System identification methods for adaptive control
by Iraj Sadighi

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering
Montana State University

Abstract:
The objective of this work was to develop a parameter estimation procedure which can overcome some of the problems with existing methods, and which can perform reasonably well under different operating conditions. As a result, an improved algorithm (CLS) is developed in this thesis for identification purposes.

In CLS the basic least-squares algorithm is extended to allow a high degree of adaptation under different operating conditions. The CLS algorithm is capable of tracking time-varying parameters, which is the prime interest in adaptive control. A new data richness test (the ratio-ratio test) is suggested; the test is as effective as many currently used tests, and it requires less computation. The CLS algorithm incorporates this richness test on the covariance matrix to determine when the parameter updating should be suspended. Also, a data normalization procedure is included to improve the numerical accuracy. These modifications were applied to keep the computational requirements for real-time estimation as low as possible. Moreover, this method is easily implemented and has computer storage requirements that are similar to those needed for the basic least-squares algorithms. These characteristics make the CLS method an attractive real-time identification procedure.

The performance characteristics and the effectiveness of the algorithm are shown via selected simulations. It is also shown that the algorithm performs well in situations where many current algorithms are known to fail. A self-tuning regulator is used to investigate the performance of the algorithm under closed-loop situations.

Many systems require models that characterize both system and noise parameters. The IVAML identification algorithm is an existing method that can be applied in some of these cases. In this thesis, attributes of the CLS algorithm are embedded in a modified IVAML algorithm to enhance its effectiveness.

Several computer programs have been written based on the above algorithms as well as other related algorithms. Relevant computer programs are included in the appendices.

SYSTEM IDENTIFICATION METHODS

FOR ADAPTIVE CONTROL

by

Iraj Sadighi

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Electrical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

October 1989

APPROVAL

of a thesis submitted by

Iraj Sadighi

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

10/30/89
Date

D. A. Pierre
Chairperson, Graduate Committee

Approved for the Major Department

10/30/89
Date

Head, Major Department

Approved for the College of Graduate Studies

December 7, 1989
Date

Harvey P. Passmore
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a doctoral degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. I further agree that copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for extensive copying or reproduction of this thesis should be referred to University Microfilms International, 300 North Zeeb Road, Ann Arbor, Michigan 48106, to whom I have granted "the exclusive right to reproduce and distribute copies of the dissertation in and from microfilm and the right to reproduce and distribute by abstract in any format."

Signature Iraj Sadigh

Date 11-20-1989

ACKNOWLEDGMENTS

I would like to thank the members of my committee, Dr. J. R. Smith, Dr. C. R. Vogel, Dr. M. Kejariwal, Dr. W. J. Jameson, and Dr. D. R. Bartholomew, for their support and their interest in this research. Special appreciation is reserved for professor D. A. Pierre, who served as my principal advisor and offered many valuable comments and criticisms throughout the doctoral research.

I would like to thank my wife, Sharareh, and children, Atieh and Talieh, for their patience and understanding during this doctoral program. I am especially grateful to my wife, who has been supportive and has endured many additional family responsibilities in order to make this research possible. I wish to thank my parents Farah Dought and Amir Housain for their continuing faith and encouragement.

The financial support of the Electrical Engineering Department of Montana State University and of the Bonneville Power Administration as well as the Electric Power Research Institute is gratefully appreciated.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABSTRACT	xv
1. INTRODUCTION	1
Adaptive Control	2
2. LITERATURE REVIEW	6
3. RELEVANT PROBABILITY CONCEPTS	12
4. MODELING	17
The Concept of a System	17
System Modeling	17
Types of Models	18
Physical Models	18
Mathematical Models	21
Principles Used in Mathematical Modeling	21
System Description	23
Linear Discrete-Time Representation	24
5. RECURSIVE IDENTIFICATION ALGORITHMS	31
Recursive Least-Squares (RLS) method	32
Other Recursive Parameter Estimation Algorithms	42
RELS	43
RIV	43
RML	46
Unified Representation	48
IVAML Method	49
6. THE NEW APPROACH	61
Introduction	61
Corrector Least-Squares Method (CLS)	63
7. IMPLEMENTATION OF RECURSIVE IDENTIFICATION ALGORITHMS	68
SQRT Factorization	68
UD Factorization	70
Covariance Blowup	73
Persistent Excitation	74
Ratio-Ratio Test	76

TABLE OF CONTENTS -- Continued

	Page
Normalization	79
Data Scaling	80
8. ADAPTIVE CONTROL	81
Optimal Control	82
Steady-State Optimal Control	84
Self-Tuning Regulator (STR)	86
9. SIMULATIONS	89
Example 1	89
Example 2	96
Example 3	103
Example 4	111
Example 5	118
Example 6	125
Example 7	133
Example 8	141
Example 9	148
Example 10	152
Example 11	154
10 CONCLUSION AND FUTURE WORK	162
REFERENCES CITED	165
APPENDICES	175
A - The Generate Subroutine	176
B - RLS Using UD Factorization	186
C - RLS Using SQRT Factorization	191
D - The ILS Subroutine	196
E - The CLS Subroutine	207
F - The IVAML Subroutine	220
G - The LQR Subroutine	233
H - The POLE Subroutine	239
I - The Other Subroutines	243

LIST OF TABLES

Table Page

1. Unified recursive parameter estimation algorithms . 49

LIST OF FIGURES

Figure		Page
1.	Analogy between mechanical and electrical systems	19
2.	System and noise model of a discrete-time process	25
3.	Block diagram representation of the IVAML method	55
4.	The flow chart representation of the CLS algorithm	67
5.	Block diagram of self-tuning regulator (STR)	87
6.	Correlated noise. Measured output	91
7.	Correlated noise. Actual output	92
8.	Correlated noise. Estimate of the actual output (IVAML)	92
9.	Correlated noise. Estimate of the actual output (RLS)	93
10.	Correlated noise. Estimate of the a coefficients (IVAML)	93
11.	Correlated noise. Estimate of the a coefficients (RLS)	94
12.	Correlated noise. Estimate of the b coefficients (IVAML)	94
13.	Correlated noise. Estimate of the b coefficients (RLS)	95
14.	Correlated noise. Input sequence	95
15.	Covariance windup. Measured output	98
16.	Covariance windup. Measured input	98
17.	Covariance windup. Estimate of the a coefficients (RLS)	99
18.	Covariance windup. Estimate of the a coefficients (ILS)	99

LIST OF FIGURES -- Continued

Figure		Page
19.	Covariance windup. Estimate of the a coefficients (CLS)	100
20.	Covariance windup. Estimate of the b coefficients (RLS)	100
21.	Covariance windup. Estimate of the a coefficients (ILS)	101
22.	Covariance windup. Estimate of the a coefficients (CLS)	101
23.	Covariance windup. Trace of the covariance matrix (RLS)	102
24.	Covariance windup. Trace of the covariance matrix (ILS)	102
25.	Covariance windup. Trace of the covariance matrix (CLS)	103
26.	Richness test. Measured output	105
27.	Richness test. Measured input	106
28.	Richness test. Trace of the covariance matrix	106
29.	Richness test. Trace of the P(t-1)/Trace of P(t)	107
30.	Richness test. Condition number of the covariance matrix	107
31.	Richness test. Con. # of P(t-1)/Con. # of P(t)	108
32.	Richness test. (Max. E.V./Min. E.V.) of the P(t)	108
33.	Richness test. (Max. E.V./Min. E.V.) of the Astrom matrix	109
34.	Richness test. Norm of (P(t) times data vector)	109
35.	Richness test. Norm (P(t-1)*D.V.(t-1))/norm (P(t)*D.V.(t))	110
36.	Richness test. Tr.P(t-2)*Tr.P(t)/[Tr.P(t-1)]**2	110

LIST OF FIGURES -- Continued

Figure		Page
37.	Richness test. C{P(t-2)} C{P(t)}/[C{P(t-1)}]**2 .	111
38.	Richness test. Measured output	112
39.	Richness test. Measured input	113
40.	Richness test. Trace of the covariance matrix ..	113
41.	Richness test. Trace of the P(t-1)/Trace of P(t)	114
42.	Richness test. Condition number of the covariance matrix	114
43.	Richness test. Con. # of P(t-1)/Con. # of P(t) ..	115
44.	Richness test. (Max. E.V./Min. E.V.) of the P(t)	115
45.	Richness test. Norm of (P(t) times data vector) .	116
46.	Richness test. Norm (P(t-1)*D.V.(t-1))/norm (P(t)*D.V.(t))	116
47.	Richness test. Tr.P(t-2)*Tr.P(t)/[Tr.P(t-1)]**2 .	117
48.	Richness test. C{P(t-2)} C{P(t)}/[C{P(t-1)}]**2 .	117
49.	Changing dynamics. Measured output	120
50.	Changing dynamics. Measured input	120
51	Changing dynamics. Estimate of the a coefficients (RLS)	121
52.	Changing dynamics. Estimate of the a coefficients (ILS)	121
53.	Changing dynamics. Estimate of the a coefficients (CLS)	122
54.	Changing dynamics. Estimate of the b coefficients (RLS)	122
55.	Changing dynamics. Estimate of the b coefficients (ILS)	123

LIST OF FIGURES -- Continued

Figure		Page
56.	Changing dynamics. Estimate of the b coefficients (CLS)	123
57.	Changing dynamics. Trace of the covariance matrix (RLS)	124
58.	Changing dynamics. Trace of the covariance matrix (ILS)	124
59.	Changing dynamics. Trace of the covariance matrix (CLS)	125
60.	Changing dynamics. Measured output	127
61.	Changing dynamics. Measured input	128
62	Changing dynamics. Estimate of the a coefficients (RLS)	128
63.	Changing dynamics. Estimate of the a coefficients (ILS)	129
64.	Changing dynamics. Estimate of the a coefficients (CLS)	129
65.	Changing dynamics. Estimate of the b coefficients (RLS)	130
66.	Changing dynamics. Estimate of the b coefficients (ILS)	130
67.	Changing dynamics. Estimate of the b coefficients (CLS)	131
68.	Changing dynamics. Trace of the covariance matrix (RLS)	131
69.	Changing dynamics. Trace of the covariance matrix (ILS)	132
70.	Changing dynamics. Trace of the covariance matrix (CLS)	132
71.	Nonlinear system. Inverted pendulum	133
72.	Nonlinear system. Measured output (RLS)	135

LIST OF FIGURES -- Continued

Figure		Page
73.	Nonlinear system. Control action (RLS)	135
74.	Nonlinear system. Measured output (ILS)	136
75.	Nonlinear system. Control action (ILS)	136
76.	Nonlinear system. Measured output (CLS)	137
77.	Nonlinear system. Control action (CLS)	137
78.	Nonlinear system. The 'a' coefficients (RLS) ...	138
79.	Nonlinear system. The 'b' coefficients (RLS) ...	138
80.	Nonlinear system. The 'a' coefficients (ILS) ...	139
81.	Nonlinear system. The 'b' coefficients (ILS) ...	139
82.	Nonlinear system. The 'a' coefficients (CLS) ...	140
83.	Nonlinear system. The 'b' coefficients (CLS) ...	140
84.	Nonlinear system. Measured output (RLS)	142
85.	Nonlinear system. Control action (RLS)	143
86.	Nonlinear system. Measured output (ILS)	143
87.	Nonlinear system. Control action (ILS)	144
88.	Nonlinear system. Measured output (CLS)	144
89.	Nonlinear system. Control action (CLS)	145
90.	Nonlinear system. The 'a' coefficients (RLS) ...	145
91.	Nonlinear system. The 'b' coefficients (RLS) ...	146
92.	Nonlinear system. The 'a' coefficients (ILS) ...	146
93.	Nonlinear system. The 'b' coefficients (ILS) ...	147
94.	Nonlinear system. The 'a' coefficients (CLS) ...	147
95.	Nonlinear system. The 'b' coefficients (CLS) ...	148
96.	Nonlinear system. Measured output (RLS)	149

LIST OF FIGURES -- Continued

Figure		Page
97.	Nonlinear system. Control action (RLS)	150
98.	Nonlinear system. Measured output (CLS)	150
99.	Nonlinear system. Control action (CLS)	151
100.	Nonlinear system. The 'a' coefficients (RLS) ...	151
101.	Nonlinear system. The 'a' coefficients (CLS) ...	152
102.	Nonlinear system. Measured output (CLS)	153
103.	Nonlinear system. Control action (CLS)	153
104.	Nonlinear system. The 'a' coefficients (RLS) ...	154
105.	Input signal enrichment. Measured output	156
106.	Input signal enrichment. Measured input (v=0) ..	156
107.	Input signal enrichment. Measured input (v=0.005)	157
108.	Input signal enrichment. Measured input (v=0.005)	157
109.	Input signal enrichment. Measured input (v=0.05)	158
110.	Input signal enrichment. Measured input (v=0.05)	158
111.	Input signal enrichment. Estimate of the 'a' coefficients (v=0)	159
112.	Input signal enrichment. Estimate of the 'a' coefficients (v=0.005)	159
113.	Input signal enrichment. Estimate of the 'a' coefficients (v=0.05)	160
114.	Input signal enrichment. Trace of the covariance matrix (v=0)	160
115.	Input signal enrichment. Trace of the covariance matrix (v=0.05)	161

LIST OF FIGURES -- Continued

Figure		Page
116.	Input signal enrichment. Trace of the covariance matrix (v=0.005)	161
117.	The generate subroutine	178
118.	RLS using UD factorization	188
119.	RLS using SQRT factorization	193
120.	The ILS subroutine	198
121.	The CLS subroutine	209
122.	The IVAML subroutine	222
123.	The LQR subroutine	235
124.	The POLE subroutine	241
125.	The other subroutines	245

ABSTRACT

The objective of this work was to develop a parameter estimation procedure which can overcome some of the problems with existing methods, and which can perform reasonably well under different operating conditions. As a result, an improved algorithm (CLS) is developed in this thesis for identification purposes.

In CLS the basic least-squares algorithm is extended to allow a high degree of adaptation under different operating conditions. The CLS algorithm is capable of tracking time-varying parameters, which is the prime interest in adaptive control. A new data richness test (the ratio-ratio test) is suggested; the test is as effective as many currently used tests, and it requires less computation. The CLS algorithm incorporates this richness test on the covariance matrix to determine when the parameter updating should be suspended. Also, a data normalization procedure is included to improve the numerical accuracy. These modifications were applied to keep the computational requirements for real-time estimation as low as possible. Moreover, this method is easily implemented and has computer storage requirements that are similar to those needed for the basic least-squares algorithms. These characteristics make the CLS method an attractive real-time identification procedure.

The performance characteristics and the effectiveness of the algorithm are shown via selected simulations. It is also shown that the algorithm performs well in situations where many current algorithms are known to fail. A self-tuning regulator is used to investigate the performance of the algorithm under closed-loop situations.

Many systems require models that characterize both system and noise parameters. The IVAML identification algorithm is an existing method that can be applied in some of these cases. In this thesis, attributes of the CLS algorithm are embedded in a modified IVAML algorithm to enhance its effectiveness.

Several computer programs have been written based on the above algorithms as well as other related algorithms. Relevant computer programs are included in the appendices.

CHAPTER 1

INTRODUCTION

Airplanes, missiles, and automobiles have dynamic properties that depend on speed, loading, etc. The dynamic properties of electric-motor drives change with load. Machines such as those used in power systems are affected by many factors that change in an unpredictable manner. The area of adaptive control is concerned with the study and design of controllers and regulators that adjust to varying properties of these kinds of systems . A feature common to these problems is that a mathematical model of a dynamical system based on observed data from the system is required at some point. This model for example may be needed in order to make decisions that have to be made **"on-line"**, i.e., during the operation of the system. Since in general the parameters of this model are not known a priori, the model should be **"updated"** at each time instant when some new data becomes available. The updating is performed by a recursive algorithm, which is called **recursive identification**. The main interest in recursive identification methods is that they are a key instrument in adaptive control, adaptive filtering, adaptive prediction, and adaptive signal-processing.

This dissertation is outlined as follows. In the next section a simple definition of adaptive control is given. In Chapter 2 the current literature relevant to this research is presented. In Chapter 3 some relevant probability concepts are presented. In Chapter 4 general notation for a general identification problem is given. In Chapter 5 general descriptions are presented for some existing algorithms for identifying parameters of a single-input single-output dynamic process. Relative advantages and disadvantages of these methods also are noted. Chapter 6 contains a detailed description of the Corrector Least-Squares (CLS) algorithm. In Chapter 7 the implementation of the recursive identification algorithms and related issues are presented. Chapter 8 covers the link between recursive identification algorithms and control laws. Chapter 9 is concerned with simulations that demonstrate the effectiveness of the results obtained in this work. The main contributions of the thesis are summarized in Chapter 10.

Adaptive Control

There does not seem to be a generally accepted definition of adaptive control although several have been suggested. According to the **Random House Dictionary**, some of the meanings of "**adapt**" and "**adaptation**" are:

> **adapt**, v.t., 1. To make suitable to requirements or conditions; adjust or modify fittingly;... v.i., 2. To adjust oneself to different conditions, environments, etc.
>
> **adaptation**, 1. The act of adapting. 2. The state of being adapted; adjustment. 3. **Biol.** a. Any alteration in the structure or function of an organism or any of its parts that results from natural selection and by which the organism becomes better fitted to survive and multiply in its environment. b. a form or structure modified to fit changed environment.

It will be noted that the definition above is expressed primarily in terms of biological adaptation to environment. The same definition serves at least to some extent for **"artificial"** or human-made adaptive systems.

From the above definition one can define an adaptive system as follows:

> *An adaptive system is a system whose structure is alterable or adjustable in such a way that its behavior or performance (according to some desired criterion) improves through contact with its environment.*

This definition implies that an adaptive control system must react or adapt itself to changes in its environment. For this purpose, **"environment"** can in turn be defined as that

set of conditions which should ordinarily be taken into account by the system's designer. This set will obviously include such elements as the nature of the system's input signals, the noise against which the system should discriminate, and the values of the various factors upon which the parameters of the system may be dependent. A simple example of an adaptive system is the Automatic Gain Control (AGC) used in radio and television receivers. The function of this circuit is to adjust the sensitivity of the receiver inversely as the average incoming signal strength. The receiver is thus able to adapt to a wide range of input levels and to produce a much narrower range of output intensities.

In general there are at least two elements that always appear in some form in adaptive control systems. These are 1. **Identification** - which refers to the measurement of the dynamic characteristics of the process to be controlled; and 2. **Actuation** - which signifies the generation of an appropriate actuation signal, which in turn modifies the adjustable characteristics.

The identification problem, which could also be defined as the reduction of the dynamic characteristics of the system to a form usable by the controller, is probably the most important aspect of adaptive control, and becomes the central element of such a system because adaptivity implies frequent, automatic, and rapid solution of the identification problem.

Consequently the general adaptive control system requires some form of process identification: the better the identification, the more reliable the control can be.

It should be mentioned that the concept of adaptive control systems leads to logical design techniques for a wide variety of intentionally nonlinear feedback control configurations, systems which would never have been realized starting from the more conventional control-design techniques. As control problems become more complex, the control engineer will probably resort more and more to adaptive control systems which automatically sense and correct themselves against extreme disturbances and a wide range of parameter changes.

CHAPTER 2

LITERATURE REVIEW

The purpose of this chapter is to review some of the literature in the area of recursive system identification.

System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the systems. The techniques of system identification have a wide application area, for example in diverse fields like chemical processes, hydrology, aeronautics, electrical power systems, etc. The main interest in recursive identification methods in control systems is that they are a key instrument in adaptive control. The literature on adaptive control and system identification is extensive. The idea of using recursively identified models for controller design goes back to Kalman [1]. The idea of comparing the controlled plant's output with that of a reference model was first expressed by Whitaker, Yamron, and Kezer [2]. Since then a large number of papers on various aspects of adaptive control and system identification have been published. Survey papers [3], [4] contain an overview of adaptive control theory and its applications and also contain many references.

In the early days of control engineering, identification was concerned with the determination of the transfer function of a system. The transfer function could be found by applying a known input signal, usually a sinusoidal signal, to a "black box" and measuring the response at the output. While this approach is useful in some cases, it cannot be easily extended to nonlinear systems or systems with measurement and process noise.

During the past several years considerable interest has been paid to identification procedures and parameter estimation methods for dynamic processes. Among all the identification algorithms the recursive least-squares (RLS) and its extensions are the most popular. There exist many papers dealing with different aspects of the least-squares method. Let us mention Ljung [5] and More [6] for consistency properties, and Strejc [7] and Sripada and Fisher [8] for implementation aspects. The books by Ljung [9], Mendel [10], Johnson [11], and Isermann [12] also contain comprehensive treatments of the RLS methods. The book by Bierman [13] contains a comprehensive derivation of UDU^T factorization to update the covariance matrix. The paper by Bierman and Thornton [14] looks at filtering and error analysis via the UDU^T covariance factorization. A tutorial paper by Isermann [15] and a survey paper by Astrom and Eykhoff [16] contain overviews of approaches to recursive identification methods and also contain many references.

The paper by Moore and Casalino [17] looks at the robustness to noise of the least-squares based adaptive control. In Saridis [18], Isermann, Baur, Bamberger, Knepo and Siebert [19], Soderstrom, Ljung and Gustavsson [20], and Dugard and Landau [21] comparative studies of different methods are made.

The RLS method is optimal in the sense of minimizing the mean-squared estimation error when the measurement noise is uncorrelated and Gaussian. When noise statistics are unknown the RLS method designed for some Gaussian measurement noise model is no longer optimal, although the estimation error covariance may converge to zero and yet produce biased estimates. The instrumental variable (IV) methods are designed to overcome the convergence problem of the least-squares method that results when the disturbance is correlated to the output measurements. The idea is to replace the sequence of the output measurements by a related sequence called an instrumental variable, which is strongly correlated to the output sequence and is not correlated to the disturbance sequence. The idea has been applied to system identification problems in several ways, depending on how the variables are chosen. Wouters [22] suggests that the instrumental variables be taken as delayed inputs. Other instrumental variables have been used by Finigan and Rowe [23] and Stoica and Soderstrom [24], [25] (delayed inputs and delayed outputs), and Gustavsson, Ljung and Soderstrom

[26] (closed-loop applications). A comparative survey with convergence analysis of these different choices is given by Soderstrom and Stoica [27].

Recursive IV methods have been extensively used by Young [28]. Young [29] has also developed "refined" variants, in which the variables are prefiltered through filters that also are estimated. Adaptive implementations of optimal IV methods have been derived by Young [29], Young and Jakeman [30], and Jakeman and Young [31]. In these papers, however, the algorithm is derived by approximating the likelihood equations. Some IV variants for the multivariable case are discussed and analyzed in Stoica and Soderstrom [32] and Jakeman and Young [31]. In Soderstrom and Stoica [27] consistency and accuracy studies of different IV methods are made. An off-line IV estimator has been shown to produce asymptotically unbiased estimates [33].

From an implementation point of view, the updating equations for RLS, which will be presented in chapter 5, can be seen as a digital filter driven by input and disturbance sequences and producing the output sequence. Needless to say, there exist countless digital filter structures for realizations of these equations. A useful form for implementation of digital filters is the "ladder" or "lattice" structure. Important attributes of lattice filters

are that the number of filter coefficients can be assigned dynamically and that they are insensitive to round-off errors.

The exact lattice equivalent to the least-squares algorithm was derived by Morf, Vieira, and Lee [34]. A comprehensive presentation of these results is given in Lee, Morf, and Friedlander [35]. The books by Cowan and Grant [36] and Haykin [37] contain a detailed derivation of lattice filter implementations of the recursive least-squares algorithm. A ladder implementation of the maximum likelihood method is discussed in Friedlander, Ljung and Morf [38]. A tutorial paper by Friedlander [39] contains an overview of approaches to lattice filters for adaptive processing, and also contains many references. In Strobach [40], [41] a fast recursive covariance ladder algorithm for AutoRegressive Moving-Average (ARMA) system identification is derived. Recursive ladder algorithms for AR and ARMA modeling have been used extensively by Lee, Morf, and Friedlander [42], [35], Lim and Parker [43], and Honig [44]. Porat, Friedlander and Morf [45] also developed a square-root covariance ladder algorithm, which is normalized and provides an efficient recursive solution to the problem of multichannel AR model fitting. In Honig and Messerschmitt [46] investigations of convergence properties of digital lattice filters for both unnormalized and normalized cases are made.

All of the above recursive identification algorithms have their counter parts in the off-line identification procedures. In off-line identification procedures first the measurements of the input and the output of the system are obtained under normal operation of the plant and then these measurements are used to obtain an estimate of the plant unknown parameters. The off-line algorithms are suitable only when estimates are required once and for all or at long intervals, or when computing is cheap, since they process the entire record every time.

Excellent books by Ljung [9], Ljung and Soderstrom [48], Johnson [11], Norton [49], Soderstrom and Stoica [33], Mehra and Lainiots [50], Sinha and Kuzta [51], Goodwin and Payne [52], Goodwin and Sin [53], Mendel [10], and Isermann [12] contain introductory as well as advance topics in the field of system identification.

The analysis of signals produced by a system can also be done by the so-called Prony methods. There exist many papers dealing with different aspects of Prony methods [54] through [64]. Hauer, et al., [55] use a Prony method to analyze power system electromechanical oscillations, and describe a computer program which gives valuable information about modes and damping in the power system. In the paper by Trudnowski et al. [65], a modified Prony method is described which results in more system information by taking advantage of knowledge about the system input.

CHAPTER 3

RELEVANT PROBABILITY CONCEPTS

In many applications of control systems, the actual signals and disturbances received by the control system are random in nature. These signals and disturbances which are random functions of time can often be described only by statistical means. In this chapter we shall briefly review some of the important definitions of probability and random processes. Some previous knowledge of basic probability theory and statistical theory on the part of the reader is assumed [95], [96]. The material presented here consists merely of basic descriptions of some important definitions and concepts of random processes which are used in this work.

Random Variable

A function whose values are real numbers determined by elements in a sample space is called a *random variable*. If a sample space contains a finite number of possibilities or countably infinite sequence with as many elements as there are whole numbers, it is called a discrete sample space, and a random variable defined over this space is called a *discrete random variable*. If a sample space contains a

continuum of possibilities, it is called a continuous sample space, and a random variable defined over this space is called a *continuous random variable*.

Probability Distribution Function

The probability distribution function F(x) is the scalar-valued function that specifies the probability that the random variable X takes on a value less than or equal to a specific value x,

$$F(x) = P(X \leq x)$$

where P means "the probability that".

Probability Density Function

The probability density function, denoted by $p(\varepsilon)$, describes the probabilistic characteristics of a random variable. It is a real-valued function such that $p(\varepsilon) \geq 0$ for all ε in the domain of p and $\int_{-\infty}^{\infty} p(\varepsilon) d\varepsilon = 1$.

Normal Distribution

The density function of the normal random variable ε, with mean μ and variance σ^2, is

$$p(\varepsilon;\mu,\sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} e^{-\frac{(\mu-\varepsilon)^2}{2\sigma^2}} \quad , \quad -\infty < \varepsilon < \infty \tag{3.1}$$

The normal distribution is often referred to as the *Gaussian distribution*.

Expected Value

First moment, average, mean, mathematical expectation, or expected value, are synonymous in the literature on random processes. The n^{th} moment about the origin of a discrete random variable x is defined to be the statistical average of the n^{th} power of x:

$$A_n = Ex^n = \sum_{\forall x} x^n f(x) \tag{3.2}$$

where f(x) is the probability density function. Therefore the first moment of x(t) is just the average, or expected value of x(t).

Correlation Function

In statistical theory, correlation functions and covariances are measures of the statistical dependence of one random signal upon another, or upon itself.

Two random signals $x_1(t)$ and $x_2(t)$ are independent or uncorrelated if

$$E[x_1(t_1)x_2(t_2)] = E[x_1(t_1)]E[x_2(t_2)] \tag{3.3}$$

The reverse is not necessarily true unless the random vectors are Gaussian.

Consider two signals $x_1(t)$ and $x_2(t)$, both functions of time. If the value of one of these signals, at any time depends, in some way, on the value of the other, the two signals are said to be correlated.

The Autocorrelation Function

If the present value of a signal x(t) influences, in some degree, its value at a time t+k in the future, that is x(t+k), where k is a time shift, then the signal x(t) is said to be correlated. A statistical measure of the dependence of the future and past values of a signal on its present value is given by the *Autocorrelation Function* of the signal [95].

The Cross-Correlation Function

It may also be possible for one variable x(t) to influence the future value of another variable y(t). In this case, there will be a correlation between x(t) and y(t+k) where k is a time shift. A quantitative measure of this inter-dependence is provided by the *cross-correlation function* of the two signals [95].

Covariance Matrix

For the random vector x(t) with E[x(t)]=0, the *covariance matrix* is defined as

$$cov[x] = E[xx^T] = \begin{bmatrix} E[x_1x_1] & E[x_1x_2] & \dots & E[x_1x_n] \\ E[x_2x_1] & E[x_2x_2] & \dots & E[x_2x_n] \\ \dots & \dots & \dots & \dots \\ E[x_nx_1] & E[x_nx_2] & \dots & E[x_nx_n] \end{bmatrix} \tag{3.4}$$

CHAPTER 4

MODELING

The Concept of a System

The term *system* is used in such a wide variety of ways that it is difficult to produce a definition broad enough to cover the many uses and, at the same time, concise enough to serve a useful purpose. We begin, therefore, with a simple definition of a system and expand upon it by introducing some of the terms that are commonly used when discussing system identification. A *system* is defined as an aggregation or assemblage of objects joined in some regular interaction or interdependence. While this definition is broad enough to include static systems, the principal interest will be in dynamic systems in which the interactions cause changes over time.

System Modeling

We define a model as the body of information about a system gathered for the purpose of studying the system. Since the purpose of the study will determine the nature of the information that is gathered, there is no unique model of a system. Different models of the same system will be

produced by different analysts interested in different aspects of the system or by the same analyst as his/her understanding of the system changes.

Types of Models

Many types of models have been used in system identification and have been classified in a number of ways. The classification is sometimes made in terms of the nature of the system they model, such as deterministic versus stochastic. For the purpose of this chapter, models will be treated as *physical models* or *mathematical models*.

Physical Models

It is sometimes possible to construct a physical model whose behavior represents the system being studied. The best known examples of physical models are scale models used in wind tunnels and water tanks to study the design of aircraft and ships.

Dynamic physical models rely upon an analogy between the system being studied and some other system of a different nature, the analogy usually depending upon an underlying similarity in the forces governing the behavior of the systems. To illustrate this type of physical model, consider the two systems shown in Figure 1. Figure 1(a) represents a mass that is subject to an applied force F(t) varying with

time, a spring whose force is proportional to its extension or contraction, and a damper that exerts a damping force proportional to the velocity of the mass.

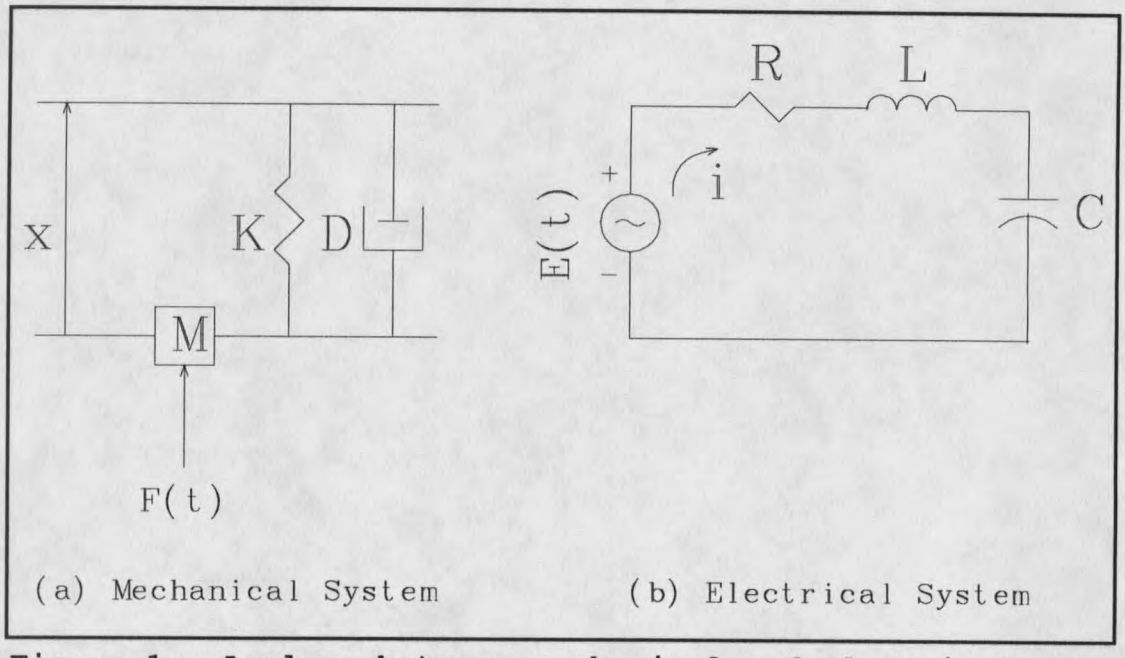

Figure 1. Analogy between mechanical and electrical systems.

It can be shown that the motion of the system is described by the following differential equation:

$$M\frac{d^2x}{dt^2}+D\frac{dx}{dt}+Kx=F(t) \tag{4.1}$$

where x is the distance moved, M is the mass, K is the stiffness of the spring, and D is the damping factor of the damper.

Figure 1(b) represents an electrical circuit with an inductor L, a resistor R, and a capacitor C, connected in series with a voltage source that varies in time according to the function E(t). If q is the charge on the capacitor, it can be shown that the behavior of the circuit is governed by the following differential equation:

$$L\frac{d^2q}{dt^2}+R\frac{dq}{dt}+\frac{q}{C}=E(t) \tag{4.2}$$

Inspection of these two equations shows that they have exactly the same form and that the following equivalences occur between the quantities in the two systems:

Displacement	x	Charge	q
Velocity	$\frac{dx}{dt}$	Current	$i=\frac{dq}{dt}$
Force	F	Voltage	E
Mass	M	Inductance	L
Damping Factor	D	Resistance	R
Spring Stiffness	K	$\frac{1}{\text{Capacitance}}$	$\frac{1}{C}$

The mechanical system and the electrical system are models of each other, and the performance of either can be studied with the other.

Mathematical Models

In a mathematical model, the relationships among the system variables are represented by mathematical expressions. Mathematical models are considered to be either static or dynamic.

A static model displays the relationships between the system variables when the system is in equilibrium.

On the other hand, a dynamic mathematical model allows the changes of system variables to be derived as a function of time.

The derivation of static or dynamic models may be made with an analytical solution or with a numerical computation, depending upon the complexity and nature of the model. In this work the main interest is on mathematical models of dynamic systems; a particular model is constructed from observed input and output data of the system.

Principles Used in Mathematical Modeling

It is not possible to provide a complete set of rules by which mathematical models are built, but a number of guiding principles can be stated. Rather than describing distinct steps carried out in building a model, they describe different viewpoints from which to judge the information to be included in the model.

a) **Block Diagram:**

The description of the system may be organized in a series of blocks, or subsystems. The aim in constructing the blocks is to simplify the specification of the interactions within the system. Each block describes a part of the system that depends upon one or more input variables and results in one or more output variables. The system as a whole can then be described in terms of the interconnections between the blocks. Correspondingly, the system can be represented graphically as a simple block diagram.

b) **Relevance:**

The model should only include those aspects of the system that are relevant to the study objectives. While irrelevant information in the model may not do any harm, it should be excluded because it increases the complexity of the model and causes more work in solving the model. Sometimes a major problem in developing a model is determining what system variable are relevant.

c) **Accuracy:**

The accuracy of the information gathered for the model should be considered.

System Description

Given a physical system S, in order to control it or to predict its evolution, one first should construct its mathematical model. In some circumstances a model can be derived theoretically starting from relationships provided by physics or mechanics, as described in the section on "Physical Models". On the other hand, in many cases one cannot obtain a model of the system from physics and mechanics alone, as in the case of a complicated chemical reaction. Hence it is of great importance to be able to characterize the mathematical model for a system based on its input and output data. The latter approach is known as system identification. Since the measured data are usually corrupted by random noise, the identified system is a system under random influences. There are two basic approaches to system identification, namely: 1) off-line identification, and 2) on-line identification.

An identification procedure is said to be of the "off-line" type if it uses a batch of input and output data which is collected from the system to construct the model.

An identification procedure is said to be of the "on-line" type if it updates the model recursively at each time instant when some new data becomes available. This model for example may be needed in order to make decisions "on-line", i.e.,

during the operation of the system. In this work we are primarily interested in "on-line" or recursive identification methods.

It should be mentioned that, in general, recursive on-line identification methods will not lead to models that are as accurate as can be obtained with off-line identification methods. Also for recursive identification methods the model structure usually must be chosen before starting the procedure. On the other hand, for off-line identification methods, different model structures can be tried in order to obtain a best model structure, one that approximates the actual system in some optimal way.

Linear Discrete-Time Representation

A general representation of a linear discrete-time process in the presence of disturbances is given by linear difference equations of the form

$$y(t) = G_1 u(t) + v(t) \tag{4.3}$$

and

$$v(t) = G_2 e(t) \tag{4.4}$$

where

t	:	Discrete time 0, 1, 2, ...
y(t)	:	Plant output (measurable).
u(t)	:	Plant input (measurable).

v(t) : Disturbance signal which is assumed to be described as an AutoRegressive Moving Average (ARMA) process.

e(t) : Gaussian noise with zero mean and variance of σ^2.

G_1 : Transfer function of the plant.

G_2 : Transfer function of the disturbance.

The block diagram representation of equations (4.3) and (4.4) is given in Figure 2.

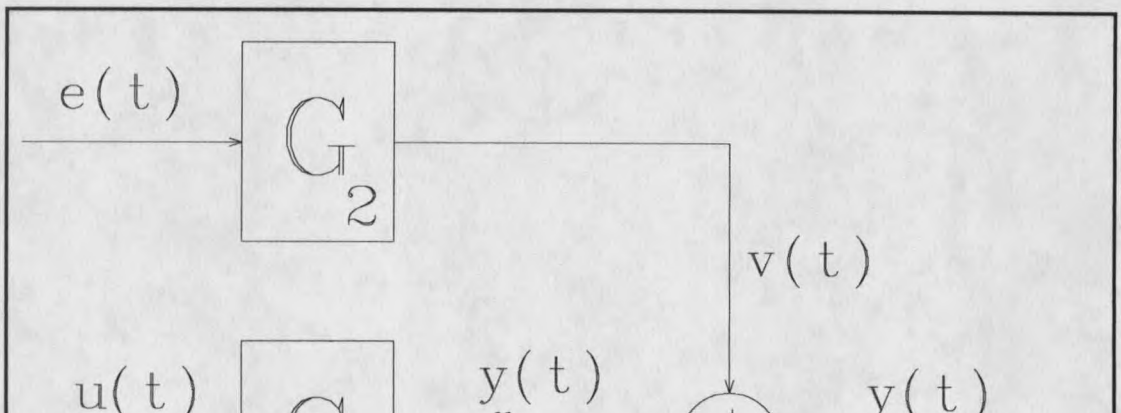

Figure 2. System and noise model of a discrete-time process.

The transfer functions G_1 and G_2 are defined as follows:

$$G_1 = \frac{B(z^{-1})}{A(z^{-1})} \tag{4.5}$$

and

$$G_2 = \frac{D(z^{-1})}{C(z^{-1})} \tag{4.6}$$

where

z^{-1} : Backward shift operator. (i.e. $y(t-1) = z^{-1}y(t)$)

$A(z^{-1})$: $1 + a_1z^{-1} + a_2z^{-2} + \ldots + a_{na}z^{-na}$

$B(z^{-1})$: $b_1z^{-1} + b_2z^{-2} + \ldots + b_{nb}z^{-nb}$

$C(z^{-1})$: $1 + c_1z^{-1} + c_2z^{-2} + \ldots + c_{nc}z^{-nc}$

$D(z^{-1})$: $1 + d_1z^{-1} + d_2z^{-2} + \ldots + d_{nd}z^{-nd}$

The objective of identification procedure is to identify or estimate the unknown parameters in the $A(z^{-1})$ and $B(z^{-1})$ polynomials from a sequence of input and output, u and y, measurements. Also it is desired if possible to estimate the unknown parameters in the $C(z^{-1})$ and $D(z^{-1})$ polynomials, but this is of secondary interest. Our main interest is in estimating the unknown parameters in the $A(z^{-1})$ and $B(z^{-1})$ polynomials.

Equations (4.3) and (4.4) can be written in the following form:

$$y(t) = -a_1y(t-1) - \ \ldots \ -a_{na}y(t-na) + b_1u(t-1) + \ \ldots \ + b_{nb}u(t-nb) + v(t) + a_1v(t-1) + \ \ldots \ + a_{na}v(t-na) \tag{4.7}$$

and

$$v(t) = -c_1v(t-1) - \ldots - c_{nc}v(t-nc) + e(t) + d_1e(t-1) + \ldots + d_{nd}e(t-nd) \tag{4.8}$$

The model (4.5) and (4.6) or (4.7) and (4.8) describes the dynamic relationship between the input, output, and disturbance signals. These equations can be expressed in terms of parameter vectors and data vectors. The parameter vectors are defined as

$$\Phi^T(t) = [a_1 \quad \dots \quad a_{na} \quad b_1 \quad \dots \quad b_{nb} \quad a_1 \quad \dots \quad a_{na}] \tag{4.9}$$

and

$$\Lambda^T(t) = [c_1 \quad \dots \quad c_{nc} \quad d_1 \quad \dots \quad d_{nd}] \tag{4.10}$$

Lagged input-output and disturbance data vectors are defined as

$$\pi^T(t) = [-y(t-1) \quad \dots \quad -y(t-na) \quad u(t-1) \quad \dots \quad u(t-nb) \quad v(t-1) \quad \dots \quad v(t-na)] \tag{4.11}$$

and

$$\Gamma^T(t) = [-v(t-1) \quad \dots \quad -v(t-nc) \quad e(t-1) \quad \dots \quad e(t-nd)] \tag{4.12}$$

Then (4.7) and (4.8) can be written as

$$y(t) = \Phi^T(t)\pi(t) + v(t) \tag{4.13}$$

and

$$v(t) = \Lambda^T(t)\Gamma(t) + e(t) \tag{4.14}$$

This model describes the observed variable y(t) as an unknown linear combination of the components of the observed vector π(t) plus the disturbance. Such a model is called a **linear regression model** in statistics and is a very common type of

model. The components of $\pi(t)$ are then called **regression variables** or **regressors**. Also of interest is the **prediction error**

$$\hat{\varepsilon}(t) = y(t) - \hat{y}(t) \tag{4.15}$$

where

$$\hat{y}(t) = \hat{\Phi}^T(t)\pi(t) + \hat{v}(t) \tag{4.16}$$

Throughout this presentation, "^" is used to denote an estimate. Therefore, for example, Φ will denote the parameters of a model, and estimated values of Φ will be denoted by $\hat{\Phi}$.

It should be mentioned that, the great majority of system identification techniques are discrete-time oriented as a result of the employment of digital computers. Therefore, discrete system models such as equations (4.13) and (4.14) are more convenient to deal with. On the other hand there exist several other representations for systems, like weighting sequence and state-space representations. Each form characterizes the system dynamics in a different way. It is important to note that these forms are uniquely related to one another; therefore a linear system can be regarded as completely identified once a particular form of representation is obtained. To show this point, consider the weighting sequence.

The weighting sequence is the response of a relaxed system to a unit pulse excitation at t=0. This can be represented by convolution summation as

$$y(t) = \sum_{i=-\infty}^{t} h_{t-i} u(i) \tag{4.17}$$

Now, consider the linear discrete-time model defined by equations (4.3) and (4.4). To obtain the relationships between the form, let $G_2 = 0$ and $v = 0$. Next referring to equation (4.13), the general n^{th}-order difference equation relating the input and the output is

$$y(t) + a_1 y(t-1) + \ \ldots \ + a_{na} y(t-na) = b_1 u(t-1) + \ \ldots \ + b_{nb} u(t-nb) \tag{4.18}$$

or

$$y(t) + \sum_{j=1}^{na} a_j y(t-j) = \sum_{j=1}^{nb} b_j u(t-j) \tag{4.19}$$

where t is the integer time index, and a_j and b_j are the constant coefficients.

Equation (4.19) can be written as

$$A(z^{-1}) y(t) = B(z^{-1}) u(t) \tag{4.20}$$

where

$$A(z^{-1}) = 1 + a_1 z^{-1} + \ \ldots \ + a_{na} z^{-na}$$

$$B(z^{-1}) = b_1 z^{-1} + \ \ldots \ + b_{nb} z^{-nb}$$

and

z^{-1} = Backward shift operator (i.e. $y(t-1) = z^{-1} y(t)$).

Therefore, the pulse transfer function of the system is

$$H(z)=\frac{B(z^{-1})}{A(z^{-1})} \tag{4.21}$$

Equating the Z-transform of the weighting sequence as defined by (4.17) to the pulse transfer function as given by equation (4.21) results in the following relation

$$\begin{aligned} Z[h(k)] &= H(z) \\ &= \frac{B(z^{-1})}{A(z^{-1})} \\ &= \frac{b_1 z^{-1} + \quad \ldots \quad + b_{nb} z^{-nb}}{1 + a_1 z^{-1} + \quad \ldots \quad + a_{na} z^{-na}} \end{aligned} \tag{4.22}$$

By expanding the right-hand side of equation (4.22) by long division we get

$$\frac{b_1 z^{-1} + \quad \ldots \quad + b_{nb} z^{-nb}}{1 + a_1 z^{-1} + \quad \ldots \quad + a_{na} z^{-na}} = h_0 + h_1 z^{-1} + \quad \ldots \tag{4.23}$$

or

$$b_1 z^{-1} + \quad \ldots \quad + b_{nb} z^{-nb} = (h_0 + h_1 z^{-1} + \quad \ldots)(1 + a_1 z^{-1} + \quad \ldots \quad + a_{na} z^{-na}) \tag{4.24}$$

and comparing the coefficients of the like terms of z^{-1} on both sides, we obtain

$$\sum_{m=0}^{i} a_m h_{i-m} = \begin{cases} b_i, & i = 1, 2, \ldots, nb \\ 0, & i > nb \end{cases} \tag{4.25}$$

in which $h_0 = 0$ and $a_0 = 1$.

This set of equations ties the weighting sequence h_t directly to the coefficients a_i and b_i in the difference equation (4.18).

CHAPTER 5

RECURSIVE IDENTIFICATION ALGORITHMS

In past years many different identification and parameter estimation methods for dynamic processes have been described in the literature (for example, Ljung [9], Ljung and Soderstrom [48], Isermann [12], [15], and Norton [49]). One reason for the existence of so many recursive identification algorithms is that several different approaches to the subject can be taken. Among these algorithms the recursive least-squares and its extensions are the most popular. Improving computational efficiency, enhancing numerical stability, and avoiding unnecessary storage are key factors influencing the design and implementation of real-time recursive least-squares identification algorithms. Also the problem of detecting changes in dynamical properties of a system has received growing attention in the past several years.

This chapter deals with the derivation and the properties of some of the recursive least-squares based algorithms. The purpose of this chapter is to give a complete derivation of the basic recursive least-squares (RLS) method and also to give a unified description of the various recursive least-squares variants and to show how they are related.

The parameter estimation methods described in this chapter differ, for example, in the assumptions on the structure of the noise or disturbance filter, as well as the general model defined by equations (4.13) and (4.14). The different algorithms are compared from a practical point of view as regards their initialization procedure, implementation, complexity, and numerical properties.

Recursive Least-Squares (RLS) Method

The least-squares theory can be tracked back to Karl Gauss [66]. He used this procedure for astronomical computations. He suggested that the most appropriate values for the unknown parameters are the ones for which the sum of the squares of the differences between the observed and estimated values multiplied by numbers that measure the degree of precision is a minimum. Since that time the method of least-squares has become a major tool for parameter estimation based on experimental data and has been applied for the solution of many technical problems. The least-squares method is very popular among engineers and scientists, because of its simplicity and optimal properties. The estimates obtained from this procedure are, in general, consistent, unbiased, and efficient. The efficiency and simplicity of RLS based algorithms make them particularly attractive for use in real-time estimation problems involving

digital computers. These features are among the reasons that RLS based techniques have been widely used in a variety of engineering applications such as power systems control [66] through [74].

Consider the linear discrete-time model[1] defined by equations (4.3) and (4.4). Let G_2, the transfer function of the disturbance, be defined as:

$$G_2 = \frac{1}{A(z^{-1})} \tag{5.1}$$

By using the above equation, equation (4.13) can be written as:

$$y(t) = \Phi^T(t)\pi(t) + e(t) \tag{5.2}$$

where

$$\Phi^T(t) = [a_1 \quad \dots \quad a_{na} \quad b_1 \quad \dots \quad b_{nb}] \tag{5.3}$$

$$\pi^T(t) = [-y(t-1) \quad \dots \quad -y(t-na) \quad u(t-1) \quad \dots \quad u(t-nb)] \tag{5.4}$$

and {e(t)} is a sequence of independent Gaussian variables with $E[e(t)]=0$ and $E[e^2(t)]=\sigma^2$; σ^2 is a real scalar.

In the literature equation (5.2) is known as a regression equation, and the elements of Φ are regression coefficients.

---

1 In system identification, discrete system models are more convenient to work with. It is a well known fact that any continuous system can be closely approximated by discrete models, if the samples are taken at appropriate sampling frequencies. The development of the least-squares identification methods in this work therefore is based on the discrete system model.

In the present setup of the problem, the model set is given by (5.2), the unknown parameter vector $\Phi(t)$ is defined by (5.3), and our goal is to obtain a good estimate $\hat{\Phi}(t)$ for $\Phi(t)$ based on $\pi(t)$ and $y(t)$.

When formulating an identification problem, a criterion is introduced to give a measure of how well a model fits the experimental data. This criterion is often expressed as

$$V(\Phi) = \sum_{m=1}^{t} g(\varepsilon(m)) \qquad (5.5a)$$

where V is the loss function, Φ is a vector of unknown parameters, t is the present value of discrete time, and ε is the input error, the output error, or a generalized error that depends on Φ. The function g is frequently chosen to be quadratic, but it is possible for it to be of many other forms.

The most intuitively motivated and frequently applied method is the least-squares identification technique, which is associated with the criterion function, $V(\Phi)$, where

$$V(\Phi) = \frac{1}{t} \sum_{m=1}^{t} \alpha_m \varepsilon_m^2(m) \qquad , \qquad t \geq na + nb$$

or

$$V(\Phi) = \frac{1}{t} \sum_{m=1}^{t} \alpha_m [y(m) - \Phi^T(t)\pi(m)]^2 \qquad (5.5b)$$

Here $\{\alpha_m\}$ is a sequence of positive numbers, which allows us to give different weights to different observations; and the Φ that minimizes $V(\Phi)$ depends on t and is denoted by $\hat{\Phi}(t)$. To effect the minimization, $V(\Phi)$ is differentiated with respect to Φ, and the result is equated to zero to determine the conditions that the optimal estimate $\hat{\Phi}$ must satisfy.

Therefore,

$$\frac{dV(\Phi)}{d\Phi}=\frac{-2}{t}\sum_{m=1}^{t}\alpha_m[y(m)-\Phi^T(t)\pi(m)]\pi(m) \tag{5.6}$$

where

$$\frac{d(\Phi^T\pi)}{d\Phi}=\pi \tag{5.7}$$

is used in obtaining the right-hand side of (5.6).

