathbf{c}_n = (\sum_{i=0}^{n} \mathbf{y}_i \mathbf{y}_i^T)^{-1} \sum_{i=0}^{n} z_i \mathbf{y}_i \tag{4.39}$$

The first step in formulating the sequential algorithm is to develop an update equation for the estimate of the AR parameters. Applying the method presented in Graupe [Ref. 9: p. 124], we first define a new matrix $\mathbf{P}_n^{-1}$ as

$$\mathbf{P}_n^{-1} = \sum_{i=0}^{n} \mathbf{y}_i \mathbf{y}_i^T \tag{4.40}$$

This is a matrix of the output data of the unknown system. At the previous time $n-1$ this matrix is written as:

$$\mathbf{P}_{n-1}^{-1} = \sum_{i=0}^{n-1} \mathbf{y}_i \mathbf{y}_i^T \tag{4.41}$$

By substituting equation (4.40) into equation (4.39) the estimate of the AR parameters can be rewritten as:

$$\mathbf{c}_n = \mathbf{P}_n \sum_{i=0}^{n} z_i \mathbf{y}_i \tag{4.42}$$

The right side of equation (4.42) needs to be converted into an expression containing the previous estimate of the parameters plus a correction term. It needs the past value of $\mathbf{c}_n$ which is $\mathbf{c}_{n-1}$. From equation (4.39) we can write:

$$\mathbf{c}_{n-1} = (\sum_{i=0}^{n-1} \mathbf{y}_i \mathbf{y}_i^T)^{-1} \sum_{i=0}^{n-1} z_i \mathbf{y}_i \tag{4.43}$$

This can be rewritten as:

$$\sum_{i=0}^{n-1} z_i \mathbf{y}_i = (\sum_{i=0}^{n-1} \mathbf{y}_i \mathbf{y}_i^T) \mathbf{c}_{n-1} \tag{4.44}$$

Premultiplying equation (4.42) by $\mathbf{P}_n^{-1}$ and then separating the last term from the summation results in:

$$\mathbf{P}_n^{-1}\mathbf{c}_n = \sum_{i=0}^{n-1} z_i\mathbf{y}_i + z_n\mathbf{y}_n \tag{4.45}$$

Substituting for $\sum_{i=0}^{n-1} z_i\mathbf{y}_i$ in equation (4.45) from its equivalent expression in equation (4.44) yields:

$$\mathbf{P}_n^{-1}\mathbf{c} = (\sum_{i=0}^{n-1}\mathbf{y}_i\mathbf{y}_i^T)\mathbf{c}_{n-1} + z_n\mathbf{y}_n \tag{4.46}$$

By adding $\mathbf{y}_n\mathbf{y}_n^T\mathbf{c}_{n-1}$ to the right side of equation (4.46) and grouping it with the summation, the summation will range from $i = 0$ to n . In order not to affect the value on the right side of equation (4.46), $\mathbf{y}_n\mathbf{y}_n^T\mathbf{c}_{n-1}$ must also be subtracted from the right-hand side, which yields:

$$\mathbf{P}_{n-1}^{-1}\mathbf{c} = (\sum_{i=0}^{n-1}\mathbf{y}_i\mathbf{y}_i^T)\mathbf{c}_{n-1} + z_n\mathbf{y}_n + \mathbf{y}_n\mathbf{y}_n^T\mathbf{c}_{n-1} - \mathbf{y}_n\mathbf{y}_n^T\mathbf{c}_{n-1} \tag{4.47}$$

Combining $\mathbf{y}_n\mathbf{y}_n^T\mathbf{c}_{n-1}$ with the summation as describe above results in:

$$\mathbf{P}_{n-1}^{-1}\mathbf{c} = (\sum_{i=0}^{n}\mathbf{y}_i\mathbf{y}_i^T)\mathbf{c}_{n-1} + z_n\mathbf{y}_n - \mathbf{y}_n\mathbf{y}_n^T\mathbf{c}_{n-1} \tag{4.48}$$

Replacing $\sum_{i=1}^{n}\mathbf{y}_i\mathbf{y}_i^T$ with its equivalent expression from equation (4.40) yields:

$$\mathbf{P}_n^{-1}\mathbf{c}_n = \mathbf{P}_n^{-1}\mathbf{c}_{n-1} + \mathbf{y}_n(z_n - \mathbf{y}_n^T\mathbf{c}_{n-1}) \tag{4.49}$$

Premultiplying by $\mathbf{P}_n$ results in the following equation for the update of the estimate of the AR parameters:

$$\mathbf{c}_n = \mathbf{c}_{n-1} + \mathbf{P}_n\mathbf{y}_n(z_n - \mathbf{y}_n^T\mathbf{c}_{n-1}) \tag{4.50}$$

This is the desired result. It relates the estimate of the parameters at time N to the estimate at the previous time, $N-1$, plus the new output data vector, $\mathbf{y}_n$, and the error at time N. Error is represented by the $z_n - \mathbf{y}_n^T\mathbf{c}_{n-1}$ term. The corresponding equation for the MA parameters is:

$$\mathbf{d}_n = \mathbf{d}_{n-1} + \mathbf{Q}_n\mathbf{u}_n(x_n - \mathbf{u}_n^T\mathbf{d}_{n-1}) \tag{4.51}$$

In equation (4.51), $\mathbf{Q}$ is a matrix of the input data of the unknown system given by:

$$\mathbf{Q}_n^{-1} = \sum_{i=0}^{n}\mathbf{u}_i\mathbf{u}_i^T \tag{4.52}$$

Finally, we need a sequential update for $\mathbf{P}_n$ and $\mathbf{Q}_n$ to complete the sequential algorithm. This is accomplished by using a form of the matrix inversion lemma.

By pulling the last term out of the summation, the definition of P_n^{-1} given in equation (4.40) can be rewritten as:

$$\mathbf{P}_n^{-1} = \sum_{i=0}^{n-1}\mathbf{y}_i\mathbf{y}_i^T + \mathbf{y}_n\mathbf{y}_n^T \tag{4.53}$$

Substituting for $\sum_{i=0}^{n-1}\mathbf{y}_i\mathbf{y}_i^T$ its equivalent expression from equation (4.41) results in

$$\mathbf{P}_n^{-1} = \mathbf{P}_{n-1}^{-1} + \mathbf{y}_n\mathbf{y}_n^T \tag{4.54}$$

Inverting both sides of the equation results in:

$$\mathbf{P}_n = (\mathbf{P}_{n-1}^{-1} + \mathbf{y}_n\mathbf{y}_n^T)^{-1} \tag{4.55}$$

Let $\mathbf{A} = \mathbf{P}_{n-1}^{-1}$, $\mathbf{B} = \mathbf{y}_n$, $\mathbf{C} = 1$, and $\mathbf{D} = \mathbf{y}_n^T$. Then, by the matrix inversion lemma:

$$\mathbf{P}_n = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1} \tag{4.56}$$

Substituting the appropriate expressions into equation (4.56) results in:

$$\mathbf{P}_n = (\mathbf{P}_{n-1}^{-1})^{-1} - (\mathbf{P}_{n-1}^{-1})^{-1}\mathbf{y}_n[1 + \mathbf{y}_n^T(\mathbf{P}_{n-1}^{-1})^{-1}\mathbf{y}_n]^{-1}\mathbf{y}_n^T(\mathbf{P}_{n-1}^{-1})^{-1} \tag{4.57}$$

This reduces to:

$$\mathbf{P}_n = \mathbf{P}_{n-1} - \mathbf{P}_{n-1}\mathbf{y}_n[1 + \mathbf{y}_n^T\mathbf{P}_{n-1}\mathbf{y}_n]^{-1}\mathbf{y}_n^T\mathbf{P}_{n-1} \tag{4.58}$$

Using this same procedure, the update for $\mathbf{Q}_n$ is:

$$\mathbf{Q}_n = \mathbf{Q}_{n-1} - \mathbf{Q}_{n-1}\mathbf{u}_n[1 + \mathbf{u}_n^T\mathbf{Q}_{n-1}\mathbf{u}_n]^{-1}\mathbf{u}_n^T\mathbf{Q}_{n-1} \tag{4.59}$$

A reduction in the computational intensity has been achieved by reducing the matrix inversion of equation (4.55) to inversion of a scalar in equation (4.57). Inversion of these scalars is much simpler than inversion of the $\mathbf{R}_{yy}$ and $\mathbf{R}_{uu}$ matrices of the block processing case.

The sequential multichannel algorithm is summarized below:

- The parameter update equations:

$$\mathbf{c}_n = \mathbf{c}_{n-1} + \mathbf{P}_n\mathbf{y}_n(z_n - \mathbf{y}_n^T\mathbf{c}_{n-1}) \tag{4.60}$$

$$\mathbf{d}_n = \mathbf{d}_{n-1} + \mathbf{Q}_n\mathbf{u}_n(x_n - \mathbf{u}_n^T\mathbf{d}_{n-1}) \tag{4.61}$$

- The update equations for the **P** and **Q** matrices:

$$\mathbf{P}_n = \mathbf{P}_{n-1} - \mathbf{P}_{n-1}\mathbf{y}_n[1 + \mathbf{y}_n^T\mathbf{P}_{n-1}\mathbf{y}_n]^{-1}\mathbf{y}_n^T\mathbf{P}_{n-1} \tag{4.62}$$

$$\mathbf{Q}_n = \mathbf{Q}_{n-1} - \mathbf{Q}_{n-1}\mathbf{u}_n[1 + \mathbf{u}_n^T\mathbf{Q}_{n-1}\mathbf{u}_n]^{-1}\mathbf{u}_n^T\mathbf{Q}_{n-1} \tag{4.63}$$

The reduction in computational intensity comes with a trade-off. Now the algorithm is more complex to use. Updates are required for **P** and **Q** as well as **c** and **d** where before, in the block multichannel algorithm, updates were only required for **c** and **d**. But, as in the sequential IV case, an added advantage of the sequential multichannel algorithm is it allows updates of the estimates of the parameters based upon new data.

