



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1981

An Adaptive Kalman Identifier and its
application to linear and non-linear ARMA modeling.

Mayoral, Leopoldo M.

Monterey, California. Naval Postgraduate School

<https://hdl.handle.net/10945/20408>

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

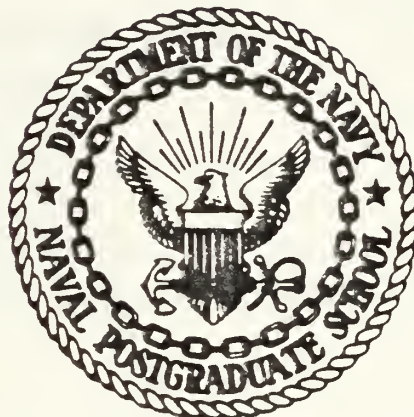
AN ADAPTIVE KALMAN IDENTIFIER
AND ITS APPLICATION TO
LINEAR AND NON-LINEAR ARMA MODELING

Leopoldo M. Mayoral



NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

AN ADAPTIVE KALMAN IDENTIFIER
AND ITS APPLICATION TO
LINEAR AND NON-LINEAR ARMA MODELING

by

Leopoldo M. Mayoral

March 1981

Thesis Advisor:

S.R. Parker

Approved for public release; distribution unlimited.

T199346

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Adaptive Kalman Identifier and Its Application to Linear and Non-Linear ARMA Modeling		5. TYPE OF REPORT & PERIOD COVERED Engineer's Thesis; March 1981
7. AUTHOR(s) Leopoldo M. Mayoral		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE March 1981
		13. NUMBER OF PAGES 162
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Kalman Filter Autoregressive Moving-Average (ARMA) Adaptive Kalman Identifier Least Mean Squares (LMS) Filter ARMA Modeling Least Mean Squares (LMS) Recursive Autogressive (AR) Filter Moving Average (MA) Adaptive Lattice Filter		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The problem of accurately replicating the parameters which define a given system for the purposes of implementing modern control strategies is important. Using an Autoregressive-Moving Average (ARMA) representation for the unknown system, a model is identified by processing input/output data to estimate the coefficients associated with the ARMA equation. Identification of unknown system parameters using Kalman filtering methods		

#20 - ABSTRACT - (CONTINUED)

was accomplished by augmenting the state vector. In this thesis the Kalman filter is formulated so that parameters can be identified explicitly. We call this approach the Adaptive Kalman Identifier (AKI).

It is shown that the Adaptive least mean square (LMS) Adaptive Recursive LMS and Adaptive Lattice filters are special suboptimal cases of the AKI. The convergence and modeling properties are compared with those of the AKI by simulation using various types of data.

With minor modification, the AKI algorithm was used to identify the linear and non-linear ARMA models of the phase locked loop (PLL). A discrete PLL using a forward Euler integration scheme was used as a source of non-linear data. The AKI technique appears to enable one to discern when a potential non-linear system enters into its non-linear mode of operation.

#19 - KEY WORDS - (CONTINUED)

System Identification
Non-Linear ARMA Modeling
Overmodeling

Approved for public release; distribution unlimited.

An Adaptive Kalman Identifier
and Its Application to
Linear and Non-Linear ARMA Modeling

by

Leopoldo M. Mayoral
Lieutenant, United States Navy
B.S., University of Texas, 1974
M.S., Naval Postgraduate School, 1980

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the
NAVAL POSTGRADUATE SCHOOL
March 1981

ABSTRACT

The problem of accurately replicating the parameters which define a given system for the purposes of implementing modern control strategies is important. Using an Autoregressive-Moving Average (ARMA) representation for the unknown system, a model is identified by processing input/output data to estimate the coefficients associated with the ARMA equation. Identification of unknown system parameters using Kalman filtering methods was accomplished by augmenting the state vector. In this thesis the Kalman filter is formulated so that parameters can be identified explicitly. We call this approach the Adaptive Kalman Identifier (AKI).

It is shown that the Adaptive least mean square (LMS), Adaptive Recursive LMS and Adaptive Lattice filters are special suboptimal cases of the AKI. The convergence and modeling properties are compared with those of the AKI by simulation using various types of data.

With minor modifications, the AKI algorithm was used to identify the linear and non-linear ARMA models of the phase locked loop (PLL). A discrete PLL using a forward Euler integration scheme was used as a source of non-linear data. The AKI technique appears to enable one to discern when a potential non-linear system enters into its non-linear mode of operation.

TABLE OF CONTENTS

I.	INTRODUCTION -----	13
	A. BACKGROUND -----	13
	B. INVESTIGATIONS AND CONTRIBUTIONS -----	16
	C. ORGANIZATION -----	17
II.	FORMULATION OF THE PROBLEM -----	18
	A. DETERMINISTIC CASE -----	18
	B. NON DETERMINISTIC CASE -----	20
	1. Autoregressive Form ($a_i = 0$) -----	24
	2. Moving Average Form ($b_i = 0$) -----	27
	3. Autoregressive-Moving Average Form -----	30
	C. OBSERVATIONS -----	33
III.	COMPARISON BETWEEN THE ADAPTIVE MA KALMAN IDENTIFIER AND WIDROW LMS ADAPTIVE FILTER -----	36
	A. PRELIMINARIES -----	36
	B. THE COMPARISON -----	38
	C. OBSERVATIONS -----	41
IV.	COMPARISON BETWEEN THE ADAPTIVE KALMAN IDEN- TIFIER AND FEINTUCH ADAPTIVE RECURSIVE LMS FILTER -----	43
	A. PRELIMINARIES -----	43
	B. THE COMPARISON -----	47
	C. OBSERVATIONS -----	53
V.	SYNTHETIC DATA GENERATING PLANTS -----	54
	A. AUTOREGRESSIVE-MOVING AVERAGE DATA -----	54
	1. Moving Average Data (Case 1) -----	55
	2. Autoregressive Data (Case 2) -----	55
	3. Autoregressive-Moving Average Data (Case 3) -----	56

B.	PHASE LOCKED LOOP DATA -----	57
VI.	SOFTWARE -----	67
A.	ADAPTIVE KALMAN IDENTIFIER -----	67
B.	LMS ADAPTIVE FILTER -----	69
C.	ADAPTIVE RECURSIVE LMS FILTER -----	73
VII.	NON-LINEAR IDENTIFICATION -----	75
VIII.	FINDINGS, CONCLUSION -----	81
A.	ORDER OF THE UNKNOWN SYSTEM IS KNOWN -----	81
1.	AKI vs LMS Adaptive Filter -----	81
2.	AKI Applied to Autoregressive Data -----	91
3.	AKI Applied to ARMA Data -----	93
4.	AKI vs Adaptive Recursive LMS Filter ----	98
5.	AKI Applied to PLL Data (Linear Region) -	103
6.	AKI Applied to PLL Data (Non-linear Region) -----	103
B.	ORDER OF THE UNKNOWN SYSTEM IS NOT KNOWN (OVERMODELING) -----	109
1.	AKI vs LMS Adaptive Filter -----	109
2.	AKI Applied to AR and ARMA Data -----	114
C.	CONCLUSION -----	115
D.	TOPICS FOR FURTHER CONSIDERATION -----	118
APPENDIX A:	The Discrete Weiner Problem -----	120
APPENDIX B:	Program ARMA -----	124
APPENDIX C:	Program PLL -----	126
APPENDIX D:	Program ADAPTSN -----	131
APPENDIX E:	Program LMS -----	147
APPENDIX F:	Program LMSR -----	151

LIST OF REFERENCES ----- 158

INITIAL DISTRIBUTION LIST ----- 161

LIST OF TABLES

TABLE

6.1	AKI Flag Meanings -----	68
6.2	Valid Flag Combinations -----	69
7.1	Non-linear Weighting Coefficients Calculated Using Three Methods -----	80
8.1	Actual vs AKI Estimates for Coefficients of Equation 8.3 -----	93
8.2	Actual vs AKI Estimates for Coefficients of Equation 8.4 at the 371st Iteration -----	94
8.3	Actual vs AKI and Adaptive Recursive LMS Filter Estimates for Coefficients of Equation 8.6 -----	100
8.4	Actual vs AKI Estimates of the ARMA Repre- sentation for the PLL (Linear Region) -----	103
8.5	Overmodeling of a MA Process Using the AKI (k = 250) -----	113
8.6	Overmodeling of a MA Process Using the AKI of orders AR = 2, MA = 6 -----	113

LIST OF FIGURES

FIGURE

2.1	Adaptive Kalman Identifier -----	34
3.1	LMS Adaptive Filter Implementation of System Identification -----	37
4.1	Adaptive Recursive LMS Filter Applied to System Identification -----	48
5.1	ARMA Digital Plant -----	56
5.2	PLL Block Diagram -----	58
5.3	Final PLL Block Diagram -----	59
5.4	PLL Step Response, Linear Region -----	60
5.5	PLL Step Response, Non-linear Region -----	61
5.6	Discrete PLL Block Diagram -----	62
5.7	Programmable PLL Implementing Forward Euler Integration -----	63
5.8	Discrete PLL Step Response, Linear Region ----	64
5.9	Discrete PLL Step Response, Non-linear Region -----	65
6.1	AKI General Program Flow -----	70
6.2	LMS Filter Program Flow -----	72
6.3	LMSR Filter Program Flow -----	74
7.1	PLL, Third Order Taylor Series Approximation -	76
8.1	LMS Adaptive Filter Weights, $k_s = -.200$ -----	83
8.2	LMS Adaptive Filter Weights, $k_s = -.300$ -----	84
8.3	LMS Adaptive Filter Weights, $k_s = -.400$ -----	85
8.4	LMS Adaptive Filter Weights, $k_s = -.500$ -----	86
8.5a	LMS Adaptive Filter Weights, $k_s = -.600$ -----	87

FIGURE

8.5b	LMS Adaptive Filter Weights, $k_s = -.200$, with Increased Measurement Noise -----	88
8.6	AKI Moving Average Plant Coefficients -----	89
8.7	LMS Adaptive Filter Weights -----	89
8.8	AKI Averaged Gains for Moving Average Model --	90
8.9	AKI Moving Average Instantaneous Gains -----	90
8.10	AKI Autoregressive Plant Coefficients -----	92
8.11	AKI Autoregressive Averaged Gains -----	92
8.12	AKI(4,5) Computed Coefficients, Perry's Example -----	95
8.13	AKI(4,5) Average Gain Values -----	95
8.14a	Pole/zero Models Produced by AKI and Lattice Algorithms for a Plant with the Character- istic Transfer Function of Equation (8.4) ----	96
8.14b	Pole/zero Models Produced by AKI and Lattice Algorithms for a Plant with the Character- istic Transfer Function of Equation (8.4) at the 371st Iteration -----	97
8.15	Instantaneous Gain Values, AKI(4,5) Model ----	98
8.16	AKI(2,2) Computed Coefficients, Feintuch Example -----	101
8.17	LMSR Computed Coefficients, Feintuch Example -	101
8.18	AKI(2,2) Averaged Gains, Feintuch Example ----	102
8.19	Pole/zero Models Produced by AKI and Adaptive Recursive LMS Algorithms for a Plant with the Characteristic Transfer Function of Equation 8.6 -----	102
8.20	PLL, AKI(2,3) Computed Coefficients -----	104
8.21	Averaged Gain Values, AKI(2,3) PLL Model ----	104
8.22	PLL Root Locus Analysis vs Computed Roots of the AKI ARMA Model -----	106

FIGURE

8.23	Variation of $\bar{\alpha}$ vs Input Noise Signal Power ---	108
8.24	Comparison of $\frac{\sin x}{x}$ and P.D.F. of Input Noise Signal -----	110
8.25a	AKI(0,4) Model of a Third Order MA Process ----	112
8.25b	Fourth Order LMS Filter Weights of a Third Order MA Process -----	112
8.26	AKI(3,1) Overmodeled AKI(4,2) -----	116
8.27	AKI(3,1) Overmodeled AKI(5,5) -----	116
8.28	AKI(4,5) Overmodeled AKI(5,6) -----	117
A.1	The Discrete Weiner Problem -----	121

ACKNOWLEDGMENT

I wish to express my gratitude and appreciation for the encouragement which I received, the patience which was extended to me, and the many countless hours which my thesis advisor, Professor S.R. Parker, unselfishly spent discussing and exchanging the many ideas that went into this thesis. Without Professor Parker's guidance this work could easily have been insurmountable.

I, however, am in much greater debt to my wife, Alice, whose tenacious and unwavering love and confidence in me allowed me to successfully progress through seemingly fruitless avenues of research. Her love, patience and understanding was neverfailing.

Lastly, I thank my son, Joshua, for demonstrating to me what life is all about.

I. INTRODUCTION

The accomplishments in the area of microprocessor technology in the last decade have made a noticeable impact in the area of modern control. The desire to implement modern control theories taking advantage of these advancements, has made it imperative that the engineer attempt to mathematically replicate the system he ultimately desires to control. Efforts in this regard have, hence, generated a growing interest in the area of system identification [Ref. 1,2,3,4]. By implication this thesis concerns itself with discrete/digital signal processing.

A. BACKGROUND

System identification or modeling can be accomplished by innovative application of existing techniques which were generally considered filtering or state estimation methods [Ref. 5]. Previous research efforts have, for obvious reasons, focused on linear modeling; however, there is a rising interest in non-linear modeling methods [Ref. 6,7,8].

An attractive form for the representation of an unknown system is the Autoregressive-Moving Average (ARMA) equation,

$$y(k) = \sum_{i=0}^{\infty} a_i u(k-i) - \sum_{i=1}^{\infty} b_i y(k-i) \quad (1.1)$$

which states that the present output, $y(k)$, is a linear combination of past outputs, $y(k-i)$, and of past and present

inputs, $u(k)$. Its attractiveness lies in its linear character and easily implementable structure using microprocessor/computer algorithms. Its relationship to the state space representation of a linear system has already been established [Ref. 9,10] making it germane to consider that a system is identifiable by an equation of the form (1.1). The system identification problem thus entails identifying the coefficients a_i and b_i .

There are several methods for computing the coefficients, a_i and b_i , however, it is not the intent of this brief introduction to attempt to develop even the majority of them. Nevertheless, it is practical to present a referenced history of those methods encountered in this thesis.

Adaptive algorithms for the purpose of estimating the coefficients of (1.1) have always been of interest. Widrow, using a least means square error criterion and implementing the method of steepest descent, developed an adaptive algorithm which estimated the coefficients of the moving average process associated with equation (1.1) [Ref. 11]. That is, the moving average model,

$$y(k) = a'_0 u(k) + a'_1 u(k-1) + a'_2 u(k-2) + \dots \quad (1.2)$$

where,

$$a'_0 = a_0$$

$$a'_1 = (a_1 - a_0 b_1)$$

$$a'_2 = (a_2 - a_0 b_2) - b_1 (a_1 - a_0 b_1)$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

was identified. The LMS theory has since been extended to include block LMS filtering methods [Ref. 12,13,14] using various search techniques [Ref. 15,4(Chapter 5)]. These methods have enjoyed much popularity in the area of Linear Prediction and digital speech processing [Ref. 16,17].

The shortcoming of representing equation (1.1) by its equivalent moving average model (1.2) lies in the practical aspect of its implementation. That is, the infinite series represented by equation (1.2) must be truncated at some point resulting in an approximation which may not adequately represent equation (1.1). Hence, efforts to adaptively estimate the coefficients for both the autoregressive (b_i) and moving average (a_i) processes continued [Ref. 18,19,20] with various degrees of success. Feintuch's Adaptive Regressive LMS procedure [Ref. 18] is still a controversial issue [Ref. 19,20].

From yet a different direction, Anderson and Moore suggested that the Kalman filtering algorithms can be used as a means to identify the a_i and b_i coefficients¹ of equation (1.1) [Ref. 21, pp. 50-52]. This thesis exploits this application.

¹Anderson and Moore in fact formulate the technique to compute the coefficients a_i, b_i $i = 1, 2, \dots, p$ where $p = n+m$.

B. INVESTIGATIONS AND CONTRIBUTIONS

This thesis formulates an adaptive application of the Kalman filter to identify the coefficients of the general ARMA equation (1.1). It is shown that the LMS adaptive filter [Ref. 11] and the Adaptive Recursive LMS filter [Ref. 18] are (1) special cases of the Adaptive ARMA Kalman identifier and (2) sub-optimal with respect to the underlying least means square (LMS) error criterion upon which the LMS adaptive filter, the adaptive recursive filter and the Kalman filter are based. It is also demonstrated that the adaptive Kalman identifier has excellent convergence properties.

By comparison, it is noted that the Kalman algorithm accounts for measurement noise where the LMS algorithms do not, a heretofore unapproached problem by LMS algorithms. The results indicate that the suggested modification by Griffiths [Ref. 22] of the convergence factor, k_s , for the LMS adaptive filter is justified.

The application of the Kalman filter algorithm is extended to identification of the coefficients associated with a special case of the generalized non-linear ARMA model [Ref. 8]. The results of the non-linear simulations suggest a technique for determining when a potential non-linear system enters its non-linear operating regime. Such a technique can prove valuable when on-line performance evaluation of a known non-linear system is required.

Lastly, the connection between the Kalman filter algorithm and the lattice filter algorithm [Ref. 6] is made. An example

which demonstrates and compares the performance of both algorithms is given.

C. ORGANIZATION

Chapter II presents and exploits the Kalman filter equations emphasizing its connections with the Yule-Walker and the discrete Weiner-Hopf equations. The theory is further developed to investigate the general ARMA case. Chapter III pursues the theoretical comparisons between the discrete Weiner-Hopf equation upon which the adaptive LMS filter is based and the MA form of the Kalman filter. The theoretical comparison between the Adaptive Recursive LMS filter and the general ARMA form of the Kalman filter is made in Chapter IV. The software methods by which linear and non-linear synthetic data is generated are discussed in Chapter V. A short user's description of the data processing programs and the options provided is given in Chapter VI. A non-linear application for the identification of the parameters of a non-linear plant is presented in Chapter VII followed by a discussion and presentation of the results in Chapter VIII.

II. FORMULATION OF THE PROBLEM

Identifying the coefficients of the Autoregressive-Moving Average (ARMA) equation,

$$y(k) + b_1 y(k-1) + \dots + b_n y(k-n) = a_0 u(k) + a_1 u(k-1) + \dots + a_m u(k-m) \quad (2.1a)$$

$$y(k) = \sum_{i=0}^m a_i u(k-i) - \sum_{i=1}^n b_i y(k-i) \quad (2.1b)$$

can be formulated as an adaptive Kalman identification problem. That is, instead of using the well known Kalman Filter equations [Ref. 23,24] to recursively estimate the states of a system, one can utilize the Kalman filter equations to adaptively estimate the coefficients of either an Autoregressive (AR), Moving Average (MA), or Autoregressive-Moving Average (ARMA) process by proper definition of the quantities involved. We call this the Adaptive Kalman Identifier for obvious reasons.

A. THE DETERMINISTIC CASE

Consider for the moment that the a_i and b_i are constant. Then by collecting a sufficient number of measurements of the input $u(k)$ and the output $y(k)$, one can readily solve a set of linear equations for the a_i and b_i coefficients. The "sufficient number" that is needed is $n+m+1$ which define the

$n+m+1$ linear equations which solve the $n+m+1$ unknown coefficients. The matrix equation to be solved takes the form:

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(m) \\ y(m+1) \\ \vdots \\ y(m+n) \end{bmatrix} = \begin{bmatrix} u(0) & 0 & \dots & 0 & | & 0 & \dots & \dots & \dots & 0 \\ u(1) & u(0) & & & | & y(0) & & & & 0 \\ \vdots & \vdots & \vdots & \vdots & | & \vdots & & & & \vdots \\ u(m) & u(m-1) & & u(0) & | & y(m-1) & y(m-2) & y(0) & & 0 \\ \hline u(m+1) & & & u(1) & | & y(m) & y(m-1) & y(1) & & 0 \\ \vdots & & & & | & & & & & \vdots \\ u(m+n) & \dots & & u(m) & | & y(n-1) & y(n-2) & & & y(0) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \\ b_1 \\ b_2 \\ \vdots \\ \cdot \end{bmatrix} \quad (2.2)$$

$$\underline{y} = T \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} \quad (2.3)$$

It is interesting to note that the $(n+m+1) \times (n+m+1)$ matrix, T , is block symmetric. The solution to (2.3) is readily apparent, namely,

$$\begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = T^{-1} \underline{y} \quad (2.4)$$

where T^{-1} is the inverse matrix of T . Since we have assumed that the elements of T are perfect noiseless measurements, and that we somehow know a-priori the number of unknown

coefficients, then T is of full rank, namely $\text{rank}(T) = n+m+1$, and the inverse of T exists.

B. THE NONDETERMINISTIC CASE

Rarely can the coefficients be modeled so ideally. A more prudent and realistic model admits that the ARMA equation coefficients are subject to random perturbations. Further, it can be said that in general measurement devices introduce noise into the measurement data. Hence, the measurement model should reflect this fact. Developing the Adaptive Kalman Identifier along the same lines as [Ref. 21] we let,

$$a_i(k+1) = a_i(k) + w_i(k) \quad i = 0, 1, 2, \dots, m \quad (2.5a)$$

$$b_j(k+1) = b_j(k) + w_j(k) \quad j = 1, 2, \dots, n \quad (2.5b)$$

where the $\{w_i(k), w_j(k)\}$ are samples from a zero mean, white, gaussian random process. Additionally, we assume that the noise sources are uncorrelated,

$$E[w_{a_r}(k)w_{a_s}(k)] = 0 \quad \text{for } r \neq s \quad (2.6a)$$

$$E[w_{b_r}(k)w_{b_s}(k)] = 0 \quad \text{for } r \neq s \quad (2.6b)$$

$$E[w_{a_r}(k)w_{b_s}(k)] = 0 \quad \text{for all } r, s \quad (2.6c)$$

Similarly, allowing for noisy measurement devices, the measurement equation (2.1b), is modelled as,

$$y(k) = \sum_{i=0}^m a_i u(k-i) - \sum_{i=1}^n b_i y(k-i) + v(k) \quad (2.7)$$

where the $\{v(k)\}$ are samples from a zero mean, white, gaussian random process. It is also assumed that,

$$E[w_i(k) v(k)] = 0 \text{ for all } i \quad (2.8a)$$

$$E[w_j(k) v(k)] = 0 \text{ for all } j \quad (2.8b)$$

That is, it is assumed that the noise perturbations associated with one coefficient are independent of the noise perturbations associated with any other coefficient, and that the measurement noise and the coefficient perturbations are independent.

At this point it is necessary to use judgment and experience and utilize all the information known about the physical system which equation (2.7) represents to assign variances for the random processes $\{w_i(k)\}$, $\{w_j(k)\}$ and $\{v(k)\}$. Let

$$Q = E \left\{ \begin{array}{c} \begin{bmatrix} w_i(k) \\ w_j(k) \end{bmatrix} \\ [w_i(k) \ w_j(k)] \end{array} \right\} \quad (2.9a)$$

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \quad (2.9b)$$

where, $Q_1 = \text{diag} (\sigma_{w_i}^2)$, $Q_2 = \text{diag} (\sigma_{w_j}^2)$ and,

$$R = E[v(k) v(k)] \quad (2.9c)$$

and $E[x]$ is the expected or average value of x .

Instead of using x 's denoting the commonly used notation for state variables, we retain the flavor of the problem by using the a_i 's and b_i 's as the "states" of the Adaptive Kalman Identifier. It is hoped that a more meaningful understanding of the Adaptive Kalman Identifier may thus be gained. Therefore, define the state vector,

$$\begin{bmatrix} a_0(k) \\ a_1(k) \\ \vdots \\ a_m(k) \\ \hline b_1(k) \\ b_2(k) \\ \vdots \\ b_n(k) \end{bmatrix} = \begin{bmatrix} \underline{a} \\ \hline \underline{b} \end{bmatrix} \quad (2.10)$$

Combining equations (2.5) and (2.10), we have the discrete, first order Gauss-Markov process [Ref. 3],

$$\begin{bmatrix} \underline{a}(k+1) \\ \hline \underline{b}(k+1) \end{bmatrix} = \begin{bmatrix} \underline{a}(k) \\ \hline \underline{b}(k) \end{bmatrix} + \underline{w}(k) \quad (2.11)$$

where,

$$\underline{w}(k) = \begin{bmatrix} \underline{w}_1(k) \\ \hline \underline{w}_j(k) \end{bmatrix} . \quad (2.12)$$

Note that $\underline{w}(k)$ is an $n+m+1$ vector whose elements are a concatenation of the noise sequences associated with the a_i and b_i . To complete the Adaptive Kalman Identifier formulation we define the measurement vector,

$$H(k) = [u(k) \ u(k-1) \ \dots \ u(k-m) \ -y(k-1) \ \dots \ -y(k-n)] \quad (2.13)$$

and its associated measurement equation,

$$y(k) = H(k) \begin{bmatrix} \underline{a}(k) \\ \text{-----} \\ \underline{b}(k) \end{bmatrix} + v(k). \quad (2.14)$$

Then the solution to the Adaptive Kalman Identifier problem [Ref. 21] is,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \text{-----} \\ \hat{\underline{b}}(k+1|k) \end{bmatrix} = [I - K(k)H(k)] \begin{bmatrix} \hat{\underline{a}}(k|k-1) \\ \text{-----} \\ \hat{\underline{b}}(k|k-1) \end{bmatrix} + K(k)y(k) \quad (2.15a)$$

$$K(k) = P(k|k-1) H^T(k) [H(k) P(k|k-1) H^T(k) + R]^{-1} \quad (2.15b)$$

$$P(k+1|k) = P(k|k-1) - K(k)H(k)P(k|k-1) + Q. \quad (2.15c)$$

Equations (2.15) are initialized by assuming an initial value for the coefficients (2.10) and assigning to our assumption a measure of our confidence in the initial guess. That is, we pick the values,

$$\begin{bmatrix} \hat{\underline{a}}(0|-1) \\ \text{-----} \\ \hat{\underline{b}}(0|-1) \end{bmatrix} \text{ and } P(0|-1) \quad (2.16)$$

where $P(0|-1)$ is defined as the error covariance of the coefficients,

$$P(k|k-1) = E \left\{ \left[\begin{array}{c} \underline{a}(k) \\ \text{---} \\ \underline{b}(k) \end{array} - \begin{array}{c} \hat{\underline{a}}(k|k) \\ \text{---} \\ \hat{\underline{b}}(k|k) \end{array} \right] \left[\begin{array}{c} \underline{a}(k) \\ \text{---} \\ \underline{b}(k) \end{array} - \begin{array}{c} \hat{\underline{a}}(k|k) \\ \text{---} \\ \hat{\underline{b}}(k|k) \end{array} \right]^T \right\} \quad (2.17)$$

for $k = 0$. Equation (2.15b) is generally referred to as the Kalman gain. Two special cases are of interest: case (1); $a_i = 0$ for all i , and case (2); $b_i = 0$ for all i .

1. Autoregressive Form ($a_i = 0$)

For case 1, equation (2.7) takes the form,

$$y(k) = - \sum_{i=1}^n b_i y(k-i) + v(k). \quad (2.18)$$

This is a recursive equation stating that the present output is a linear combination of past outputs corrupted with additive gaussian noise, $v(k)$. Equation (2.18) is more formally recognized as an autoregressive process of order n [Ref. 25, 26]. Writing the Kalman solution for the coefficients of the AR process,

$$\hat{\underline{b}}(k+1|k) = [I - K(k)K(k)] \hat{\underline{b}}(k|k-1) + K(k)y(k) \quad (2.19a)$$

$$K(k) = P(k|k-1) H^T(k) [H(k)P(k|k-1) H^T(k) + R]^{-1} \quad (2.19b)$$

$$P(k+1|k) = P(k|k-1) - K(k)H(k)P(k|k-1) + Q. \quad (2.19c)$$

Alternate forms of the Kalman equations given by Maybeck [Ref. 27] are,

$$\begin{aligned} \hat{\underline{b}}(k+1|k) &= P(k|k-1) P(k|k-1)^{-1} \hat{\underline{b}}(k|k-1) \\ &+ P(k+1|k) H^T(k) R^{-1} y(k) \end{aligned} \quad (2.20a)$$

$$P(k+1|k) = (P(k|k-1))^{-1} + H^T(k) R^{-1} H(k)^{-1}. \quad (2.20b)$$

We can model the fact that we have no a-priori knowledge about the initial values of the coefficients by letting,

$$[P(0|-1)]^{-1} \approx 0. \quad (2.21)$$

Further, if we remain ignorant and totally doubt our previous estimate, then $P(k|k-1)$ is modeled as,

$$[P(k|k-1)]^{-1} \approx 0. \quad (2.22)$$

Equations (2.20) reduce to,

$$\hat{\underline{b}}(k+1|k) = [H(k) R^{-1} H^T(k)]^{-1} H(k) R^{-1} y(k), \quad (2.23)$$

the weighted least squares estimate, previously encountered by Swerling [Ref. 28,29,30], of the coefficients, \underline{b} . Carrying the analysis further and letting $R^{-1} = 1$, we have the Penrose pseudo-inverse solution [Ref. 31,32],

$$\hat{\underline{b}}(k+1|k) = [H(k) H^T(k)]^{-1} H(k) y(k) \quad (2.24)$$

Swerling [Ref. 28] has shown that if the weighting matrix in (2.23) is not the inverse of the covariance matrix of the measurement errors, then the accuracy of the estimated coefficients (2.24) will be degraded.

The aforementioned notwithstanding, we press further into the analysis of equation (2.24). The product $H(k)H^T(k)$ can be written as,

$$H(k)H^T(k) = \begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-n) \end{bmatrix} [y(k-1) \ y(k-2) \ \dots \ y(k-n)] \quad (2.25)$$

$$H(k)H^T(k) = \begin{bmatrix} y^2(k-1) & y(k-1)y(k-2) & \dots & y(k-1)y(k-n) \\ y(k-2)y(k-1) & y^2(k-2) & & \vdots \\ \vdots & & & \vdots \\ y(k-n)y(k-1) & \dots & \dots & y(k-n)y(k-n) \end{bmatrix} \quad (2.26)$$

As we let $k \rightarrow \infty$ the expected value of (2.26) becomes the autocorrelation matrix,

$$E\{\lim_{k \rightarrow \infty} H(k) H^T(k)\} = R_{YY}(k) \quad (2.27)$$

where,

$$R_{YY}(k) = \begin{bmatrix} R_{YY}(0) & R_{YY}(1) & \dots & R_{YY}(n) \\ R_{YY}(-1) & R_{YY}(0) & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ R_{YY}(-n) & \dots & \dots & R_{YY}(0) \end{bmatrix} \quad (2.28)$$

Similarly, the product $H(k)y(k)$ as $k \rightarrow \infty$ becomes

$$r_{YY}(k) = \begin{bmatrix} R_{YY}(-1) \\ R_{YY}(-2) \\ \cdot \\ \cdot \\ R_{YY}(-n) \end{bmatrix} \quad (2.29)$$

The steady state solution for the estimate of the coefficients $\hat{\underline{b}}(k+1|k)$ is,

$$\hat{\underline{b}}_{SS}(k+1|k) = R_{YY}^{-1}(k) r_{YY}(k) \quad (2.30)$$

Equation (2.30) is one of the starting points from which Perry [Ref. 6] develops his Lattice modeling algorithms. Equation (2.30) can also be recognized as the solution to the Yule-Walker or Normal equations [Ref. 26].

2. Moving Average Form ($b_i = 0$)

Referring once again to equation (2.7) and setting the coefficients $b_i = 0$ for all i we have case (2),

$$y(k) = \sum_{i=0}^m u(k-i) + v(k) \quad (2.31)$$

Since $v(k)$ is by definition noise associated with the measurement, we can combine $y(k)$ and $v(k)$ such that,

$$z(k) = y(k) - v(k) = \sum_{i=0}^m a_i u(k-i). \quad (2.32)$$

Equation (2.32) simply states that the present measurement is a linear combination of past and present inputs or by definition, a Moving Average (MA) process [Ref. 25,33]. The Adaptive Kalman Identifier estimate for the coefficients of the MA process is as before,

$$\hat{\underline{a}}(k+1|k) = [I - K(k)H(k)] \hat{\underline{a}}(k+1|k) + K(k)z(k) \quad (2.33a)$$

$$K(k) = P(k|k-1) H^T(k) [H(k)P(k|k-1) H^T(k) + R]^{-1} \quad (2.33b)$$

$$P(k+1|k) = P(k|k-1) - K(k)H(k)P(k|k-1) + Q. \quad (2.33c)$$

Using the alternate forms of the above equations we arrive at,

$$\begin{aligned} \hat{\underline{a}}(k+1|k) &= [P(k+1|k) [P(k|k-1)]^{-1}] \hat{\underline{a}}(k|k-1) \\ &\quad + [P(k+1|k) H^T(k) R^{-1}] z(k) \end{aligned} \quad (2.34a)$$

$$P(k+1|k) = [[P(k|k-1)]^{-1} + H^T(k) R^{-1} H(k)]^{-1}. \quad (2.34b)$$

Arguing as we did for the autoregressive case, no a-priori knowledge about the initial values of the moving average coefficients, implies that,

$$[P(0|-1)]^{-1} \approx 0. \quad (2.35)$$

And even if after we processed one measurement we still admitted no knowledge as to the accuracy of the previous estimate, we imply that,

$$[P(k|k-1)]^{-1} \approx 0. \quad (2.36)$$

Substituting these implications into equations (2.34) our estimate becomes,

$$\hat{\underline{a}}(k+1|k) = [H(k) R^{-1} H^T(k)]^{-1} H(k) R^{-1} z(k) \quad (2.37)$$

the by now familiar weighted least squares estimate [Ref. 28, 29,30]. If once again we allow R^{-1} to be unity, let k tend toward infinity and take the expectation of equation (2.37), one arrives at the discrete form of the Wiener-Hopf equation [Ref. 11], namely

$$\hat{\underline{a}}(k+1|k) = R_{uu}^{-1}(k) r_{uz}(k) \quad (2.38)$$

where it can easily be shown that,

$$R_{uu}(k) = \begin{bmatrix} R_{uu}(0) & R_{uu}(-1) & \dots & R_{uu}(-m) \\ R_{uu}(-1) & R_{uu}(0) & \dots & R_{uu}(1-m) \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ R_{uu}(-m) & \dots & & R_{uu}(0) \end{bmatrix} \quad (2.39)$$

and,

$$R_{uy}(k) = \begin{bmatrix} R_{uy}(0) \\ R_{uy}(-1) \\ \vdots \\ R_{uy}(-m) \end{bmatrix} \quad (2.40)$$

Equation (2.38) provided another point of departure from which Perry [Ref. 6] develops the MA Lattice modeling algorithms. Appendix A develops the Wiener-Hopf equation otherwise known as the all-zero model from yet another approach.

The all-zero model is fundamental to Widrow least mean square (LMS) adaptive filters [Ref. 11] and linear prediction theory [Ref. 16].

3. Autoregressive-Moving Average Form

Returning to the alternate form (equations 2.20) of the Kalman Filter equations (2.19) the development of the Autoregressive Moving Average (ARMA) Adaptive Kalman Identifier follows. The estimate for the a_i and b_i coefficients is,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \text{-----} \\ \hat{\underline{b}}(k+1|k) \end{bmatrix} = [P(k+1|k) [P(k|k-1)]^{-1}] \begin{bmatrix} \hat{\underline{a}}(k|k-1) \\ \text{-----} \\ \hat{\underline{b}}(k|k-1) \end{bmatrix} + [P(k+1|k) H^T(k) R^{-1}] y(k) \quad (2.41a)$$

$$P(k+1|k) = [(P(k|k-1))^{-1} + H^T(k) R^{-1} H(k)]^{-1} \quad (2.41b)$$

Progressing in the same manner as in the previous two cases, we assume,

$$[P(0|-1)]^{-1} \approx 0 \quad (2.42a)$$

$$[P(k|k-1)]^{-1} \approx 0 \quad (2.42b)$$

The estimate then becomes,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \text{-----} \\ \hat{\underline{b}}(k+1|k) \end{bmatrix} = [H^T(k) R^{-1} H(k)]^{-1} H(k) R^{-1} y(k) \quad (2.43)$$

Taking equation (2.43) a step further by letting R^{-1} be unity, letting k approach infinity and taking the expectation we have,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \text{-----} \\ \hat{\underline{b}}(k+1|k) \end{bmatrix} = \begin{bmatrix} R_{nu}(k) & -R_{uy}(k-1) \\ \text{-----} \\ -R_{uy}^T(k-1) & R_{yy}(k) \end{bmatrix}^{-1} \begin{bmatrix} R_{uy}(k) \\ \text{-----} \\ R_{yy}(k) \end{bmatrix} \quad (2.44)$$

Note that the time varying measurement vector, $H(k)$, is of the form

$$H(k) = [u(k) \ u(k-1) \ \dots \ u(k-m) \ | \ -y(k-1) \ -y(k-2) \ \dots \ -y(k-n)] \quad (2.45)$$

Shown in detail, equation (2.44) has the characteristic form (equation 2.46, next page). It is the Toeplitz and symmetric nature of equation (2.46) that is exploited by the Levinson [Ref. 34] and Lattice [Ref. 35] algorithms.

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \hline \hat{\underline{b}}(k+1|k) \end{bmatrix} = \begin{bmatrix} R_{uu}(0) & R_{uu}(-1) & \dots & R_{uu}(-m) & -R_{uy}(-1) & -R_{uy}(-2) & \dots & -R_{uy}(-n) \\ R_{uu}(-1) & R_{uu}(0) & \dots & R_{uu}(1-m) & -R_{uy}(0) & -R_{uy}(-1) & \dots & -R_{uy}(1-n) \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ R_{uu}(1-m) & \dots & R_{uu}(0) & -R_{uy}(m-1) & \dots & \dots & -R_{uy}(m-n) \\ \hline -R_{yu}(-1) & -R_{yu}(0) & \dots & -R_{yu}(m-1) & R_{yy}(0) & R_{yy}(-1) & \dots & R_{yy}(-n) \\ -R_{yu}(-2) & -R_{yu}(-1) & \dots & \cdot & R_{yy}(-1) & R_{yy}(0) & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ -R_{yu}(-n) & \dots & -R_{yu}(m-n) & R_{yy}(-n) & \dots & \dots & R_{yy}(0) \end{bmatrix}^{-1} \begin{bmatrix} R_{uy}(0) \\ R_{uy}(1) \\ \cdot \\ \cdot \\ \cdot \\ R_{uy}(m) \\ R_{yy}(1) \\ R_{yy}(2) \\ \cdot \\ \cdot \\ \cdot \\ R_{yy}(n) \end{bmatrix} \quad (2.46)$$

The Adaptive Kalman ARMA modeling technique can be best visualized by a block diagram. Referring to Figure 2.1, the unknown system is excited by a white, zero mean, gaussian noise sequence of sample values from a random process. The input and its associated output are then passed through M and N delays respectively in serial form. The parallel inputs (M+1) and the outputs (N) are concatenated to form the measurement vector $H(k)$ which is represented by equation (2.13). The inputs N and M are selected a-priori as the model orders for the Autoregressive and Moving Average processes one desires to identify. It can be easily seen that the remainder of the figure simply implements equation (2.15a). The outputs of the Adaptive Kalman Identifier are an estimate of the coefficients,

$$\begin{bmatrix} \hat{\underline{b}}(k|k) \\ \text{-----} \\ \hat{\underline{a}}(k|k) \end{bmatrix}$$

and a one step prediction, $\hat{y}(k|k)$.

C. OBSERVATIONS

In this section it has been shown that a direct connection can be established between the Adaptive Kalman Identifier and the Yule-Walker equations associated with an AR process. Secondly, the steady state Adaptive Kalman Identifier closely resembles the discrete Weiner-Hopf equation associated with the MA process when the measurement matrix, $H(k)$, is of the form,

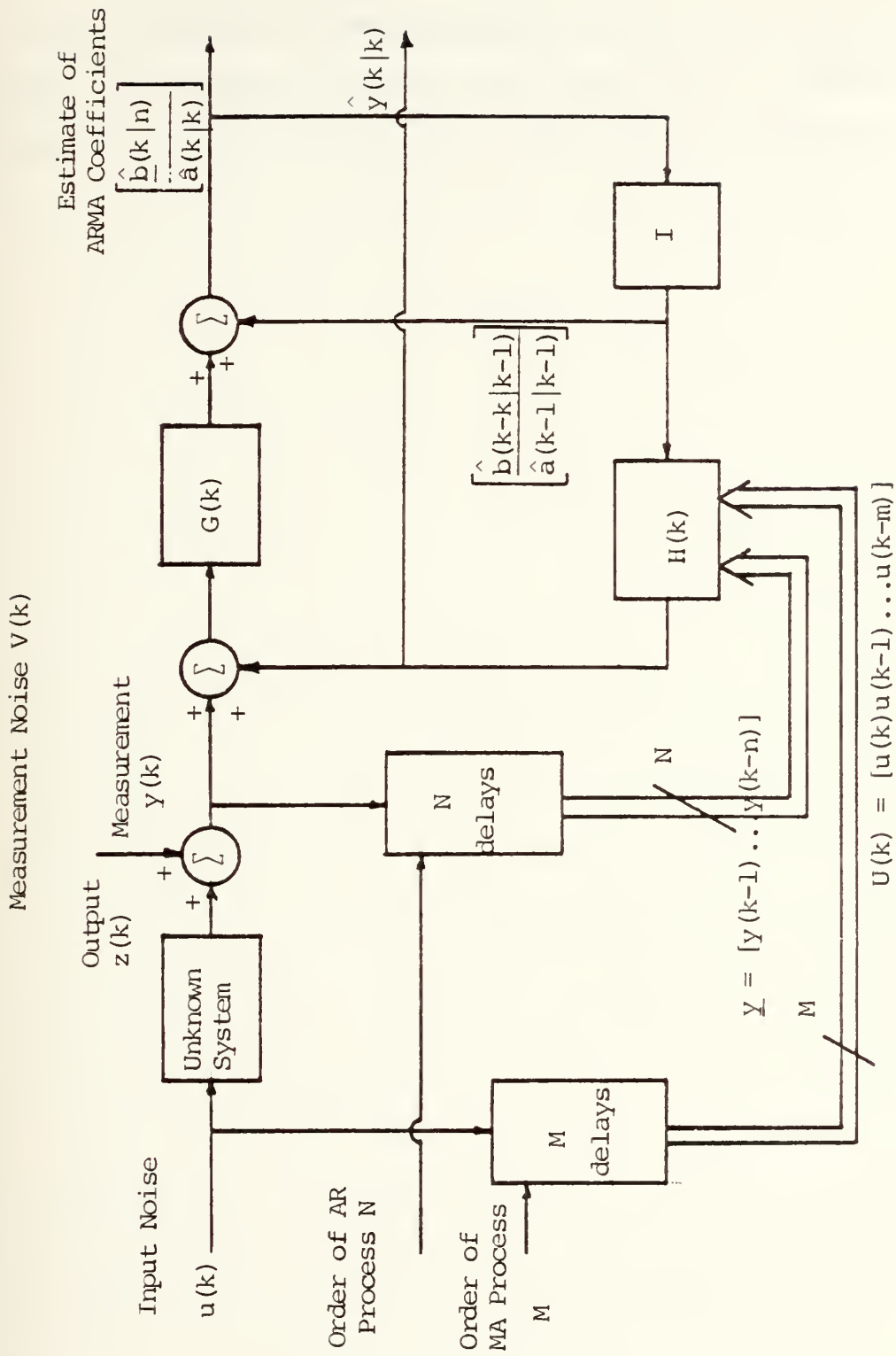


Figure 2.1. Adaptive Kalman Identifier

$$H(k) = [u(k) \ u(k-1) \ u(k-2) \ \dots \ u(k-m)]$$

And, thirdly under the same assumptions as in the previous two cases, the Adaptive Kalman ARMA Identifier is similar to the form Perry [Ref. 6] exploits using Lattice modeling.