When (5.6) is equated to zero, it follows that

$$\sum_{m=1}^{t}\alpha_m\pi(m)[y(m)-\hat{\Phi}^T(t)\pi(m)]=0$$

which reduces to

$$\hat{\Phi}(t)=\left[\sum_{m=1}^{t}\alpha_m\pi(m)\pi^T(m)\right]^{-1}\sum_{m=1}^{t}\alpha_m\pi(m)y(m) \tag{5.8}$$

provided the inverse exists. This result is known as the weighted least-squares estimate of Φ. If $\{\alpha_m\}\equiv\{1\}$, then the resulting solution is known as the ordinary least-squares estimate of Φ.

It is important to note that equation (5.8) is not in recursive form, but it can be rewritten in a recursive fashion. The need for a recursive solution arises when fresh experimental data are continuously in supply and we wish to improve our parameter estimates by making use of this new information. With the recursive formulation, the estimates can be updated step by step without repeatedly solving the matrix equation (5.8). The recursive procedure is often called an on-line estimation procedure.

To derive this recursive procedure we proceed as follows. First define

$$R(t) = \sum_{m=1}^{t} \alpha_m \pi(m) \pi^T(m) \tag{5.9}$$

or

$$R(t) = R(t-1) + \alpha_t \pi(t) \pi^T(t) \tag{5.10}$$

Solving equation (5.10) for R(t-1) gives

$$R(t-1) = R(t) - \alpha_t \pi(t) \pi^T(t) \tag{5.11}$$

Also from (5.8) we have

$$R(t-1)\hat{\Phi}(t-1) = \sum_{m=1}^{t-1} \alpha_m \pi(m) y(m) \tag{5.12}$$

Substitution of (5.11) into (5.12) gives

$$[R(t) - \alpha_t \pi(t) \pi^T(t)]\hat{\Phi}(t-1) = \sum_{m=1}^{t-1} \alpha_m \pi(m) y(m) \tag{5.13}$$

Similarly, (5.9) is used in (5.8) to obtain

$$\hat{\Phi}(t) = R^{-1}(t) \sum_{m=1}^{t} \alpha_m \pi(m) y(m) \tag{5.14}$$

but

$$\sum_{m=1}^{t} \alpha_m \pi(m) y(m) = \alpha_t \pi(t) y(t) + \sum_{k=1}^{t-1} \alpha_k \pi(k) y(k)$$

or

$$\sum_{m=1}^{t} \alpha_m \pi(m) y(m) = \alpha_t \pi(t) y(t) + [R(t) - \alpha_t \pi(t) \pi^T(t)] \hat{\Phi}(t-1) \tag{5.15}$$

therefore

$$\hat{\Phi}(t) = R^{-1}(t) \{\alpha_t \pi(t) y(t) + [R(t) - \alpha_t \pi(t) \pi^T(t)] \hat{\Phi}(t-1)\}$$

or

$$\hat{\Phi}(t) = R^{-1} \alpha_t \pi(t) y(t) + \hat{\Phi}(t-1) - R^{-1} \alpha_t \pi(t) \pi^T(t) \hat{\Phi}(t-1) \tag{5.16}$$

then

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + R^{-1}(t) \alpha_t \pi(t) [y(t) - \pi^T(t) \hat{\Phi}(t-1)]$$

or

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + R^{-1}(t) \alpha_t \pi(t) [y(t) - \hat{\Phi}^T(t-1) \pi(t)] \tag{5.17}$$

where we used the fact that $\pi^T(t) \hat{\Phi}(t-1) = \hat{\Phi}^T(t-1) \pi(t)$. The foregoing results can be summarized by

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + R^{-1}(t) \alpha_t \pi(t) [y(t) - \hat{\Phi}^T(t-1) \pi(t)] \tag{5.18}$$

and

$$R(t) = R(t-1) + \alpha_t \pi(t) \pi^T(t) \tag{5.19}$$

Equations (5.18) and (5.19) are one form of recursive version of equation (5.8). To obtain $\hat{\Phi}(t)$ using (5.18),

however, it is apparent that we would have to invert R(t), which is an (na+nb)x(na+nb) matrix. This time-consuming operation can be avoided by defining

$$P(t) = R^{-1}(t) \tag{5.20}$$

and by updating P(t) directly, instead of using equation (5.19). In order to update P(t) directly, we state the following well-known matrix inversion lemma.

LEMMA. Let A, B, C, and D be matrices of compatible dimensions, so that the product BCD and the sum A+BCD exist. Then

$$[A+BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1} \tag{5.21}$$

Proof. The proof directly follows that given by Astrom [75]. Premultiply both sides of (5.21) by A + BCD.

$$I = (A+BCD)[A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1}] \tag{5.22}$$

The objective now is to prove the identity (5.22). By direct manipulation, we get

$$\begin{aligned} I &= I + BCDA^{-1} - B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} - BCDA^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \\ &= I + BCDA^{-1} - B[I + CDA^{-1}B][C^{-1} + DA^{-1}B]^{-1}DA^{-1} \\ &= I + BCDA^{-1} - BC[C^{-1} + DA^{-1}B][C^{-1} + DA^{-1}B]^{-1}DA^{-1} \\ &= I + BCDA^{-1} - BCDA^{-1} \end{aligned} \tag{5.23}$$

which completes the proof. ⇐

Using (5.20) in (5.19)

$$P^{-1}(t) = P^{-1}(t-1) + \alpha_t \pi(t)\pi^T(t) \tag{5.24}$$

or

$$P(t) = [P^{-1}(t-1) + \alpha_t \pi(t)\pi^T(t)]^{-1} \tag{5.25}$$

Now we can apply the matrix inversion lemma, equation (5.21), to (5.25) with

$$A = P^{-1}(t-1) \tag{5.26}$$

$$B = \pi(t) \tag{5.27}$$

$$C = \alpha_t \tag{5.28}$$

$$D = \pi^T(t) \tag{5.29}$$

which gives

$$P(t) = P(t-1) - P(t-1)\pi(t)\left[\pi^T(t)P(t-1)\pi(t) + \frac{1}{\alpha_t}\right]^{-1}\pi^T(t)P(t-1) \tag{5.30}$$

or

$$P(t) = P(t-1) - \frac{P(t-1)\pi(t)\pi^T(t)P(t-1)}{\frac{1}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)} \tag{5.31}$$

From (5.31) we also find that

$$\alpha_t P(t)\pi(t) = \alpha_t P(t-1)\pi(t) - \frac{\alpha_t P(t-1)\pi(t)\pi^T(t)P(t-1)\pi(t)}{\frac{1}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)}$$

or

$$\alpha_t P(t)\pi(t) = \frac{P(t-1)\pi(t)}{\frac{1}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)} \tag{5.32}$$

Thus the recursive least-squares estimation can be easily carried out by the following recursive algorithm:

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + K(t)[y(t) - \hat{\Phi}^T(t-1)\pi(t)] \tag{5.33}$$

$$K(t) = \frac{P(t-1)\pi(t)}{\frac{1}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)} \tag{5.34}$$

and

$$P(t) = P(t-1) - \frac{P(t-1)\pi(t)\pi^T(t)P(t-1)}{\frac{1}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)} = [I - K(t)\pi^T(t)]P(t-1) \tag{5.35}$$

The equations (5.33) through (5.35) are known as the recursive least-squares (RLS) algorithm. The above results simply show that the new estimate, $\hat{\Phi}(t)$, is given by the old estimate, $\hat{\Phi}(t-1)$ plus a correction term.

The recursive least-squares algorithm presented above, equations (5.33) through (5.35), is based on an assumption that the unknown parameters are constant. Therefore, this algorithm is not well suited for tracking time-varying parameters. However, a simple modification of the least-squares algorithm can compensate for slowly changing parameters. The method uses an exponential weighting factor on past data so that slow changes in dynamics of the system may be tracked. Consider the criterion function:

$$V(\Phi) = \frac{1}{t}\sum_{m=1}^{t} \tau^{t-m}\alpha_m\varepsilon_m^2(m) \quad , \quad \text{for} \quad 0 < \tau \leq 1 \tag{5.36}$$

in which τ^{t-m} gives an exponential forgetting profile. In such a case we refer to τ as the forgetting factor. The forgetting factor τ is introduced to weight the influence of past errors, and therefore to make the algorithm more adaptive. By minimizing (5.36) with respect to Φ, and following the same procedures as before, one can arrive at the following RLS equations:

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + K(t)[y(t) - \hat{\Phi}^T(t-1)\pi(t)] \tag{5.37}$$

$$K(t) = \frac{P(t-1)\pi(t)}{\frac{\tau}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)} \tag{5.38}$$

and

$$P(t) = \left[P(t-1) - \frac{P(t-1)\pi(t)\pi^T(t)P(t-1)}{\frac{\tau}{\alpha_t} + \pi^T(t)P(t-1)\pi(t)}\right]\frac{1}{\tau} = [I - K(t)\pi^T(t)]\frac{P(t-1)}{\tau} \tag{5.39}$$

We note that the basic RLS algorithm, equations (5.33) through (5.35), is a special case of (5.37) through (5.39) with $\tau \equiv 1$. We observe that the smaller the τ, the larger the norm of P(t) and K(t), hence the heavier are the weights assigned to the more recent data. This implies that the algorithm will always be alert to track changing dynamics or parameter variations. The forgetting factor τ is often chosen experimentally for a given task. We shall say more about τ in chapter 7.

The RLS method converges to the true parameter values if the following two conditions are satisfied:

1) {v(t)} in Figure 2 is a sequence of independent random variables with zero mean values. This is a direct consequence of the model used for the transfer function of the disturbance (i.e. $G_2=\frac{1}{A(z^{-1})}$). Because of this transfer function, v(t) does not depend on what has happened up to time t-1, and hence $E[v(t)\pi(t)]=0$.

2) The input sequence {u(t)} is independent of the zero-mean noise sequence {v(t)}.

Note that if {v(t)} is not white noise, then (usually) $E[v(t)\pi(t)]$ is not equal to zero. This means that we may expect $\hat{\Phi}(N)$ to tend to Φ, the true value, as N approaches infinity, usually only when the two conditions listed above are satisfied.

There exist many papers dealing with different aspects of the least-squares method (see the discussion on page 7).

Other Recursive Parameter Estimation Algorithms

The recursive parameter estimation algorithms such as Recursive Extended Least-Squares (RELS), Recursive Instrumental Variable (RIV), Recursive Maximum Likelihood method (RML) and STochastic Approximation method (STA) have been developed for a wide variety of processes. In this section the basic algorithms along with some important properties of each of these methods are presented.

RELS

This method [12], [9] is more powerful than the least-squares method, because it can provide more accurate estimates under conditions where G_2, the transfer function of disturbance in Figure 2, assumes the following form:

$$G_2 = \frac{D}{A} \tag{5.40}$$

or

$$v(t) = \frac{D}{A} e(t) \tag{5.41}$$

In this case, v(t) is modeled as an ARMA (AutoRegresive Moving Average) process. This type of modeling arises in system identification when colored noise is present in the system; when standard RLS methods are applied to these cases, biased estimates generally result.

RIV

The IV method has a basis in classical statistical estimation theory; but does not require a priori information on the signal and noise statistics.

A disadvantage with RLS estimates is that in general $\pi(t)$ and v(t) will be found to be correlated, and then $\hat{\Phi}(N)$ will not converge to Φ. In such a case, we might replace $\pi(t)$ by a vector $\delta(t)$, such that $\delta(t)$ and v(t) are uncorrelated. The resulting method is called an

Instrumental Variable (IV) method, and $\delta(t)$ is called an instrumental variable vector. An IV method can be as simple as the RLS method, and yet can yield consistent estimates when $\pi(t)$ and $v(t)$ are correlated. The properties of $\delta(t)$ can be simply interpreted to mean that the instrumental variables are uncorrelated with the random disturbances $v(t)$, but strongly correlated with $u(t)$ and $y(t)$ in $\pi(t)$. The most attractive feature of an IV method is its ability to obtain asymptotically unbiased estimates of the process parameters even when the measured data from the output of the process are contaminated with high levels of noise with unknown statistical properties. Also IV estimates are consistent under mild requirements on the system and experimental conditions.

It is obvious that there can be many ways for selecting instrumental variables that satisfy the general conditions stated above. The instrumental variables can be formed by different combinations of inputs, delayed inputs, delayed outputs, filtered inputs and certain external variations. The problem of choosing the instrumental variables is indeed a central one for IV estimation. We can expect that different choices of $\delta(t)$ will lead to different efficiencies and accuracies for the estimate $\hat{\Phi}$.

One choice for $\delta(t)$, Isermann [12], is as follows:

$$\delta^T(t) = [-h(t-1) \quad \dots \quad -h(t-na) \quad u(t-1) \quad \dots \quad u(t-nb)] \tag{5.42}$$

where

$$h(t) = \hat{y}(t) = \Phi_{aux}^T(t)\pi(t) \tag{5.43}$$

$$\Phi_{aux}(t) = (1-\Omega)\Phi_{aux}(t-1) + \Omega\hat{\Phi}(t-1) \tag{5.44}$$

for $0.01 \leq \Omega \leq 0.1$. As can be seen from equation (5.42), Isermann assumed that {y(t)} is correlated with {v(t)}, but that {u(t)} is uncorrelated with {v(t)}.

Wouters [22] suggests that the instrumental variables be taken as delayed inputs. Also, the correlation method developed by Isermann and Baur [77] is an IV method with this interpretation of instrumental variables. Other instrumental variables have been used by Finigan and Rowe [23] and Stoica and Soderstrom [25], [32] (delayed inputs and delayed outputs), and Gustavsson, Ljung and Soderstrom [26] (closed-loop applications). A comparative survey with convergence analysis of these different choices is given by Soderstrom and Stoica [27]. Recursive instrumental variable methods have been extensively used by Young [28], [29]. Young [29] has also developed "refined" variants, in which the variables are prefiltered through filters that also are estimated. Adaptive implementations of optimal IV methods has been derived by Young [29], Young and Jakeman [30], and Jakeman and Young [31]. In these papers, however, the algorithm is derived by approximating the likelihood equations. We

will derive this IV method later in this section. Some IV variants for the multivariable case are discussed and analyzed in Stoica and Soderstrom [32] and Jakeman and Young [31]. In Soderstrom and Stoica [27] consistency and accuracy studies of different instrumental variable methods are made. The book by Soderstrom and Stoica [33] also contains comprehensive treatments of IV methods. Analysis by Soderstrom and Stoica [33] has shown that IV methods are particularly accurate for problems involving large batches of data and infrequent estimate calculations. Also, they show that the IV variants are more appropriate for real-time applications under the presence of correlated noise.

RML

The method of maximum likelihood is a very old standard technique in estimation theory. A basic assumption in RML methods is that, given $\{\varepsilon\}$ as a discrete white random process, we know the form of its probability density function $p(\varepsilon)$. Then, the estimate is the value of the parameter vector $\hat{\Phi}$ which maximizes the probability of obtaining the observations y_i, $i=1, \ldots,t$ with $\pi(t)$ specified. In order to do so, one should first define a likelihood function and then maximize it with respect

to Φ. There exist several different Maximum Likelihood Estimators (MLE), depending on the type of noise distributions. For example:

1) Gaussian distribution $N(0,\sigma^2)$

$$p(\varepsilon)=\frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}e^{\frac{-\varepsilon^2}{2\sigma^2}} \tag{5.45}$$

$$F(\varepsilon)=\log(2\pi\sigma^2)^{\frac{1}{2}}+\frac{\varepsilon^2}{2\sigma^2} \tag{5.46}$$

where $F(\varepsilon)=-\log p(\varepsilon)$.

2) Laplace distribution $L(0,a)$

$$p(\varepsilon)=\frac{1}{2a}e^{\frac{-|\varepsilon|}{a}} \tag{5.47}$$

$$F(\varepsilon)=\log 2a+\frac{|\varepsilon|}{a} \tag{5.48}$$

3) Rectangular distribution $R(0,2a)$

$$p(\varepsilon)=\begin{cases}\frac{1}{2a} & , \quad |\varepsilon|\le a\\ 0 & , \quad |\varepsilon|>a\end{cases} \tag{5.49}$$

$$F(\varepsilon)=\begin{cases}\log(2a) & , \quad |\varepsilon|\le a\\ \infty & , \quad |\varepsilon|>a\end{cases} \tag{5.50}$$

4) Cauchy distribution $C(0,a)$

$$p(\varepsilon)=\frac{1}{\pi a(1+a^{-2}\varepsilon^2)} \tag{5.51}$$

$$F(\varepsilon)=\log(\pi a)+\log(1+a^{-2}\varepsilon^2) \tag{5.52}$$

We can expect that each different choice of distribution function leads to different efficiency and accuracy for the estimate $\hat{\Phi}$. The RML methods are (in general) optimal and stable under certain assumptions. It should be mentioned that, if the actual noise distribution is different from the assumed form, the corresponding RML method may be unstable.

The papers by Astrom [78] and Soderstrom [79] contain a comprehensive derivation of the RML method. Similar methods are derived by Fuhrt [80], Hastings-James and Sage [81] and by Gertler and Banyasz [82]. In the latter two papers, procedures are developed for models where the noise term is an AR rather than MA (Moving Average) process. The books by Ljung [9], and Ljung and Soderstrom [48] contain comprehensive treatments of recursive maximum likelihood methods. Also the paper by Friedlander, Ljung, and Morf [38] contains a ladder implementation of an RML algorithm.

Unified Representation

As mentioned by Isermann [12], most of the least-squares based algorithms such as RLS, RIV, STA, RELS, and RML can be represented uniquely by the following basic equations:

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + K(t)\varepsilon(t) \tag{5.53}$$

$$K(t) = \mu(t+1)P(t)\Omega(t+1) \tag{5.54}$$

and

$$\varepsilon(t) = y(t) - \hat{\Phi}^T(t-1)\Theta(t) \tag{5.55}$$

The only differences between these algorithms are the choices of $\hat{\Phi}$, μ, Ω, Θ, and the noise filter. These differences are summarized in Table 1.

METHOD→	RLS	RIV	RELS	RML
$\hat{\Phi}$	$[\hat{a}_1 \ldots \hat{a}_{na} \hat{b}_1 \ldots \hat{b}_{nb}]$	as RLS	$[\hat{a}_1 \ldots \hat{a}_{na} \hat{b}_1 \ldots \hat{b}_{nb}$ $\hat{d}_1 \ldots \hat{d}_{nd}]$	as RELS
$\Theta^T(t)$	$[-y(t-1) \ldots -y(t-na)$ $u(t-1) \ldots u(t-nb)]$	as RLS	$[-y(t-1) \ldots -y(t-na)$ $u(t-1) \ldots u(t-nb)$ $e(t-1) \ldots e(t-nd)]$	as RELS
$\mu(t)$	$\frac{1}{1+\Theta^T(t)P(t)\Theta(t)}$	$\frac{1}{1+\Theta(t)^T P(t)\Omega(t)}$	as RLS	$\frac{1}{1+\Omega^T(t)P(t)\Omega(t)}$
$P(t+1)$	$[I-K(t)\Theta^T(t)]P(t)$	$[I-K(t)\Omega^T(t)]P(t)$	as RLS	$[I-K(t)\Omega^T(t)]P(t)$
$\Omega(t)$	$\Theta(t)$	$[-h(t-1) \ldots -h(t-na)$ $u(t-1) \ldots u(t-nb)]$ Instrumental Variables	$\Theta(t)$	$[-y_f(t-1) \ldots -y_f(t-na)$ $u_f(t-1) \ldots u_f(t-nb)$ $e_f(t-1) \ldots e_f(t-nd)]$ where $v_f = \frac{1}{D(z^{-1})} v$
Produces unbiased estimates when G_2 of Figure 2 is:	$\frac{1}{A(z^{-1})}$	$\frac{D(z^{-1})}{C(z^{-1})}$ where $C(z^{-1})$ assumed to be known.	$\frac{D(z^{-1})}{A(z^{-1})}$	$\frac{D(z^{-1})}{A(z^{-1})}$

Table 1. Unified recursive parameter estimation algorithms.

IVAML Method

The main goal of this section is to give a general derivation of IVAML (Instrumental Variable Approximate

Maximum Likelihood) parameter identification algorithm. This derivation is for single-input single-output systems. The derivation includes a method for system parameter identification in real-time applications and parallels the approach of Young [29], [30].

Motivated by the need for a more accurate method under conditions of correlated noise specially when the $C(z^{-1})$, last row of the RIV column in Table 1, is unknown, Young [29], [30] introduced the IVAML algorithm. Although the IVAML method requires considerably more storage and computation than the previously mentioned methods, it is much more accurate than these methods when the disturbance is correlated noise.

Consider the linear discrete-time model defined by equations (4.3) and (4.4). The {e(t)} is assumed to have zero mean, with variance of σ^2. Also it should be statistically independent of the input signal {u(t)}, i.e.,

$$E[e_k]=0 \quad ; \quad E[e_j e_k]=\sigma^2\delta_{jk} \quad ; \quad E[e_j u_k]=0 \tag{5.56}$$

for all j and k, where

$$\delta_{jk}=1 \quad , \quad \text{for} \quad j=k \tag{5.57}$$

and

$$\delta_{jk}=0 \quad , \quad \text{for} \quad j\neq k \tag{5.58}$$

In what follows we assume that the estimation of the unknown parameters in the system represented by equations (4.3) and (4.4) is given by

$$y(t)=\frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}u(t)+\hat{v}(t) \tag{5.59}$$

where

$$\hat{v}(t)=\frac{\hat{D}(z^{-1})}{\hat{C}(z^{-1})}\hat{e}(t) \tag{5.60}$$

and the "^" operator signifies the estimate of the corresponding terms in equations (4.3) and (4.4).

Criterion Function

Given a prediction model (5.59) for the system given by equations (4.3) and (4.4), a natural measure of its validity is the prediction error,

$$\varepsilon(t)=y(t)-\hat{y}(t) \tag{5.61}$$

This error can be evaluated using the data up to time t. In the literature there exist a number of different criterion functions for minimizing this error; for example least-squares based identification algorithms try to minimize the sum of the mean-square observation errors, given by equation (5.5). The convergence of these algorithms is guaranteed if {ε(t)} is a sequence of independent (Guassian) random variables with zero mean values. This means that the {y(t)} should be uncorrelated to {ε(t)}. This is not the case in general. On the other hand, if {y(t)} is correlated to {ε(t)} we cannot expect that minimization of the above criterion will lead to convergence of the parameter estimates

to their true values. The IVAML algorithm is designed to overcome this convergence problem and can be applied when {ε(t)} is correlated to {y(t)}. The IVAML method is based on the log likelihood function[2] as follows:

$$L(a,b,\sigma^2,y,u) = -\frac{T}{2}\mathrm{Ln}(2\pi) - \frac{T}{2}\mathrm{Ln}(\sigma^2) - \frac{1}{2\sigma^2}\left[\frac{C}{D}y - \frac{BC}{AD}u\right]^T\left[\frac{C}{D}y - \frac{BC}{AD}u\right] \quad (5.62)$$

where A, B, C, and D are defined in Chapter 4, and

$$a = [a_1 \quad a_2 \quad \ldots \quad a_n]^T$$

$$b = [b_1 \quad b_2 \quad \ldots \quad b_n]^T$$

$$y = [y_1 \quad y_2 \quad \ldots \quad y_n]^T$$

$$u = [u_1 \quad u_2 \quad \ldots \quad u_n]^T$$

The problem we are faced with is that of finding an estimate of a and b, based on observations of y and u. A reasonable estimator of a and b is one that selects them so that the observed event becomes as "likely as possible". This means that we want

$$\max_{a,b} L(\quad \text{as above} \quad) \quad (5.63)$$

---

2 The following notation is used for the joint probability density function for Φ and {y(t)},

$$f(\Phi; y(1), \ldots, y(n)) = f(\Phi; y^n)$$

This is a deterministic function of Φ once the numerical values of the observations y(i) are inserted. It is called the likelihood function.

and let the maximizing vector $\Phi_{ML}(y,u)$ be our estimator. Estimators based on this criterion are known as maximum likelihood estimators (MLE).

To continue the development of the IVAML algorithm, the derivation is divided into two parts. The first part is concerned with estimation of unknown coefficients in the A and B polynomials while assuming that the C and D polynomials are known. The IV method is applied to estimate these unknown coefficients, which describe the dynamic part of the system in Figure 2. The second part is concerned with estimation of unknown coefficients in the C and D polynomials while assuming that the A and B polynomials are known. The AML method is applied to estimate these unknown coefficients, which describes the disturbance part of the system in Figure 2.

Part 1 (IV)

We need to find the maximum of (5.62); this can be accomplished by taking the partial differential of L with respect to a_i, b_i, (i=1,2,...,n), and σ^2 respectively. This differentiation will result in the following equations:

$$\frac{\partial L}{\partial a_i}=\frac{1}{\sigma^2}\sum_{k=2n+1}^{T}\left[\frac{C}{D}y(k)-\frac{BC}{AD}u(k)\right]\frac{BC}{A^2D}z^{-i}u(k)=0 \qquad (5.64)$$

for i = 1, 2, ..., n

$$\frac{\partial L}{\partial b_i} = \frac{1}{\sigma^2} \sum_{k=2n+1}^{T} \left[\frac{C}{D} y(k) - \frac{BC}{AD} u(k) \right] \frac{C}{AD} z^{-i} u(k) = 0 \tag{5.65}$$

for i = 1, 2, ..., n

$$2\frac{\partial L}{\partial \sigma^2} = -\frac{T}{\sigma^2} + \frac{1}{\sigma^4} \sum_{k=2n+1}^{T} \left[\frac{C}{D} y(k) - \frac{BC}{AD} u(k) \right]^2 = 0 \tag{5.66}$$

The above equations are highly nonlinear in the a_i and b_i parameters, for given C and D polynomials. In order to make these equations linear in a_i and b_i we define the following discrete 'pre-filter' variables, as shown in Figure 3.

$$y^*(t) = \frac{\hat{C}}{\hat{A}\hat{D}} y(t) \tag{5.67}$$

$$u^*(t) = \frac{\hat{C}}{\hat{A}\hat{D}} u(t) \tag{5.68}$$

$$\hat{x}^*(t) = \frac{\hat{B}}{\hat{A}} u^*(t) \tag{5.69}$$

By assuming that $\hat{A}$, $\hat{B}$, $\hat{C}$, and $\hat{D}$ are sufficiently close to A, B, C, and D, respectively and by substitution of equations (5.67), (5.68), and (5.69) into equations (5.64) and (5.65), the following equations, which are linear in a_i and b_i, are obtained:

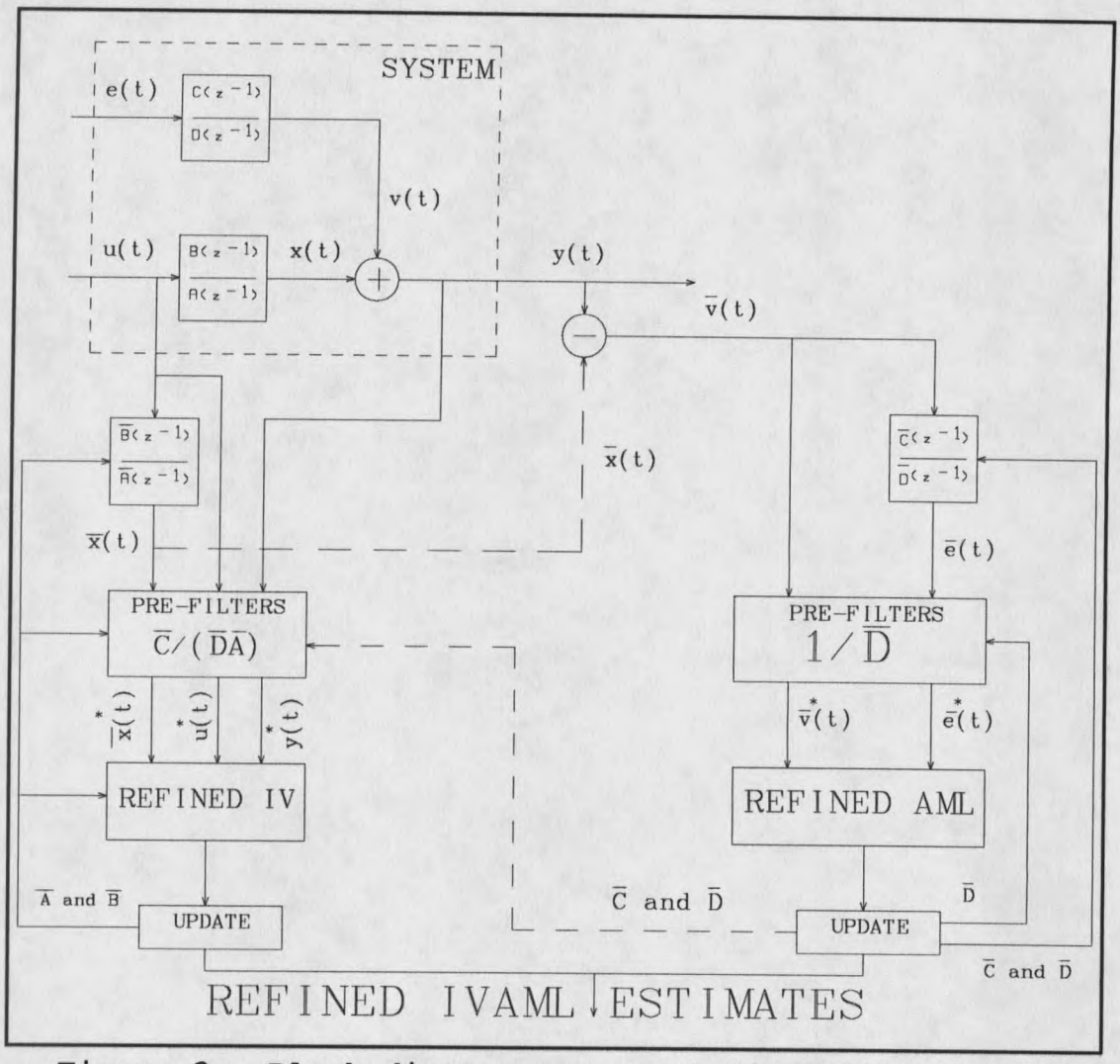

Figure 3. Block diagram representation of the IVAML method.

$$\sum_{k=2n+1}^{T} [A y^*(k) - B u^*(k)] \hat{x}^*(k-i) = 0 \qquad \text{for} \quad i = 1,2,\ldots,n \tag{5.70}$$

$$\sum_{k=2n+1}^{T} [A y^*(k) - B u^*(k)] u^*(k-i) = 0 \qquad \text{for} \quad i = 0,1,2,\ldots,n \tag{5.71}$$

By using the above equations, (5.70) and (5.71), it can be shown that a_i and b_i, the unknown coefficients in A and B polynomials, can be estimated recursively by using the following equations:

$$\hat{\Phi}(t) = \hat{\Phi}(t-1) + P(t)\pi(t)[y^*(t) - \Gamma^T(t)\hat{\Phi}(t-1)] \tag{5.72}$$

and

$$P(t) = \left[P(t-1) - \frac{P(t-1)\pi(t)\pi^T(t)P(t-1)}{\tau + \pi^T(t)P(t-1)\pi(t)} \right] \frac{1}{\tau} \tag{5.73}$$

where

$$\hat{\Phi}(t) = [\hat{a}_1 \quad \dots \quad \hat{a}_n \quad \hat{b}_1 \quad \dots \quad \hat{b}_n]^T \tag{5.74}$$

$$\pi(t) = [-\hat{x}^*(t-1) \quad \dots \quad -\hat{x}^*(t-n) \quad u^*(t-1) \quad \dots \quad u^*(t-n)]^T \tag{5.75}$$

$$\Gamma(t) = [-y^*(t-1) \quad \dots \quad -y^*(t-n) \quad u^*(t-1) \quad \dots \quad u^*(t-n)]^T \tag{5.76}$$

τ : Forgetting factor.

P : Covariance matrix.

π : Instrumental variable vector.

Γ : Data vector.

$\hat{\Phi}(t)$ and P(t) of (5.72) and (5.73) are computed in the "refined IV" block of Figure 3.

Part 2 (AML)

Now let us find the maximum of equation (5.62) with the A and B polynomials assumed to be known. This can be accomplished by taking the partial differential of L with respect to c_i and d_i, i=1,2,...,n. This differentiation results in the following equations:

$$\frac{\partial L}{\partial c_i} = \frac{1}{\sigma^2} \sum_{k=2n}^{T} \left[\frac{C}{D} y(k) - \frac{BC}{AD} u(k) \right] \left[\frac{1}{D} z^{-i} y(k) - \frac{B}{AD} z^{-i} u(k) \right] = 0 \tag{5.77}$$

$$\frac{\partial L}{\partial d_i}=\frac{1}{\sigma^2}\sum_{k=2n}^{T}\left[\frac{C}{D}y(k)-\frac{BC}{AD}u(k)\right]\left[-\frac{C}{D^2}z^{-i}y(k)+\frac{BC}{AD^2}z^{-i}u(k)\right]=0 \qquad (5.78)$$

for i = 1,2, ..., n.

From (5.59) and (5.60) we have

$$\hat{v}(t)=y(t)-\frac{\hat{B}}{\hat{A}}u(t) \qquad (5.79)$$

and

$$\hat{e}(t)=\frac{\hat{C}}{\hat{D}}\hat{v}(t) \qquad (5.80)$$

Substitution of equations (5.79) and (5.80) into equations (5.77) and (5.78) results in the following two equations:

$$\sum_{k=2n}^{T}\left[\frac{C}{D}y(k)-\frac{BC}{AD}u(k)\right]\frac{1}{D}z^{-i}v(k)=0 \qquad (5.81)$$

$$\sum_{k=2n}^{T}\left[\frac{C}{D}y(k)-\frac{BC}{AD}u(k)\right]\frac{1}{D}z^{-i}e(k)=0 \qquad (5.82)$$

for i=1,2, ..., n.

The terms in the square brackets of equations (5.81) and (5.82) can be written as

$$\frac{C}{D}y(k)-\frac{BC}{AD}u(k)=\frac{C}{D}v(k)=e(k) \qquad (5.83)$$

Also,

$$e(t)=v(t)+c_1v(t-1)+\ldots+c_nv(t-n)-d_1e(t-1)-\ldots-d_ne(t-n)$$

$$=Cv(t)-D^*e(t) \quad ; \quad D^*=d_1z^{-1}+\ldots+d_nz^{-n} \qquad (5.84)$$

The following 'pre-filter' variables are defined:

$$v^*(t)=\frac{1}{D}v(t) \tag{5.85}$$

and

$$e^*(t)=\frac{1}{D}e(t) \tag{5.86}$$

By using equations (5.83), (5.84), (5.85), and (5.86), we can rewrite equations (5.81) and (5.82), as follows:

$$\sum_{k=2n}^{T}[Cv(k)-D^*e(k)]v^*(k-i)=0 \tag{5.87}$$

and

$$\sum_{k=2n}^{T}[Cv(k)-D^*e(k)]e^*(k-i)=0 \tag{5.88}$$

for i = 1,2, ..., n.

By using the above equations, (5.87) and (5.88), it can be shown that c_i and d_i, the unknown coefficients in the C and D polynomials, can be estimated recursively by using the following equations:

$$\hat{\Phi}(t)=\hat{\Phi}(t-1)+P(t)\pi(t)[v(t)-\Gamma^T(t)\hat{\Phi}(t-1)] \tag{5.89}$$

and

$$P(t)=\left[P(t-1)-\frac{P(t-1)\pi(t)\pi^T(t)P(t-1)}{\tau+\pi^T(t)P(t-1)\pi(t)}\right]\frac{1}{\tau} \tag{5.90}$$

where

$$\Gamma(t)=[-v(t-1)\quad\ldots\quad -v(t-n)\quad \varepsilon(t-1)\quad\ldots\quad \varepsilon(t-n)]^T \tag{5.91}$$

$$\varepsilon(t)=v(t)-\Gamma^T(t)\hat{\Phi}(t-1) \tag{5.92}$$

$$\pi(t) = \frac{1}{D}\Gamma(t) \tag{5.93}$$

and

$$\hat{\Phi}(t) = [\hat{c}_1 \quad \dots \quad \hat{c}_n \hat{d}_1 \quad \dots \quad \hat{d}_n] \tag{5.94}$$

where $\varepsilon(t)$ in (5.92) is an estimate of e(t) in Figure 3. Equations (5.89) through (5.93) are associated with the "refined AML" block of Figure 3. This concludes our derivation of the IVAML algorithm.

In order to improve the performance of the IVAML algorithm significantly, we suggest the following modifications:

1) Pre-filter the data vectors, instead of pre-filtering the single measurements of the input and the output as suggested by Young [30]. This has the effect of producing better filtered values, assuming that at the end of each step the estimation procedure produces a better estimate of the unknown parameters.
2) Update the pre-filters parameters, as shown in Figure 3, only when the updated filter is stable. If the updating procedure produces an unstable filter then keep the old filter parameters for the next time step. An alternative approach is to project the unstable roots of the filter polynomials into the unit circle, as suggested by Ljung [9].

Therefore the algorithm can be outlined as follows:

Part 1 (IV)

$$\hat{x}(t) = \frac{\hat{B}}{\hat{A}} u(t)$$

$$\Pi(t) = [-\hat{x}(t-1) \quad \ldots \quad -\hat{x}(t-n) \quad u(t-1) \quad \ldots \quad u(t-n)]^T$$

$$\gamma(t) = [-y(t-1) \quad \ldots \quad -y(t-n) \quad u(t-1) \quad \ldots \quad u(t-n)]^T$$

$$\pi(t) = \frac{\hat{C}}{\hat{D}\hat{A}} \Pi(t)$$

$$\Gamma(t) = \frac{\hat{C}}{\hat{D}\hat{A}} \gamma(t)$$

$$y^*(t) = \frac{\hat{C}}{\hat{D}\hat{A}} y(t)$$

then use equations (5.72) and (5.73) to update the estimates, and the covariance matrix.

Part 2 (AML)

$$\hat{v}(t) = y(t) - \frac{\hat{B}}{\hat{A}} u(t)$$

$$\hat{e}(t) = \frac{\hat{C}}{\hat{D}} \hat{v}(t)$$

$$\hat{v}^*(t) = \frac{1}{\hat{D}} \hat{v}(t)$$

$$\hat{e}^*(t) = \frac{1}{\hat{D}} \hat{e}(t)$$

$$\Psi(t) = [-\hat{v}(t-1) \quad \ldots \quad -\hat{v}(t-n) \quad \hat{e}(t-1) \quad \ldots \quad \hat{e}(t-n)]^T$$

$$\pi(t) = \frac{1}{\hat{D}} \Psi(t)$$

then use equations (5.89) and (5.90) to update the estimates, and the covariance matrix.

CHAPTER 6

THE NEW APPROACH

Introduction

In many investigations of system identification, severe restrictions on the system are necessary in order to insure the validity of the estimation or identification algorithms; e.g., the plant parameters are constrained to be time-invariant. In many cases of interest, however, the system parameters and/or plant are not constant, but are time varying and perhaps even exhibit random properties.

As mentioned before, the basic recursive least-squares algorithm is known to have optimal properties when the parameters are time invariant [9]. However, it is unsuitable for tracking time-varying parameters for several reasons. For example, consider the case where the parameters are held fixed for a long period of time and the covariance matrix approaches zero. The gain vector in equation (5.38) then approaches zero also. If the parameters now change suddenly, the identifier gives little weight to the incoming information because of the nulled gain vector, and the parameter estimates will take a relatively long time in approaching the new parameter values even if a forgetting factor is

used. Other examples can be cited from electrical power systems. Large power systems exhibit time varying and nonlinear dynamics, which violate the assumption of a linear, time-invariant system. This makes it necessary to incorporate some adaptive mechanism in the identification procedure to provide it with the ability to adjust the estimates to follow these changes. Considerable research effort has been directed towards the development of modified versions of identification algorithms for tracking time-varying parameters.

An interesting idea in this regard is the variable forgetting factor algorithm [83]. Another approach is to modify the covariance matrix directly. Probably the simplest of these algorithms involves periodic resetting of the covariance matrix to an identity matrix times a constant [53] or the covariance matrix may be modified by adding a positive semidefinite matrix to it [84]. An alternative is proposed here which appears to have certain advantages. The approach implements the notion of adaptation of the covariance matrix (or the gain vector) in a recursive identification algorithm so that both slow, fast and abrupt changes in dynamics of the system may be tracked with good accuracy [85].

The Corrector Least-Squares Method (CLS)

Consider the RLS algorithm as described by equations (5.37) through (5.39). Let

$$\delta\hat{\Phi}(t) \equiv \hat{\Phi}(t) - \hat{\Phi}(t-1) = K(t)\,[y(t) - \hat{\Phi}^T(t-1)\pi(t)] \tag{6.1}$$

or

$$\delta\hat{\Phi}(t) = K(t)\varepsilon(t) \tag{6.2}$$

In the CLS method, the sequence $\{\delta\hat{\Phi}\}$ is low-pass filtered to give the sequence $\{\beta\}$:

$$\beta(t) = \rho\beta(t-1) + (1-\rho)\,|\delta\hat{\Phi}(t)| \quad , \quad 0 < \rho < 1 \tag{6.3}$$

Also define $\chi(t)$ in terms of the components of $\beta(t)$ and $\hat{\Phi}(t)$, as follows:

$$\chi(t) = \left[\frac{|\delta\hat{\Phi}_1(t)|}{\beta_1(t)} \quad \frac{|\delta\hat{\Phi}_2(t)|}{\beta_2(t)} \quad \cdots \quad \frac{|\delta\hat{\Phi}_{na+nb}(t)|}{\beta_{na+nb}(t)} \right] \tag{6.4}$$

Then the sequence $\{\chi\}$ can be used to detect changes in the estimated value of the unknown parameters. Once any element(s) in this sequence crosses a pre-specified threshold value, and at the same time the following conditions, namely,

$$|\varepsilon(t)| > |\varepsilon(t-1)|$$

and

$$|\varepsilon(t)| > |\text{Some percentage of } y(t)|$$

are satisfied, then the corresponding diagonal term(s) of the covariance matrix can be modified to signify the uncertainty in the estimation of this particular element(s).

The amount by which the diagonal terms of the covariance matrix should change in response to the detected changes in χ can be found as follows.

Note that for m^{th} element of the $\hat{\Phi}$ vector the following relation is true:

$$|\delta\hat{\Phi}_m(t)| = |K_m(t)\varepsilon(t)| = |K_m(t)|\,|\varepsilon(t)| \tag{6.5}$$

Now let

$$|\varepsilon_{new}(t)| = |\varepsilon(t-1)| \tag{6.6}$$

Substitute equation (6.6) in equation (6.5) and solve the resulting equation for $K_m(t)$ to obtain

$$|K_{m\,new}(t)| = \frac{|\delta\hat{\Phi}_m(t)|}{|\varepsilon_{new}(t)|} \tag{6.7}$$

where

$$K_{m\,new}(t) = P_{m1}\pi_1 + P_{m2}\pi_2 + \ldots + P_{mm\,new}\pi_m + \ldots + P_{mn}\pi_n \quad , \quad n = na + nb \tag{6.8}$$

or

$$K_{m\,new}(t) = \pi_m\left[P_{mm\,new} + P_{m1}\frac{\pi_1}{\pi_m} + \ldots + P_{mn}\frac{\pi_n}{\pi_m}\right] \tag{6.9}$$

or

$$K_{m\,new}(t) = \pi_m(P_{mm\,new} + \eta) \tag{6.10}$$

where η incorporates the contributions of all off-diagonal terms.

Substitution of equation (6.10) into (6.7) gives:

$$|\pi_m|\,|P_{mm\,new}+\eta| = \frac{|\delta\hat{\Phi}_m(t)|}{|\varepsilon_{new}(t)|} \tag{6.11}$$

Then

$$|P_{mm\,new}+\eta| = \frac{|\delta\hat{\Phi}_m|}{|\varepsilon_{new}(t)\pi_m|} \tag{6.12}$$

Also we know that

$$|\Lambda+\omega| \le |\Lambda| + |\omega| \tag{6.13}$$

and this is applied in equation (6.12) to obtain:

$$|P_{mm\,new}(t)| \ge \frac{|\delta\hat{\Phi}_m(t)|}{|\varepsilon_{new}(t)\pi_m|} - |\eta| \tag{6.14}$$

Since P(t) is the covariance of a white noise sequence then ideally η should be identically equal to zero. By setting η equal to zero in equation (6.14) it follows that

$$|P_{mm\,new}(t)| \ge \frac{|\delta\hat{\Phi}_m(t)|}{|\varepsilon_{new}(t)\pi_m|} \tag{6.15}$$

The right-hand member of (6.15) can be viewed as a *corrector* term. By augmenting the diagonal term(s) of the covariance matrix by this amount, namely

$$\text{new}\,P_{mm}(t) = \text{old}\,P_{mm}(t) + \frac{|\delta\hat{\Phi}(t)|}{|\varepsilon_{new}(t)\pi_m|} \tag{6.16}$$

increases are made in the trace of the covariance matrix and in the gain vector, signifying the uncertainty in the estimates. This quantity also can be added to the corresponding term of the D matrix, in the case of the UD

factorization procedure. It should be mentioned that the addition of this quantity does not affect the positive semidefinite property of the covariance matrix, P(t).

The flow chart representation of the CLS algorithm is presented in Figure 4. Additional details on the implementation of CLS are given in Chapter 7.

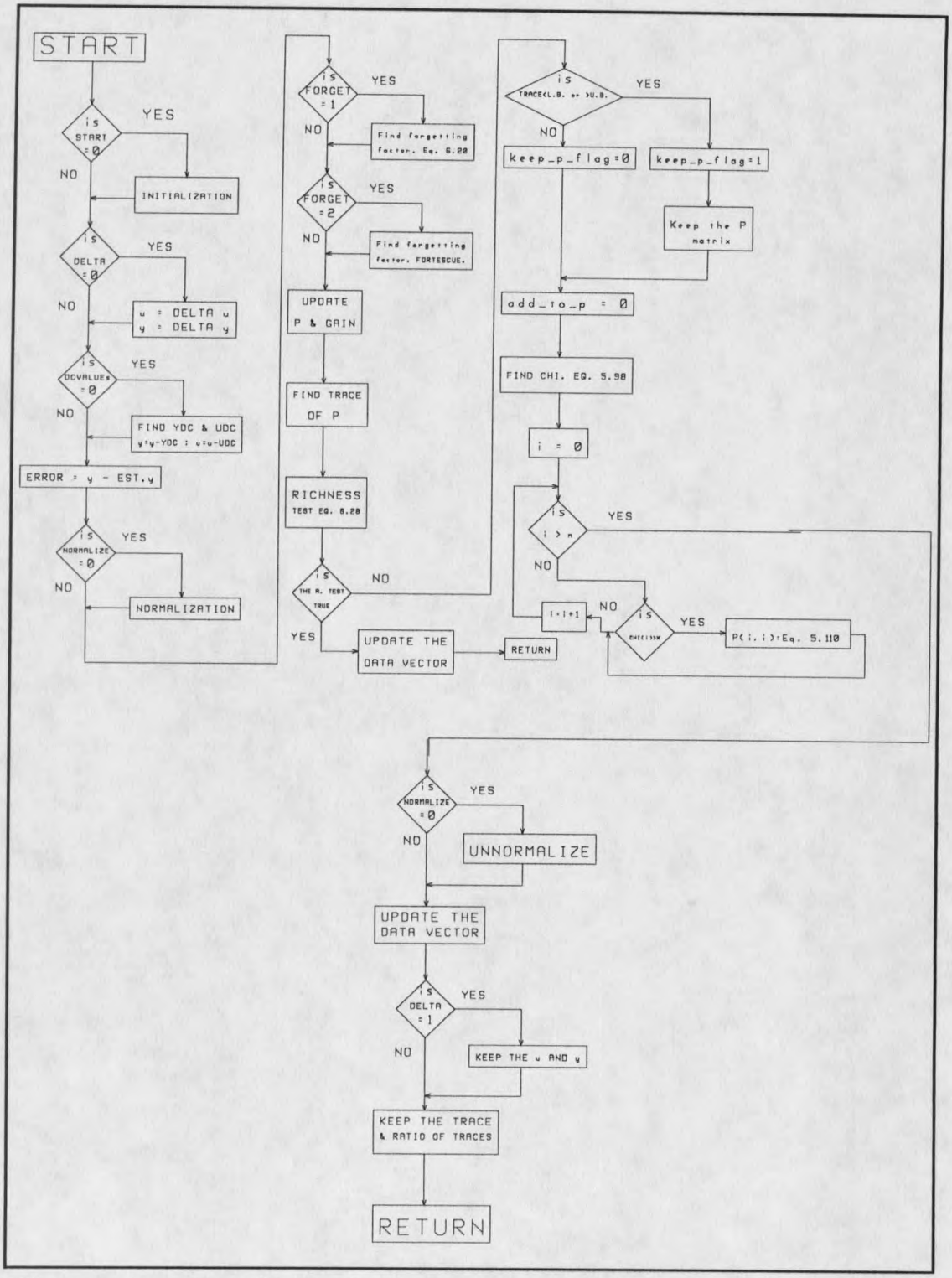

Figure 4. The flow chart representation of the CLS algorithm.

CHAPTER 7

IMPLEMENTATION OF RECURSIVE IDENTIFICATION ALGORITHMS

Since there are different ways of organizing the computations, there are a large number of different identification methods available. This chapter looks at some of the issues concerning the implementation of the previous identification schemes. Particular attention is paid to the updating procedures for the covariance matrix, the richness of the input signal used in the estimator and its effect on the parameter convergence and the convergence rate, and the use of a variable forgetting factor to improve the performance of the estimator.

SQRT Factorization

Numerical experience with least-squares based estimation in practical applications indicates that it is sensitive to the effects of computer round-off error. It is well known that this round-off error mainly affects the computations that generate the covariance matrix, $P(t)$. This problem can cause the covariance matrix to lose its positive definite property.

The SQRT method is more robust, from the numerical point of view, than direct implementation of equations (5.38) and (5.39). This method is based on a factorization of covariance matrix as

$$P = SS^T \tag{7.1}$$

where S is the square-root of P. The SQRT algorithm is reputed to be more accurate and stable than the conventional RLS algorithm as given in equations (5.38) and (5.39). The SQRT algorithm is as follows [13]:

Let $\tilde{P} = \tilde{S}\tilde{S}^T$, where $\tilde{S}$ is an upper triangular matrix. The update for the upper triangular covariance square-root factor $\tilde{S}$, and the gain vector K(t) can be obtained from the following algorithm:

$$\bar{f} = \tilde{S}^T\hat{\Phi} \quad , \quad \bar{f}^T = [\bar{f}_1 \ldots \bar{f}_n] \quad , \quad n = na + nb \tag{7.2}$$

$$\alpha_0 = r \quad , \quad R_2^T = [\tilde{S}_{11}\bar{f}_1 \, 0 \ldots 0] \tag{7.3}$$

where r is the observation error variance.