V. SUMMARY

In this thesis we set out to develop two algorithms for modeling unknown systems as ARMA processes. These are the IV method of system identification presented in Chapter 3, which is a modification of the method of ordinary least-squares, and the iterative multichannel method presented in Chapter 4.

The IV method is not a new concept in either its block or sequential processing forms. However, our derivation of the sequential algorithm was done independently of other IV sequential algorithms. We chose the IV method because it reportedly has good noise handling capabilities and yields consistent and unbiased estimates of the unknown system's parameters. It also remains as easy to use as the method of ordinary least-squares. We also wanted to gain familiarity with it because it was a possible candidate for incorporation into the multichannel method.

Operating alone, the IV method produces accurate estimates of the unknown system's parameters. Convergence was within 20 iterations for several second-order systems that we tested. Convergence slows down as the system order increases. However, the results do converge to the actual system parameters given sufficient processing time. The performance of the IV method is similar to the performance of the method of ordinary least-squares.

The proposed iterative multichannel algorithm is new in both its block and sequential processing forms to the best of our knowledge. It is very simple to use in the block form. It achieves accurate results for second-order systems but worse results for third-order systems with block correlation elements calculated based on only 500 data points. Implementing the stopping parameter increases the accuracy when the parameter estimates converge but not to the true parameters. Due to its ability to separately process the input and output data from the unknown system, correlation matrices in the multichannel block processing case are half the size of correlation matrices required for the single channel block processing case. This feature reduces the computational intensity over what is required for the conventional least-squares processing case. The number of iterations required for convergence seems to be independent of the order of the system. However, the accuracy of the estimates suffer as the order of the system increases. Using linear prediction to estimate the initial values of the AR parameters did not speed up convergence or increase the accuracy of the parameter estimates. However, using modified linear prediction significantly increased the accuracy of the AR parameter estimates, although it had no effect on the MA parameter estimates.

We formulated the multichannel sequential algorithm. This allows the estimates of the parameters to be updated as new data becomes available. But we have not tested this algorithm. It needs checking using a variety of second and third-order test cases to verify that it works. During testing, guidelines need to be developed for the best methods to initialize the **P** and **Q** matrices to achieve the quickest convergence and the most accurate parameter estimates.

As mentioned above, one of the reasons for investigating the IV method of system identification was for possible inclusion into the multichannel algorithm, the hope being that the favorable noise performance of the IV method would improve the performance of the multichannel method. This is another area that remains unexplored.

The block multichannel and IV methods achieved similar results for second-order test cases. Convergence to the actual system parameters came within 20 iterations for both algorithms. However, for third-order systems, convergence was much quicker with the multichannel block algorithm than with the sequential IV algorithm. But the parameter estimates by the IV method were more accurate than by the multichannel block method. A combination of the two algorithms has the potential for incorporating their unique advantages into a better overall parameter estimation method.

Areas for further research are listed below:

- Verify that the multichannel sequential algorithm developed in Chapter 4 works as a means of modeling an unknown system as an ARMA process.
- Investigate the possibility of incorporating the IV method into the multichannel sequential algorithm.
- Analyze why initializing the AR parameters to the values calculated by linear prediction improves the speed of convergence of the AR parameters in the multichannel block algorithm but does not improve the convergence of the MA parameters. Identify a method for obtaining an initial estimate of the MA parameters to improve their speed of convergence and accuracy.
- Investigate the effects of increasing the number of data points used to calculate the correlation matrices for the multichannel block algorithm on the accuracy of the parameter estimates and their speed of convergence.
- Investigate the performance of the IV method with noise present. Compare this performance with the performance of the method of ordinary least-squares with noise present.