III. COMPARISON BETWEEN THE ADAPTIVE MA KALMAN IDENTIFIER AND THE WIDROW LMS ADAPTIVE FILTER

A. PRELIMINARIES

It is instructive to investigate the similarities between the Adaptive Kalman Identifier and the Widrow LMS Adaptive Filter when the concepts are applied to system identification. Basically, the LMS algorithm implementation of system identification considers a block diagram as is shown in Figure 3.1. The output, $y(k)$, of the Adaptive filter is simply a weighted linear combination of the past and present known inputs. The same input is fed into both the unknown system to be identified and the adaptive filter. The output of the unknown system is designated the desired response, $d(k)$, from which an error signal, $\epsilon(k)$, is derived. The error signal, $\epsilon(k)$, provides the criterion through which the weights, w_i , are adjusted such that the error, $\epsilon(k+1)$, is driven toward its minimum. A more detailed analysis of the operation of the LMS algorithm can be found in Ref. 11.

The weights w_i are adjusted from time step to time step in the following manner,

$$\underline{w}(k+1) = \underline{w}(k) - 2k_s \epsilon(k) \underline{X}(k) \quad (3.1)$$

where we define the following quantities,

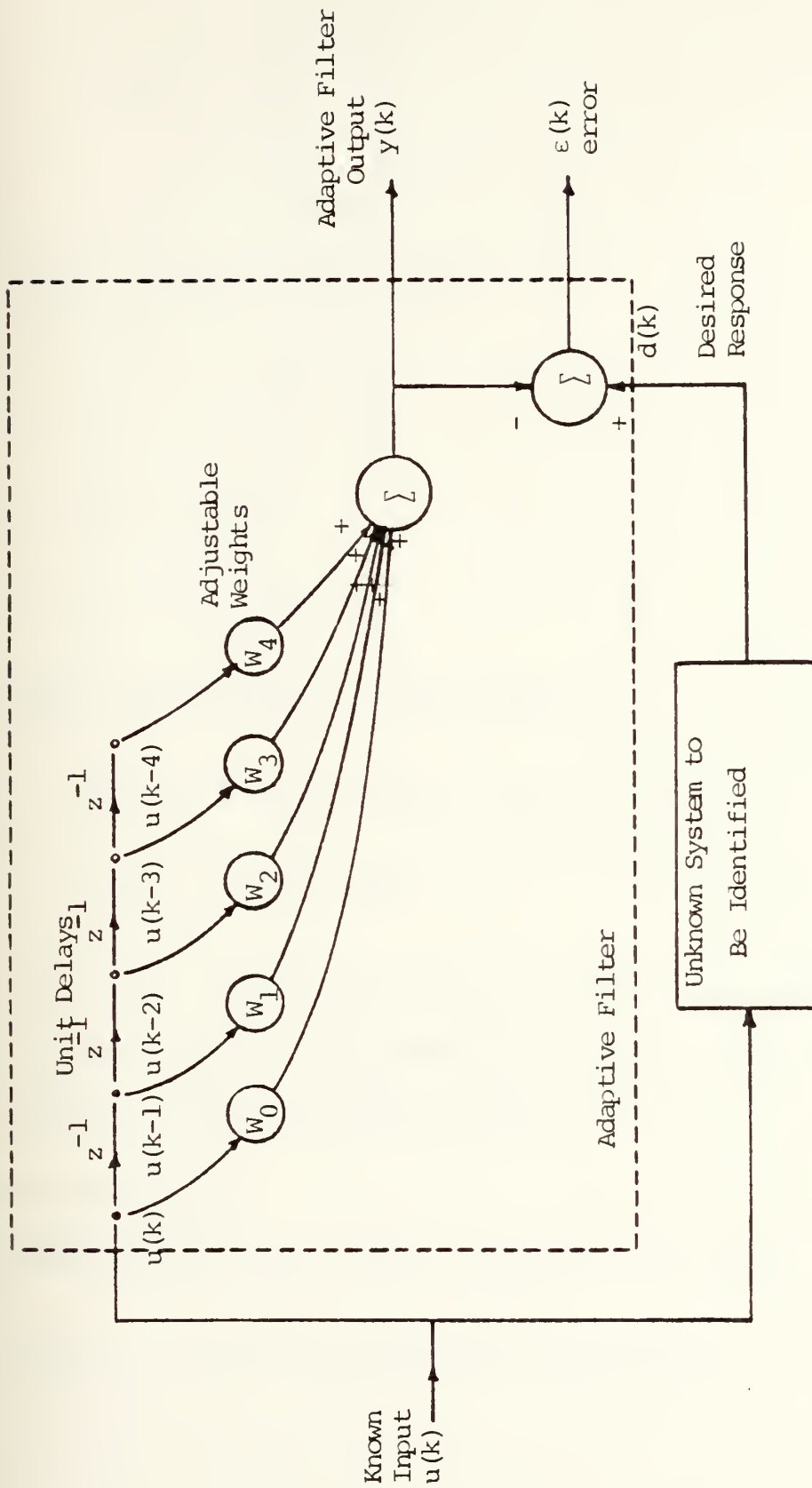


Figure 3.1. LMS Adaptive Filter Implementation of System Identification

$$\underline{w}(k) = \begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_m(k) \end{bmatrix} \quad (3.2)$$

the weight vector at time step k ,

$$\underline{X}(k) = \begin{bmatrix} u(k) \\ u(k-1) \\ \vdots \\ u(k-m) \end{bmatrix} \quad (3.3)$$

the input signal vector at time step k ,

$$\varepsilon(k) = d(k) - y(k) \quad (3.4)$$

$$y(k) = \underline{w}^T(k) \underline{X}(k) \quad (3.5)$$

the error, $\varepsilon(k)$, and the output, $y(k)$, at time step k , and, k_s , a scalar constant controlling the rate of convergence and the stability of the adaptive filter.

B. THE COMPARISON

Substituting (3.4) and (3.5) into (3.1) gives,

$$\underline{w}(k+1) = \underline{w}(k) - 2k_s \underline{X}(k) [d(k) - \underline{w}^T(k) \underline{X}(k)]. \quad (3.6)$$

Recalling the MA form of the Adaptive Kalman Identifier, equation (2.33a), rewritten here for convenience as

$$\hat{\underline{a}}(k+1|k) = \hat{\underline{a}}(k|k-1) + K(k) [z(k) - H(k) \hat{\underline{a}}(k|k-1)], \quad (3.7)$$

one can make the following associations:

$$\hat{\underline{a}}(k+1|k) \langle \Rightarrow \rangle \underline{w}(k+1) \quad (3.8a)$$

$$\hat{\underline{a}}(k|k-1) \langle \Rightarrow \rangle w(k) \quad (3.8b)$$

$$z(k) \langle \Rightarrow \rangle d(k) \quad (3.8c)$$

$$H^T(k) \langle \Rightarrow \rangle \underline{X}(k) \quad (3.8d)$$

$$K(k) \langle \Rightarrow \rangle -2k_s \underline{X}(j) \quad (3.8e)$$

Recall that for the MA form of the AKI, the measurement vector $H(k)$ represents a vector of past and present inputs. Namely,

$$H(k) = [u(k) \ u(k-1) \ \dots \ u(k-m)] . \quad (3.9)$$

Therefore the associations (3.8a)-(3.8d) are straightforward. However, it is not so clear as to what is meant by the association (3.8e). Digressing a moment to present an equivalent expression [Ref. 27] for the Kalman gain, $K(k)$,

$$K(k) = P(k|k) H^T(k) R^{-1} \quad (3.10)$$

and substituting (3.10) into the association (3.8e) and entertaining the conjecture that the quantities are equivalent

under certain conditions, we have:

$$P(k|k) H^T(k) R^{-1} = -2k_s \underline{X}(k) . \quad (3.11)$$

The conditions alluded to are (1) that the Adaptive Kalman Identifier is in steady state and (2) that the statistics of the input forcing function are stationary. Denoting the steady state error covariance, $P(k|k)$, as P_∞ , equation (3.11) can be solved for k_s ,

$$k_s I = -\frac{1}{2} R^{-1} P_\infty . \quad (3.12)$$

Invoking the entire Kalman gain history in its more popular form of equation (2.33b) and equating it to the Widrow gain (3.8e), we instead arrive at,

$$-2k_s \underline{X}(k) = P(k|k-1) H^T(k) [H(k) P(k|k-1) H^T(k) + R]^{-1} \quad (3.13a)$$

$$H^T(k) = \underline{X}(k) . \quad (3.13b)$$

Solving for the convergence factor, k_s , we obtain,

$$-2k_s \underline{X}(k) \underline{X}^T(k) = P(k|k-1) \underline{X}(k) [\underline{X}^T(k) P(k|k-1) \underline{X}(k) + R]^{-1} \underline{X}^T(k)$$

$$-2k_s I = P(k|k-1) \underline{X}(k) [\underline{X}^T(k) P(k|k-1) \underline{X}(k) + R]^{-1} \underline{X}^T(k) [\underline{X}(k) \underline{X}^T(k)]^{-1} .$$

But $[\underline{X}^T(k) P(k|k-1) \underline{X}(k) + R]^{-1}$ is a scalar. Therefore,

$$-2k_s I = \frac{P(k|k-1)}{\underline{X}^T(k) P(k|k-1) \underline{X}(k) + R} \quad (3.14)$$

Griffiths [Ref. 22] suggests that the convergence factor k_s , which he denotes as $\mu(n)$, should be chosen such that,

$$\mu(n) = \frac{\alpha}{L \hat{\sigma}_x^2(n)} \quad (3.15)$$

where $0 < \alpha < 1$ is a normalized adaptive stepsize parameter and the term $\hat{\sigma}_x^2(n)$ is an estimate of the input power level which may be computed using a geometrically-weighted average for an L weight adaptive filter,

$$\hat{\sigma}_x^2(n) = (1 - \frac{\alpha}{L}) \hat{\sigma}_x^2(n-1) + \frac{\alpha}{L} x^2(n) . \quad (3.16)$$

Comparing equation (3.14) and (3.15), one observes that if $P(k, k-1)$ is equal to the identity matrix then we obtain a term proportional to the input power. Further if measurement noise is considered negligible, $R \rightarrow 0.0$, then 3.14 and 3.15 are essentially equivalent. The salient point to make, however, is that the choice of the convergence constant, k_s , in the LMS algorithm provides no clue as to how to deal with measurement noise whereas the Adaptive MA Kalman Identifier explicitly incorporates measurement error into its algorithm.

C. OBSERVATIONS

It has been known that the LMS algorithm was suboptimal since the actual gradient of equation (3.1) is replaced by the approximation, $2\varepsilon(k) x(k)$, [Ref. 24]. However, the degree of suboptimality is difficult to quantify. In previous linear prediction research no mention was made regarding the

role of measurement noise. Equation (3.14) gives us some insight into wiser selections of the rate of convergence constant, k_s , since it takes into account the effects of measurement noise. Further, it appears that the LMS adaptive filter is a special case of the Adaptive MA Kalman Identifier.

IV. COMPARISON BETWEEN THE ADAPTIVE ARMA KALMAN IDENTIFIER AND THE ADAPTIVE RECURSIVE LMS FILTER

This chapter endeavors to investigate the similarities and differences between the Adaptive ARMA Kalman Identifier and the Adaptive Recursive LMS Filter [Ref. 18]. LMS filters [Ref. 11] have enjoyed much popularity in the recent past due to their ease of implementation, simple unimodal algorithm, robustness and ability to "adapt" to the unknown statistics of the signal environment. Being transversal in nature, they have a finite impulse response being able to produce only zeros in the input/output transfer function. One may, however, decide to model the transfer function,

$$H_1(z) = \frac{1}{1 - .9z^{-1}} \quad (4.1)$$

using a transversal filter, only to realize that a large number of delays are required in order to arrive at an adequate approximation. The germane point which is being made is that one weighted feedback tap can realize an infinite string of feed-forward coefficients. Moreover, it is very desirable to adjust the feedback weights adaptively in some optimal fashion to the statistics of the signal environment.

A. PRELIMINARIES

The approach taken by Feintuch [Ref. 18] is patterned after the analysis first presented by Widrow [Ref. 11]. A summary of Feintuch's derivation will be presented here with emphasis on its application to ARMA modeling.

Recall the general ARMA equation (2.1b),

$$y(k) = \sum_{i=0}^m a_i u(k-i) + \sum_{i=1}^n b_i y(k-i) \quad (4.2a)$$

which can be rewritten in vector notation,

$$y(k) = \underline{a}^T \underline{u} + \underline{b}^T \underline{y} \quad (4.2b)$$

where,

$$\underline{a}^T = [a_0 \ a_1 \ \dots \ a_m] \quad (4.3a)$$

$$\underline{b}^T = [b_1 \ b_2 \ \dots \ b_n] \quad (4.3b)$$

$$\underline{y} = [y(k-1) \ y(k-2) \ \dots \ y(k-n)]^T \quad (4.3c)$$

$$\underline{u} = [u(k) \ u(k-1) \ \dots \ u(k-m)]^T \quad (4.3d)$$

Given that (4.2) is the assumed mathematical description of the unknown system where \underline{u} and \underline{y} , the input and output data sequences, are known, then by solving (4.2) for \underline{a} and \underline{b} , identification of the unknown system can be made. The LMS algorithms, in general, employ a "desired" signal $d(k)$ with which to "train" the adaptive filter. If the desired signal was assumed to be the response of some unknown system to a known input signal, then the algorithm presented by Feintuch [Ref. 18] can be used as a means by which to identify the system parameters.

The first step in the Recursive LMS derivation is to form the error between the desired signal, $d(k)$, and the output

of the filter, $y(k)$,

$$e(k) = d(k) - y(k) = d(k) - \underline{a}^T \underline{u} - \underline{b}^T \underline{y} \quad (4.4)$$

Forming the square of (4.4) and taking the expectation, we have the mean square error representation of the filter,

$$\begin{aligned} E\{e^2(k)\} &= E\{d^2(k)\} + \underline{a}^T R_{uu} \underline{a} + \underline{b}^T R_{yy} \underline{b} \dots \\ &\quad - 2\underline{a}^T R_{du} - 2\underline{b}^T R_{dy} + 2\underline{a}^T R_{uy} \underline{b} \end{aligned} \quad (4.5)$$

where,

$$R_{uu}(k) = E\{u(k) u^T(k)\} \quad (4.6a)$$

$$R_{yy}(k-1) = E\{y(k-1) y^T(k-1)\} \quad (4.6b)$$

$$R_{du}(k) = E\{d(k) u(k)\} \quad (4.6c)$$

$$R_{dy}(k) = E\{d(k) y(k)\} \quad (4.6d)$$

$$R_{uy}(k-1) = E\{u(k-1) y(k-1)\} . \quad (4.6e)$$

It is assumed that the statistics (4.6) are constants allowing the gradient of (4.5) to be taken with respect to the a_i and b_i . This assumption does not stretch the theory since in practice one uses an input test signal with stationary characteristics and if the unknown system's output process is not stationary, then no identification of the system parameters can be made. The respective gradients are,

$$\frac{\partial}{\partial \underline{a}} [E\{e^2(k)\}] = 2R_{uu}(k)\underline{a} - 2R_{du}(k) + 2R_{uy}(k)\underline{b} \quad (4.7a)$$

$$\frac{\partial}{\partial \underline{b}} [E\{e^2(k)\}] = 2R_{yy}(k)\underline{b} - 2R_{dy}(k) + 2R_{uy}^T(k)\underline{a} . \quad (4.7b)$$

Since the second order statistics are assumed to be known, equations (4.7) can be solved for \underline{a} and \underline{b} by setting the gradients equal to zero. In matrix partitioned form we have,

$$\begin{bmatrix} R_{uu}(k) & R_{uy}(k) \\ \hline R_{uy}^T(k) & R_{yy}(k) \end{bmatrix} \begin{bmatrix} \underline{a} \\ \underline{b} \end{bmatrix} = \begin{bmatrix} R_{du}(k) \\ R_{dy}(k) \end{bmatrix} . \quad (4.8)$$

At this point we digress momentarily to mention that Johnson and Larimore [Ref. 19] and Widrow and McCool [Ref. 20] agree with Feintuch's derivation. The following steps, however, are controversial [Ref. 19,20].

Feintuch continues and states that in general, the statistics involved in (4.8) are not available a-priori. One method of estimating the statistics is to make the filter adaptive in an iterative fashion using the method of steepest descent. The method of steepest descent employs an algorithm of the form,

$$\underline{a}(k+1) = \underline{a}(k) + k_a \frac{\partial}{\partial \underline{a}} [E\{e^2(k)\}] \quad (4.9a)$$

$$\underline{b}(k+1) = \underline{b}(k) + k_b \frac{\partial}{\partial \underline{b}} [E\{e^2(k)\}] . \quad (4.9b)$$

The gradient involved in (4.9) can be approximated using the techniques outlined in [Ref. 11]. The iterative LMS algorithm

for estimating the coefficients \underline{a} and \underline{b} is,

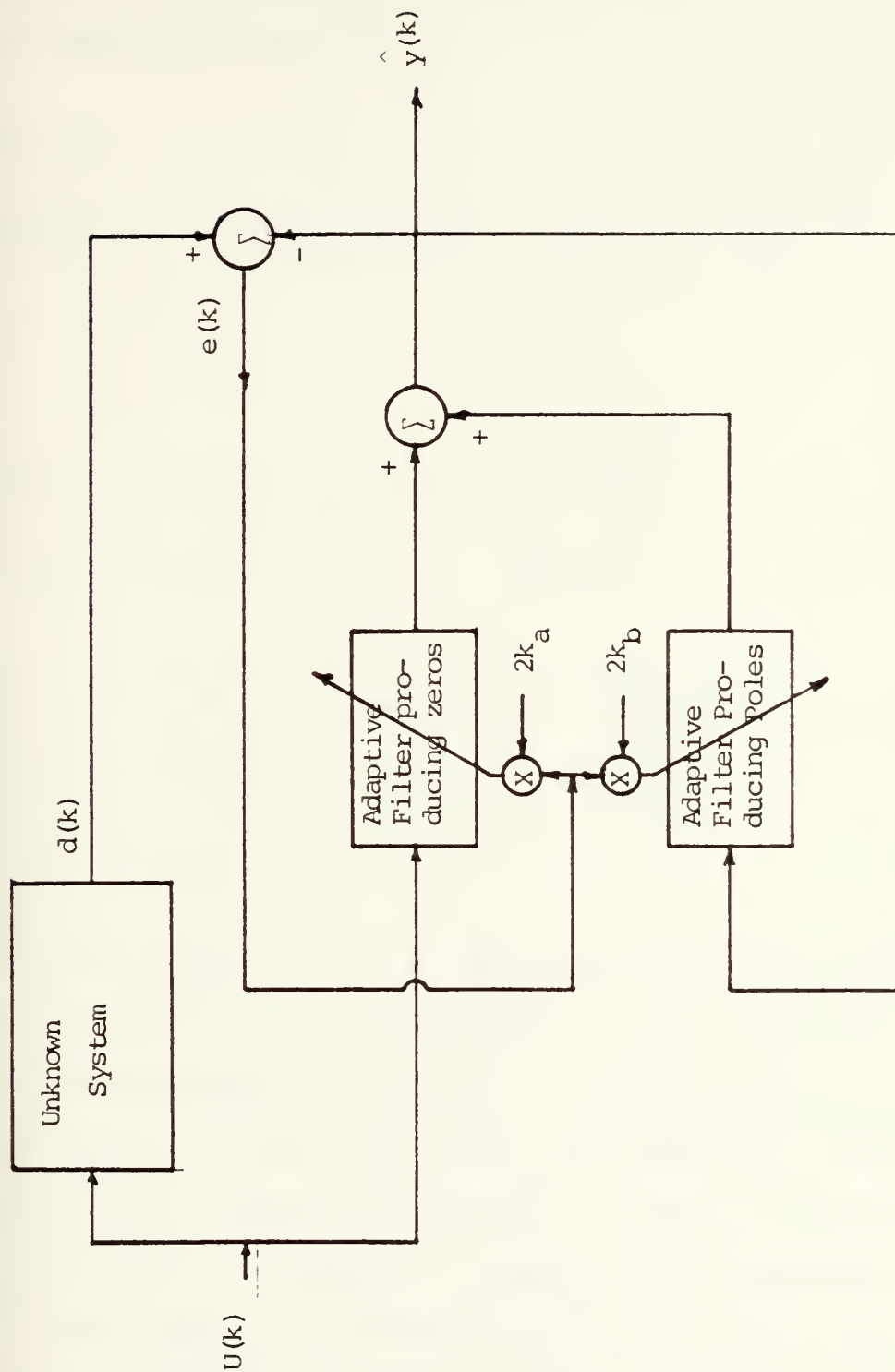
$$\underline{a}(k+1) = \underline{a}(k) - 2k_a e(k) \underline{u}(k) \quad (4.10a)$$

$$\underline{b}(k+1) = \underline{b}(k) - 2k_b e(k) \underline{y}(k). \quad (4.10b)$$

Figure 4.1 is a block diagram implementation of the Adaptive Recursive LMS Filter algorithm as applied to system identification. The input signal, $u(k)$, to both the unknown system and the adaptive filter is a zero mean, white gaussian noise sequence of samples from a random process. The output of the unknown system is $d(k)$ which is used as the training signal for the filter. The output of the adaptive filter, $y(k)$, is compared with the desired signal, $d(k)$, from which an error signal is derived. The error signal, $e(k)$, is then used in the LMS algorithm, equation (4.10), to iteratively adjust the \underline{a} and \underline{b} coefficients. Theoretically, when the coefficients of the zeros-producing adaptive filter and the poles-producing adaptive filter correspond exactly with those of the unknown system, the error sequence should be zero and the unknown system is modeled by the Adaptive Recursive LMS Filter. The convergence constants, k_a and k_b , are chosen a-priori as are the initial values, $\underline{a}(0)$ and $\underline{b}(0)$, for the coefficients.

B. THE COMPARISON

One can begin the comparison between the Adaptive Kalman Identifier and the Adaptive Recursive LMS filter by noting



Filter 4.1. Adaptive Recursive LMS Filter Applied to System Identification

the similarities between the equations (2.43) and (4.8), repeated here for convenience,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \hline \hat{\underline{b}}(k+1|k) \end{bmatrix} = \begin{bmatrix} R_{uu}(k) & | & -R_{uy}(k-1) \\ \hline -R_{yu}^T(k-1) & | & R_{yy}(k) \end{bmatrix}^{-1} \begin{bmatrix} R_{uy}(k) \\ \hline R_{yy}(k) \end{bmatrix} \quad (2.43a)$$

$$A_1 = B_1 C_1 \quad (2.43b)$$

$$\begin{bmatrix} R_{uu}(k) & | & R_{uy}(k-1) \\ \hline R_{uy}^T(k-1) & | & R_{yy}(k) \end{bmatrix} \begin{bmatrix} \underline{a} \\ \hline \underline{b} \end{bmatrix} = \begin{bmatrix} R_{du}(k) \\ \hline R_{dy}(k) \end{bmatrix} \quad (4.8a)$$

$$\begin{bmatrix} \hat{\underline{a}} \\ \hline \hat{\underline{b}} \end{bmatrix} = \begin{bmatrix} R_{uu}(k) & | & R_{uy}(k-1) \\ \hline R_{uy}^T(k-1) & | & R_{yy}(k) \end{bmatrix}^{-1} \begin{bmatrix} R_{du}(k) \\ \hline R_{dy}(k) \end{bmatrix} \quad (4.8b)$$

$$A_2 = B_2 C_2 \quad (4.8c)$$

The upper left partitions of B_1 and B_2 are obviously identical, in that the AKI and the Adaptive Recursive LMS both implement instantaneous estimates of the input covariance functions, $R_{uu}(k)$. The lower right partitions of B_1 and B_2 are similar with one subtle difference. Both are instantaneous covariance functions; however, B_1 employs the covariance, $R_{yy}(k)$, of the actual data whereas B_2 uses, in fact, previous filter estimates to compute the output covariance, $R_{yy}(k)$, more properly denoted by $R_{yy}^{\hat{\hat{}}}(k)$. The negative sign of the upper right and lower left partitions in B_1 , in comparison with B_2 , is a result of the initial definition for the recursive

weights, \underline{b}_i . That is, the transfer function for the general ARMA equation can be written as,

$$H_{\text{ARMA}}(z) = \frac{a_0 + \sum_{i=1}^m a_i z^{-i}}{1 - \sum_{i=1}^m b_i z^{-i}} \quad (4.9a)$$

or,

$$H'_{\text{ARMA}}(z) = \frac{a_0 + \sum_{i=1}^m a_i z^{-i}}{1 + \sum_{i=1}^n b_i z^{-i}} \quad (4.9b)$$

The second and more important observation is that cross-correlations employed by equation (4.8b) use the instantaneous values for the past estimates of the outputs while (2.43a) uses the actual past values for $y(k)$. Similarly, comparing C_1 with C_2 , one finds that the lower partition of C_2 uses the cross-correlation between the desired signal and the past estimates of the adaptive recursive LMS filter.

Equation (4.8a) can convey the above information more clearly if it is instead written as,

$$\begin{bmatrix} \hat{\underline{a}} \\ \hat{\underline{b}} \end{bmatrix} = \begin{bmatrix} R_{uu}(k) & R_{u\hat{y}}(k-1) \\ R_{u\hat{y}}^T(k-1) & R_{\hat{y}\hat{y}}(k) \end{bmatrix}^{-1} \begin{bmatrix} R_{du}(k) \\ R_{d\hat{y}}(k) \end{bmatrix} \quad (4.10)$$

The convergence factors, k_a and k_b , can be compared to the steady state Adaptive Kalman Identifier gains by making

the identical analogies (3.8) as was done in the previous chapter. One also observes that since the output data that is processed by the Adaptive Recursive LMS filter are in actuality filter estimates, a more correct version of Feintuch's original algorithm (4.10) is written as,

$$\underline{a}(k+1) = \underline{a}(k) - 2k_a e(k) \underline{u}(k) \quad (4.11a)$$

$$\underline{b}(k+1) = \underline{b}(k) - 2k_b e(k) \hat{\underline{y}}(k) . \quad (4.11b)$$

In augmented form, equation (4.11) is,

$$\begin{bmatrix} \underline{a}(k+1) \\ \text{-----} \\ \underline{b}(k+1) \end{bmatrix} = \begin{bmatrix} \underline{a}(k) \\ \text{-----} \\ \underline{b}(k) \end{bmatrix} - \begin{bmatrix} 2k_a \underline{u}(k) \\ \text{-----} \\ 2k_b \underline{y}(k) \end{bmatrix} [d(k) - \underline{a}(k) \underline{u}(k) - \underline{b}(k) \underline{y}(k)]$$

$$\begin{bmatrix} \underline{a}(k+1) \\ \text{-----} \\ \underline{b}(k+1) \end{bmatrix} = \begin{bmatrix} \underline{a}(k) \\ \text{-----} \\ \underline{b}(k) \end{bmatrix} - \begin{bmatrix} 2k_a \underline{U}(k) \\ \text{-----} \\ 2k_b \underline{y}(k) \end{bmatrix} \begin{bmatrix} d(k) - [\underline{u}(k) \quad \underline{y}(k)] \begin{bmatrix} \underline{a}(k) \\ \text{-----} \\ \underline{b}(k) \end{bmatrix} \end{bmatrix} . \quad (4.12)$$

Recalling equation (2.15a) in a slightly different form we have,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \text{-----} \\ \hat{\underline{b}}(k+1|k) \end{bmatrix} = \begin{bmatrix} \hat{\underline{a}}(k|k-1) \\ \text{-----} \\ \hat{\underline{b}}(k|k-1) \end{bmatrix} + K(k) \begin{bmatrix} z(k) - H(k) \begin{bmatrix} \hat{\underline{a}}(k|k-1) \\ \text{-----} \\ \hat{\underline{b}}(k|k-1) \end{bmatrix} \end{bmatrix} . \quad (4.13)$$

It is a simple matter to explore the similarities between (4.12) and (4.13) and make the following associations,

$$\begin{bmatrix} \hat{\underline{a}}(k+1|k) \\ \text{-----} \\ \hat{\underline{b}}(k+1|k) \end{bmatrix} \quad \langle \Rightarrow \rangle \quad \begin{bmatrix} \underline{a}(k+1) \\ \text{-----} \\ \underline{b}(k+1) \end{bmatrix} \quad (4.14a)$$

$$\begin{bmatrix} \hat{\underline{a}}(k|k-1) \\ \text{-----} \\ \hat{\underline{b}}(k|k-1) \end{bmatrix} \quad \langle \Rightarrow \rangle \quad \begin{bmatrix} \underline{a}(k) \\ \text{-----} \\ \underline{b}(k) \end{bmatrix} \quad (4.14b)$$

$$z(k) \quad \langle \Rightarrow \rangle \quad d(k) \quad (4.14c)$$

$$H(k) = \begin{bmatrix} u(k) & \dots & & \\ u(k-m) & & y(k-1) & \\ \dots & & y(k-n) & \end{bmatrix} \quad \langle \Rightarrow \rangle \quad [\underline{u}(k) \quad \underline{y}(k)] \quad (4.14d)$$

$$K(k) \quad \langle \Rightarrow \rangle \quad \begin{bmatrix} 2k_a \underline{u}(k) \\ \text{-----} \\ 2k_b \underline{y}(k) \end{bmatrix} \quad (4.14e)$$

The associations (4.14d) and (4.14e) are the major differences between the AKI and the Adaptive Recursive LMS. It has been shown that the mean square error surface employing not only estimated feed forward coefficients but also estimated feed-back (recursive) coefficients is in general multimodal [Ref. 36]. Hence, the Adaptive Recursive LMS algorithm does not minimize the mean square error [Ref. 19,20], and in general the gradient algorithm in this case does not seek the global minimum.

C. OBSERVATIONS

The complex nature of estimating recursive coefficients is yet a formidable problem [Ref. 19,20]. It seems that Feintuch's algorithm is successful due to the a-priori knowledge of the minimal order generating the desired signal. This knowledge is used in setting the order of the adaptive recursive filter.

V. SYNTHETIC DATA GENERATING PLANTS

The Adaptive ARMA Kalman Identifier was tested using computer derived data from several models. The input/output data was collected by driving a known plant with a zero mean, unit variance, white gaussian noise sequence of samples. The AKI algorithm was repeatedly tested using data from progressively more complex models.

A. AUTOREGRESSIVE-MOVING AVERAGE DATA

Autoregressive-Moving Average (ARMA) data was generated using an equation of the form,

$$\begin{aligned} y(k) = & b_1 y(k-1) + b_2 y(k-2) + \dots + b_n y(k-n) \\ & + a_0 u(k) + \dots + a_m u(k-m) \end{aligned} \quad (5.1)$$

Taking the Z-transform of (5.1) we have,

$$\begin{aligned} Y(z) = & b_1 z^{-1} Y(z) + b_2 z^{-2} Y(z) + \dots + b_n z^{-n} Y(z) \\ & + a_0 U(z) + \dots + a_m z^{-m} U(z) \end{aligned} \quad (5.2)$$

where $Y(z)$ ($U(z)$) represents the Z-transform of the output (input) and $z^{-n_0} Y(z)$ ($z^{-n_0} U(z)$) represents the Z-transform of the output (input) delayed n_0 time steps. Equation (5.1) has already been defined as the general form of the ARMA process provided that the sequence $u(k-i)$ $i = 0, 1, \dots$ comes from a gaussian random process [Ref. 25]. The ratio,

$$H(z) = \frac{Y(z)}{U(z)} = \frac{a_0 + a_1 z^{-1} + \dots + a_m z^{-m}}{1 - b_1 z^{-1} - \dots - b_n z^{-n}} \quad (5.3)$$

can then be readily recognized as the general transfer function [Ref. 37] of a digital plant. That is, the $u(k)$ is the observed input and the $y(k)$ is the perfectly measured output. Consider that the measurements of the output include some random error, $v(k)$, due to the inaccuracy of the measuring instruments, then the resulting situation is as depicted in Figure 5.1. A practical, judicious and reasonable description of the measurement error is that this be a stochastic process whose distribution is zero mean gaussian. The noisy measurement, $z(k)$, is then the output, $y(k)$, plus the measurement error, $v(k)$.

There are three cases of interest: Case 1: $b_i = 0$ for all i ; Case 2: $a_i = 0, i = 1, 2, \dots, m$; Case 3: $a_i \neq 0, b_i \neq 0$ for all i .

1. Moving Average Data (Case 1)

Case one is readily recognized as the all zero plant which produces moving average data. A simple second order plant was defined where $a_0 = 1.0, a_1 = 2.0$ and $a_2 = 3.0$. The gaussian noise sequences, $u(k)$ and $v(k)$, were obtained using the general purpose IMSL subroutine, GGNML, provided by the computer center at the Naval Postgraduate School.

2. Autoregressive Data (Case 2)

Case two results in the all pole plant producing autoregressive data. The coefficients were selected such that the

plant of Figure 5.1 produced stable data. In precise terms, the poles of the function,

$$H(z) = \frac{1}{1 + \frac{26}{24} z^{-1} + \frac{9}{24} z^{-2} + \frac{1}{24} z^{-3}} \quad (5.4)$$

were located completely within the unit circle in the Z-plane ensuring a stable plant [Ref. 37,38]. The noise sequences were obtained as before.

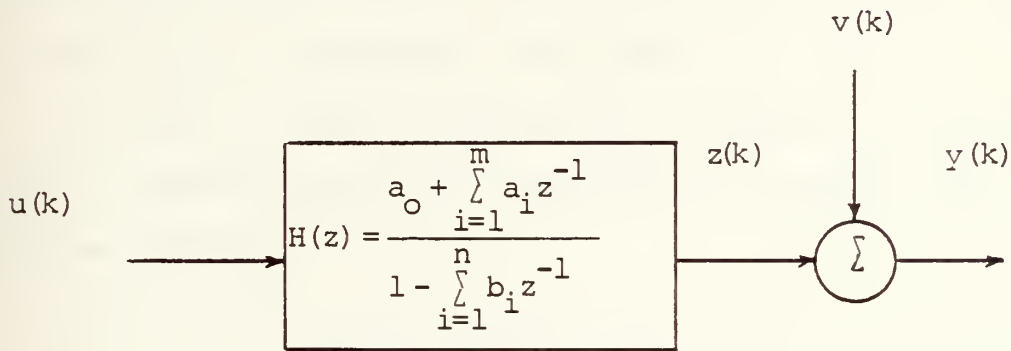


Figure 5.1. ARMA Digital Plant

3. Autoregressive-Moving Average Data (Case 3)

Case three follows from a logical combination of the two previous cases. That is, the transfer function, $H(z)$, now has both zeros and poles. In order to compare with previous work done in the area of system identification [Ref. 6], one of Perry's models [Ref. 6] was used. Namely, the transfer function $H(z)$ used for this case was,

$$H(z) = \frac{1 + 1.4z^{-1} + .98z^{-2}}{1 - 1.14z^{-1} + 1.4549z^{-2} - .88490z^{-3} + .40745z^{-4}} \quad (5.5)$$

The input, $u(k)$, was obtained as in the previous two cases using the IMSL subroutine GGNML set to unit variance and zero mean. The output measurement noise sequence, however, was set to a variance of .0001. The basic software, program ARMA, which generated the different data is included as Appendix B. Basically, the coefficients A1-A10 change the character of the general ARMA equation (5.1) which can be selected to produce the desired plant data. The input/output data is then written onto a disk file for subsequent analysis.

In order to compare some of the findings in this thesis with the results previously obtained by Feintuch [Ref. 18], the transfer function,

$$H(z) = \frac{0.05 - 0.40z^{-1}}{1.0 - 1.1314z^{-1} + 0.25z^{-2}} \quad (5.6)$$

was also used as a source of synthetic data. Feintuch used (5.6) as a source of data with which to analyze the operation of his Adaptive Recursive LMS Filter.

B. PHASE LOCKED LOOP DATA

The second class of data used to exercise the Adaptive ARMA Kalman Identifier was derived from a computer simulated phase locked loop (PLL) developed by Romeo [Ref. 7]. The basic PLL algorithm, however, implemented a forward Euler integration scheme.

Briefly, in block diagram form, the phase tracking characteristics of the PLL in the frequency domain can be depicted as in Figure 5.2.

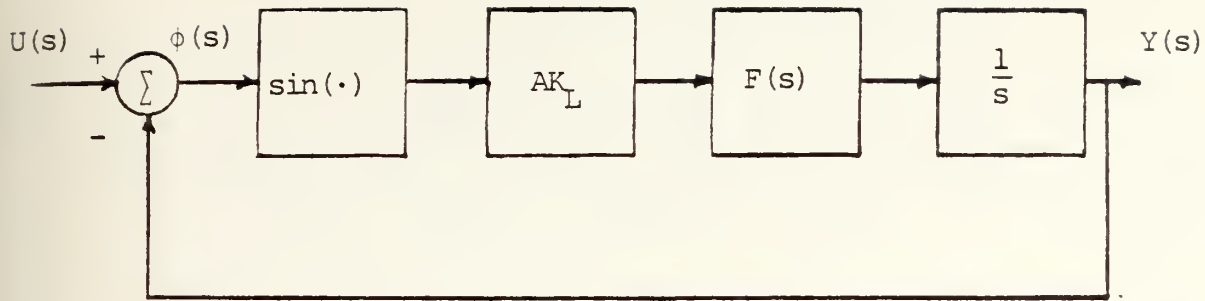


Figure 5.2. PLL Block Diagram

Where:

A is the input voltage

K is the loop gain

F(s) is the filter transfer function

For most applications, it is assumed that $\phi(t) \ll \pi/2$ allowing one to make the approximation, $\sin \alpha \approx \alpha$. This is the linear mode of operation. Romeo [Ref. 7] chose the filter characteristics $F(s) = 1 + K/S$ and the same characteristics were used in this thesis in order to compare results. The parameters of the overall system were adjusted to obtain a damping coefficient, ζ , of 0.3 and a natural frequency, ω_n of 3.33 rad/sec. This resulted in a step response overshoot of about 46.7% at $t \approx .75$ sec. Solving for the time domain step response analytically in the linear region one obtains,

$$y(t) = 1 + 1.048e^{-t} \sin(3.178t - 1.266) \quad (5.7)$$

The final block diagram of the PLL system is shown in Figure 5.3.

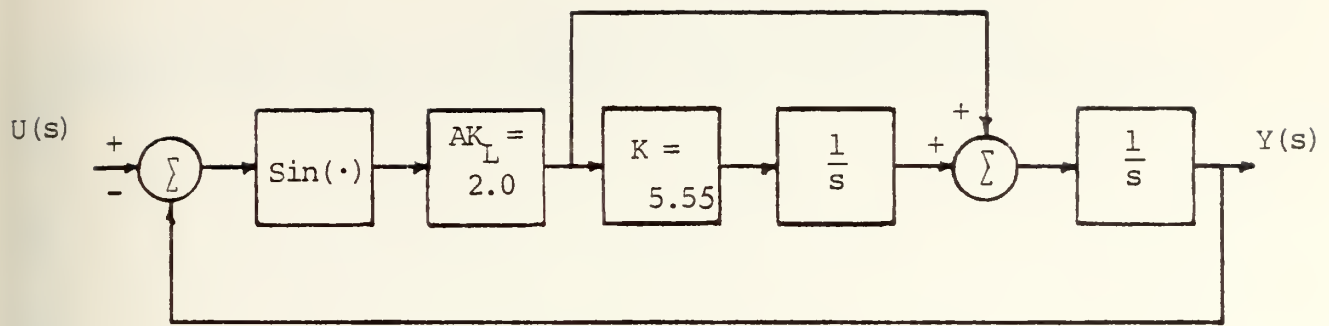
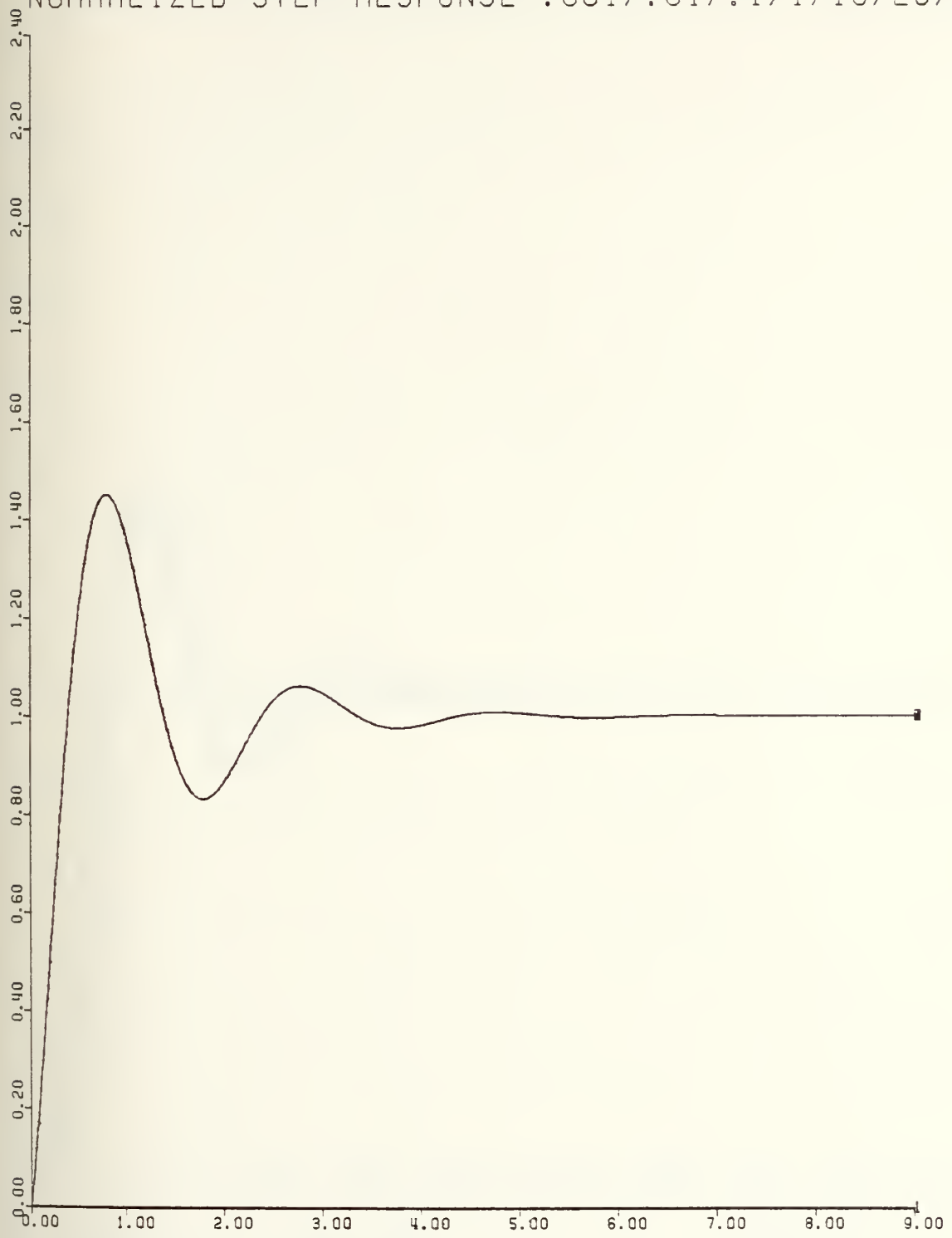


Figure 5.3. Final PLL Block Diagram

Normalized step responses for several step magnitudes using Digital Simulation Language (DSL) were obtained. Figure 5.4 shows the step responses of several step inputs in the range $0 < u(t) < 30$. It is obvious that the PLL is in its approximate linear region and does not exhibit any discernible distortion. Figure 5.5 is a repeat of the above test; however, the range of step inputs are $30 \leq u(t) \leq 170$. It is apparent from Figure 5.5 that the $\sin(\cdot)$ nonlinearity begins to have some effect on the operation of the PLL in that the amplitude of the responses are reduced and delayed.