For j = 1, .., n (n=na + nb), recursively cycle through equations (7.4) to (7.8),

$$\alpha_j = \alpha_{j-1} + \bar{f}_j^2 \tag{7.4}$$

$$X_j = \left[\frac{\alpha_{j-1}}{\alpha_j}\right]^{\frac{1}{2}} \quad , \quad \gamma_j = \frac{\bar{f}_j}{X_j\alpha_j} \tag{7.5}$$

$$\hat{S}_{jj} = \tilde{S}_{jj}X_j \tag{7.6}$$

$$\hat{S}_{ij} = \tilde{S}_{ij}X_j - \gamma_j R_j(i) \quad , \quad i = 1,2,\ldots,j-1 \quad \text{and} \quad j \neq 1 \tag{7.7}$$

$$R_{j+1}(i) = R_j(i) + \bar{f}_j \tilde{S}_{ij} \quad , \quad i = 1, 2, \ldots, j \tag{7.8}$$

and

$$K(t) = \frac{R_{n+1}}{\alpha_n} \tag{7.9}$$

Equation (5.37) is used to update the estimate of the parameter vector.

Proofs and details about this algorithm can be found in [13]. FORTRAN subroutines for the SQRT algorithm are included in [86], [87], and Appendix C.

UD Factorization

Equations (5.37), (5.38), and (5.39) are one way to mechanize the recursive update of the estimates, the estimator gain, and the covariance matrix. Since these equations are not well-conditioned from a numerical point of view, one can use UD factorization of P to make the algorithm numerically more robust in the updating of the estimates, the estimator gain, and the covariance matrix. Also in many practical problems of parameter estimation there arises the problem of solving an overdetermined ill-conditioned set of algebraic equations. To circumvent this difficulty it is possible to apply the method for propagating the error covariance matrix in a square-root like form. This method is very successful in maintaining the positive semidefinite nature of the error covariance

matrix. The outstanding numerical characteristics and relative simplicity of this recursive approach led to its implementation in many practical applications.

This method is based on a factorization of the covariance matrix as

$$P(t) = U(t)D(t)U^T(t) \tag{7.10}$$

where D is a diagonal matrix, and U is an upper-triangular matrix with unity diagonal entries. This algorithm is considered to be of the square-root type since $UD^{\frac{1}{2}}$ is a covariance square root. The UD updating algorithm is more popular than the SQRT algorithm because: 1) it appears to have the accuracy characteristic of a square-root algorithm; and 2) it does not require the calculation of scalar square roots. Thus it qualifies for use in real-time applications that involve minicomputers with limited capability. The UD algorithm is as follows [13]:

Suppose that an *a priori* estimate $\hat{\Phi}(t)$ and a covariance given by $\tilde{P}(t) = \tilde{U}\tilde{D}\tilde{U}^T$, with $\tilde{U}$ unit upper triangular and $\tilde{D}$ diagonal, is to be combined with the scalar observation $y(t) = \Phi^T(t)\pi(t) + e(t), \quad E[e(t)] = 0, \quad E[e^2(t)] = r$. The gain K(t) and the updated covariance factors $\hat{U}$ and $\hat{D}$ can be obtained from the following algorithm [13]:

$$f = \tilde{U}^T\hat{\Phi} \quad , \quad f^T = [f_1 \dots f_n] \quad , \quad n = na + nb \tag{7.11}$$

$$v = \tilde{D}f \quad , \quad v_i = \tilde{d}_i f_i \quad , \quad i = 1, 2, \dots, n \tag{7.12}$$

$$\hat{d}_1 = \frac{\tilde{d}_1 r}{\alpha_1} \quad , \quad \alpha_1 = r + v_1 f_1 \quad ; \quad R_2^T = [v_1 \, 0 \ldots 0] \tag{7.13}$$

For j=2, ...,n (n=na+nb), recursively cycle through equations (7.14) to (7.17),

$$\alpha_j = \alpha_{j-1} + v_j f_j \tag{7.14}$$

$$\hat{d}_j = \frac{\tilde{d}_j \alpha_{j-1}}{\alpha_j} \tag{7.15}$$

$$\hat{u}_j = \tilde{u}_j + \gamma_j R_j \quad , \quad \gamma_j = -\frac{f_j}{\alpha_{j-1}} \tag{7.16}$$

$$R_{j+1} = R_j + v_j \tilde{u}_j \tag{7.17}$$

where

$$\tilde{U} = [\tilde{u}_1 \quad \ldots \quad \tilde{u}_n] \quad , \quad \hat{U} = [\hat{u}_1 \quad \ldots \quad \hat{u}_n] \tag{7.18}$$

and the estimator gain, K(t), is given by

$$K(t) = \frac{R_{n+1}}{\alpha_n} \tag{7.19}$$

To update the estimate of the parameter vector, equation (5.37) can be used.

Various proofs and details about this algorithm can be found in references [13], [14] and [88]. A FORTRAN subroutine for the UD algorithm is included in [87], and a Pascal program for it is contained in [89]. These code listings in [87] and [89] contain minor typographical errors which are corrected in Appendix B.

Covariance Blowup

If the forgetting factor τ is less than one and constant in equation (5.39), P(t) can become very large, i.e. covariance "blowup" or "windup" can occur. This problem may be overcome by periodic or regular resetting of the covariance matrix as suggested by Goodwin [53]. Also this problem may be eliminated by calculating the forgetting factor before each iteration to maintain a constant trace for the covariance matrix, as follows:

$$\tau(t) = \frac{1-B}{2} + \frac{[TrP(t-1)^2(B+1)^2 - 4TrP(t-1)TrA]^{\frac{1}{2}}}{2TrP(t-1)} \tag{7.20}$$

where

$$B = \pi(t)^T P(t-1)\pi(t) \tag{7.21a}$$

and

$$A = P(t-1)\pi(t)\pi^T(t)P(t-1) \tag{7.21b}$$

An alternative approach suggested by Fortescue et al. [83] may be used for deterministic systems. Fortescue suggested the use of a time-varying forgetting factor, such that the estimation is always based on the same amount of information for all time. This means that the value of the forgetting factor will be decreased when the new measurements contain good information, and will be close to unity if the new measurements contain no new information. Sripada and Fisher [8] show that covariance windup and

therefore parameter drifting can occur when this variable forgetting factor is used in certain stochastic cases. The ratio-ratio test developed later in this chapter also eliminates covariance "blowup".

Persistent Excitation

It is intuitively clear that the accuracy of the parameter estimates will depend upon the signal input. This leads us to the concept of a persistently exciting input signal, which is a well-known prerequisite for achieving parameter convergence in identification schemes. This concept was first introduced by Astrom and Bohlin [90]. Loosely speaking, a persistently exciting input is one which is sufficiently varied or rich so that the resulting input-output data provides enough information about the characteristics of the system on which it acts. The definition of a persistently exciting signal is as follows:

Definition 6.1 : A square summable signal u is called persistently exciting of order n if the following matrix C_n given by

$$C_n = \begin{bmatrix} c(0) & c(1) & \dots & c(n-1) \\ c(1) & c(0) & \dots & c(n-2) \\ \dots & \dots & \dots & \dots \\ c(n-1) & c(n-2) & \dots & c(0) \end{bmatrix} \tag{7.22}$$

is positive definite, where

$$c(k) = \lim_{t \to \infty} \frac{1}{t} \sum_{i=1}^{t} u(i)u(i-k) \tag{7.23}$$

It is clear that if u is a stochastic process with full rank innovations, then it is persistently exciting of any finite order.

It should be mentioned that, if the input is generated entirely by feedback, as in many adaptive control schemes, it is difficult to guarantee that it will be persistently exciting. The important point which should be considered, in these situations is that the updating of parameter estimates should be carried out only when there is sufficient excitation in the input signal. A variety of detection procedures have been devised which can be used in the control loop to ensure this. For example, Sripada and Fisher [8] suggest that the parameter estimation be turned off when the condition number of the covariance matrix, C{P(t)}, is greater than some user prespecified number. At the very least, the parameter estimation should be terminated when the magnitude of the known input signal decreases below a given threshold for a significant period of time.

An alternative approach presented here seems to have some advantages. The approach is based on a ratio involving recent values of the trace of the covariance matrix; we call it the ratio-ratio test.

Ratio-Ratio Test

Consider equation (5.20). R(t) in this equation is known as the information matrix, and contains the information about the past measurements. This equation tells us that a decrease in |R(t)| produces an increase in |P(t)| and vise verse. Now consider the case when an unknown plant is operating near a steady-state condition, or when the input signal is not a persistently exciting signal. Then the |R(t)| becomes smaller and smaller as time increases, causing the eigenvalues of R(t) to spread more and more, and making R(t) illconditioned [47]. This can be seen from simulation example number 3 in Chapter 9. This causes the covariance matrix, P(t), to become larger and larger as time progresses.

Now consider equation (5.39); experience with this equation shows that the second term in the bracket approaches zero as the information in R(t) decreases. This means that as information in R(t) decreases the rows of the covariance matrix becomes orthogonal to the data vector, and this is especially true when the input signal is not persistently exciting. As the second term in the bracket of equation (5.39) approaches zero, this equation becomes

$$P(t) \approx \frac{P(t-1)}{\tau} \tag{7.24}$$

Therefore P(t) is constantly scaled by the forgetting factor which is less than one, and will grow exponentially resulting

in covariance "blowup". This problem in adaptive control can cause temporary instability or complete instability; an example of this is given in simulation problem number 9 in Chapter 9. Therefore, it is of particular importance to limit the update of the parameters to instants when sufficient new information has been obtained.

In order to avoid such problems, several approaches, like putting an upper bound on the diagonal elements of P(t), their trace or their condition number, have been suggested in the literature. Although these procedures have had some success, the particular bounds to choose are not always clear, and in some cases the procedures are computationaly expensive, especially those that require the computation of the condition number of a matrix. The ratio-ratio test as given below overrides these difficulties.

Consider equation (7.24). Taking the trace of both sides produces

$$TrP(t) \approx \frac{TrP(t-1)}{\tau} \tag{7.25}$$

or

$$\frac{TrP(t-1)}{TrP(t)} \approx \tau \tag{7.26}$$

This equation tells us that the ratio of the trace of the covariance matrix at time t-1 to the trace of the covariance matrix at t is approximately equal to the forgetting factor.

We go one step further and find the ratio of the ratio of traces at time t-1 to ratio of traces at time t; this produces the following

$$\frac{\frac{TrP(t-2)}{TrP(t-1)}}{\frac{TrP(t-1)}{TrP(t)}} \approx \frac{\tau}{\tau} = 1.0 \tag{7.27}$$

or

$$\frac{TrP(t-2)*TrP(t)}{(TrP(t-1))^2} \approx 1.0 \tag{7.28}$$

Equation (7.28) can be used to check the measured data for persistent excitation. Therefore, if the ratio-ratio test produces a value close to unity over several successive steps, then the estimation should be stopped. The estimation procedure can be started again when this ratio-ratio test produces values significantly different than unity. It should be mentioned that the covariance matrix is held at the value where the estimation stopped; only a "trial" update of P(t) is made in order to conduct the test to determine if the parameter update should be restarted. Note that this test produces the same result if the trace of the covariance matrix stays at a constant value, which is a false indicator, and therefore in these cases the estimation should be continued.

Simulation problem number 3 in Chapter 9 compares several of the procedures which can be used to check the data vector for persistent excitation. As can be seen from this sim-

ulation the ratio-ratio test gives us essentially the same information as other more complicated and more computationaly expensive methods; therefore the advantage of this procedure over the other ones should be clear.

Normalization

In order to make the algorithm numerically robust normalization of the data vector and the output of the system can be carried out before use in the estimator as described by equations (5.37) through (5.39). The regressor or data vector and the new plant output measurement are normalized by

$$n(t) = \max[\| \pi(t) \|_{\infty}, | y |] \tag{7.29}$$

therefore,

$$\pi_n(t) = \frac{\pi(t)}{n(t)} \tag{7.30}$$

and

$$y_n = \frac{y}{n(t)} \tag{7.31}$$

which results in all the elements of $\pi_n(t)$ and y being less than or equal to 1 in magnitude.

Data Scaling

Like normalization, data scaling makes the algorithm numerically more robust. The regressor vector and the covariance matrix are transformed or scaled by a diagonal scaling matrix S,

$$P_s(t-1) = S(t-1)P(t-1)S(t-1) \tag{7.32}$$

$$\pi_s(t) = S(t-1)^{-1}\pi(t) \tag{7.33}$$

before use in the estimator. The diagonal scaling matrix S(t-1) will be found in such a way to minimize C{SQ} where C{SQ} is the condition number of SQ and Q is square root or Cholesky factor of P, i.e.,

$$P = QQ^T \tag{7.34}$$

The diagonal matrix S that minimizes C{SQ} is given by choosing $S_{ii} = 1/Z_i$ where Z_i are the absolute row-sums of Q. The row-sums of SQ are then all equal. For proof see Nobel [91]. This scaling procedure was implemented in the ILS code [8], but is not included in CLS.

CHAPTER 8

ADAPTIVE CONTROL

Up to now we have only discussed how to estimate an unknown parameter vector $\Phi(t)$ based on input-output measurements. We shall now show how to use these estimates for state estimation and optimal control of stochastic systems. The general area of on-line control system design for imperfectly known (and possible time-varying) systems is known as *adaptive control*. The self-adaptive control system must react or adapt itself to changes in its environment. Such a system must therefore continuously measure the characteristics of the inputs and outputs of the control system, and of the process under control, and on the basis of these measurements, adjust the overall system towards some previously defined operating conditions or characteristics.

The techniques we shall discuss in this chapter are appropriate for the optimal control of physical systems. To be precise, in this chapter we want a general quadratic index to have a minimum value.

This chapter is devoted mainly to the type of adaptive control in which parameter estimation methods and optimal control algorithms are incorporated in an iterative manner as the process unfolds.

Optimal Control

Optimal control is concerned with finding the optimal input control sequence which changes the state of a system in order to achieve a desired objective. The objective is often expressed in a "performance criterion" which is a scalar measure of performance but which may be a function of all the state variables and control variables of the system. The optimal input control sequence will be that which causes the performance function to be minimized, normally under conditions which enforce constraints on the control and the state variables. The selection of the performance criterion, or index, is thus basic to the concept of optimal control.

To design an optimal control it is necessary to specify a performance criterion. A large class of optimal control design has the objective of minimizing a performance criterion which is of the form

$$J = \sum_{k=0}^{N-1} \chi[x(k+1), u(k), k] \tag{8.1}$$

where $\chi[x(k+1), u(k), k]$ is a differentiable scalar function.

The minimization of J is subject to the equality constraint of

$$x(k+1) = \mathrm{X}[x(k), u(k), k] \tag{8.2}$$

which is recognized as the state equation of the system, as well as other equality and inequality constraints.

For the purpose of this chapter let the state equation of the system be described by

$$x(k+1) = G(k)x(k) + H(k)u(k) \tag{8.3}$$

where x at time i is given. The dimension of x is n, while G(k) and H(k) are given real matrices of appropriate dimensions. The design objective is to find $u_{opt}(k)$ on the time interval [i,N-1] so that the performance function

$$J_N = \frac{1}{2}x^T(N)Sx(N) + \frac{1}{2}\sum_{k=i}^{N-1}[x^T(k)Qx(k) + u^T(k)Ru(k)] \tag{8.4}$$

is minimized. Q and S are given positive semi-definite matrices, and R is a given positive-definite matrix. This is the discrete *linear quadratic Regulator* (LQR) problem. Minimization of (8.4) subjected to the constraint (8.3) results in the well-known optimal control signal, Ogata [94], of the form

$$u_{opt}(k) = -K(k)x(k) \tag{8.5}$$

where

$$K(k) = [R + H(k)^T P(k+1)H(k)]^{-1} H(k)^T P(k+1)G(k) \tag{8.6}$$

and

$$P(k) = Q + G(k)^T P(k+1) G(k) - G(k)^T P(k+1) H(k) [R + H(k)^T P(k+1) H(k)]^{-1} H(k)^T P(k+1) G(k) \quad (8.7a)$$

or

$$P(k) = Q + G^T(k) P(k+1) [G(k) - H(k) K(k)] \quad (8.7b)$$

with boundary condition of

$$P(N) = S \quad (8.8)$$

The inclusion of the variable k as the argument of matrices G(k) and H(k), in equations (8.3), (8.6), and (8.7) implies that these matrices are time-varing. In the case when these matrices are assumed to be constant, or time-invariant, then the argument k can be removed.

The Riccati equation given in equation (8.7) is only one of many equivalent forms which satisfy the optimal linear regulator design. In general, a particular form of the Riccati equation is best suited for a given purpose.

Steady-State Optimal Control

For an infinite number of stages, N=∞, the performance index is

$$J = \frac{1}{2} \sum_{k=0}^{\infty} [x^T(k) Q x(k) + u^T(k) R u(k)] \quad (8.9)$$

In this case the terminal cost is eliminated, since as N approaches infinity, the final state x(N) should approach

the zero equilibrium state, so that the "S term" in equation (8.4) is no longer necessary. It is also assumed in this case that G and H of (8.3) approach constant matrices.

The following conditions are sufficient for the existence of a unique solution to the steady-state linear regulator problem:

1) The system (8.3) should be completely controllable by state feedback; and
2) the system should be completely observable.

The terms "completely controllable" and "completely observable" have precise definition which are well documented in control literature.

The solution to the steady-state linear regulator problem can be obtained by letting $N \to \infty$, in which case the Riccati gain matrix K(k) becomes a constant matrix, i.e.,

$$\lim_{k \to \infty} K(k) = K \tag{8.10}$$

Replacing K(k) and P(k) by K and P, respectively, in equations (8.6) and (8.7), we have the steady-state Riccati equation

$$P = Q + G^T P G - G^T P H [R + H^T P H]^{-1} H^T P G \tag{8.11}$$

and the gain equation

$$K = [R + H^T P H]^{-1} H^T P G \tag{8.12}$$

Equation (8.11) is often referred to as the algebraic Riccati equation (ARE).

One difficulty in the physical implementation of the optimal control with state-variable feedback is that time is required for the computer to compute $u_{opt}(k)$ from the state vector $x(k)$ so that time delay is usually inevitable. Another difficulty lies in the fact that in practice not all the state variables in a physical process are accessible. Usually, only the inputs and the outputs of the process are directly measurable. Therefore, an observer or an estimator is needed to compute or estimate the state variables from the inputs and the outputs measurements; this is possible if the system is observable.

Self-Tuning Regulator (STR)

In the previous section on LQR control design, the assumption was that the mathematical model which adequately describes the behavior of the system is precisely known. This assumption is not valid in many applications. In this section we consider a self-tuning regulator, which combines a recursive identification algorithm and a control law based on the updated model parameters, as shown in Figure 5. The literature on adaptive control and self-tuning control is extensive. Excellent survey papers by Astrom [3], [92] give an overview of the subject and contain many references.

Adaptive control and self-tuning regulator concepts are in rapid development; for example, recent developments relevant to power systems are considered by Pierre [93], [67] through [69], Pierre and Smith [70], Smith, et al. [71] through [73], and Smith [74].

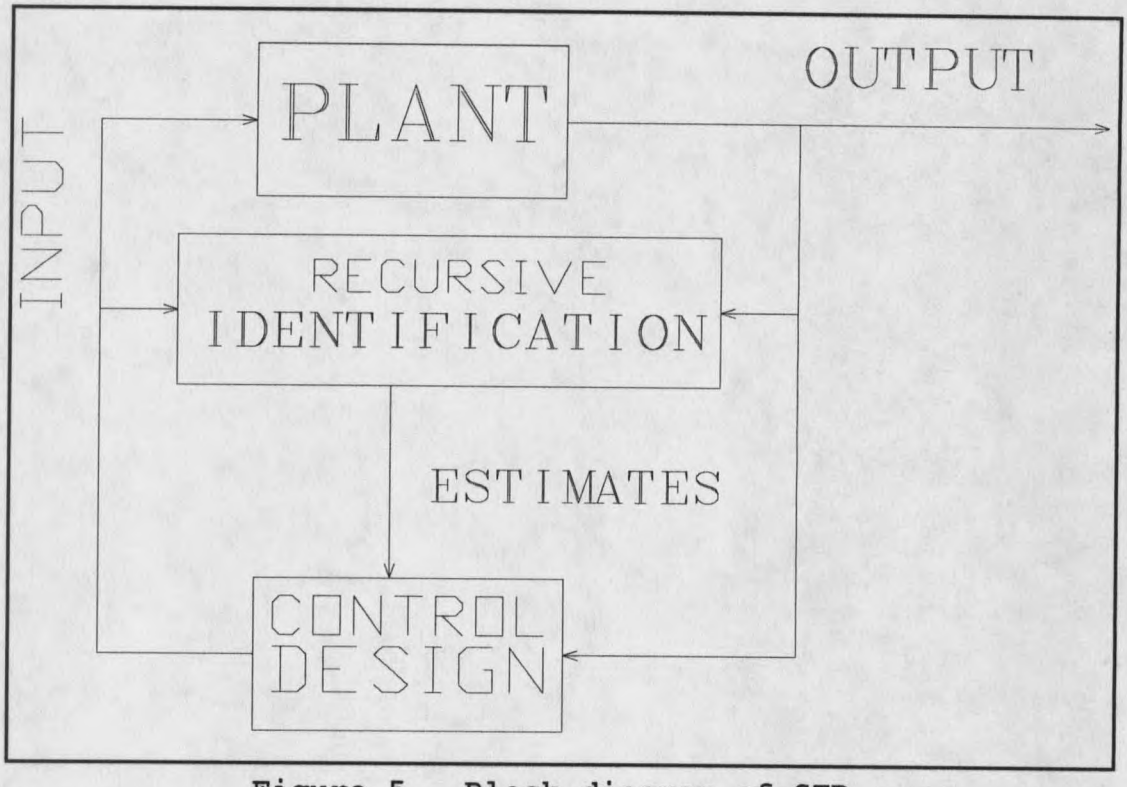

Figure 5. Block diagram of STR.

In designing a self-tuning regulator one has to choose:

1) A class of models to represent the system under consideration. This can be accomplished by using equations (4.3) and (4.4) to model the unknown system.
2) An estimation procedure for the identification task. The identification algorithms presented in Chapters 5 and 6 can be used for this propose.

3) And a control algorithm. This can be the LQR regulator as described in the previous sections.

CHAPTER 9

SIMULATIONS

This chapter presents a collection of simulation results for several identification algorithms RLS, IVAML, ILS, and CLS. Each of these algorithms was implemented in FORTRAN on a VAX/VMS computer with double precision. To show the performance and the stability of these identification algorithms, various dynamic characteristics were used to model the unknown plant dynamics operating in open loop or closed loop configurations. For the closed loop configuration, we used the self-tuning adaptive control strategy to stabilize the unstable system. In all cases except Examples 7 through 10, the sampling period used is 0.1 sec.

Example 1

This simple example demonstrates the performance of the IVAML and the RLS algorithms when the disturbance added to the output of the system is correlated to the actual output of the system. In particular, we wish to compare them in terms of the rate of the convergence and the quality of the estimates in presence of a correlated disturbance. Consider the following second-order oscillatory process:

$$y(t) = \frac{1.0z^{-1} + 0.5z^{-2}}{1.0 - 1.5z^{-1} + 0.7z^{-2}} u(t) + v(t)$$

where

$$v(t) = \frac{1.0 - 0.189z^{-1}}{1.0 - 1.027z^{-1} + 0.264z^{-2}} e(t)$$

$$e(t) = 10\sin(5t)$$

and

$u(t) = \pm 1.0$, where + or - is chosen at random.

The coefficients of the above system were estimated using IVAML as well as the RLS method. The results of this simulation are presented in Figures 6 through 14. As can be seen from the simulation results, the IVAML method is far more accurate than the RLS method, especially when disturbances are correlated with the output of the system. The results from the RLS method shows a biased estimate in the unknown parameters; this is a direct consequence of the disturbance model assumed in the derivation of the algorithm (see Table 1)..

The measured and actual system output are shown in Figures 6 and 7, respectively. The estimates of the actual output obtained by the IVAML and the RLS methods are presented in Figures 8 and 9, respectively. The input sequence is shown in Figure 14. The estimates of the unknown a parameters by using the IVAML and the RLS methods are presented in Figures

10 and 11, respectively. The estimates of the unknown b coefficients by using the IVAML and the RLS methods are shown in Figures 12 and 13, respectively.

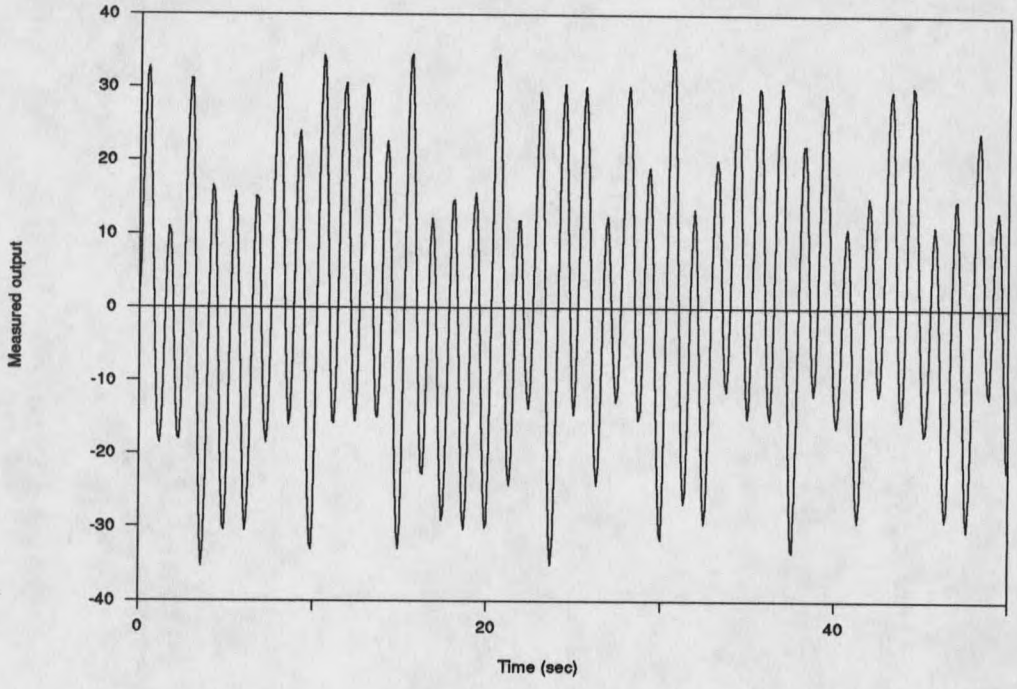

Figure 6. Measured output.

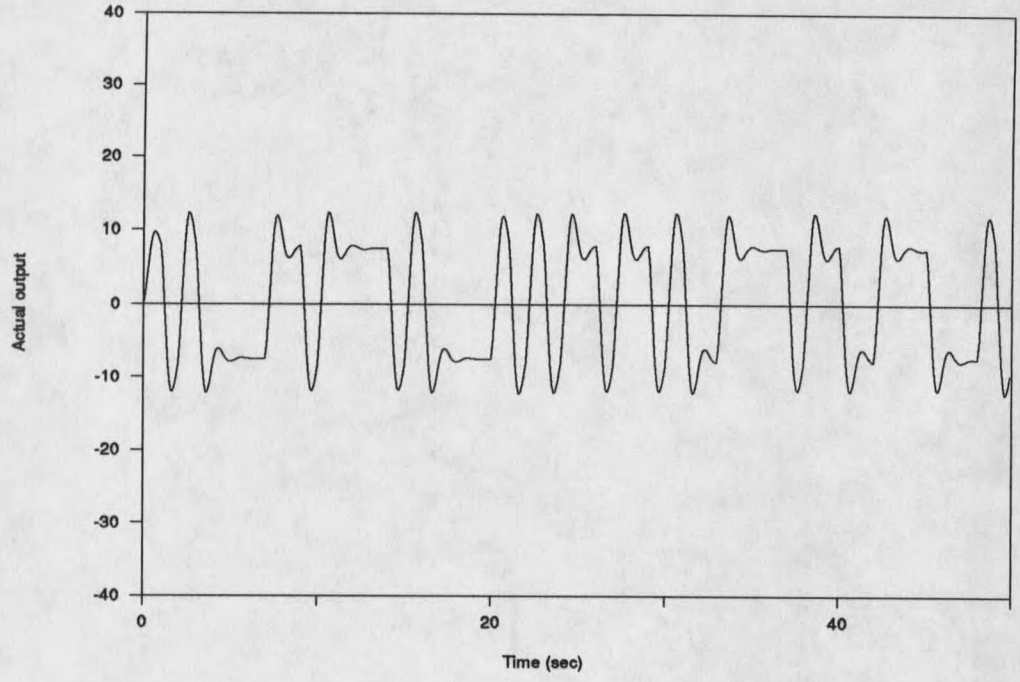

Figure 7. Actual output.

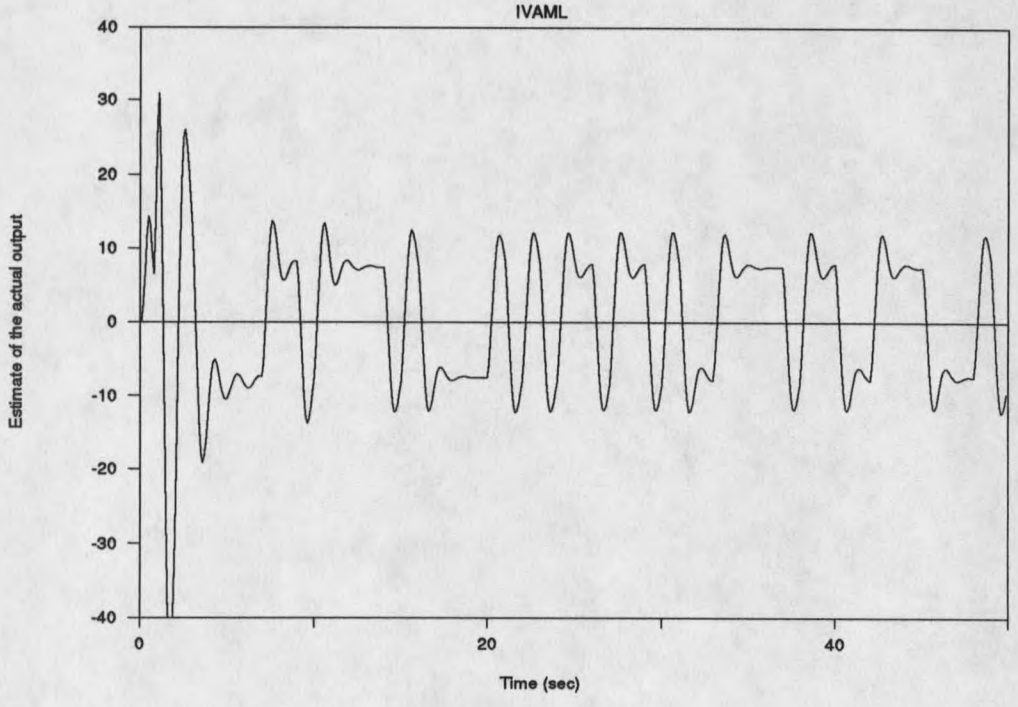

Figure 8. Estimate of the actual output (IVAML).

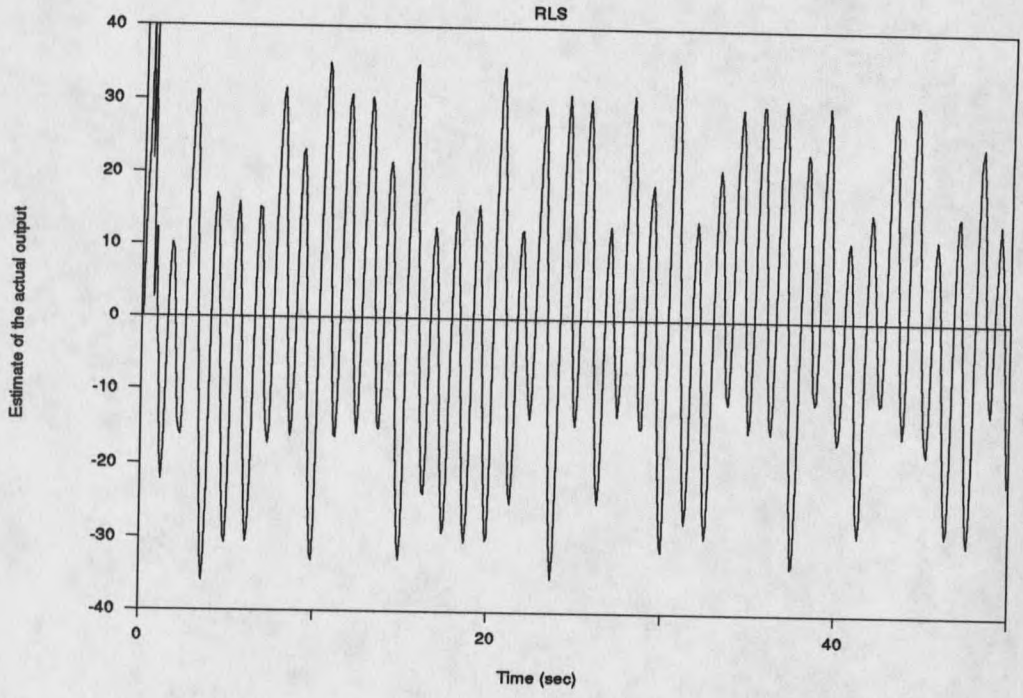

Figure 9. Estimate of the actual output (RLS).

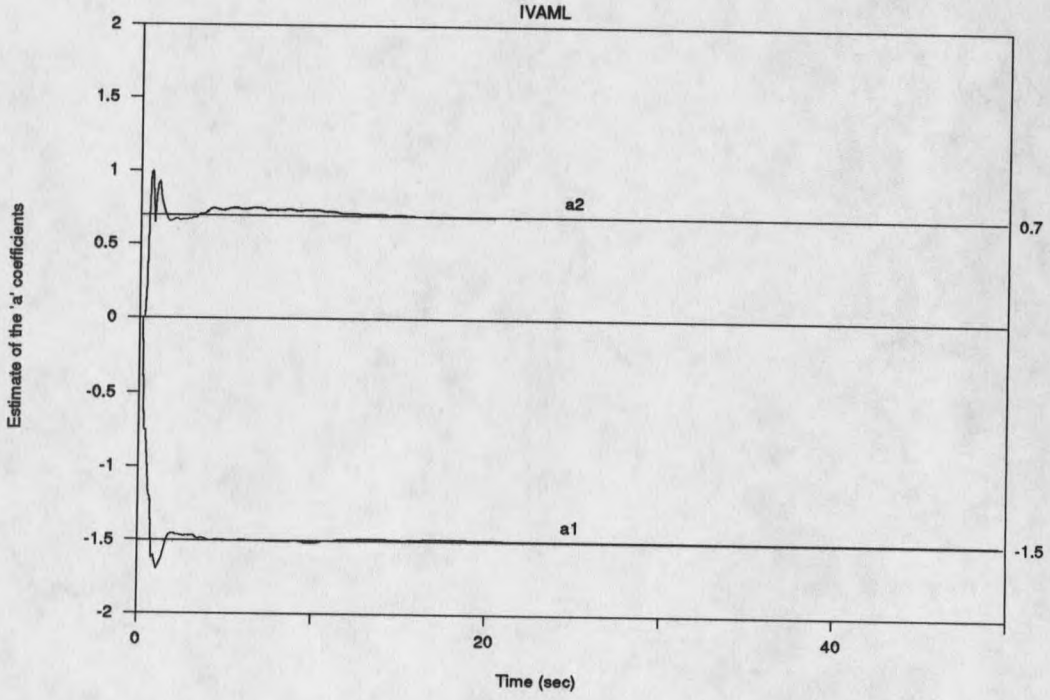

Figure 10. Estimate of the a coefficients (IVAML).

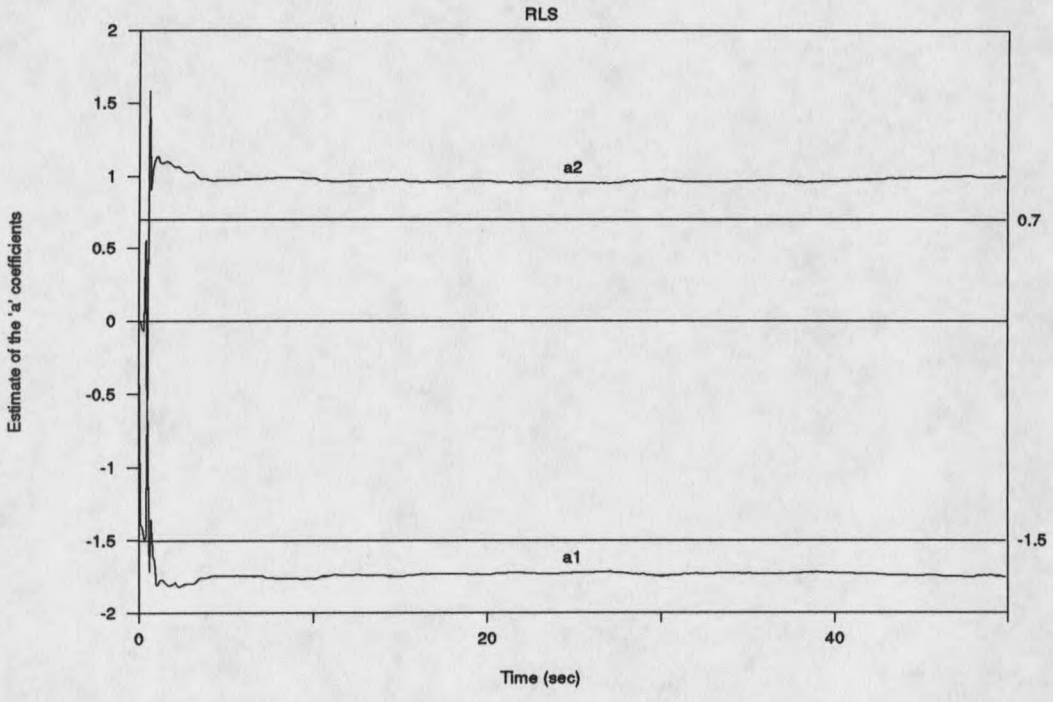

Figure 11. Estimate of the a coefficients (RLS).

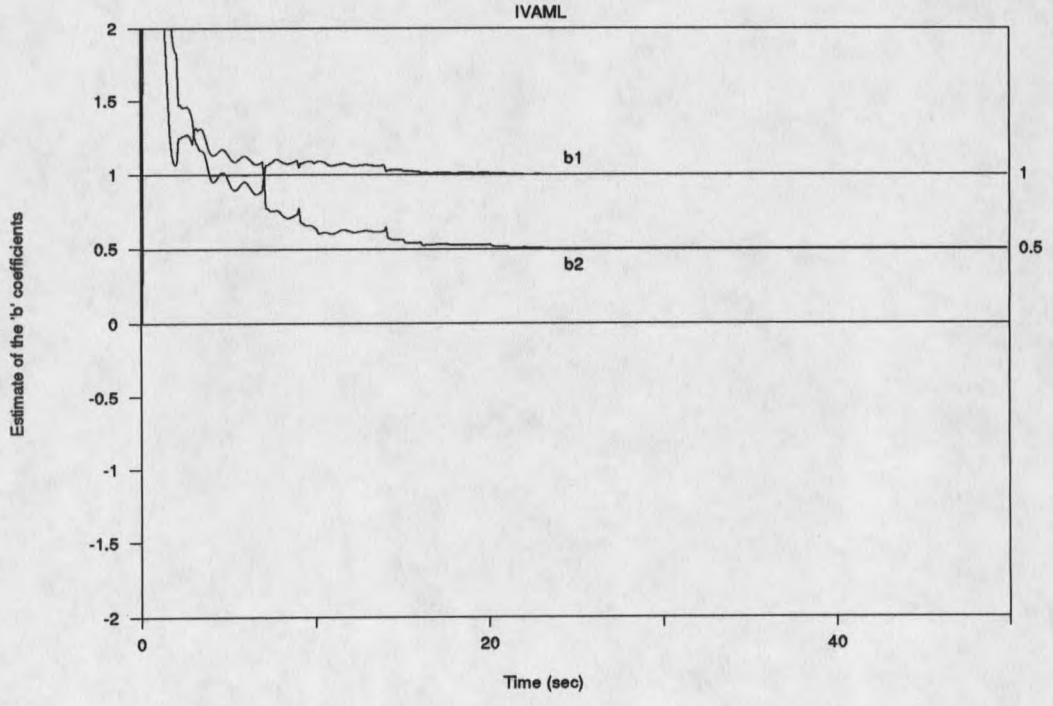

Figure 12. Estimate of the b coefficients (IVAML).

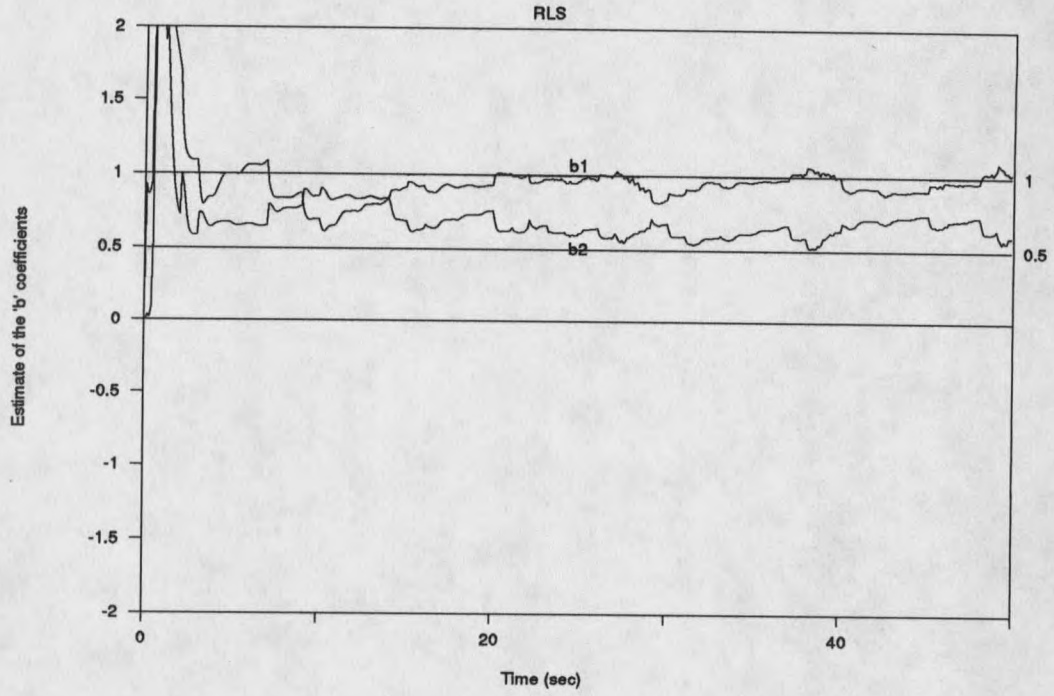

Figure 13. Estimate of the b coefficients (RLS).

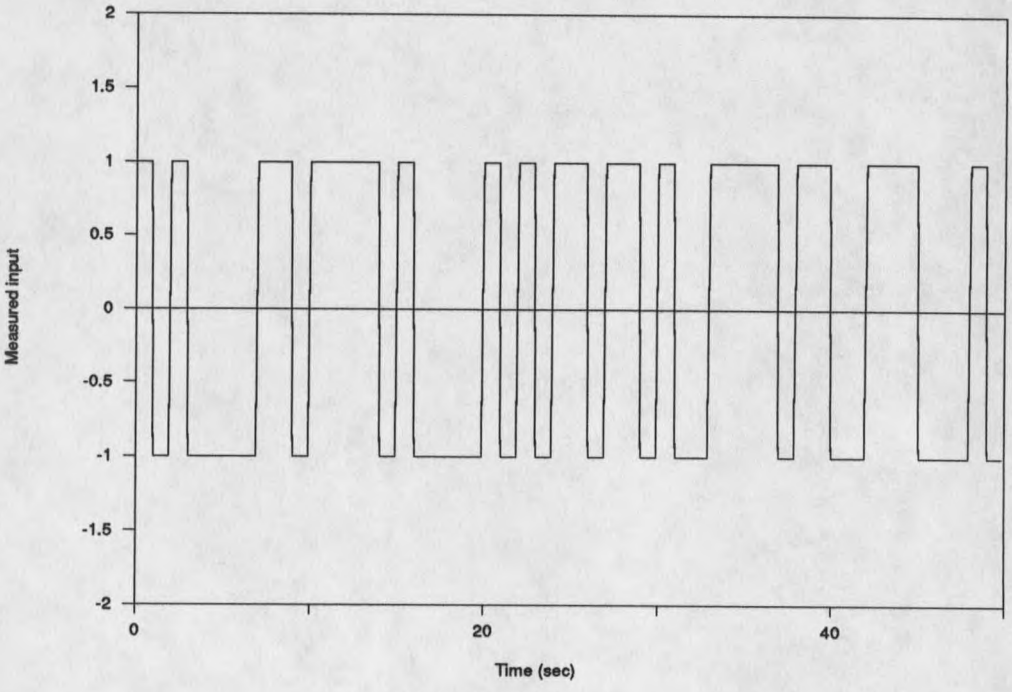

Figure 14. Input sequence.

Example 2

In this example, we present computer-simulation results to illustrate the problem of covariance "windup" or "blowup", which causes the parameters to drift from their nominal values. This problem has a direct connection with the richness of the measured data. This problem can be overcome by testing the measured data for richness and by suspending the parameter estimates when appropriate. A variety of detection procedures have been devised which can be used in the estimation procedure. In the following example we will present the results of several of these detection procedures as well as the results of the ratio-ratio procedure proposed in Chapter 7. In particular we will compare RLS, ILS, and CLS algorithms.

Consider the following second-order oscillatory process:

$$y(t)=\frac{1.0z^{-1}+0.5z^{-2}}{1.0-1.5z^{-1}+0.7z^{-2}}u(t)+\frac{1.0}{1.0-1.5z^{-1}+0.7z^{-2}}e(t)$$

where

$$u(t)=\begin{cases}\pm 1.0 & 0\le t\le 5\\ 0.0 & t>5\end{cases}$$

and

$e(t)$ is Gaussian white noise with $\sigma^2=1.0$.

The coefficients of the above system were estimated using RLS, ILS as well as the CLS method. The results of this simulation are presented in Figures 15 through 25. As can

be seen from the simulation results, the parameter estimates obtained by all of the above procedures converge rapidly to some reasonable values during the initial time interval (i.e., from 0 to 5 sec). But after 5 sec, with the input held at zero, covariance blowup tends to occur, which is due to poor excitation of the measured data, and causes the estimates obtained by the RLS procedure to drift from their previously obtained reasonable values. This parameter drift is due to the presence of the noise in the system.

The measured system output and input are shown in Figures 15 and 16, respectively. The estimates of the unknown a parameters obtained by using the RLS, ILS and CLS methods are presented in Figures 17, 18 and 19, respectively. The estimates of the unknown b coefficients obtained by using the RLS, ILS, and CLS methods are shown in Figures 20, 21, and 22, respectively. The trace of the covariance matrix obtained by each of these methods are presented in Figures 23 through 25. The ILS method uses the condition number of the covariance matrix to detect the richness of the data. The CLS method uses the procedure presented in Chapter 7 to detect the richness of the measured data.

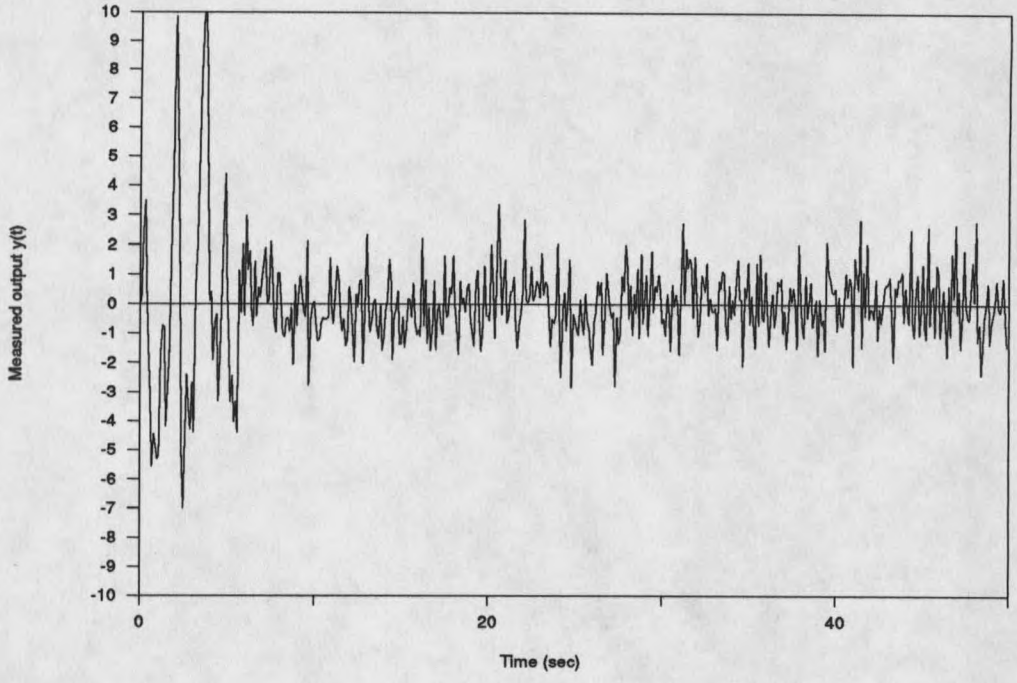

Figure 15. Measured output.

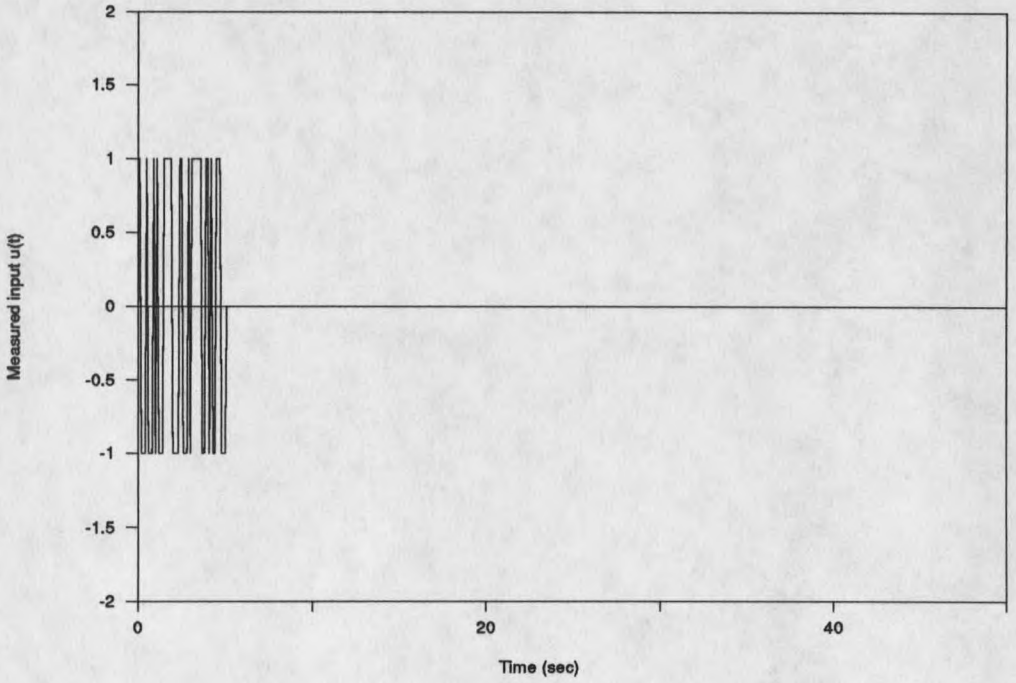

Figure 16. Measured input.