APPENDIX

A. INSTRUMENTAL VARIABLE ALGORITHM

```
************************************************************************
*                                                                      *
*                          PAUL DAL SANTO                              *
*                           IV ALGORITHM                               *
*                             4/12/88                                  *
*                                                                      *
*     THIS PROGRAM CALCULATES THE AR AND MA PARAMETERS OF A            *
*     TEST SYSTEM BASED UPON ITS INPUT AND OUTPUT DATA                 *
*     BY USING THE SEQUENTIAL IV METHOD.                               *
*                                                                      *
*                                                                      *
************************************************************************

****    VARIABLE DEFINITIONS       ****

*     RIVCOF     ARRAY FOR STORING THE AR AND MA PARAMETERS
*                THE PROGRAM CALCULATES
*     Z          VECTOR OF DATA FROM THE OUTPUT OF THE AUXILIARY
*                MODEL AND THE INPUT TO THE TEST SYSTEM
*     Z TPO      TRANSPOSE OF VECTOR Z
*     X          VECTOR OF DATA FROM THE OUTPUT AND INPUT OF THE
*                TEST SYSTEM
*     X TPO      TRANSPOSE OF THE X VECTOR
*     U          STORAGE FOR INPUT DATA
*     Y          STORAGE FOR OUTPUT OF TEST SYSTEM
*     W          STORAGE FOR OUTPUT OF THE AUXILIARY MODEL
*     QMAT       THE Q MATRIX OF THE IV ALGORITHM
*     IV         VECTOR OF PARAMETERS CALCULATED BY THE ALGORITHM
*     COR        RESULT OF INTERMEDIATE STEP IN ALGORITHM CALCULATION
*     COEF       VECTOR OF TRUE PARAMETERS OF TEST SYSTEM
*     SCALAR     RESULT OF SCALAR INVERSION IN INTERMEDIATE STEP COR
*     SCALR2     INTERMEDIATE STEP WHEN CALCULATING THE NEW IV VECTOR
*     SEED       INITIALIZATION FOR IMSL GAUSSIAN ROUTINE
*     WSIZE      ORDER OF AR PART OF THE AUXILIARY MODEL
*     YSIZE      ORDER OF THE AR PART OF THE TEST SYSTEM
*     USIZE      ORDER OF THE MA PART OF THE TEST SYSTEM AND AUXILIARY
*                MODEL

*     VARIABLES THAT END IN R CONTAIN THE ROW SIZE OF THEIR RESPECTIVE
*     MATRICES.  VARIABLES THAT END IN C CONTAIN THE COLUMN SIZE OF
*     THEIR RESPECTIVE MATRICES.

****    VARIABLE DECLARATIONS     ****

      COMMON /D/ RIVCOF(0:1000,10)

      REAL Z(10,10),Z TPO(10,10),X(10,10),X TPO(10,10)
      REAL U(1),Y(10,10),W(10,10)
```

```
      REAL QMAT(10,10),Q TEMP(10,10),QTEMP2(10,10),QTEMP3(10,10)         IVA00490
      REAL IV(10,10),IVTEMP(10,10),COR(10,10),COEF(10,10)                IVA00500
      REAL SCALAR,SCALR2                                                 IVA00510
                                                                         IVA00520
      DOUBLE PRECISION SEED                                              IVA00530
                                                                         IVA00540
      INTEGER I,J,K,WSIZE,YSIZE,USIZE                                    IVA00550
      INTEGER ZMR,ZMC,ZTR,ZTC,XMR,XMC,XTR,XTC                            IVA00560
      INTEGER UMR,UMC,YMR,YMC,WMR,WMC                                    IVA00570
      INTEGER QMR,QMC,QTR,QTC,QT2R,QT2C,QT3R,QT3C                        IVA00580
      INTEGER IVR,IVC,IVTR,IVTC,CMR,CMC,COEFMR,COEFMC                    IVA00590
      INTEGER IVCR,IVCC                                                  IVA00600
      LOGICAL EOF                                                        IVA00610
                                                                         IVA00670
      READ(4,*,END=22) YSIZE,USIZE,COEFMR,COEFMC,ITERA                   IVA00680
      CALL RDMAT(COEF,COEFMR,COEFMC)                                     IVA00690
                                                                         IVA00700
* INITIALIZE VARIABLES                                                   IVA00630
                                                                         IVA00640
      EOF = .FALSE.                                                      IVA00650
      XTR = COEFMC                                                       IVA00710
      XTC = COEFMR                                                       IVA00720
      ZTR = COEFMC                                                       IVA00730
      ZTC = COEFMR                                                       IVA00740
      IVR = COEFMR                                                       IVA00750
      IVC = COEFMC                                                       IVA00760
      QMR = COEFMR                                                       IVA00770
      QMC = COEFMR                                                       IVA00780
      SEED = 888.0D1                                                     IVA00790
      IVCR = 0                                                           IVA00800
      IVCC = COEFMR                                                      IVA00810
      WSIZE = YSIZE                                                      IVA00820
                                                                         IVA00830
* ZERO OUT THE IV PARAMETER VECTOR, THE AUXILIARY MODEL DATA VECTOR      IVA00840
* AND THE TEST SYSTEM DATA VECTOR.                                       IVA00850
                                                                         IVA00860
      CALL INIT(IV,IVR,IVC,0.0)                                          IVA00870
      CALL INIT(Z TPO,ZTR,ZTC,0.0)                                       IVA00880
      CALL INIT(X TPO,XTR,XTC,0.0)                                       IVA00890
                                                                         IVA00900
* INITIALIZE THE QMAT AS A DIAGONAL MATRIX WHOSE DIAGONAL ELEMENTS       IVA00910
* EQUAL 100                                                              IVA00920
                                                                         IVA00930
      CALL INITD(QMAT,QMR,QMC,100.0)                                     IVA00940
                                                                         IVA00950
* GET VALUE FOR U(0).  U IS A GAUSSIAN RANDOM VARIABLE.                  IVA00960
                                                                         IVA00970
      CALL GGNML (SEED,1,U)                                              IVA00990
                                                                         IVA01000
* SHIFT U(0) INTO X & Z VECTORS TO CREATE X(0) & Z(0)                    IVA01010
                                                                         IVA01020
      Y(1,1) = 0.0                                                       IVA01030
      CALL SHIFT(X TPO,XTC,YSIZE,USIZE,Y(1,1),U(1))                      IVA01040
      W(1,1) = 0.0                                                       IVA01050
      CALL SHIFT(Z TPO,ZTC,WSIZE,USIZE,W(1,1),U(1))                      IVA01060
```

```
* CALCULATE Y(0) = X TPO(0) * COEFFICIENT VECTOR

      CALL MULTI(X TPO,XTR,XTC,COEF,COEFMR,COEFMC,Y,1,1)

* CALCULATE W(0) = Z TPO(0) * IV VECTOR

      CALL MULTI(Z TPO,ZTR,ZTC,IV,IVR,IVC,W,1,1)
      DO 90 I = 0,ITERA
         CALL GGNML(SEED,1,U)
         CALL TPOSE(IV,IVR,IVC,IVTEMP,IVTR,IVTC)

* SAVE THE IV PARAMETERS

         DO 91 J = 1,IVTC
            RIVCOF(I,J) = IVTEMP(1,J)
91       CONTINUE
         CALL PRMAT(IVTEMP,IVTR,IVTC)

* SHIFT Y(M) AND U(M+1) INTO X TPO(M) TO GET X TPO(M+1)
         CALL SHIFT(X TPO,XTC,YSIZE,USIZE,Y(1,1),U(1))

* SHIFT W(M) AND  U(M+1) INTO Z TPO(M) TO GET Z TPO(M+1)
         CALL SHIFT(Z TPO,ZTC,WSIZE,USIZE,W(1,1),U(1))

* CALCULATE Y(M+1) AND Z(M+1)
         CALL MULTI(X TPO,XTR,XTC,COEF,COEFMR,COEFMC,Y,1,1)
         CALL MULTI(Z TPO,ZTR,ZTC,IV,IVR,IVC,W,1,1)

* CALCULATE THE NEW Q MATRIX

         CALL MULTI(X TPO,XTR,XTC,QMAT,QMR,QMC,Q TEMP,QTR,QTC)
         CALL TPOSE(Z TPO,ZTR,ZTC,Z,ZMR,ZMC)
         CALL CORE(QMAT,QMR,QMC,Z,ZMR,ZMC,X TPO,XTR,XTC,COR,CMR,CMC)
         CALL MULTI(COR,CMR,CMC,Q TEMP,QTR,QTC,QTEMP2,QT2R,QT2C)
         CALL SUBTRC(QMAT,QMR,QMC,QTEMP2,QT2R,QT2C,QMAT,QMR,QMC)

* CALCULATE THE NEW IV VECTOR

         CALL MULTI(X TPO,XTR,XTC,IV,IVR,IVC,IVTEMP,IVTR,IVTC)
         SCALR2 = Y(1,1) - IVTEMP(1,1)
         CALL SMULTI(SCALR2,COR,CMR,CMC)
         CALL ADD(IV,IVR,IVC,COR,CMR,CMC,IV,IVR,IVC)

90    CONTINUE

* PLOT THE IV PARAMETERS VS THE ITERATION NUMBER
      CALL PLOT2(ITERA,USIZE,YSIZE,COEF)

22    STOP
      END

*     ******************************************************************
      SUBROUTINE CORE(MAT1,I1R,I1C,MAT2,I2R,I2C,MAT3,I3R,I3C,RMAT,IRR,
     +IRC)
*     ******************************************************************
```

```
      REAL MAT1(10,10),MAT2(10,10),MAT3(10,10),RMAT(10,10)
      REAL Q TEMP(10,10),QTEMP2(10,10)
      INTEGER IRR,IRC,QTR,QTC,QT2R,QT2C

* CALCULATE THE CORE: Q(M)Z(M+1)°1+X'(M+1)Q(M)Z(M+1) **-1
* MAT1 IS THE Q MATRIX, MAT2 IS THE Z VECTOR, AND MAT3 IS THE
* X VECTOR.

      CALL MULTI(MAT1,I1R,I1C,MAT2,I2R,I2C,Q TEMP,QTR,QTC)
      CALL MULTI(MAT3,I3R,I3C,Q TEMP,QTR,QTC,QTEMP2,QT2R,QT2C)
      SCALAR = 1/(1 + QTEMP2(1,1))
      CALL SMULTI(SCALAR,Q TEMP,QTR,QTC)
      CALL ADD(Q TEMP,QTR,QTC,Q TEMP,QTR,QTC,RMAT,IRR,IRC)
      CALL SMULTI(0.5,RMAT,IRR,IRC)
      RETURN
      END

*     ******************************
      SUBROUTINE PLOT2(ITERA,ICURVN,ICURVD,MAT1)
*     ******************************

      COMMON /D/ RIVCOF(0:1000,10)
      REAL X(0:1000),Y(0:1000),MAT1(10,10),MAX,MIN
      INTEGER I,J,ITERA,ICURVN,ICURVD,ISTP

      CALL LIMITS(ICURVN,ICURVD,NMAX,NMIN,NSTP,
     +DMAX,DMIN,DSTP,ITERA)

* GENERATE THE ITERATION NUMBER

      DO 90 I = 0,ITERA
         X(I) = I
         Y(I) = 0.0
90    CONTINUE

* CALCULATE X AXIS LABELING INTERVAL
      ISTP = ITERA/10

* SET UP THE PLOT

      CALL SHERPA('IVGRAF  ','A',3)
      CALL RESET('ALL')
      CALL PAGE(8.5.11.0)
      CALL HEIGHT(0.14)
      CALL HWROT('AUTO')
      CALL XINTAX
      CALL YAXANG(0)
      CALL PHYSOR(1.5,6.0)
      CALL AREA2D(5.0,3.5)
      CALL COMPLX
      CALL XNAME('ITERATIONS$',100)
      CALL YNAME('COEFFICIENT VALUES$',100)
      CALL MESSAG('(A)$',100,2.4,-0.8)
      CALL THKFRM(0.03)
      CALL FRAME
```

```
      CALL GRAF(0,ISTP,ITERA,NMIN,NSTP,NMAX)                    IVA04190
                                                                IVA04210
* PLOT THE NUMERATOR VALUES                                     IVA04230
                                                                IVA04240
      DO 93 J = ICURVD + 1,ICURVN + ICURVD                      IVA04260
         DO 94 I = 0,ITERA                                      IVA04270
            Y(I) = RIVCOF(I,J)                                  IVA04280
94       CONTINUE                                               IVA04290
         CALL CURVE(X,Y,ITERA,0)                                IVA04300
93    CONTINUE                                                  IVA04310
                                                                IVA04320
* PLOT DASHED LINES FOR THE COEFS' TRUE VALUES                  IVA04330
      CALL DASH                                                 IVA04350
                                                                IVA04360
* PLOT NUMERATOR COEFS' TRUE VALUES                             IVA04370
                                                                IVA04380
      DO 95 K = ICURVD + 1,ICURVD + ICURVN                      IVA04390
         DO 96 J = 0,ITERA                                      IVA04400
            Y(J) = MAT1(K,1)                                    IVA04410
96       CONTINUE                                               IVA04420
         CALL CURVE(X,Y,ITERA,0)                                IVA04430
95    CONTINUE                                                  IVA04440
      CALL ENDGR(0)                                             IVA04450
                                                                IVA04460
* SET UP SECOND PLOT FOR DENOMINATOR PARAMETERS                 IVA04470
                                                                IVA04480
      CALL RESET('DASH')                                        IVA04490
      CALL PHYSOR(1.5,1.5)                                      IVA04500
      CALL AREA2D(5.0,3.5)                                      IVA04510
      CALL COMPLX                                               IVA04520
      CALL XNAME('ITERATIONS$',100)                             IVA04550
      CALL YNAME('PARAMETER VALUES$',100)                       IVA04560
      CALL MESSAG('(B)$',100,2.4,-0.8)                          IVA04570
      CALL THKFRM(0.03)                                         IVA04580
      CALL FRAME                                                IVA04590
      CALL GRAF(0,ISTP,ITERA,DMIN,DSTP,DMAX)                    IVA04600
                                                                IVA04610
* PLOT THE DENOMINATOR VALUES                                   IVA04620
                                                                IVA04630
      DO 91 J = 1,ICURVD                                        IVA04640
         DO 92 I = 0,ITERA                                      IVA04650
            Y(I) = -RIVCOF(I,J)                                 IVA04660
92       CONTINUE                                               IVA04670
         CALL CURVE(X,Y,ITERA,0)                                IVA04680
91    CONTINUE                                                  IVA04690
                                                                IVA04700
*     PLOT DENOMINATOR COEFS' TRUE VALUES                       IVA04710
                                                                IVA04720
      CALL DASH                                                 IVA04730
      DO 99 K =  1,ICURVD                                       IVA04740
         DO 100 J = 0,ITERA                                     IVA04750
            Y(J) = -MAT1(K,1)                                   IVA04760
100      CONTINUE                                               IVA04770
         CALL CURVE(X,Y,ITERA,0)                                IVA04780
99    CONTINUE                                                  IVA04790
                                                                IVA04800
```

```
      CALL ENDPL(0)                                                     IVA04810
      CALL DONEPL                                                       IVA04820
      RETURN                                                            IVA04830
      END                                                               IVA04840
                                                                        IVA04850
*     ************************************************************      IVA05760
      SUBROUTINE SUBTRC(MAT1,I1R,I1C,MAT2,I2R,I2C,RMAT,IRR,IRC)         IVA05770
*     ************************************************************      IVA05780
                                                                        IVA05790
****    PURPOSE - ROUTINE SUBTRACTS MAT2 FROM MAT1 AND PUTS THE         IVA05800
*       RESULT IN A THIRD MATRIX.                                       IVA05810
                                                                        IVA05820
      REAL MAT1(10,10),MAT2(10,10),RMAT(10,10)                          IVA05850
      INTEGER IRR,IRC                                                   IVA05860
                                                                        IVA05870
      CALL SMULTI(-1.0,MAT2,I2R,I2C)                                    IVA05880
      CALL ADD(MAT1,I1R,I1C,MAT2,I2R,I2C,RMAT,IRR,IRC)                  IVA05890
      IRR = I1R                                                         IVA05900
      IRC = I1C                                                         IVA05910
      RETURN                                                            IVA05920
      END                                                               IVA05930
                                                                        IVA05940
*     ******************************************************            IVA05970
      SUBROUTINE TPOSE(MAT1,I1R,I1C,RMAT,IRR,IRC)                       IVA05980
*     ******************************************************            IVA05990
                                                                        IVA06000
****   PURPOSE-SUBROUTINE TRANSPOSES A MATRIX AND PUTS THE RESULT       IVA06010
       INTO A NEW MATRIX                                                IVA06020
                                                                        IVA06030
      REAL MAT1(10,10),RMAT(10,10)                                      IVA06040
      INTEGER I,J,IRR,IRC                                               IVA06050
                                                                        IVA06060
      DO 93 I=1,I1C                                                     IVA06070
         DO 94 J=1,I1R                                                  IVA06080
            RMAT(I,J) = MAT1(J,I)                                       IVA06090
94       CONTINUE                                                       IVA06100
93    CONTINUE                                                          IVA06110
      IRR = I1C                                                         IVA06120
      IRC = I1R                                                         IVA06130
      RETURN                                                            IVA06140
      END                                                               IVA06150
```