The DSL PLL system, however, was not used as a source of synthetic data because of the problem of nonstationary statistics of the input signal using the random sequence generators available under DSL already discussed in Ref. 7. These simulations were nevertheless used as a basis for comparison of the operation of the discrete PLL developed in Reference 7 but modified to implement a forward Euler integration scheme which is used in this thesis.

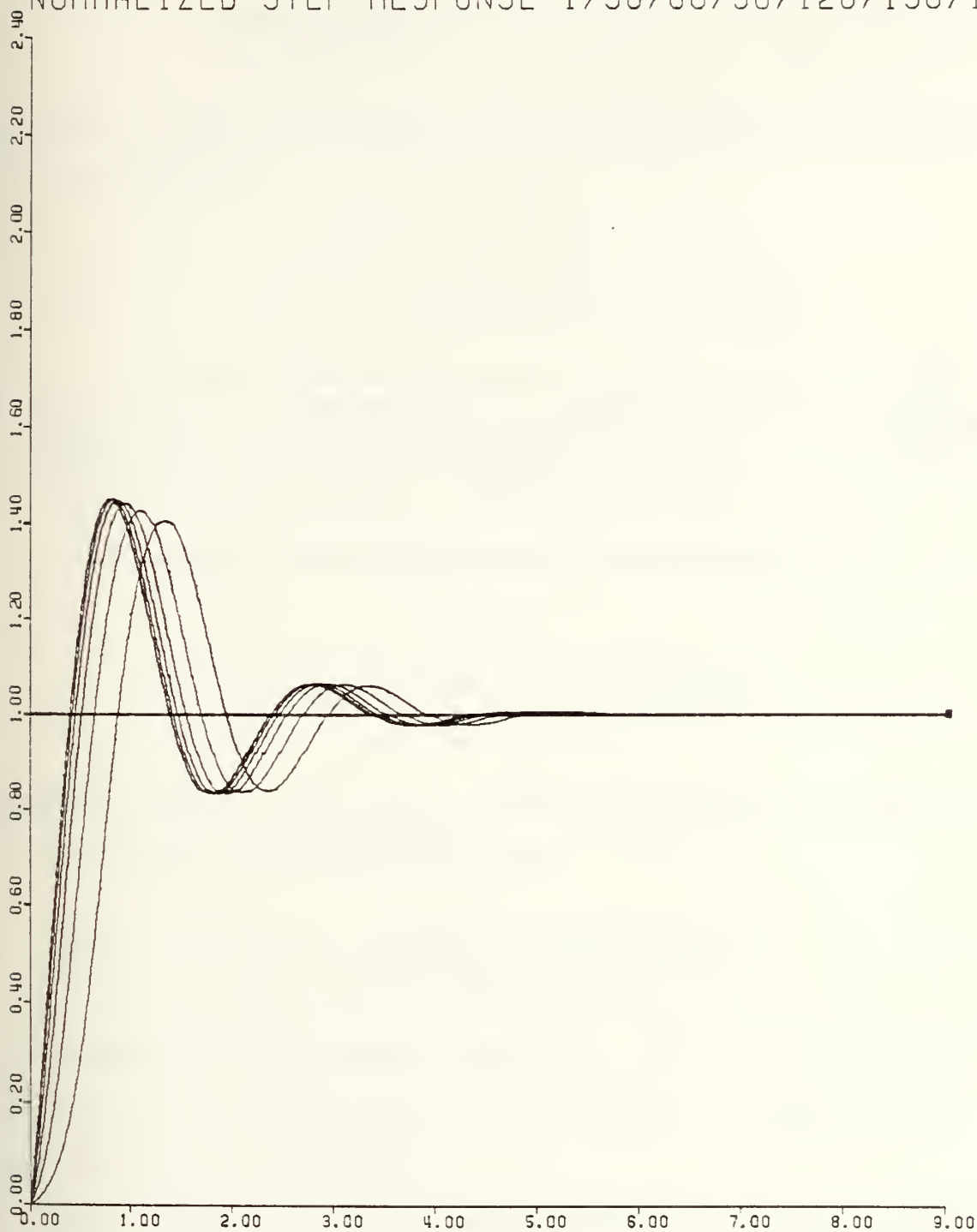
2ND ORDER PLL -- TRAPEZOIDAL INTEGRATION
NORMALIZED STEP RESPONSE .001/.01/.1/1/10/20/30



XSCALE= 1.00 UNITS/INCH RUN NO. 1
YSCALE= 0.20 UNITS/INCH PLOT NO. 1

Figure 5.4. PLL Step Response, Linear Region

2ND ORDER PLL -- TRAPEZOIDAL INTEGRATION
NORMALIZED STEP RESPONSE 1/30/60/90/120/150/170



XSCALE= 1.00
YSCALE= 0.20

UNITS/INCH
UNITS/INCH

RUN NO. 1
PLOT NO. 1

Figure 5.5. PLL Step Response, Non-Linear Region

The forward Euler integrator takes the form,

$$\frac{1}{s} \iff \frac{Tz^{-1}}{1-z^{-1}}, \quad T = .01 \quad (5.8)$$

The block diagram of Figure 5.3 changes character to appear as in Figure 5.6.

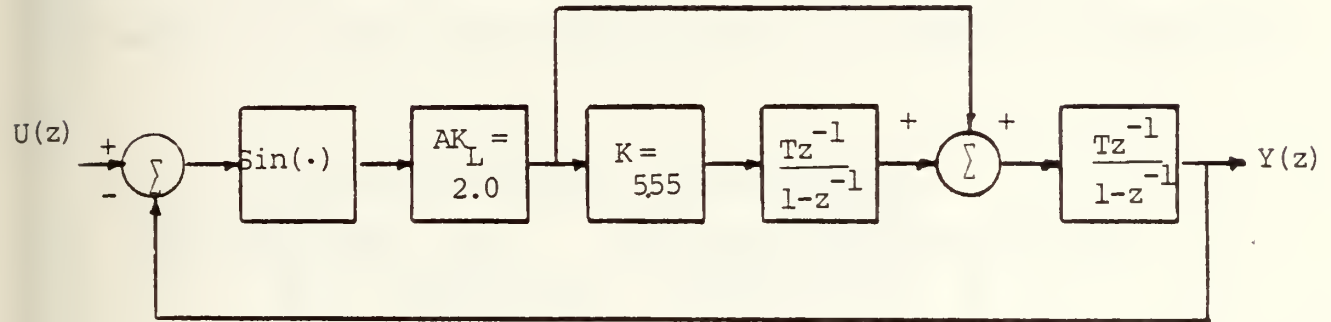


Figure 5.6. Discrete PLL Block Diagram

Applying Mason's gain rule to the block diagram of Figure 5.6 in the linear region we arrive at,

$$\frac{Y(z)}{U(z)} = H(z) = \frac{0.02z^{-1} - 0.018890z^{-2}}{1 - 1.980z^{-1} + .981110z^{-2}} \quad (5.9)$$

the linearized PLL transfer function. The discrete PLL system has zeros at (0.0, .9445) and a complex conjugate pole pair at (0.990 ± j.03178).

The discrete PLL system was implemented using a Fortran program on the IBM 370 in double precision. The source code for the discrete PLL is included as Appendix C. The discrete

PLL was tested using various magnitude step inputs. Results were similar to those from the DSL simulation. That is, the discrete PLL exhibited the same approximate overshoot, rise time, natural frequency and nearly identical time of max overshoot in the linear region. To implement the PLL using forward Euler integration, the block diagram of Figure 5.6 was reconfigured to appear as Figure 5.7. Note that Figure 5.7 does not employ any delay free loops, hence it is easily programmable.

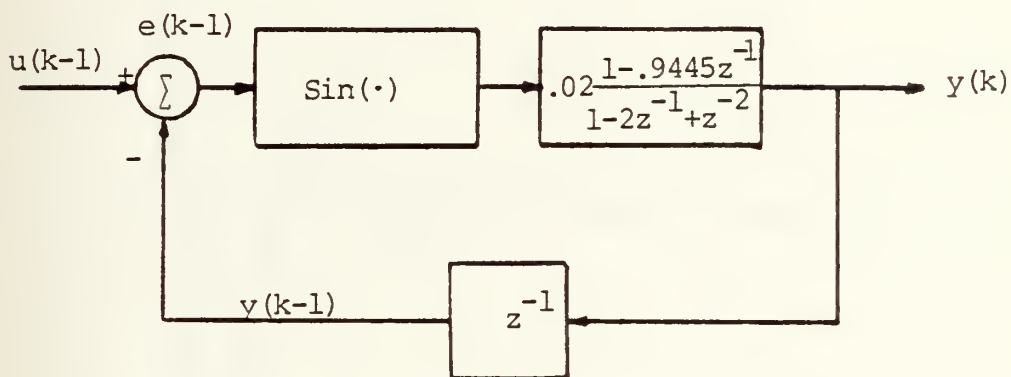


Figure 5.7. Programmable PLL Implementing Forward Euler Integration

Figure 5.8 summarizes the results of the discrete PLL simulation for normalized step inputs in the range $0 < u(t) \leq 30$. Similarly, the discrete PLL was tested in the nonlinear region using normalized step inputs in the range $30 \leq u(t) \leq 170$. The discrete PLL displayed the same characteristics of decreased output amplitude and delay which was first observed in the DSL simulation. Figure 5.9 summarizes the test results of the discrete PLL in the nonlinear region.

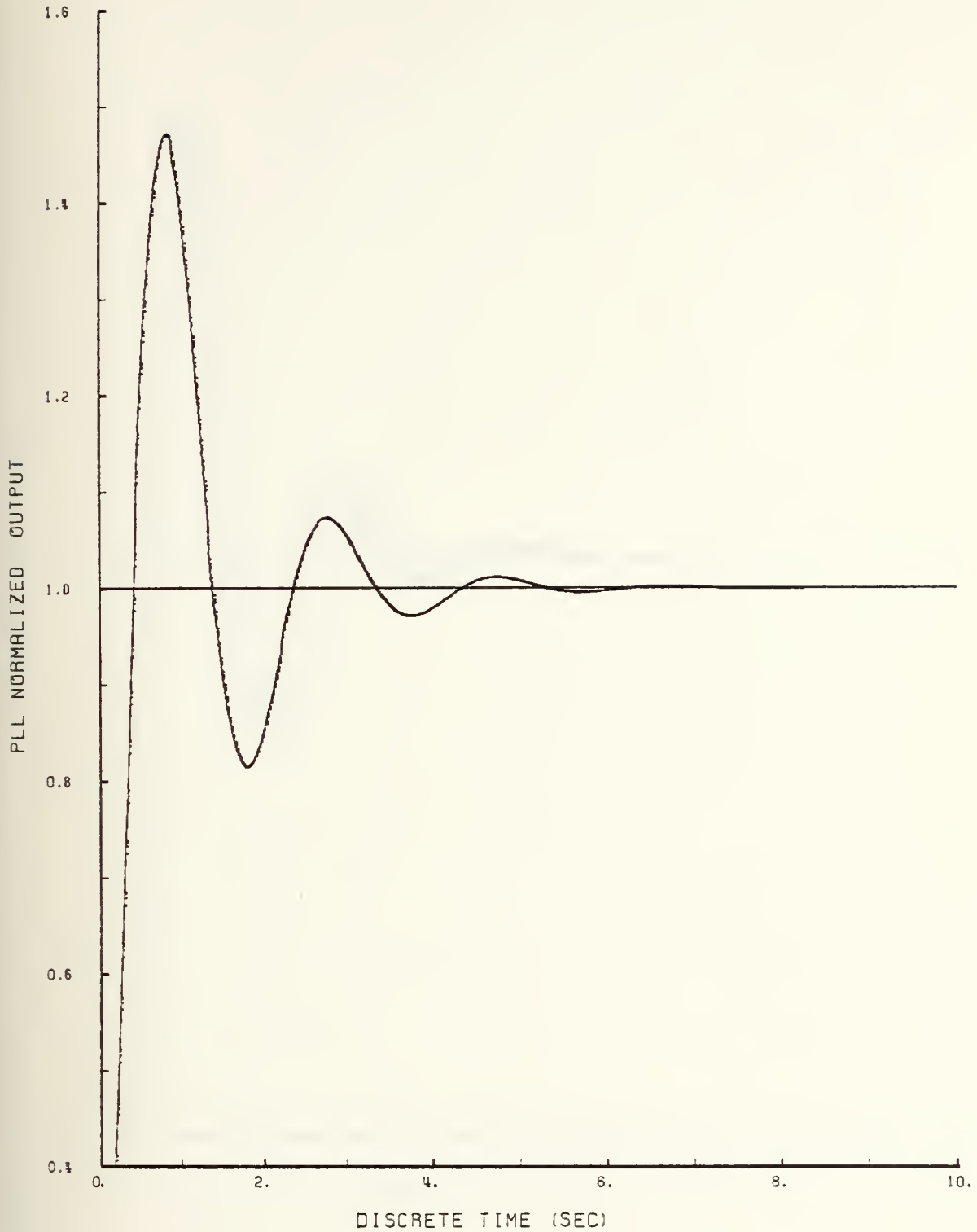


Figure 5.8. Discrete PLL Step Response, Linear Region

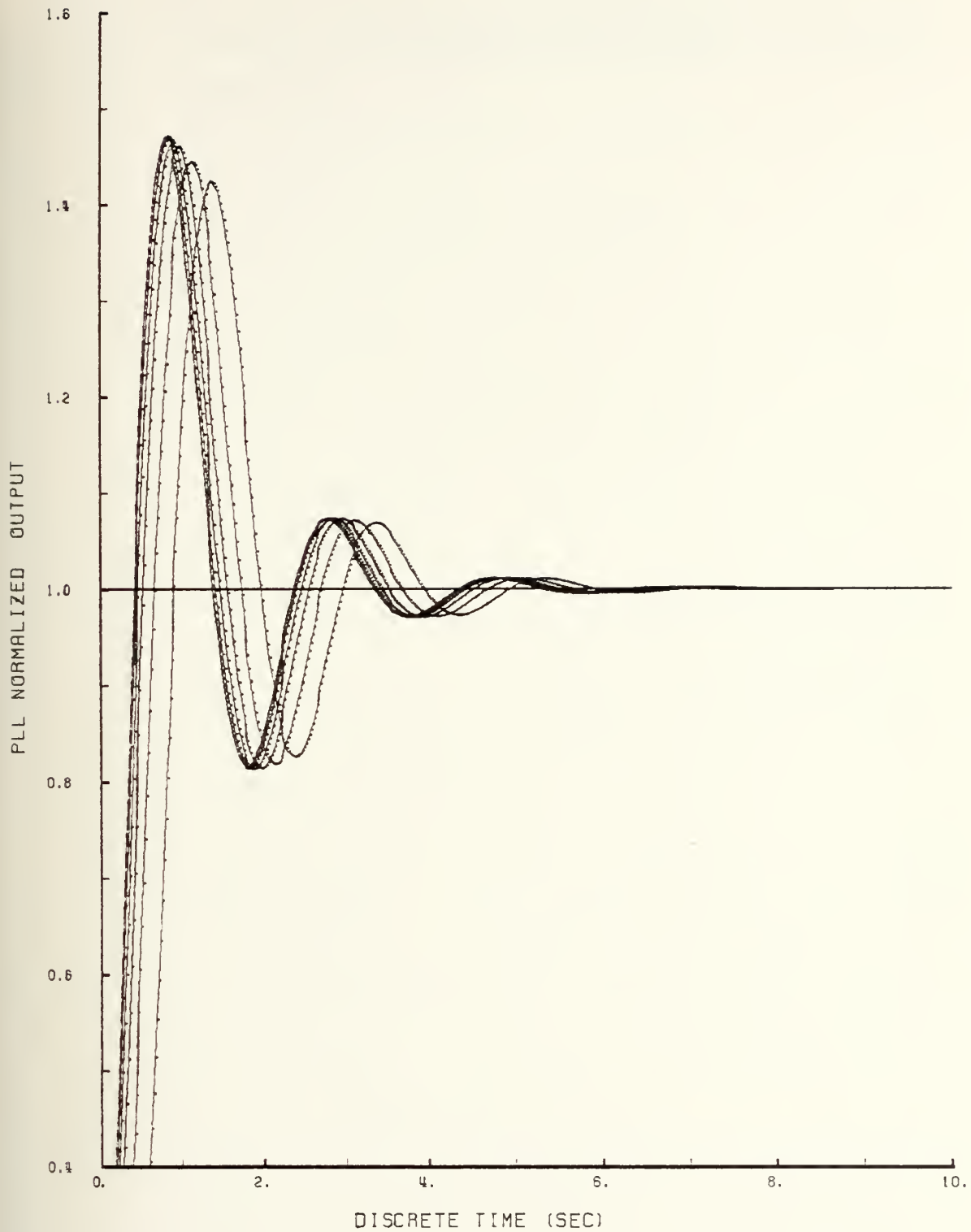


Figure 5.9. Discrete PLL Step Response, Nonlinear Region

It was felt that the PLL was adequately modeled for use as a source of data to be subsequently analyzed by the Adaptive Kalman Identifier.

VI. MODEL IDENTIFICATION SOFTWARE

Three basic Fortran IV programs for use on the IBM 370 were developed:

- (1) ADAPTSN: Adaptive Kalman Identifier
- (2) LMS: Adaptive Least Mean Square Filter
- (3) LMSR: Adaptive Recursive Least Mean Square Filter

All three programs used double precision arithmetic to minimize the effects of truncation error, limit cycles and roundoff error. In all cases the orders of the autoregressive and moving average processes were read in unformatted form from a disk file along with the input/output data of the unknown system to be analyzed. An attempt was made to minimize memory storage, but not at the expense of program flow and clarity.

A. ADAPTIVE KALMAN IDENTIFIER

Program ADAPTSN is a versatile Fortran IV software program which implements equations (2.15) to identify the coefficients of an MA, AR, or ARMA process. Additionally, at each iteration the poles and zeros of the evolving transfer function are computed. ADAPTSN can also be used to perform regression analysis on non-linear terms as discussed in Chapter VII. The versatility of the AKI lies in the various options which are available to the user. By properly selecting the flags N, M, and NL, the user can perform regression analysis on data whose combined order $(N+M+1)$ is less than or equal to 20. Table 6.1 lists the ascribed meanings of the various flags.

Table 6.1

AKI FLAG MEANINGS

- N: the order of the autoregressive process
- M: one greater than the order of the moving average process
- NL = 0: AKI is used to identify the coefficients associated with the general ARMA equation (2.1b) (linear regression)
- NL = 1: AKI is used to identify the coefficients of the linear general ARMA equation and the weighting coefficient associated with $u^3(k-i)$
- NL = 2: AKI is used as in NL = 0,1 and additionally identifies the weighting coefficient associated with $y^3(k-i)$
- NL = 3: AKI is used as in NL = 0,1,2 and additionally identifies the weighting coefficient associated with $u^2(k-i)y(k-i)$
- NL = 4: AKI is used as in NL = 0,1,2,3, and additionally identifies the weighting coefficient associated with $y^2(i-i)u(k-i)$
- NL = 5: AKI is implemented to analyze time series and identify the ARMA coefficients associated with the Box-Jenkins model [Ref. 26].

The option given by NL = 5 was not extensively tested and as such only limited results are available. Table 6.2 outlines the allowable flag combinations which will produce a valid analysis of the data.

For purposes of this thesis, the non-linear (NL) options were configured to implement the expansion terms associated with the Taylor series expansion of the sine function. This, however, can be changed at the user's discretion to implement any other series expansion by inserting the appropriate Fortran statements at the proper location in the AKI program.

Table 6.2

VALID FLAG COMBINATIONS

OPTION (NL)	MAX COMBINED ORDER
0	$N + M \leq 20$
1	$N + 2M \leq 20; N, M \neq 0$
2	$2N + 2M \leq 20; N, M \neq 0$
3	$3N + 2M \leq 20; N, M \neq 0$
4	$4N + 2M \leq 20; N, M \neq 0$
5	$N + M \leq 20$

The overall structure of the AKI program uses subroutine calls to compute the various quantities necessary for the eventual computation of the system ARMA coefficients. These subroutines are: GAIN, RECKON, PRINT, NEXT. A brief description of the function each performs is given at the beginning of each subroutine. Figure 6.1 describes the general program flow of the AKI. The source code for program ADAPTSN is included as Appendix D.

B. ADAPTIVE LEAST MEAN SQUARE FILTER

Program LMS uses the equations developed in Chapter III to compute the coefficients (or weights) associated with a moving average process. The LMS filter program is capable of computing up to 12 MA coefficients; or equivalently, is capable of identifying an eleventh order moving average process. The general program flow of the LMS filter is presented in Figure 6.2 and the annotated source code is included as Appendix E.

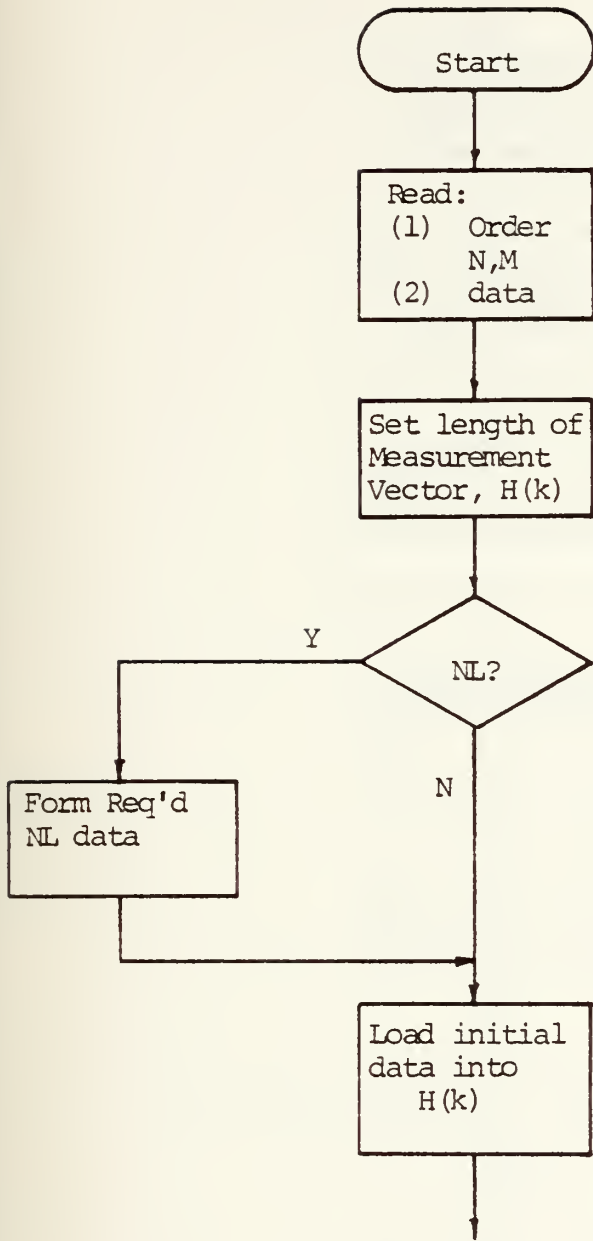


Figure 6.1. AKI General Program Flow

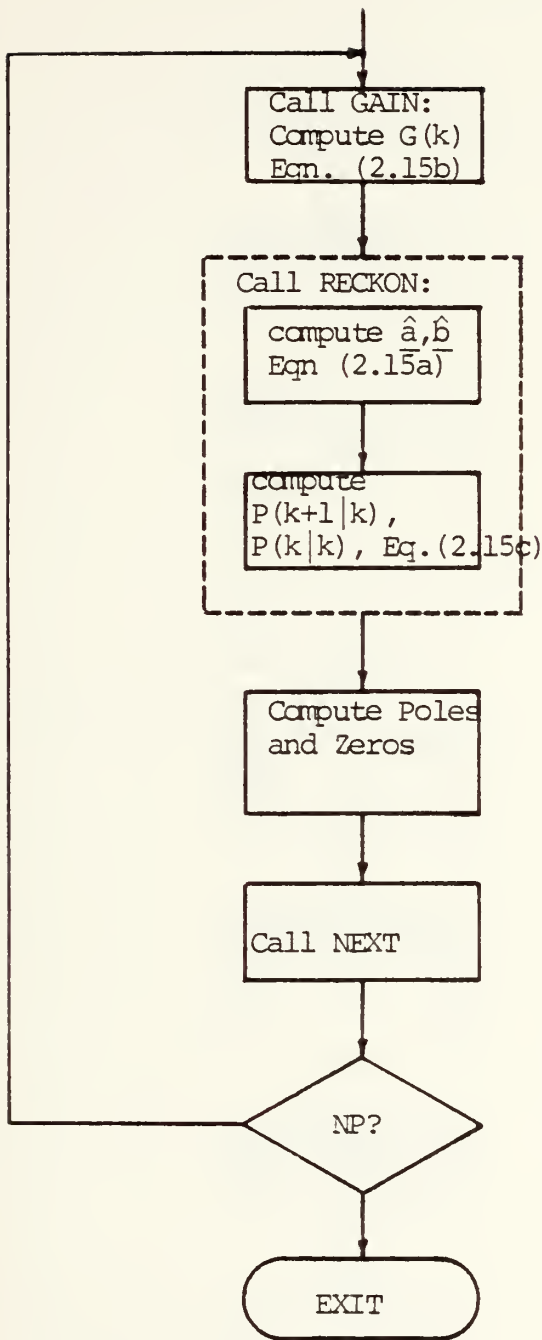


Figure 6.1. (CONTINUED)

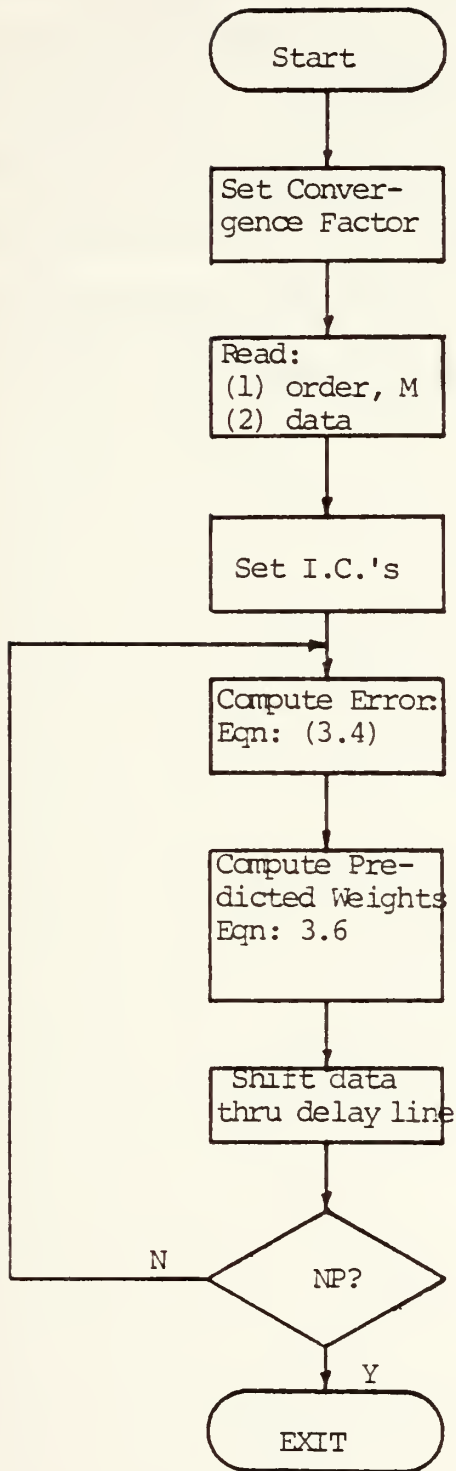


Figure 6.2. LMS Filter Program Flow

C. ADAPTIVE RECURSIVE LEAST MEAN SQUARE FILTER

Program LMSR realizes the algorithm proposed by Feintuch and recapped in Chapter IV. It was designed to handle a combined AR and MA order of 11. That is, the program can be used to estimate $m + 1 = M$ coefficients associated with the MA process and $n = N$ coefficients associated with the AR process such that $N + M \leq 12$. The general program flow for the Adaptive Recursive LMS filter is shown in Figure 6.3 and the source code is included as Appendix F.

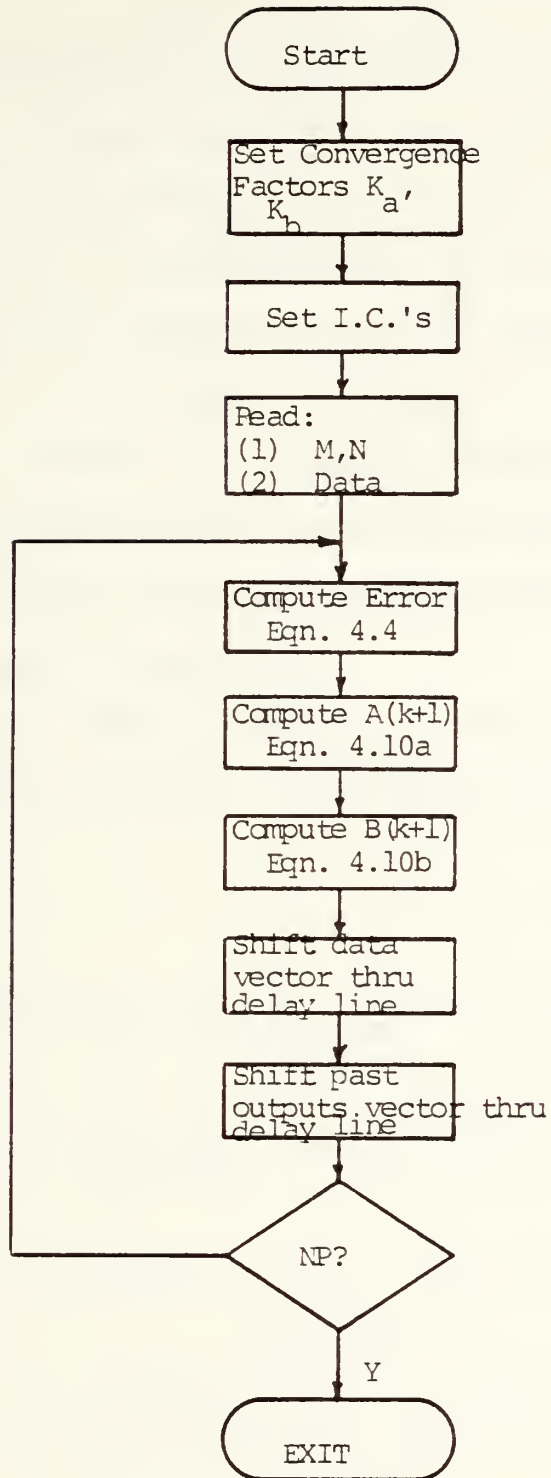


Figure 6.3. LMSR Filter Program Flow

VII. NON-LINEAR IDENTIFICATION

The Adaptive Kalman Identifier can be modified to estimate the weighting coefficients associated with higher order terms produced by some non-linearity within the unknown system. In general, much has to be known about the non-linearity such that the functional description chosen for it is a close approximation to the effects it causes. In this thesis one possible application of the Adaptive Kalman Identifier toward identifying a system with a known non-linearity is explored. No attempt has been made to present an extensive treatment of non-linear analysis techniques.

The approach taken has been previously explored by Parker [Ref. 8]. A special case of the generalized non-linear ARMA model [Ref. 8],

$$\begin{aligned}
 y(k) = & \sum_{i_1=0}^{\infty} a(i_1)u(k-i_1) + \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} a(i_1, i_2)u(k-i_1)u(k-i_2) + \dots \\
 & + \sum_{i_1=0}^{\infty} \dots \sum_{i_m=0}^{\infty} a(i_1, \dots, i_m)u(k-i_1) \dots u(k-i_m) \\
 & + \sum_{j_1=1}^{\infty} b(j_1)y(k-j_1) + \sum_{j_1=1}^{\infty} \sum_{j_2=1}^{\infty} b(j_1, j_2)y(k-j_1)y(k-j_2) + \dots \\
 & + \sum_{j_1=1}^{\infty} \dots \sum_{j_n=1}^{\infty} b(j_1, j_2, \dots, j_n)y(k-j_1)y(k-j_2) \dots y(k-j_n) \\
 & + \sum_{i_j=0}^{\infty} \sum_{j_1=1}^{\infty} c(i_1, j_1)u(k-i_1)y(k-j_1) + \dots + \sum_{i_1=0}^{\infty} \dots \sum_{i_m=0}^{\infty} \sum_{j_1=1}^{\infty} \\
 & \dots \sum_{j_n=1}^{\infty} c(i_1, \dots, i_m, j_1 \dots j_n)u(k-i_1) \dots u(k-i_m)y(k-j_1) \dots y(k-j_n)
 \end{aligned}
 \tag{7.1}$$

is used to model the input/output relationships, non-linear transfer function, of the phase locked loop. The non-linear element, the sine function, of the PLL can be replaced by a Taylor power series expansion,

$$\sin(x) = x - \frac{1}{3!} x^3 + \frac{1}{5!} x^5 - \dots \quad (7.2)$$

Other expansions can be used, e.g., Legendre polynomials, Volterra series, ..., etc.; however, only the Taylor expansion was investigated in this thesis. Practical implementation of (7.2) suggests truncation at the third order term. The sine function is therefore approximated as

$$\sin(x) = x - \frac{1}{3!} x^3 . \quad (7.3)$$

Substituting (7.3) into the block diagram of Figure (5.6) we have Figure 7.1.

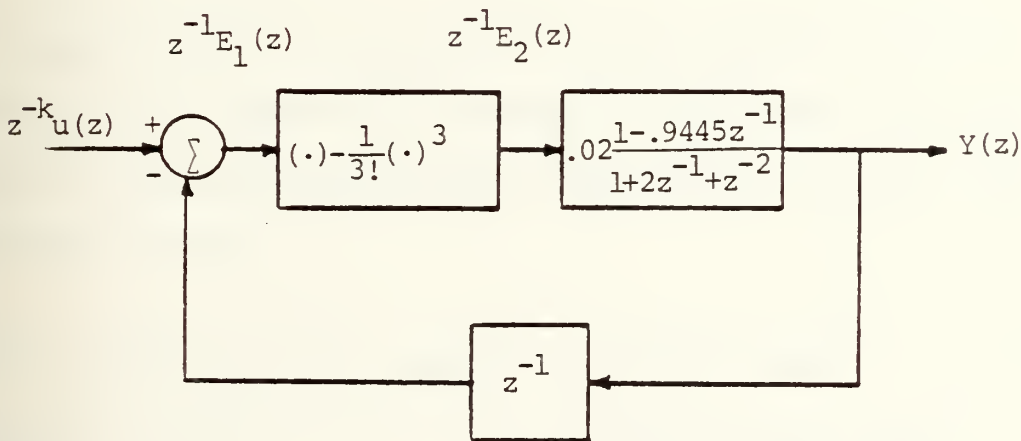


Figure 7.1. PLL, Third Order Taylor Series Approximation

The discrete time domain equations at the different nodes are:

$$e_1(k-1) = u(k-1) - y(k-1) \quad (7.4a)$$

$$e_2(k-1) = e_1(k-1) - \frac{1}{3!} e_1^3(k-1) \quad (7.4b)$$

$$y(k) = .02e_2(k-1) - .01889e_2(k-2) + 2.0y(k-1) - y(k-2) \quad (7.4c)$$

Manipulating equations (7.4a)-(7.4c) gives

$$\begin{aligned} y(k) = & .02u(k-1) - .01889u(k-2) + 1.98y(k-1) - .98111y(k-2) \\ & - .003333u^3(k-1) + .003148333u^3(k-2) \\ & + .003333y^3(k-1) - .003148333y^3(k-2) \quad (7.5) \\ & + .01u^2(k-1)y(k-1) - .09445u^2(k-2)y(k-2) \\ & - .01u(k-1)y^2(k-1) + .09445u(k-2)y^2(k-2). \end{aligned}$$

Equation (7.5) indicates which non-linear terms of equation (7.1) should be retained. Therefore, the third order non-linear approximation model should contain the following terms:

$$u(k-1), u(k-2), y(k-1), y(k-2) \quad \text{linear terms} \quad (7.6a)$$

$$u^3(k-1), u^3(k-2) \quad \text{input cubic terms} \quad (7.6b)$$

$$y^3(k-1), y^3(k-2) \quad \text{output cubic terms} \quad (7.6c)$$

$$\left. \begin{array}{l} u^2(k-1)y(k-1), u^2(k-2)y(k-2) \\ y^2(k-1)u(k-1), y^2(k-2)u(k-2) \end{array} \right\} \begin{array}{l} \text{third order cross} \\ \text{product terms} \end{array} \quad (7.6d)$$

The AKI algorithm is primarily modified to include the hybrid signals (7.6) in the structure of $H(k)$, the measurement vector. Since now the measurement vector, $H(k)$ takes the form,

$$\begin{aligned} H(k) = & [u(k) \dots u(k-m), -y(k-1) \dots -y(k-n), u^3(k) \dots \\ & \dots u^3(k-m), -y^3(k-1) \dots -y^3(k-n), -(u^2(k-1)y(k-1)) \dots \\ & \dots -(u^2(k-n)y(k-n)), (u(k-1)y^2(k-1)) \dots (u(k-n)y^2(k-n))] \end{aligned} \quad (7.7)$$

the AKI algorithm calculates the coefficients associated with the special case of the generalized non-linear ARMA model, equation (7.1).

Romeo using a similar approach [Ref. 7] computes a least squares curve fit for the third order truncation of the Taylor series for the sine function over the interval $(0, \pi/2)$. Beginning with the approximation,

$$\sin(x) \approx \alpha x + \beta x^3 \quad (7.8)$$

Romeo finds the values for α and β to be .865 and -.095 respectively. Performing the same operations as those used in obtaining equation (7.5) we arrive at,

$$\begin{aligned}
y(k) = & .02\alpha u(k-1) - .01889\alpha u(k-2) + (2.0 - .02\alpha)y(k-1) \\
& + (.01889\alpha - 1)y(k-2) - .02\beta u^3(k-1) + .01889\beta u^3(k-2) \\
& - .01889\beta y^3(k-2) + .06\beta u^2(k-1)y(k-1) + .02\beta y^3(k-1) \\
& - .056670\beta u^2(k-2)y(k-2) - .06\beta y^2(k-1)u(k-1) \\
& + .056670\beta y^2(k-2)u(k-2) \tag{7.9}
\end{aligned}$$

Table 7.1 tabulates the resulting weighting coefficients using the three methods just described: (1) analytically calculated values using the truncated Taylor series, (2) least squares estimate of the third order sine approximation, as computed by Romeo [Ref. 7] and (3) the coefficient estimates as computed using the AKI algorithm.

The tabulation clearly shows that the AKI outperforms the first two methods overall. That is, the linear terms were identified without question; however, the AKI failed to identify the coefficients associated with the $y^3(i-1)$, $y^3(k-2)$, $u^2(k-2)y(k-2)$ and $y^2(k-2)u(k-2)$ terms. The reason for the failure is not known and was not investigated.