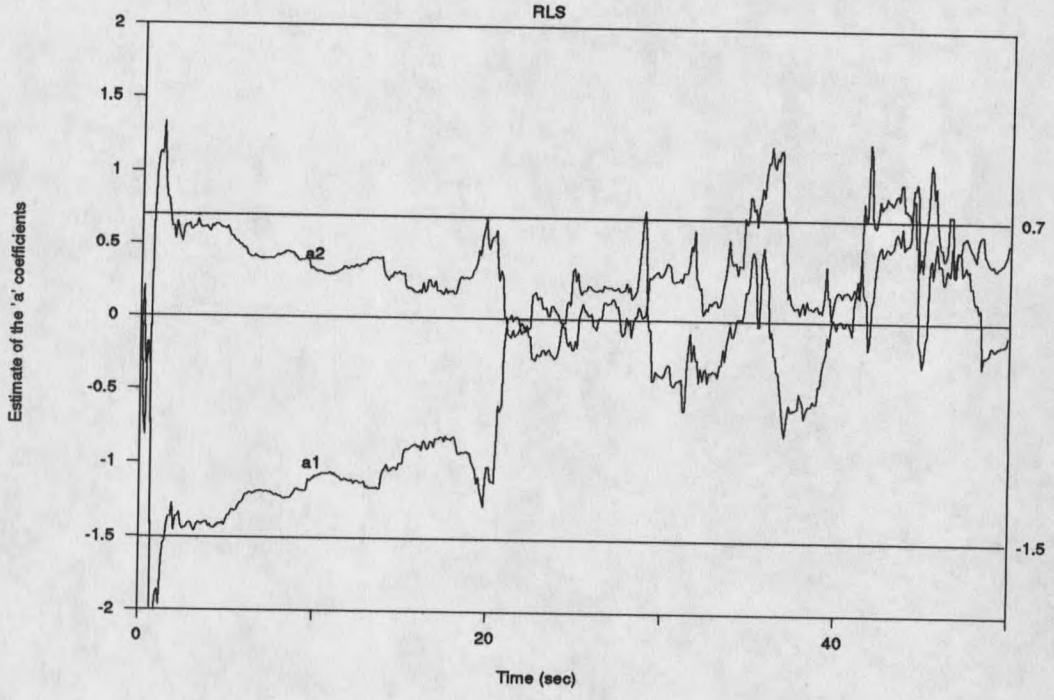

Figure 17. Estimate of the a coefficients (RLS).

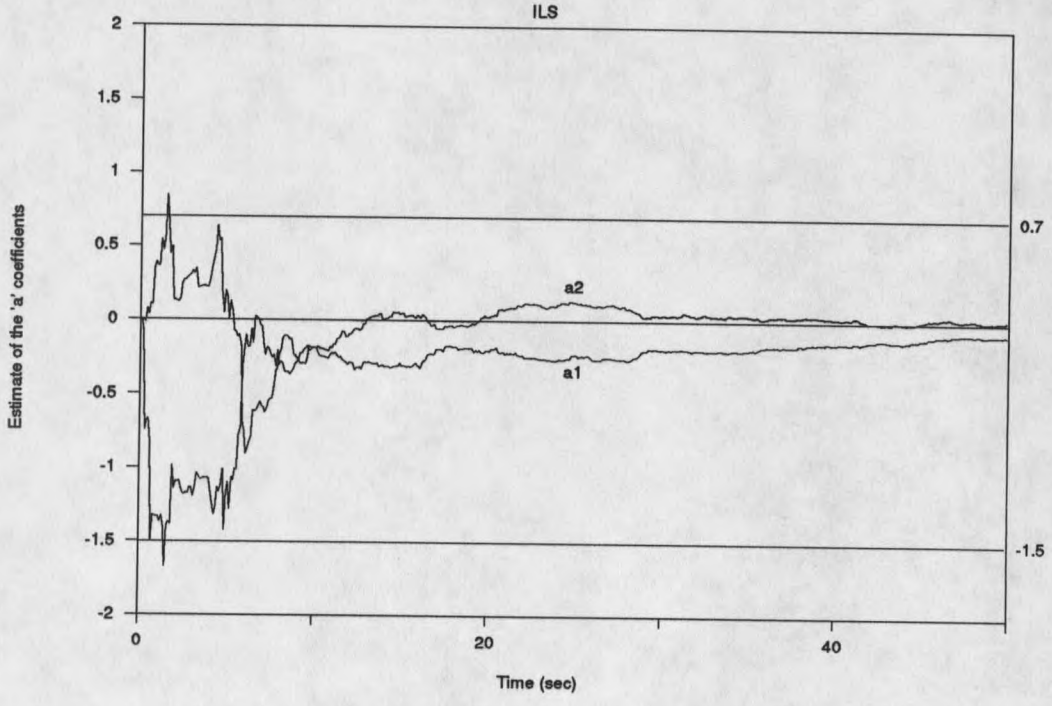

Figure 18. Estimate of the a coefficients (ILS).

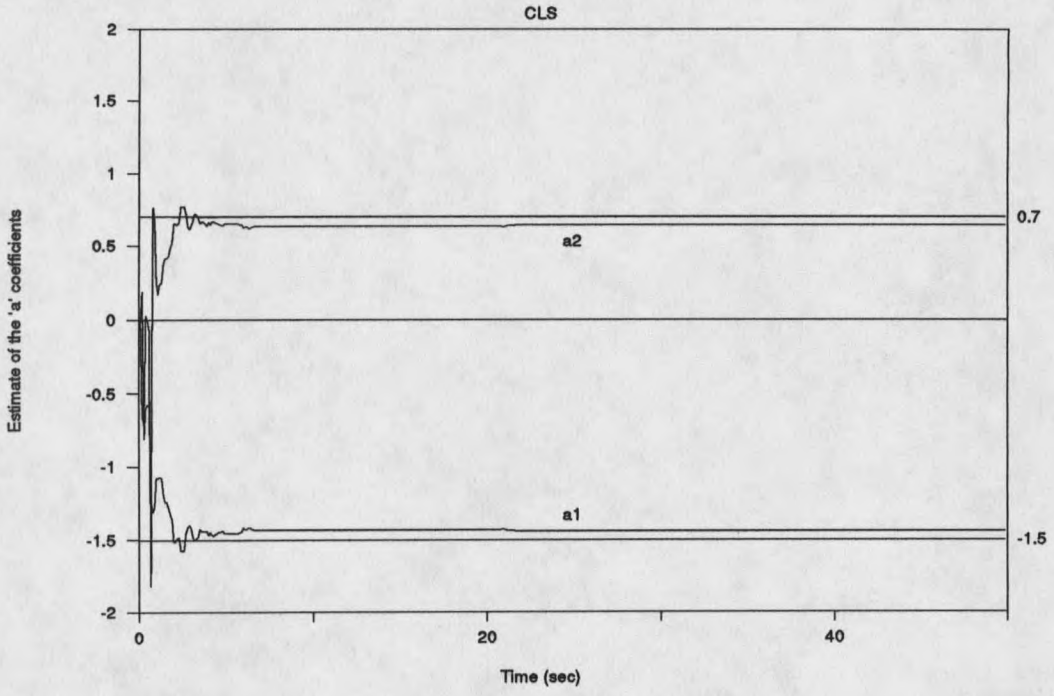

Figure 19. Estimate of the a coefficients (CLS).

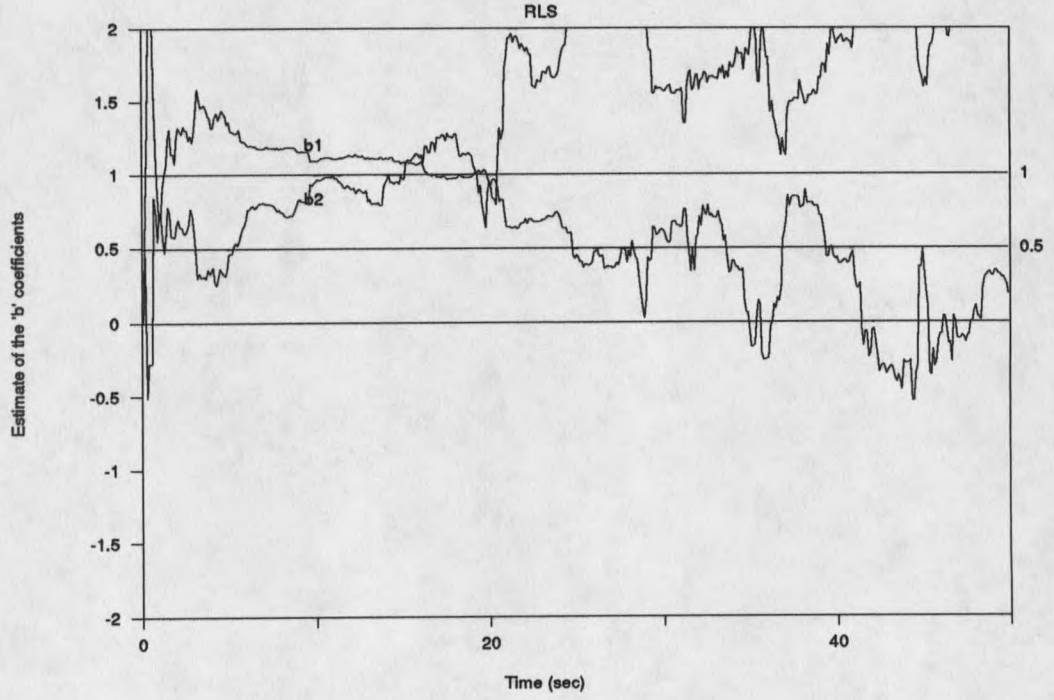

Figure 20. Estimate of the b coefficients (RLS).

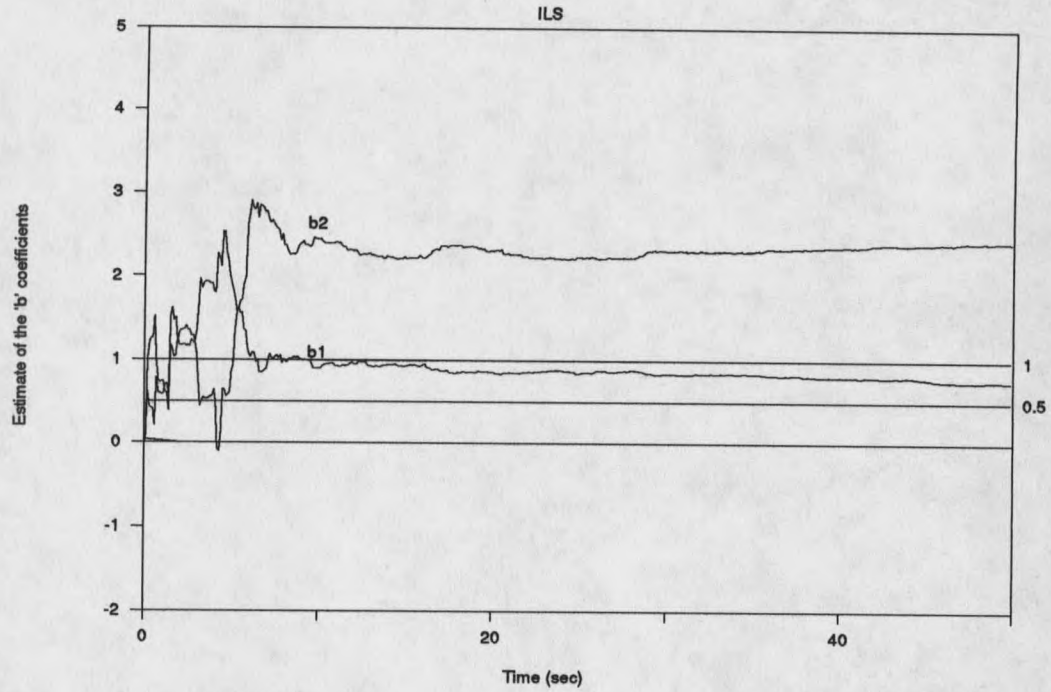

Figure 21. Estimate of the b coefficients (ILS).

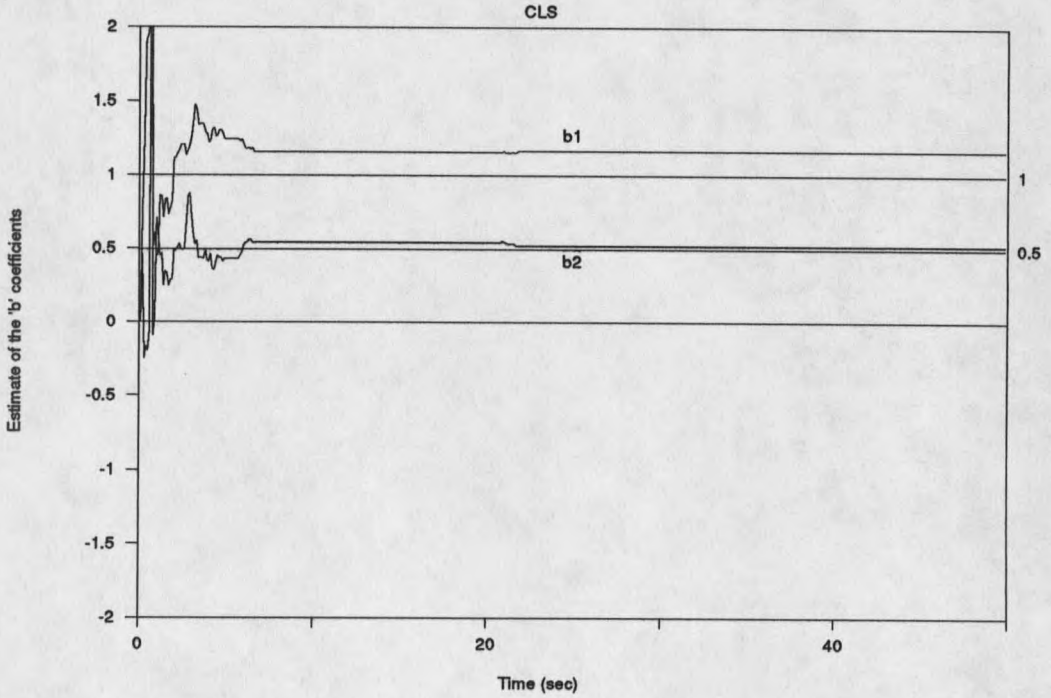

Figure 22. Estimate of the b coefficients (CLS).

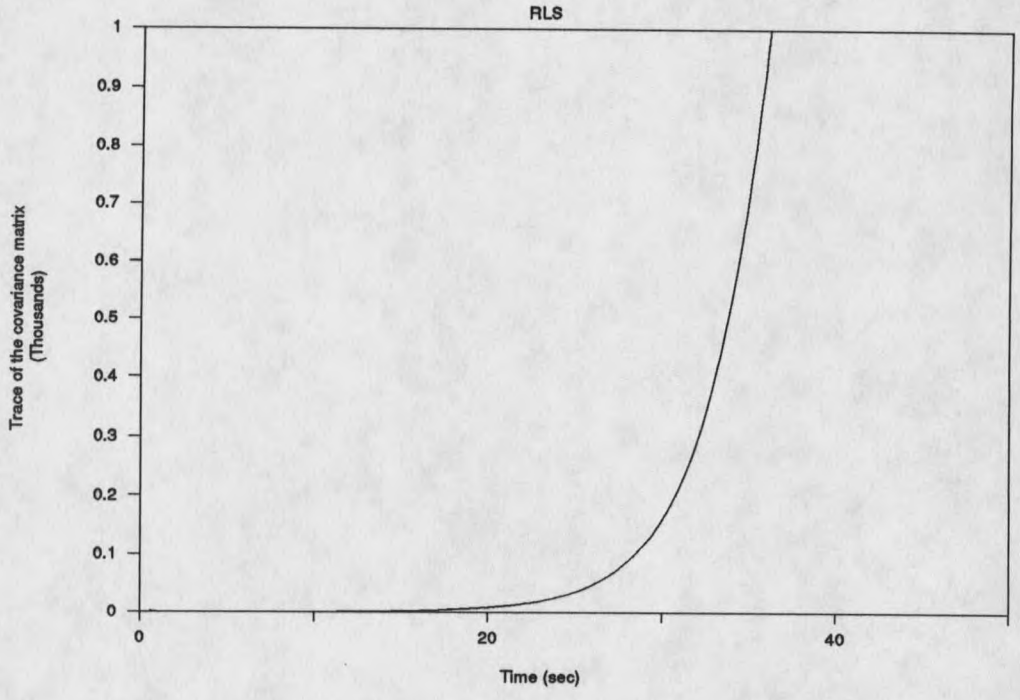

Figure 23. Trace of the covariance matrix (RLS).

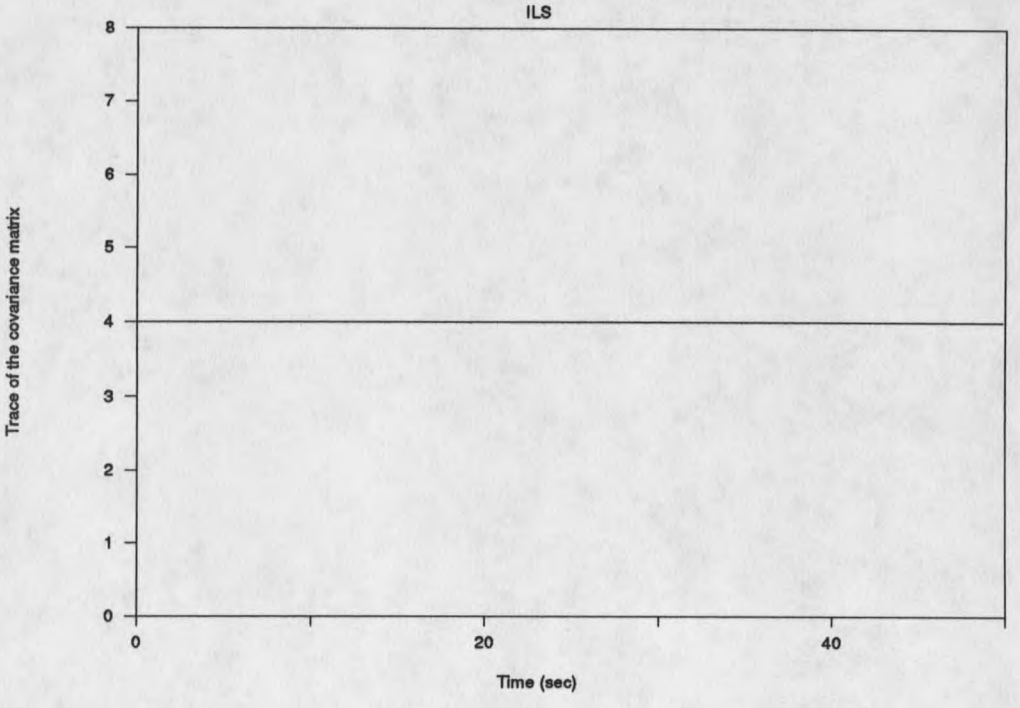

Figure 24. Trace of the covariance matrix (ILS).

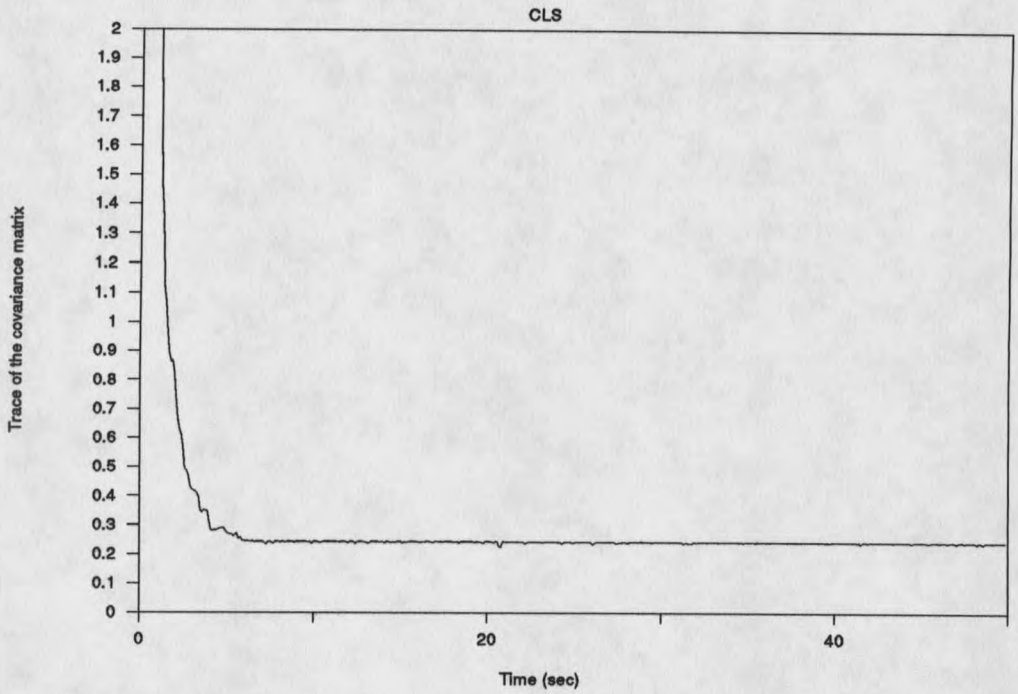

Figure 25. Trace of the covariance matrix (CLS).

Example 3

In this example, we investigate different measures of richness suggested in the literature as well as the new measure used in the ratio-ratio test.

Consider the following second-order oscillatory process:

$$y(t) = \frac{1.0z^{-1} + 0.5z^{-2}}{1.0 - 1.5z^{-1} + 0.7z^{-2}} u(t) + \frac{1.0}{1.0 - 1.5z^{-1} + 0.7z^{-2}} e(t)$$

where

$u(t) = \pm 1.0$ with duration of 200 samples.

and

$e(t)$ is Gaussian white noise with $\sigma^2 = 1.0$.

A variety of detection procedures have been suggested in the literature. A rigorous way of testing for persistency of excitation is to estimate the positive-definite properties of C_n in (6.22) at every time instant. Other checks which can be used for this purpose are as follows:

$$TrP(t) \le \alpha_1 \tag{9.1}$$

$$\tau - \alpha_2 \le \frac{TrP(t-1)}{TrP(t)} \le \tau + \alpha_2 \qquad \text{where} \quad \tau \quad \text{is the forgetting factor} \tag{9.2}$$

$$C\{P(t)\} \le \alpha_3 \quad \text{where C stands for "condition number of"} \tag{9.3}$$

$$\frac{C\{P(t-1)\}}{C\{P(t)\}} \le \alpha_4 \tag{9.4}$$

$$\frac{\text{Max}.eigenvalue of P(t)}{\text{Min} eigenvalue of P(t)} \le \alpha_5 \tag{9.5}$$

$$\frac{\text{Max}.eigenvalue of C_n}{\text{Min} eigenvalue of C_n} \le \alpha_6, \qquad C_n \quad \text{defined in (7.22)} \tag{9.6}$$

$$||P(t)*DataVector||_\infty \le \alpha_7 \tag{9.7}$$

$$1 - \alpha_8 \le \frac{TrP(t-2)*TrP(t)}{TrP(t-1)^2} \le 1 + \alpha_8 \tag{9.8}$$

$$\frac{C\{P(t-2)\}*C\{P(t)\}}{C\{P(t-1)\}^2} \le \alpha_9 \tag{9.9}$$

where α_i, $i = 1,...,9$ are user specified bounds.

These functions expressed in (9.1) through (9.9) were evaluated by simulation in this example. The measured system output and input of the system are shown in Figures 26 and 27, respectively. Figures 28 through 37 present the results

corresponding to (9.1) through (9.9). As can be seen from these results, the ratio-ratio function (Figure 36) gives essentially the same information as the more complicated and more computationally expensive condition number function (Figure 37).

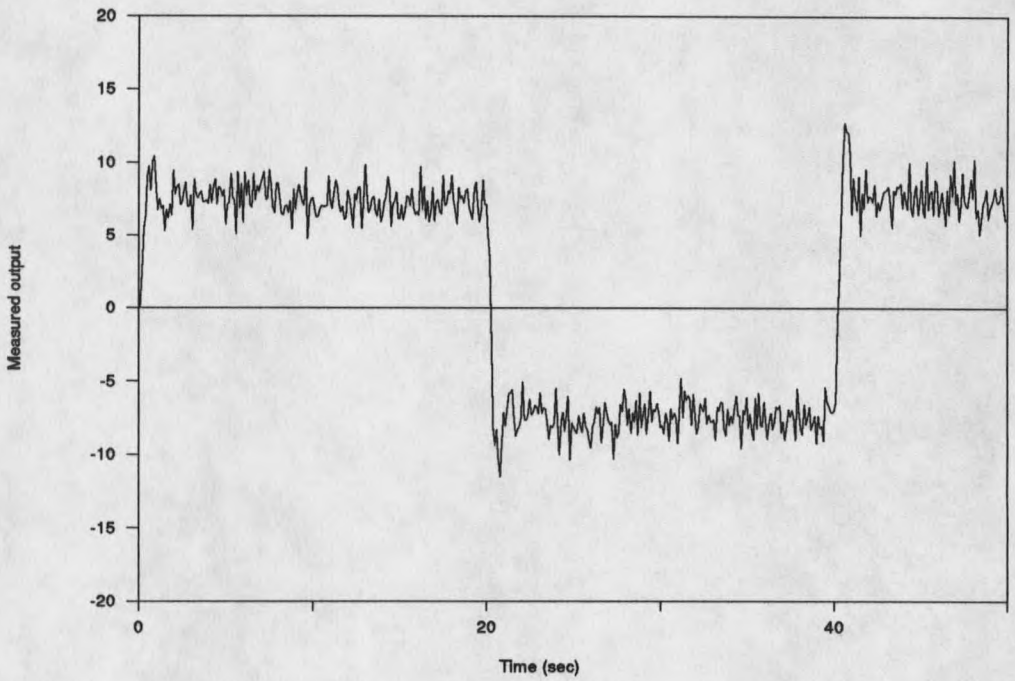

Figure 26. Measured output.

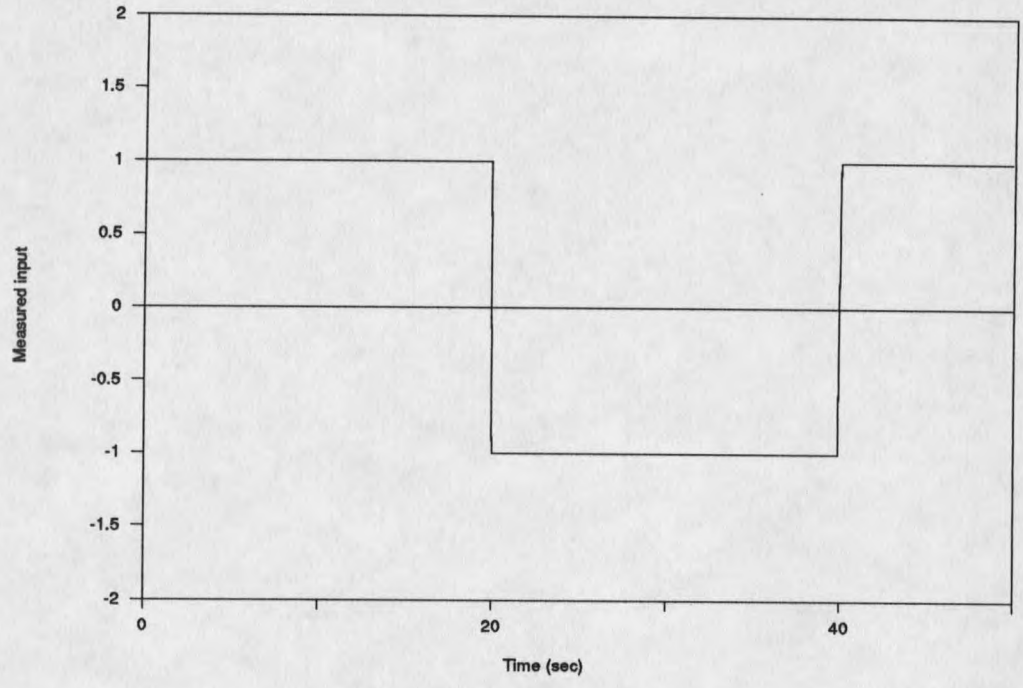

Figure 27. Measured input.

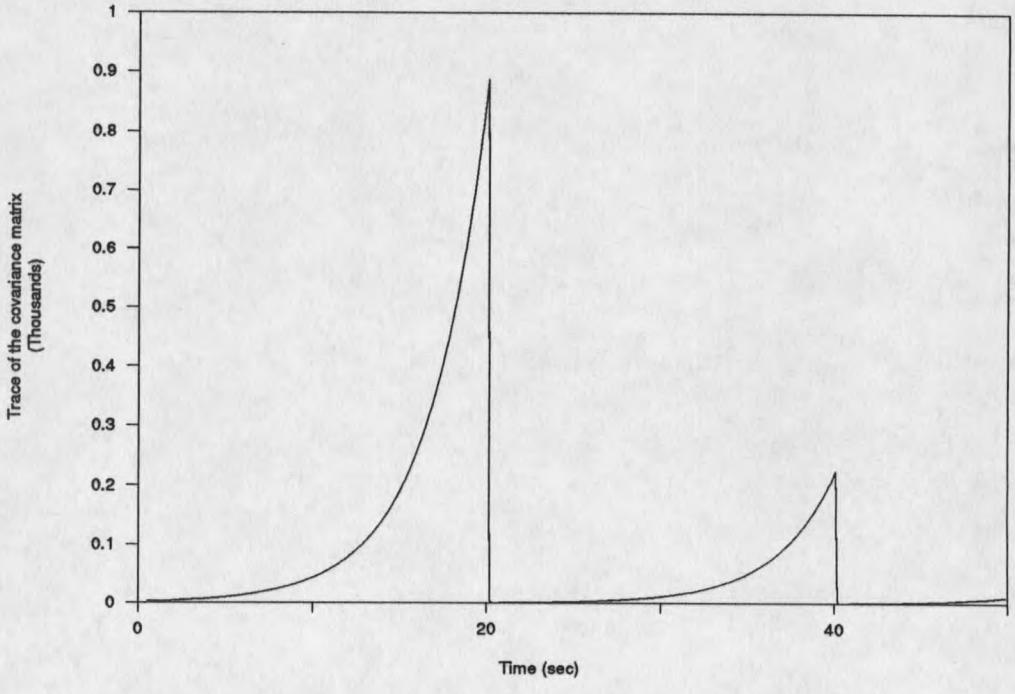

Figure 28. Trace of the covariance matrix.

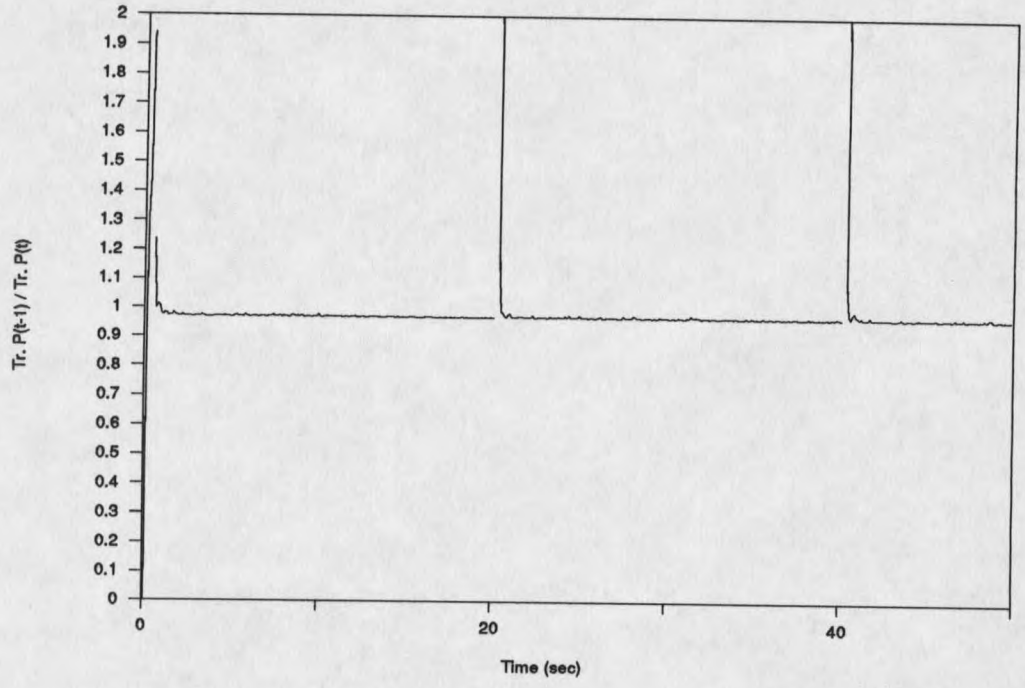

Figure 29. Trace of P(t-1)/Trace of P(t).

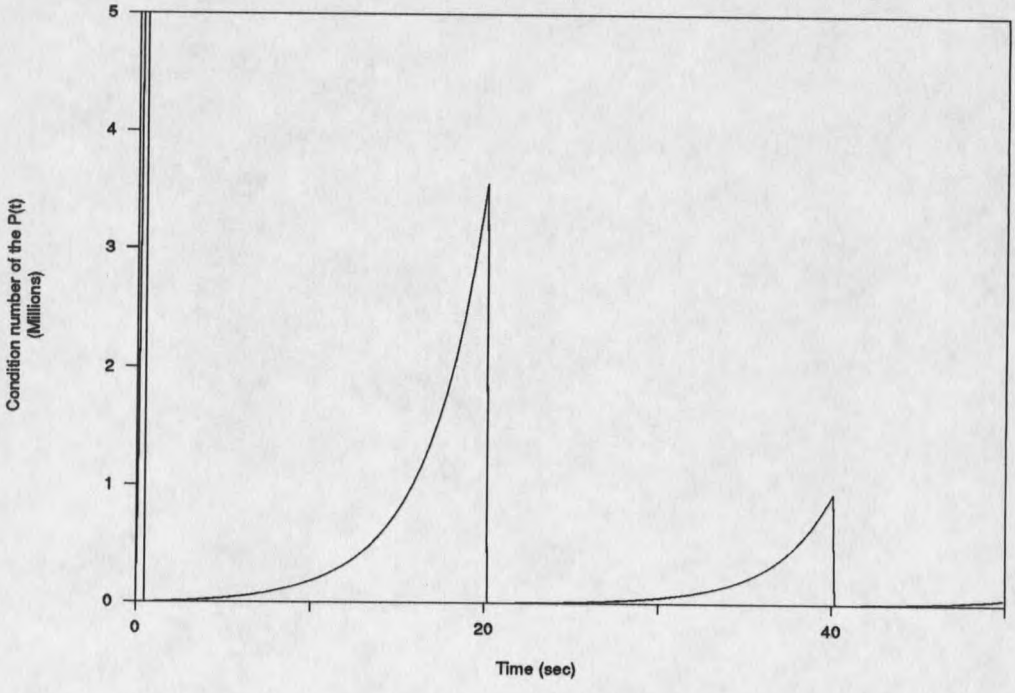

Figure 30. Condition number of the covariance matrix.

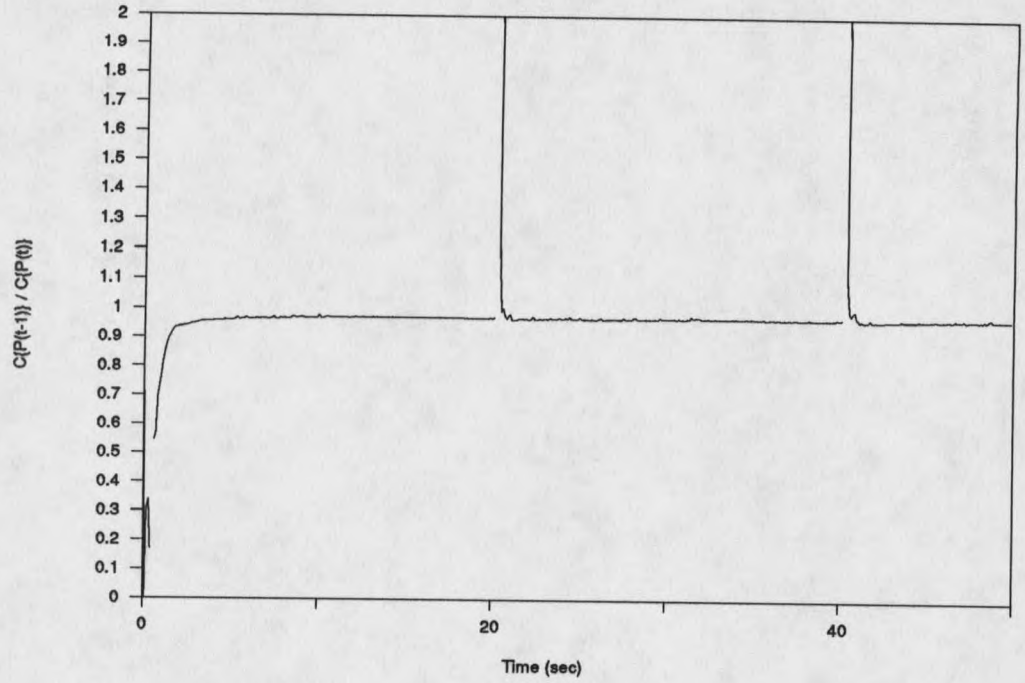

Figure 31. Con. # of P(t-1)/Con. # of P(t).

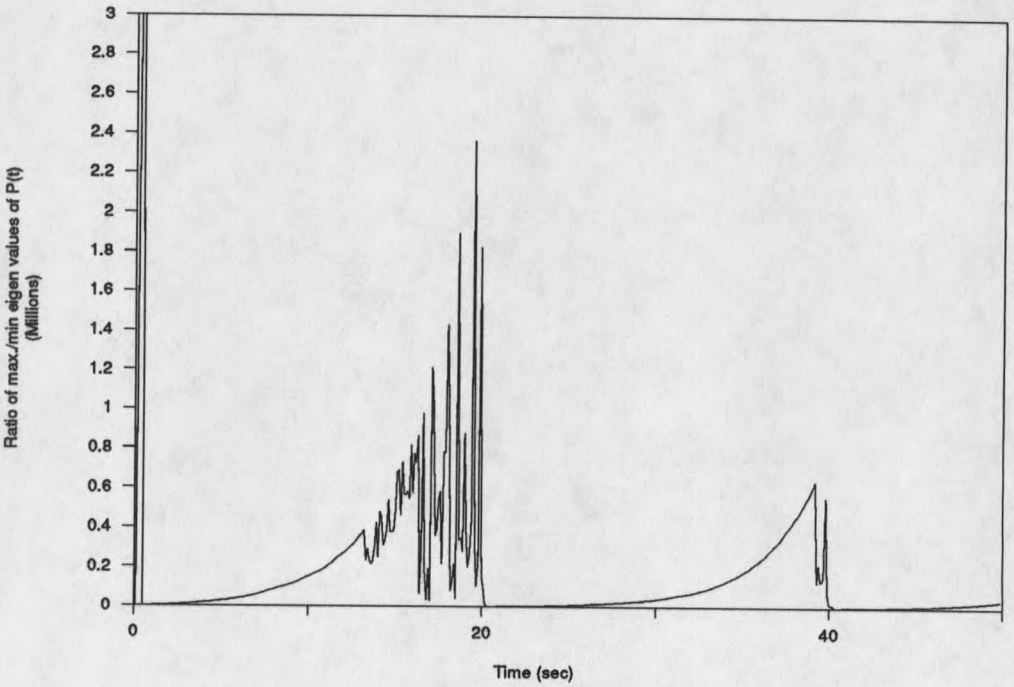

Figure 32. (Max. E.V./min. E.V.) of the P(t) matrix.

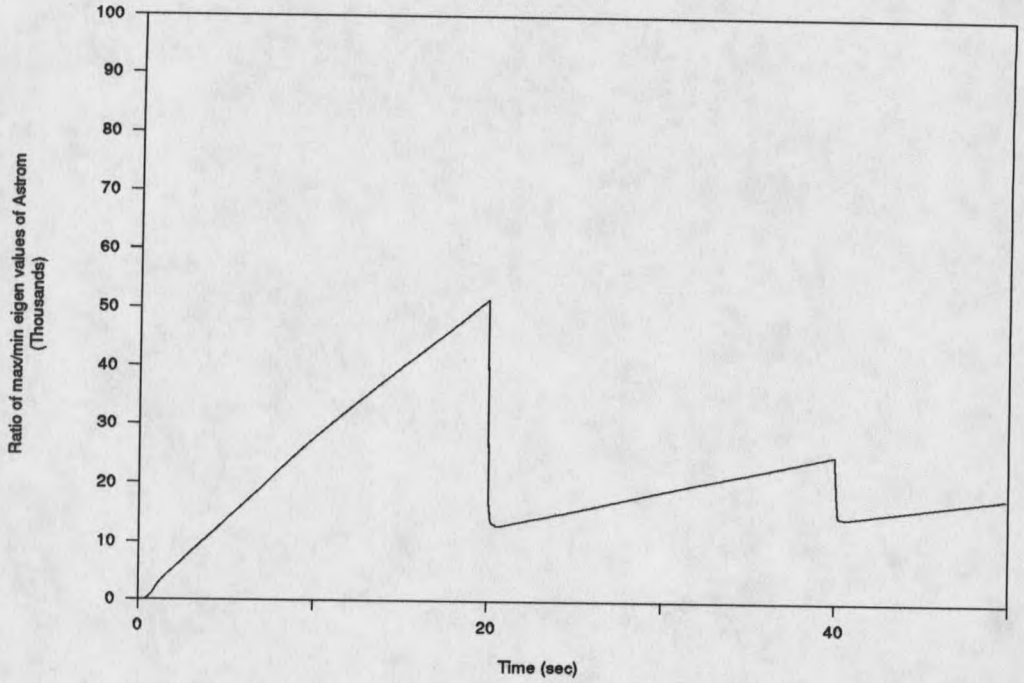

Figure 33. (Max. E.V./min. E.V.) of the Astrom matrix.

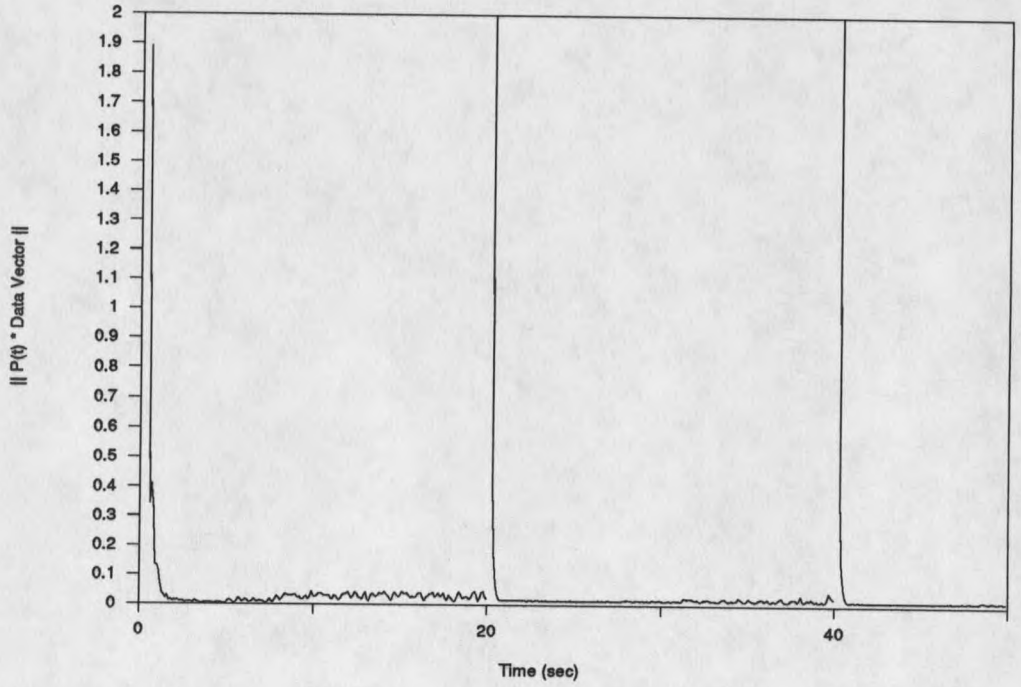

Figure 34. Norm of (P(t) times data vector)

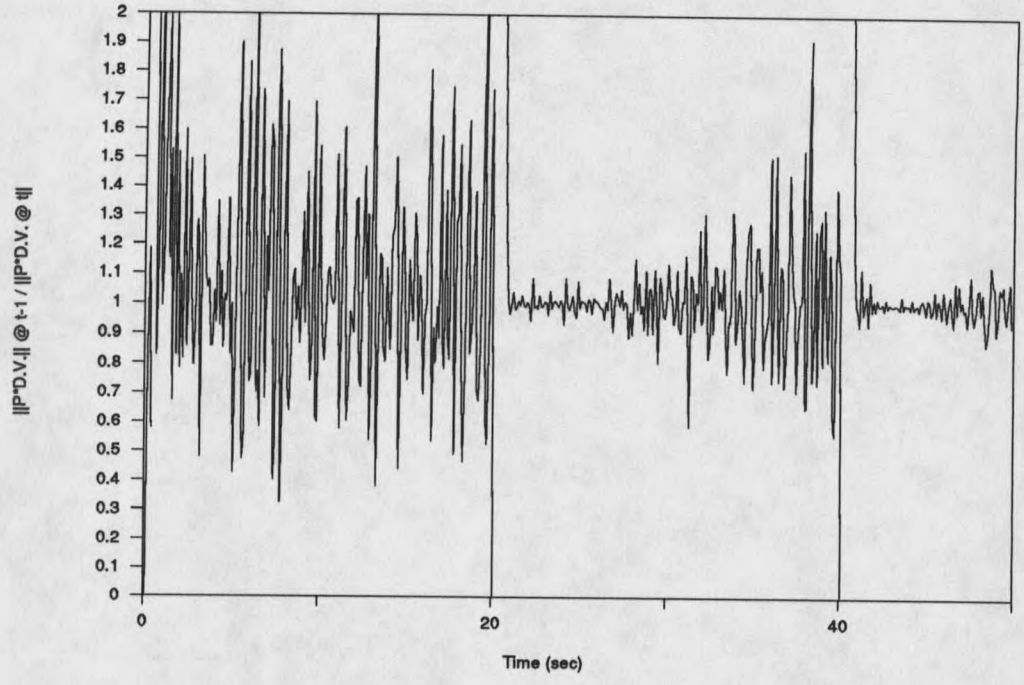

Figure 35. $||P(t-1)*D.V.(t-1)||/||P(t)*D.V.(t)||$.

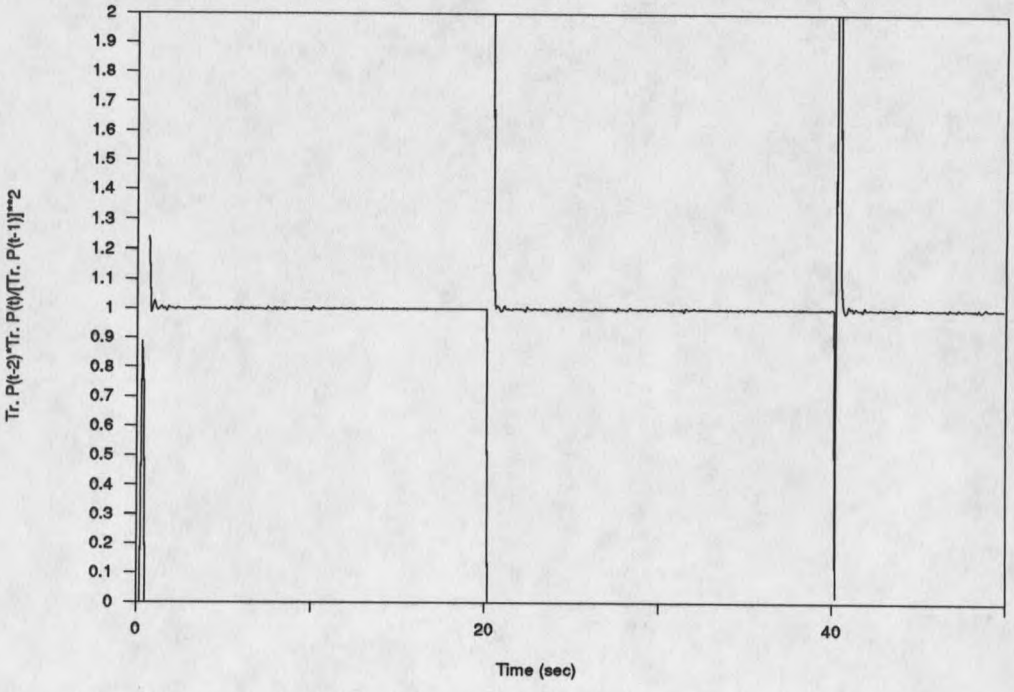

Figure 36. $Tr.\ P(t-2)*Tr.\ P(t)/[Tr.\ P(t-1)]^2$.

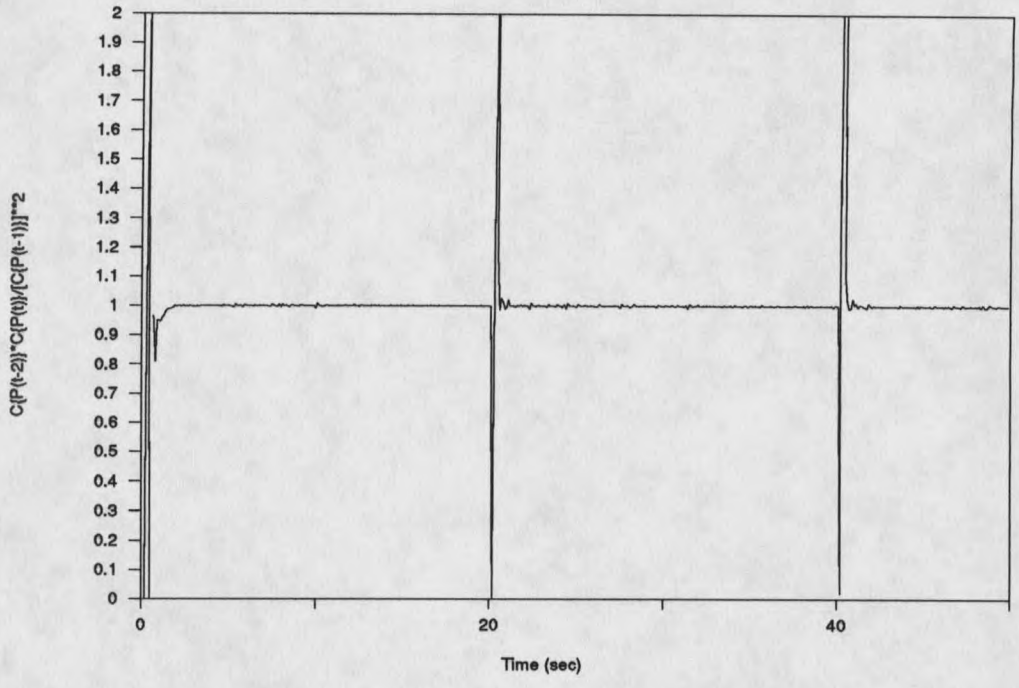

Figure 37. $C\{P(t-2)\}\ C\{P(t)\}\ /\ [C\{P(t-1)\}]^2$.