B. MULTICHANNEL ALGORITHM

```
C     ****************************************************************  DUA00010
C     *                                                              *  DUA00020
C     *         PAUL DAL SANTO    8/15/88                            *  DUA00030
C     *                                                              *  DUA00040
C     *         TWO-CHANNEL SYSTEM IDENTIFICATION ALGORITHM          *  DUA00050
*     *                                                              *  DUA00060
*     *         PROGRAM CALCULATES THE AR AND MA PARAMETERS          *  DUA00070
*     *         BASED UPON THE INPUT AND OUTPUT DATA OF A            *  DUA00080
*     *         TEST SYSTEM.                                         *  DUA00090
C     *                                                              *  DUA00100
C     *         SHIFT USED TO CALC RYY FOR THE LINEAR                *  DUA00110
```

```
C     *       OF THE AR PARAMETERS IS READ FROM                      *
*     *       THE COEFF DATA FILE                                    *
C     *                                                              *
C     *       NUMBER OF ITERATIONS IS READ FROM COEFF                *
C     *       DATA FILE                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     ****************************************************************

*     ****    VARIABLE DEFINITIONS     ****

*     RAWDAT    ARRAY WHICH CONTAINS THE INPUT AND OUTPUT DATA
*               OF THE SYSTEM UNDER TEST
*     E1        ARRAY FOR STORING THE STOPPING PARAMETER AND THE
*               COEFFICIENT ERROR
*     ARRD      ARRAY FOR STORING THE AR PARAMETER ESTIMATES
*     ARRN      ARRAY FOR STORING THE MA PARAMETER ESTIMATES
*     RYYM      AUTOCORRELATION MATRIX OF THE OUTPUT DATA
*     RYUM,RUYM   CROSSCORRELATION MATRICES
*     RUUM      AUTOCORRELATION MATRIX OR THE INPUT DATA
*     RUUINV    INVERSE OF THE AUTOCORRELATION MATRIX OF THE INPUT DATA
*     RYYINV    INVERSE OF THE AUTOCORRELATION MATRIX OF THE OUTPT DATA
*     DM        VECTOR OF CURRENT MA PARAMETER ESTIMATES
*     CM        VECTOR OF CURRENT AR PARAMETER ESTIMATES
*     TRUNRM    FUNCTION WHICH CALCULATES THE NORM OF THE TRUE VALUES
*               OF THE TEST SYSTEM'S PARAMETERS
*     WKMAT     WORKING MATRIX FOR DOING MATRIX INVERSIONS
*     X TPO     TRANSPOSE OF THE VECTOR OF INPUT DATA
*     Y         MATRIX FOR THE OUTPUT OF THE TEST SYSTEM
*     U         INPUT TO THE TEST SYSTEM
*     COEFM     VECTOR OF TRUE COEFFICIENTS OF THE TEST SYSTEM
*     ITERA     NUMBER OF ITERATIONS TO PERFORM
*     CORRLN    LENGTH OF CORRELATIONS TO USE TO CALCULATE RYY, RUU
*               RYU, AND RUY
*     YSIZE     ORDER OF THE AR PART OF THE TEST SYSTEM
*     USIZE     ORDER OF THE MA PART OF THE TEST SYSTEM
*     KTIME     THE CURRENT ITERATION
*     SHFT      SIZE OF THE STARTING LAG FOR LINEAR PREDICTION OF
*               THE AR PARAMETERS
*     SEED      INITIALIZATION PARAMETER FOR IMSL ROUTINE WHICH
*               GENERATES RANDOM GAUSSIAN NUMBERS

*     INTEGER VARIABLES THAT END WITH R CONTAIN THE ROW SIZE OF
*     A PARTICULAR ARRAY.  INTEGER VARIABLES THAT END WITH C CONTAIN
*     THE COLUMN SIZE OF A PARTICULAR ARRAY.

*     ****    VARIABLE DECLARATIONS    ****

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)

      REAL RYYM(5,5),RYUM(5,5),RUUM(5,5),RUYM(5,5),RUUINV(5,5)
      REAL RYYINV(5,5),DM(5,5),CM(5,5),TRUNRM

      INTEGER RYYR,RYYC,RUUR,RUUC,RYUR,RYUC,RUYR,RUYC
```

```
      INTEGER DR,DC,CR,CC

      REAL WKMAT(12,12),X TPO(10,10),Y(2,2),U(1),COEFM(10,10)
      INTEGER WKR,WKC,XTR,XTC,COEFR,COEFC,ERR

      INTEGER ITERA,CORRLN,YSIZE,USIZE,KTIME,SHFT

      DOUBLE PRECISION SEED

* TEMPORATY MATRICES FOR PERFORMING CALCUALTIONS

      REAL T1M(5,5),T2M(5,5),T3M(5,5)
      INTEGER T1R,T1C,T2R,T2C,T3R,T3C


* BEGIN MAIN PROGRAM
*-----------------------------------------------------

* READ IN THE SIZE OF THE TEST SYSTEM, THE NUMBER OF ITERATIONS
* TO PERFORM AND THE BEGINNING LAG OF LINEAR PREDICTION OF THE
* AR PARAMETERS

      READ(4,*,END=22) YSIZE,USIZE,COEFR,COEFC,ITERA,SHFT

* READ IN THE TRUE VALUES OF THE COEFFICIENTS OF THE TEST SYSTEM

      CALL RDMAT(COEFM,COEFR,COEFC)

* INITIALIZE ROW AND COLUMN SIZES OF AS WELL AS OTHER VARIABLES

      CORRLN = 500
      RUUR = USIZE
      RUUC = USIZE
      RYYR = YSIZE + 1
      RYYC = YSIZE + 1
      T3R = YSIZE
      T3C = YSIZE
      DR = USIZE
      DC = 1
      CR = YSIZE + 1
      CC = 1
      XTR = COEFC
      XTC = COEFR
      SEED = 888.0D1
      II = 0
      Y(1,1) = 0.0
      E1(1,0) = 0.0
      DIVISR = TRUNRM(COEFM,COEFR,YSIZE,USIZE)

* ZERO OUT THE CORRELATION MATRICES, THE MA PARAMETER VECTOR AND
* VECTOR OF INPUT DATA

      CALL INIT(RYYM,RYYR,RYYC,0.0)
      CALL INIT(RUUM,RUUR,RUUC,0.0)
      CALL INIT(RYUM,RYYR,RUUC,0.0)
      CALL INIT(RUYM,RUUR,RYYC,0.0)
```

```
      CALL INIT(DM,DR,DC,0.0)
      CALL INIT(X TPO,XTR,XTC,0.0)

* INITIALIZE THE AR PARAMETER VECTOR

      CALL INITD(CM,CR,CC,1.0)
      CALL PRMAT(CM,CR,CC)

* RUN THE FILTER FOR 2000 TIME STEPS TO GENERATE OUTPUT DATA
*----------------------------------------------------------

      DO 70 KTIME = 0,2000

*     GET VALUE FOR U(K).  U IS A GAUSSIAN RANDOM VARIABLE.
         CALL GGNML (SEED,1,U)
*     SHIFT U(K) INTO X VECTOR TO CREATE X(K)
         CALL SHIFT(X TPO,XTR,XTC,YSIZE,USIZE,Y(1,1),U(1))
*     CALCULATE VALUE OF Y(K)
         CALL MULTI(X TPO,XTR,XTC,COEFM,COEFR,COEFC,Y,1,1)

*      SAVE THE INPUT AND OUTPUT DATA

         RAWDAT(1,KTIME) = Y(1,1)
         RAWDAT(2,KTIME) = U(1)

70    CONTINUE

15    FORMAT (2X,I4,2X,F8.5,2X,F8.5)

* CALCULATE THE CORRELATION MATRICES

      CALL AUTCOR(1,RYYM,RYYR,RYYC,CORRLN)
      CALL AUTCOR(2,RUUM,RUUR,RUUC,CORRLN)
      CALL CRSCOR(1,2,RYUM,RYYR,RUUC,CORRLN)
      CALL CRSCOR(2,1,RUYM,RUUR,RYYC,CORRLN)

* INVERT THE AUTOCORRELATION MATRICES OF THE OUTPUT AND INPUT DATA

      CALL LINV2F(RYYM,RYYR,RYYC,RYYINV,0,WKMAT,ERR)
      CALL LINV2F(RUUM,RUUR,RUUC,RUUINV,0,WKMAT,ERR)

* MULTIPLY THE INVERSE OF THE AUTOCORRELATION MATRICES BY THEIR
* RESPECTIVE CROSSCORRELATION MATRICES

      CALL MULTI(RUUINV,RUUR,RUUC,RUYM,RUUR,RYYC,T1M,T1R,T1C)
      CALL MULTI(RYYINV,RYYR,RYYC,RYUM,RYYR,RUUC,T2M,T2R,T2C)

* ESTIMATE THE AR PARAMETERS BY LINEAR PREDICTION
      IF (SHFT.GE.1) THEN
         CALL CORLA4(DR,CM,CR,CC,T3M,CORRLN,SHFT)
      ENDIF

      WRITE (3,26) II,DM(1,1),DM(2,1),DM(3,1),DM(4,1),DM(5,1)
      WRITE (3,16) II,CM(1,1),CM(2,1),CM(3,1),CM(4,1),CM(5,1)
      WRITE (3,*) '   '
      CALL SAVE(1,0,II,DM,DR,DC)
```

```
      CALL SAVE(2,0,II,CM,CR,CC)

* CALCULATE THE COEFFICIENT ERROR
      CALL ERR2(COEFM,COEFR,CM,CR,DM,DR,II,YSIZE,USIZE,DIVISR)

*     BEGIN THE ITERATIVE PROCEDURE
*-----------------------------------------------------------------

      DO 100 II = 1,ITERA

*     CALCULATE THE VALUES OF THE PARAMETERS AT ITERATION II

         CALL MULTI(T1M,T1R,T1C,CM,CR,CC,DM,DR,DC)
         CALL MULTI(T2M,T2R,T2C,DM,DR,DC,CM,CR,CC)
         CALL SAVE(1,0,II,DM,DR,DC)
         CALL SAVE(2,0,II,CM,CR,CC)

*     CALCULATE THE STOPPING PARAMETER
         CALL ERROR(CM,CR,CC,DM,DR,DC,II)

*     CALCULATE THE COEFFICIENT ERROR
         CALL ERR2(COEFM,COEFR,CM,CR,DM,DR,II,YSIZE,USIZE,DIVISR)

         WRITE (3,26) II,DM(1,1),DM(2,1),DM(3,1),DM(4,1),DM(5,1)
26       FORMAT (I4,1X,'NUM COEF =',5(1X,F9.6))
         WRITE (3,16) II,CM(1,1),CM(2,1),CM(3,1),CM(4,1),CM(5,1)
16       FORMAT (I4,1X,'DNM COEF =',5(1X,F9.6))
         WRITE (3,*) '     ERROR = ',E1(1,II),'   COEF ERROR = ',E1(2,II)
         WRITE (3,*) '     '

         CM(1,1) = 1.0

100   CONTINUE

* END OF THE ITERATIVE PROCEDURE
*-----------------------------------------------------------------

* PLOT THE PARAMETER VALUES
      CALL PLOT1(ITERA,DR,CR,COEFM)

22    STOP
      END

*     **************************************************
      FUNCTION TRUNRM (MAT1,M1R,YSZ,USZ)
*     **************************************************

      REAL MAT1(M1R,1)
      INTEGER YSZ,USZ

      VALUE = 0
      DO 91 I = 1,YSZ + USZ
         VALUE = VALUE + (MAT1(I,1))**2
91    CONTINUE
      TRUNRM = SQRT(VALUE + 1)
      RETURN
```

```
      END

*     **********************************************
      SUBROUTINE AUTCOR(IFST,MAT1,M1R,M1C,CORRLN)
*     **********************************************

* THIS SUBROUTINE CALCULATES THE AUTOCORRELATION MATRIX USING
* CORRELATIONS OF SIZE CORRLN

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL MAT1(M1R,M1C)
      INTEGER I,J,K,CORRLN