Table 7.1

NON-LINEAR WEIGHTING COEFFICIENTS CALCULATED
USING THREE METHODS

(1)	(2)	(3)
TAYLOR SERIES	LEAST SQUARES EST.	AKI USING ACTUAL
$\sin(x) = x - \frac{1}{3!}x^3$	$\sin(x) = \alpha x + \beta x^3$ $\alpha = .865, \beta = -.095$	$\sin(x)$ data base
$y(k-1)$	1.982700	1.980
$y(k-2)$	-.98366015	-.9815
$u(k)$	0.0	.00003691
$u(k-1)$.01730000	.02003
$u(k-2)$	-.01633985	-.01892
$u^3(k)$	0.0	.00001471
$u^3(k-1)$.00190	-.003278
$u^3(k-2)$	-.00179455	.003032
$y^3(k-1)$	-.00190	-.05695
$y^3(k-2)$.00179455	.05799
$u(k-1)y(k-1)$	-.005700	.01032
$u^2(k-2)y(k-2)$.00538365	-.009610
$y^2(k-1)y(k-1)$.005700	-.01640
$y^2(k-2)u(k-2)$	-.00538365	.01913

VIII. FINDINGS AND CONCLUSION

The performance of the Adaptive Kalman Identifier was compared to the LMS Adaptive Filter and the Adaptive Recursive LMS Filter using data derived from the models discussed in Chapter V. Results for the case where the order of the model (that is, m and n of equation (2.1b)) are assumed known are presented first followed by the analysis of the PLL data. The findings for the overmodeled case are presented next. The graphs show typical runs and do not represent ensemble averages.

A. ORDER OF THE UNKNOWN SYSTEM IS KNOWN

1. AKI vs. LMS Adaptive Filter

Using the MA form of the Adaptive Kalman Identifier, its performance can be compared against that of the LMS adaptive filter. Synthetic data derived from a plant whose transfer function is,

$$H(z) = 1.0 + 2.0z^{-1} + 3.0z^{-2} \quad (8.1)$$

was used. However, a fair comparison necessitated that the LMS Adaptive filter be "tuned" by adjustment of the convergence factor, k_s . Several values for k_s in the range $[-.600, -.200]$ were used. The objective of tuning the LMS filter was to achieve a fast convergence time with little or no steady state error while not compromising filter stability. The three filter weights were normalized and plotted for five

convergence factor values. It can be seen from Figures 8.1-8.3 that convergence is essentially reached by step thirty-five, ($k \approx 35$). As the convergence factor is further increased it can be noted in Figure 8.4 that the filter weights become more noisy, and that when $k_s = -.600$, filter stability is being compromised (Figure 8.5a).

Using the same data, the AKI converges to the MA coefficients in less than five iterations. Referring to Figure 8.5b, it was also noted that as the measurement noise $v(k)$ was increased, the LMS algorithm yielded more noisy estimates whereas the AKI tended to compensate for measurement noise. This is intuitively reasonable since the AKI incorporates into its algorithm the effects of measurement error due to noisy sensors. Comparing Figure 8.6 with Figure 8.7 the latter figure clearly shows the faster convergence of the AKI to the plant coefficient.

It was shown in Chapter III that the AKI gains would approach a steady state value and indeed they did, as Figure 8.8 shows. However, the instantaneous Kalman gains display a more erratic pattern as shown in Figure 8.9. The information of Figure 8.8 was gleaned from Figure 8.9 by computing the average of the individual gains at each iteration using the algorithm,

$$E\{\underline{G}(k+1)\} = E\{\underline{G}(k)\} + \frac{1}{k+1}[\underline{G}(k) - E\{\underline{G}(k)\}] \quad (8.2)$$

where,

$E\{\underline{G}(k+1)\}$ is the one step prediction of the average value for the gains,

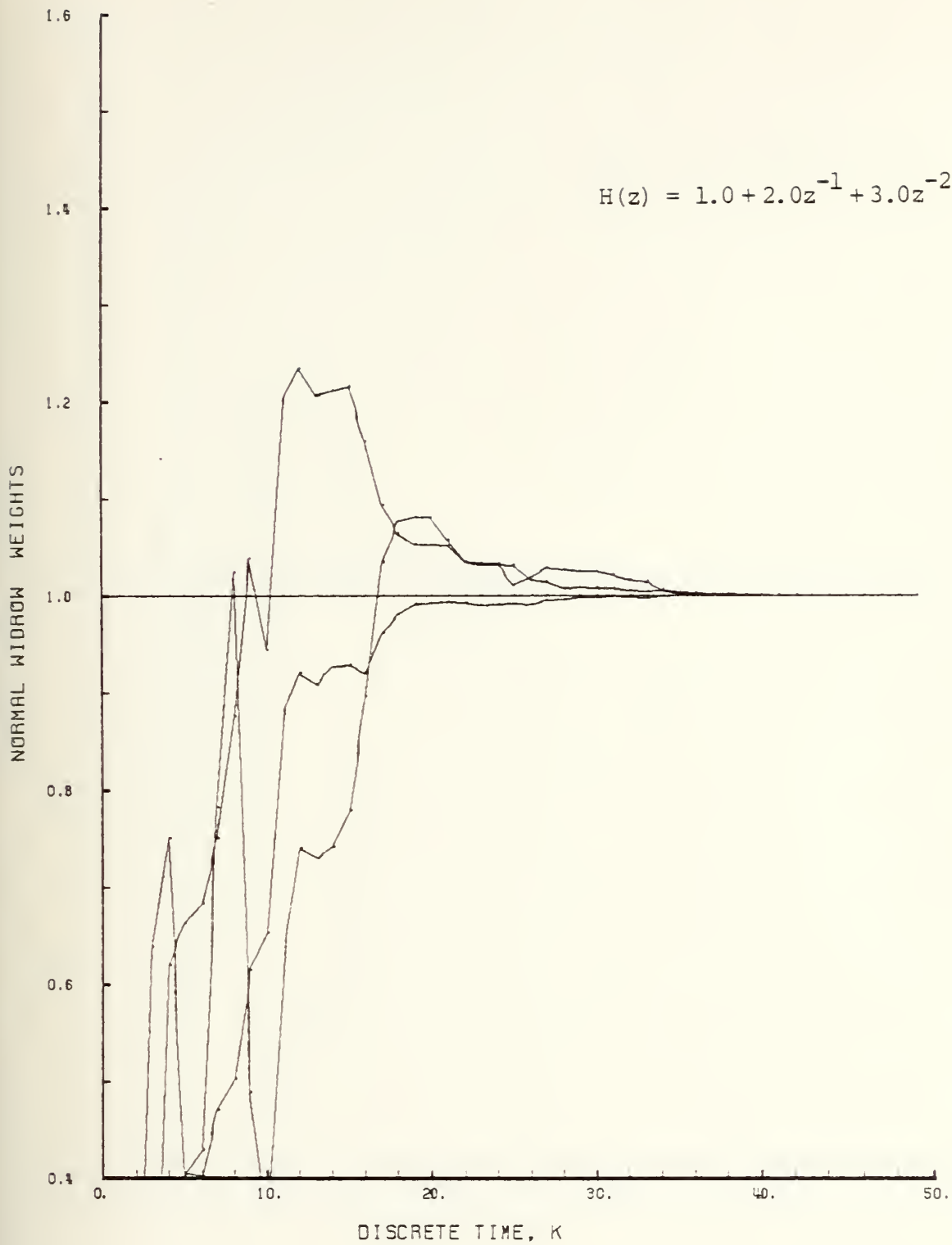


Figure 8.1. LMS Adaptive Filter Weights, $k_s = -.200$

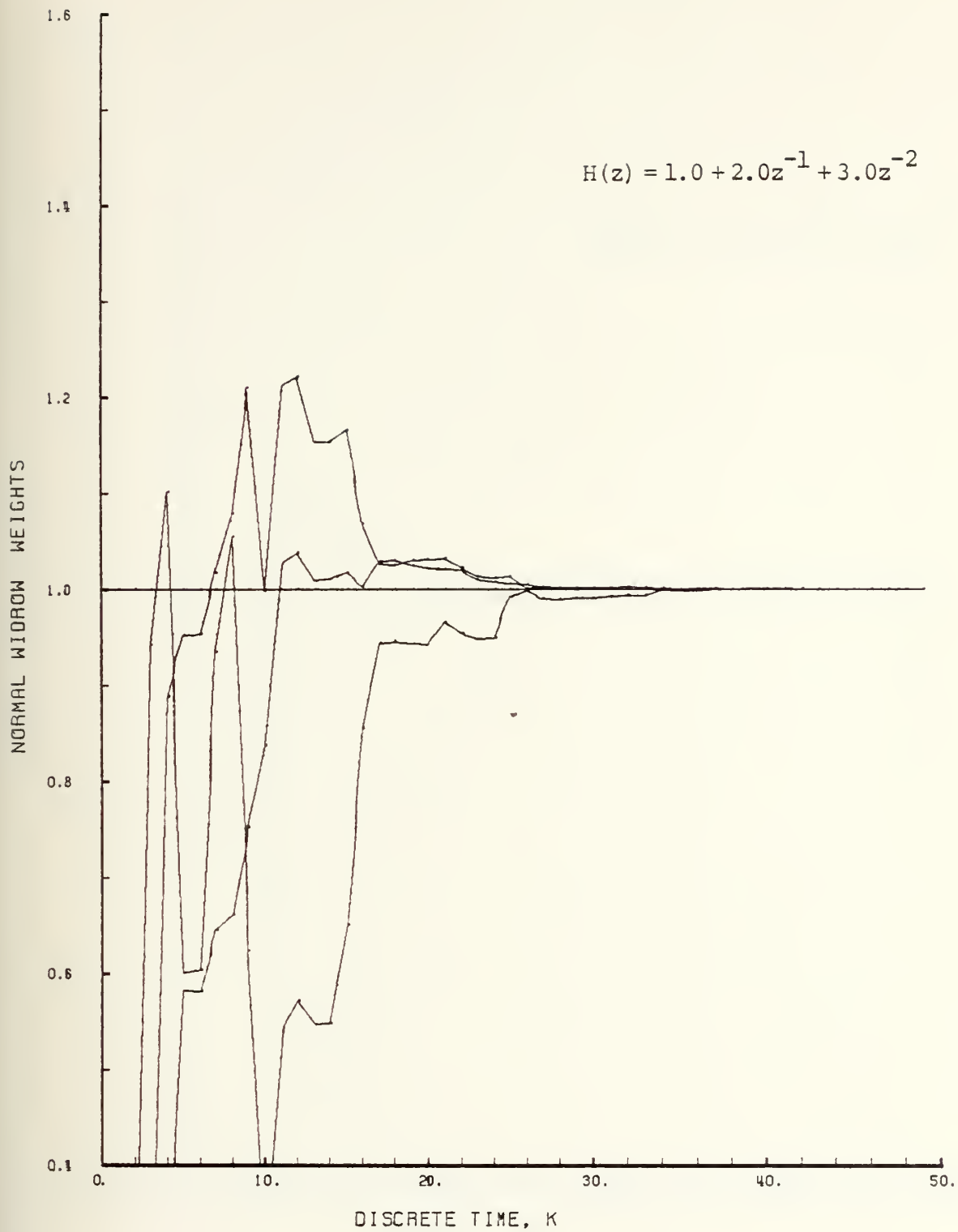


Figure 8.2. LMS Adaptive Filter Weights, $k_s = -.300$

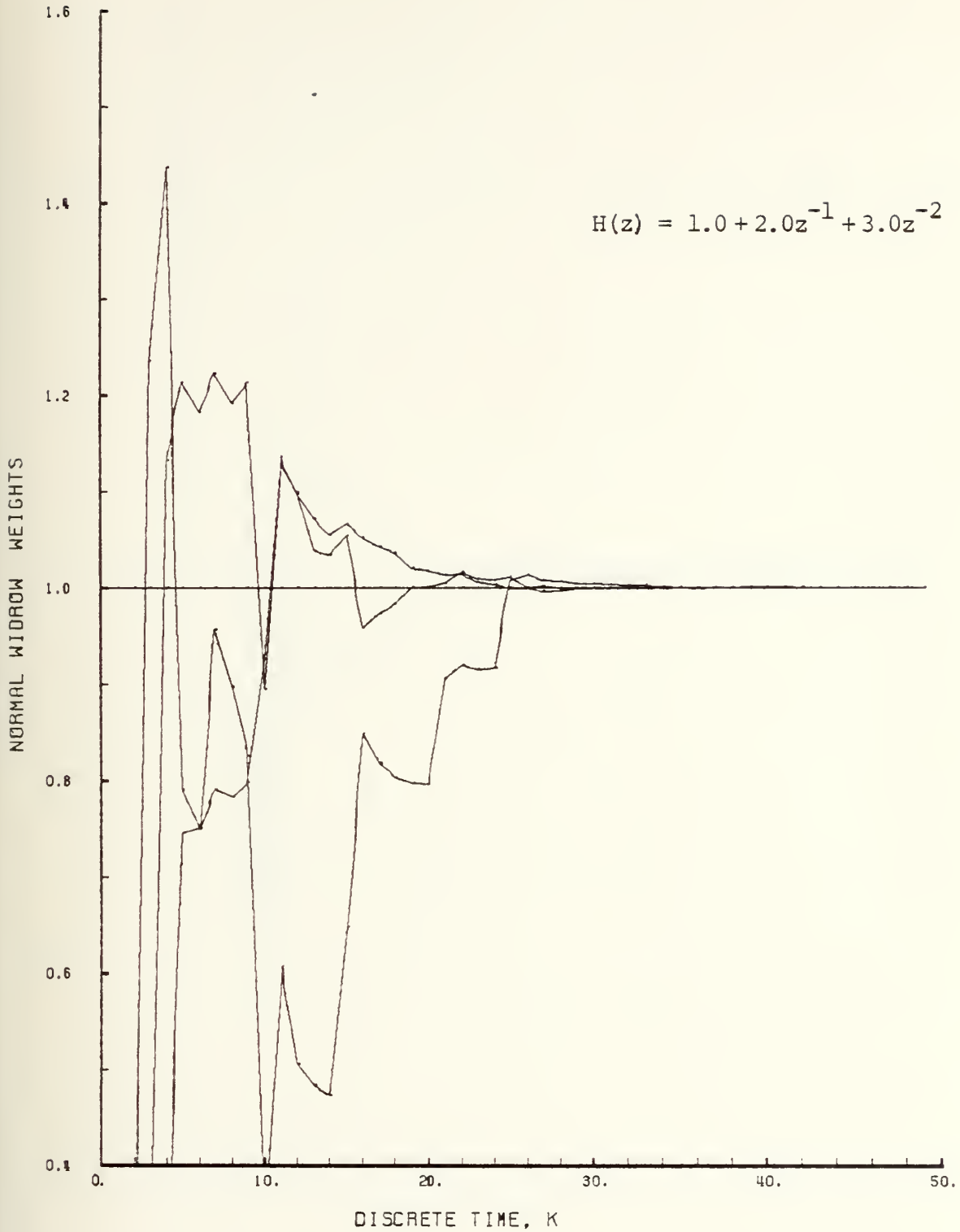


Figure 8.3. LMS Adaptive Filter Weights, $k_s = -.400$

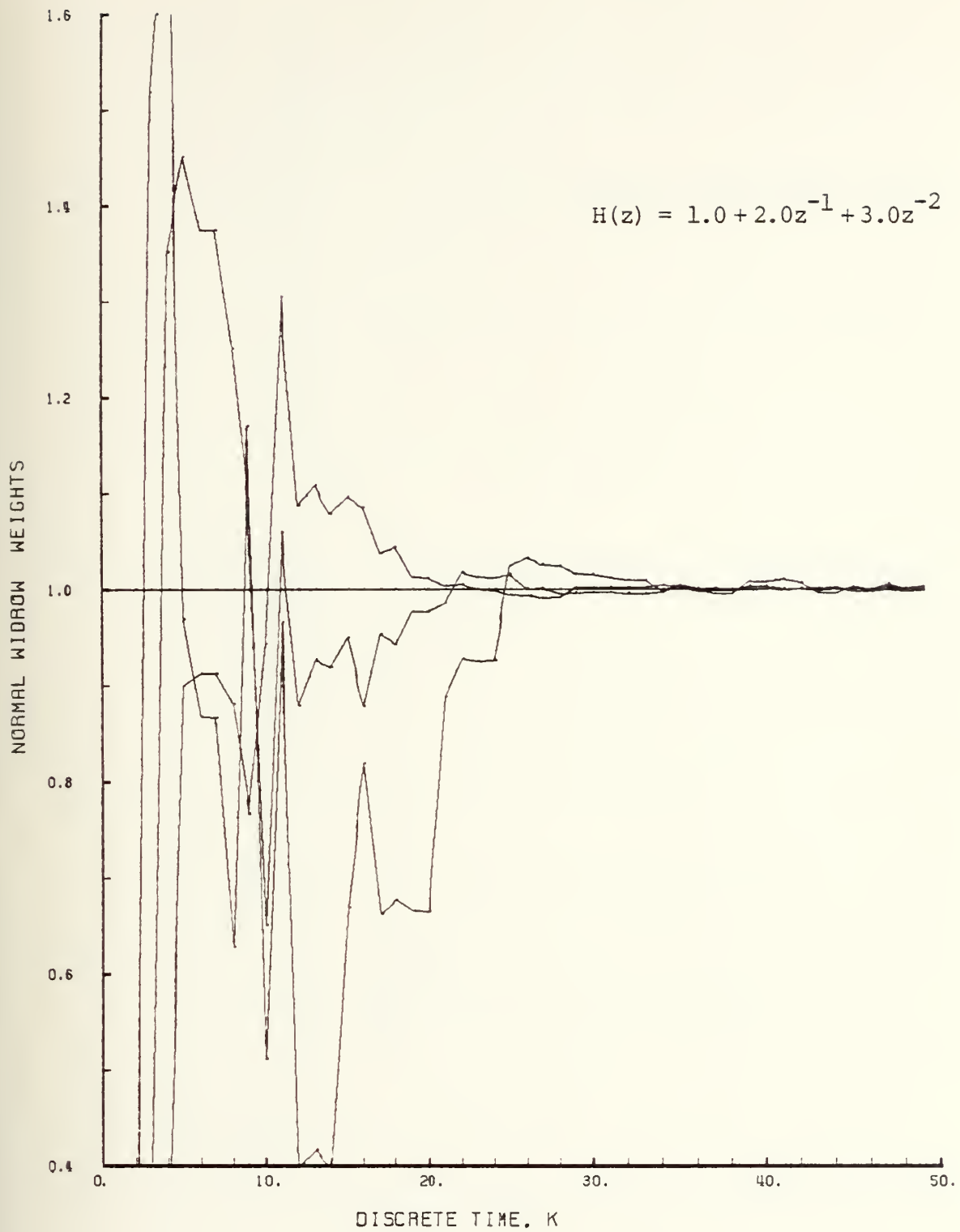


Figure 8.4. LMS Adaptive Filter Weights, $k_s = -.500$

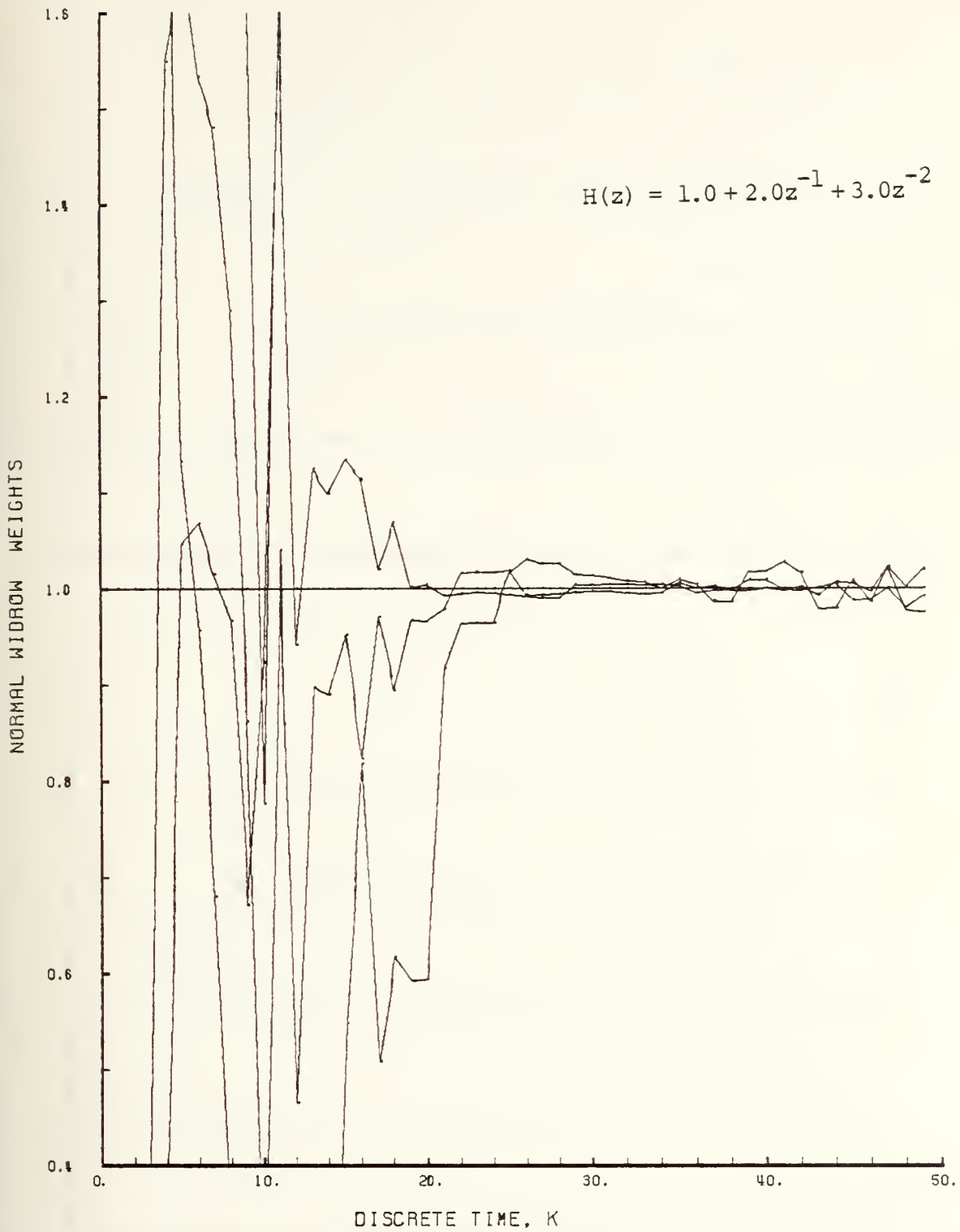


Figure 8.5a. LMS Adaptive Filter Weights, $k_s = -.600$

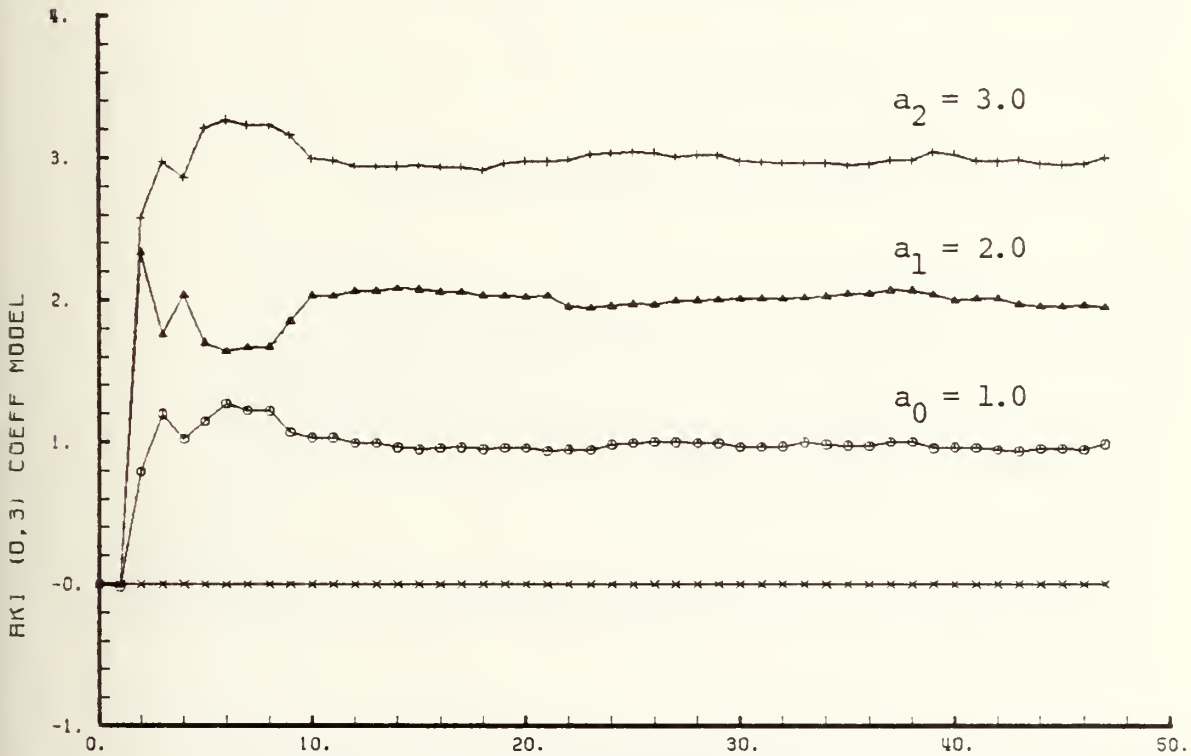
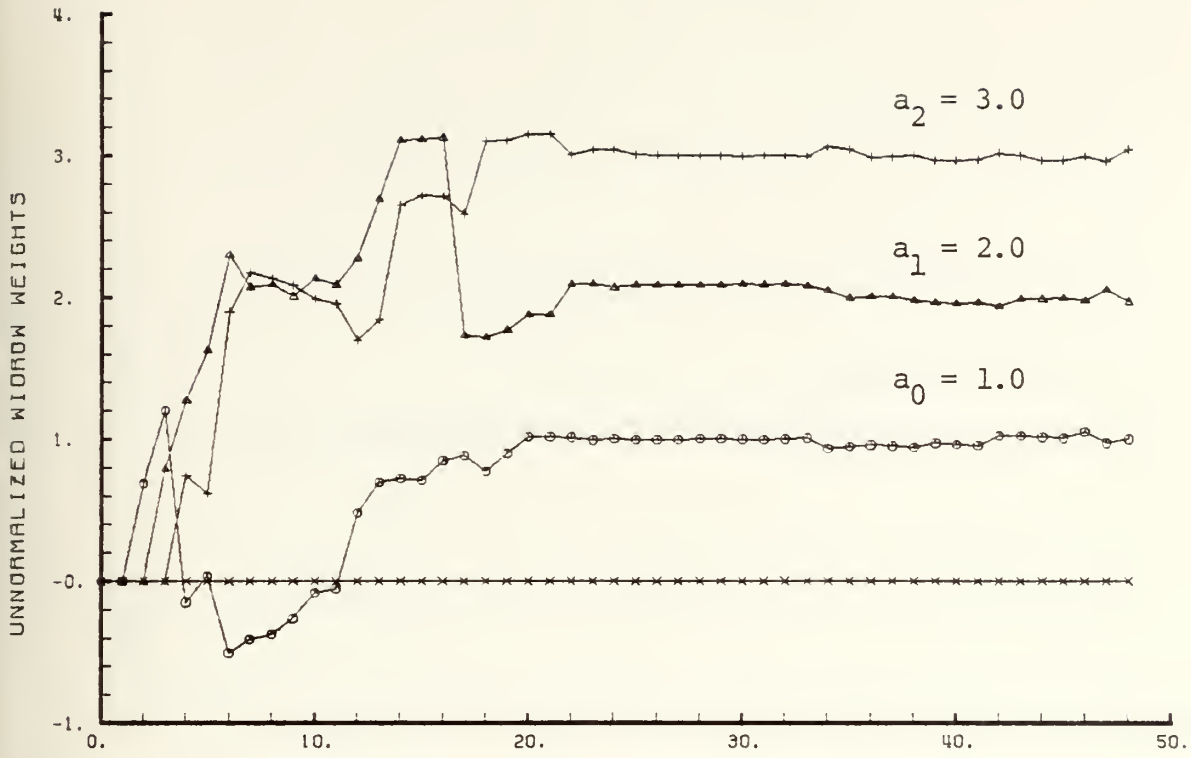


Figure 8.5b. LMS Adaptive Filter Weights, $k_s = -.200$, with increased measurement noise

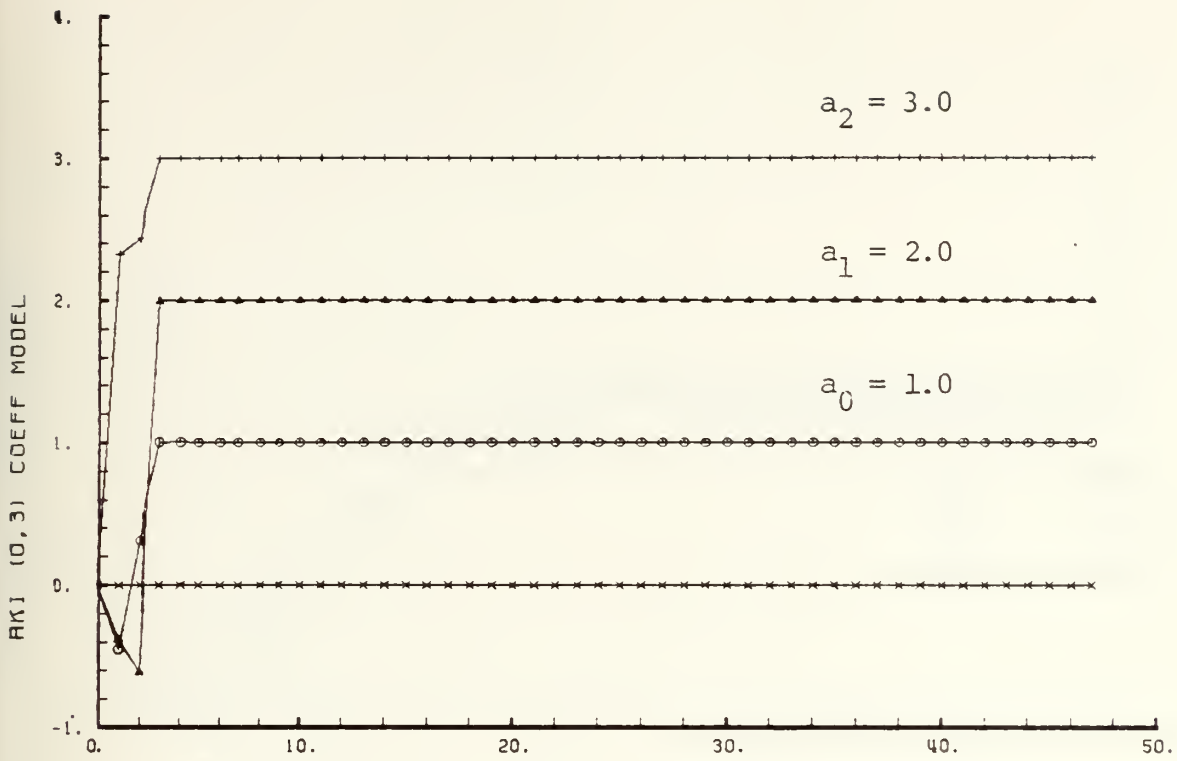


Figure 8.6. AKI Moving Average Plant Coefficients

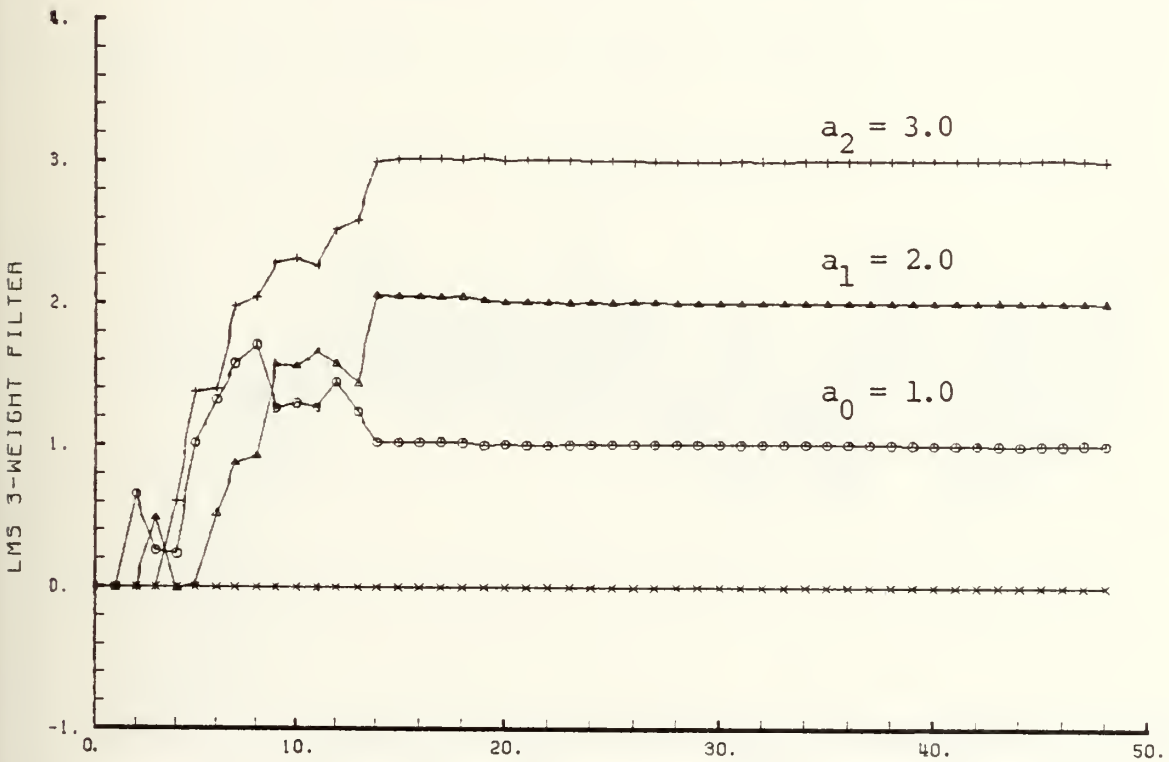


Figure 8.7. LMS Adaptive Filter Weights

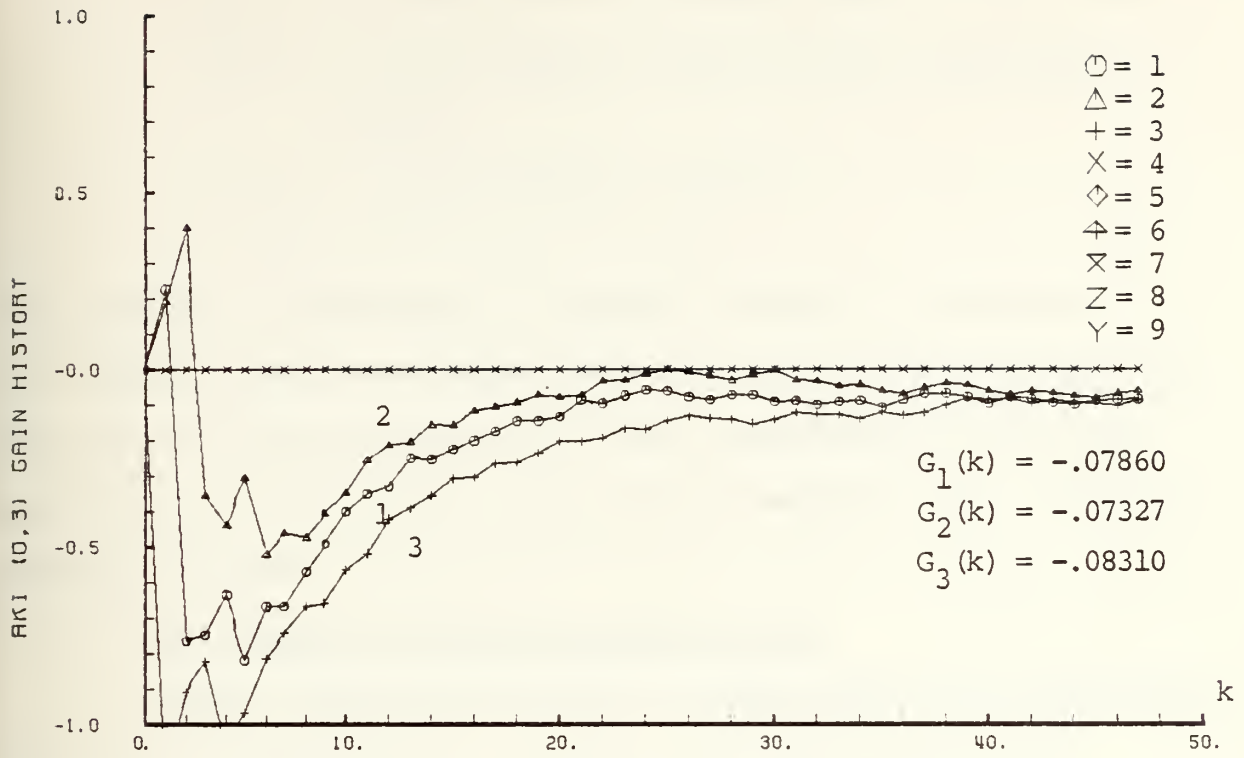


Figure 8.8. AKI Averaged Gains for Moving Average Model

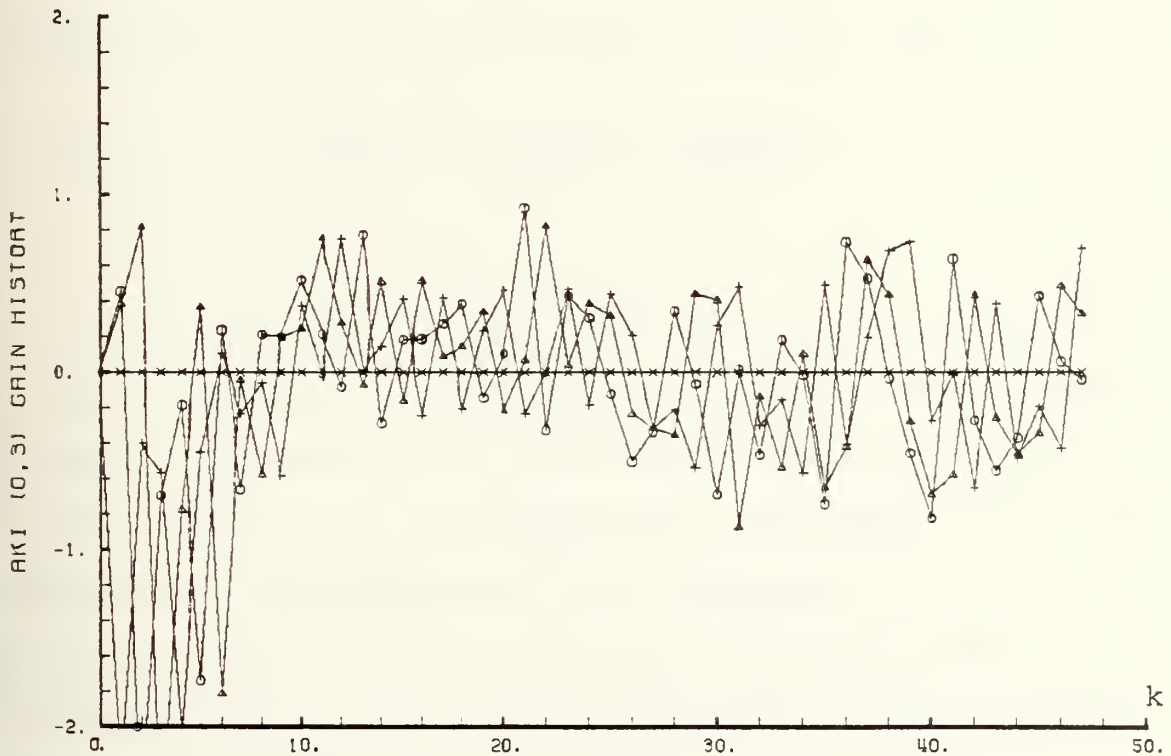


Figure 8.9. AKI Moving Average Instantaneous Gains

$E\{\underline{G}(k)\}$ is the present average value for the gain

$\underline{G}(k)$ is the present value of the instantaneous gains

$\frac{1}{k+1}$ is the averaging gain.

This method of obtaining a "running" average is generally well known, (see for example Ref. 39). The example presented was not the only one used but a good representative of the operation of the AKI when the data was derived from a moving average process.

2. AKI Applied to Autoregressive Data

Before applying the AKI to identifying the coefficients of a complex ARMA plant, it was first tested using data derived from a plant whose transfer function specified an autoregressive process. The transfer function used was,

$$\begin{aligned} H(z) &= \frac{1}{1.0 + \frac{26.0}{24.0}z^{-1} + \frac{9.0}{24.0}z^{-2} + \frac{1.0}{24.0}z^{-3}} \\ &= \frac{a_0}{1 - b_1z^{-1} - b_2z^{-2} - b_3z^{-3}} \end{aligned} \quad (8.3)$$

The AKI had no problem converging on the plant coefficients (Figure 8.10) producing the following estimates at the 42nd iteration (Table 8.1). It essentially converged on the plant coefficients in approximately five iterations. The gains computed by the AKI for this case also displayed convergence to a steady state value. The gain history for this example is depicted in Figure 8.11.