Example 4

This example basically is the same as Example 3, except that correlated noise is added to the output measurements to investigate the performance of the richness tests under this condition.

Consider the following second-order oscillatory process:

$$y(t) = \frac{1.0z^{-1} + 0.5z^{-2}}{1.0 - 1.5z^{-1} + 0.7z^{-2}} u(t) + v(t)$$

where

$$v(t) = \frac{1.0 - 0.189z^{-1}}{1.0 - 1.027z^{-1} + 0.264z^{-2}} e(t)$$

$e(t) = 10\sin(5t) + \text{White Noise with} \quad \sigma^2 = 1.0$

and

$u(t) = \pm 1.0$, where + or - is chosen at random.

The functions expressed in (9.1) through (9.9) were evaluated by simulation. The measured system output and input from the system are shown in Figures 38 and 39, respectively. Figures 40 through 41 present the results corresponding to (9.1) through (9.9). As can be seen from these results, the ratio-ratio function (Figure 47) gives essentially the same information as the more complicated and more computationally expensive condition number function (Figure 48).

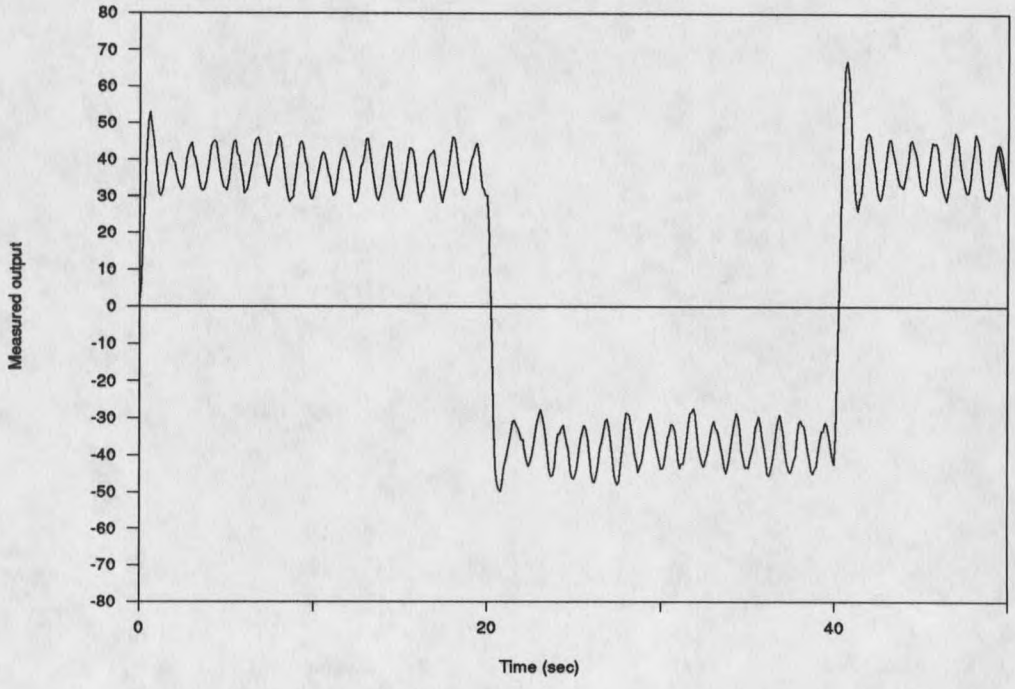

Figure 38. Measured output.

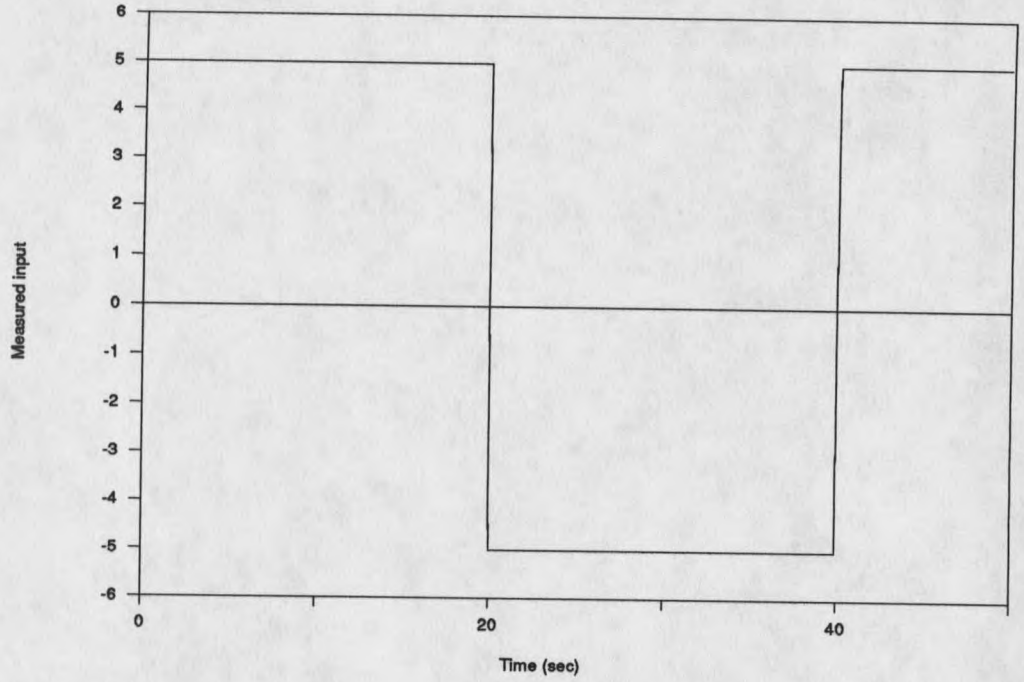

Figure 39. Measured input.

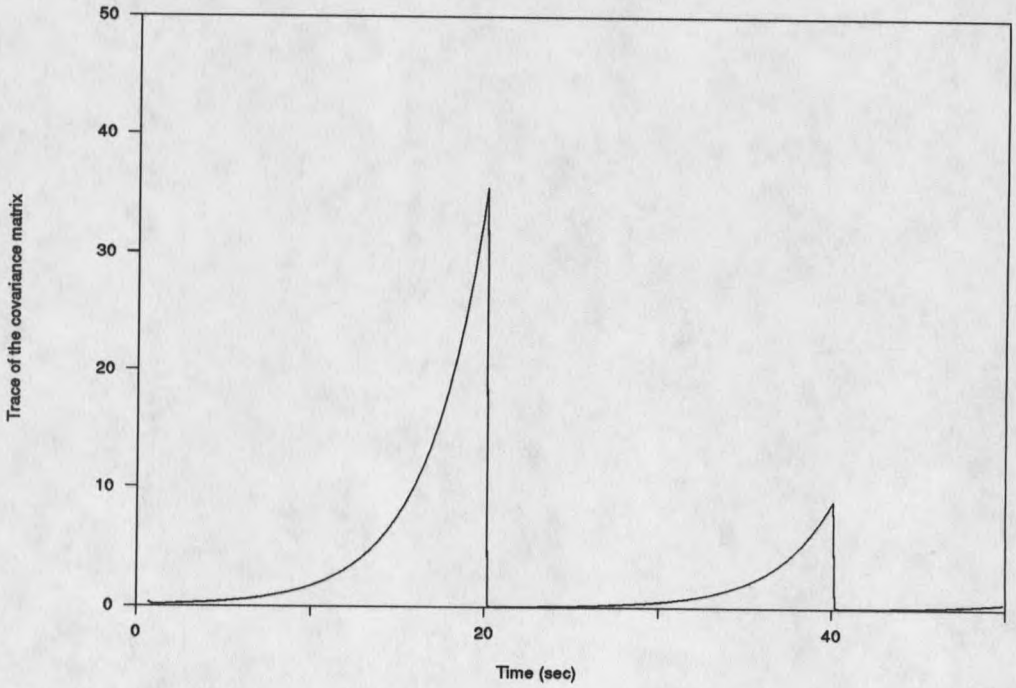

Figure 40. Trace of the covariance matrix.

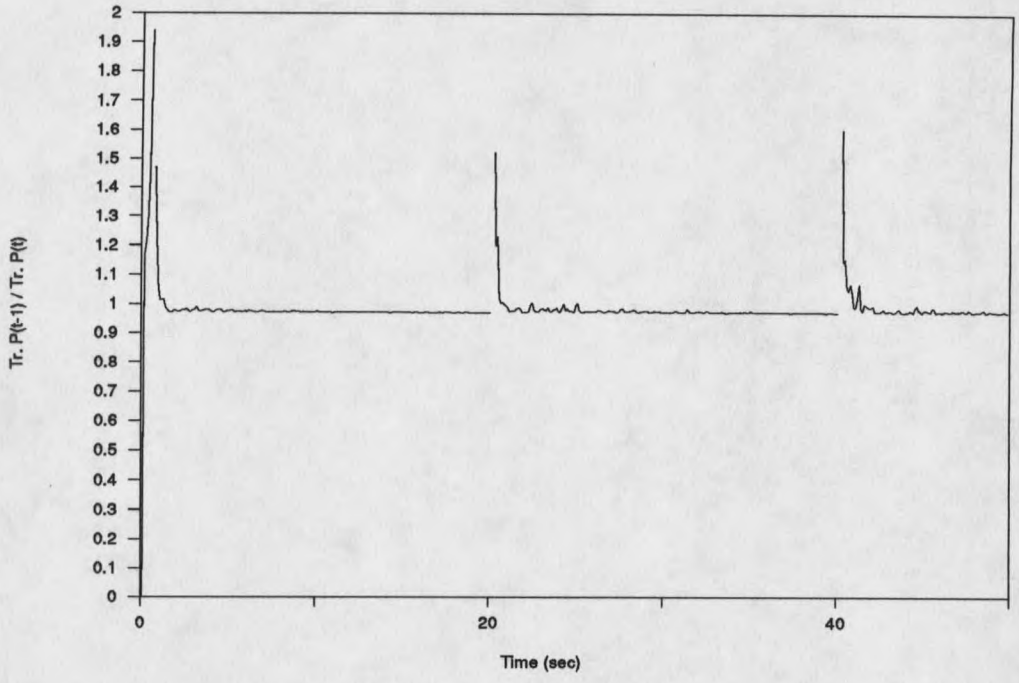

Figure 41. Trace of P(t-1)/Trace of P(t).

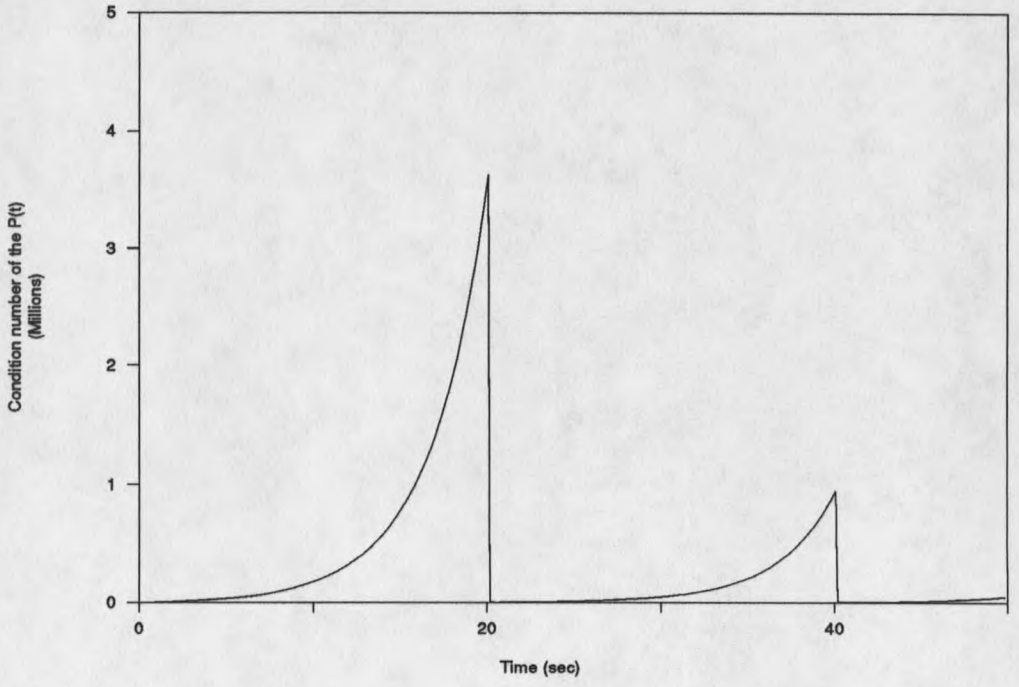

Figure 42. Condition number of the covariance matrix.

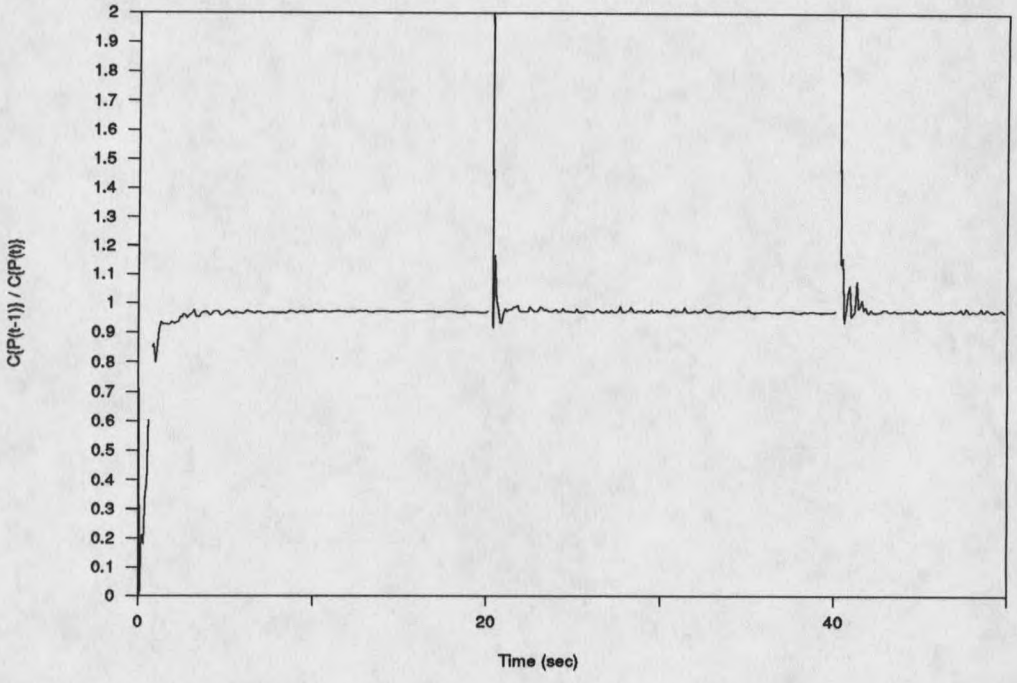

Figure 43. Con. # of P(t-1)/Con. # of P(t).

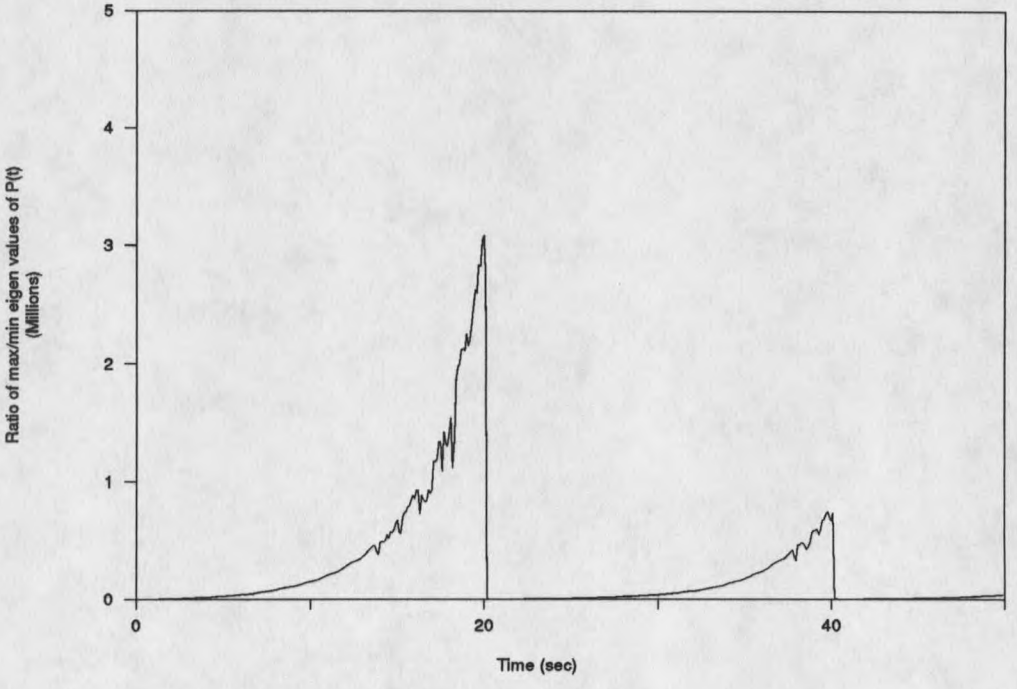

Figure 44. (Max. E.V./min. E.V.) of the P(t) matrix.

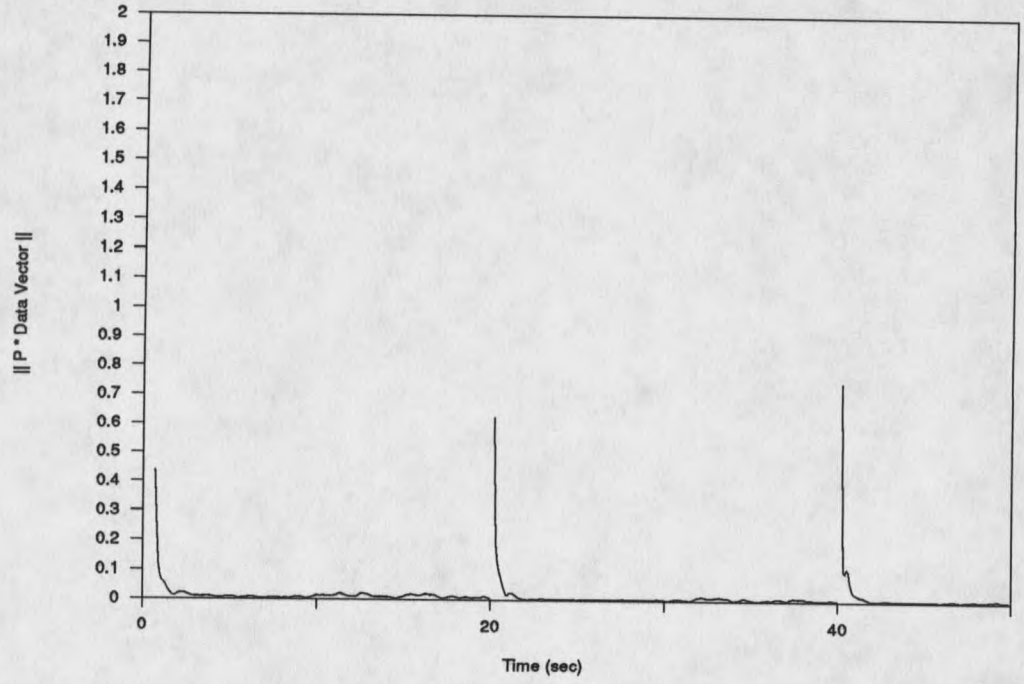

Figure 45. Norm of (P(t) times data vector)

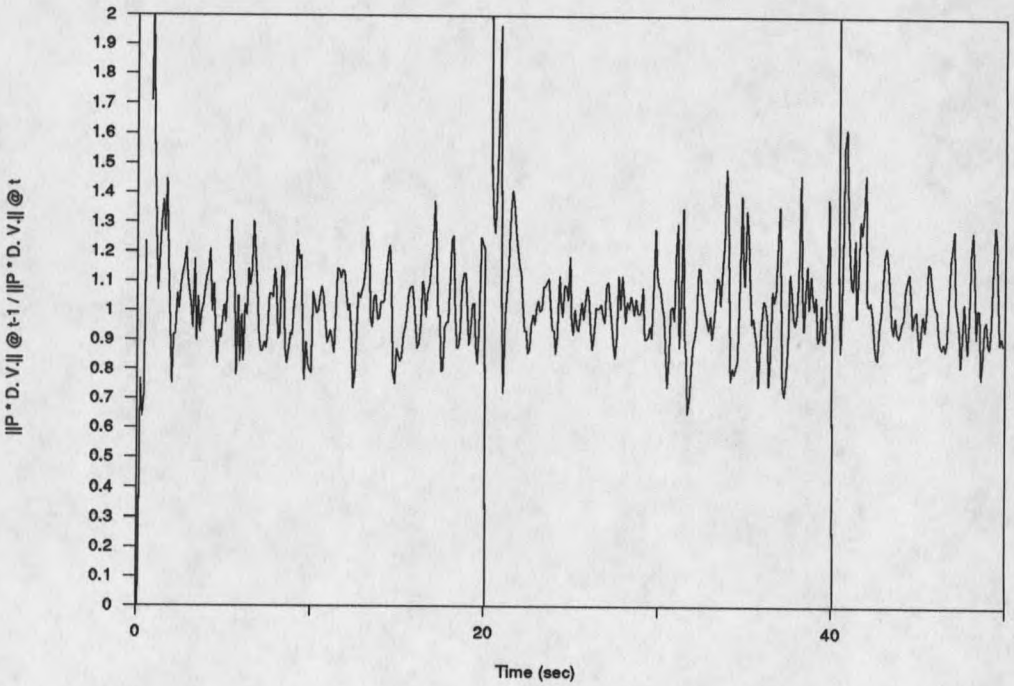

Figure 46. ||P(t-1)*D.V.(t-1)||/||P(t)*D.V.(t)||.

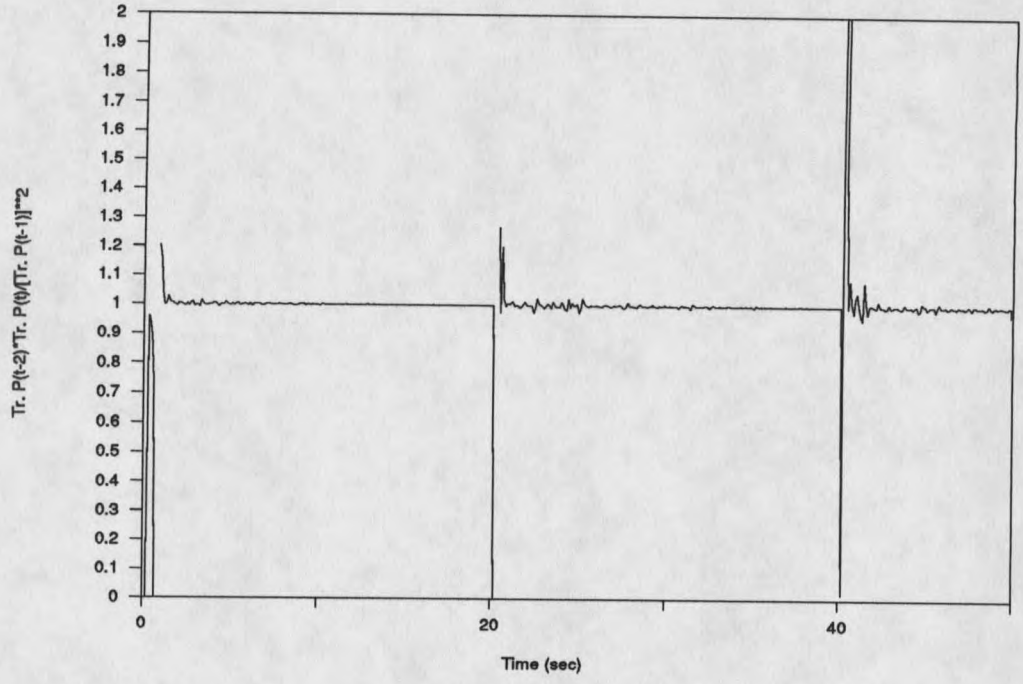

Figure 47. Tr. P(t-2)*Tr. P(t)/[Tr. P(t-1)]^2.

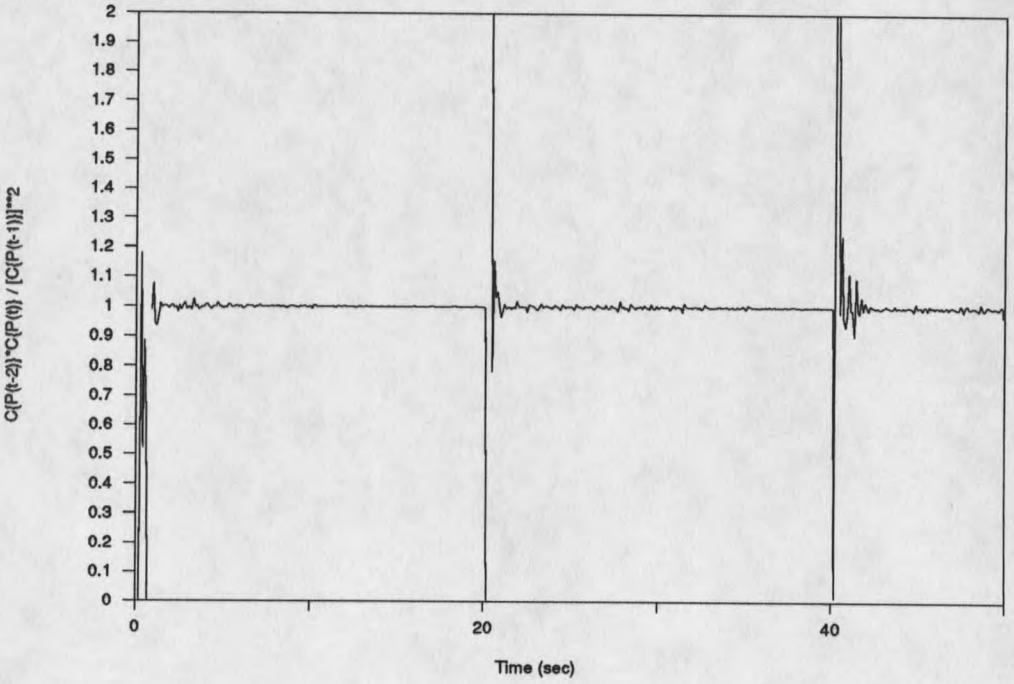

Figure 48. C{P(t-2)} C{P(t)} / [C{P(t-1)}]^2.

Example 5

In this example, we investigate the performance of the RLS, ILS, and CLS algorithms, when the system under consideration is a time-varying system.

Consider the following second-order oscillatory process:

$$y(t)=\frac{1.0z^{-1}+0.5z^{-2}}{1.0-1.5z^{-1}+0.7z^{-2}}u(t)+\frac{1.0}{1.0-1.5z^{-1}+0.7z^{-2}}e(t)$$

where

$u(t)=\pm 1.0$, where + or - choosen at random

and

$e(t)$ is Guassian white noise with $\sigma^2=1.0$

Also the a_1 parameter is changed as follows

$$a_1(t) = \begin{cases} -1.5 & \text{for } 0 \le t < 20 \\ 0.0 & \text{for } 20 \le t < 40 \\ -1.0 & \text{for } 40 \le t \le 50 \end{cases}$$

The results of identification for this system are given in Figures 49 through 59. Figures 49 and 50 present the measured system output and input, respectively. Figures 51 through 56 give the parameter estimates obtained during the simulations by using RLS, ILS, and CLS algorithms. The traces of the covariance matrices obtained by each of these methods are presented in Figures 57 through 59.

The simulation results clearly show the problem of turn-off, with the RLS algorithm. The parameter estimates obtained by all the above procedures converge rapidly to some reasonable values during the initial time interval. As can be seen from the simulation results, the RLS estimator was not able to track the parameter changes at 20 sec and 40 sec (Figure 52). This is due to the turn-off problem in the RLS algorithm. As can be seen from Figure 57, the trace of the covariance matrix approaches zero as time increases, therefore causing the estimator to turn-off.

The ILS algorithm shows good parameter tracking due to the variable forgetting factor used to hold the trace of the covariance matrix constant for all time (Figure 58). Keeping the trace of the covariance matrix constant, makes the algorithm more alert, and therefore it can track the time varying system. On the other hand this procedure makes the estimator more sensitive to noise, as can be seen from Figure 52 wherein noisy estimates of the parameters are evident. Therefore keeping the trace of covariance matrix constant is not a good practice when noise is present in the system measurements.

The CLS algorithm performs with good parameter tracking as well as good noise rejection, as can be seen in Figure 53.

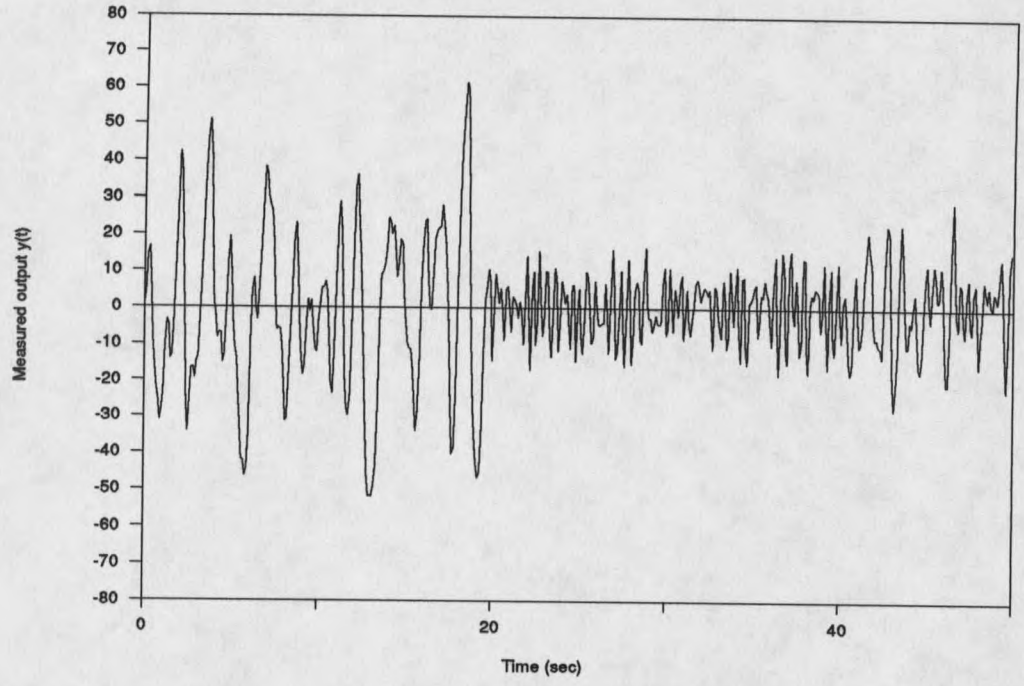

Figure 49. Measured output.

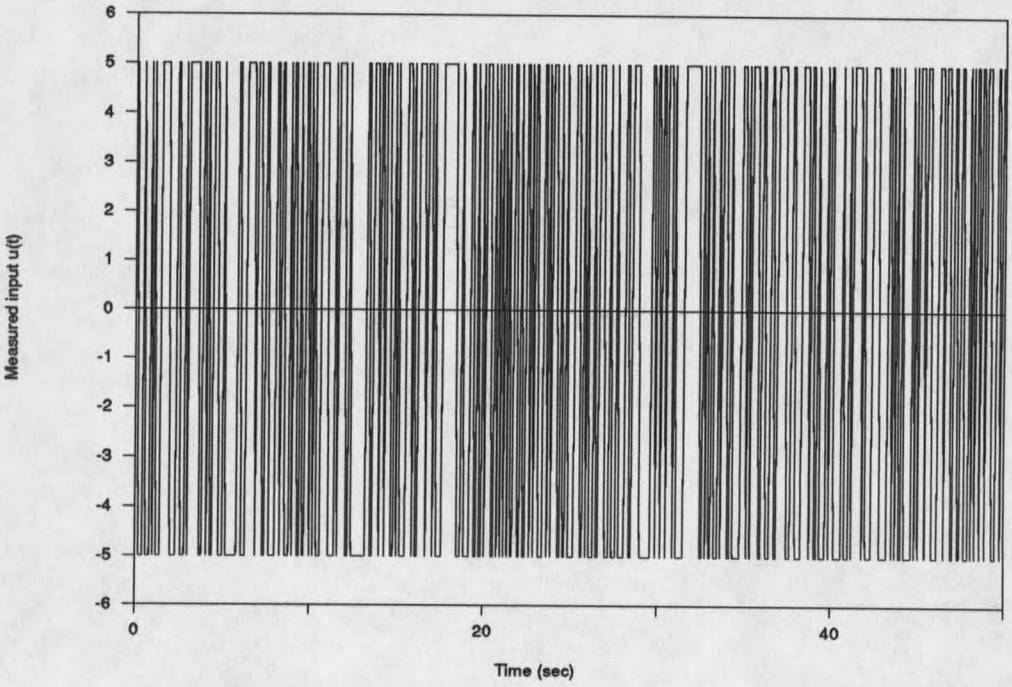

Figure 50. Measured input.

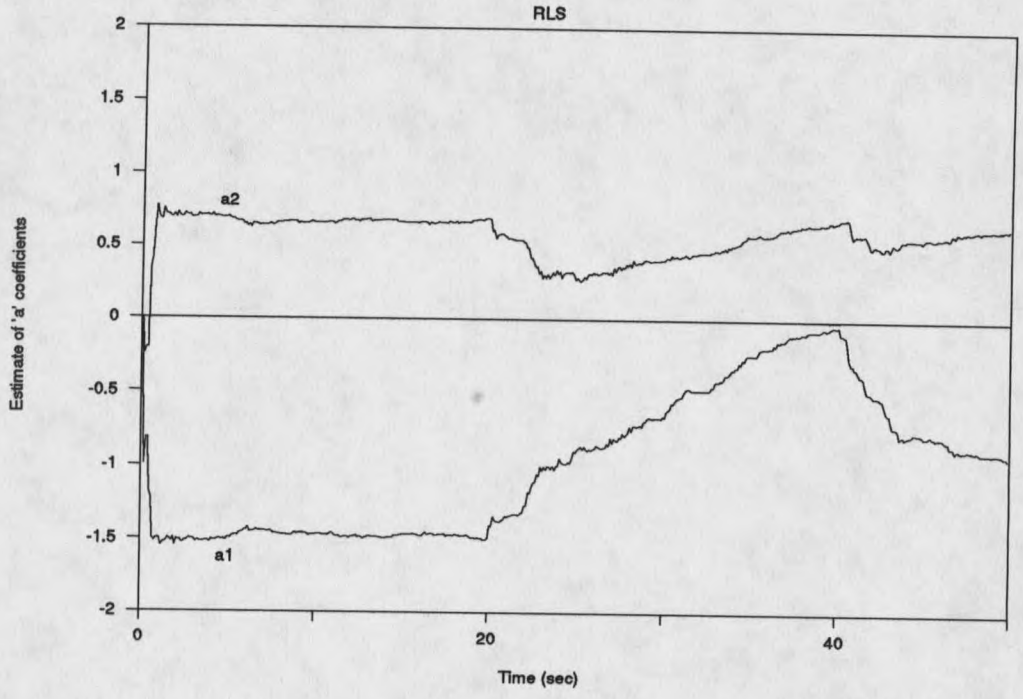

Figure 51. Estimate of the 'a' coefficients (RLS).

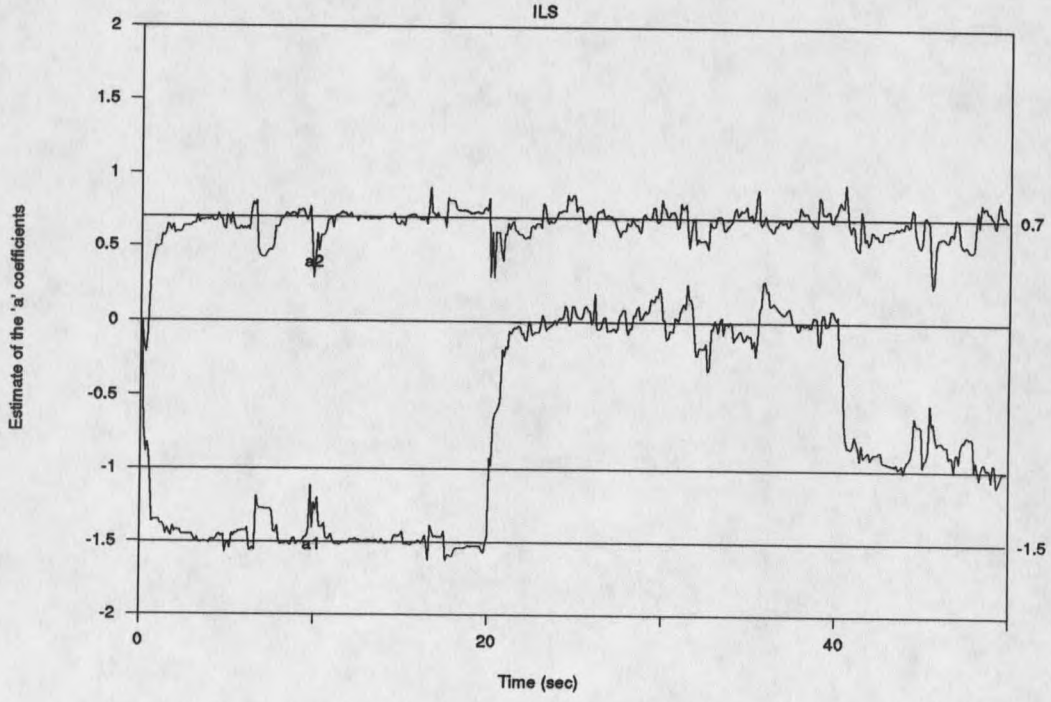

Figure 52. Estimate of the 'a' coefficients (ILS).

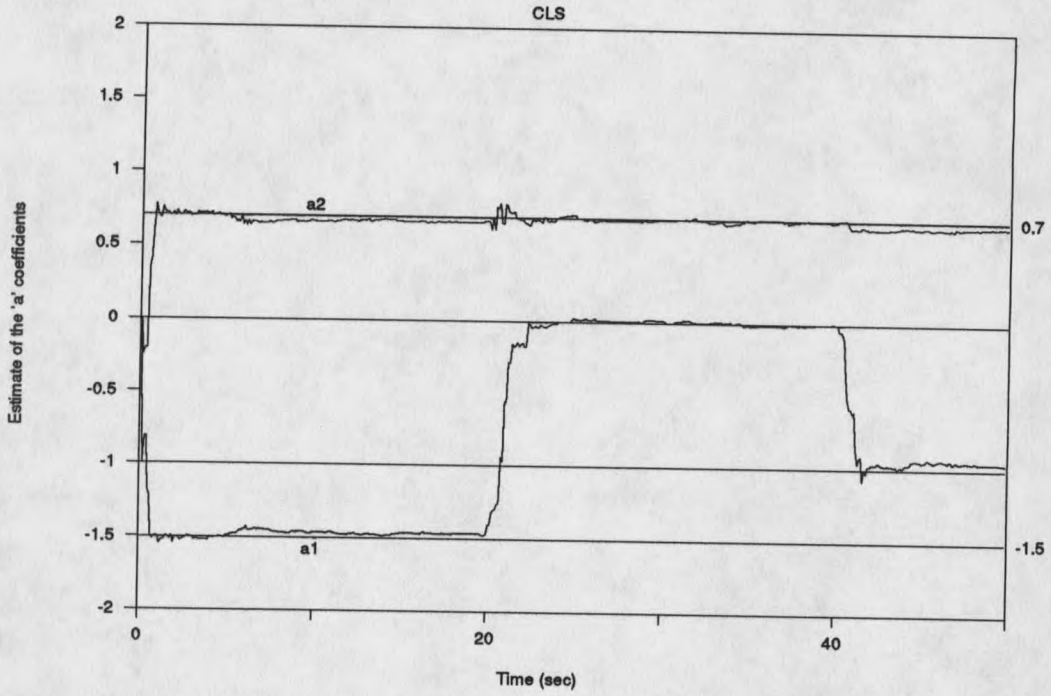

Figure 53. Estimate of the 'a' coefficients (CLS).

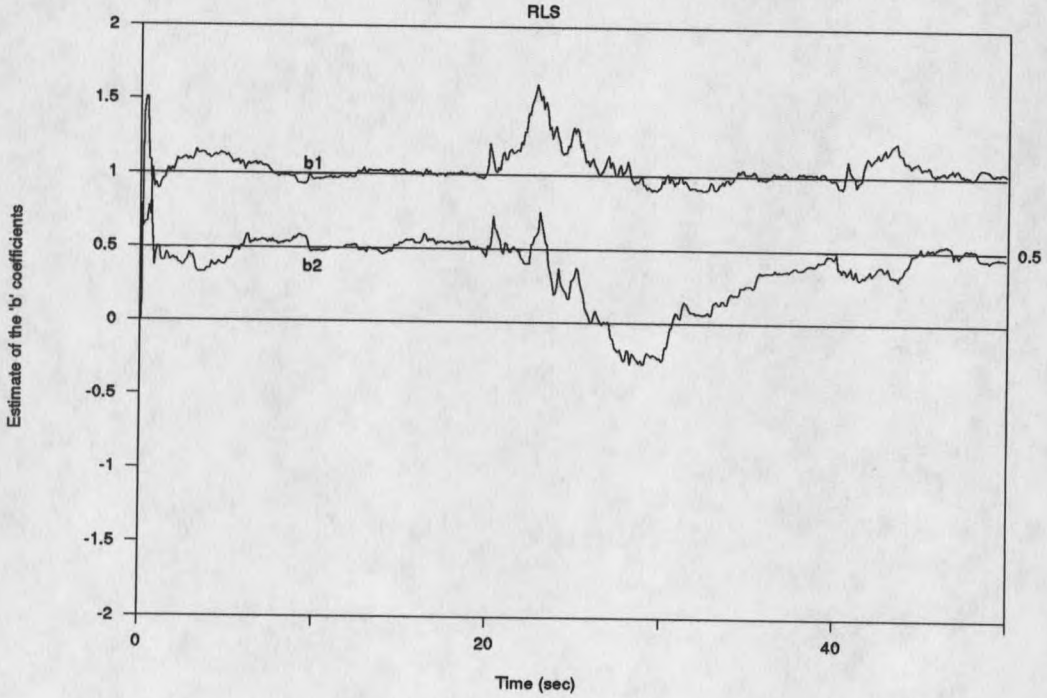

Figure 54. Estimate of the 'b' coefficients (RLS).

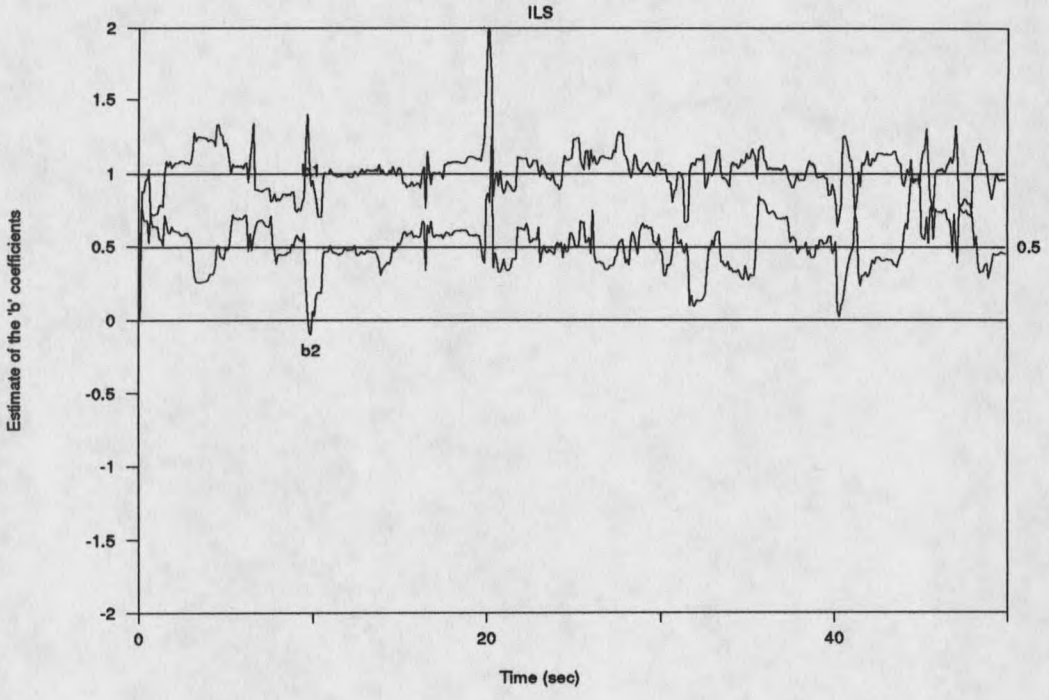

Figure 55. Estimate of the 'b' coefficients (ILS).

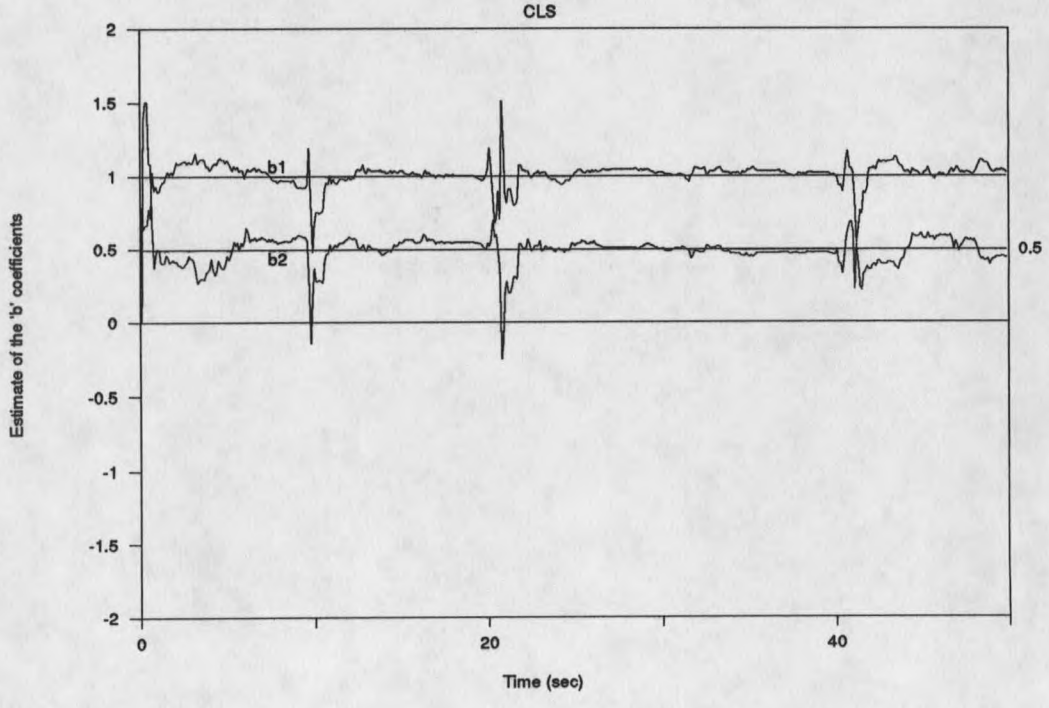

Figure 56. Estimate of the 'b' coefficients (CLS).

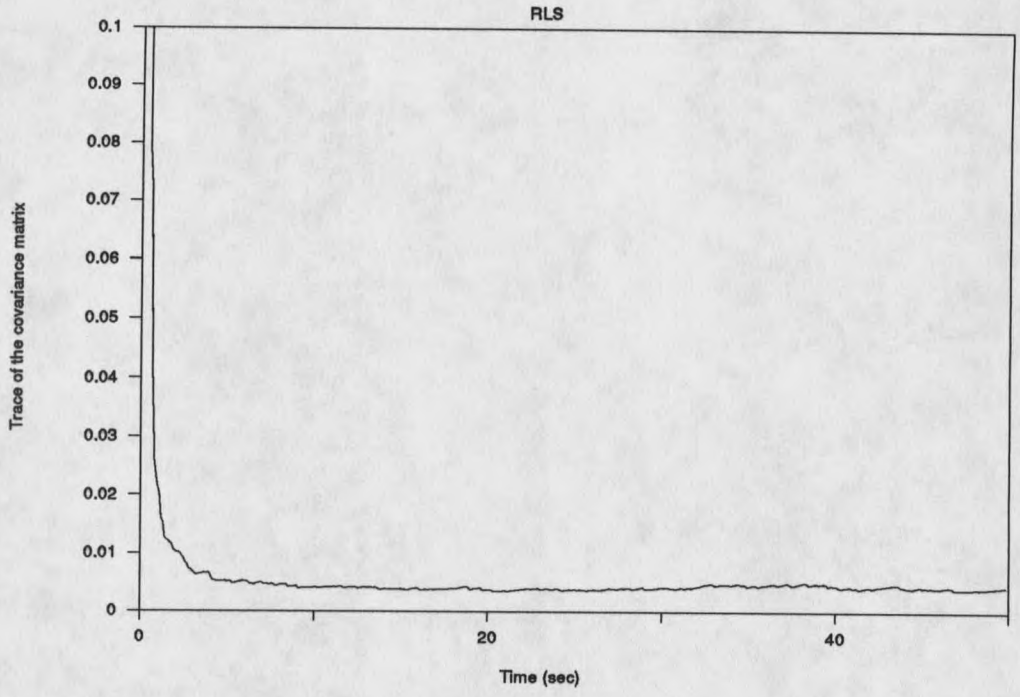

Figure 57. Trace of the covariance matrix (RLS).

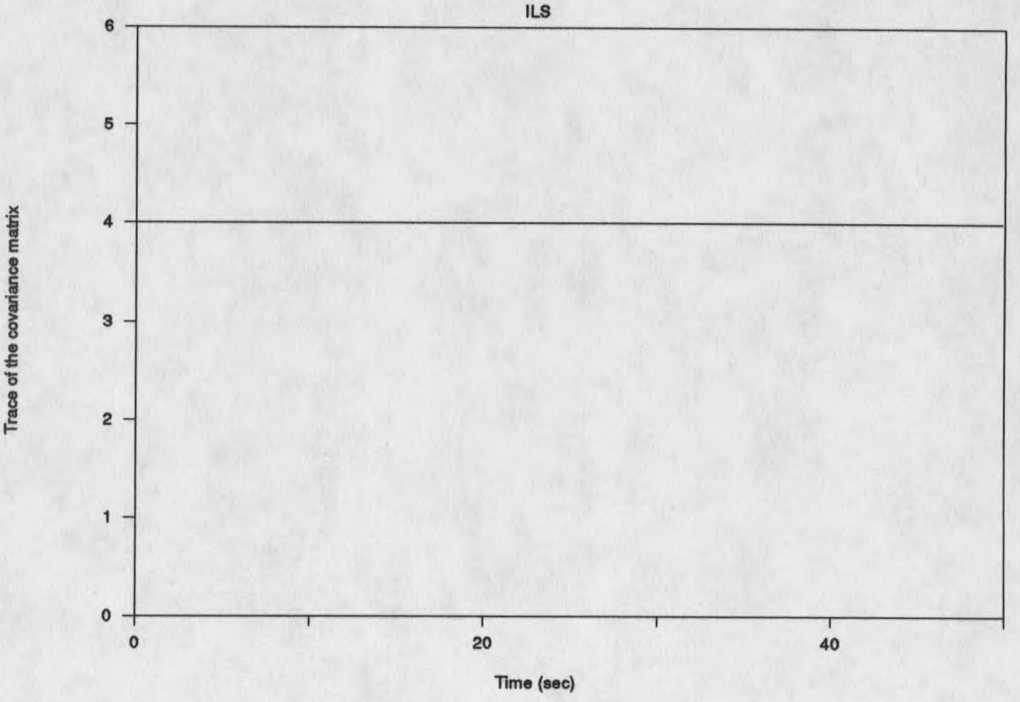

Figure 58. Trace of the covariance matrix (ILS).