      CALL INIT(MAT1,M1R,M1C,0.0)

*     CALC CORRELATIONS ALONG THE FIRST ROW OF THE MATRIX.  THE
*     SHIFT = ABS(NUMBER OF THE COLUMN - 1)
*     LENGTH OF CORR = D LENGTH + ABS(1 - THE NUMBER OF THE COLUMN)

*     ONCE CORR HAVE BEEN CALCULATED FOR THE FIRST ROW, THEY CAN
*     BE COPIED INTO OTHER ROWS.  HORIZ DISTANCE OF THE PARTICULAR
*     ELEMENT FROM THE MAIN DIAGONAL DETERMINES WHICH CORR TO
*     COPY. THIS DISTANCE IS GIVEN BY ABS(ROW NUMBER - COLUMN
*     NUMBER).

*     CORRELATIONS START 200 POINTS FROM THE BEGINNING OF THE
*     DATA TO ELIMINATE THE TRANSIENT OF THE TEST SYSTEM.

      DO 90 J = 1,M1C
      ENDPT = 200 + CORRLN - ABS(1 - J)
         DO 93 K = 200,ENDPT
            MAT1(1,J) = MAT1(1,J) + RAWDAT(IFST,K-(1-J))*RAWDAT(IFST,K)
93       CONTINUE
         MAT1(1,J) = MAT1(1,J)/(CORRLN + 1 - ABS(1 - J))
90    CONTINUE

      DO 91 I = 2,M1R
         DO 92 J = 1,M1C
            MAT1(I,J) = MAT1(1,1+ABS(I-J))
92       CONTINUE
91    CONTINUE

      RETURN
      END

*     ******************************************
      SUBROUTINE COPY(MAT1,M1R,M1C,MAT2,M2R,M2C)
*     ******************************************

* THIS SUBROUTINE COPIES A MATRIX INTO A SECOND MATRIX.

      REAL MAT1(M1R,M1C),MAT2(M2R,M2C)
      INTEGER I,J

      DO 90 I = 1,M1R
```

```
         DO 900 J = 1,M1C                                                DUA03450
            MAT2(I,J) = MAT1(I,J)                                        DUA03460
900      CONTINUE                                                        DUA03470
90    CONTINUE                                                           DUA03480
      M2R = M1R                                                          DUA03500
      M2C = M1C                                                          DUA03510
      RETURN                                                             DUA03530
      END                                                                DUA03540
                                                                         DUA03550
*     **********************************************                     DUA03560
      SUBROUTINE CRSCOR(IFST,ISND,MAT1,M1R,M1C,CORRLN)                   DUA03570
*     **********************************************                     DUA03580
                                                                         DUA03590
* THIS SUBROUTINE CALCULATES THE CROSSCORRELATION MATRIX USING           DUA03600
* CORRELATIONS OF LENGTH CORRLN                                          DUA03610
                                                                         DUA03620
      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),                          DUA03630
     +ARRD(0:1000,5),ARRN(0:1000,5)                                      DUA03640
      REAL MAT1(M1R,M1C)                                                 DUA03660
      INTEGER I,J,K,CORRLN                                               DUA03670
                                                                         DUA03680
      CALL INIT(MAT1,M1R,M1C,0.0)                                        DUA03690
                                                                         DUA03700
*     FOR CROSS CORRELATION MUST CALC EACH ELEMENT OF THE CORR MATRIX    DUA03710
*     SEPARATELY.  CORRELATIONS START 200 POINTS FROM THE BEGINNING      DUA03720
*     OF THE DATA TO ELIMINATE THE TRANSIENT OF THE TEST SYSTEM.         DUA03730
                                                                         DUA03740
      DO 94 I = 1,M1R                                                    DUA03760
         DO 95 J = 1,M1C                                                 DUA03770
            ENDPT = 200 + CORRLN - ABS(I - J)                            DUA03780
            IF (J.GE.I) THEN                                             DUA03790
               DO 96 K = 200,ENDPT                                       DUA03800
                 MAT1(I,J) = MAT1(I,J) + RAWDAT(IFST,K-(I-J))            DUA03810
     +*RAWDAT(ISND,K)                                                    DUA03820
96             CONTINUE                                                  DUA03830
            ELSE                                                         DUA03840
               DO 97 K = 200,ENDPT                                       DUA03850
                 MAT1(I,J) = MAT1(I,J) + RAWDAT(IFST,K)*                 DUA03860
     +RAWDAT(ISND,K+(I-J))                                               DUA03870
97             CONTINUE                                                  DUA03880
            ENDIF                                                        DUA03890
            MAT1(I,J) = MAT1(I,J)/(CORRLN + 1 - ABS(I-J))                DUA03900
95       CONTINUE                                                        DUA03910
94    CONTINUE                                                           DUA03920
      RETURN                                                             DUA03950
      END                                                                DUA03960
                                                                         DUA03970
*     ************************************************                   DUA03980
      SUBROUTINE CORLA4(ISIZE,MAT2,M2R,M2C,MAT3,CORRLN,SHIFTT)           DUA03990
*     ************************************************                   DUA04000
                                                                         DUA04010
* THIS SUBROUTINE CALCULATES THE CORRELATION MATRIX                      DUA04020
* AND THE CORRELATION VECTOR USED FOR LINEAR PREDICTION OF THE           DUA04030
* AR PARAMETERS.  IT THEN CALCULATES THE INITIAL ESTIMATE OF THE         DUA04040
* AR PARAMETERS AND PASSES THEM BACK TO THE MAIN PROGRAM IN MAT2.        DUA04050
                                                                         DUA04060
```

```
      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL MAT2(M2R,M2C),MAT3(M2R-1,M2R-1)
      REAL T2M(5,1),T3M(5,1),T4M(5,5),WKMAT(9,9)
      INTEGER ERR,T1R,T1C,T2R,T2C,T3R,T3C,ISIZE,SHIFTT,CORRLN

*     GENERATE THE FIRST ROW OF THE RYY MATRIX.
*     SHIFTS ARE GREATER THAN THE ORDER OF THE
*     NUMERATOR.

      M3R = M2R-1
      M3C = M3R
      CALL INIT(MAT3,M3R,M3C,0.0)

      DO 90 J = 1,M3C
         ENDPT = 200 + CORRLN - SHIFTT
         DO 91 K = 200,ENDPT
            MAT3(1,J) = MAT3(1,J) + RAWDAT(1,K+SHIFTT)*RAWDAT(1,K)
91       CONTINUE
         MAT3(1,J) = MAT3(1,J)/(CORRLN-SHIFTT+1)
         SHIFTT = SHIFTT + 1
90    CONTINUE

*     COPY ELEMENTS FROM THE FIRST ROW INTO OTHER LOCATIONS

      DO 92 I = 2,M3R
         DO 93 J = 1,M3C
            MAT3(I,J) = MAT3(1,1+ABS(I-J))
93       CONTINUE
92    CONTINUE

*     GENERATE THE RYY VECTOR BY COPYING ELEMENTS OF THE RYY MATRIX

      T2R = M3C
      T2C = 1
      CALL FILL(1,T2R-1,1,1,T2M,T2R,1,2,1,MAT3,M3R,M3C)

*     GENERATE THE LAST ELEMENT IN THE RYY CORRELATION VECTOR.

      FINELE = 0.0
      ENDPT = 200 + CORRLN - SHIFTT
      DO 94 K = 200,ENDPT
         FINELE = FINELE + RAWDAT(1,K+SHIFTT)*RAWDAT(1,K)
94    CONTINUE
         FINELE = FINELE/(CORRLN+1-SHIFTT)

*     COPY THE FINAL ELEMENT INTO THE VECTOR
      T2M(T2R,1) = FINELE

*     CALCULATE THE INITIAL ESTIMATE OF THE AR PARAMETERS

      CALL LINV2F(MAT3,M3R,M3C,T4M,0,WKMAT,ERR)
      CALL MULTI(T4M,M3R,M3C,T2M,T2R,T2C,T3M,T3R,T3C)

*     COPY THE AR PARAMETERS INTO THE RETURN ARGUMENT
```

```
      MAT2(1,1) = 1.0
      DO 95 L = 1,3
         MAT2(L + 1,1) = T3M(L,1)
*        MAT2(L,1) = T3M(L-1,1)
*        WRITE (3,*) MAT2(L,1)
95    CONTINUE

      RETURN
      END

*     ********************************************
      SUBROUTINE ERROR(MAT1,M1R,M1C,MAT2,M2R,M2C,ITNUM)
*     ********************************************

* THIS SUBROUTINE CALCULATES THE STOPPING PARAMETER.

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL MAT1(M1R,M1C),MAT2(M2R,M2C)
      INTEGER I,J,K

      ERVAL = 0.0
      XVAL = 0.0
      ZVAL = 0.0

      DO 90 I = 400,450
         DO 91 J = M1R,1,-1
            XVAL = XVAL + MAT1(J,1)*RAWDAT(1,I+M1R-J)
91       CONTINUE
         DO 92 K = M2R,1,-1
            ZVAL = ZVAL + MAT2(K,1)*RAWDAT(2,I+M2R-K)
92       CONTINUE

         ERVAL = ERVAL + (XVAL-ZVAL)**2
         XVAL = 0.0
         ZVAL = 0.0
90    CONTINUE

      E1(1,ITNUM) = ERVAL/51

      RETURN
      END

*     ********************************************
      SUBROUTINE ERR2(MAT1,M1R,MAT2,M2R,MAT3,M3R,ITNUM,
     +YSZ,USZ,DIVISR)
*     ********************************************