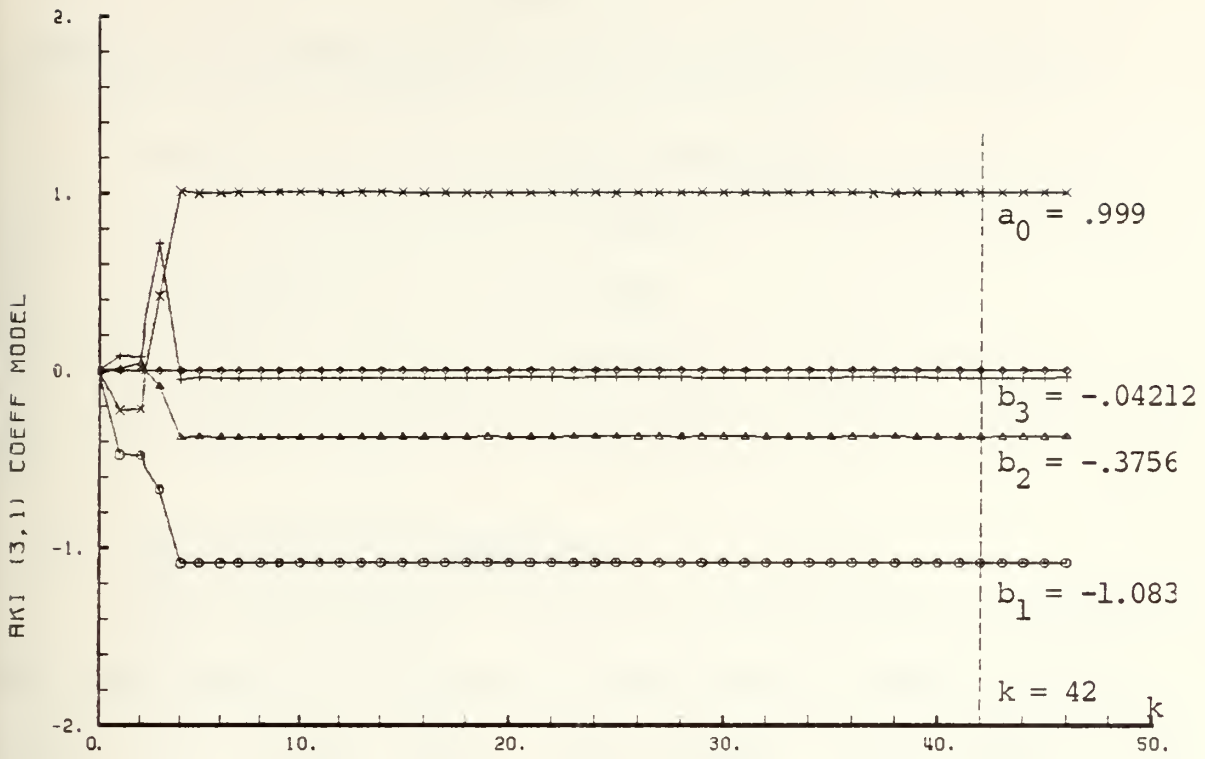


Figure 8.10. AKI Autoregressive Plant Coefficients

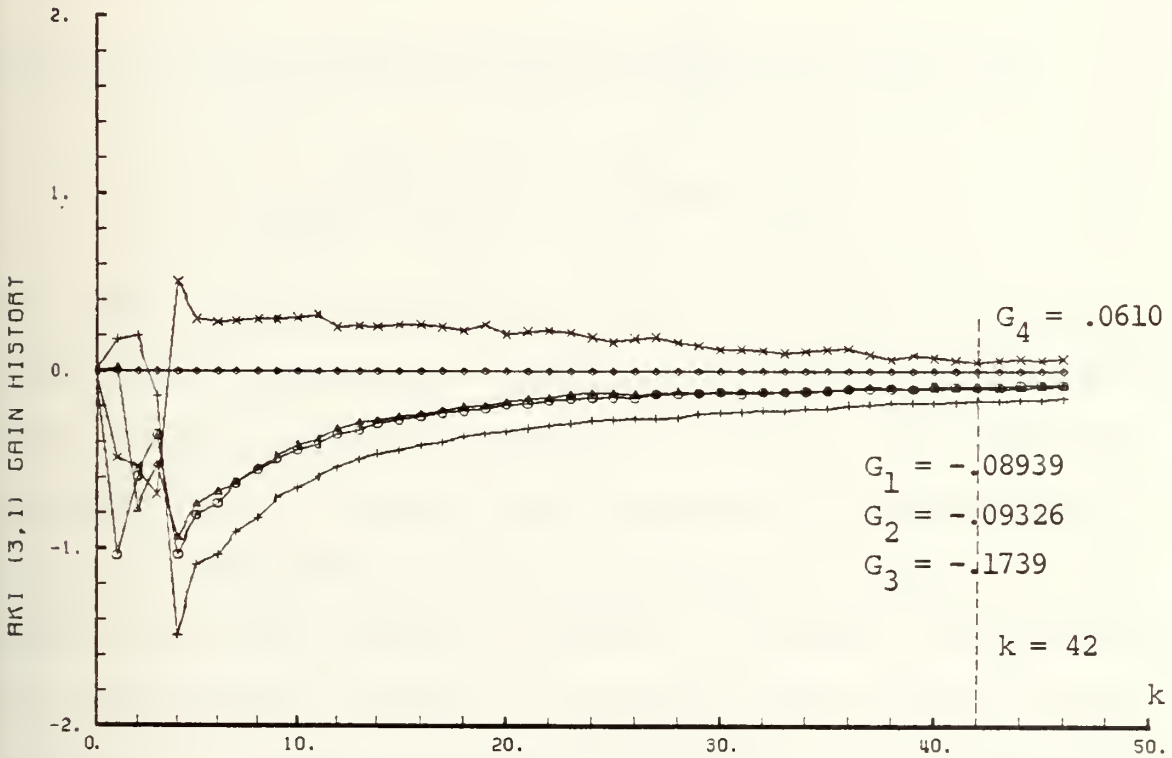


Figure 8.11. AKI Autoregressive Averaged Gains

Table 8.1

ACTUAL VS AKI ESTIMATES FOR COEFFICIENTS OF EQUATION 8.3

	ACTUAL	AKI ESTIMATES
a_0	1.0	0.9995
b_1	-1.08333 ...	-1.083
b_2	-0.3750	-0.3756
b_3	-0.04166 ...	-0.04212

3. AKI Applied to ARMA Data

The next logical step was to use the AKI to identify the coefficients of a general ARMA process. One of Perry's models [Ref. 6] was used for this purpose. Specifically, the transfer function of the plant was,

$$\begin{aligned}
 H(z) &= \frac{1.0 + 1.4z^{-1} + .98z^{-2}}{1 - 1.14z^{-1} + 1.4549z^{-2} - .88490z^{-3} + .40745z^{-4}} \\
 &= \frac{a_0 + a_1z^{-1} + a_2z^{-2}}{1 - b_1z^{-1} - b_2z^{-2} - b_3z^{-3} - b_4z^{-4}} \quad (8.4)
 \end{aligned}$$

The ARMA plant was subjected to the same conditions which Perry [Ref. 6] describes. That is, unit variance, zero mean, white gaussian noise was used as the input. The reader is reminded that the output signal processed by the AKI was corrupted by measurement noise as described in Chapter V. The output data used in Perry's examples, however, reflects noiseless measurements. Table 8.2 tabulates the results for the coefficient estimates computed by the AKI. Even though the results presented in Table 8.2 represent the coefficient

Table 8.2

ACTUAL VS AKI ESTIMATES FOR COEFFICIENTS OF EQUATION
8.4 AT THE 371ST ITERATION

	ACTUAL	AKI ESTIMATES
a_0	1.00000	0.98760
a_1	1.40000	1.40100
a_2	0.98000	0.99280
a_3	0.00000	0.01262
a_4	0.00000	-0.00922
b_1	1.14000	1.14000
b_2	-1.45490	-1.45900
b_3	0.88490	.88740
b_4	-0.40745	-.40980

estimates at the 371st iteration, it can be seen from Figure 8.12 that the AKI has essentially converged by the 28th iteration. We note also the characteristic convergence of the averaged gains to steady state values in Figure 8.13. As a means of comparison with Perry's results, the poles and zeros of the AKI estimates at the 28th, 90th and 371st iteration are plotted in Figure 8.14a and Figure 8.14b. Perry's results Figure 3.7 [Ref. 6:pp. 107,108] for his lattice modeling of the plant represented by equation (8.4) are reproduced for convenience.

As was noted for the previous cases, the instantaneous gains appeared erratic, Figure 8.15, whereas the averaged gains converged to some steady state value.

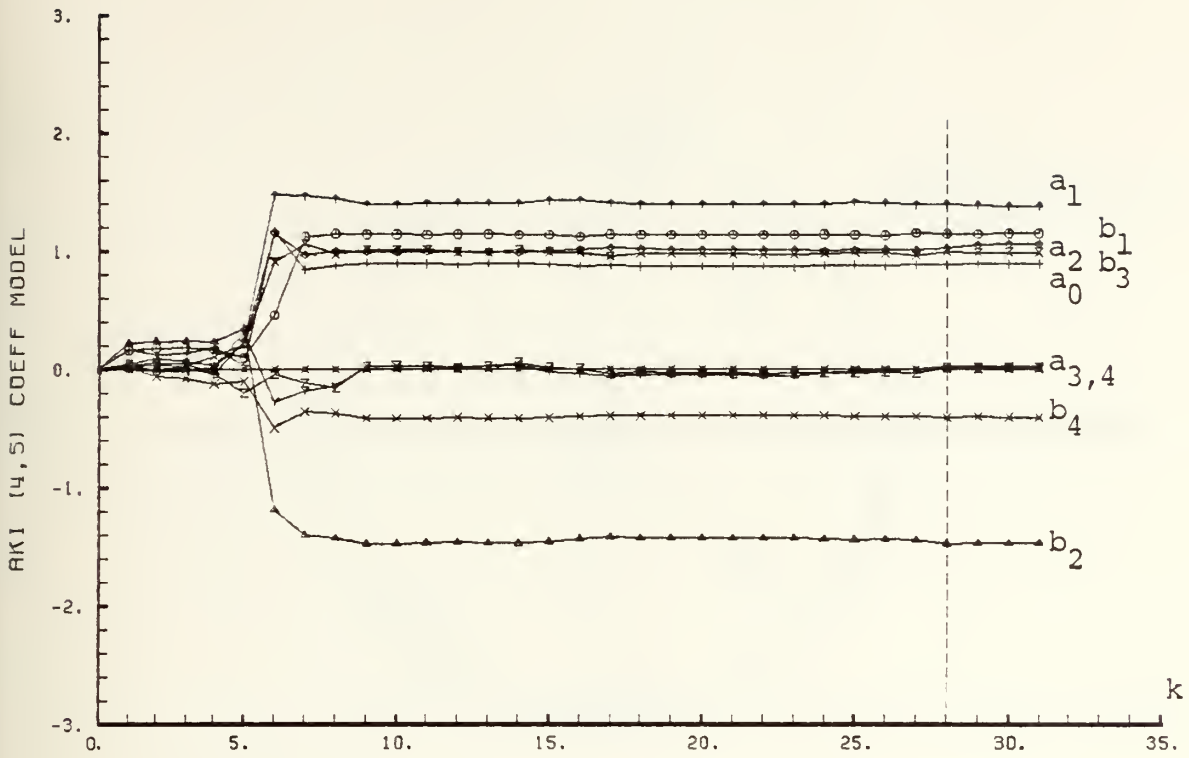


Figure 8.12. AKI(4,5) Computer Coefficients, Perry's Example

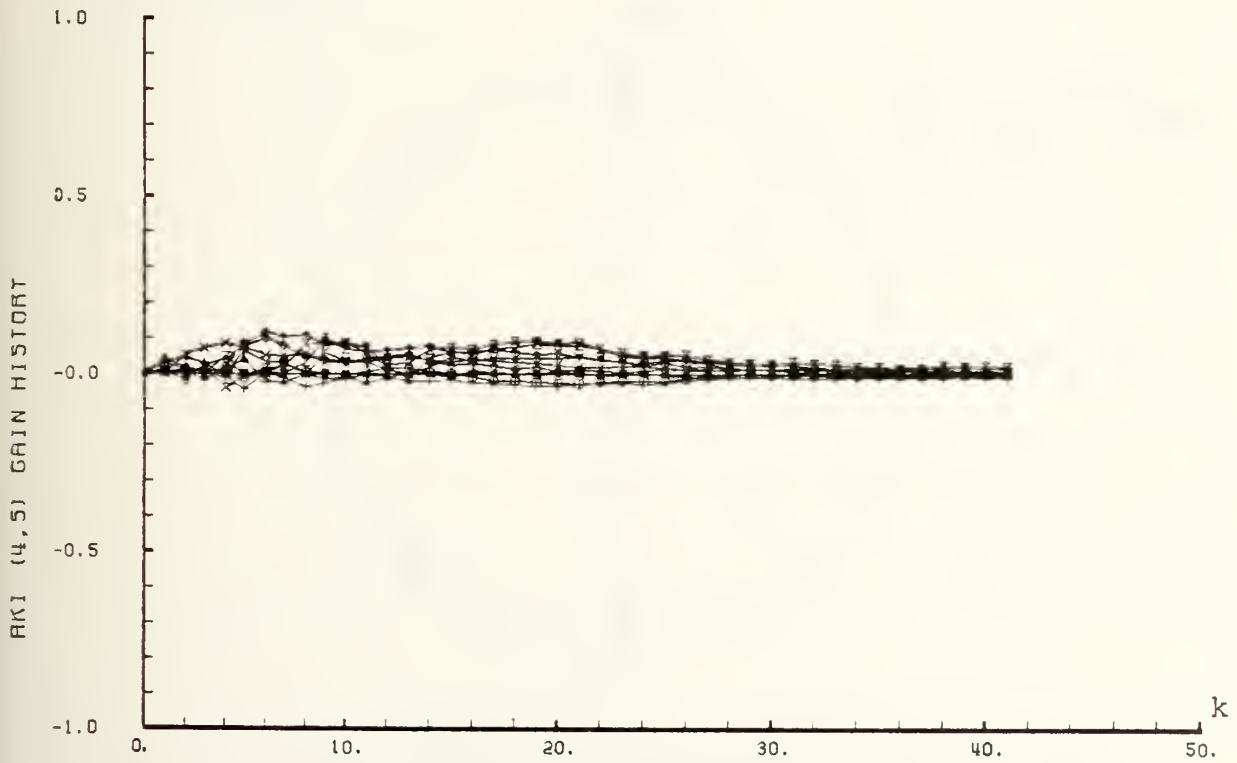
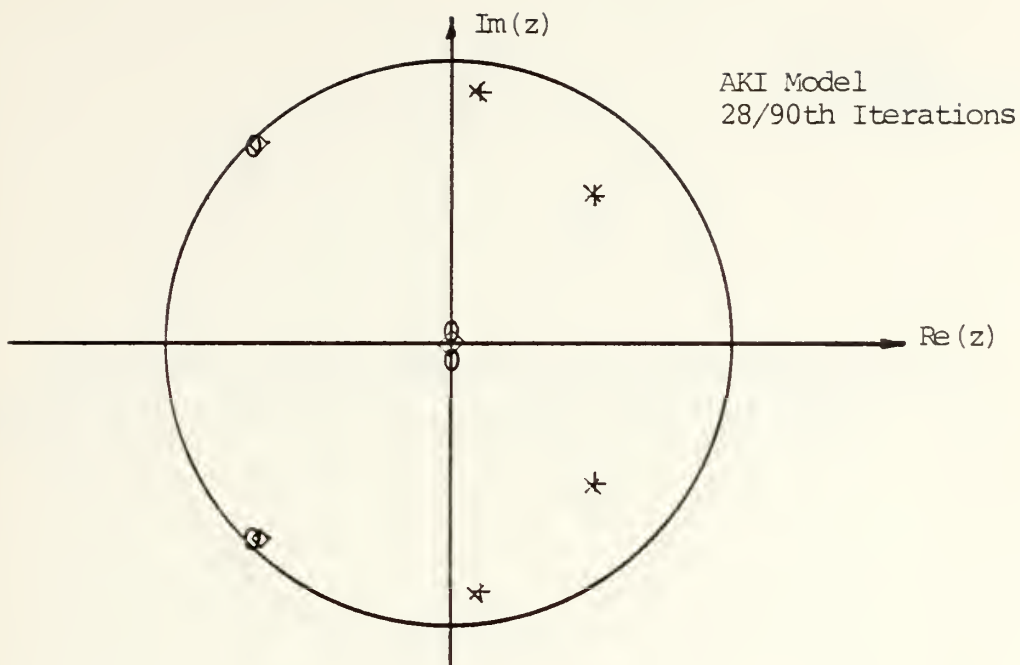


Figure 8.13. AKI(4,5) Averaged Gain Values



X Model Pole
O Model Zero
+ System Pole
◇ System Zero

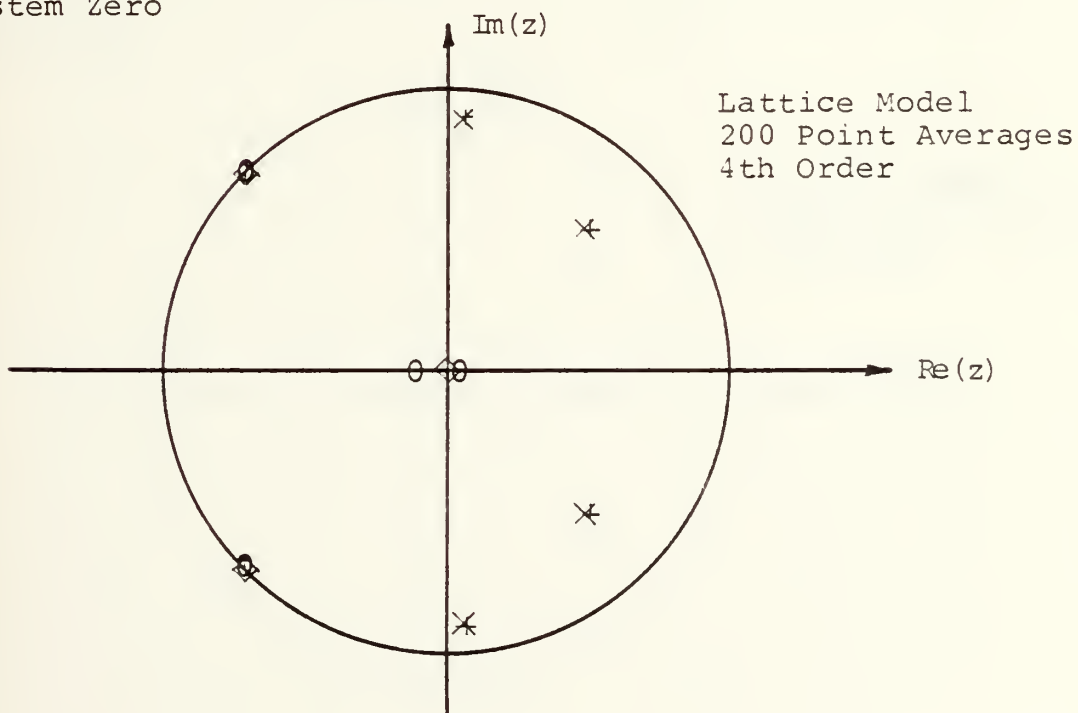
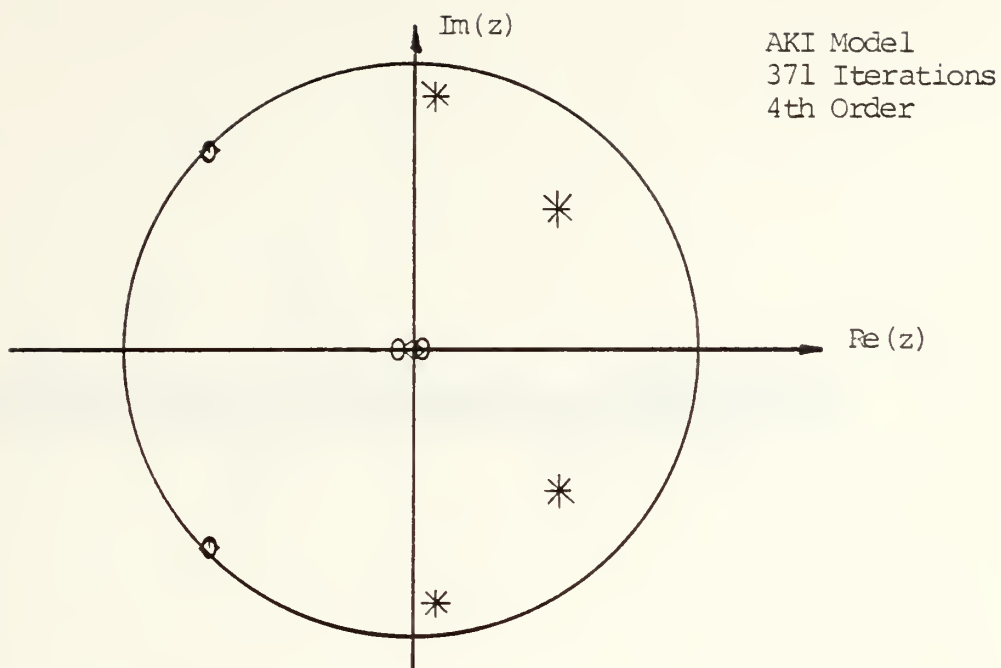


Figure 8.14a. Pole/Zero Models Produced by AKI and Lattice Algorithms for a Plant with the Characteristic Transfer Function of Eqn. (8.4)



X Model Pole
O Model Zero
+ System Pole
◇ System Zero

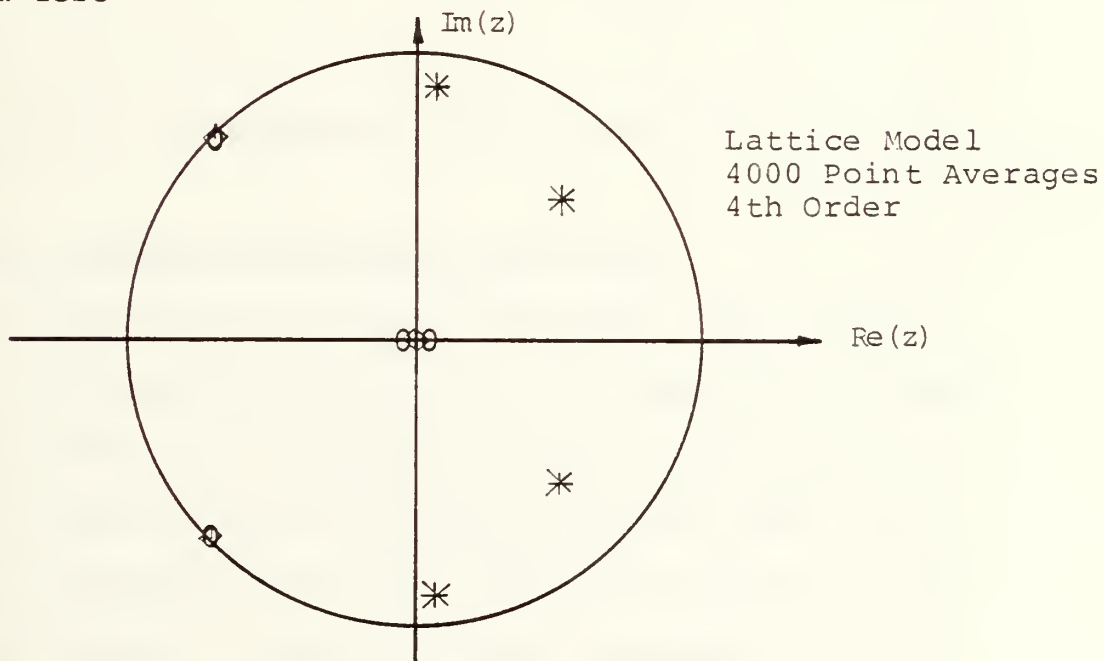


Figure 8.14b. Pole/Zero Models Produced by AKI and Lattice Algorithms for a Plant with the Characteristic Transfer Function of Eqn. (8.4) at 371st Iteration

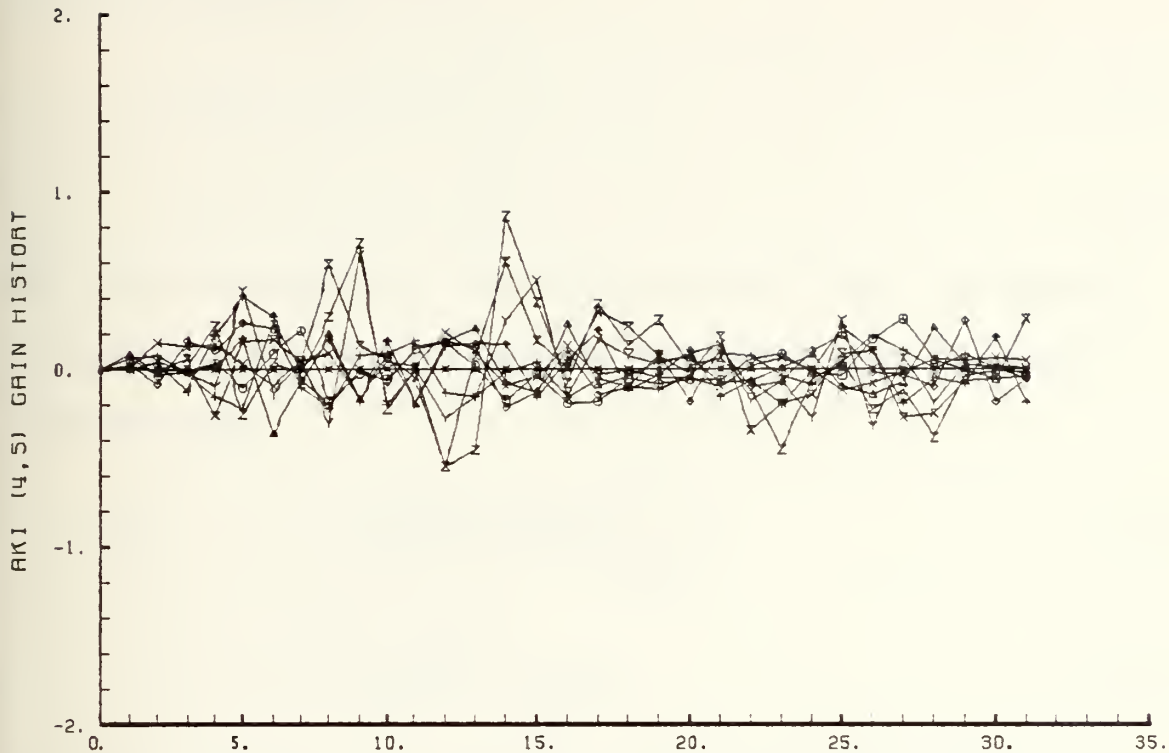


Figure 8.15. Instantaneous Gain Values, AKI(4,5) Model

4. AKI vs Adaptive Recursive LMS Filter

Using Feintuch's proposed algorithm [Ref. 18] and repeating the simulation presented in the rebuttal to Johnson and Larimore [Ref. 19], a comparison was made between the operation of the AKI and the Adaptive Recursive LMS filter. The Adaptive Recursive LMS filter required "tuning" of the convergence factors, k_a and k_b , to what appeared to be the optimal performance features of (1) fast convergence and

(2) stable estimates. The "tuning" process was a trial and error procedure beginning with the convergence constants given by Feintuch,

$$k_a = -4.3 \times 10^{-4} \quad (8.5a)$$

$$k_b = -4.3 \times 10^{-4} \quad (8.5b)$$

The Adaptive Recursive LMS algorithm was then implemented to estimate the coefficients of the plant whose transfer function was,

$$H(z) = \frac{0.05 - 0.40z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}} \quad (8.6a)$$

$$= \frac{a_0 + a_1z^{-1}}{1 - b_1z^{-1} - b_2z^{-2}} \quad (8.6b)$$

The best response obtained by trial and error resulted in the convergence constants,

$$k_a = -4.3 \times 10^{-3} \quad (8.7a)$$

$$k_b = -14.3 \times 10^{-3} \quad (8.7b)$$

Feintuch reported that after $8,192^2$ iterations the estimates had converged on the coefficients of (8.6a) with a .2096 normalized rms error. Using the convergence constants (8.7) convergence was essentially reached by the $4,000^{\text{th}}$ iteration.

²Feintuch reports the resulting estimates for the coefficients at several intermediate iterations from 8,192 to 65,536, however 8,192 was the minimum number of iterations reported.

It was felt that the operation of the adequately tuned Adaptive Recursive LMS filter could be fairly compared with the operation of the AKI.

Table 8.3 tabulates the performance of the two system identification methods and Figures 8.16 and 8.17 graphically present the responses. Figure 8.18 not only shows at what point the gains of the AKI reach a constant value but also

Table 8.3

ACTUAL VS AKI AND ADAPTIVE RECURSIVE LMS FILTER
ESTIMATES FOR COEFFICIENTS OF EQUATION 8.6

	ACTUAL	AKI ESTIMATES (k = 150)	ADAPTIVE RECURSIVE LMS (k = 3,990)
a_0	0.05	0.05107	0.05045
a_1	-0.40	-0.4007	-0.3990
b_1	1.1314	1.13200	1.12800
b_2	-0.25	-0.25090	-0.2442

provides a measure of confidence that the unknown system has been identified. It is evident from Figure 8.19 that both methods seem to identify the poles and zeros of the actual plant; however, the poles computed by the AKI are closer. The zeros computed by the AKI are approximately .080 units farther from the true zero than is the Adaptive Recursive LMS filter estimate. All aspects considered, the AKI takes considerably fewer iterations to arrive at its estimate.

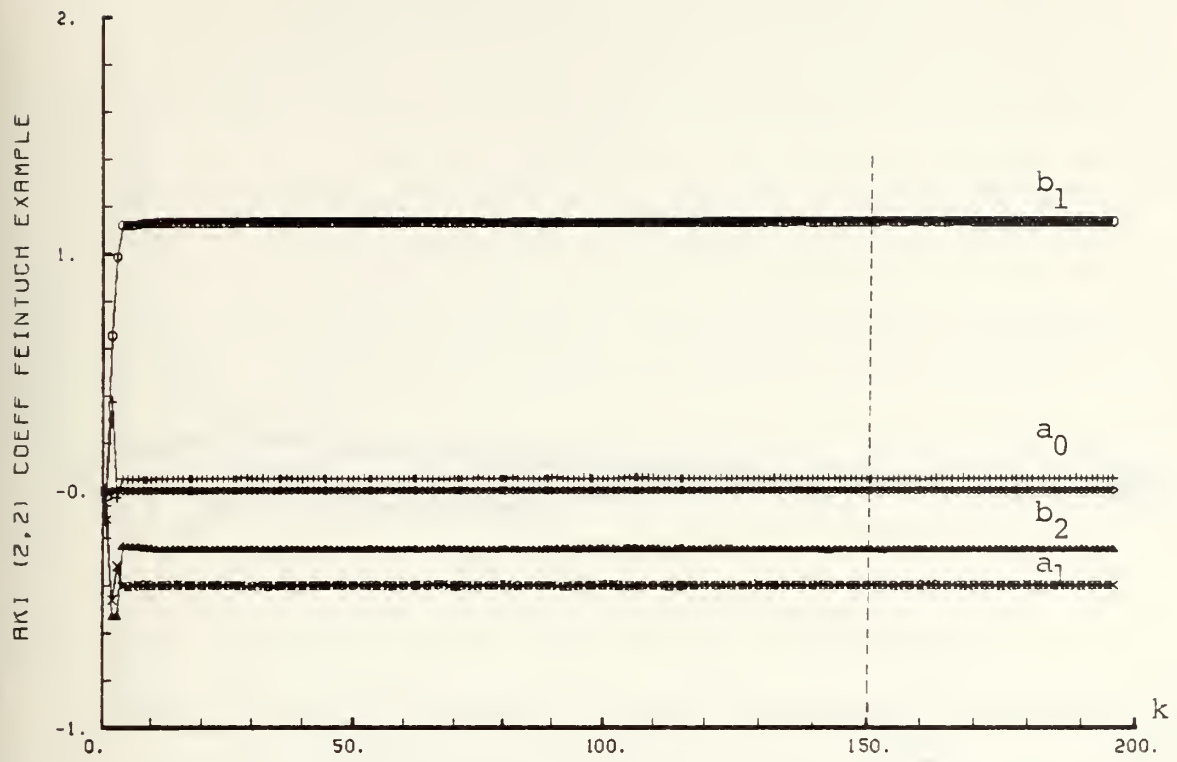


Figure 8.16. AKI(2,2) Computed Coefficients, Feintuch Example

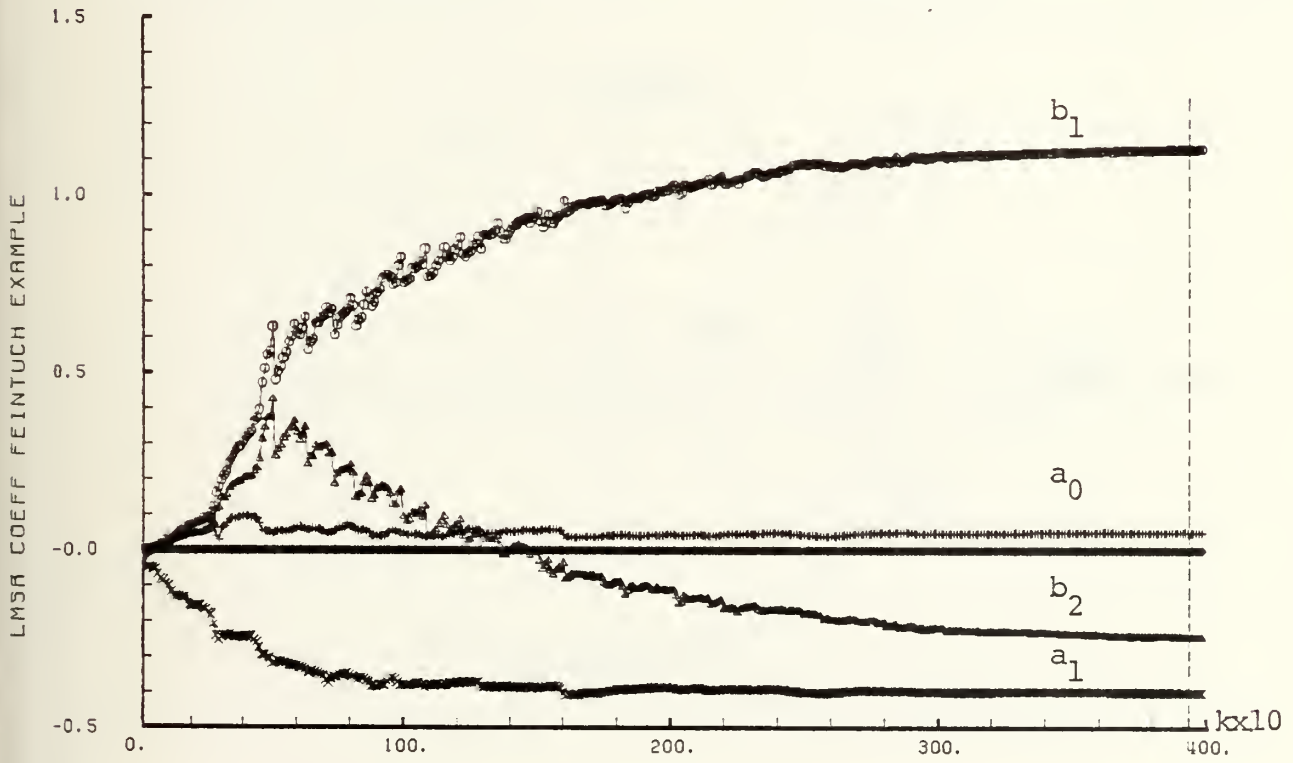


Figure 8.17. LMSR Computed Coefficients, Feintuch Example

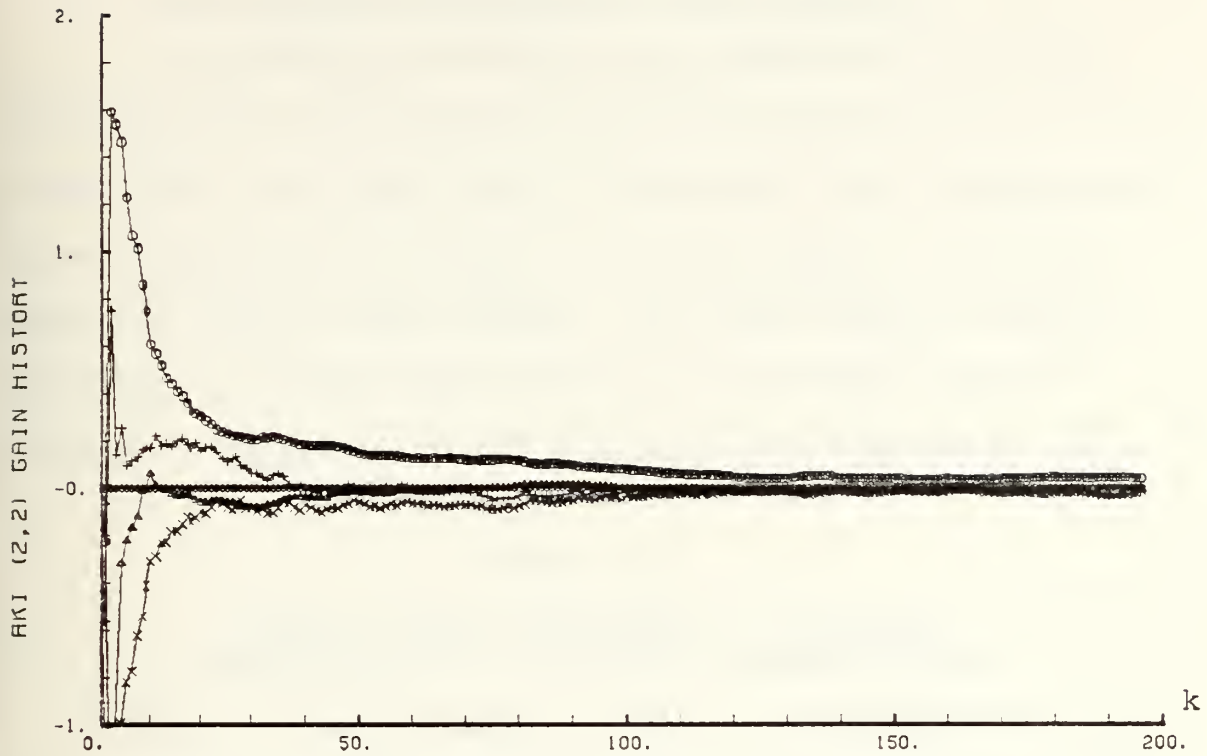


Figure 8.18. AKI(2,2) Averaged Gains, Feintuch Example

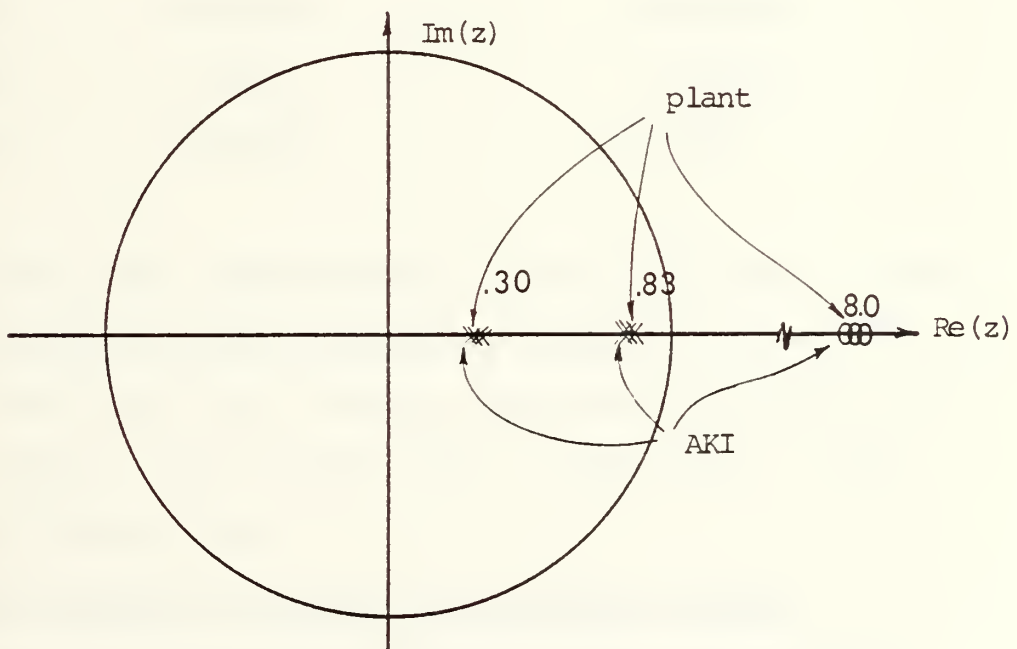


Figure 8.19. Pole/Zero Models Produced by AKI and Adapt. Rec. LMS Algorithms for a Plant with the Characteristic Transfer Function of Eqn. (8.6)

5. AKI Applied to PLL Data (Linear Region)

To identify the coefficients associated with the general autoregressive-moving average representation for the phase locked loop, the input noise signal power (mean square value) was kept at $(10 \text{ deg})^2$. In this manner, the linear region of the PLL was invoked. The input/output data was analyzed by the AKI resulting in the estimates shown in Table 8.4. Figures 8.20 and 8.21 show the response of the AKI

Table 8.4

ACTUAL VS AKI ESTIMATES OF THE ARMA REPRESENTATION FOR THE PLL (LINEAR REGION)

	ACTUAL	AKI ESTIMATES (k = 82)	AKI ESTIMATES (k = 350)
a_0	0.000000	-0.0001046	-0.000692
a_1	0.020000	0.0198500	0.0199800
a_2	-0.018890	-0.018720	-0.0188900
b_1	1.980000	1.980000	1.978000
b_2	-0.981110	-0.981000	-0.9790

when applied to the identification of the PLL data. It was noted that though the AKI correctly identified the coefficients of the linear PLL, the AKI gains were large due to the weak signals (input/output data) incorporated in the measurement vector, $H(k)$.