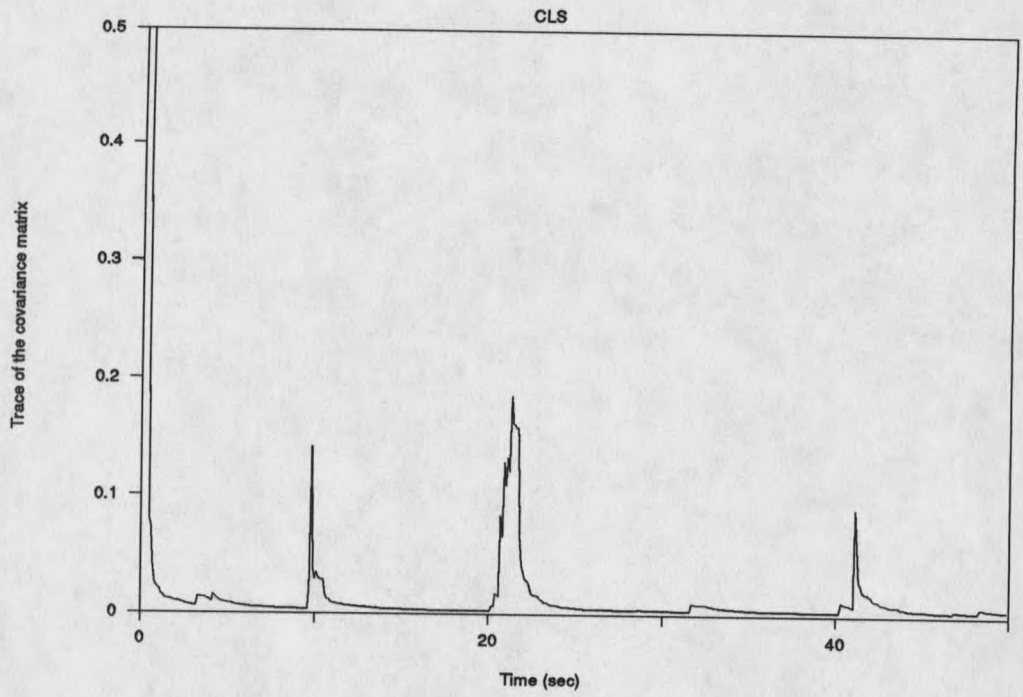

Figure 59. Trace of the covariance matrix (CLS).

Example 6

This example basically is the same as Example 5, except that the a_1 coefficient changes differently. The a_1 coefficient assumed the following values during the simulations;

$$a_1(t) = \begin{cases} -1.5 & \text{for} \quad 0 \le t < 10 \\ 0.0 & \text{for} \quad 10 \le t < 20 \\ a_1(t) - 0.05 & \text{for} \quad 20 \le t < 30 \\ -1.0 & \text{for} \quad 30 \le t \le 50 \end{cases}$$

The results of identification for this system are given in Figures 60 through 70. Figures 60 and 61 present the measured system output and input, respectively. Figures 62 through 67 give the parameter estimates obtained during the simulations by using RLS, ILS, and CLS algorithms. The traces of the covariance matrices obtained by each of these methods are presented in Figures 68 through 70.

The simulation results clearly show the problem of turn-off, with the RLS algorithm. The parameter estimates obtained by all the above procedures converge rapidly to some reasonable values during the initial time interval. As can be seen from the simulation results, the RLS estimator was not able to track the parameter changes (Figure 62). This is due to the turn-off problem in the RLS algorithm. As can be seen from Figure 68, the trace of the covariance matrix approaches zero as time increases, therefore causing the estimator to turn-off.

The ILS algorithm shows good parameter tracking due to the variable forgetting factor used to hold the trace of the covariance matrix constant for all time (Figure 69). Keeping the trace of the covariance matrix constant, makes the algorithm more alert and therefore able to track the time varying system. On the other hand this procedure makes the estimator more sensitive to noise, as can be seen from Figure 63 wherein noisy estimates of the parameters are

evident. Therefore, once again, keeping the trace of covariance matrix constant is not a good practice when noise is presented in the system measurements.

The CLS algorithm performs with good parameter tracking as well as good noise rejection, as can be seen in Figure 64.

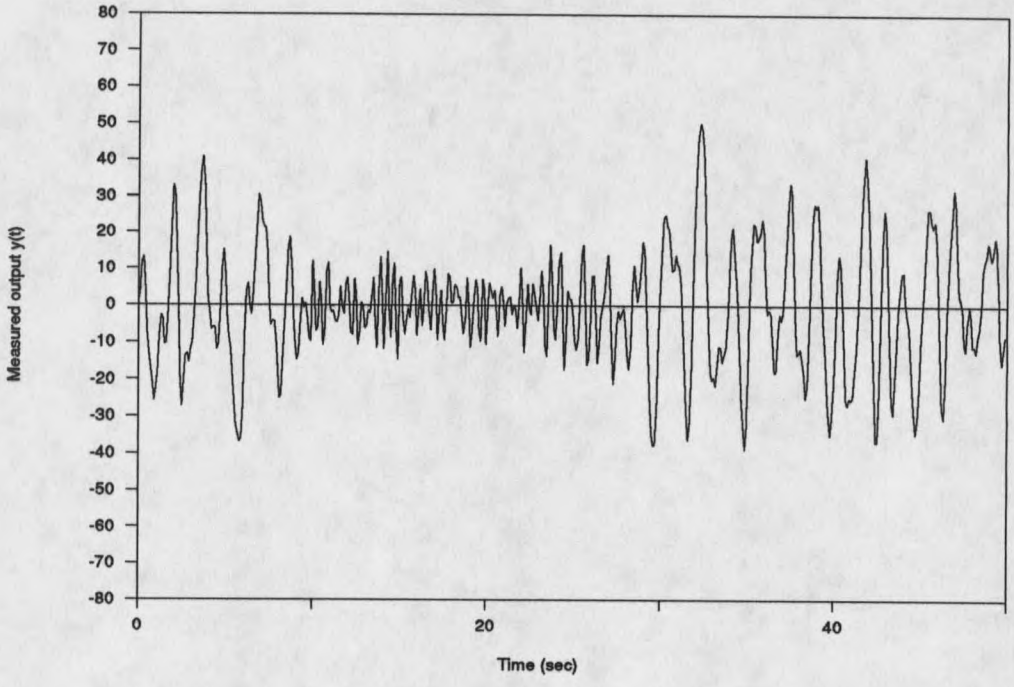

Figure 60. Measured output.

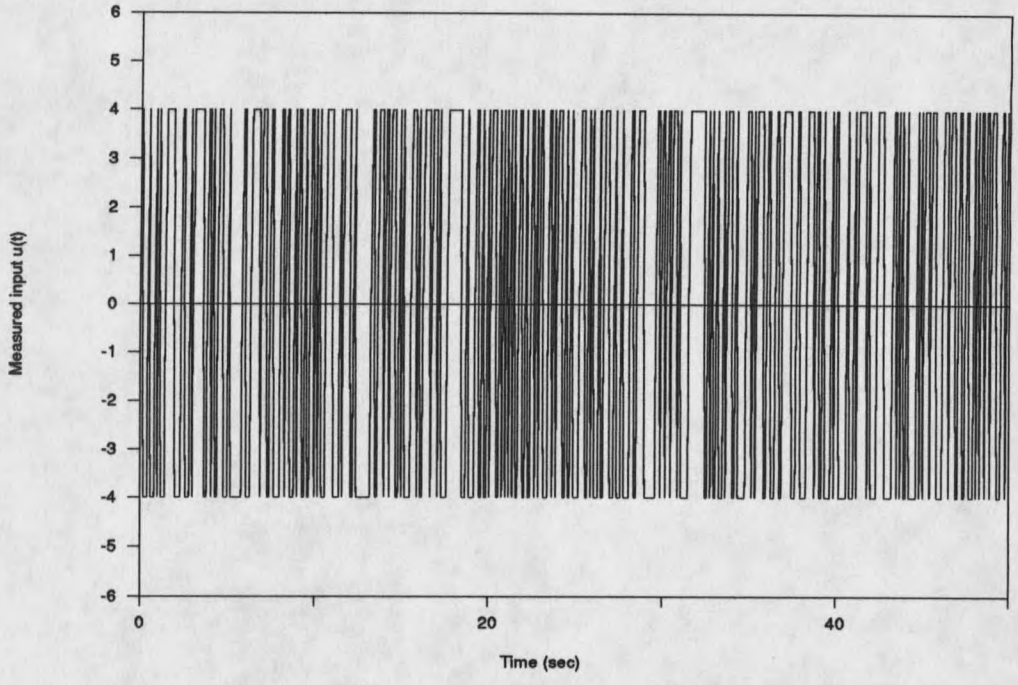

Figure 61. Measured input.

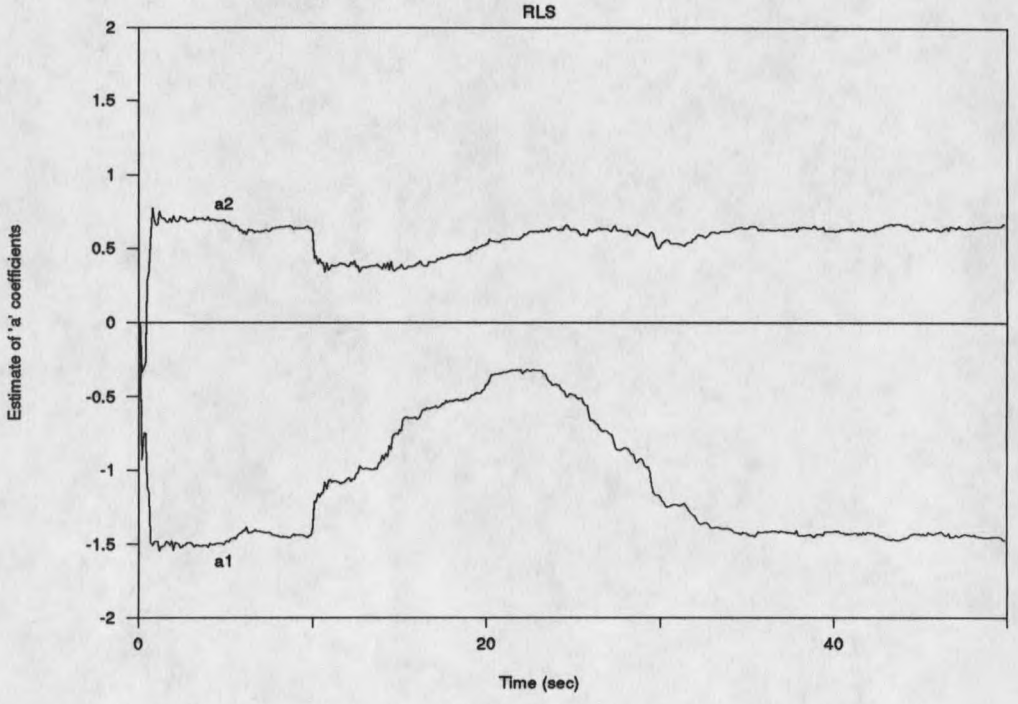

Figure 62. Estimate of the 'a' coefficients (RLS).

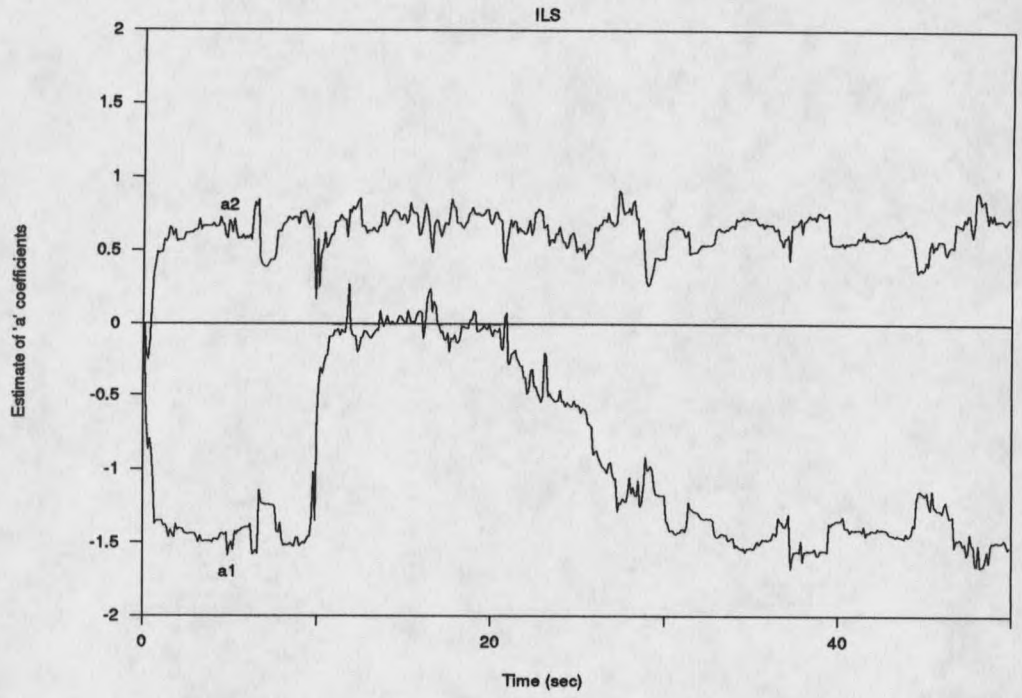

Figure 63. Estimate of the 'a' coefficients (ILS).

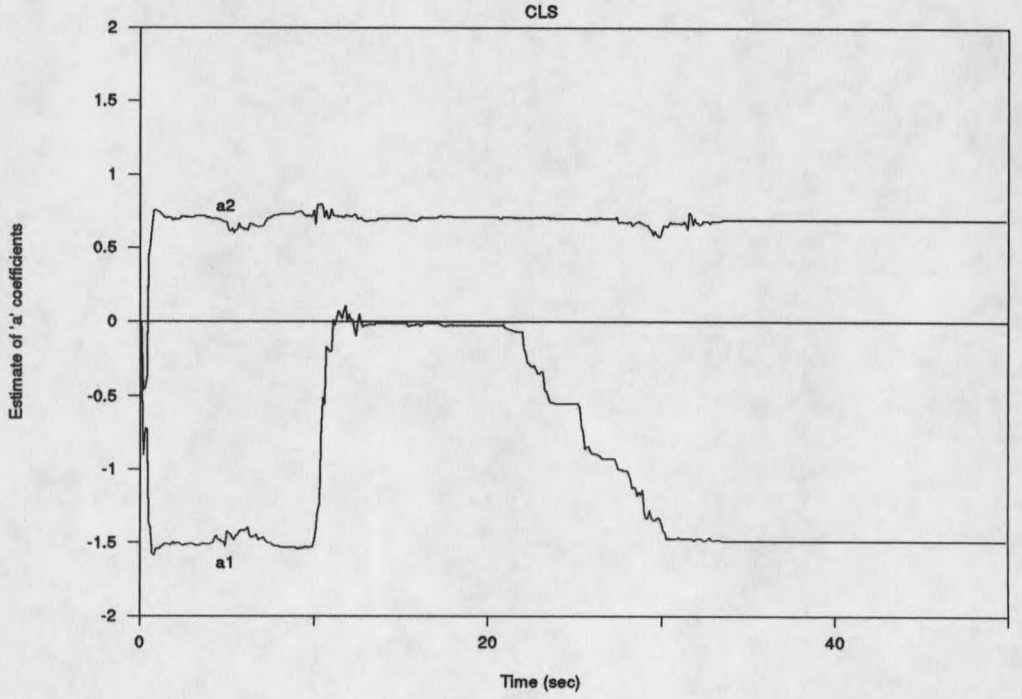

Figure 64. Estimate of the 'a' coefficients (CLS).

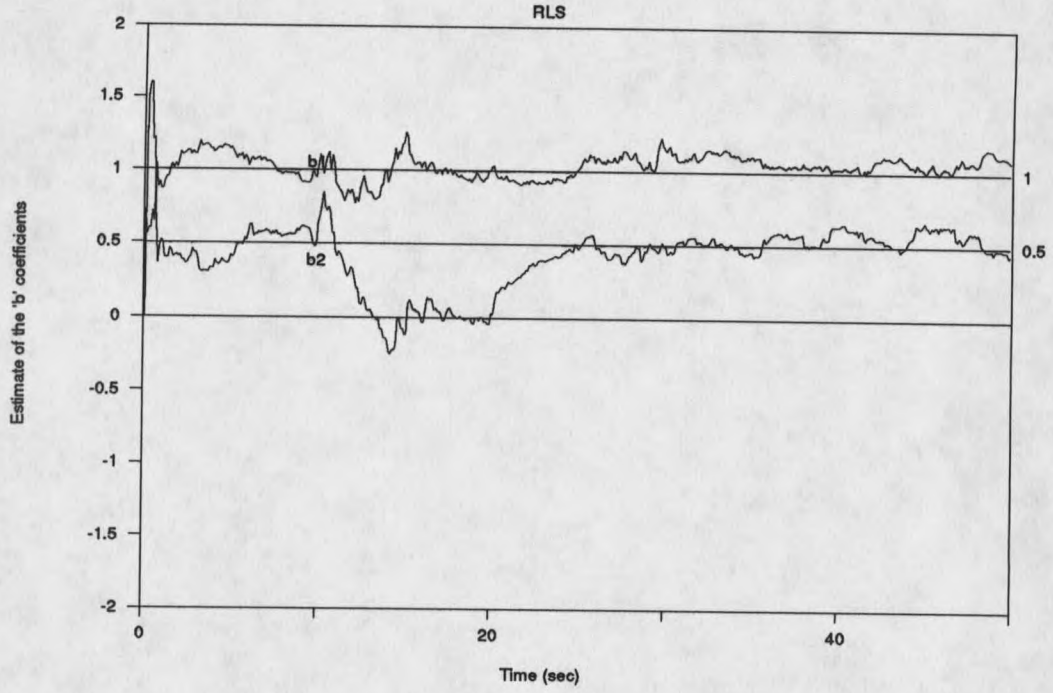

Figure 65. Estimate of the 'b' coefficients (RLS).

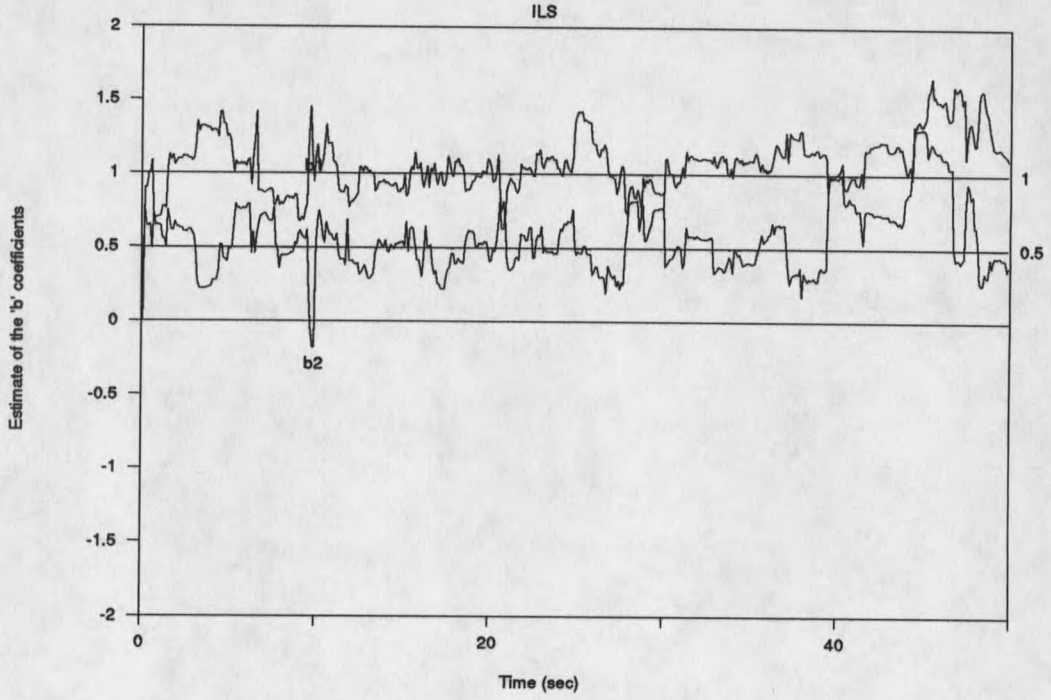

Figure 66. Estimate of the 'b' coefficients (ILS).

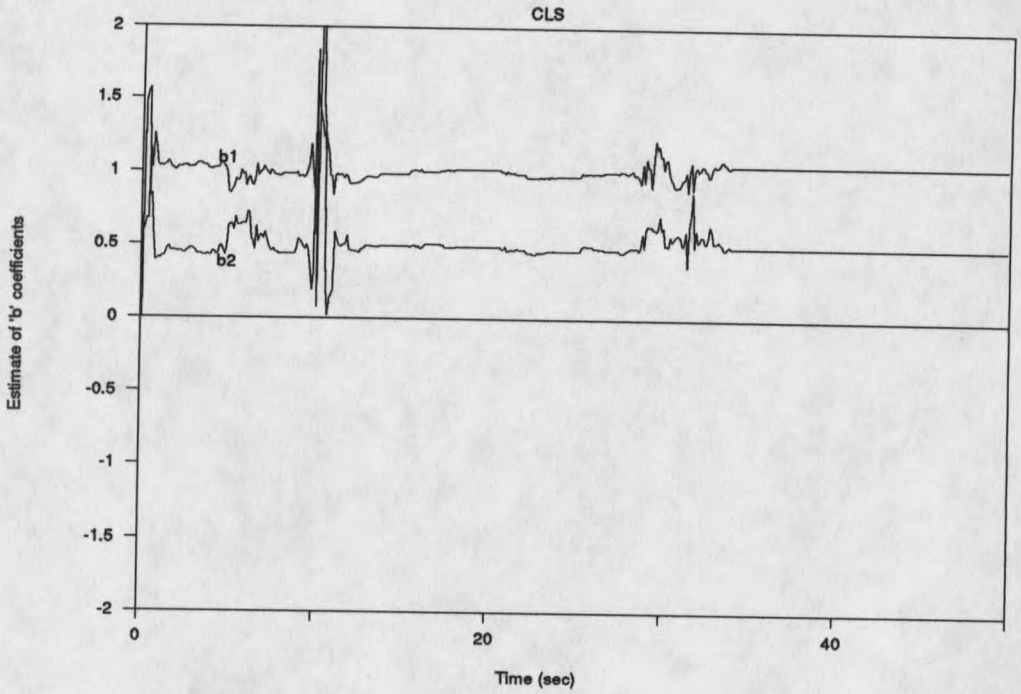

Figure 67. Estimate of the 'b' coefficients (CLS).

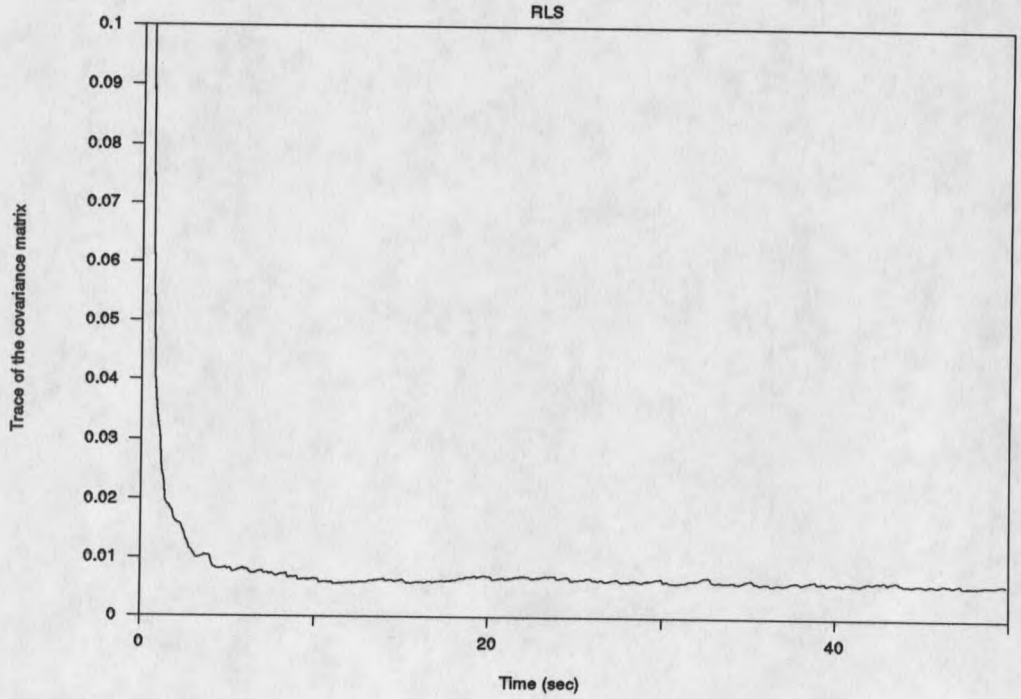

Figure 68. Trace of the covariance matrix (RLS).

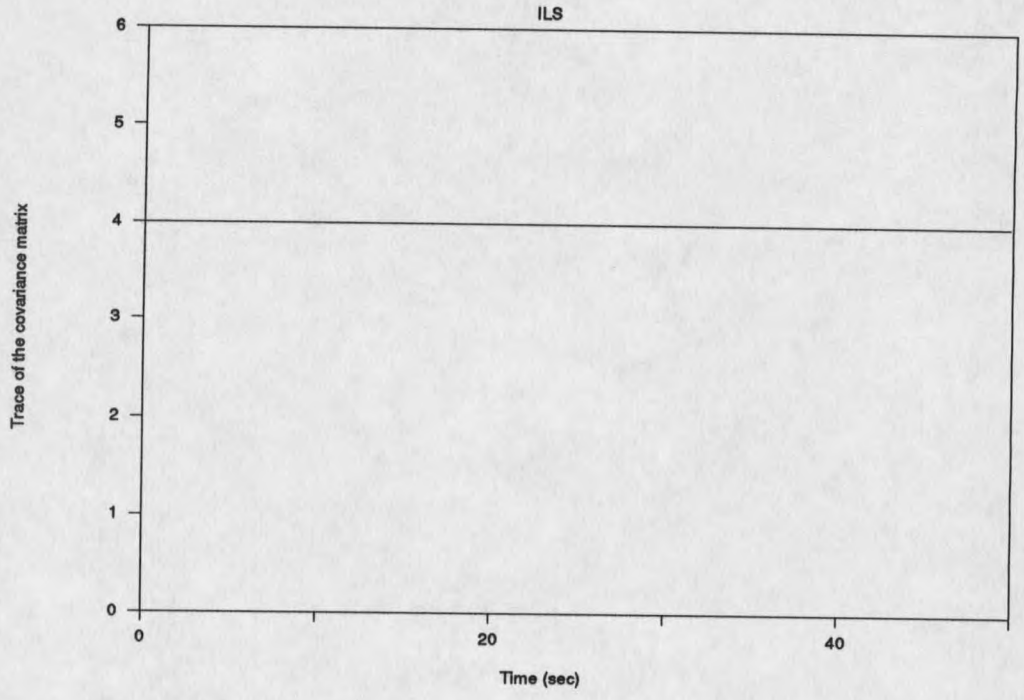

Figure 69. Trace of the covariance matrix (ILS).

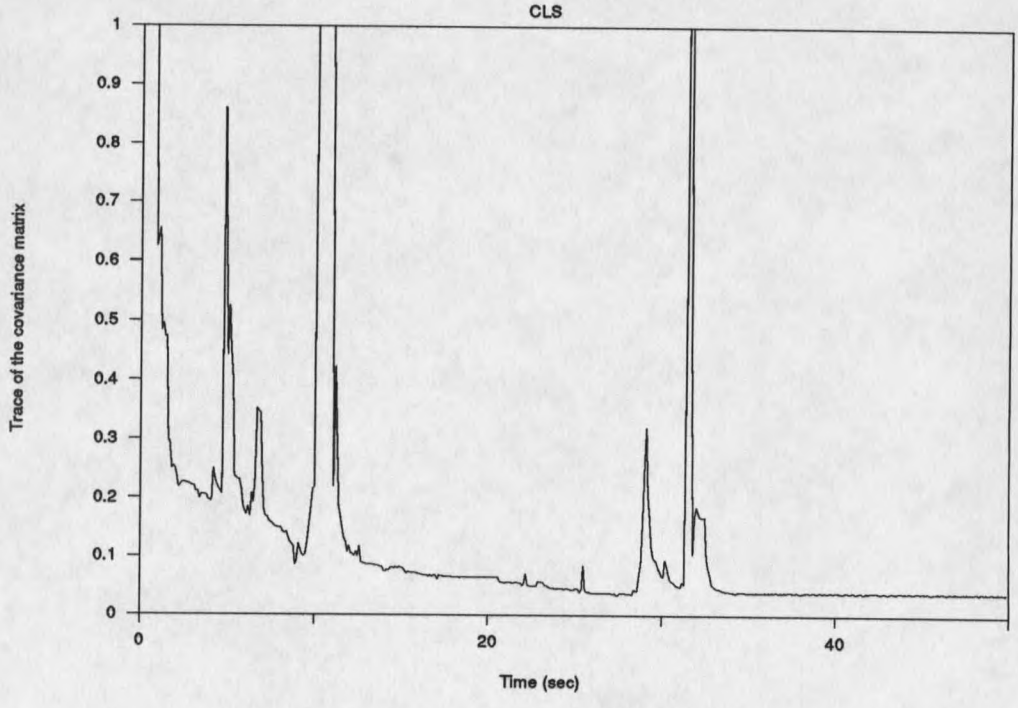

Figure 70. Trace of the covariance matrix (CLS).

Example 7

In this example we investigate the applicability of the parameter adaptive control algorithm to a nonlinear system.

Consider the inverted pendulum as shown in Figure 71. The differential equations describing the system are as follows:

$$(M+m)\ddot{y}_1 + ml\ddot{\theta}\cos(\theta) - ml\dot{\theta}^2\sin(\theta) = F \tag{9.10}$$

$$\ddot{y}_1\cos(\theta) + l\ddot{\theta} - g\sin(\theta) = 0 \tag{9.11}$$

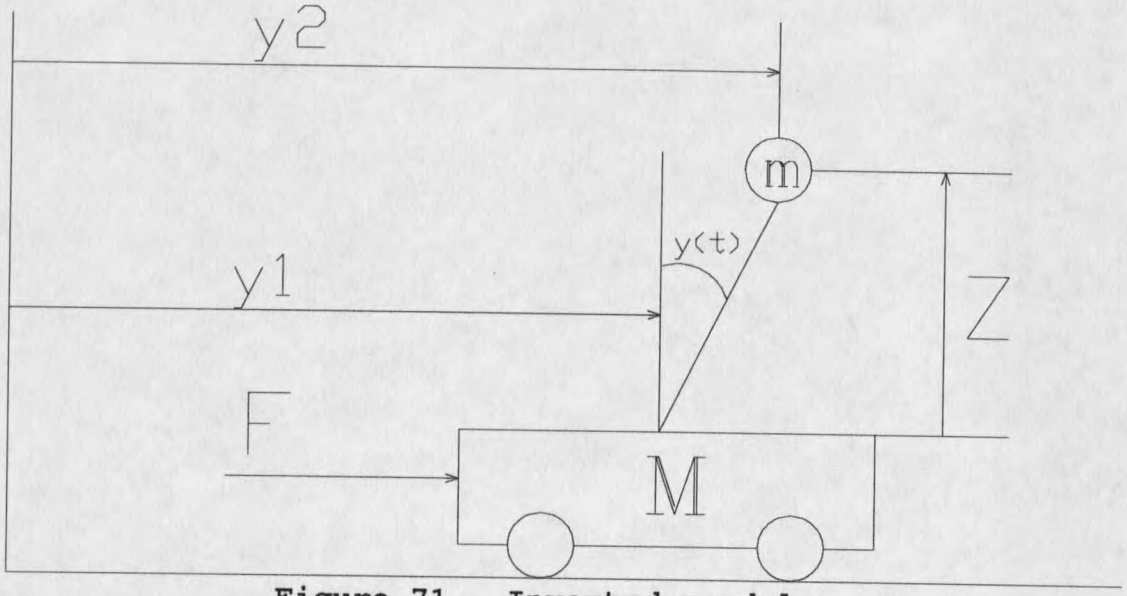

Figure 71. Inverted pendulum.

The variable y in Figure 71 is identified with θ of equations (9.10) and (9.11).

We use this nonlinear system in the closed loop configuration for the purpose of controlling the angle θ. Parameter values used are m=0.2, M=3.0, l=0.5, and g=9.81. At time t=0 we apply a pulse input (amplitude=45, duration=1 sample period) in order to set the system in motion, then we use a second-order identifier to build a model, on-line, for the system. It is assumed that θ and F are the only measurable quantities. After the model is estimated at each time instant, we use the LQR control procedure, as described in Chapter 8, to generate the control action F. Therefore the objective is to keep the angle θ as close to zero as possible, after setting the system in motion. The control action is further restricted to be between -60 and 60.

The simulation results are presented in Figures 72 through 83. Figures 72 through 77 present the measured outputs of the system and the control actions generated by the controller using the models produced by the RLS, ILS, and CLS algorithms, respectively. The generated model coefficients, using RLS, ILS, and CLS algorithms are presented in Figures 79 through 83, respectively. The sampling rate used was 20 samples per second except for the ILS algorithm which used 33 samples per second in order to obtain convergence.

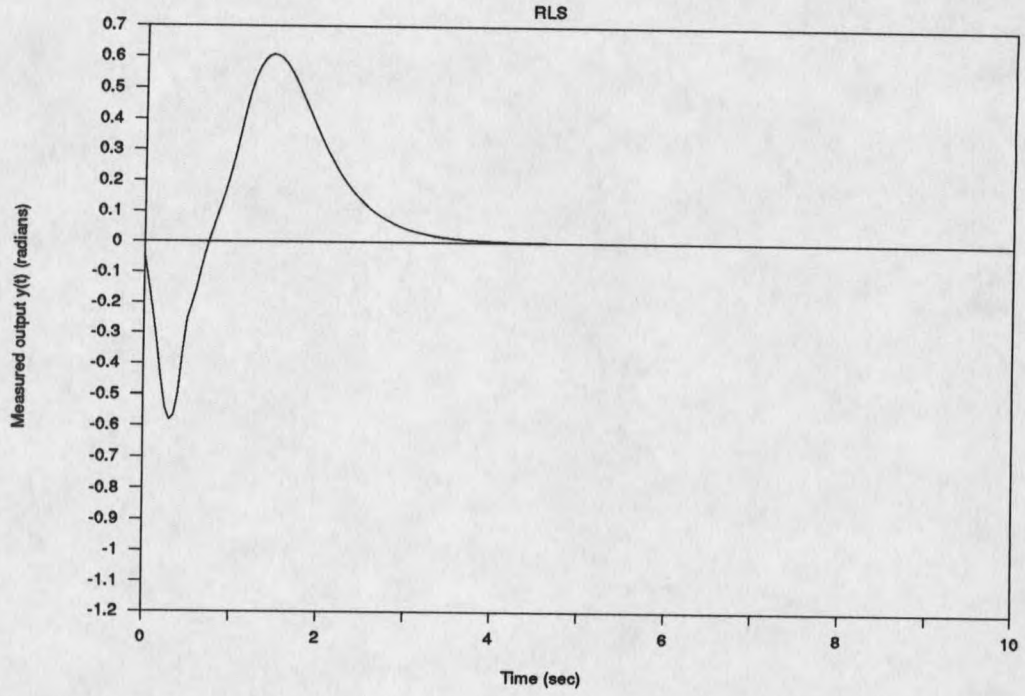

Figure 72. Measured output (RLS).

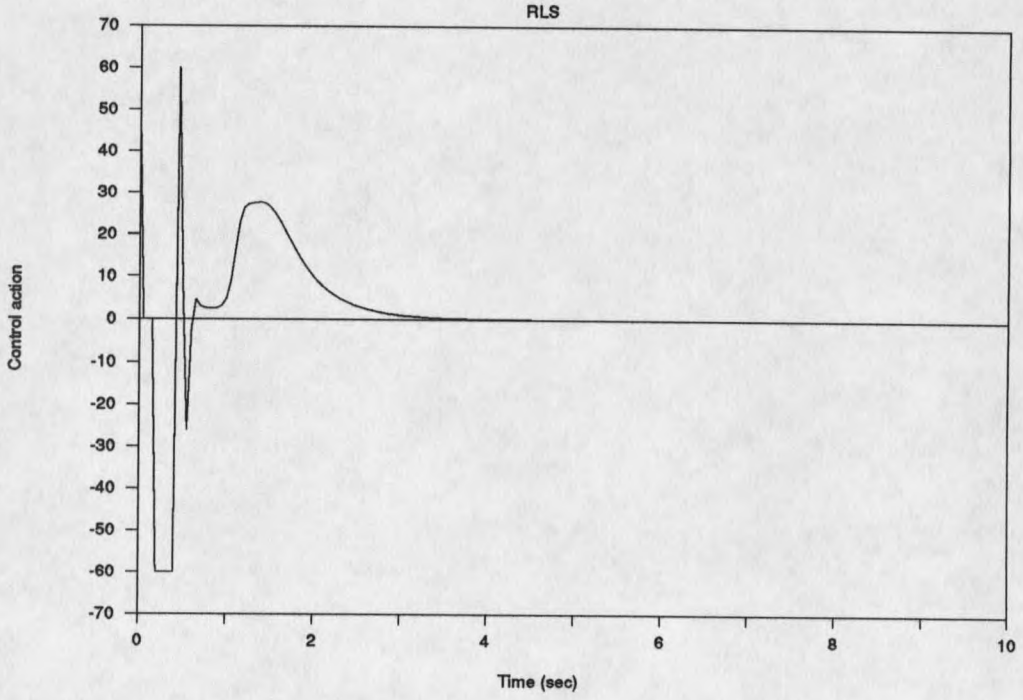

Figure 73. Control action (RLS).

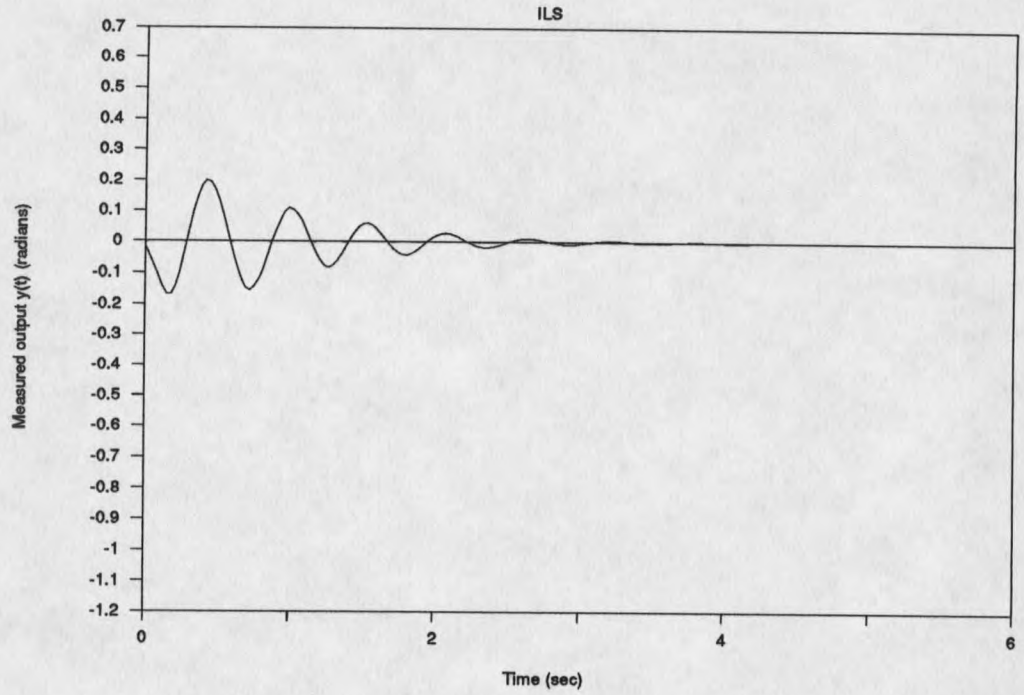

Figure 74. Measured output (ILS).

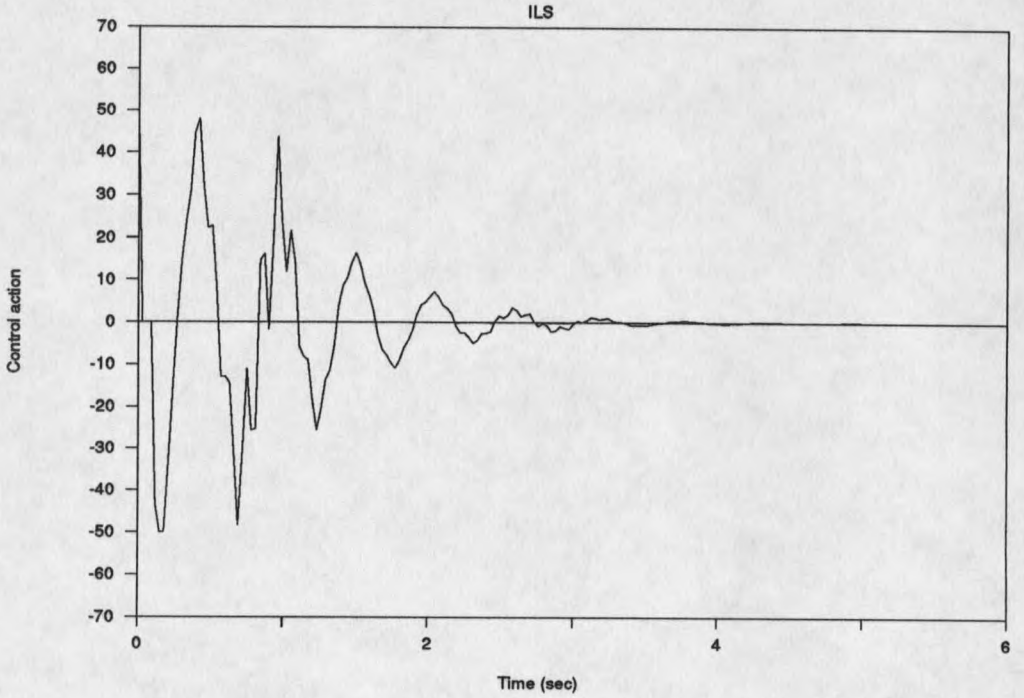

Figure 75. Control action (ILS).

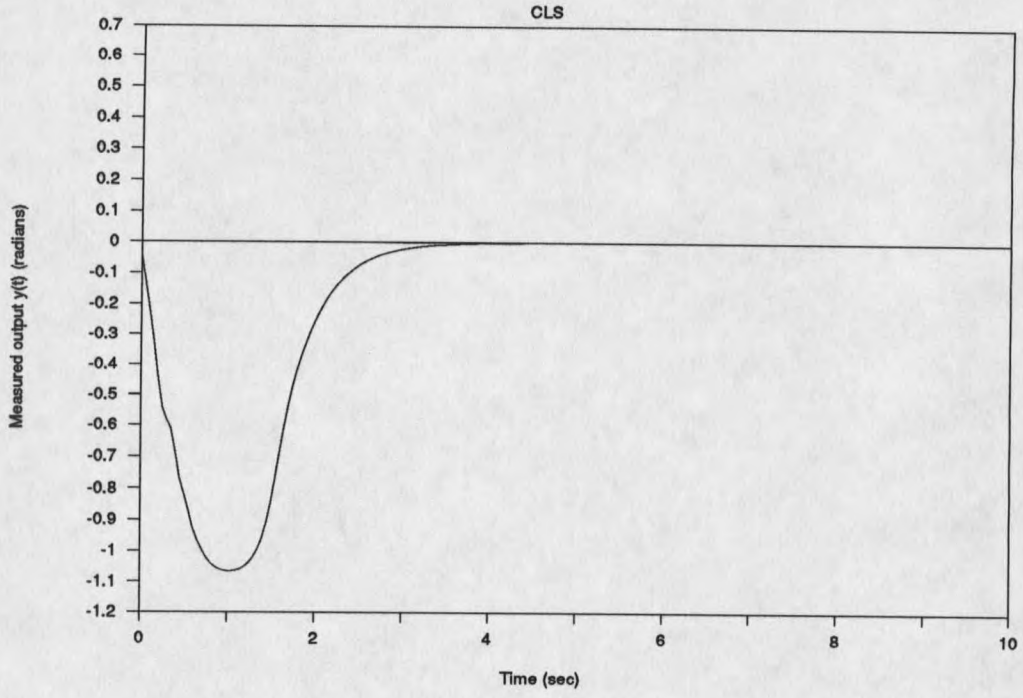

Figure 76. Measured output (CLS).

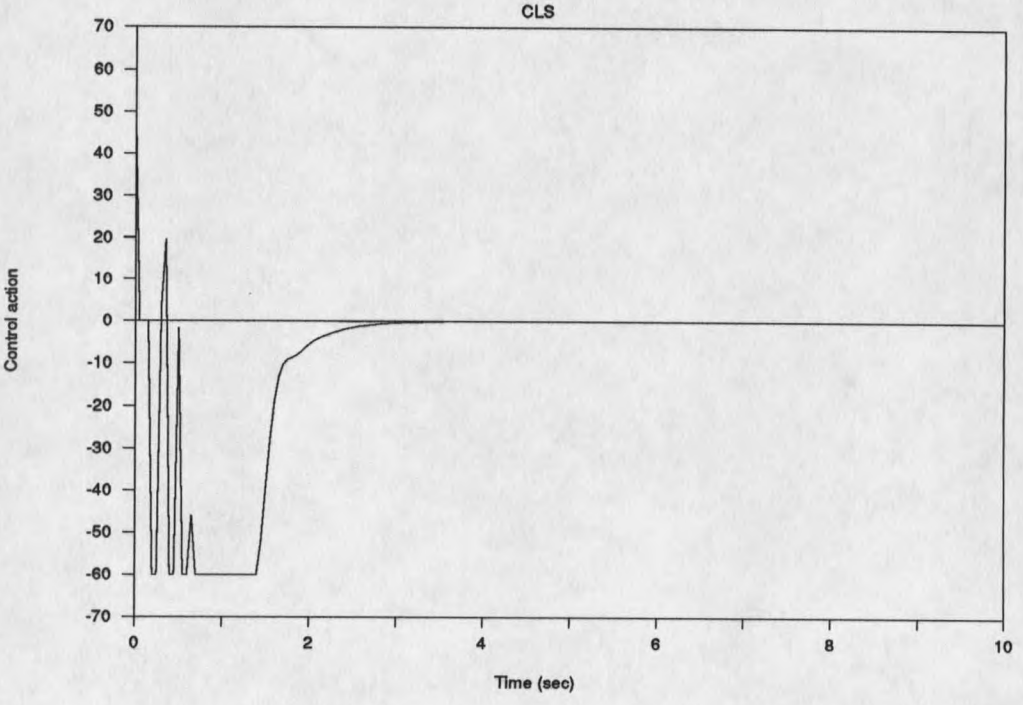

Figure 77. Control action (CLS).

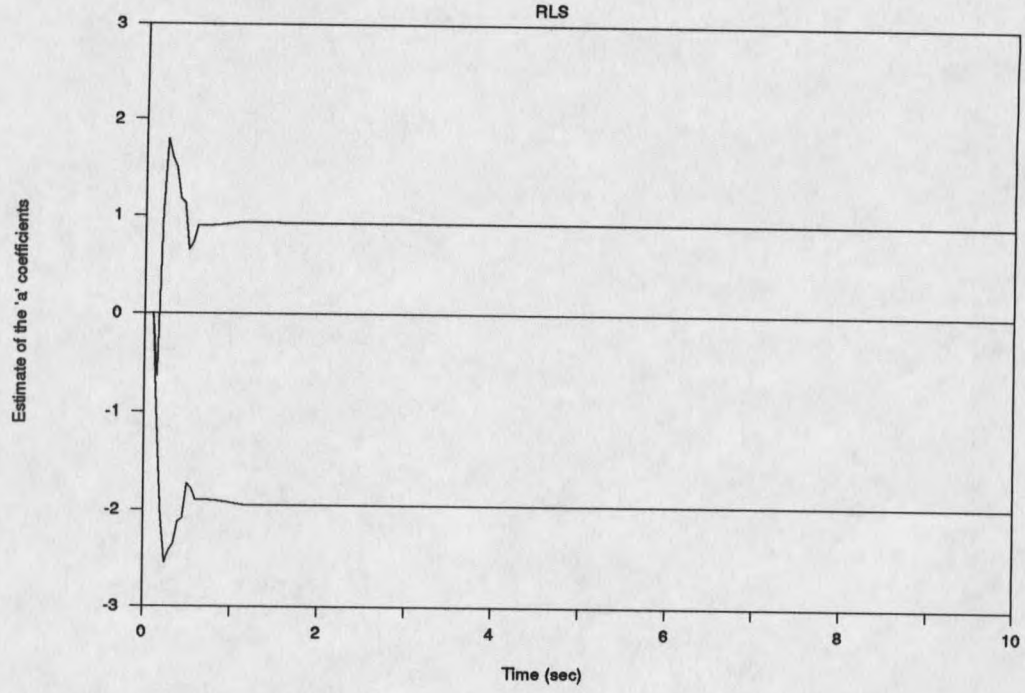

Figure 78. The 'a' coefficients (RLS).

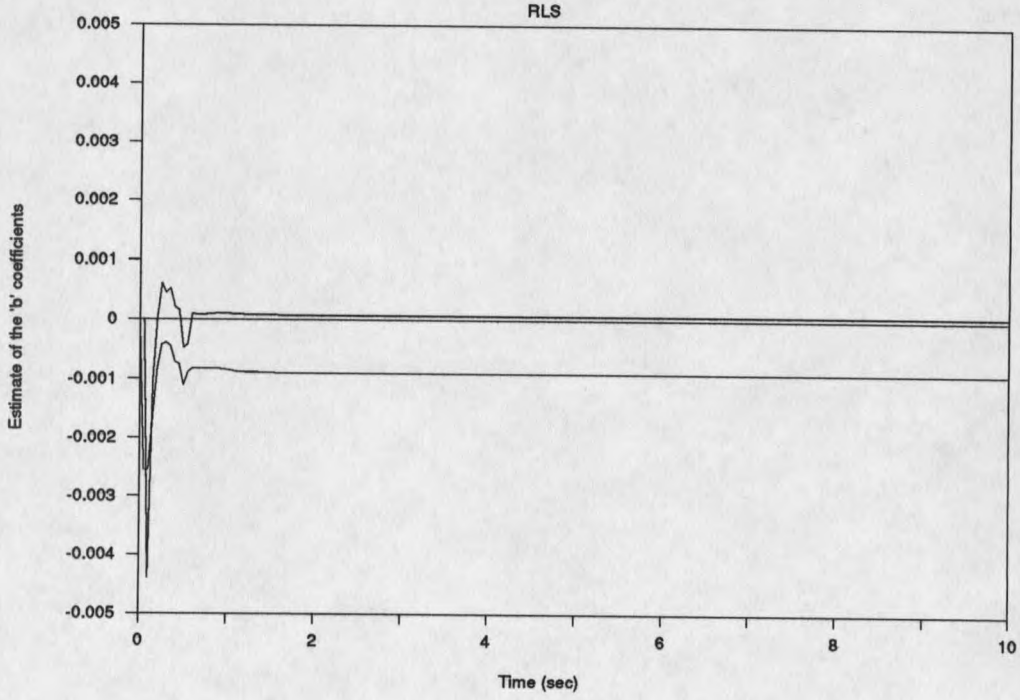

Figure 79. The 'b' coefficients (RLS).