* THIS SUBROUTINE CALCUALTES THE COEFFICIENT ERROR.

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL MAT1(M1R,1),MAT2(M2R,1),MAT3(M3R,1)
      INTEGER I,YSZ,USZ

      ERRVAL = (1 - MAT2(1,1))**2
```

```
      DO 90 I = 1,YSZ
         ERRVAL = ERRVAL + (MAT1(I,1) + MAT2(I+1,1))**2
90    CONTINUE

      DO 92 I = 1,USZ
         ERRVAL = ERRVAL + (MAT1(I+YSZ,1) - MAT3(I,1))**2
92    CONTINUE

      E1(2,ITNUM) = SQRT(ERRVAL)/DIVISR

      RETURN
      END

*     ********************************************************
      SUBROUTINE FILL(I,J,K,L,MAT1,M1R,M1C,R2,C2,MAT2,M2R,M2C)
*     ********************************************************

* THIS ROUTINE FILLS MAT1 FROM MAT2.  POSITIONS
* IN MAT1 FROM (I,K) TO (J,L) ARE FILLED WITH AN EQUAL NUMBER
* OF ELEMENTS FROM MAT2 STARTING AT POSITION (R2,C2).

      REAL MAT1(M1R,M1C),MAT2(M2R,M2C)
      INTEGER ROW,COL,R2,C2,C22

      C22 = C2

      DO 90 ROW = I,J
         DO 91 COL = K,L
            MAT1(ROW,COL) = MAT2(R2,C2)
            C2 = C2 + 1
91       CONTINUE
         R2 = R2 + 1
         C2 = C22
90    CONTINUE

      RETURN
      END

*     ****************************************************
      SUBROUTINE LIM2(EMAX,ESTP,CEMAX,CESTP,ITERA)
*     ****************************************************

* ROUTINE CALCULATES THE LIMITS FOR THE GRAPH OF THE STOPPING
* PARAMETER AND THE COEFFICIENT ERROR.
* STOPPING PARAMETER LIMITS ARE RETURNED IN EMAX AND ESTP.
* COEFFICIENT ERROR LIMITS ARE RETURNED IN CEMAX AND CESTP.

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL EMAX,ESTP
      INTEGER ITERA

      EMAX = 0.0
      DO 90 I = 0,ITERA
         IF (E1(1,I).GT.EMAX) THEN
```

```
            EMAX = E1(1,I)
         ENDIF
 90   CONTINUE

      EMAX = 1.25 * EMAX
      ESTP = EMAX/5
      CEMAX = 0.0

      DO 91 I = 0,ITERA
         IF (E1(2,I).GT.CEMAX) THEN
            CEMAX = E1(2,I)
         ENDIF
 91   CONTINUE

      CEMAX = 1.25 * CEMAX
      CESTP = CEMAX/5
      RETURN
      END

*     ******************************
      SUBROUTINE PLOT1(ITERA,ICURVN,ICURVD,MAT1)
*     ******************************

* THIS ROUTINE GENERATES SEPARATE PLOTS OF THE MA
* AND AR PARAMETERS.  IT THEN REPLOTS THE THESE CURVES ALONG
* WITH THE STOPPING PARAMETER.  FINALLY IT PLOTS THE STOPPING
* PARAMETER AND THE COEFFICIENT ERROR ON THE SAME GRAPH.

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL X(0:1000),Y(0:1000),MAT1(10,10)
      REAL STP,RNMIN,RNSTEP,RNMAX,RITERA
      INTEGER I,J,I1R,I1C,VAL,ITERA,ICURV

      CALL LIMITS(ICURVN,ICURVD,DMAX,DMIN,DSTEP,
     +NMAX,NMIN,NSTEP,ITERA)
      CALL LIM2(EMAX,ESTP,CEMAX,CESTP,ITERA)

* GENERATE THE ITERATION NUMBER

      DO 90 I = 0,ITERA
         X(I) = I
         Y(I) = 0.0
 90   CONTINUE

* CALCULATE X AX[illegible]S [illegible]BELING INTERVAL
      STP = ITER[illegible]10.

* SET THE DEVICE CALLS FOR ALL OF THE PLOTS

      CALL SHERPA('MCGRAF  ','A',3)
      CALL RESET('ALL')

* SECTION 1
* THIS SECTION GRAPHS THE NUMERATOR AND DENOMINATOR
* PARAMETERS ON SEPARATE GRAPHS ON THE SAME PAGE.
```

```
* SET UP THE PLOT FOR THE NUMERATOR COEFF

      CALL PAGE(8.5,11.0)
      CALL HEIGHT(0.14)
      CALL HWROT('AUTO')
      CALL XINTAX
      CALL PHYSOR(1.5,6.0)
      CALL AREA2D(5.0,3.5)
      CALL COMPLX
      CALL YAXANG(0)
      CALL XNAME('ITERATIONS$',100)
      CALL YNAME('PARAMETER VALUES$',100)
      CALL MESSAG('(A)$',100,2.4,-0.8)
      CALL THKFRM(0.03)
      CALL FRAME
      CALL GRAF(0.,STP,ITERA,NMIN,NSTEP,NMAX)

* PLOT THE NUMERATOR PARAMETERS

      DO 93 J = 1,ICURVN
         DO 94 I = 0,ITERA
               Y(I) = ARRN(I,J)
94       CONTINUE
         CALL CURVE(X,Y,ITERA,0)
93    CONTINUE

* PLOT DASHED LINES FOR THE TRUE VALUE OF THE PARAMETERS

      CALL DASH
      DO 97 K = ICURVD,ICURVD+ICURVN-1
         DO 98 J = 0,ITERA
            Y(J) = MAT1(K,1)
98       CONTINUE
         CALL CURVE(X,Y,ITERA,0)
97    CONTINUE
      CALL ENDGR(0)

* SET UP THE PLOT FOR THE DENOMINATOR PARAMETERS

      CALL RESET('DASH')
      CALL PHYSOR(1.5,1.5)
      CALL AREA2D(5.0,3.5)
      CALL XNAME('ITERATIONS$',100)
      CALL YNAME('PARAMETER VALUES$',100)
      CALL MESSAG('(B)$',100,2.4,-0.8)
      CALL THKFRM(0.03)
      CALL FRAME
      CALL GRAF(0.,STP,ITERA,DMIN,DSTEP,DMAX)

* PLOT THE DENOMINATOR PARAMETERS

      DO 95 J = 1,ICURVD
         DO 96 I = 0,ITERA
               Y(I) = ARRD(I,J)
96       CONTINUE
```

```
      CALL CURVE(X,Y,ITERA,0)
95    CONTINUE

* PLOT DASHED LINES FOR THE TRUE VALUES OF THE DENOM PARAMETERS

      CALL DASH
      DO 99 K = 1,ICURVD-1
         DO 100 J = 0,ITERA
            Y(J) = -MAT1(K,1)
100      CONTINUE
         CALL CURVE(X,Y,ITERA,0)
99    CONTINUE

      DO 105 J = 0,ITERA
         Y(J) = 1.0
105   CONTINUE
      CALL CURVE(X,Y,ITERA,0)
      CALL ENDPL(0)

* SECTION 2
* THIS SECTION PUTS THE STOPPING PARAMETER ON THE
* GRAPHS OF THE NUMERATOR AND DENOMINATOR PARAMETERS.

* SET UP THE PLOT FOR THE NUMERATOR PARAMETERS

      CALL RESET('DASH')
      CALL HWROT('AUTO')
      CALL XINTAX
      CALL PHYSOR(1.5,6.0)
      CALL AREA2D(5.0,3.5)
      CALL COMPLX
      CALL YAXANG(0)
      CALL XNAME('ITERATIONS$',100)
      CALL YNAME('PARAMETER VALUES$',100)
      CALL MESSAG('(A)$',100,2.4,-0.8)
      CALL THKFRM(0.03)
      CALL FRAME
      CALL GRAF(0.,STP,ITERA,NMIN,NSTEP,NMAX)

* PLOT THE NUMERATOR PARAMETERS

      DO 200 J = 1,ICURVN
         DO 201 I = 0,ITERA
              Y(I) = ARRN(I,J)
201       CONTINUE
         CALL CURVE(X,Y,ITERA,0)
200    CONTINUE

* PLOT DASHED LINES FOR THE TRUE VALUES OF THE PARAMETERS

      CALL DASH
      DO 202 K = ICURVD,ICURVD+ICURVN-1
         DO 203 J = 0,ITERA
            Y(J) = MAT1(K,1)
203       CONTINUE
```

```
      CALL CURVE(X,Y,ITERA,0)
202   CONTINUE


*  PLOT THE STOPPING PARAMETER ON THE SAME GRAPH

      CALL DOT
      CALL YGRAXS(0.0,ESTP,EMAX,3.5,'STOPPING PARAMETER$',
     +-100,5.0,0.0)

      DO 204 J = 0,ITERA
         Y(J) = E1(1,J)
204   CONTINUE
      CALL CURVE(X,Y,ITERA,0)
      CALL ENDGR(0)


* SET UP THE PLOT FOR THE DENOMINATOR PARAMETERS

      CALL RESET('DOT')
      CALL PHYSOR(1.5,1.5)
      CALL AREA2D(5.0,3.5)
      CALL XNAME('ITERATIONS$',100)
      CALL YNAME('PARAMETER VALUES$',100)
      CALL MESSAG('(B)$',100,2.4,-0.8)
      CALL THKFRM(0.03)
      CALL FRAME
      CALL GRAF(0.,STP,ITERA,DMIN,DSTEP,DMAX)

* PLOT THE DENOMINATOR PARAMETERS

      DO 205 J = 1,ICURVD
         DO 206 I = 0,ITERA
            Y(I) = ARRD(I,J)
206      CONTINUE
         CALL CURVE(X,Y,ITERA,0)
205   CONTINUE

* PLOT DASHED LINES FOR THE TRUE VALUES OF THE PARAMETERS

      CALL DASH
      DO 207 K = 1,ICURVD-1
         DO 208 J = 0,ITERA
            Y(J) = -MAT1(K,1)
208      CONTINUE
         CALL CURVE(X,Y,ITERA,0)
207   CONTINUE

      DO 209 J = 0,ITERA
         Y(J) = 1.0
209   CONTINUE
      CALL CURVE(X,Y,ITERA,0)

* PLOT THE STOPPING PARAMETER ON THE SAME GRAPH

      CALL DOT
```

```
      CALL YGRAXS(0.0,ESTP,EMAX,3.5,'STOPPING PARAMETER$',
     +-100,5.0,0.0)

      DO 210 J = 0,ITERA
         Y(J) = E1(1,J)
210   CONTINUE
      CALL CURVE(X,Y,ITERA,0)
      CALL ENDPL(0)

* SECTION 3
* THIS SECTION PLOTS THE STOPPING PARAMETER AND THE COEFFICIENT
* ERROR ON THE SAME GRAPH.

* SETUP THE PLOT FOR THE STOPPING PARAMETER

      CALL DOT
      CALL HWROT('AUTO')
      CALL PHYSOR(1.5,6.0)
      CALL AREA2D(5.0,3.5)
      CALL XNAME('ITERATIONS$',100)
      CALL YNAME('STOPPING PARAMETER$',100)
      CALL THKFRM(0.03)
      CALL FRAME
      CALL GRAF(0.,STP,ITERA,0.,ESTP,EMAX)


      DO 306 J = 0,ITERA
         Y(J) = E1(1,J)
306   CONTINUE
      CALL CURVE(X,Y,ITERA,0)


* PLOT THE COEFFICIENT ERROR ON THE SAME GRAPH

      CALL CHNDOT
      CALL YGRAXS(0.0,CESTP,CEMAX,3.5,'COEFFICIENT ERROR$',
     +-100,5.0,0.0)

      DO 307 J = 0,ITERA
         Y(J) = E1(2,J)
307   CONTINUE
      CALL CURVE(X,Y,ITERA,0)
      CALL ENDPL(0)
      CALL DONEPL
      RETURN
      END

*     ***************************************
      SUBROUTINE SAVE(VAL,M,K,MAT1,M1R,M1C)
*     ***************************************

* ROUTINE SAVES PARAMETER ESTIMATES IN EITHER ARRD OR ARRN
* DEPENDING UPON THE VALUE OF VAL.

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
```

```
      REAL MAT1(M1R,M1C)
      INTEGER I,J,K,M,VAL

      DO 90 I = 1,M1R
         DO 900 J = 1,M1C
            IF (VAL.EQ.1) THEN
               ARRN(K,I) = MAT1(I,J)
            ELSEIF (VAL.EQ.2) THEN
               ARRD(K,I) = MAT1(I,J)
            ENDIF
900      CONTINUE
90    CONTINUE

      RETURN
      END
```