6. AKI Applied to PLL Data (Non-linear Region)

When analyzing any non-linear system the engineer must bring to bear all his analysis techniques on the problem. The PLL was therefore studied using classical root locus

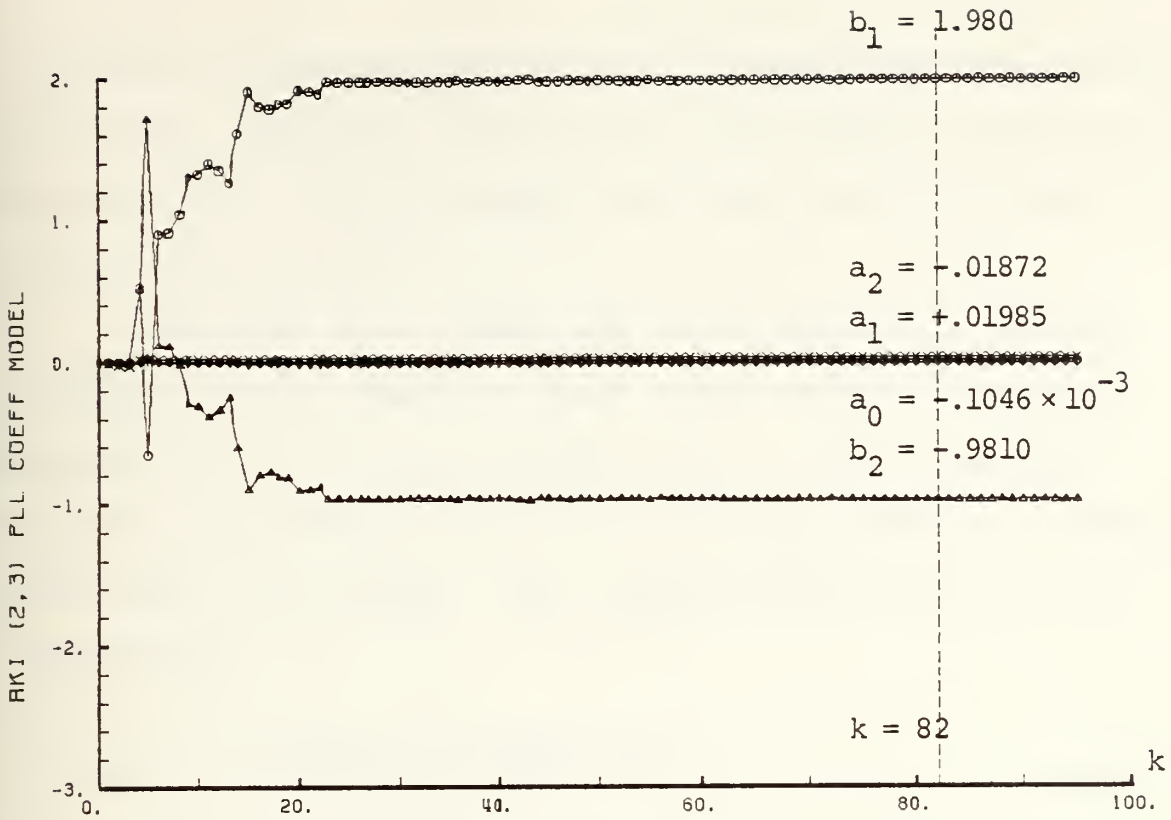


Figure 8.20. PLL, AKI(2,3) Computed Coefficients

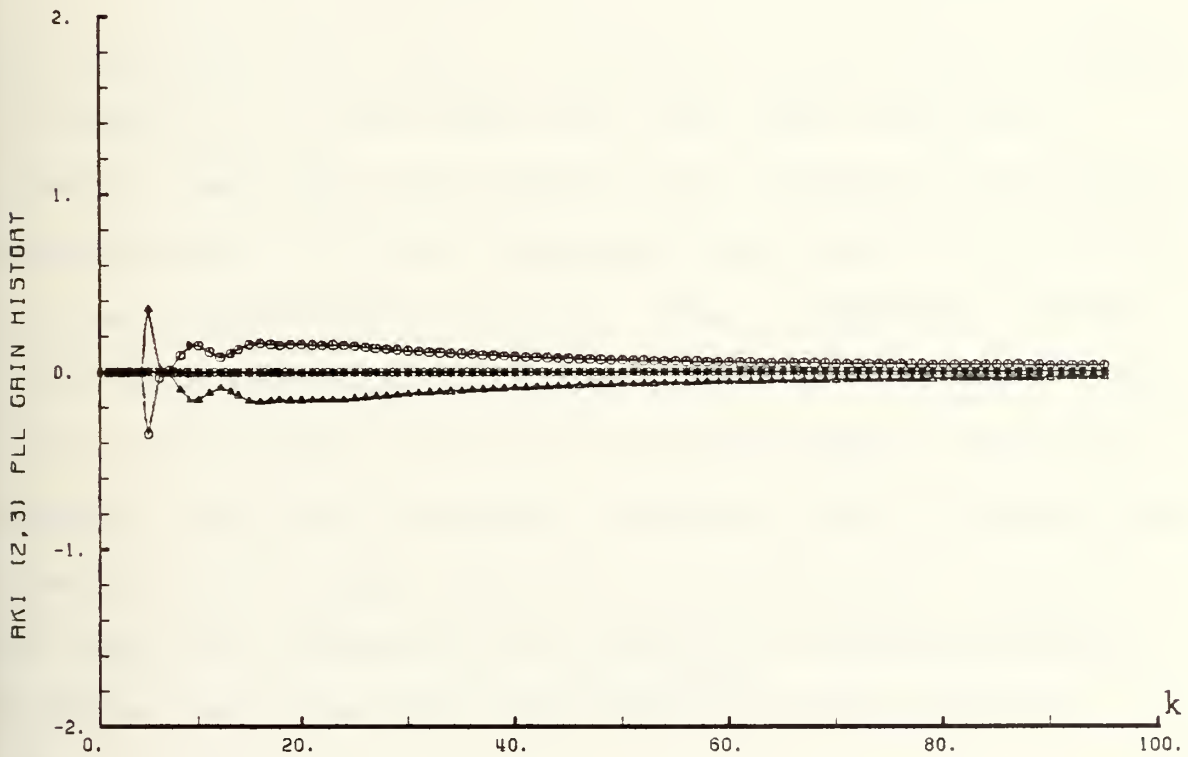


Figure 8.21. Averaged Gain Values, AKI(2,3) PLL Model

techniques to preview the possible outcomes when being modeled by the AKI. Stability analysis of a more complex PLL system using the root locus technique has been previously presented in the literature [Ref. 40].

The root locus technique can be applied to the PLL presented in this thesis by first assuming that the $\sin(\cdot)$ block of Figure 5.7 is a variable gain, λ . This is not a restrictive assumption since the PLL during operation generally tracks small deviations. The loop gain can be written by inspection as,

$$\begin{aligned}
 L(z) &= \frac{\lambda(.02)(1 - .9445z^{-1})z^{-1}}{(1 - z^{-1})^2} & (8.8) \\
 &= \frac{\lambda(.02)(z - .9445)z}{(z - 1)^2}
 \end{aligned}$$

From equation (8.8) the root locus of $L(z)$ is drawn as shown in Figure 8.22. Even though the root locus technique is generally used when the signals in the system are considered deterministic, it is not surprising that some of the results obtained are nevertheless valid. When a moderately strong input noise identification signal $[E\{u^2(u)\} < (25 \text{ deg})^2]$ was used, the pole-zero locations of the PLL seem to follow the classical root locus analysis. However, when the input noise identification signal power is increased beyond $(25 \text{ deg})^2$ the pole-zero locations do not follow the expected behavior predicted using the root locus method as can be seen in Figure 8.22.

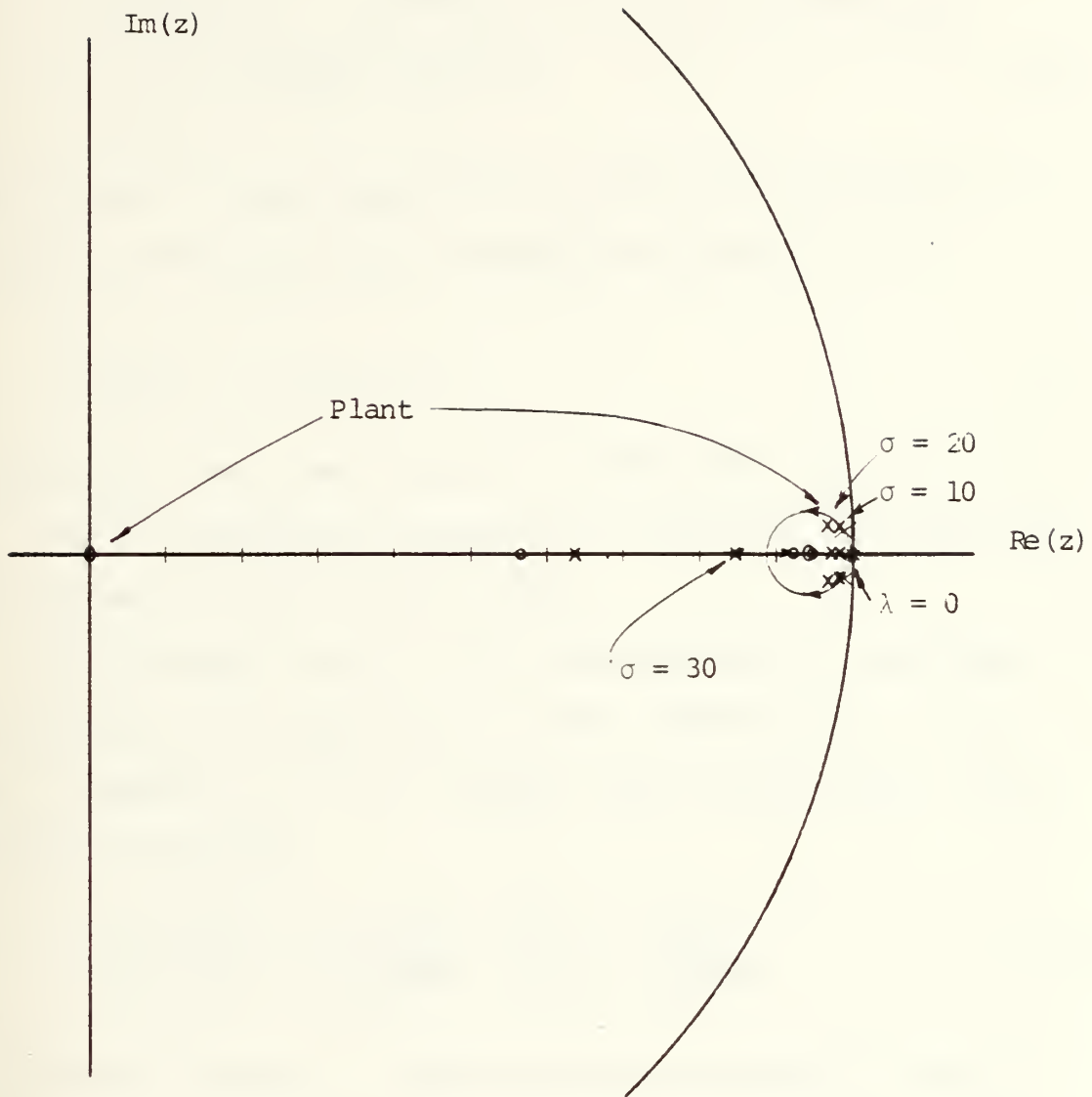


Figure 8.22. PLL Root Locus Analysis vs Computer Roots of the AKI ARMA Model

The departure of the pole-zero behavior from what was expected was analyzed by closer investigation of the linear terms of equation (7.8).

$$\begin{aligned}
 y(k)_{\text{lin}} = & .02\alpha u(k-1) - .01889\alpha u(k-2) + (2.0 - .02\alpha)y(k-1) \\
 & + (.01889\alpha - 1)y(k-2)
 \end{aligned} \tag{8.9}$$

Recalling that a third order Taylor series approximation of the sine is the functional expressed by equation (7.7),

$$\sin(x) \approx \alpha x + \beta x^3 \tag{7.7}$$

one notes that the linear region is described when $\alpha = 1$ and $\beta = 0$. It is therefore reasonable to study the variation of α with respect to input noise power.

The average value $\bar{\alpha}$ of α , was computed by equating the estimated AKI coefficients to the coefficients of like terms in equation (8.9) for several input noise power levels. The relation used was

$$\bar{\alpha} = \frac{1}{4} \left[\frac{2.0 - \hat{b}_1}{.02} + \frac{1.0 + \hat{b}_2}{.01889} + \frac{\hat{a}_1}{.02} - \frac{\hat{a}_2}{.01889} \right] \tag{8.10}$$

at the 350th iteration. The relationship between $\bar{\alpha}$ and the input noise power level is readily apparent from Figure 8.23. This result is plausible since if one considers the input-output relationship of the sine block of Figure 5.7 one obtains Figure 8.24. Superimposing the gaussian probability functions of the different input noise signals, it can be seen that for

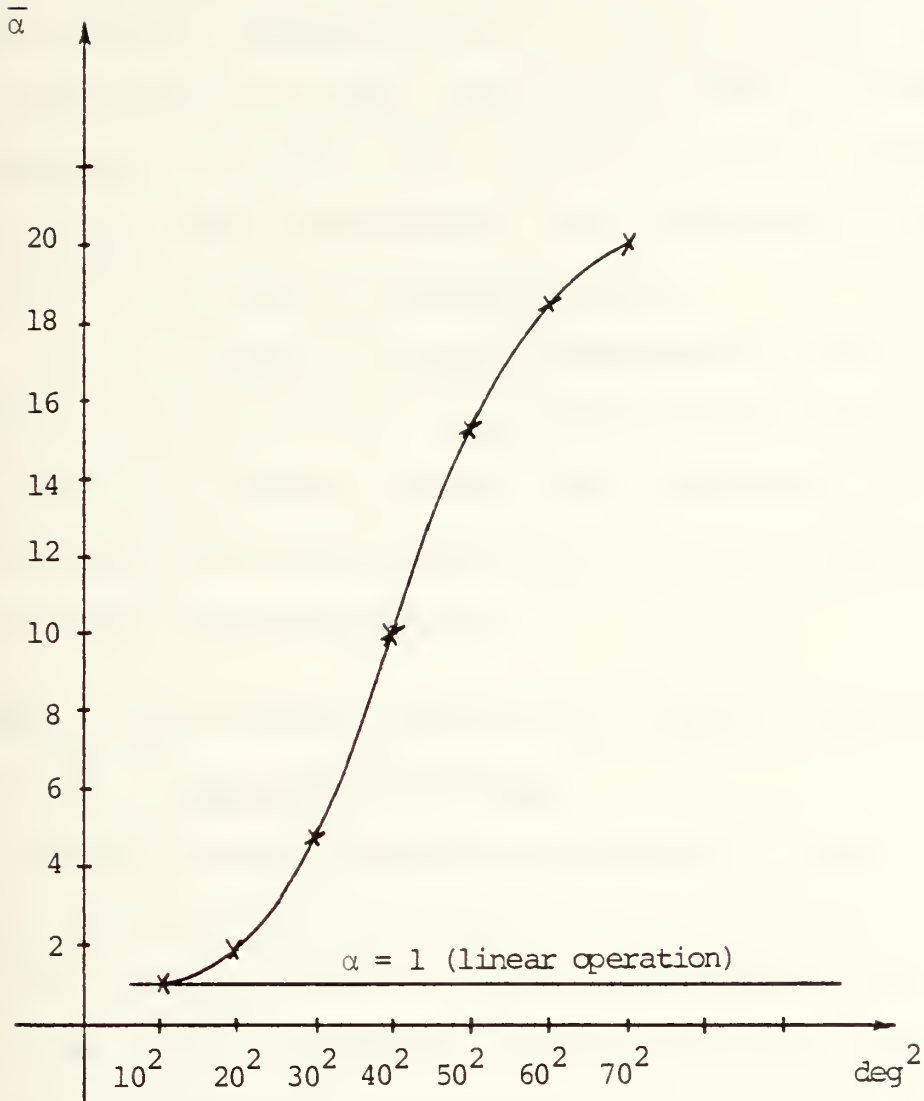


Figure 8.23. Variation of $\bar{\alpha}$ vs Input Noise Signal Power

low power levels of the input signal, identification of the linear parameters of the overall system (equation 5.9) can be made. As the input power is increased beyond $(25 \text{ deg})^2$ the input-output characteristics of the $\sin x$ block are no longer approximately linear, showing its effect in Figure 8.24 as a departure from its linear operation, $\bar{\alpha} = 1$. Therefore, by monitoring $\bar{\alpha}$ one can determine when the overall system is entering its non-linear operating regime.

Since the same functional dependence between each of the AKI estimates and $\bar{\alpha}$ did not exist for all of the input noise powers considered, knowing $\bar{\alpha}$ did not provide any information of the pole locations but did provide a measure of the degree of non-linear operation.

B. ORDER OF THE UNKNOWN SYSTEM IS NOT KNOWN (OVERMODELING)

1. AKI vs LMS Adaptive Filter

Using the data derived by operating a plant with the transfer characteristic of equation (8.1) the operation of the AKI and the LMS adaptive filter was compared when the orders used in the identifier and the LMS filter were greater than the known process that generated it. The following overmodeling cases were studied:

- (1) MA model is greater than plant MA process
- (2) ARMA model is fitted to MA plant

When the MA model order is greater than that of the plant MA process, it was found that both the AKI and the LMS filter would compute coefficients close to zero for the higher

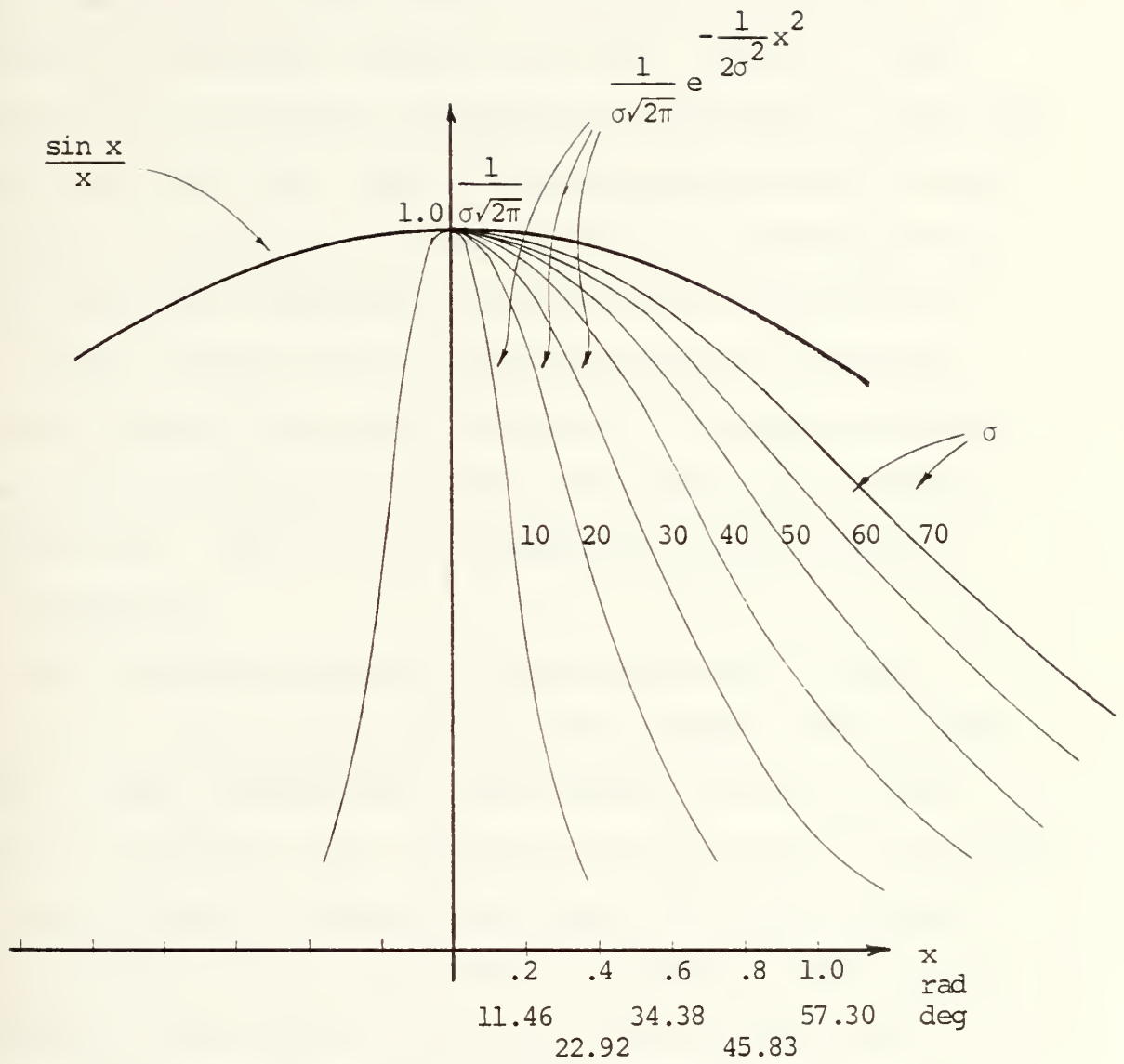


Figure 8.24. Comparison of $\frac{\sin x}{x}$ and p.d.f. of input noise signals

order coefficients. This effect can be seen in Figures 8.25a and 8.25b when a fourth order MA process is used to model the actual second order process represented by equation (8.1). Since these are a "zeros only" plant and model the overmodeling essentially causes zeros to appear at the origin of the Z-plane of the model transfer function. Table 8.5 summarizes the results for two overmodeling conditions at the 250th iteration using the AKI. When a purely moving average process represented by equation (8.1) was modeled as an ARMA process, one must direct his attention to the poles and zeros of the model transfer function which the AKI computed and compare them to the actual plant poles and zeros. It was not readily apparent from the resulting coefficients that the plant had been identified. The following example will help clarify what is happening.

For the data produced by the second order plant (equation 8.1), an autoregressive moving average (ARMA) model of orders 2 and 5 respectively was fitted. It can be seen from Table 8.6 that no firm conclusions can be drawn about which coefficients actually identify the plant. We, the analysts, knowing the form of the plant which produced the data, could qualitatively state that b_1 , b_2 , a_3 , a_4 and a_5 are small enough to be ignored. Hence, we can identify the plant correctly. However, inspection of the poles and zeros of the transfer function which the AKI computed,

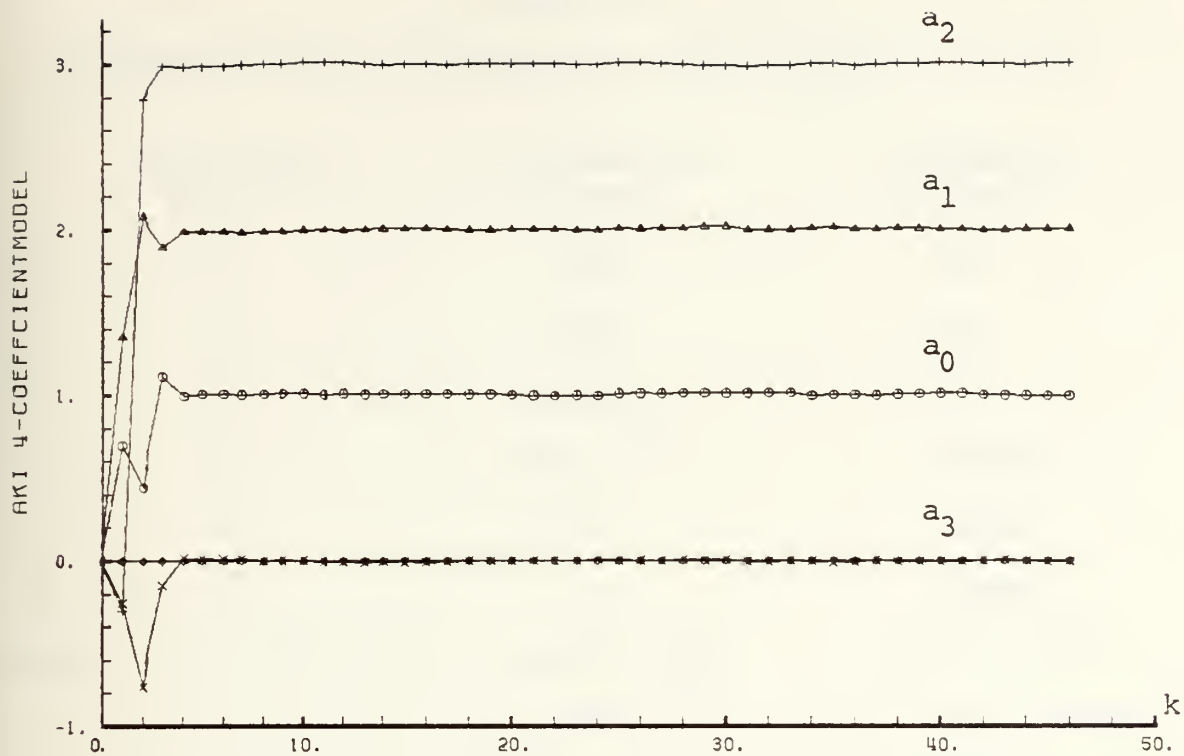


Figure 8.25a. AKI(0,4) Model of a Third Order MA Process

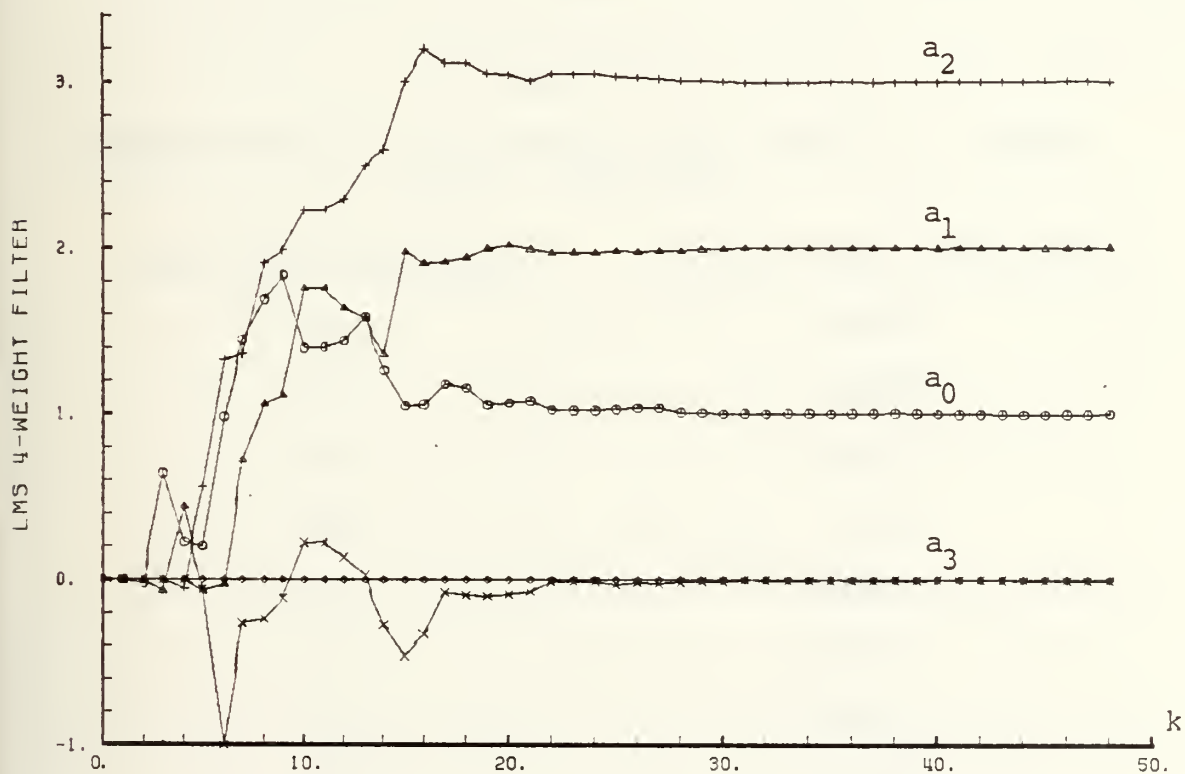


Figure 8.25b. Fourth Order LMS Filter Weights of a Third Order MA Process

Table 8.5

OVERMODELING OF A MA PROCESS USING THE AKI ($k = 250$)

	Actual Model Coeffs	3rd Order AKI MA Model	5th Order AKI MA Model
a_0	1.0	.9995	.9991
a_1	2.0	2.0000	2.000
a_2	3.0	3.0000	3.000
a_3	0.0	$.893 \times 10^{-3}$.0003191
a_4	0.0	---	.001082
a_5	0.0	---	.0003198
zeros:	$-1.0 \pm 1.414j$	$-1.000 \pm 1.414j$	$-1.001 \pm 1.414j$
		$-.2977 \times 10^{-3}$	$.0227 \pm .0428j$
			-.4534

Table 8.6

OVERMODELING OF A MA PROCESS USING THE AKI OF ORDERS
AR = 2, MA = 6

	Actual Plant Coefficients	AKI(2,6) Model Coefficients
b_1	0.0	.00467
b_2	0.0	.10500
a_0	1.0	1.0000
a_1	2.0	1.9940
a_2	3.0	2.8840
a_3	0.0	-0.2257
a_4	0.0	-0.3142
a_5	0.0	0.0001517

$$H_{AKI(2,6)}(z) = \frac{1.0 + 1.994z^{-1} + 2.8840z^{-2} - 0.2257z^{-3} - 0.3142z^{-4} + 0.001517z^{-5}}{1 - .00467z^{-1} - .1050z^{-2}} \quad (8.11)$$

results in:

poles: -.8217
 .3264
 zeros: -.9996 ± 1.414j
 .4827 × 10⁻³
 -.3212
 .3262

The observation to be made is that there are two pole-zero combinations which are near cancellation. This suggests that the AKI algorithm be rerun with the AR and MA orders reduced by at least two.

The implication of the analysis of this section is that a model can in theory be found for a given set of input/output data. Further, a parsimonious model can be identified by careful observation of pole-zero combinations which are near cancellation and of stray zeros near the origin.

2. AKI Applied to AR and ARMA Data

Essentially the same characteristic results found in Section VIII.B.1 were confirmed when the data produced by plants defined by the transfer functions of equations (8.3) and (8.4) were analyzed by the overmodeled AKI. That is, pole-zero pairs near cancellation and zeros near the origin

were produced by the AKI. Figures 8.26 through 8.28 have been chosen as representative pole-zero plots of the transfer functions computed by the AKI for the AR plant of Section VIII.A.2 and for Perry's model (Section VIII.A.3).

C. CONCLUSION

This work indicates that the Kalman filter algorithm heretofore generally used as a state estimator, or in augmented form to estimate parameters (in which case the parameters are treated as states), can also be formulated in an adaptive manner to iteratively estimate the coefficients of an ARMA equation explicitly. This approach, termed the Adaptive Kalman Identifier (AKI), summarily identifies the unknown system whose input/output data is being processed. The LMS adaptive algorithm of Widrow, and its modification by Griffiths (in which the convergence factor is selected to be inversely proportional to the input signal power) are shown to be sub-optimal cases of the AKI. An additional insight provided by the AKI is that it indicates clearly how measurement noise might be taken into account in the LMS adaptive formulation.

The operation of the AKI was checked by way of simulation and compared with two existing identification techniques: (1) the LMS Adaptive nonrecursive (MA) algorithm and (2) the Adaptive Recursive ARMA LMS algorithm. It was found that not only are the two LMS filtering techniques special suboptimal cases of the AKI; but, further, the AKI exhibits superior convergence and modeling properties for the cases where (1) the

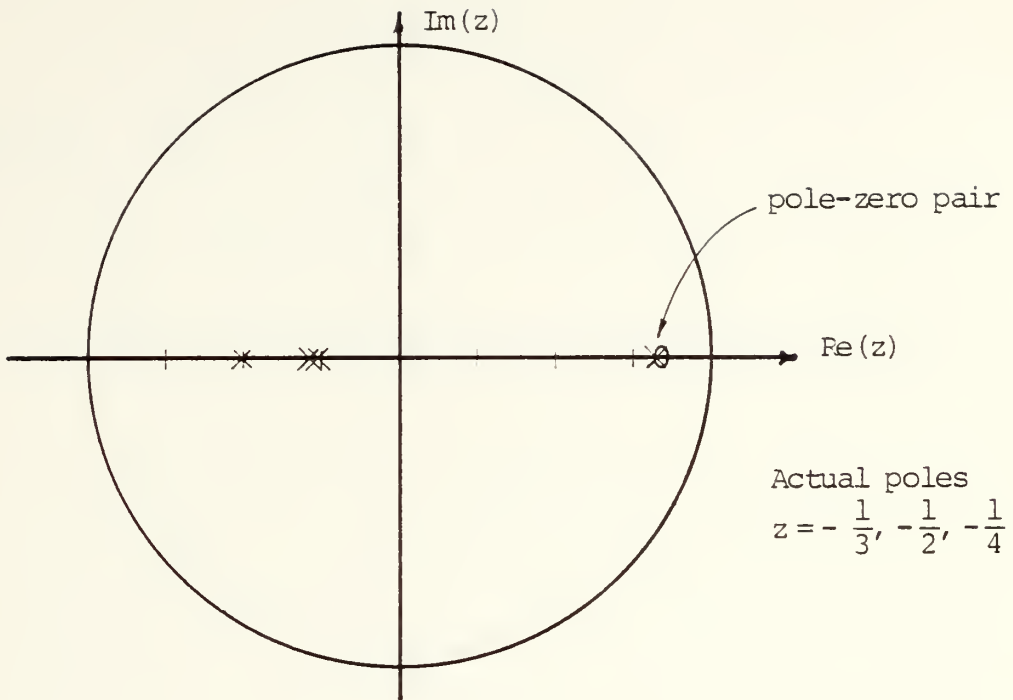


Figure 8.26. AKI(3,1) Overmodeled AKI(4,2)

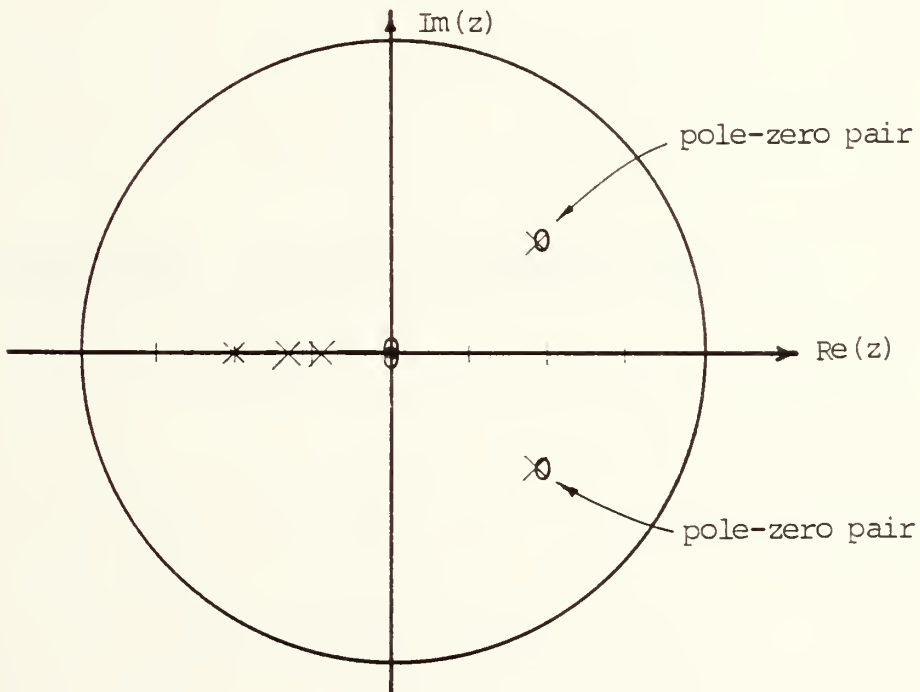


Figure 8.27. AKI(3,1) Overmodeled AKI(5,5)

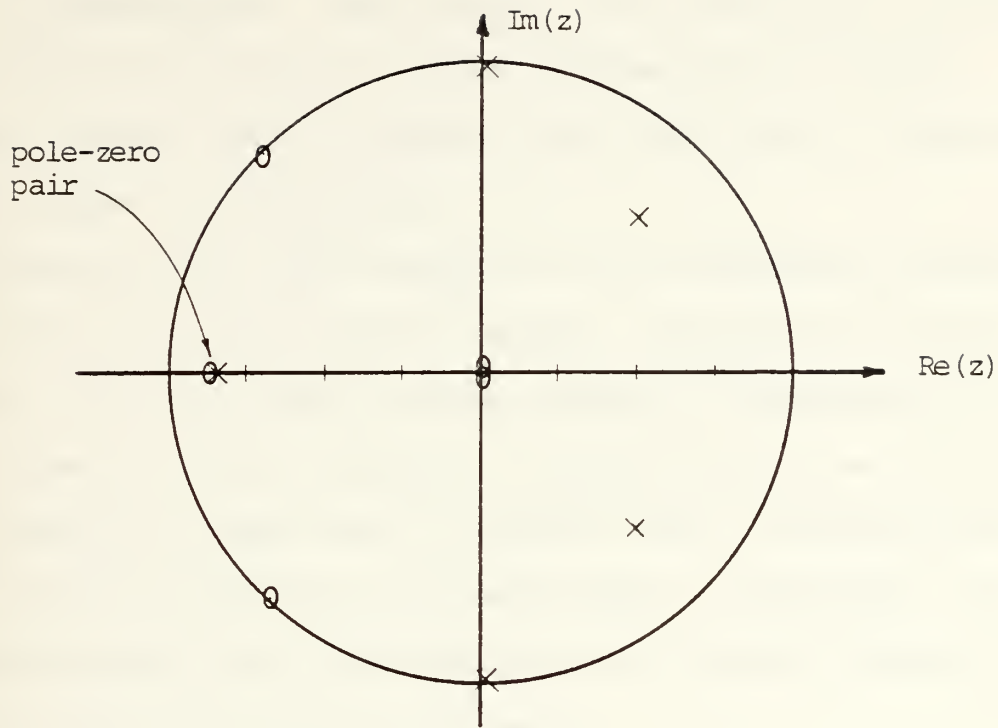


Figure 8.28. AKI(4,5) Overmodeled AKI(5,6)

order of the unknown plant is known and (2) the order of the plant is overmodeled. Additionally, the simulations indicate that accuracies similar to those obtained using lattice modeling methods, can be achieved using the AKI at a decidedly smaller number of iterations.

By making minor modifications to the measurement vector, $H(k)$, that is by using hybrid signals, the AKI was used to identify the linear and non-linear ARMA representations of a phase locked loop with success. Interestingly, the AKI technique appears to enable one to discern when a potential non-linear system enters its non-linear mode of operation, by closely monitoring the coefficients of the linear portion of the generalized non-linear ARMA model.

D. TOPICS FOR FURTHER CONSIDERATION

Several areas for further study directly and indirectly related to the AKI were uncovered. Foremost, a rigorous convergence proof is desirable. Although the connection was made between the AKI algorithm and the initial equation from which the lattice modeling algorithm is developed, similar comparisons (such as Chapters III and IV) could provide more insight into the operation of both. The multichannel AKI is the logical development of the single channel AKI presented. And, lastly, refinement of the AKI software (using the NL = 5 option) to include ARMA modeling of time series using a Box-Jenkins approach [Ref. 26] is feasible. A limited number of simulations using Monterey rain data and series C [Ref. 26]

data seem to indicate that an application exists for the AKI
in this area.

APPENDIX A

THE DISCRETE WEINER PROBLEM

The discrete Wiener problem considers optimally filtering a desired signal from unwanted stationary noise. The criterion of optimality used is minimization of the mean squared error between the output and the desired signal. Generally one desires that the optimal filter, which is the device being sought, be time invariant and that it be able to accept a signal, $s(k)$, and noise, $n(k)$, where each are samples from stationary random processes. In other words, we want a device which can accept

$$x(k) = s(k) + n(k) \quad (\text{A.1})$$

as an input and produce at its output $s(k+\Delta)$ or some linear function thereof where Δ is some known delay.

Assuming that the desired output, $d(k)$, is the response to a specified sampled data linear system whose transfer function, $H_d(z)$, is given, then the error between the desired output and the output of the filter, $y(k)$, we seek is,

$$e(k) = d(k) - y(k) . \quad (\text{A.2})$$

Figure A.1 depicts the discrete Wiener problem formulation. The derivation from here follows the one presented by Maybeck [Ref. 27] for the continuous case. From the block diagram we have,

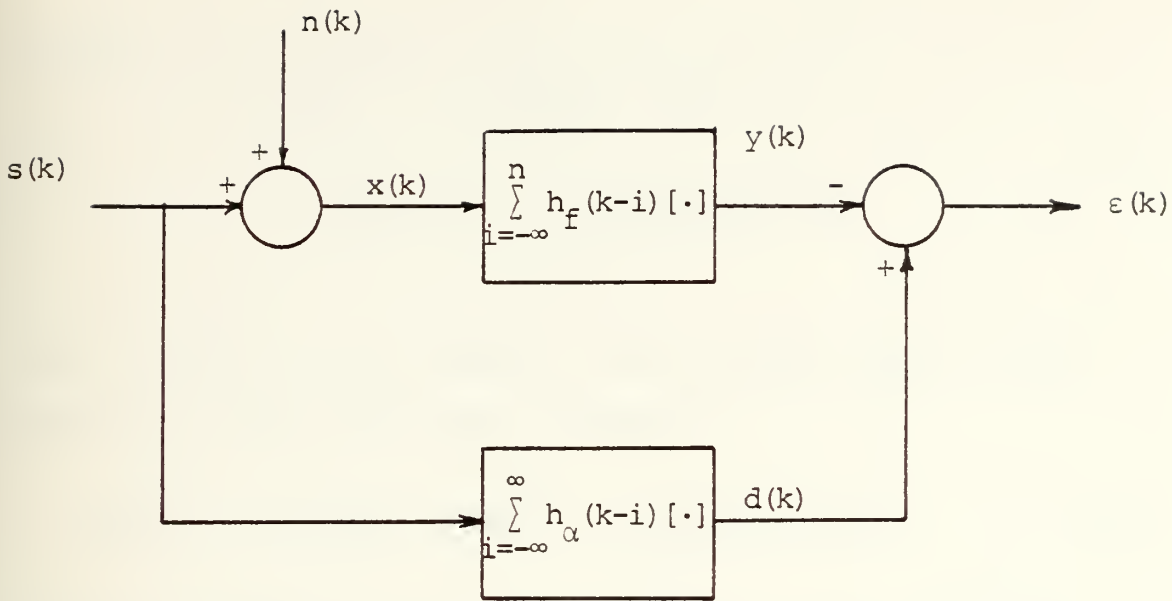


Figure A.1. The Discrete Wiener Problem

$$y(k) = \sum_{i=-\infty}^{\infty} h_f(k-i) x(i) \quad (\text{A.3})$$

or equivalently,

$$y(k) = \sum_{i=-\infty}^{\infty} x(k-i) h_f(i) \quad (\text{A.4})$$

Substituting equation A.4 into A.2 and squaring we have,

$$e^2(k) = d^2(k) - 2d(k) \sum_{i=-\infty}^{\infty} h_f(i) x(k-i) + \left[\sum_{i=-\infty}^{\infty} x(i) h_f(k-i) \right] \left[\sum_{i=-\infty}^{\infty} x(i) h_f(k-i) \right] \quad (\text{A.5})$$

Taking the expected value of A.5 we obtain,

$$\begin{aligned}
 E\{e^2(k)\} &= \psi_{dd}(0) - 2 \sum_{i=-\infty}^{\infty} h_f(i) \psi_{dx}(i) \\
 &+ \sum_{i=-\infty}^{\infty} h_f(i) \sum_{k=-\infty}^{\infty} h_f(k) \psi_{xx}(k-i)
 \end{aligned}
 \tag{A.6}$$

where notation $\psi_{uv}(m)$ denotes the expected value of the product of $u(k)$ and $v(k+m)$. That is,

$$\psi_{uv}(m) = E\{u(k)v(k+m)\} = E\{u(k)v(k-m)\}
 \tag{A.7}$$

otherwise known as the autocorrelation of $u(k)$ and $v(k)$.