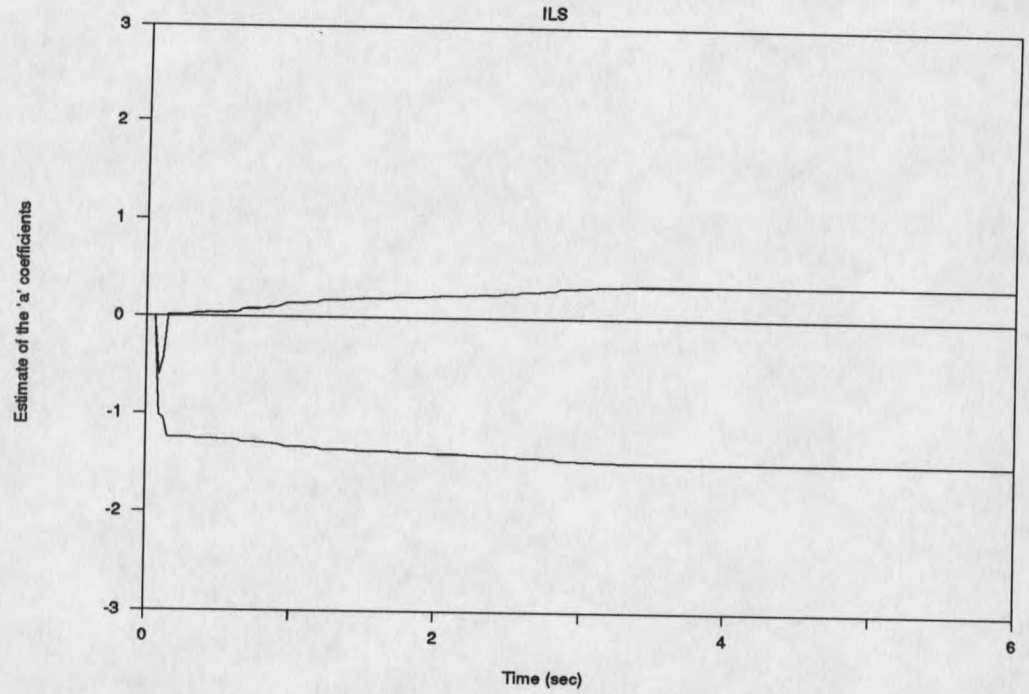

Figure 80. The 'a' coefficients (ILS).

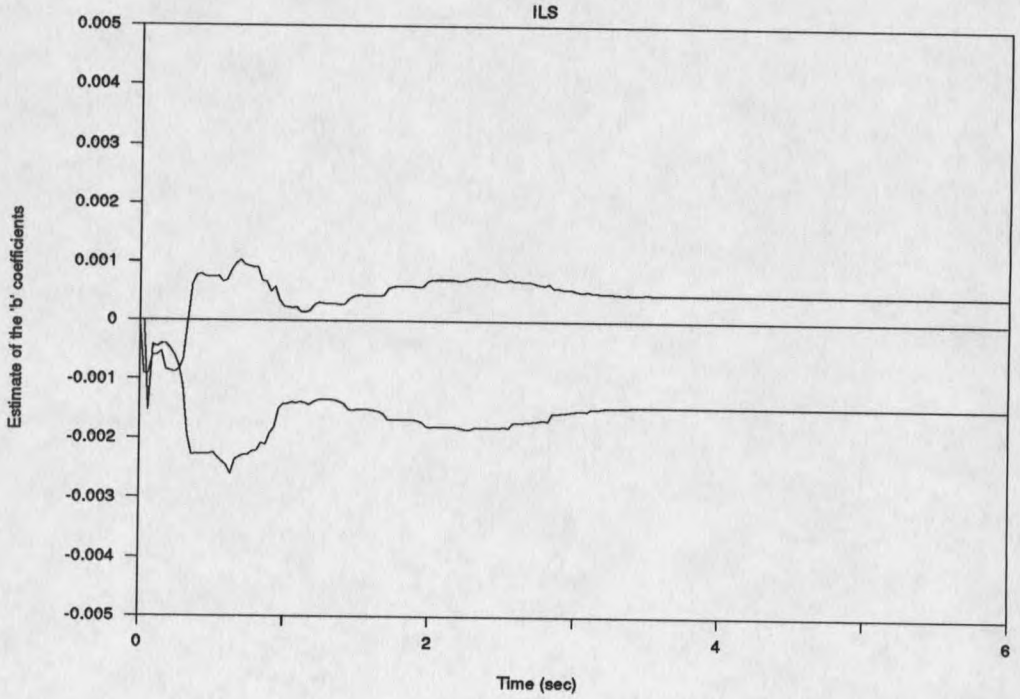

Figure 81. The 'b' coefficients (ILS).

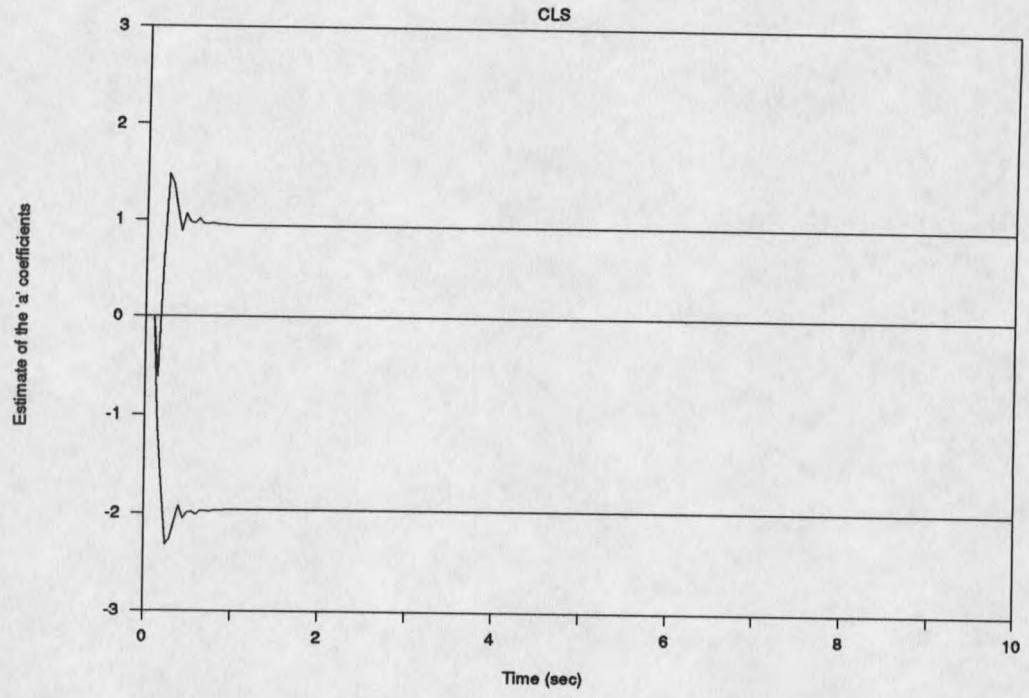

Figure 82. The 'a' coefficients (CLS).

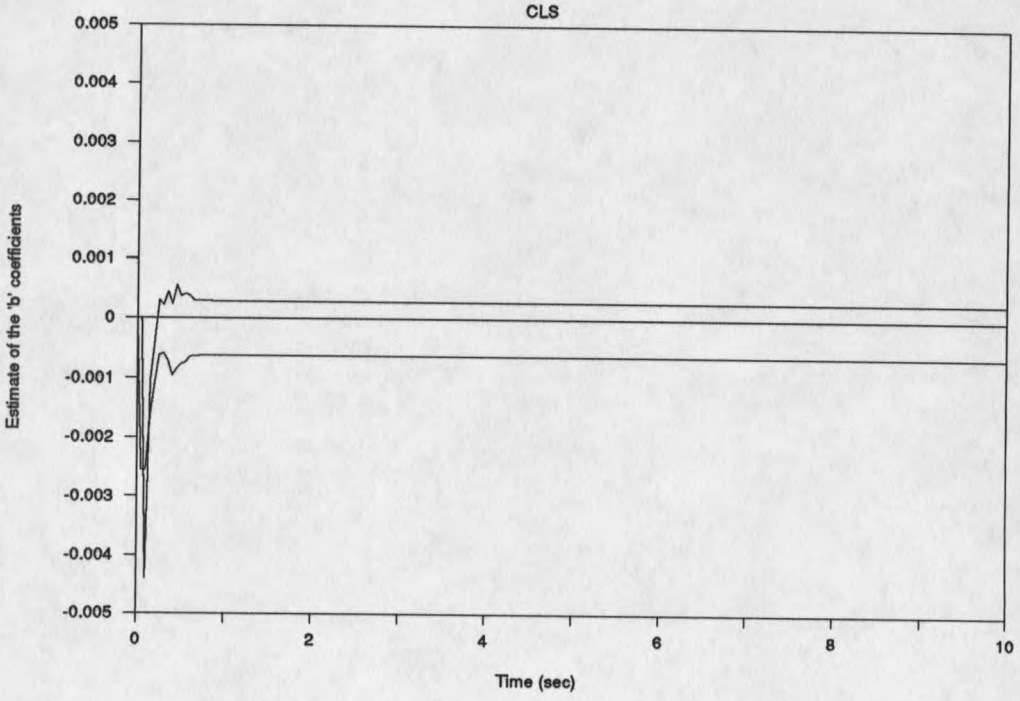

Figure 83. The 'b' coefficients (CLS).

Example 8

Consider the inverted pendulum as described in Example 7. In this example we investigate the effect of choosing a higher-order model for the identifier. For the purpose of this simulation, we use a fourth-order model for the identifier. The effect of the higher-order identifier should be clear from the simulation results, as presented in Figures 84 through 95.

We use this nonlinear system in the closed loop configuration for the purpose of controlling the angle θ. At time t=0 we apply a pulse input in order to set the system in motion, then we use a fourth-order identifier to build a model, on-line, for the system. It is assumed that the θ and F are the only measurable quantities. After the model is estimated at each time instant, we use the LQR control procedure, as described in Chapter 8, to generate the control action F. Therefore the objective is to keep the angle θ as close to zero as possible, after setting the system in motion.

The simulation results are presented in Figures 84 through 95. Figures 84 through 89 present the measured outputs of the system and the control actions generated by the controller using the model produced by the RLS, ILS, and CLS algorithms,

respectively. The generated model coefficients, using RLS, ILS, and CLS algorithms are presented in Figures 90 through 95, respectively.

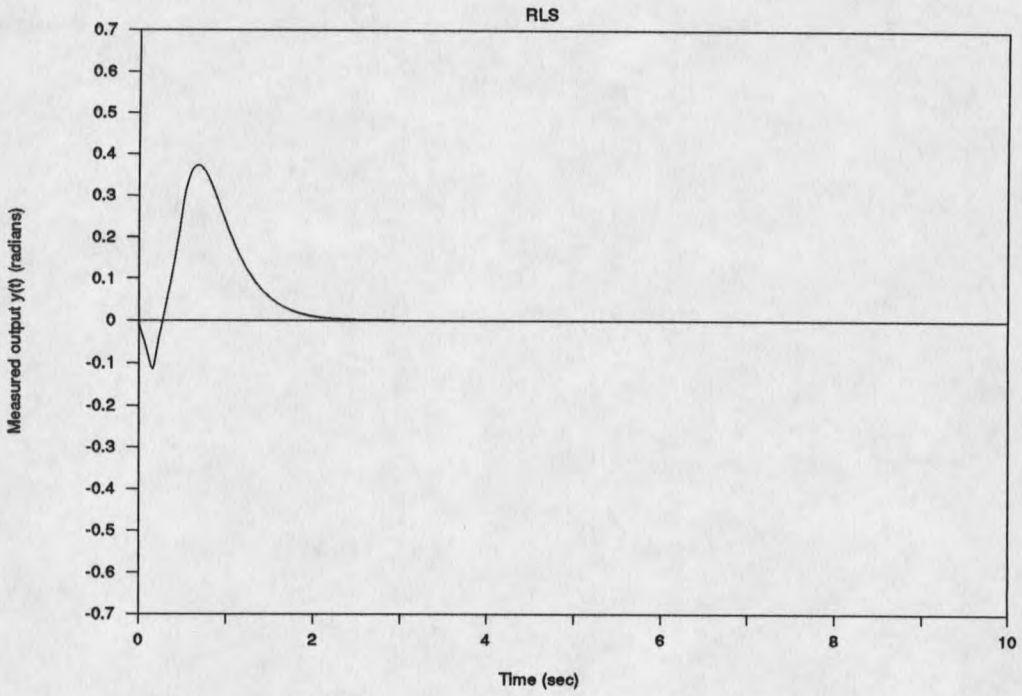

Figure 84. Measured output (RLS).

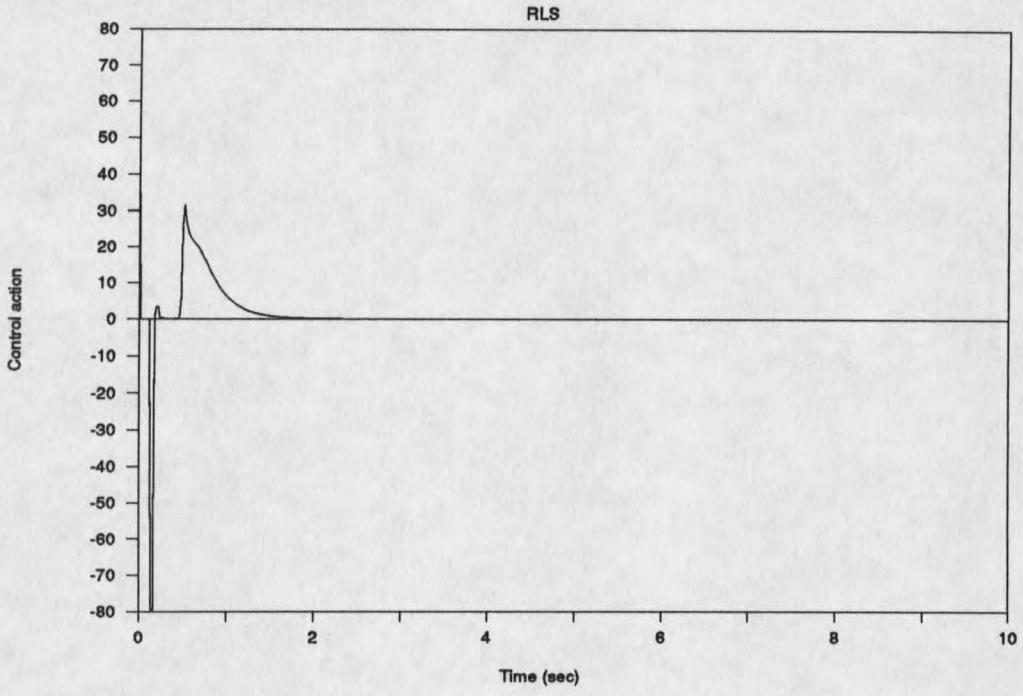

Figure 85. Control action (RLS).

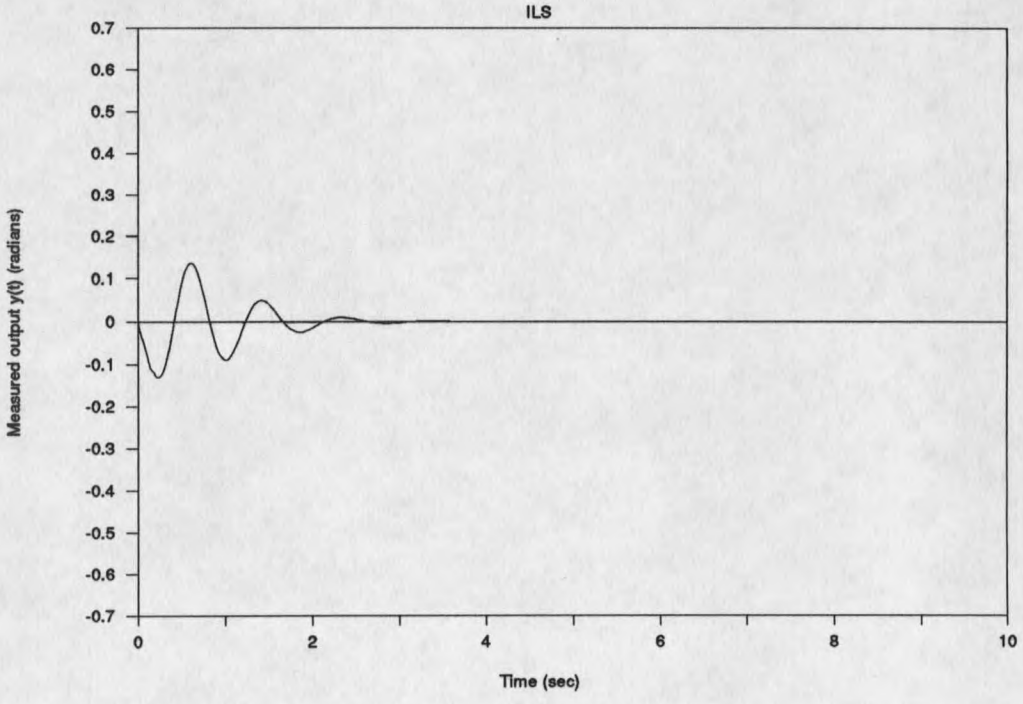

Figure 86. Measured output (ILS).

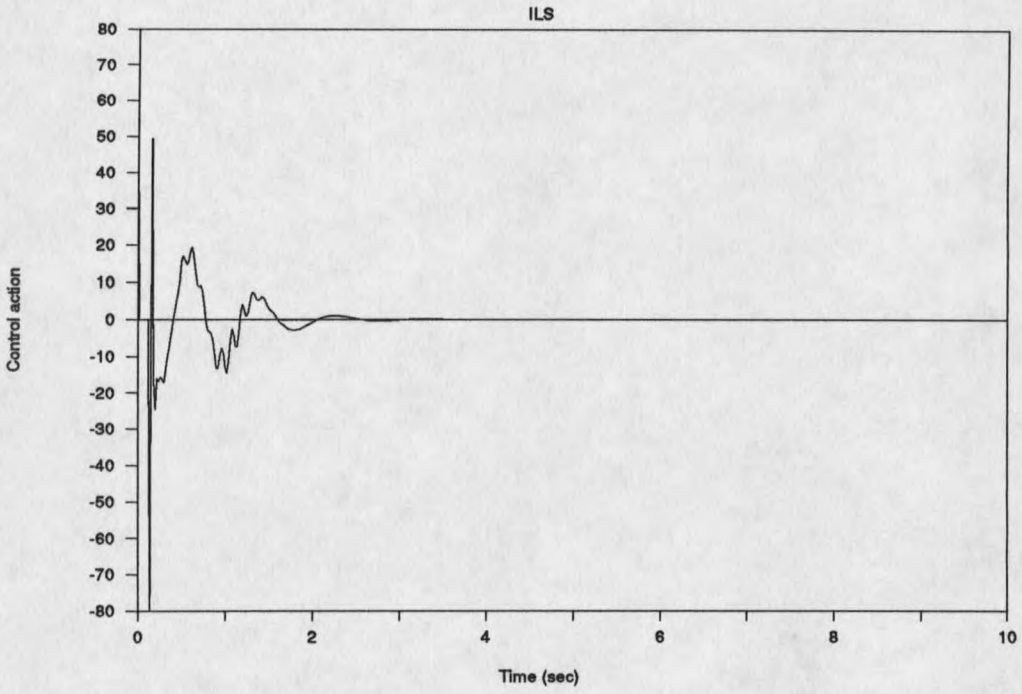

Figure 87. Control action (ILS).

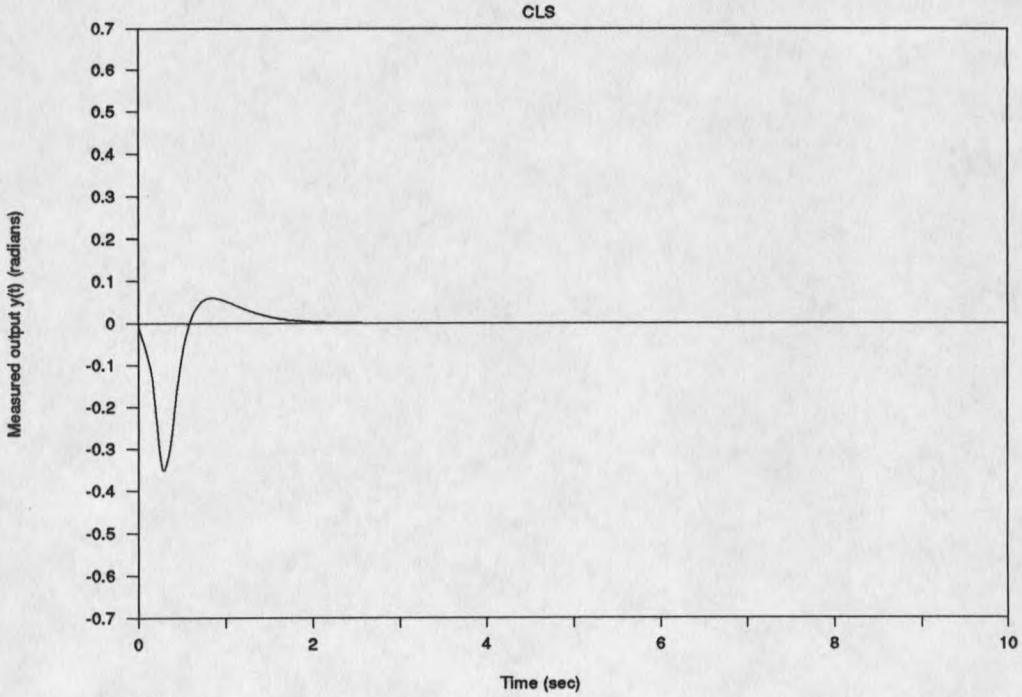

Figure 88. Measured output (CLS).

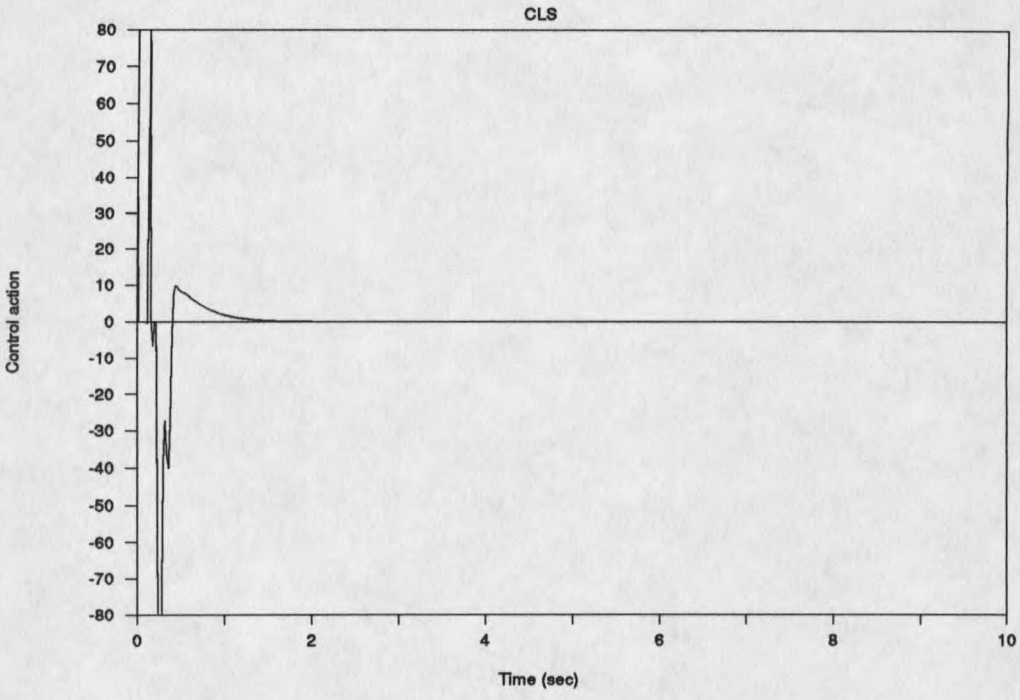

Figure 89. Control action (CLS).

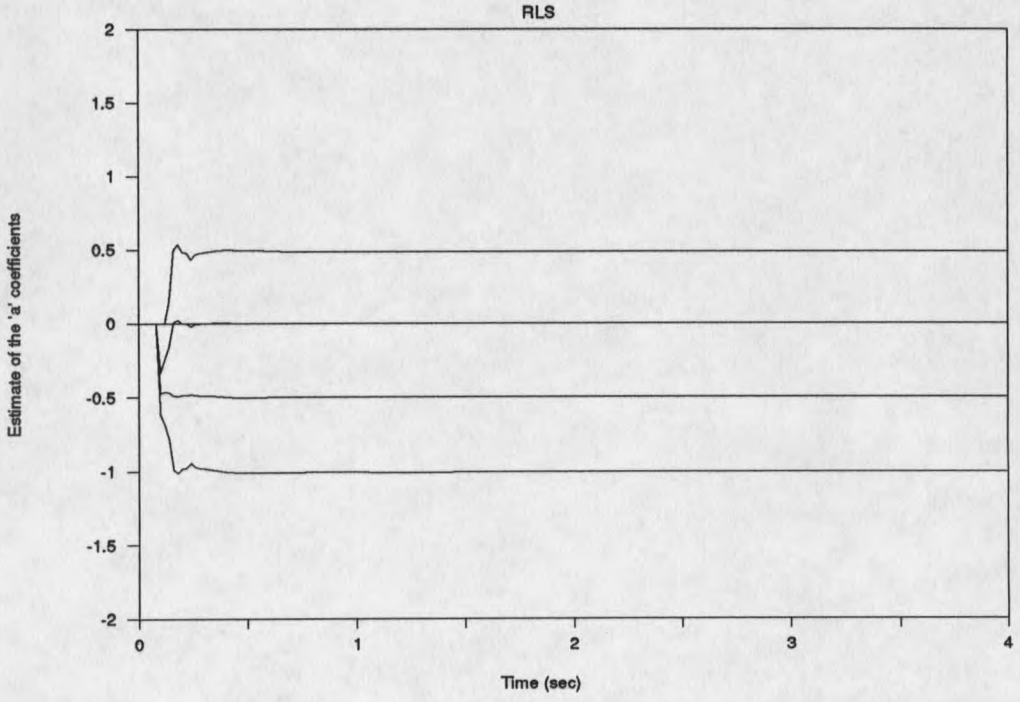

Figure 90. The 'a' coefficients (RLS).

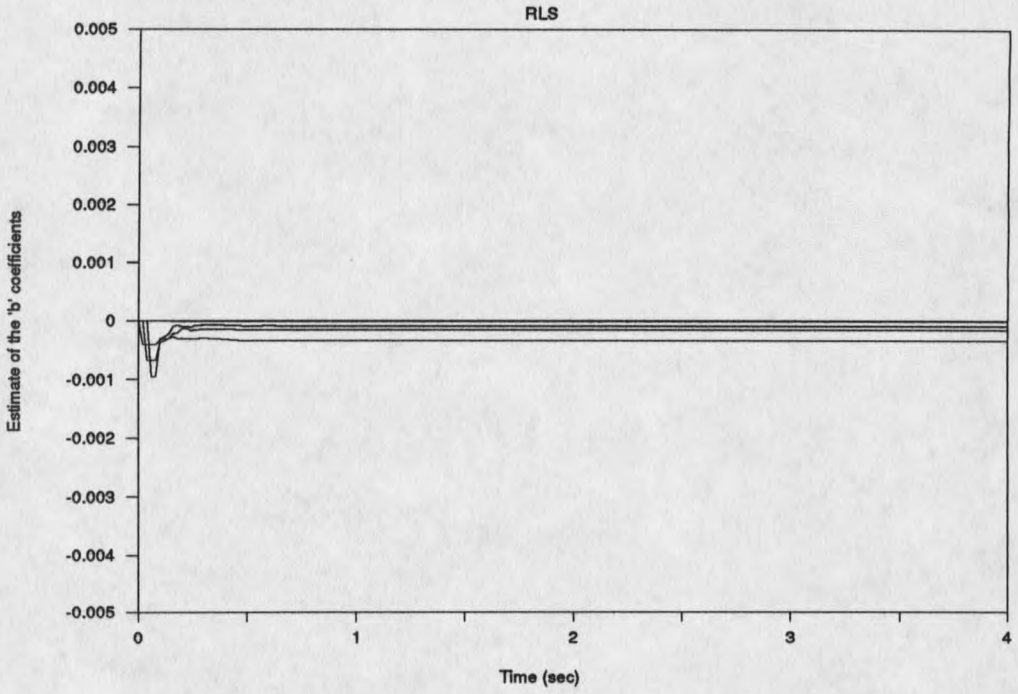

Figure 91. The 'b' coefficients (RLS).

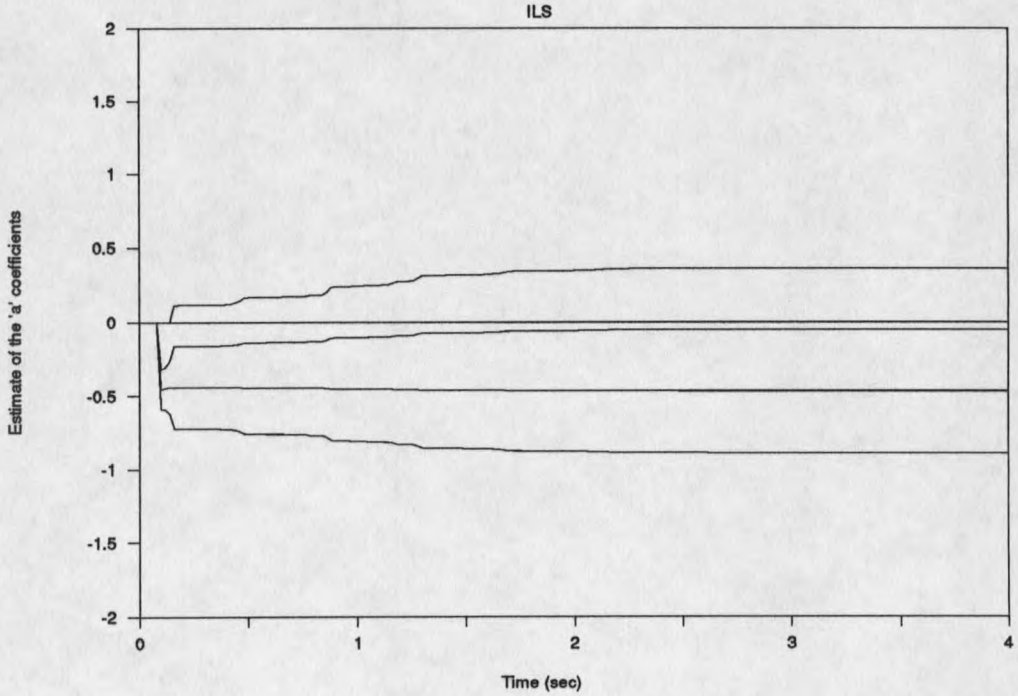

Figure 92. The 'a' coefficients (ILS).

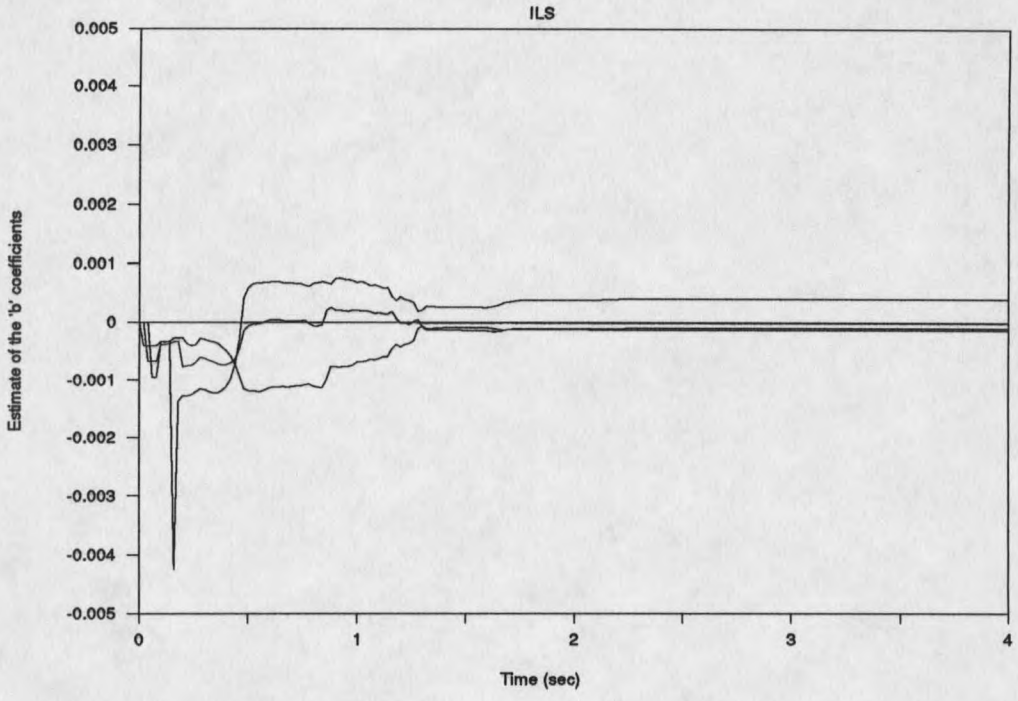

Figure 93. The 'b' coefficients (ILS).

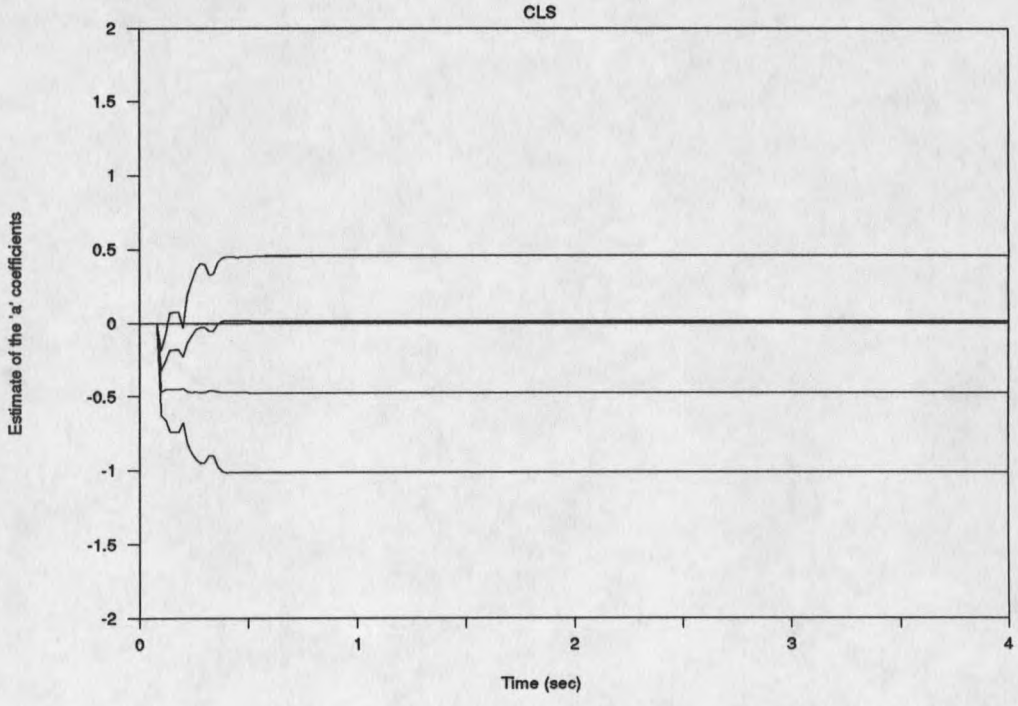

Figure 94. The 'a' coefficients (CLS).

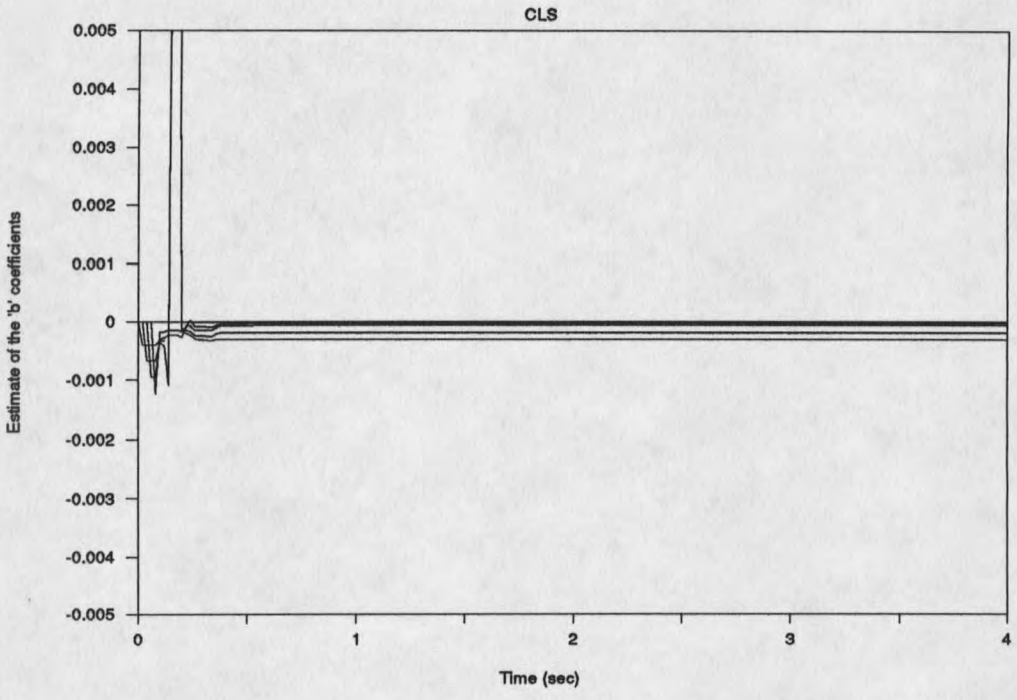

Figure 95. The 'b' coefficients (CLS).

Example 9

This example basically is the same as Example 8, except that Gaussian white noise with zero mean and $\sigma^2 = 0.01$ is added to the output measurements.

The simulation results are presented in Figures 96 through 101. Figures 96 through 99 present the measured outputs of the system and the control actions generated by the controller using the model produced by the RLS and CLS algorithms, respectively. The generated models 'a' coefficients, using RLS and CLS algorithms are presented in Figures 100 and 101, respectively.

As can be seen from the simulation results the control action produced by using the model generated by the RLS algorithm fails to return the angle to zero. On the other hand, the control action produced by using the model generated by CLS works well and returns the angle to zero. This is because CLS uses the richness test to stop the estimation and therefore to produce better coefficients.

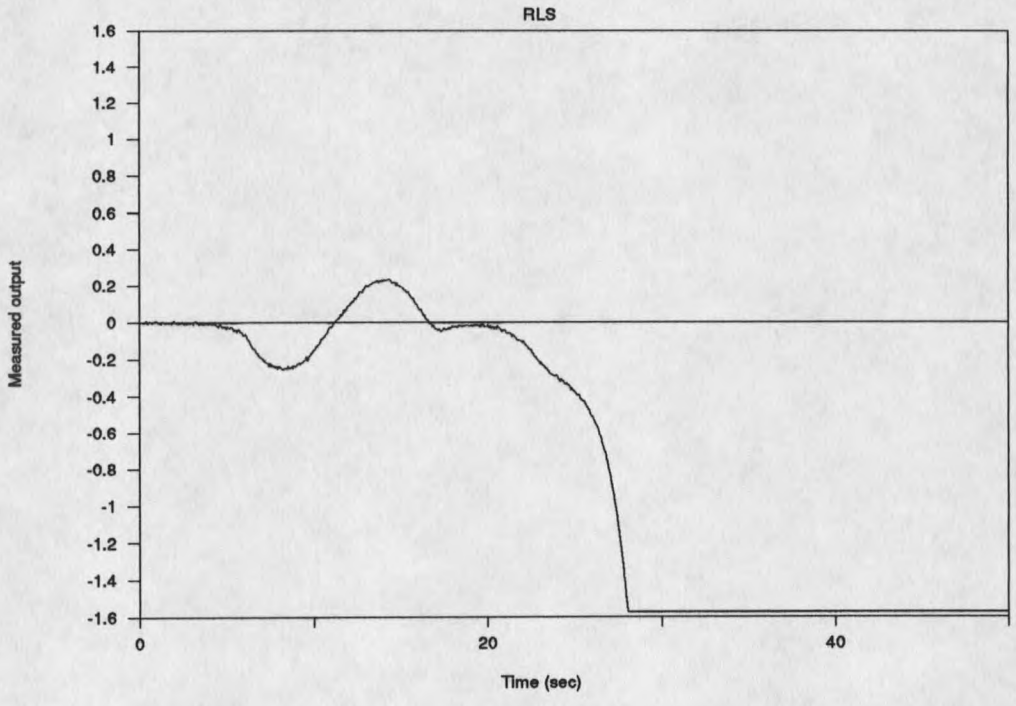

Figure 96. Measured output (RLS).

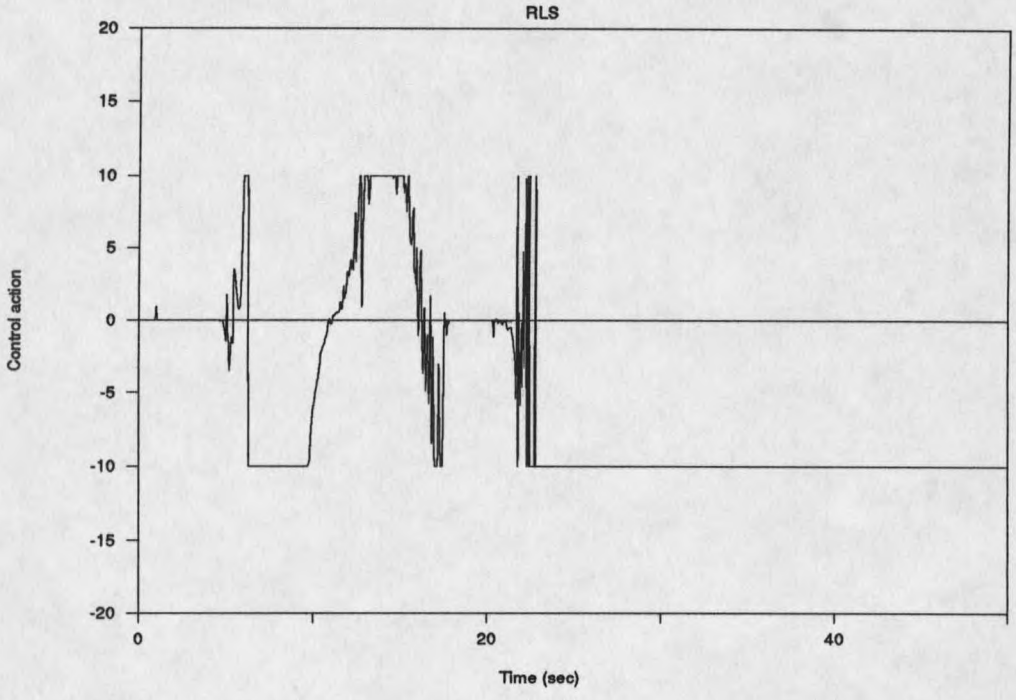

Figure 97. Control action (RLS).

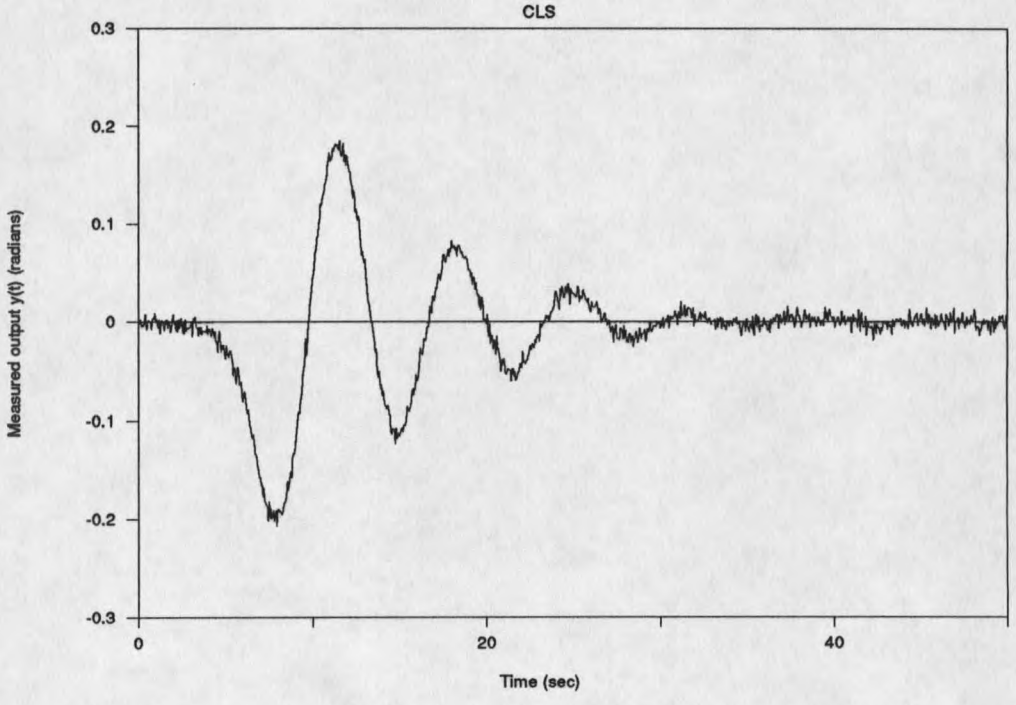

Figure 98. Measured output (CLS).

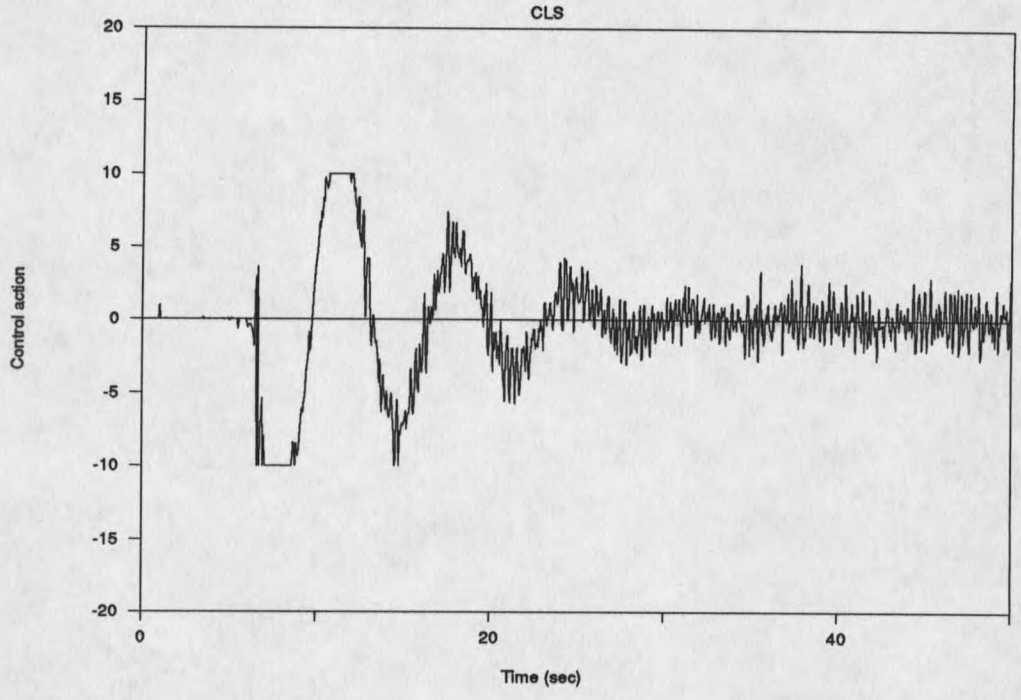

Figure 99. Control action (CLS).

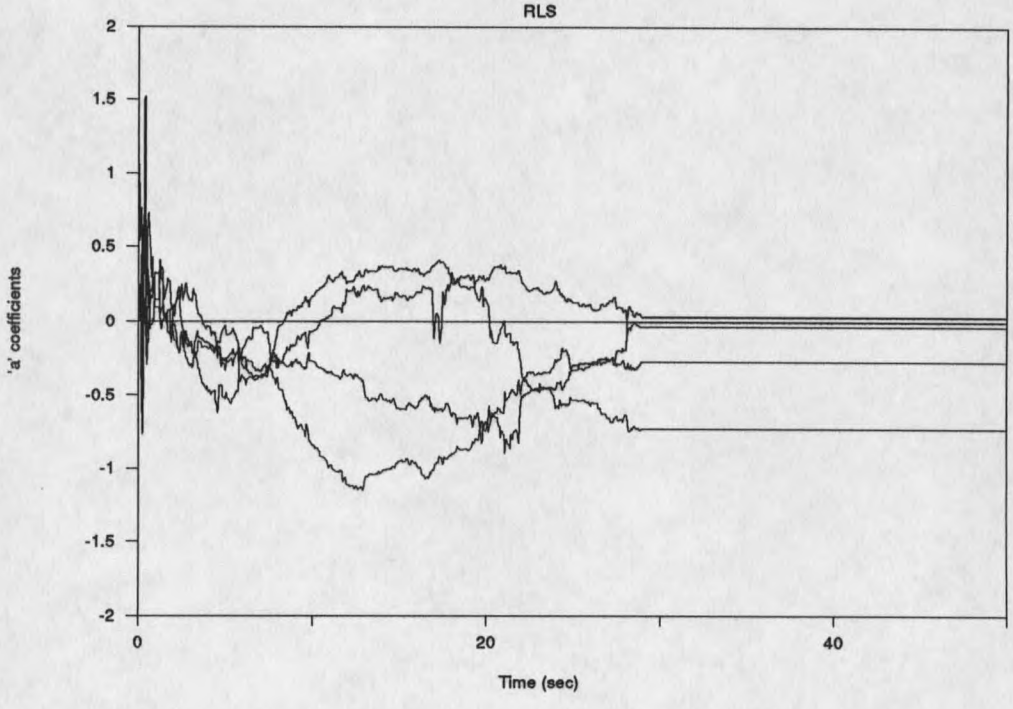

Figure 100. The 'a' coefficients (RLS).