C. SUBPROGRAMS COMMON TO BOTH SYSTEM IDENTIFICATION ALGORITHMS

```
*     ****************************************************
      SUBROUTINE ADD (MAT1,IR1,IC1,MAT2,IR2,IC2,RMAT,IRR,IRC)
*     ****************************************************

*     THIS SUBROUTINE ADDS TWO EQUAL SIZE MATRICIES AND PUTS THE RESULT
*     IN A THIRD MATRIX.

      REAL MAT1(IR1,IC1),MAT2(IR2,IC2),RMAT(IR1,IC1)
      INTEGER I,J,IRR,IRC

      DO 92 I=1,IR1
         DO 920 J=1,IC1
            RMAT(I,J) = MAT1(I,J) + MAT2(I,J)
920      CONTINUE
92    CONTINUE
      IRR = IR1
      IRC = IC1
      RETURN
      END

*     *************************************
      SUBROUTINE INIT(MAT1,M1R,M1C,INITVL)
*    *************************************

*     THIS SUBOUTINE INITIALIZES A MATRIX TO INITVL

      REAL MAT1(M1R,M1C),INITVL
      INTEGER I,J

      DO 94 I=1,M1R
         DO 95 J=1,M1C
            MAT1(I,J)=INITVL
95       CONTINUE
94    CONTINUE
      RETURN
```

```
      END

*     ************************************
      SUBROUTINE INITD(MAT1,M1R,M1C,INITVL)
*     ************************************

*     THIS SUBROUTINE INITIALIZES A MATRIX TO AS A DIAGONAL MATRIX
*     WHOSE DIAGONAL ELEMENTS EQUAL INITVL.

      REAL MAT1(M1R,M1C),INITVL
      INTEGER I,J

      DO 94 I=1,M1R
         DO 95 J=1,M1C
            IF (I.EQ.J) THEN
               MAT1(I,J)=INITVL
            ELSE
               MAT1(I,J)=0.0
            ENDIF
95       CONTINUE
94    CONTINUE
      RETURN
      END

*     ***************************************************
      SUBROUTINE LIMITS(NSZ,DSZ,DMAX,DMIN,DSTEP,NMAX,NMIN,NSTEP,ITERA)
*     ***************************************************

*     ROUTINE CALCULATES THE LIMITS FOR THE GRAPHS
*     CALCULATES DENOMINATOR AND NUMERATOR LIMITS SEPARATELY
*     IN PREPARATION FOR MAKING TWO GRAPHS

      COMMON /D/ RAWDAT(2,0:2000),E1(2,0:1000),
     +ARRD(0:1000,5),ARRN(0:1000,5)
      REAL DMAX,DMIN,DSTEP,NMAX,NMIN,NSTEP
      INTEGER DSZ,NSZ

*     CALCULATE THE DENOMINATOR LIMITS

      DMAX = 1.0
      DMIN = 0.0

      DO 90 I = 1,DSZ
         DO 91 J = 0,ITERA
         IF ((ARRD(J,I)).GT.DMAX) THEN
            DMAX = ARRD(J,I)
         ENDIF
         IF ((ARRD(J,I)).LT.DMIN) THEN
            DMIN = ARRD(J,I)
         ENDIF
91       CONTINUE
90    CONTINUE

      IF (DMAX.GT.0) THEN
         DMAX = 1.25 * DMAX
      ELSE
```

```
         DMAX = 0.0
      ENDIF

      IF (DMIN.GT.0) THEN
         DMIN = 0.0
      ELSE
         DMIN = 1.25 * DMIN
      ENDIF
      DSTEP = (DMAX - DMIN)/5

*     CALCULATE THE NUMERATOR LIMITS

      NMAX = 0.0
      NMIN = 0.0
      DO 92 I = 1,NSZ
         DO 93 J = 0,ITERA
         IF (ARRN(J,I).GT.NMAX) THEN
            NMAX = ARRN(J,I)
         ENDIF

         IF (ARRN(J,I).LT.NMIN) THEN
            NMIN = ARRN(J,I)
         ENDIF
93       CONTINUE
92    CONTINUE

      IF (NMAX.GT.0) THEN
         NMAX = 1.25 * NMAX
      ELSE
         NMAX = 0.0
      ENDIF

      IF (NMIN.GT.0) THEN
         NMIN = 0.0
      ELSE
         NMIN = 1.25 * NMIN
      ENDIF
      NSTEP = ABS(NMAX - NMIN)/5

      RETURN
      END

*     ****************************************************
      SUBROUTINE MULTI (MAT1,M1R,M1C,MAT2,M2R,M2C,RMAT,M3R,M3C)
*     ****************************************************

*     ROUTINE MULTIPLIES TWO MATRICES AND PUT THE RESULT IN A
*     THIRD MATRIX.

      REAL MAT1(M1R,M1C),MAT2(M2R,M2C),RMAT(M1R,M2C)
      INTEGER I,J,K,IRR,IRC

      CALL INIT(RMAT,M1R,M2C,0.0)

      DO 91 I=1,M1R
         DO 910 J=1,M2C
```

```
            DO 9100 K=1,M1C                                          SUB01650
               RMAT(I,J)=RMAT(I,J) +MAT1(I,K)*MAT2(K,J)              SUB01660
9100        CONTINUE                                                 SUB01670
910       CONTINUE                                                   SUB01680
91      CONTINUE                                                     SUB01690
        M3R = M1R                                                    SUB01700
        M3C = M2C                                                    SUB01710
        RETURN                                                       SUB01720
        END                                                          SUB01730
                                                                     SUB01740
*       ******************************                               IVA04910
        SUBROUTINE PRMAT(MAT1,I1R,I1C)                               IVA04920
*       ******************************                               IVA04930

*       SUBROUTINE PRINTS A MATRIX OUT TO THE FILE DEFINED
*       AS UNIT 3

        REAL MAT1(10,10)                                             IVA04970
        INTEGER I,J                                                  IVA04980
                                                                     IVA04990
          DO 92 I = 1,I1R                                            IVA05010
            WRITE(3,302) (MAT1(I,J),J = 1,I1C)                       IVA05020
302         FORMAT (7(2X,F8.5))                                      IVA05030
92        CONTINUE                                                   IVA05040
        RETURN                                                       IVA05050
        END                                                          IVA05060
                                                                     IVA05070
*       ******************************                               SUB01760
        SUBROUTINE RDMAT(MAT1,M1R,M1C)                               SUB01770
*       ******************************                               SUB01780
                                                                     SUB01790
*       ROUTINE READS A MATRIX FROM FILE SPECIFIED AS UNIT 4.        SUB01800
                                                                     SUB01810
        REAL MAT1(M1R,M1C)                                           SUB01840
        INTEGER I,J                                                  SUB01850
                                                                     SUB01860
*       READ IN MATRIX                                               SUB01870
                                                                     SUB01880
          DO 92 I = 1,M1R                                            SUB01890
            READ (4,*) (MAT1(I,J),J=1,M1C)                           SUB01900
C301         FORMAT(10F3.1)                                          SUB01910
92        CONTINUE                                                   SUB01920
        RETURN                                                       SUB01930
        END                                                          SUB01940
                                                                     SUB01950
*       ***************************************************          SUB01970
        SUBROUTINE SHIFT(MAT1,M1R,M1C,DSIZE,NSIZE,OUTDAT,INDAT)      SUB01980
*       ***************************************************          SUB01990
                                                                     SUB02000
*       ROUTINE SHIFTS NEW INPUT AND OUTPUT VALUES INTO DATA VECTOR  SUB02010
*       OF THE TEST SYSTEM.  THE OLDEST VALUES ARE LOST TO MAKE      SUB02020
*       ROOM FOR THE NEW VALUES.                                     SUB02030
                                                                     SUB02060
        REAL MAT1(M1R,M1C),OUTDAT(1),INDAT(1)                        SUB02070
        INTEGER J,DSIZE,NSIZE,NSTART                                 SUB02080
                                                                     SUB02090
```

```
      NSTART = DSIZE + 1                                                SUB02100
      DO 92 J = DSIZE,2,-1                                              SUB02110
         MAT1(1,J) = MAT1(1,J-1)                                        SUB02120
92    CONTINUE                                                          SUB02130
      MAT1(1,1) = OUTDAT(1)                                             SUB02140
                                                                        SUB02150
      DO 93 J = M1C,NSTART+1,-1                                         SUB02160
         MAT1(1,J) = MAT1(1,J-1)                                        SUB02170
93    CONTINUE                                                          SUB02180
      MAT1(1,NSTART) = INDAT(1)                                         SUB02190
      WRITE(12,19) (MAT1(1,J),J=1,M1C)                                  SUB02200
19    FORMAT (7(1X,F8.4))                                               SUB02210
                                                                        SUB02220
      RETURN                                                            SUB02230
      END                                                               SUB02240
                                                                        SUB02260
*     ******************************************                        SUB02270
      SUBROUTINE SMULTI (CONST,MAT1,M1R,M1C)                            SUB02280
*     ******************************************                        SUB02290
                                                                        SUB02300
*     ROUTINE MULTIPLIES A MATRIX BY A SCALAR.                          SUB02310
                                                                        SUB02320
      REAL MAT1(M1R,M1C),CONST                                          SUB02350
      INTEGER I,J                                                       SUB02370
                                                                        SUB02380
      DO 93 I=1,M1R                                                     SUB02390
         DO 930 J=1,M1C                                                 SUB02400
            MAT1(I,J) = MAT1(I,J) * CONST                               SUB02410
930      CONTINUE                                                       SUB02420
93    CONTINUE                                                          SUB02430
                                                                        SUB02440
      RETURN                                                            SUB02450
      END                                                               SUB02460
                                    SUB02480
```