Using variational techniques and letting,

$$h_f(i) = h_{opt}(i) + \epsilon \Delta h(i)
 \tag{A.8}$$

substituting equation A.8 into equation A.6, taking the partial derivative with respect to epsilon, ϵ , and setting the partial derivative equal to zero we have:

$$\sum_{n=-\infty}^{\infty} \Delta h(n) \left[\sum_{i=-\infty}^{\infty} h_{opt}(i) \psi_{xx}(n-i) - \psi_{xd}(n) \right] = 0
 \tag{A.9}$$

The term within the brackets is the discrete form of the Weiner-Hopf equation and must be equal to zero if equation A.9 is to be valid since $\Delta h(n) > 0$ by definition. Therefore,

$$\sum_{i=-\infty}^{\infty} h_{opt}(i) \psi_{xx}(n-i) = \psi_{xd}(n)
 \tag{A.10}$$

Equation A.10 is the most often encountered in truncated form in linear prediction theory, Pade approximation, adaptive filtering and lattice filtering. The truncated version of A.10 which is generally used in solving the discrete Weiner problem is,

$$\sum_{i=0}^M h_{\text{opt}}(i) \psi_{\text{xx}}(n-i) = \psi_{\text{xd}}(n) \quad (\text{A.11})$$

or in matrix form,

$$\begin{bmatrix} \psi_{\text{xx}}(0) & \psi_{\text{xx}}(-1) & \dots & \psi_{\text{xx}}(-n) \\ \psi_{\text{xx}}(1) & \psi_{\text{xx}}(0) & \dots & \psi_{\text{xx}}(1-n) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{\text{xx}}(n) & \psi_{\text{xx}}(n-1) & \dots & \psi_{\text{xx}}(0) \end{bmatrix} \begin{bmatrix} h_{\text{opt}}(0) \\ h_{\text{opt}}(1) \\ \vdots \\ h_{\text{opt}}(n) \end{bmatrix} = \begin{bmatrix} \psi_{\text{xd}}(0) \\ \psi_{\text{xd}}(1) \\ \vdots \\ \psi_{\text{xd}}(n) \end{bmatrix} \quad (\text{A.12})$$

APPENDIX B

```

PROGRAM ADMA: ADJCKECLSIVL-MOVING AVERAGE SIMULATOR
      THIS PROGRAM GENERATES THE VARIOUS TYPES OF LINEAR SYNTHETIC
      DATA WHICH IS PROCESSED BY THE ADAPTIVE KALMAN IDENTIFIER (AKI).
      THE PROGRAM PERFORMS ALL COMPUTATIONS IN DOUBLE PRECISION AND
      STORES THE OUTPUT DATA ON A DISK FILE FOR ANALYSIS BY THE AKI. THE
      ENDERS OF THE ADJCKECLSIVL (AKI) AND MOVING-AVERAGE (MA) PHASES
      PRECEDE THE DATA IN UNCOMPRESSED FORM.
      REAL*8 Y(8000),U(8000),Z(8000),V(8000)
      REAL*4 W(8000),W2(8000)
      WRITE (6,100)
      100  FORMAT (1H1,IX,4HTIME,10X,5HINPUT,9X,6HOUTPUT,4X,11HMEASUREMENT,/)
      CALL U-92545117.00
      SLOW = 1.0D-3
      SLOW = 1.0D-3
      NP= NUMBER OF DATA POINTS GENERATED
      NF= NUMBER OF AN COEFFICIENTS
      NW= NUMBER OF MA COEFFICIENTS
      NI= MODE OF OPERATION FOR THE AKI
      NICE= ICIAL ORDER OF SYSTEM
      NP=400
      NF=2
      NW=0
      NI=0
      NICE=NP*NF
      WRITE (6,101)NP,NL
      101  CALL UCNPL (USEF0,NF,N)
      CALL UCNPL (USEF1,NF,N2)
      DO 10 I=1,NP
      V(I) = SLOW*DBLE(W2(I))
      U(I) = LELE(W(I))
      10
      A1=0.000
      A2=0.000
      A3=0.000
      A4=0.000
      A5=1.000
      A6=2.000
      A7=3.000
      A8=0.000
  
```


APPENDIX C

```

000000 PROGRAM PLL: PHASE LOCKED LOOP (PLL)
000001
000002 THIS PROGRAM RUNS THE NONLINEAR PLL MODEL THAT PRODUCES THE
000003 DATA TABLE ANALYZED BY THE ADAPTIVE KALMAN IDENTIFIER (AKI).
000004 THE SIMULATED PLL EMPLOYS A FORWARD FULLY INTEGRIATED SCHEME
000005 WITH THE WELL KNOWN NONLINEARITY SIN(.). THE LINEAR
000006 PORTION OF THE MODEL IS MATHEMATICALLY REPRESENTED BY
000007 THE EQUATION:
000008
000009 
$$P_{02}(Z^{**}-1) * \frac{1 + A1*Z^{**}-1 + A2*Z^{**}-2}{1 - B1*Z^{**}-1 - B2*Z^{**}-2} \dots$$

000010
000011 K=4, P=12, I, C(12), I), X(1200), Y(1200), Z(2000),
000012 TIME, INFLU(C)
000013 N=PI*6.1(2000), Y(12,2000), W(2000), W(2000), V(2000)
000014 CLAT=4 X1, X2, X3
000015 LABEL TG, FORD, NP
000016
000017 NPE=00
000018 W=4
000019 P=5
000020 ND=0
000021
000022 * SET THE ORDER OF SIGMA (N,N) TO BE IDENTIFIED AND THE AKI
000023 (P, I, C, M, U, I, I, I, C) FILE (8)
000024 * SET (2) N, M, N1
000025
000026 I=7
000027 CUMI=2
000028 ID=1
000029 PI=4.0, KALAN(1.0)
000030
000031 DEFINE THE PSYCHO PLANT GAINS
000032 G(1,1)=0.0200
000033
000034 DEFINE THE COEFFICIENTS OF THE LINEAR PORTION OF THE SYSTEM
000035 C(1,1)=2.0
000036 C(2,1)=-1.0
000037 C(3,1)=1.0
000038 C(4,1)=-0.999999
000039
000040 * DEFINE THE INPUT FORCING FUNCTION(S), AND MEASUREMENT ERROR SET
000041 W(1)=0.007(12,00)
000042 GAIN G(1) (0.01, 1200, W)
000043 GAIN SCHEM (0.01, 1200, V)
000044 sigma = 0.001
000045 sigma = 1.00E-4
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099
000100
000101
000102
000103
000104
000105
000106
000107
000108
000109
000110
000111
000112
000113
000114
000115
000116
000117
000118
000119
000120
000121
000122
000123
000124
000125
000126
000127
000128
000129
000130
000131
000132
000133
000134
000135
000136
000137
000138
000139
000140
000141
000142
000143
000144
000145
000146
000147
000148
000149
000150
000151
000152
000153
000154
000155
000156
000157
000158
000159
000160
000161
000162
000163
000164
000165
000166
000167
000168
000169
000170
000171
000172
000173
000174
000175
000176
000177
000178
000179
000180
000181
000182
000183
000184
000185
000186
000187
000188
000189
000190
000191
000192
000193
000194
000195
000196
000197
000198
000199
000200
000201
000202
000203
000204
000205
000206
000207
000208
000209
000210
000211
000212
000213
000214
000215
000216
000217
000218
000219
000220
000221
000222
000223
000224
000225
000226
000227
000228
000229
000230
000231
000232
000233
000234
000235
000236
000237
000238
000239
000240
000241
000242
000243
000244
000245
000246
000247
000248
000249
000250
000251
000252
000253
000254
000255
000256
000257
000258
000259
000260
000261
000262
000263
000264
000265
000266
000267
000268
000269
000270
000271
000272
000273
000274
000275
000276
000277
000278
000279
000280
000281
000282
000283
000284
000285
000286
000287
000288
000289
000290
000291
000292
000293
000294
000295
000296
000297
000298
000299
000300
000301
000302
000303
000304
000305
000306
000307
000308
000309
000310
000311
000312
000313
000314
000315
000316
000317
000318
000319
000320
000321
000322
000323
000324
000325
000326
000327
000328
000329
000330
000331
000332
000333
000334
000335
000336
000337
000338
000339
000340
000341
000342
000343
000344
000345
000346
000347
000348
000349
000350
000351
000352
000353
000354
000355
000356
000357
000358
000359
000360
000361
000362
000363
000364
000365
000366
000367
000368
000369
000370
000371
000372
000373
000374
000375
000376
000377
000378
000379
000380
000381
000382
000383
000384
000385
000386
000387
000388
000389
000390
000391
000392
000393
000394
000395
000396
000397
000398
000399
000400
000401
000402
000403
000404
000405
000406
000407
000408
000409
000410
000411
000412
000413
000414
000415
000416
000417
000418
000419
000420
000421
000422
000423
000424
000425
000426
000427
000428
000429
000430
000431
000432
000433
000434
000435
000436
000437
000438
000439
000440
000441
000442
000443
000444
000445
000446
000447
000448
000449
000450
000451
000452
000453
000454
000455
000456
000457
000458
000459
000460
000461
000462
000463
000464
000465
000466
000467
000468
000469
000470
000471
000472
000473
000474
000475
000476
000477
000478
000479
000480
000481
000482
000483
000484
000485
000486
000487
000488
000489
000490
000491
000492
000493
000494
000495
000496
000497
000498
000499
000500
000501
000502
000503
000504
000505
000506
000507
000508
000509
000510
000511
000512
000513
000514
000515
000516
000517
000518
000519
000520
000521
000522
000523
000524
000525
000526
000527
000528
000529
000530
000531
000532
000533
000534
000535
000536
000537
000538
000539
000540
000541
000542
000543
000544
000545
000546
000547
000548
000549
000550
000551
000552
000553
000554
000555
000556
000557
000558
000559
000560
000561
000562
000563
000564
000565
000566
000567
000568
000569
000570
000571
000572
000573
000574
000575
000576
000577
000578
000579
000580
000581
000582
000583
000584
000585
000586
000587
000588
000589
000590
000591
000592
000593
000594
000595
000596
000597
000598
000599
000600
000601
000602
000603
000604
000605
000606
000607
000608
000609
000610
000611
000612
000613
000614
000615
000616
000617
000618
000619
000620
000621
000622
000623
000624
000625
000626
000627
000628
000629
000630
000631
000632
000633
000634
000635
000636
000637
000638
000639
000640
000641
000642
000643
000644
000645
000646
000647
000648
000649
000650
000651
000652
000653
000654
000655
000656
000657
000658
000659
000660
000661
000662
000663
000664
000665
000666
000667
000668
000669
000670
000671
000672
000673
000674
000675
000676
000677
000678
000679
000680
000681
000682
000683
000684
000685
000686
000687
000688
000689
000690
000691
000692
000693
000694
000695
000696
000697
000698
000699
000700
000701
000702
000703
000704
000705
000706
000707
000708
000709
000710
000711
000712
000713
000714
000715
000716
000717
000718
000719
000720
000721
000722
000723
000724
000725
000726
000727
000728
000729
000730
000731
000732
000733
000734
000735
000736
000737
000738
000739
000740
000741
000742
000743
000744
000745
000746
000747
000748
000749
000750
000751
000752
000753
000754
000755
000756
000757
000758
000759
000760
000761
000762
000763
000764
000765
000766
000767
000768
000769
000770
000771
000772
000773
000774
000775
000776
000777
000778
000779
000780
000781
000782
000783
000784
000785
000786
000787
000788
000789
000790
000791
000792
000793
000794
000795
000796
000797
000798
000799
000800
000801
000802
000803
000804
000805
000806
000807
000808
000809
000810
000811
000812
000813
000814
000815
000816
000817
000818
000819
000820
000821
000822
000823
000824
000825
000826
000827
000828
000829
000830
000831
000832
000833
000834
000835
000836
000837
000838
000839
000840
000841
000842
000843
000844
000845
000846
000847
000848
000849
000850
000851
000852
000853
000854
000855
000856
000857
000858
000859
000860
000861
000862
000863
000864
000865
000866
000867
000868
000869
000870
000871
000872
000873
000874
000875
000876
000877
000878
000879
000880
000881
000882
000883
000884
000885
000886
000887
000888
000889
000890
000891
000892
000893
000894
000895
000896
000897
000898
000899
000900
000901
000902
000903
000904
000905
000906
000907
000908
000909
000910
000911
000912
000913
000914
000915
000916
000917
000918
000919
000920
000921
000922
000923
000924
000925
000926
000927
000928
000929
000930
000931
000932
000933
000934
000935
000936
000937
000938
000939
000940
000941
000942
000943
000944
000945
000946
000947
000948
000949
000950
000951
000952
000953
000954
000955
000956
000957
000958
000959
000960
000961
000962
000963
000964
000965
000966
000967
000968
000969
000970
000971
000972
000973
000974
000975
000976
000977
000978
000979
000980
000981
000982
000983
000984
000985
000986
000987
000988
000989
000990
000991
000992
000993
000994
000995
000996
000997
000998
000999
001000

```



```

10 DEFINE THE # OF PARAMETER STUDIES
11 JKOR=1
12
13 CONVERT THE INPUTS TO PATIENS AND SET THE FOLWK LEVEL
14 DO 500 J=1,JP
15   W(J) = ((PI*W(1))/180.0)*SICK
16
17 BEGIN MAIN ITERATION LOOP
18 DO 1000 I=1,IKON
19
20   DEFINE INPUT X, AS A GAUSSIAN INPUT SEQUENCE
21   DO 100 J=1,JP
22     X(J)=DELE(W(J))
23
24   RUN THE SIMULATED PHASE LOCKED LOOP
25   CALL WBLGN(PG,C,IG,MCRD,X,Y,NP,10)
26
27   PRINT OUT THE RESULTS OF THE SIMULATION
28   WRITE(CPZC)
29   PRINT(ATTG,11X,'TIME.....INPUT.....OUTPUT.....',/)
30
31   WRITE CHEGICAL DATA TO FILE 3 FOR FURTHER ANALYSIS
32   DO 175 I=1,JP
33     X1 = X(I)
34     X2 = SINC(Y(I))+5UCV4V(I)
35     Z(I) = GALE(X2)
36     WRITE(IG) X1,X2
37   CONTINUE
38
39   PRINT OUT THE GATHERED DATA
40   DO 200 S=1,JP
41     IJF=0.01501(OAI(J)-1)
42     WRITE(CZS) 11H,KA(J),Y(J),Z(J)
43   CONTINUE
44   JCC
45   END THE MAIN ITERATION LOOP
46
47   STOP
48   END

```

```

FLL00520
FLL00530
FLL00540
FLL00550
FLL00560
FLL00570
FLL00580
FLL00590
FLL00600
FLL00610
FLL00620
FLL00630
FLL00640
FLL00650
FLL00660
FLL00670
FLL00680
FLL00690
FLL00700
FLL00710
FLL00720
FLL00730
FLL00740
FLL00750
FLL00760
FLL00770
FLL00780
FLL00790
FLL00800
FLL00810
FLL00820
FLL00830
FLL00840
FLL00850
FLL00860
FLL00870
FLL00880
FLL00890
FLL00900
FLL00910
FLL00920
FLL00930
FLL00940
FLL00950

```



```

5  APPLY SCALE FACTOR TO ARRAY
6  DO 75 I=2,IY
7  DO 75 I=1,MYRD
8  ARRAY(I,L)=ARRAY(I,L)*PG(I-1,L)
9
10 WRITE(6,15)
11 FORNAT(7,' THE SCALED COEFFICIENTS ARE:')
12 WRITE(6,16) ((ARRAY(I,L),I=1,IC),L=1,MLRP)
13
14 LA=0
15 DO 65 I=1,100
16 DO 65 I=1,MYRD
17 LA=LA+1
18 CSCL(LA,I)=ARRAY(I,L)
19
20 INITIALIZE FILTER STATES TO ZERO
21 DO 100 I=1,100
22 DO 100 L=1,MYRD
23 FF(I,L)=L.CDC
24
25 GEARABLE NR POINTS OF OUTPUT Y(INS)
26 DO 1000 IAS=1,100
27
28 DEFINE INPUT VECTOR U AT INSTANCES
29 FF(1,I)=Y(INS)
30 FF(2,I)=X(INS)**2
31 FF(3,I)=X(INS)**2
32 FF(4,I)=X(INS)*Y(INS)
33 FF(5,I)=X(INS)**3
34 FF(6,I)=X(INS)**3
35 FF(7,I)=X(INS)*X(INS)**2
36 FF(8,I)=X(INS)*X(INS)**2
37 FF(9,I)=X(INS)**3
38
39 FF(10,IC)=L.CDC
40 WRITE(6,17)
41 FORNAT(7,10),FF(I,J),I=1,10)
42 FORNAT(7,10),L=1,MYRD)
43
44 DEFINE
45 Y=0.0
46
47 FORNAT(7,10),Y(INS),I=1,10)
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```


APPENDIX D

```

PROGRAM ALAP13N: ADAPTIVE KALMAN IDENTIFIER (AKI)
      THIS PROGRAM REPRESENTS THE MAIN SOFTWARE DEVELOPED WHICH IM-
      PLEMENTS THE ADAPTIVE KALMAN IDENTIFIER (AKI) FORMULATED IN THE
      BODY OF THE THESIS. THE ADAPTIVE KALMAN IDENTIFIER (AKI) IDENTI-
      FIES THE COEFFICIENTS OF THE GENERAL AUTOREGRESSIVE-AVERAGE
      (GAMA) REPRESENTATION OF AN UNKNOWN SYSTEM. HERETOFORE, UNKNOWN
      SYSTEM PARAMETERS WERE SIMPLY AUGMENTED TO THE STATE VECTOR OF THE
      KALMAN FILTER AND, HENCE, ESTIMATED. USING THE AKI PRO-
      CEDURE, SUCH AUGMENTATION OF EXTENSION IS MADE; BUT, RATHER,
      THE PARAMETERS OF THE SYSTEM (FOR EQUIVALENTLY THE COEFFICIENTS OF
      THE ARMA EQUATION) ARE IDENTIFIED DIRECTLY.
      IDENTIFYING THE ARMA COEFFICIENTS REQUIRES PROCESSING THE
      INPUT/OUTPUT DATA WHICH IS TEMPORARILY STORED ON A DISK FILE PRIOR
      TO BEING READ BY THE AKI PROGRAM. THE LEVELS OF THE AUTOREGRESSIVE
      (AR) AND MOVING-AVERAGE (MA) PROCESS AND THE OPTION FLAG PRECEDE
      THE UNFORMATTED DATA ALSO IN UNFORMATTED FORM. THE AKI PROGRAM
      HAS SEVERAL MULTIPURPOSE FLAGS WHICH ARE DESCRIBED THROUGHOUT THE
      SOURCE CODE IN THE FORM OF COMMENT STATEMENTS.
      REAL *8 A(NK,2),H,PKK,PARM1,U,V,W,FI,Z,R,LU,YY,CY,YU,UG
      COMMON A(NK(20,300)),C(20,300),I(20,20),PKK(20,20),PARM1(20,20)
      I,UG(800),YY(800),CY(800),YL(800),GG(C2C,800)
      Z,J(800),Y(800),C(20,20),EI(20,20),Z,R,N,K,M,NTOT,KK,IDIAG,NI
      3,1,1,1,1
      REAL *8 SLOW
      REAL*4 V1(300),V2(600),A(20),B(20)
      COMMON A1(20),F2(20)
      INTEGER N,M,RP,NTOT,IDIAG,IPL,IFCOT,I,R,IPRT,NI
      N=100
      M=10
      RP=300
      IFCOT=MP-100
      IPL = A PLOTTING FLAG; IF IPL=1 SUBROUTINE GRAPH IS CALLED
      IDIAG = A DIAGNOSTIC FLAG; IF IDIAG=1 MATRIX INTERMEDIATE
      VALUES ARE PRINTED OUT; IFCOT = A COEF CALCULATION FLAG;
      IF IFCOT=1 INTERMEDIATE VALUES FOR THE RECS ARE COMPUTED.
      IPRT = A PRINTING FLAG; PRINTFC OUTPUT RESULTS FOR K>IPRT.
      IPL=1
      IDIAG=0
      IFCOT=1
  
```



```

20 READ IN THE INPUT/OUTPUT DATA OF THE UNKNOWN PLANT TO
21 BE IDENTIFIED
22 IF NA THEN CALCULATE AIA BCX-JENKINS, READ IN Y(K) AND
23 ZERO OUT ALL U(K)
24 IF NP>Z*NTOT
25 IF (M*.5) GO TO 21
26 DO 22 I=1,NP
27 READ (8) VZ(I)
28
29 CONVERT INPUT DATA TO DOUBLE PRECISION
30 DO 23 I=1,NP
31 U(I) = C.CDU
32 VZ(I) = EBLE(VZ(I))
33 GO TO 24
34
35 CONTINUE
36
37 DO 20 I=1,NP
38 READ (8) V1(I),V2(I)
39
40 CONVERT THE INPUT/OUTPUT DATA INTO DOUBLE PRECISION
41 DO 40 I=1,NP
42 U(I) = EBLE(V1(I)+NTOT(I))
43 Y(I) = EBLE(V2(I)+NTOT(I))
44 GO TO 24
45
46 CONTINUE
47
48 FURTHER REQUIRED NON-LINEAR INPUTS FOR THE NEURAL
49 IF (N1.EQ.0) GO TO 25
50 DO 25 I=1,NP
51 U(I) = U(I)**3
52 IF (N1.EQ.1) GO TO 25
53
54 DO 26 I=1,NP
55 Y(I) = Y(I)**5
56 IF (N1.EQ.2) GO TO 25
57
58 DO 27 I=1,NP
59 U(I) = (U(I)**2)*Y(I)
60 IF (N1.EQ.3) GO TO 25
61
62 DO 28 I=1,NP
63 Y(I) = (Y(I)**2)*U(I)
64
65 CONTINUE

```

```

ALAC1000
ALAC1010
ALAC1020
ALAC1030
ALAC1040
ALAC1050
ALAC1060
ALAC1070
ALAC1080
ALAC1090
ALAC1100
ALAC1110
ALAC1120
ALAC1130
ALAC1140
ALAC1150
ALAC1160
ALAC1170
ALAC1180
ALAC1190
ALAC1200
ALAC1210
ALAC1220
ALAC1230
ALAC1240
ALAC1250
ALAC1260
ALAC1270
ALAC1280
ALAC1290
ALAC1300
ALAC1310
ALAC1320
ALAC1330
ALAC1340
ALAC1350
ALAC1360
ALAC1370
ALAC1380
ALAC1390
ALAC1400
ALAC1410
ALAC1420
ALAC1430
ALAC1440
ALAC1450
ALAC1460
ALAC1470

```



```

10 LOAD THE INITIAL VALUES OF THE MEASUREMENT VECTOR
11 IF (N.EQ.0) GO TO 40
12 LN = LN+1
30 LOAD THE MEASUREMENT VECTOR WITH Y(K-1) ELEMENTS
31 DO 30 I=1,N
32 L = LN+I
33 H(I,L) = Y(L+NTU(I)-N-1)
40 CONTINUE
41 IF (N.EQ.0) GO TO 60
42 M = M+1
50 LOAD THE MEASUREMENT VECTOR WITH U(K-1) ELEMENTS
51 DO 50 I=1,M
52 LL = MM+I
53 H(I,LL) = U(L+NU(I)-M)
54 IF ((N.EQ.0).OR.(M.EQ.5)) GO TO 60
60 LOAD THE MEASUREMENT VECTOR WITH NON-LINEAR ELEMENTS U(K-1)**3
61 IF (M.EQ.0) GO TO 53
62 M = M+1
63 DO 60 I=1,M
64 LI = LM+I
65 H(I,LI+NM) = UU(L+NU(I)-M)
66 IF ((N.EQ.0).OR.(M.EQ.2)) GO TO 60
70 LOAD THE MEASUREMENT VECTOR WITH NON-LINEAR ELEMENTS Y(K-1)**3
71 IF (N.EQ.0) GO TO 53
72 LN = LN+1
73 LP = LP+1
74 H(LP,LP+2*N) = YY(L+NU(L)-N-1)
75 IF ((N.EQ.0).OR.(LP.EQ.2)) GO TO 60
80 LOAD THE MEASUREMENT VECTOR WITH NON-LINEAR ELEMENTS Y(K)*U(K)**2
81 IF ((N.EQ.0).OR.(LP.EQ.0)) GO TO 60
82 LN = LN+1
83 LP = LP+1
84 H(LP,LP+2*N) = YY(L+NU(L)-N-1)
85 IF ((N.EQ.0).OR.(LP.EQ.3)) GO TO 60
90 LOAD THE MEASUREMENT VECTOR WITH NON-LINEAR ELEMENTS U(K)*Y(K)**2
91 LN = LN+1
92 DO 90 I=1,N

```

```

AFACI1490
AFACI1490
AFACI1500
AFACI1510
AFACI1520
AFACI1530
AFACI1540
AFACI1550
AFACI1560
AFACI1570
AFACI1580
AFACI1590
AFACI1600
AFACI1610
AFACI1620
AFACI1630
AFACI1640
AFACI1650
AFACI1660
AFACI1670
AFACI1680
AFACI1690
AFACI1700
AFACI1710
AFACI1720
AFACI1730
AFACI1740
AFACI1750
AFACI1760
AFACI1770
AFACI1780
AFACI1790
AFACI1800
AFACI1810
AFACI1820
AFACI1830
AFACI1840
AFACI1850
AFACI1860
AFACI1870
AFACI1880
AFACI1890
AFACI1900
AFACI1910
AFACI1920
AFACI1930
AFACI1940
AFACI1950

```



```

1  C(1)=1/M-1
2  IF (I(1,1)+5*(N-1)) = YU(1+N(I(1,1)-N-1))
3  GO CONTINUE
4  THE MEASUREMENT VECTOR HAS NOW BEEN READ IN
5
6  WRITE (6,99)
7  FORMAT (7,2), 'THE INITIAL VECTOR IS: '//
8  WRITE (6,42) (H(I,J),J=1,NTOT)
9  FORMAT (10,15.4)
10
11 WRITE (6,200) N,M
12 FORMAT (10,1,7,5X,'THE ARMA (1,2,1F,1L,
13 2L) IS BEING IDENTIFIED, '//)
14
15 IF ((M.EQ.0).OR.(N.LQ.5)) GO TO 205
16 WRITE (6,201)
17 (N.LQ.2) WRITE (6,202)
18 (N.LQ.3) WRITE (6,203)
19 (N.LQ.4) WRITE (6,204)
20 FORMAT (5X,'THE DATA IS ALSO REFERENCED IN THE FOLLOWING',
21 1X,'LEAFLETS: ',10X,'U(K-1)**3')
22 FORMAT (10X,'Y(K-1)**3')
23 FORMAT (10X,'Y(K-1)**2')
24 FORMAT (10X,'U(K-1)*Y(K-1)**2')
25 CONTINUE
26
27 (N.LQ.5) GO TO 45
28 WRITE (6,95)
29 FORMAT (7,5X,'ONLY OUTPUT DATA AVAILABLE AND A REGRESSION (N',
30 1X,'), THE MEASUREMENT AVERAGE TERMS UP TO ORDER "M" IS DESIRED.',
31 2X,'SINCE THE U(K-1) TERMS WILL BE SUBSTITUTED WITH PREDICTED',
32 3X,'TERMS ALA COX-JENKINS, '//)
33 CONTINUE
34
35 FORM COEFFICIENT REGRESSION IDENTITY MATRIX
36 70 I=1,N(I)
37 70 J=1,N(I)
38 IF (I,J)=0.001
39 IF (I.LQ.J) I(I,J)=1.000
40
41 FORM THE MEASUREMENT COVARIANCE
42 (I,J)=0.001
43 IF (I.LQ.J) C(I,J)=10*SIGMA
44
45 WRITE (6,95)
46 FORMAT (7,5X,'DATA IS: '//)

```

```

ALAC1760
ALAC1770
ALAC1780
ALAC1790
ALAC2000
ALAC2010
ALAC2020
ALAC2030
ALAC2040
ALAC2050
ALAC2060
ALAC2070
ALAC2080
ALAC2090
ALAC2100
ALAC2110
ALAC2120
ALAC2130
ALAC2140
ALAC2150
ALAC2160
ALAC2170
ALAC2180
ALAC2190
ALAC2200
ALAC2210
ALAC2220
ALAC2230
ALAC2240
ALAC2250
ALAC2260
ALAC2270
ALAC2280
ALAC2290
ALAC2300
ALAC2310
ALAC2320
ALAC2330
ALAC2340
ALAC2350
ALAC2360
ALAC2370
ALAC2380
ALAC2390
ALAC2400
ALAC2410
ALAC2420
ALAC2430

```



```

DO 21 I=1,NML
  21 WRITE (7,52) (Q(I,J),J=1,NTOT)
  52 FORMAT (10E12.4)
DO 30 K=1,NITERATION LOOP
  30 INK=0
  30 NSTAT = N+1
  30 DO 31 K=1,NTOT,IP
  31 KK = INK+1
  30 COMPUTE THE GAINS, C(K), AND FREQU COVARIANCES, PCK(K) & PKNM(F)
  30 CALL CALN
  30 COMPUTE ESTIMATE OF THE ARMA COEFFICIENTS
  30 CALL REGRN
  30 ADAPTIVE KALMAN ARMA COEFFICIENTS ARE NOW COMPUTE FOR
  30 THE STATE, N, AND ARE READY FOR PRINTING
  30 IF (K.EQ.JPNT) GO TO 100
  30 CALL PRNT
  30 IF (K.EQ.LS) GO TO 100
  30 COMPUTE THE POLES AND ZEROS
  30 M=N-1
  30 DO 120 I=1,M,50 TO 170
  30 M=N+1
  30 DO 120 I=1,M,NML
  30 M=N+1
  30 MCT=M+1) = SGL(AHKK(I,K+1))
  30 CALL ZERLY (A,11,FL,IER)
  30 WRITE (7,130)
  30 FORMAT (7F5.8,10HLINE ZEROS: /,9X,ZHRE,9X,ZHIM,/)
  30 MCT = M+1
  30 DO 140 I=1,M
  30 I=10
  30 I=15,+
  30 WRITE (7,160) I,IR
  30 CONTINUE
  30 IF (N.EQ.0) GO TO 100
  30 I(1) = 1.0
  30 DO 140 I=1, .
  30 I(1) = -SGL(CPK(K(I,K+1))
  30 CALL ZERLY (O,4,K2,IER)

```

```

ALAC244L
ALAC2450
ALAC2460
ALAC2470
ALAC2480
ALAC2490
ALAC2500
ALAC2510
ALAC2520
ALAC2530
ALAC2540
ALAC2550
ALAC2560
ALAC2570
ALAC2580
ALAC2590
ALAC2600
ALAC2610
ALAC2620
ALAC2630
ALAC2640
ALAC2650
ALAC2660
ALAC2670
ALAC2680
ALAC2690
ALAC2700
ALAC2710
ALAC2720
ALAC2730
ALAC2740
ALAC2750
ALAC2760
ALAC2770
ALAC2780
ALAC2790
ALAC2800
ALAC2810
ALAC2820
ALAC2830
ALAC2840
ALAC2850
ALAC2860
ALAC2870
ALAC2880
ALAC2890
ALAC2900
ALAC2910

```



```

SUBROUTINE MAIN
THIS SUBROUTINE COMPUTES G(I,K),PKR(K),PKRM(K)
      REAL*8 A(K,K),G,H,PKR,PKRM1,G,Y,G,FI,Z,R,GU,YY,U,YU,G,
      COEFFON,Z(KK(20,600)),G(20,800),F(20,20),PKR(20,20),PKRM1(20,20)
      DIMENSION Y(600),UY(800),YL(800),G(20,800)
      DIMENSION Y(600),G(20,20),FI(20,20),Z,R,K,N,NTOT,KK,IDIAG,KL
      COMMON TEMP(20,20),TEMP1(20,20),TEMP2(20,20),TEMP3(20,20)
      COMPUTE G(K) = F(K/N-1)*F1(K)*H(K)*P(K/N-1)*H1(K+1)*K*-1
      INITIALIZE THE WORKING MATRICES TO ZERO LELE PRECISION
      DO 50 I=1,NTOT
      DO 50 J=1,NTOT
      TEMP1(I,J) = 0.000
      TEMP2(I,J) = 0.000
      TEMP3(I,J) = 0.000
      TEMP4(I,J) = 0.000
      CALL TRANS (TOTOT,NTOT,TEMPA)
      IF (COPRAB(CO,0) .GT. 10.0)
      WRITE (CYL12)
      121 FORMAT (/Y,YA,TEMPA:0,/)
      122 DO 122 I=1,NTOT
      123 WRITE (CYL23) (TEMP1(I,J),J=1,NTOT)
      124 FORMAT (CYL23)
      125 CONTINUE
      CALL PRLE (PKRM1,TEMPA,NTOT,KLU,I,TEMPB)
      IF (COPRAB(CO,0) .GT. 10.0)
      WRITE (CYL15) K
      119 FORMAT (/Z,ZA,TEMPB: TIME=0,/)
      120 DO 120 I=1,NTOT
      121 WRITE (CYL16) (TEMP1(I,J),J=1,NTOT)
      122 FORMAT (CYL16)
      123 CONTINUE
      CALL PRLE (G,TEMP3,NTOT,I,TEMPC)
      124 FORMAT (/I=TEMPC(I,1),K
      125 FORMAT (/Y,YA,TEMPC: TIME=0,/)
      126 FORMAT (/Y,YA,TEMPC: TIME=0,/)
      127 CONTINUE

```

```

ALAC3570C
ALAC3570D
ALAC3570E
ALAC3570F
ALAC3570G
ALAC3570H
ALAC3570I
ALAC3570J
ALAC3570K
ALAC3570L
ALAC3570M
ALAC3570N
ALAC3570O
ALAC3570P
ALAC3570Q
ALAC3570R
ALAC3570S
ALAC3570T
ALAC3570U
ALAC3570V
ALAC3570W
ALAC3570X
ALAC3570Y
ALAC3570Z
ALAC3570AA
ALAC3570AB
ALAC3570AC
ALAC3570AD
ALAC3570AE
ALAC3570AF
ALAC3570AG
ALAC3570AH
ALAC3570AI
ALAC3570AJ
ALAC3570AK
ALAC3570AL
ALAC3570AM
ALAC3570AN
ALAC3570AO
ALAC3570AP
ALAC3570AQ
ALAC3570AR
ALAC3570AS
ALAC3570AT
ALAC3570AU
ALAC3570AV
ALAC3570AW
ALAC3570AX
ALAC3570AY
ALAC3570AZ
ALAC3570BA
ALAC3570BB
ALAC3570BC
ALAC3570BD
ALAC3570BE
ALAC3570BF
ALAC3570BG
ALAC3570BH
ALAC3570BI
ALAC3570BJ
ALAC3570BK
ALAC3570BL
ALAC3570BM
ALAC3570BN
ALAC3570BO
ALAC3570BP
ALAC3570BQ
ALAC3570BR
ALAC3570BS
ALAC3570BT
ALAC3570BU
ALAC3570BV
ALAC3570BW
ALAC3570BX
ALAC3570BY
ALAC3570BZ
ALAC3570CA
ALAC3570CB
ALAC3570CC
ALAC3570CD
ALAC3570CE
ALAC3570CF
ALAC3570CG
ALAC3570CH
ALAC3570CI
ALAC3570CJ
ALAC3570CK
ALAC3570CL
ALAC3570CM
ALAC3570CN
ALAC3570CO
ALAC3570CP
ALAC3570CQ
ALAC3570CR
ALAC3570CS
ALAC3570CT
ALAC3570CU
ALAC3570CV
ALAC3570CW
ALAC3570CX
ALAC3570CY
ALAC3570CZ
ALAC3570DA
ALAC3570DB
ALAC3570DC
ALAC3570DD
ALAC3570DE
ALAC3570DF
ALAC3570DG
ALAC3570DH
ALAC3570DI
ALAC3570DJ
ALAC3570DK
ALAC3570DL
ALAC3570DM
ALAC3570DN
ALAC3570DO
ALAC3570DP
ALAC3570DQ
ALAC3570DR
ALAC3570DS
ALAC3570DT
ALAC3570DU
ALAC3570DV
ALAC3570DW
ALAC3570DX
ALAC3570DY
ALAC3570DZ
ALAC3570EA
ALAC3570EB
ALAC3570EC
ALAC3570ED
ALAC3570EE
ALAC3570EF
ALAC3570EG
ALAC3570EH
ALAC3570EI
ALAC3570EJ
ALAC3570EK
ALAC3570EL
ALAC3570EM
ALAC3570EN
ALAC3570EO
ALAC3570EP
ALAC3570EQ
ALAC3570ER
ALAC3570ES
ALAC3570ET
ALAC3570EU
ALAC3570EV
ALAC3570EW
ALAC3570EX
ALAC3570EY
ALAC3570EZ
ALAC3570FA
ALAC3570FB
ALAC3570FC
ALAC3570FD
ALAC3570FE
ALAC3570FF
ALAC3570FG
ALAC3570FH
ALAC3570FI
ALAC3570FJ
ALAC3570FK
ALAC3570FL
ALAC3570FM
ALAC3570FN
ALAC3570FO
ALAC3570FP
ALAC3570FQ
ALAC3570FR
ALAC3570FS
ALAC3570FT
ALAC3570FU
ALAC3570FV
ALAC3570FW
ALAC3570FX
ALAC3570FY
ALAC3570FZ
ALAC3570GA
ALAC3570GB
ALAC3570GC
ALAC3570GD
ALAC3570GE
ALAC3570GF
ALAC3570GG
ALAC3570GH
ALAC3570GI
ALAC3570GJ
ALAC3570GK
ALAC3570GL
ALAC3570GM
ALAC3570GN
ALAC3570GO
ALAC3570GP
ALAC3570GQ
ALAC3570GR
ALAC3570GS
ALAC3570GT
ALAC3570GU
ALAC3570GV
ALAC3570GW
ALAC3570GX
ALAC3570GY
ALAC3570GZ
ALAC3570HA
ALAC3570HB
ALAC3570HC
ALAC3570HD
ALAC3570HE
ALAC3570HF
ALAC3570HG
ALAC3570HH
ALAC3570HI
ALAC3570HJ
ALAC3570HK
ALAC3570HL
ALAC3570HM
ALAC3570HN
ALAC3570HO
ALAC3570HP
ALAC3570HQ
ALAC3570HR
ALAC3570HS
ALAC3570HT
ALAC3570HU
ALAC3570HV
ALAC3570HW
ALAC3570HX
ALAC3570HY
ALAC3570HZ
ALAC3570IA
ALAC3570IB
ALAC3570IC
ALAC3570ID
ALAC3570IE
ALAC3570IF
ALAC3570IG
ALAC3570IH
ALAC3570II
ALAC3570IJ
ALAC3570IK
ALAC3570IL
ALAC3570IM
ALAC3570IN
ALAC3570IO
ALAC3570IP
ALAC3570IQ
ALAC3570IR
ALAC3570IS
ALAC3570IT
ALAC3570IU
ALAC3570IV
ALAC3570IW
ALAC3570IX
ALAC3570IY
ALAC3570IZ
ALAC3570JA
ALAC3570JB
ALAC3570JC
ALAC3570JD
ALAC3570JE
ALAC3570JF
ALAC3570JG
ALAC3570JH
ALAC3570JI
ALAC3570JJ
ALAC3570JK
ALAC3570JL
ALAC3570JM
ALAC3570JN
ALAC3570JO
ALAC3570JP
ALAC3570JQ
ALAC3570JR
ALAC3570JS
ALAC3570JT
ALAC3570JU
ALAC3570JV
ALAC3570JW
ALAC3570JX
ALAC3570JY
ALAC3570JZ
ALAC3570KA
ALAC3570KB
ALAC3570KC
ALAC3570KD
ALAC3570KE
ALAC3570KF
ALAC3570KG
ALAC3570KH
ALAC3570KI
ALAC3570KJ
ALAC3570KK
ALAC3570KL
ALAC3570KM
ALAC3570KN
ALAC3570KO
ALAC3570KP
ALAC3570KQ
ALAC3570KR
ALAC3570KS
ALAC3570KT
ALAC3570KU
ALAC3570KV
ALAC3570KW
ALAC3570KX
ALAC3570KY
ALAC3570KZ
ALAC3570LA
ALAC3570LB
ALAC3570LC
ALAC3570LD
ALAC3570LE
ALAC3570LF
ALAC3570LG
ALAC3570LH
ALAC3570LI
ALAC3570LJ
ALAC3570LK
ALAC3570LM
ALAC3570LN
ALAC3570LO
ALAC3570LP
ALAC3570LQ
ALAC3570LR
ALAC3570LS
ALAC3570LT
ALAC3570LU
ALAC3570LV
ALAC3570LW
ALAC3570LX
ALAC3570LY
ALAC3570LZ
ALAC3570MA
ALAC3570MB
ALAC3570MC
ALAC3570MD
ALAC3570ME
ALAC3570MF
ALAC3570MG
ALAC3570MH
ALAC3570MI
ALAC3570MJ
ALAC3570MK
ALAC3570ML
ALAC3570MN
ALAC3570MO
ALAC3570MP
ALAC3570MQ
ALAC3570MR
ALAC3570MS
ALAC3570MT
ALAC3570MU
ALAC3570MV
ALAC3570MW
ALAC3570MX
ALAC3570MY
ALAC3570MZ
ALAC3570NA
ALAC3570NB
ALAC3570NC
ALAC3570ND
ALAC3570NE
ALAC3570NF
ALAC3570NG
ALAC3570NH
ALAC3570NI
ALAC3570NJ
ALAC3570NK
ALAC3570NL
ALAC3570NM
ALAC3570NO
ALAC3570NP
ALAC3570NQ
ALAC3570NR
ALAC3570NS
ALAC3570NT
ALAC3570NU
ALAC3570NV
ALAC3570NW
ALAC3570NX
ALAC3570NY
ALAC3570NZ
ALAC3570OA
ALAC3570OB
ALAC3570OC
ALAC3570OD
ALAC3570OE
ALAC3570OF
ALAC3570OG
ALAC3570OH
ALAC3570OI
ALAC3570OJ
ALAC3570OK
ALAC3570OL
ALAC3570OM
ALAC3570ON
ALAC3570OO
ALAC3570OP
ALAC3570OQ
ALAC3570OR
ALAC3570OS
ALAC3570OT
ALAC3570OU
ALAC3570OV
ALAC3570OW
ALAC3570OX
ALAC3570OY
ALAC3570OZ
ALAC3570PA
ALAC3570PB
ALAC3570PC
ALAC3570PD
ALAC3570PE
ALAC3570PF
ALAC3570PG
ALAC3570PH
ALAC3570PI
ALAC3570PJ
ALAC3570PK
ALAC3570PL
ALAC3570PM
ALAC3570PN
ALAC3570PO
ALAC3570PP
ALAC3570PQ
ALAC3570PR
ALAC3570PS
ALAC3570PT
ALAC3570PU
ALAC3570PV
ALAC3570PW
ALAC3570PX
ALAC3570PY
ALAC3570PZ
ALAC3570QA
ALAC3570QB
ALAC3570QC
ALAC3570QD
ALAC3570QE
ALAC3570QF
ALAC3570QG
ALAC3570QH
ALAC3570QI
ALAC3570QJ
ALAC3570QK
ALAC3570QL
ALAC3570QM
ALAC3570QN
ALAC3570QO
ALAC3570QP
ALAC3570QQ
ALAC3570QR
ALAC3570QS
ALAC3570QT
ALAC3570QU
ALAC3570QV
ALAC3570QW
ALAC3570QX
ALAC3570QY
ALAC3570QZ
ALAC3570RA
ALAC3570RB
ALAC3570RC
ALAC3570RD
ALAC3570RE
ALAC3570RF
ALAC3570RG
ALAC3570RH
ALAC3570RI
ALAC3570RJ
ALAC3570RK
ALAC3570RL
ALAC3570RM
ALAC3570RN
ALAC3570RO
ALAC3570RP
ALAC3570RQ
ALAC3570RR
ALAC3570RS
ALAC3570RT
ALAC3570RU
ALAC3570RV
ALAC3570RW
ALAC3570RX
ALAC3570RY
ALAC3570RZ
ALAC3570SA
ALAC3570SB
ALAC3570SC
ALAC3570SD
ALAC3570SE
ALAC3570SF
ALAC3570SG
ALAC3570SH
ALAC3570SI
ALAC3570SJ
ALAC3570SK
ALAC3570SL
ALAC3570SM
ALAC3570SN
ALAC3570SO
ALAC3570SP
ALAC3570SQ
ALAC3570SR
ALAC3570SS
ALAC3570ST
ALAC3570SU
ALAC3570SV
ALAC3570SW
ALAC3570SX
ALAC3570SY
ALAC3570SZ
ALAC3570TA
ALAC3570TB
ALAC3570TC
ALAC3570TD
ALAC3570TE
ALAC3570TF
ALAC3570TG
ALAC3570TH
ALAC3570TI
ALAC3570TJ
ALAC3570TK
ALAC3570TL
ALAC3570TM
ALAC3570TN
ALAC3570TO
ALAC3570TP
ALAC3570TQ
ALAC3570TR
ALAC3570TS
ALAC3570TT
ALAC3570TU
ALAC3570TV
ALAC3570TW
ALAC3570TX
ALAC3570TY
ALAC3570TZ
ALAC3570UA
ALAC3570UB
ALAC3570UC
ALAC3570UD
ALAC3570UE
ALAC3570UF
ALAC3570UG
ALAC3570UH
ALAC3570UI
ALAC3570UJ
ALAC3570UK
ALAC3570UL
ALAC3570UM
ALAC3570UN
ALAC3570UO
ALAC3570UP
ALAC3570UQ
ALAC3570UR
ALAC3570US
ALAC3570UT
ALAC3570UU
ALAC3570UV
ALAC3570UW
ALAC3570UX
ALAC3570UY
ALAC3570UZ
ALAC3570VA
ALAC3570VB
ALAC3570VC
ALAC3570VD
ALAC3570VE
ALAC3570VF
ALAC3570VG
ALAC3570VH
ALAC3570VI
ALAC3570VJ
ALAC3570VK
ALAC3570VL
ALAC3570VM
ALAC3570VN
ALAC3570VO
ALAC3570VP
ALAC3570VQ
ALAC3570VR
ALAC3570VS
ALAC3570VT
ALAC3570VU
ALAC3570VV
ALAC3570VW
ALAC3570VX
ALAC3570VY
ALAC3570VZ
ALAC3570WA
ALAC3570WB
ALAC3570WC
ALAC3570WD
ALAC3570WE
ALAC3570WF
ALAC3570WG
ALAC3570WH
ALAC3570WI
ALAC3570WJ
ALAC3570WK
ALAC3570WL
ALAC3570WM
ALAC3570WN
ALAC3570WO
ALAC3570WP
ALAC3570WQ
ALAC3570WR
ALAC3570WS
ALAC3570WT
ALAC3570WU
ALAC3570WV
ALAC3570WW
ALAC3570WX
ALAC3570WY
ALAC3570WZ
ALAC3570XA
ALAC3570XB
ALAC3570XC
ALAC3570XD
ALAC3570XE
ALAC3570XF
ALAC3570XG
ALAC3570XH
ALAC3570XI
ALAC3570XJ
ALAC3570XK
ALAC3570XL
ALAC3570XM
ALAC3570XN
ALAC3570XO
ALAC3570XP
ALAC3570XQ
ALAC3570XR
ALAC3570XS
ALAC3570XT
ALAC3570XU
ALAC3570XV
ALAC3570XW
ALAC3570XX
ALAC3570XY
ALAC3570XZ
ALAC3570YA
ALAC3570YB
ALAC3570YC
ALAC3570YD
ALAC3570YE
ALAC3570YF
ALAC3570YG
ALAC3570YH
ALAC3570YI
ALAC3570YJ
ALAC3570YK
ALAC3570YL
ALAC3570YM
ALAC3570YN
ALAC3570YO
ALAC3570YP
ALAC3570YQ
ALAC3570YR
ALAC3570YS
ALAC3570YT
ALAC3570YU
ALAC3570YV
ALAC3570YW
ALAC3570YX
ALAC3570YY
ALAC3570YZ
ALAC3570ZA
ALAC3570ZB
ALAC3570ZC
ALAC3570ZD
ALAC3570ZE
ALAC3570ZF
ALAC3570ZG
ALAC3570ZH
ALAC3570ZI
ALAC3570ZJ
ALAC3570ZK
ALAC3570ZL
ALAC3570ZM
ALAC3570ZN
ALAC3570ZO
ALAC3570ZP
ALAC3570ZQ
ALAC3570ZR
ALAC3570ZS
ALAC3570ZT
ALAC3570ZU
ALAC3570ZV
ALAC3570ZW
ALAC3570ZX
ALAC3570ZY
ALAC3570ZZ

```