LIST OF REFERENCES

1. Ljung, L., *System Identification: Theory for the User*, Prentice-Hall, Incorporated, 1987.

2. Marple, S.L., *Digital Spectral Analysis with Applications*, Prentice-Hall, Incorporated, 1987.

3. Hsia, T.C., *System Identification*, D.C. Heath and Company, 1977.

4. Friedlander. B., "Instrumental Variable Methods for ARMA Spectral Estimation", *IEEE Transactions on Accoustics, Speech, and Signal Processing*, v. 31, pp. 404-415, April 1983.

5. Friedlander, B., "System Identification Techniques for Adaptive Noise Cancelling", *IEEE Transactions on Accoustics, Speech, and Signal Processing*, v. 30, pp. 699-709, October 1982.

6. Whittle, P., "On the fitting of multivariate autoregressions, and the approximate canonical factorization of a spectral density matrix", *Biometrika*, v. 50, pp. 129-134, 1963.

7. Perry, F.A. and Parker, S.R., "Recursive Solutions for Zero Pole Modeling", paper presented at the Asilomar Conference on Circuits and Systems, Thirteenth, Monterey, CA, November, 1979.

8. Haykin, S., *Introduction to Adaptive Filters*, Macmillan Publishing Company, 1984.

9. Graupe, D., *Identification of Systems*, Van Nostrand Reinhold Company, 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code 62 Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Dr. Murali Tummala, Code 62Tu Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93943-5000	2
5.	Dr. Lawrence J. Ziomek, Code 62Zm Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Dr. Rabi N. Madan Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5000	1
7.	Commander Naval Ocean Systems Command Attn: Dr. Eugene P. Cooper, Code 013 San Diego, CA 92152-5000	1
8.	Commandant (G-PTE) United States Coast Guard 2100 Second Street, N.W. Washington, DC 20593	2
9.	United States Coast Guard Academy Department of Electrical Engineering Attn: LT Paul S. Dal Santo New London, CT 06320	2