```

111 DO 40 I=1,NHL1
112   C(I,K+1) = (TEMP(I,1)/TEMP(I,1))
113   GO(I,I+1) = GO(I,K) + (L./ (FUFAT(KK)+1.)) * (G(I,K+1) - GG(I,K))
114   IF (ALFAC(I,0) GE 10.70)
115     WRITE (C,I17)
116     FORMAT (7,2X,'THE CALLS ARE:',/)
117     DO 41 I=1,NHL1
118       WRITE (C,I13) G(I,K+1)
119     CONTINUE
120   CTEMP(I,K) = (1 - C(K)) * P(K/K-1)
121   DO 40 J=1,NHL1
122     TEMP(I,J) = 0.000
123     CALL FURT (TEMP,I,NHL1,I,NHL1,TEMP)
124   IF (TEMP(I,0) GE 10.6)
125     WRITE (C,I151)
126     FORMAT (7,5X,'THE QTY. S(K)*H(K):',/)
127     DO 42 I=1,NHL1
128       CALL (C,I13) (TEMP(I,J),J=1,NHL1)
129     CONTINUE
130   CALL SUB (C,I,TEMP,NHL1,NHL1,TEMPA)
131   IF (TEMP(I,0) GE 10.5)
132     WRITE (C,I141)
133     FORMAT (7,5X,'THE QTY. (1-G(K))*P(K):',/)
134     DO 43 I=1,NHL1
135       CALL (C,I13) (TEMP(I,J),J=1,NHL1)
136     CONTINUE
137   CALL PFCU (TEMP,PKS,I,NHL1,NHL1,PKK)
138   CALL CHT (P(K/K-1) = (K-1/K-1) + G
139   DO 44 I=1,NHL1
140     GO(I,J=1,NHL1)
141   PFCU(I,J)=PFC(I,J) + w(I,J)
142   X=J+1
143   IF (X)

```



```

24 CONTINUE
25 DO 20 J=1,NM1
26  A(I,K+1) = A(I,K) + C(I,F+1)*E(I)
27  RETURN
28  END
29
30 SUBROUTINE TRANS (A,N,M,C)
31 THIS SUBROUTINE TURNS THE TRANSPOSE OF THE N*M
32 MATRIX A AND RETURNS IT AS THE MATRIX C
33
34  REAL*8 A,C
35  INTEGER N,M
36  DO 10 I=1,N
37  DO 10 J=1,M
38  C(J,I) = A(I,J)
39  END
40
41 SUBROUTINE PROD (A,M,N,L,C)
42 THIS SUBROUTINE COMPUTES THE PRODUCT OF THE N*M MATRIX A
43 AND THE M*L MATRIX B AND RETURNS THE RESULTING L*N MATRIX C
44
45  REAL*8 A,C,L
46  INTEGER A(2),C(2),L(2),M,N,L
47  DO 10 I=1,N
48  DO 10 J=1,L
49  C(I,J) = 0.0
50
51  DO 20 K=1,M
52  C(I,J) = C(I,J) + A(I,K)*L(K,J)
53  END
54  END
55
56  RETURN
57  END

```

```

A(1,4)=70
A(1,4)=50
A(1,4)=30
A(1,4)=10
A(1,4)=20
A(1,4)=30
A(1,4)=40
A(1,4)=50
A(1,4)=60
A(1,4)=70
A(1,4)=80
A(1,4)=90
A(1,4)=100
A(1,4)=110
A(1,4)=120
A(1,4)=130
A(1,4)=140
A(1,4)=150
A(1,4)=160
A(1,4)=170
A(1,4)=180
A(1,4)=190
A(1,4)=200
A(1,4)=210
A(1,4)=220
A(1,4)=230
A(1,4)=240
A(1,4)=250
A(1,4)=260
A(1,4)=270
A(1,4)=280
A(1,4)=290
A(1,4)=300
A(1,4)=310
A(1,4)=320
A(1,4)=330
A(1,4)=340
A(1,4)=350
A(1,4)=360
A(1,4)=370
A(1,4)=380
A(1,4)=390
A(1,4)=400
A(1,4)=410
A(1,4)=420
A(1,4)=430
A(1,4)=440
A(1,4)=450
A(1,4)=460
A(1,4)=470
A(1,4)=480
A(1,4)=490
A(1,4)=500
A(1,4)=510
A(1,4)=520
A(1,4)=530
A(1,4)=540
A(1,4)=550
A(1,4)=560
A(1,4)=570
A(1,4)=580
A(1,4)=590
A(1,4)=600
A(1,4)=610
A(1,4)=620
A(1,4)=630
A(1,4)=640
A(1,4)=650
A(1,4)=660
A(1,4)=670
A(1,4)=680
A(1,4)=690
A(1,4)=700
A(1,4)=710
A(1,4)=720
A(1,4)=730
A(1,4)=740
A(1,4)=750
A(1,4)=760
A(1,4)=770
A(1,4)=780
A(1,4)=790
A(1,4)=800
A(1,4)=810
A(1,4)=820
A(1,4)=830
A(1,4)=840
A(1,4)=850
A(1,4)=860
A(1,4)=870
A(1,4)=880
A(1,4)=890
A(1,4)=900
A(1,4)=910
A(1,4)=920
A(1,4)=930
A(1,4)=940
A(1,4)=950
A(1,4)=960
A(1,4)=970
A(1,4)=980
A(1,4)=990
A(1,4)=1000

```



```

SUBROUTINE SUB (C, R, N, M, C)
  THIS SUBROUTINE SUBTRACTS TWO NXH MATRICES AND RETURNS THE
  RESULT IN MATRIX C
  DIMENSION A(20,20), B(20,20), C(20,20)
  INTEGER N, M
  DO 10 I=1, N
    DO 10 J=1, M
      C(I, J) = A(I, J) - B(I, J)
  END DO
  RETURN
  END

SUBROUTINE ADD (A, F, N, M, C)
  THIS SUBROUTINE ADDS THE NXH MATRICES A AND B AND RETURNS THE
  RESULT IN THE MATRIX C
  DIMENSION A(20,20), B(20,20), C(20,20)
  INTEGER N, M
  DO 10 I=1, N
    DO 10 J=1, M
      C(I, J) = A(I, J) + B(I, J)
  END DO
  END

```

```

A17C5100
A17C5101
A17C5110
A17C5120
A17C5130
A17C5140
A17C5150
A17C5160
A17C5170
A17C5180
A17C5190
A17C5200
A17C5210
A17C5220
A17C5230
A17C5240
A17C5250
A17C5260
A17C5270
A17C5280
A17C5290
A17C5300
A17C5310
A17C5320
A17C5330
A17C5340
A17C5350
A17C5360
A17C5370
A17C5380
A17C5390
A17C5400

```



```

3000 CONTINUE
3010 WRITE(6)THE SMALLEST LOADS IN THE NEXT AT THE PROPER
VELOCITY, I(K), OUT AND LOADS IN THE PROPER
LOCATION
3020 H(K) = ( I(I-1)...Y(K-N) ) G(K)...U(K-N) FC...
3030
3040 IF (K) 20, P(K), P(K+1), U, Y, G, E, Z, W, U, Y, U, Y, U, Y, G
CONTINUE, A(K)(20,60), G(20,60), F(20,20), P(K)(20,20), P(K+1)(20,20)
3050 G(0,0), Y(0,0), UY(0,0), YU(0,0), G(20,60), G(20,60)
3060 G(0,0), Y(0,0), G(20,20), F(20,20), Z, R, K, K, M, N, I, K, K, I, O, I, C, N, L
3070
3080
3090
3100
3110
3120
3130
3140
3150
3160
3170
3180
3190
3200
3210
3220
3230
3240
3250
3260
3270
3280
3290
3300
3310
3320
3330
3340
3350
3360
3370
3380
3390
3400
3410
3420
3430
3440
3450
3460
3470
3480
3490
3500
3510
3520
3530
3540
3550
3560
3570
3580
3590
3600
3610
3620
3630
3640
3650
3660
3670
3680
3690
3700
3710
3720
3730
3740
3750
3760
3770
3780
3790
3800
3810
3820
3830
3840
3850
3860
3870
3880
3890
3900
3910
3920
3930
3940
3950
3960
3970
3980
3990
4000

```



```

AFAC6670
AFAC6671
AFAC6672
AFAC6673
AFAC6674
AFAC6675
AFAC6676
AFAC6677
AFAC6678
AFAC6679
AFAC6680
AFAC6681
AFAC6682
AFAC6683
AFAC6684
AFAC6685
AFAC6686
AFAC6687
AFAC6688
AFAC6689
AFAC6690
AFAC6691
AFAC6692
AFAC6693
AFAC6694
AFAC6695
AFAC6696
AFAC6697
AFAC6698
AFAC6699
AFAC6700
AFAC6701
AFAC6702
AFAC6703
AFAC6704
AFAC6705
AFAC6706
AFAC6707
AFAC6708
AFAC6709
AFAC6710
AFAC6711
AFAC6712
AFAC6713
AFAC6714
AFAC6715
AFAC6716
AFAC6717
AFAC6718
AFAC6719
AFAC6720
AFAC6721
AFAC6722
AFAC6723
AFAC6724
AFAC6725
AFAC6726
AFAC6727
AFAC6728
AFAC6729
AFAC6730
AFAC6731
AFAC6732
AFAC6733
AFAC6734
AFAC6735
AFAC6736
AFAC6737
AFAC6738
AFAC6739
AFAC6740
AFAC6741
AFAC6742
AFAC6743
AFAC6744
AFAC6745
AFAC6746
AFAC6747
AFAC6748
AFAC6749

```

```

70 INPUT/OUTPUT IN DATA
IF (N.EQ.0) GO TO 100
IF (N.EQ.1) GO TO 110
L1=0
L2=0
L3=1+2*Y
DO 120 I=1,L1
  L1=L1-1
  H(I,L1+L2) = H(I,L1+L3-1)
  H(I,L3+L1) = Y*(I)
100 IF (N.EQ.2) GO TO 140
INPUT/CROSS INPUT/OUTPUT IN DATA
IF (N.EQ.0) OR (N.EQ.1) GO TO 40
IF (N.EQ.1) GO TO 140
L1=0
L2=0
L3=1+2*Y
DO 130 I=1,L1
  L1=L1-1
  H(I,L1+L2) = H(I,L1+L3-1)
  H(I,L3+L1) = G(I)
140 IF (N.EQ.3) GO TO 160
INPUT/CROSS INPUT/OUTPUT IN DATA
IF (N.EQ.1) GO TO 160
L1=0
L2=0
L3=1+2*Y
DO 150 I=1,L1
  L1=L1-1
  H(I,L1+L2) = H(I,L1+L3-1)
  H(I,L3+L1) = Y*(I)
160 CONTINUE
      IF (N.EQ.4)
      GOTO

```



```

DOUGLASS, CARP, (KP)
THIS SUBROUTINE SIMPLY PRINTS OUT THE VARIABLES MARK AND
G(N) IN CURVATURE FORM FOR PLOTTING PURPOSES
      NR=80  ZI=NS,NI,PKK,PKML,D,Y,G,CI,Z,R,LU,YY,LY,(U,GG
      CURVE, MARK(20,800),G(20,800),F(20,20),PKK(20,ZC),PKK4I(20,20)
      I,DD(800),YY(800),LY(800),YL(800),GG(20,ECC)
      I,DD(800),Y(800),Z(20,20),FI(20,20),Z,K,N,K,K,4,NLI),KK,IDLAC,KL
      I,DD(800)
      REAL *4  X,(20,800),XI(20,E(C)),XE(20,E(C))
      PA=MP-NLI*FI
      I,DD(3,100)  OIUI,KA
      MARK(4,100)  NIUI,KA
      LOC  I,DD(4,15)
      DO 10  I=1,NLI
      DO 10  J=KLI,IP
      XI(I,J) = SGN(MARK(I,J))
      MARK(I,J) = XI(I,J)
      DO 20  I=1,NLI
      DO 20  J=KLI,IP
      X2(I,J) = SGN(G(I,J))
      G(I,J) = X2(I,J)
      LOC  I,DD(4,15)  X7(I,J)
      LOC  I,DD(4,15)
      I,DD(4,15)
      END

```

```

ALAC6700
ALAC6701
ALAC6702
ALAC6703
ALAC6704
ALAC6705
ALAC6706
ALAC6707
ALAC6708
ALAC6709
ALAC6710
ALAC6711
ALAC6712
ALAC6713
ALAC6714
ALAC6715
ALAC6716
ALAC6717
ALAC6718
ALAC6719
ALAC6720
ALAC6721
ALAC6722
ALAC6723
ALAC6724
ALAC6725
ALAC6726
ALAC6727
ALAC6728
ALAC6729
ALAC6730
ALAC6731
ALAC6732
ALAC6733
ALAC6734
ALAC6735
ALAC6736
ALAC6737
ALAC6738
ALAC6739
ALAC6740
ALAC6741
ALAC6742
ALAC6743
ALAC6744
ALAC6745
ALAC6746
ALAC6747
ALAC6748
ALAC6749
ALAC6750
ALAC6751
ALAC6752
ALAC6753
ALAC6754
ALAC6755
ALAC6756
ALAC6757
ALAC6758
ALAC6759
ALAC6760
ALAC6761
ALAC6762
ALAC6763
ALAC6764
ALAC6765
ALAC6766
ALAC6767
ALAC6768
ALAC6769
ALAC6770
ALAC6771
ALAC6772
ALAC6773
ALAC6774
ALAC6775
ALAC6776
ALAC6777
ALAC6778
ALAC6779
ALAC6780
ALAC6781
ALAC6782
ALAC6783
ALAC6784
ALAC6785
ALAC6786
ALAC6787
ALAC6788
ALAC6789
ALAC6790
ALAC6791
ALAC6792
ALAC6793
ALAC6794
ALAC6795
ALAC6796
ALAC6797
ALAC6798
ALAC6799
ALAC6800

```


APPENDIX E

```

100 PROGRAM L4S: WIDROW LEAST MEAN SQUARES FILTER
110 THIS PROGRAM COMPUTES THE MOVING-AVERAGE, MA, (OR EQUIVALENTLY
120 THE TRANSVERSAL FILTER WEIGHTS) COEFFICIENTS OF A LEAST MEANS
130 SQUARES FILTER USING AN ADAPTIVE ALGORITHM WHICH IMPLEMENTS THE
140 METHOD OF STEEPEST DESCENT. THIS ALGORITHM WAS FIRST DEVISED
150 BY W.IDRCW.
160 REAL *8 Y(2000),W(12),X(12),XX(12),U(2000),F,KS,D,C
170 REAL *4 V1(2000),V2(2000),WW(12,2000)
180 INTEGER M,NP,KK
190 THE CONVERGENCE FACTOR IS KS
200 KS = -2.6D-1
210 WRITE (6,100) KS
220 FORMAT (I11,10X,28H)THE CONVERGENCE FACTOR: KS =,F10.4,/ )
230 THE ORDER OF THE FILTER IS M
240 READ (8) M1,M,K2
250 WRITE (6,200) M
260 FORMAT (10X,15,2X,21H)WEIGHTS ARE ESTIMATED,/ )
270 THE DIAGNOSTIC FLAG IS IDIAG
280 IDIAG = 0
290 WRITE (6,300)
300 FORMAT (1X,11H) TIME: 3X,5H)ERROR,4X,6H)OUTPUT,6X,4HW(1),6X,4HW(2)
310 1,6X,4HW(3),6X,4HW(4),6X,4HW(5),6X,4HW(6),6X,4HW(7),6X,4HW(8)
320 2,6X,4HW(9),5X,5HW(10),/ )
330 THE NUMBER OF ITERATIONS IS NP
340 NP = 100
350 DO 10 I=1,M
360 X(I) = C.0D0
370 Y(I) = C.0D0
380 XX(I) = C.0D0
390 DO 20 I=1,300
400 READ (8) V1(I),V2(I)
410 DO 30 I=1,MP
420 U(I) = DBLE(V1(I+10))
430 Y(I) = DBLE(V2(I+10))
440

```

LMS00040
LMS00050
LMS00060
LMS00070
LMS00080
LMS00090
LMS00100
LMS00110
LMS00120
LMS00130
LMS00140
LMS00150
LMS00160
LMS00170
LMS00180
LMS00190
LMS00200
LMS00210
LMS00220
LMS00230
LMS00240
LMS00250
LMS00260
LMS00270
LMS00280
LMS00290
LMS00300
LMS00310
LMS00320
LMS00330
LMS00340
LMS00350
LMS00360
LMS00370
LMS00380
LMS00390
LMS00400
LMS00410
LMS00420
LMS00430
LMS00440
LMS00450
LMS00460
LMS00470
LMS00480
LMS00490
LMS00500
LMS00510


```

C      BEGIN ADAPTIVE ITERATION LOOP
C      DO 40 K=2,NP
C      U = Y(N-1)
C      IF (IDLAG.EQ.0) GO TO 80
C      WRITE (6,500) D
C      100 FORMAT (7,5X,'DESIRED SIGNAL D(K):',5X,E15.5,/)
C      80 CONTINUE
C      COMPUTE ERROR
C      IF (IDLAG.EQ.0) GO TO 110
C      WRITE (6,120)
C      120 FORMAT (7,5X,'W(I):',10X,'X(I):',/,)
C      DO 130 I=1,M
C      130 WRITE (6,140) W(I),X(I)
C      140 FORMAT (5X,2E15.4)
C      110 CONTINUE
C      CALL AMULT1(W,X,1,C)
C      E=U-C
C      1400 WRITE (6,400) K,E,C,(W(I),I=1,M)
C      1400 FORMAT (14,12E15.3)
C      150 KL=K-1
C      DO 150 L=1,M
C      150 W(L,KL) = SNGL(W(L))
C      500 IF (IDLAG.EQ.0) GO TO 50
C      WRITE (6,500)
C      500 FORMAT (7,5X,'X(I):',/,)
C      WRITE (6,600) (X(I),I=1,M)
C      600 FORMAT (5X,115.4)
C      50 CONTINUE
C      COMPUTE W(K+1): THE NEXT STEP PREDICTION
C      DO 60 I=1,M
C      60 X(I) = KS*(X(I)
C      W(I) = W(I) - X(I)
C      SHIFT THE INPUT DATA VECTOR THRU THE DELAY LINE BY ONE
C      L = M+1
C      N = M-1
C      DO 70 I=1,N
C      LL = L-I
C      70 X(LL) = X(LL-1)
C      X(I) = U(K)

```

```

LMS00520
LMS00530
LMS00540
LMS00550
LMS00560
LMS00570
LMS00580
LMS00590
LMS00600
LMS00610
LMS00620
LMS00630
LMS00640
LMS00650
LMS00660
LMS00670
LMS00680
LMS00690
LMS00700
LMS00710
LMS00720
LMS00730
LMS00740
LMS00750
LMS00760
LMS00770
LMS00780
LMS00790
LMS00800
LMS00810
LMS00820
LMS00830
LMS00840
LMS00850
LMS00860
LMS00870
LMS00880
LMS00890
LMS00900
LMS00910
LMS00920
LMS00930
LMS00940
LMS00950
LMS00960
LMS00970
LMS00980
LMS00990

```



```

C      40 CONTINUE
      NPP=NP-1
      WRITE (4,160) M,NPP
      160 FORMAT(2I5)
      DO 170 I=1,M
      DO 170 J=1,NPP
      170 WRITE (4,180) WM(I,J)
      180 FORMAT(4E20.5)
      STOP
      END
C
C

```

```

LMS01000
LMS01010
LMS01020
LMS01030
LMS01040
LMS01050
LMS01060
LMS01070
LMS01080
LMS01090
LMS01100
LMS01110
LMS01120
LMS01130
LMS01140

```



```

SUBROUTINE MMULTI(B,C,M,D)
REAL*8 B,C,D
DIMENSION B(12),C(12)
INTEGER M
      D=0.000
      DO 10 I=1,M
10  D=D+B(I)*C(I)
      RETURN
      END

```

```

LMS01170
LMS01180
LMS01190
LMS01200
LMS01210
LMS01220
LMS01230
LMS01240
LMS01250
LMS01260

```



```

208 FORMAT (/ ,10X,'ONLY THE LAST',I3,' ITERATIONS ARE PRINTED',/)
C
300 WRITE (6,300)
300 FORMAT (9X,'TIME ',10X,5HFRRR,5X,6HOUTPUT,/)
C
C   TOTAL NUMBER OF ITERATIONS IS NP
NP = 4000
N=N+1
NPRT = NP-1PRT
IF ((M.LT.2).OR.(N.LT.1)) GO TO 205
DO 10 I=1,M
A(I) = 0.000
X(I) = 0.000
10 XX(I) = 0.000
C
DO 11 I=1,N
B(I) = 0.000
11 Y(I) = 0.000
C
DO 20 I=1,NP
20 READ (8) V1(I),V2(I)
C
NP = NP-10
DO 30 I=1,NP
U(I) = DBLE(V1(I+10))
30 Y(I) = DBLE(V2(I+10))
C
BEGIN ADAPTIVE ITERATION LOOP
DO 40 K=2,NP
D = Y(K-1)
K3 = K-1
C
IF (D.LT.AG.EQ.0) GO TO 80
WRITE (6,900) K3,D
900 FORMAT (/ ,5X,'DESIRED SIGNAL D(',I3,'):',5X,F15.5,/)
80 CONTINUE
C
COMPUTE ERROR
IF (D.LT.AG.EQ.0) GO TO 110
WRITE (6,122)
122 FORMAT (/ ,5X,'COMPUTE THE ERROR WITH:',/)
WRITE (6,120)
120 FORMAT (/ ,10X,'A(I):',10X,'Y(I):',/)
DO 130 I=1,M
130 WRITE (6,140) A(I),X(I)
140 FORMAT (5X,2F15.4)

```

```

LMS00520
LMS00530
LMS00540
LMS00550
LMS00560
LMS00570
LMS00580
LMS00590
LMS00600
LMS00610
LMS00620
LMS00630
LMS00640
LMS00650
LMS00660
LMS00670
LMS00680
LMS00690
LMS00700
LMS00710
LMS00720
LMS00730
LMS00740
LMS00750
LMS00760
LMS00770
LMS00780
LMS00790
LMS00800
LMS00810
LMS00820
LMS00830
LMS00840
LMS00850
LMS00860
LMS00870
LMS00880
LMS00890
LMS00900
LMS00910
LMS00920
LMS00930
LMS00940
LMS00950
LMS00960
LMS00970
LMS00980
LMS00990

```


LMSC1000
 LMSC1010
 LMSC1020
 LMSC1030
 LMSC1040
 LMSC1050
 LMSC1060
 LMSC1070
 LMSC1080
 LMSC1090
 LMSC1100
 LMSC1110
 LMSC1120
 LMSC1130
 LMSC1140
 LMSC1150
 LMSC1160
 LMSC1170
 LMSC1180
 LMSC1190
 LMSC1200
 LMSC1210
 LMSC1220
 LMSC1230
 LMSC1240
 LMSC1250
 LMSC1260
 LMSC1270
 LMSC1280
 LMSC1290
 LMSC1300
 LMSC1310
 LMSC1320
 LMSC1330
 LMSC1340
 LMSC1350
 LMSC1360
 LMSC1370
 LMSC1380
 LMSC1390
 LMSC1400
 LMSC1410
 LMSC1420
 LMSC1430
 LMSC1440
 LMSC1450
 LMSC1460
 LMSC1470

```

    WRITE (6,121)
    FORMAT (/,10X,'B(1):',10X,'Y(1):',/,/)
121 DO 131 I=1,N
131 WRITE (6,140) B(1),Y(1)
    CONTINUE
C
    CALL WMULT (A,X,M,C1)
    CALL MULTS (B,Y1,N,C2)
C
    IF (LDIAG.EQ.0) GO TO 111
    WRITE (6,112) C1,C2
112 FORMAT (/,5X,'A*X=C1=',E15.4/,5X,'B*Y=C2=',E15.4,/)
111 CONTINUE
C
    Y1(1)=C1+C2
    E=D-Y1(1)
C
    IF (N.GE.NPFI) WRITE (6,400) K8,E,Y1(1)
400 FORMAT (110,2E15.4)
C
    DO 150 L=1,M
150 AA(L,K8) = SNGI(A(L))
    DO 151 L=2,N
151 BB(L-1,K8) = SNGI(B(L))
C
    IF (LDIAG.EQ.0) GO TO 50
    WRITE (6,500)
500 FORMAT (/,5X,'X(1):',/,)
    WRITE (6,600) (X(I),I=1,M)
    WRITE (6,501)
501 FORMAT (/,5X,'Y(1):',/,)
500 WRITE (6,600) (Y1(I),I=2,N)
50 CONTINUE
C
    COMPUTE A(K+1) AND B(K+1): THE NEXT STEP PREDICTION
    DO 60 I=1,M
    AA(I) = KA*F*X(I)
60 A(I) = A(I)-X(I)
C
    DO 61 I=2,N
    YY(I) = KB*F*Y1(I)
61 B(I) = B(I)-YY(I)
C
    IF (LDIAG.EQ.0) GO TO 62
    WRITE (6,C3)
63 FORMAT (/,10X,'AFTER UPDATE:',/,10X,'A(1):',/,)
  
```



```

WRITE (6,64) (A(I), I=1,M)
64 FORMAT (1X,F14.4)
WRITE (6,65)
65 FORMAT (/,10X,'AFTER UPDATE:',/,10X,'B(I):',/,)
66 WRITE (6,64) (B(I), I=1,N)
67 CONTINUE
C
C   SHIFT THE INPUT DATA VECTOR THRU THE DELAY LINE BY ONE
C   IF (M.EQ.1) GO TO 72
L1 = M+1
N1 = M-1
DO 70 I=1,N1
L1 = L1-1
70 X(L1) = X(L1-1)
72 X(L1) = G(K)
C
L2 = N+1
L3 = N-1
DO 71 I=1,L3
L4 = L2-1
71 Y1(L4) = Y1(L4-1)
73 CONTINUE
C
C   WRITE THE ORDER AND 4 PTS TO OUTPUT FILE FOR GRAPHING
NPP=NP-1
NPP1 = 400
N5=N-1
WRITE (4,160) M,NPP1
WRITE (2,160) N5,NPP1
160 FORMAT(2I5)
C
DO 170 I=1,M
DO 170 J=1,NPP,10
170 WRITE (4,180) AA(I,J)
C
DO 171 I=1,N5
DO 171 J=1,NPP,10
171 WRITE (3,180) BB(I,J)
180 FORMAT(4L20.5)
C
WRITE (6,201) (13LA(I), I=1,M)
201 FORMAT (1H1,12(0X,A4),/)
C
DO 202 L=NPP1,NPP
202 WRITE (6,203) (AA(I,L), I=1,M)
203 FORMAT (2X,1D11.3)

```

```

LMS01480
LMS01490
LMS01500
LMS01510
LMS01520
LMS01530
LMS01540
LMS01550
LMS01560
LMS01570
LMS01580
LMS01590
LMS01600
LMS01610
LMS01620
LMS01630
LMS01640
LMS01650
LMS01660
LMS01670
LMS01680
LMS01690
LMS01700
LMS01710
LMS01720
LMS01730
LMS01740
LMS01750
LMS01760
LMS01770
LMS01780
LMS01790
LMS01800
LMS01810
LMS01820
LMS01830
LMS01840
LMS01850
LMS01860
LMS01870
LMS01880
LMS01890
LMS01900
LMS01910
LMS01920
LMS01930
LMS01940
LMS01950

```


LMS01960
 LMS01970
 LMS01980
 LMS01990
 LMS02000
 LMS02010
 LMS02020
 LMS02030
 LMS02040
 LMS02050
 LMS02060
 LMS02070
 LMS02080
 LMS02090
 LMS02100
 LMS02110
 LMS02120

```

C      WRITE (6,201) (I,LAB(I),I=1,N5)
C      L=204 L=NPRI,NPP
C      204 WRITE (6,203) (BB(I,L),I=1,N5)
C          GO TO 206
C      205 WRITE (6,207)
C      207 FORMAT (1H1,/,5X,'EITHER THE NUMERATOR OR THE DENOMINATOR ORDER'
C      1, ' IS INCORRECT: THAT IS,/,5X,'EITHER M OR N IS INCORRECTLY,'
C      2, ' SPECIFIED. READ IN THE CORRECT',/,5X,'VALUES FOR M AND N AND'
C      3, ' TRY AGAIN.',/)
C      206 CONTINUE
C          STOP
C          END
C      20
  
```



```

SUBROUTINE MULT5 (A,B,N,D)
  REAL*8 A(12),B(12),D
  INTEGER N
  D=0.000
  DO 10 I=2,N
    D=D+A(I)*B(I)
  10
  RETURN
END

```

```

LMS02300
LMS02310
LMS02320
LMS02330
LMS02340
LMS02350
LMS02360
LMS02370
LMS02380
LMS02390
LMS02400

```


LIST OF REFERENCES

1. Sage, A.P. and Melsa, J.L., System Identification, Academic Press, 1971.
2. Eykhoff, P., System Identification, Parameter and State Estimation, Wiley, 1974.
3. Astrom, K.J. and Eykhoff, P., "System Identification--A Survey," Automatica, Vol. 7, pp. 123-162, 1971.
4. Mehra, R.K. and Lanriotis, D.G., System Identification: Advances and Case Studies, Academic Press, 1976.
5. Willsky, A.S., "Relationships Between Digital Signal Processing and Control and Estimation Theory," Proceedings of IEEE, Vol. 66, No. 9, September 1978.
6. Perry, F.A., Parametric Modeling of Linear and Nonlinear Systems, Doctoral Dissertation, Naval Postgraduate School, June 1980.
7. Romeo, M., Multichannel ARMA Lattice Modeling with Applications to the Phase Locked Loop, Thesis, Naval Postgraduate School, December 1980.
8. Parker, S.R., "An Autoregressive Moving Average (ARMA) Discrete Nonlinear Model," Proc. of the 1980 Symposium on Circuits and Systems, pp. 918-920, April 1980.
9. Lee, R.C.K., Optimal Estimation, Identification, and Control, M.I.T. Press, Cambridge Mass., 1964.
10. Akaike, H., "Markovian Representation of Stochastic Processes and Its Application to the Analysis of Autoregressive Moving Average Processes," Institute of Statistical Mathematics, Tokyo. Annals., Vol. 26, No. 3, 1974.
11. Widrow, B., "Adaptive Filters I: Fundamentals," Stanford Electronics Labs, Stanford, Ca., Rept. SEL-66-126 (Tech. Rept 6764-6), December, 1966.
12. Clark, G.A., Block Adaptive Filtering and its Implementation to Seismic Event Detection, Doctoral Dissertation, Univ. of California, Santa Barbara, September 1980.
13. Burrus, C.S., "Block Implementation of Digital Filters," IEEE Trans. on Circuit Theory, Vol. CT-18, No. 6, pp. 697-701, November 1971.

14. Burrus, C.S., "Block Realization of Digital Filters," IEEE Trans. on Audio and Electroacoustics, Vol. AV-20, No. 4, pp. 230-235, October 1972.
15. Widrow, B. and McCool, J.M., "A Comparison of Adaptive Algorithms Based on Methods of Steepest Descent and Random Search," IEEE Trans. on Antennas and Propagation, Vol. AP-24, No. 5, pp. 615-637.
16. Markel, J.D. and Gray, Jr., A.H., Linear Prediction of Speech, Springer-Verlag, 1976.
17. Willsky, A.S., Digital Signal Processing and Control and Estimation Theory: Points of Tangency, Areas of Intersection and Parallel Directions, The MIT Press, 1979.
18. Feintuch, P.L., "An Adaptive Recursive LMS Filter," Proceedings IEEE, pp. 1622-1624, November 1976.
19. Johnson, C.R. and Larimore, M.G., "Comments on and Additions to an Adaptive Recursive LMS Filter," Proceedings IEEE, pp. 1399-1402.
20. Widrow, B. and McCool, J., "Comments on an Adaptive Recursive LMS Filter," Proceedings IEEE, pp. 1402-1404, September 1977.
21. Anderson, B.D.O. and Moore, J.R., Optimal Filtering, Prentice Hall, 1979.
22. Griffiths, L.J., "A Comparison of Lattice-Based Adaptive Algorithms," IEEE Symposium on Circuits and Systems, Proc., pp. 742-743, 1980.
23. Kalman, R.E., "New Methods and Results in Linear Prediction and Filtering Theory," Technical Report 61-1, Research Institute for Adv. Studies, Baltimore 12, Md., 1961.
24. Kalman, R.E., "A New Approach to Linear Filter and Prediction Problems," Journal of Basic Engineering, March 1961.
25. Gelb, A., editor, Applied Optimal Estimation, Analytical Science Corp., The M.I.T. Press, Cambridge, Mass., 1974.
26. Box, G.E.D. and Jenkins, G.M., Time Series Analysis, Rev. Ed., Holden-Day, 1967.
27. Maybeck, P.S., Stochastic Models, Estimation and Control, Vol. 1, Academic Press, 1979.
28. Swerling, P., "Topics in Generalized Least Squares Signal Estimation," SIAM J. Appl. Math., Vol. 14, pp. 998-1031, September 1966.

29. Swerling, P., "Modern State Estimation Methods from the Viewpoint of the Method of Least Squares," IEEE Trans. on Auto. Cont., Vol. AC-16, No. 6, December 1971.
30. Swerling, P., "Classes of Signal Processing Procedures Suggested by Exact Minimum Mean Square Error Procedures," SIAM J. Appl. Math., Vol. 14, No. 6, pp. 1199-1224, November 1966.
31. Penrose, R., "A Generalized Inverse for Matrices," Proceedings of the Cambridge Philosophical Society, Vol. 51, 1955.
32. Penrose, R., "On Best Approximate Solutions of Linear Matrix Equations," Proceedings of the Cambridge Philosophical Society, Vol. 52, 1956.
33. Anderson, T.W., The Statistical Analysis of Time Series, Wiley, 1971.
34. Levinson, N., "The Weiner RMS Error Criteria in Filter Design and Prediction," J. Math Physics, Vol. 25, pp. 261-278, January 1947.
35. Ahmed, N. and Fogler, R.J., "On an Adaptive Lattice Prediction and a Related Application," IEEE Circuits and Systems Magazine, Vol. 1, No. 4, pp. 17-23, 1979.
36. Ben-Yakov, U., Efficient Adaptive FIR and IIR Filters, Doctoral Dissertation, Naval Postgraduate School, December 1979.
37. Antoniou, A., Digital Filters Analysis and Design, McGraw-Hill, 1979.
38. Oppenheim, A.V. and Schafer, R.N., Digital Signal Processing, Prentice Hall, Inc., 1975.
39. Kirk, D.E., "Optimal Estimation: An Introduction to the Theory and Applications," Notes at the Naval Postgraduate School, 1975.
40. Carpenter, D.D., "The Effect of Time Delays on the Stability of Practical Phase Locked Receivers with Multiple Conversion by Utilizing Root Locus with Positive Feedback," 1973 IEEE Int. Conf. on Communications, Vol. II, June 11-13, 1973, pp. 33-14--33-19.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4. Professor S.R. Parker, Code 62Px Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	6
5. Professor G.A. Myers, Code 62Mv Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
6. Professor F.R. Richards, Code 55Rh Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
7. Professor D.J. Collins, Code 67Co Department of Aerospace Engineering Naval Postgraduate School Monterey, California 93940	1
8. Professor D.E. Kirk, Code 62Ki Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
9. Professor G. Thaler, Code 62Tr Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
10. Cdr. R.A. Persell, Code 32 Curricular Officer Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1

11. Lt. F.A. Perry, Code 71 1
c/o Commander Naval Ocean Systems Command
San Diego, California 92152
12. Lcdr. D.J. Grandia, SMC 2708 1
Naval Postgraduate School
Monterey, California 93940
13. Lcdr. D.J. Klich, SMC 1433 1
Naval Postgraduate School
Monterey, California 93940
14. LCOL. A. Feit, SMC 1073 1
Naval Postgraduate School
Monterey, California 93940
15. Lt. L.M. Mayoral 4
c/o Commander Carrier Group Seven
FPO San Francisco 96601



192538

Thesis

M3979 Mayoral

c.1

An Adaptive Kalman
Identifier and its
application to linear
and non-linear ARMA
modeling.

thesM3979

An Adaptive Kalman Identifier and its ap



3 2768 001 03338 4

DUDLEY KNOX LIBRARY