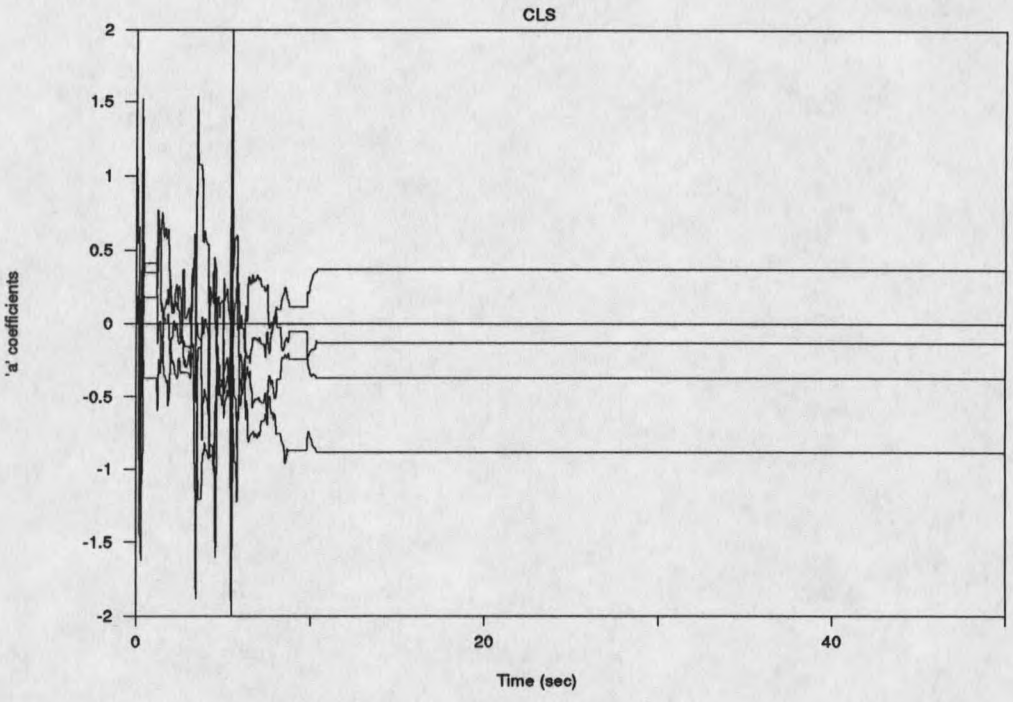

Figure 101. The 'a' coefficients (CLS).

Example 10

This example basically is the same as Example 8, except that Gaussian white noise with zero mean and $\sigma^2 = 0.05$ is added to the output measurements.

The results from this simulation are shown in Figures 102 through 104.

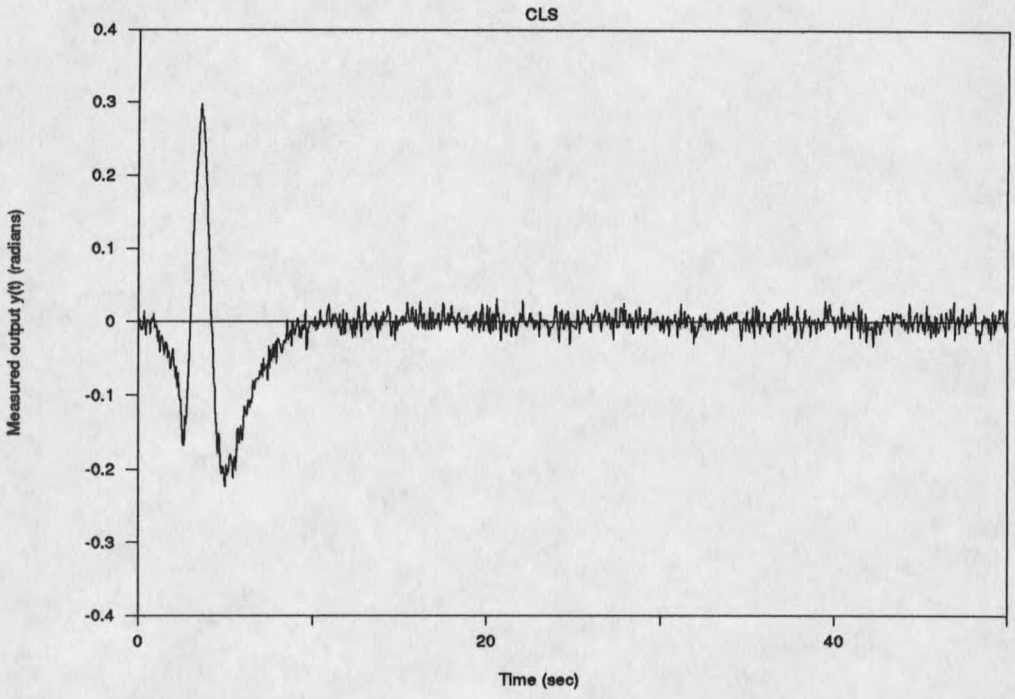

Figure 102. Measured output (CLS).

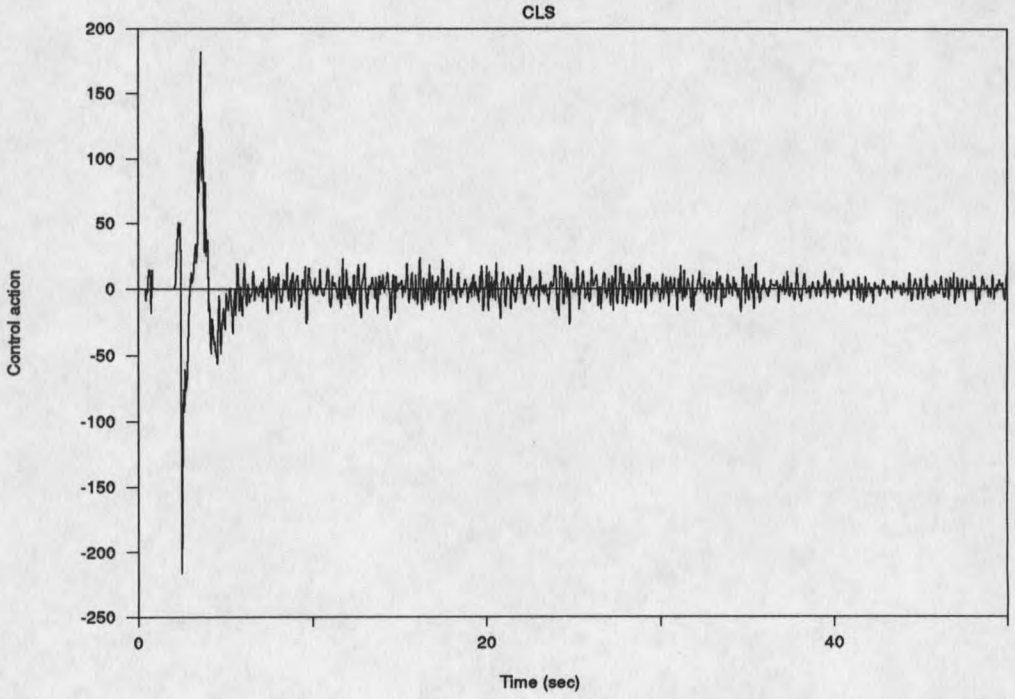

Figure 103. Control action (CLS).

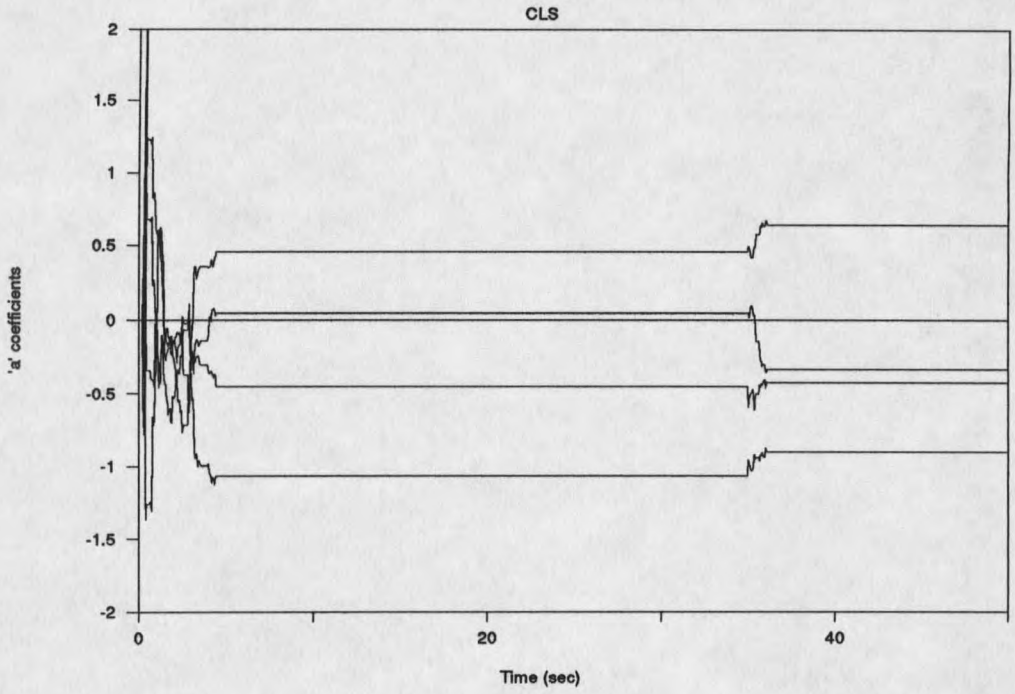

Figure 104. The 'a' coefficients (CLS).

Example 11

In this example, we present computer-simulation results to illustrate the effect of adding a known signal (in particular, a pseudo-random square wave) to the input signal. For the purpose of the simulation we use the basic RLS algorithm for estimating the unknown parameters.

Consider the following fifth-order process:

$$y(t) = \frac{0.3461z^{-1} - 1.361z^{-2} + 2.03z^{-3} - 1.361z^{-4} + 0.3461z^{-5}}{1.0 - 3.77256z^{-1} + 5.521138z^{-2} - 3.8658z^{-3} + 1.271z^{-4} - 0.1533z^{-5}} u(t)$$

where

$u(t) = \sin(0.05t) \pm v$, where + or - choosen at random and v=constant.

The sampling period is 0.1 sec. The results of identification for this system are given in Figures 105 through 116. Figure 105 presents the measured output. Figures 106, 107, and 108 present the measured input for the v=0, v=0.005, and v=0.05, respectively. Figures 108 and 110 give a portion of the input signal for v=0.005 and v=0.05, respectively. Figures 111 through 113 give the estimate of 'a' parameters obtained during the simulations. The traces of the covariance matrices obtained during the simulations are presented in Figures 114 through 116, for v=0, v=0.005, and v=0.05, respectively.

The simulation results clearly show that the addition of a pseudo-random square wave to the input signal enables better parameter identification even for the v=0.005 case which is considerably smaller than the actual amplitude of the input signal (i.e. -5 to 5).

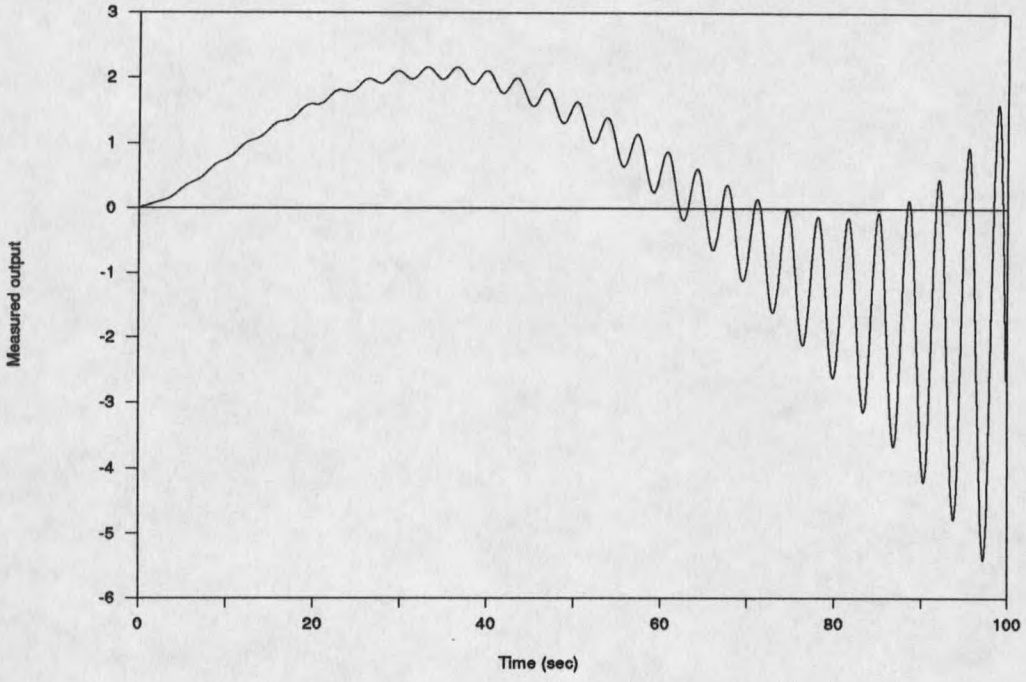

Figure 105. Measured output.

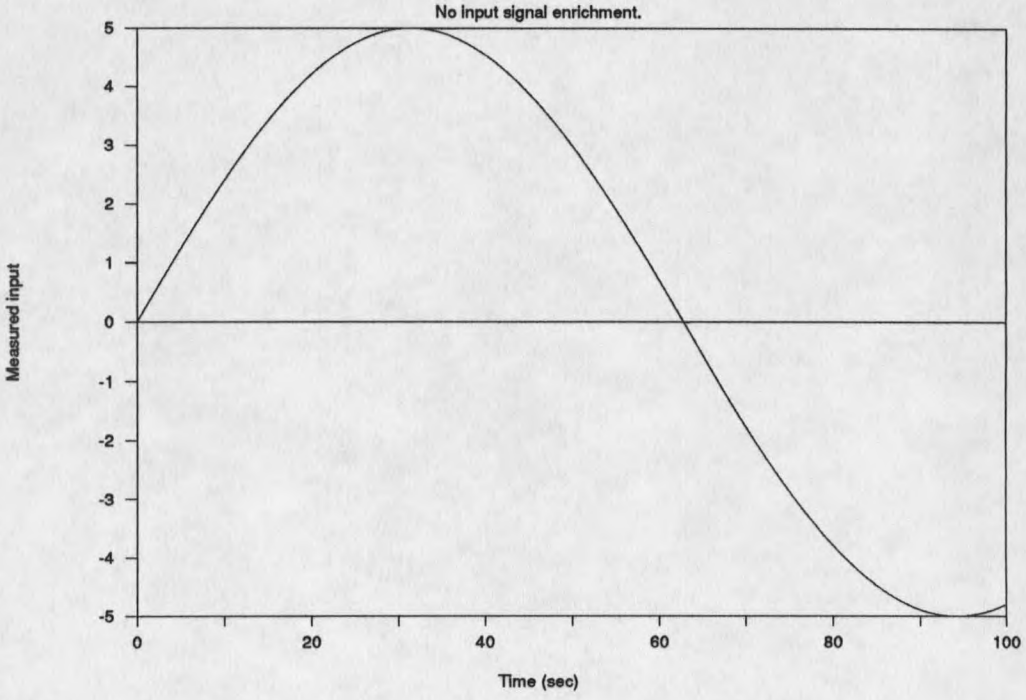

Figure 106. Measured input (v=0).

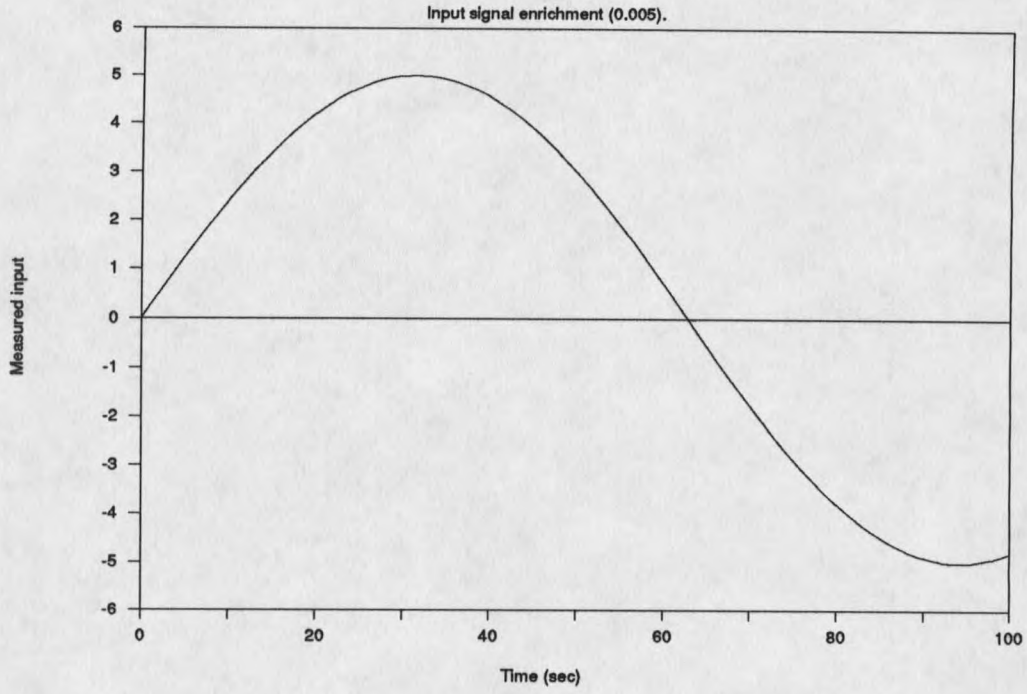

Figure 107. Measured input (v=0.005).

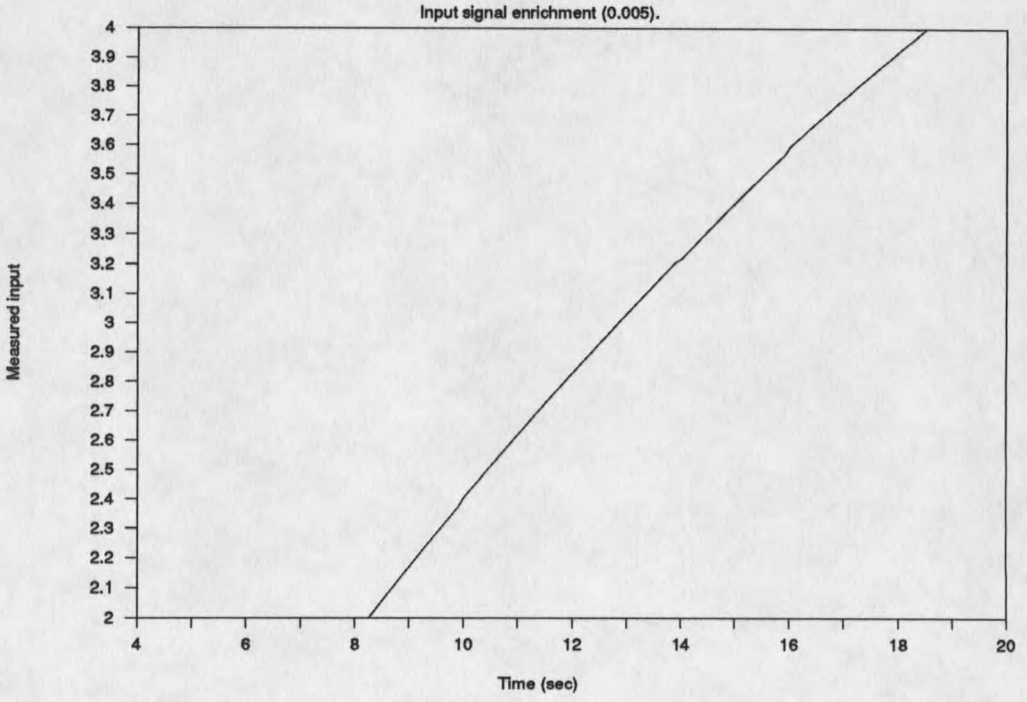

Figure 108. Measured input (v=0.005).

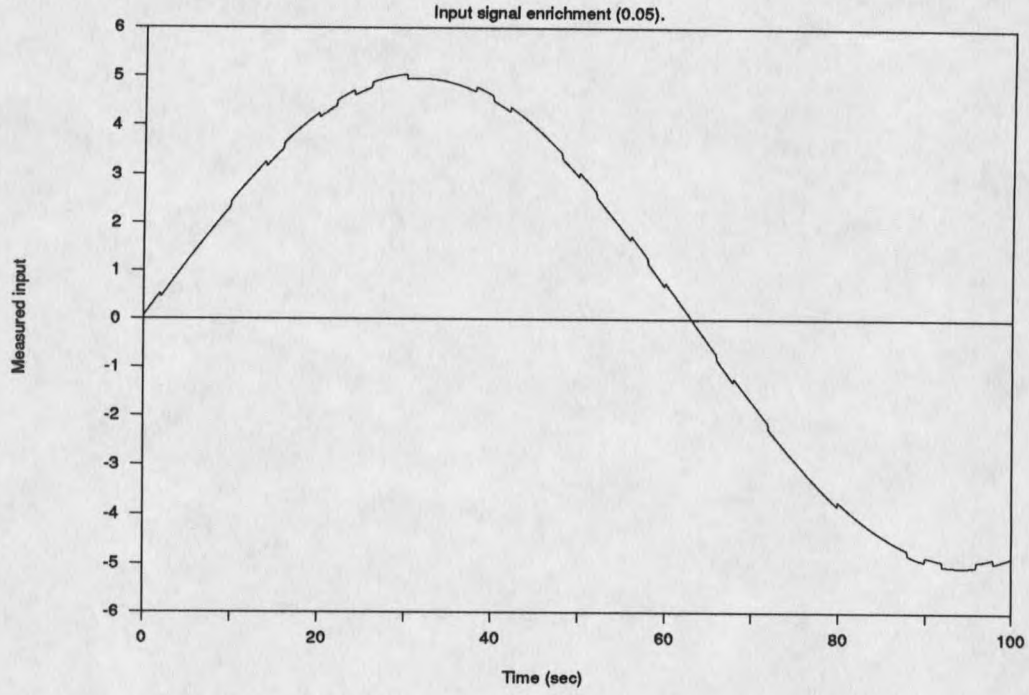

Figure 109. Measured input (v=0.05).

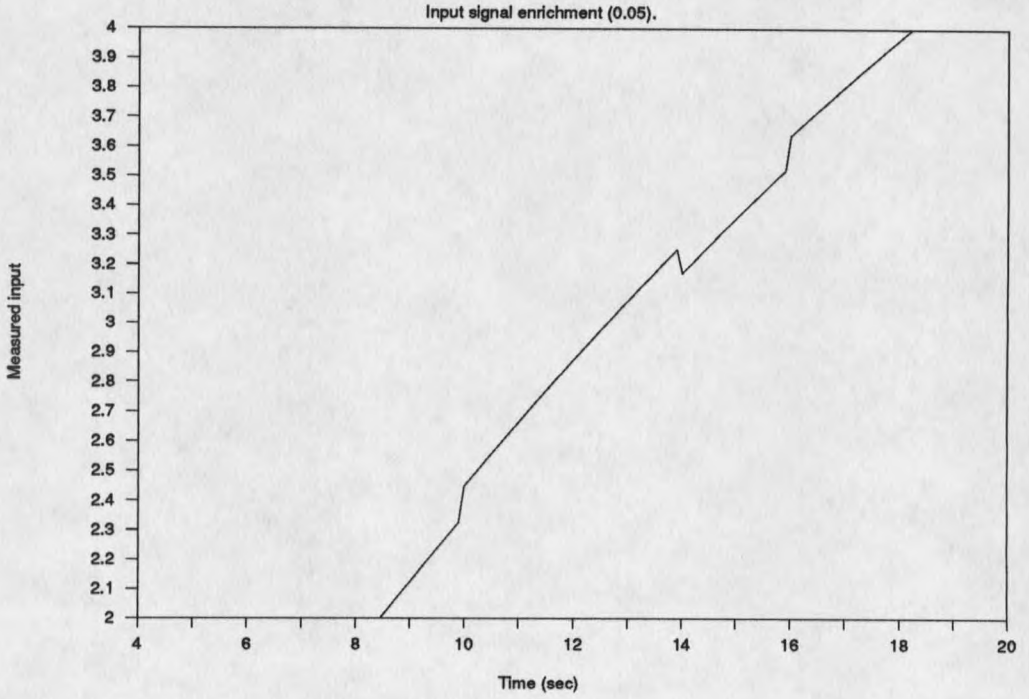

Figure 110. Measured input (v=0.05).

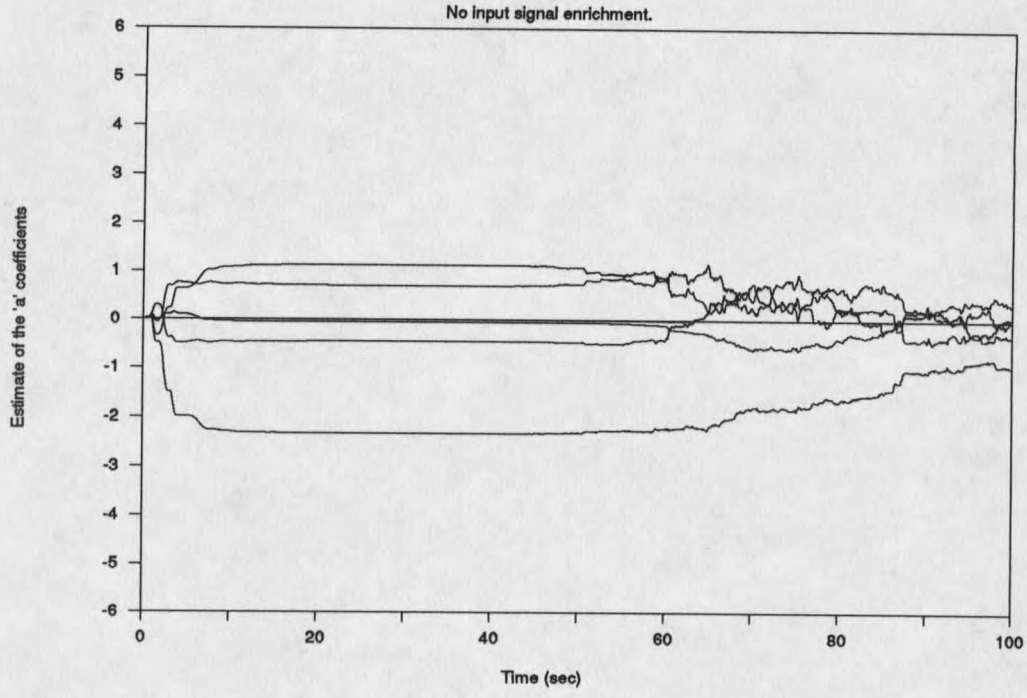

Figure 111. Estimate of the 'a' coefficients (v=0).

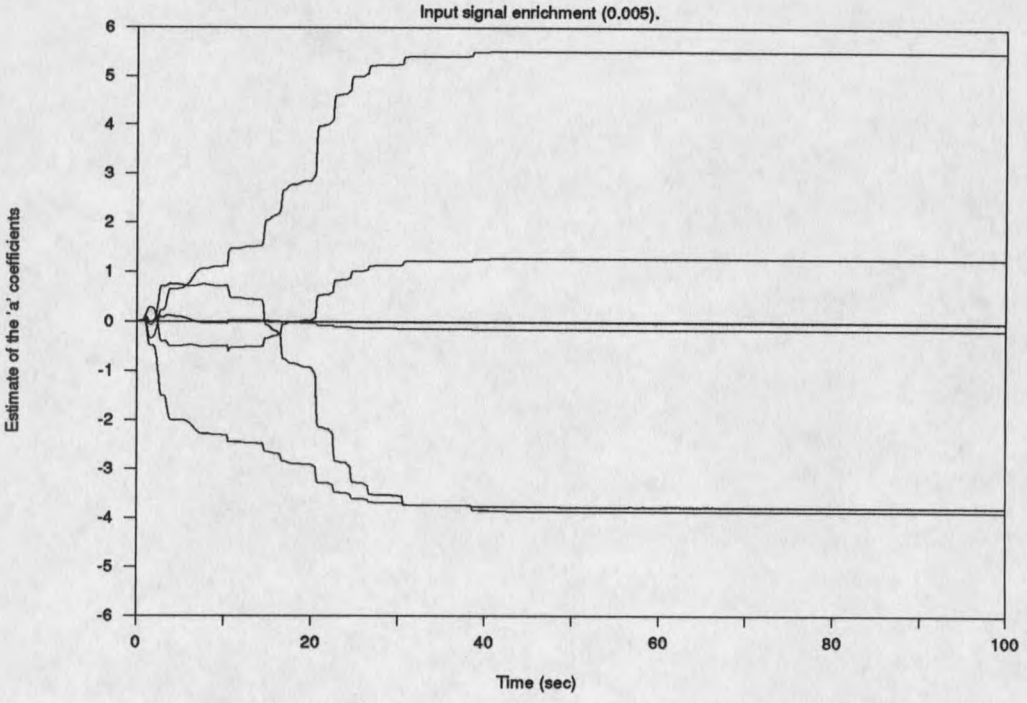

Figure 112. Estimate of the 'a' coefficients (v=0.005).

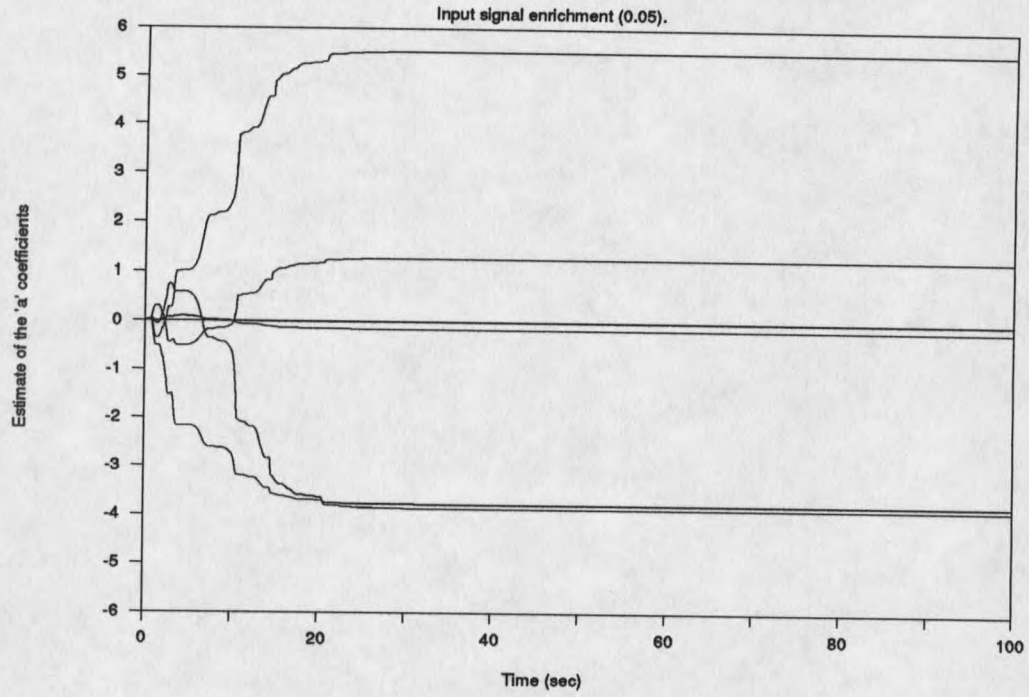

Figure 113. Estimate of the 'a' coefficients (v=0.05).

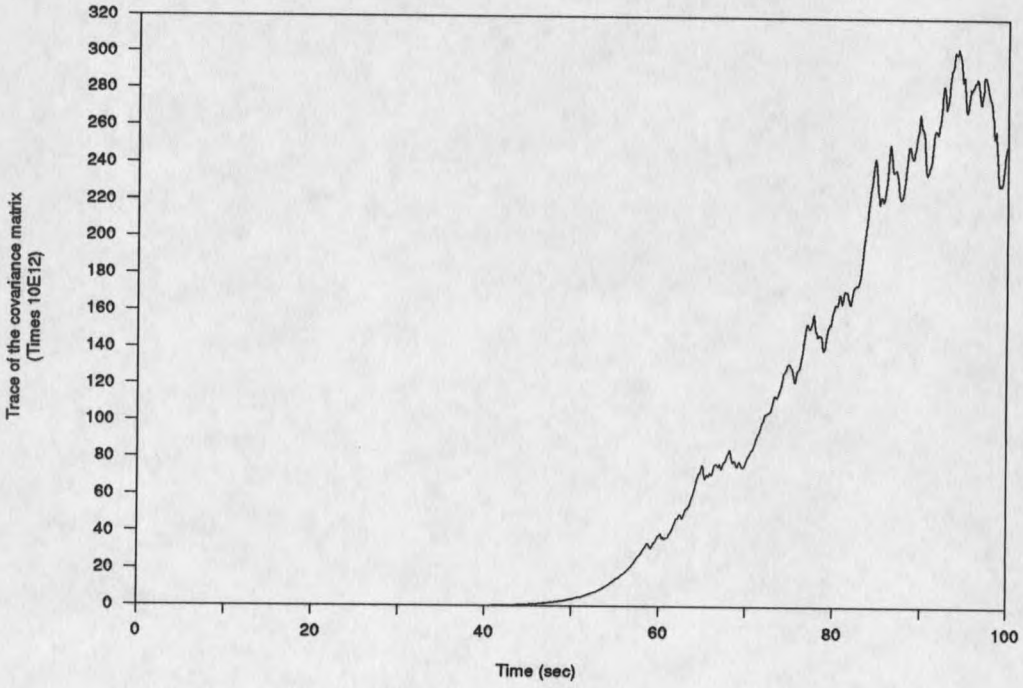

Figure 114. Trace of the covariance matrix (v=0).

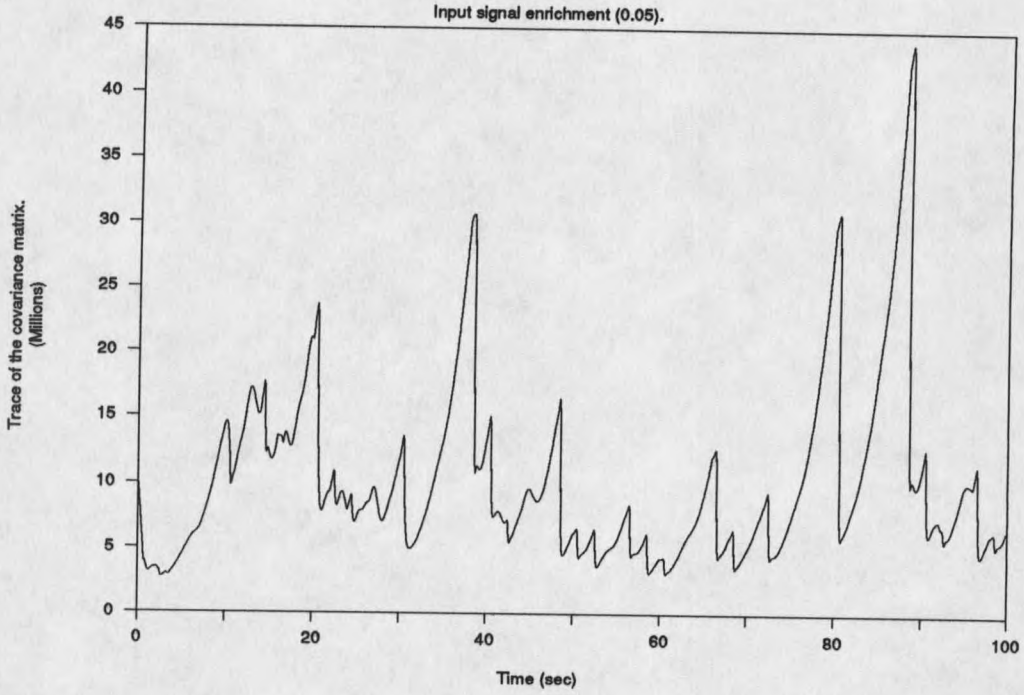

Figure 115. Trace of the covariance matrix (v=0.005).

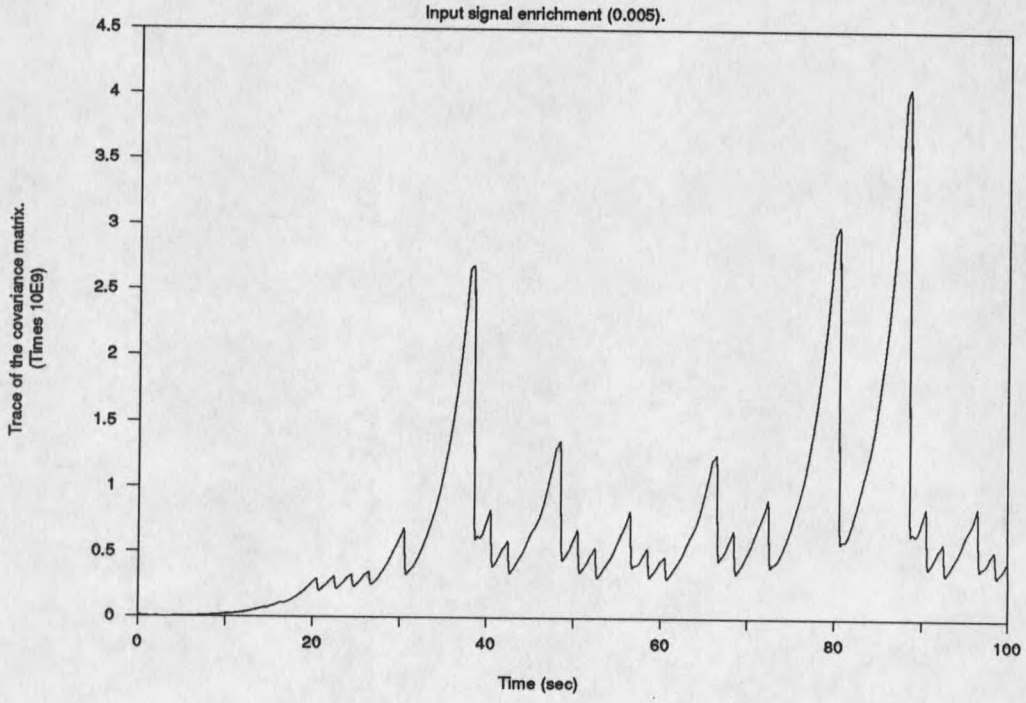

Figure 116. Trace of the covariance matrix (v=0.05).

CHAPTER 10

CONCLUSION AND FUTURE WORK

An implicit assumption in the theory of adaptive control is the availability of a mathematical model which adequately describes the behavior of the system concerned. Therefore the importance of reasonably good estimates of the system model in adaptive control problems is evident.

In many investigations of system identification, severe restrictions on the system are necessary in order to insure the validity of the estimation or identification algorithm, e.g., the plant parameters are constrained to be time-invariant. In many cases of interest, however, the system parameters and/or plant are not constant, but are time varying and perhaps exhibit random properties. Therefore the problem of detecting changes in dynamical properties of a system is of prime interest.

The most significant contribution of this thesis is the CLS algorithm for identification purposes, which is capable of identifying systems with changing parameters. It was developed to overcome some of the practical problems encountered in the application of identification algorithms. The approach implements the notion of adaptation of the covariance matrix (or the gain vector) in a recursive identification algorithm so that both slow, fast and abrupt changes of the system may be tracked with good accuracy.

The basic least-squares algorithm is extended to allow a high degree of adaptation under different operating conditions. The UD form of the least-squares algorithm was selected as the underlying structure, because of its known convergence and stability properties. The CLS algorithm combines the optimal properties of the RLS algorithm with a good tracking capability.

The CLS algorithm shows excellent improvement in estimation in the presence of disturbances. In every case studied CLS outperformed the basic RLS method. The accuracy improvement of CLS over RLS generally was substantial.

The second most significant contribution of this work is the new richness test on the measured data. This test can be used to eliminate the problem of covariance blowup which causes the parameters to drift from their nominal values, as described in Chapter 7. Several approaches, like putting an upper bound on the diagonal elements of the covariance matrix, their trace or their condition number, have been suggested in the literature. Although these procedures have had some success, the particular bounds to choose are not always clear, and in some cases the procedures are computationally expensive, especially those that require the computation of the condition number of a matrix. The ratio-ratio test as presented in Chapter 7 overrides these difficulties. The approach is based on a ratio involving recent values of the trace of the covariance matrix;

we call it the ratio-ratio test. It was shown that the ratio-ratio test gives essentially the same information as other more computationally expensive methods.

The simulation problems in Chapter 9 reveal the excellent performance of the CLS algorithm as well as the ratio-ratio test.

It should be mentioned that many systems require models that characterize both system and noise parameters. The IVAML identification algorithm is an existing method that can be applied in some of these cases. Therefore the attributes of the CLS algorithm can be embedded in the IVAML algorithm to enhance its performance.

This study was limited to single-input single-output (SISO) systems in a noisy as well as time-varying environment. It however, opens up a whole new area of further study. More simulations are necessary to completely understand its applicability. The extension to the multivariable case is of obvious interest.

As can be seen from the updating equation used for covariance matrix, equation (5.39), any modification to the diagonal terms of P will produce a change in all the estimates which may be undesirable. As a result a mechanism should be found to eliminate this undesirable effect.

REFERENCES CITED

1 Kalman, R. E. "Design of a Self Optimizing Control System," Trans. ASME, Vol. 80, pp. 468-478, 1958.

2 Whitaker, H. P., J. Yamron, and A. Kezer. "Design of Model-Reference Adaptive Control Systems for Aircraft," Massachusetts: Instrumentation Laboratory Report No. R-164, Massachusetts Institute of Technology, 1958.

3 Astrom, K. J. "Theory and Application of Adaptive Control - A Survey," Automatica, Vol. 19, pp. 471-486, Sept. 1983.

4 Pierre, D. A. "A Perspective on Adaptive Control of Power Systems," IEEE Trans. on Power Systems, Vol. PWRS-2, No. 2, pp. 387-396, May 1987.

5 Ljung, L. "Consistency of the Least-Squares Identification Method," IEEE Trans. on Automatic Control, Vol. AC-21, pp. 779-781, 1976.

6 Moore, J. B. "On Strong Consistency of Least-Squares Identification Algorithms," Automatica, Vol. 14, pp. 505-509, 1978.

7 Strejc, V. "Least-Squares Parameter Estimation," Automatic Control, Vol. 16, pp. 535-550, 1980.

8 Sripada, N. R. and D. G. Fisher. "Improved Least-Squares Identification for Adaptive Controllers," In the Proceedings of the American Control Conference, Minneapolis, MN, pp. 2027-2037, June 1987.

9 Ljung, L. System Identification Theory for the User. New Jersey: Prentice Hall, Inc., 1987.

10 Mendel, J. M. Discrete Techniques of Parameter Estimation : The Equation Error Formulation. New York: Dekker, 1973.

11 Johnson, C. R. Lectures on Adaptive Parameter Estimation. New Jersey: Prentice Hall Advanced Reference Series, Englewood Cliffs, 1988.

12 Isermann, R. Digital Control Systems. New York : Springer Verlag, 1981.

13 Bierman, G. J. Factorization Method for Discrete Sequential Estimation. New York: Academic Press, 1977.

14 Bierman, G. J. and C. L. Thornton. "Filtering and Error Analysis via the UDU^T Covariance Factorization," IEEE Trans. on Automatic Control, Vol. Ac-23, pp. 901-907, Oct. 1978.

15 Isermann, R. "Parameter Adaptive Control Algorithms - A Tutorial," Automatica, Vol. 18, pp. 513-528, Sept. 1982.

16 Astrom, K. J. and P. Eykhoff. "System Identification - A Survey," Automatica, Vol. 7, pp. 123-167, 1971.

17 Moore, J. B. and G. Casalino. "On Robustness to Noise of Least-Squares Based Adaptive Control," Automatica, Vol. 23, No. 2, pp. 203-208, 1987.

18 Saridis, G. N. "Comparison of Six on-line Identification Algorithms," Automatica, Vol. 10, pp. 69-79, 1974.

19 Iserman, R., U. Baur, W. Bamberger, P. Knepo, and H. Siebert. "Comparison of Six on-line Identification and Parameter Estimation Methods," Automatica, Vol. 10, pp. 81-103, 1974.

20 Soderstrom, T., L. Ljung, and I. Gustavsson. "A Theoretical Analysis of Recursive Identification Methods," Automatica, Vol. 14, pp. 231-244, 1978.

21 Dugard, L. and I. D. Landau. "Recursive Output Error Identification Algorithms," Automatica, Vol. 16, pp. 443-462, 1980.

22 Wouters, W. R. "On-Line Identification in an Unknown Stochastic Environment," IEEE Trans. on Systems of Man and Cybernetics, Vol. SMC-2, pp. 666-668, 1972.

23 Finigan, B. M. and I. H. Rowe. "Strongly Consistent Parameter Estimation by the Introduction of Strong Instrument Variables," IEEE Trans. on Automatic Control, Vol. AC-19, No. 6, pp. 825-830, Dec. 1974.

24 Stoica, P. and T. Soderstrom. "Asymptotic Behaviour of Some Bootstart Estimators," International Journal of Control, Vol. 33, pp. 433-454, 1981.

25 Stoica, P. and T. Soderstrom. "Optimal Instrumental Variable Estimation and Approximate Implementations," IEEE Trans. on Automatic Control, Vol. AC-28, pp. 757-772, July 1983.

26 Gustavsson, I., L. Ljung, and T. Soderstrom. "Survey Paper - Identification of Processes in Closed Loop - Identifiability and Accuracy Aspects," Automatica, Vol. 13, pp. 59-75, 1977.

27 Soderstrom, T. and P. Stoica. "Comparison of Some Instrumental Variable Methods - Consistency and Accuracy Aspects," Automatica, Vol. 17, No. 1, pp. 101-115, 1981.

28 Young, P. C. "An Instrumental Variable Method for Real-Time Identification of Noisy Process," Automatica, Vol. 6, pp. 271-287, 1970.

29 Young, P. C. "Some Observations on Instrumental Variable Methods of Time-Series Analysis," International Journal of Control, Vol. 23, No. 5, pp. 593-612, 1976.

30 Young, P. C. and A. Jakeman. "Refined Instrumental Variable Methods of Recursive Time-Series Analysis. Part I : Single input, Single Output Systems," International Journal of Control, Vol. 29, No. 1, pp. 1-30, 1979.

31 Jackeman, A. and P. C. Young. "Refined Instrumental Variable Methods of Recursive Time-Series Analysis. Part II : Multivariable Systems," International Journal of Control, Vol. 29, pp. 621-644, 1979.

32 Stoica, P. and T. Soderstrom. "Optimal Instrumental Variable Methods for the Identification of Multivariable Linear Systems," Automatica, Vol. 19, No. 4, pp. 425-429, 1983.

33 Soderstrom, T. and P. G. Stocia. Instrumental Variable Method for System Identification. Springer-Verlag, New York, 1983.

34 Morf, M., A. Vieira, and D. T. Lee. "Ladder Forms for Identification and Speech Processing," In the Proceedings of the IEEE Conf. on Decision and Control, New Orleans, 1977.

35 Lee, D. T., B. Friedlander, and M. Morf. "Recursive Ladder Algorithms for ARMA Modeling," IEEE Trans. on Automatic Control, Vol. Ac-27, No. 4, pp. 753-764, 1982.

36 Cowan, C. F. and P. M. Grant. Adaptive Filters. New Jersey: Prentice Hall, Inc., Englewood Cliffs, 1985.

37 Haykin, S. Adaptive Filter Theory, New Jersey: Prentice Hall, Englewood Cliffs, 1986.

38 Friedlander, B., L. Ljung, and M. Morf. "Lattice Implementation of the Recursive Maximum Likelihood Algorithm," In the Proceedings of the IEEE Conference on Decision and Control, San Diego, California, 1981.

39 Friedlander, B. "Lattice Filters for Adaptive Processing," Proceedings of the IEEE, Vol. 70, No. 8, pp. 829-867, Aug. 1982.

40 Strobach, P. "Pure Order Recursive Least-Squares Ladder Algorithms," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. Assp-34, No. 4, pp. 880-897, August 1986.

41 Strobach, P. "Recursive Covariance Ladder Algorithms for ARMA System Identification," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 36, No. 4, pp. 560-580, April 1988.

42 Lee, D. T., M. Morf, and B. Friedlander. "Recursive Least Squares Ladder Estimation Algorithms," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, pp. 627-641, June 1981.

43 Lim, Y. C. and S. R. Parker. "On the Identification of Systems From Data Measurements Using ARMA Lattice Models," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-34, No. 4, pp. 824-828, August 1986.

44 Honig, M. L. "Convergence Models for Lattice Joint Process Estimators and Least-Squares Algorithms," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-31, No. 2, pp. 415-425, April 1983.

45 Porat, B., B. Friedlander, and Morf. "Square Root Covariance Ladder Algorithms," IEEE Trans. on Automatic Control, Vol. Ac-27, No. 4, pp. 813-829, August 1982.

46 Honig, M. L. and D. G. Messerschmitt. "Convergence Properties of an Adaptive Digital Lattice Filter," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, No. 3, pp. 642-653, June 1981.

47 Fortescue, T. R., L. S. Kershenbaum, and B. E. Ydstie. "Implementation of Self-Tuning Regulators With Variable Forgetting Factors," Automatica, Vol. 17, No. 6, pp. 831-835, 1981.

48 Ljung, L. and T. Soderstrom. Theory and Practice of Recursive Identification. The MIT Press, Cambridge, Massachusetts, 1983.

49 Norton, J. P. An Introduction to Identification. Academic Press, New York, 1986.

50 Mehra, R. K. and D. G. Lainiots. System Identification - Advances and Case Studies. Academic Press, New York, 1976.

51 Sinha, N. K. and B. Kuzta. Modeling and Identification of Dynamic Systems. Van Nostrand Reinhold Company, New York, 1983.

52 Goodwin, G. C. and R. L. Payne. Dynamic System Identification : Experiment Design and Data Analysis. Academic Press, New York, 1977.

53 Goodwin, G. C. and K. S. Sin. Adaptive Filtering Prediction and Control. Englewood Cliffs, NJ: Prentice-Hall, 1984.

54 Lim, J. S. and A. V. Oppenheim. Advanced Topics in Signal Processing, pp. 420-428, Prentice Hall, NJ, 1988.

55 Hauer, J. F., C. J. Demeure, and L. L. Scharf. "Initial Results in Prony Analysis of Power System Response Signals," Paper no. 89SM702-2-PWRS, IEEE Summer Power Meeting, July 9-14, 1989.

56 Hildebrand, F. B. Introduction to Numerical Analysis, pp. 378-380, McGraw Hill, New York, 1956.

57 Day, S. M. and S. L. Marple. "Spectrum Analysis - A Modern Perspective," Proc. IEEE, pp. 1380-1419, November 1981.

58 Van Blaricum, M. L., and R. Mittra. "Problems and Solutions Associated With Prony's Method for Processing Transient Data," IEEE Trans. on Antennas and Propagation, pp. 174-182, Jan. 1987.

59 Tufts, D. W. and R. Kumaresan. "Singular Value Decomposition and Improved Frequency Estimation Using Linear Prediction," IEEE Trans. on Acoustics, Speech, and Signal Processing, pp. 671-675, Aug. 1982.

60 Tufts, D. W. and R. Kumaresan. "Frequency Estimation of Multiple Sinusoids : Making Linear Prediction Perform Like Maximum Likelihood," Proc. IEEE, pp. 975-989, Sept. 1982.

61 Kumaresan, R. and D. W. Tufts. "Estimating the Parameters of Eponentially Damped Sinusoids and Pole-Zero Modeling in Noise," IEEE Trans. on Acoustics, Speech, and Signal Processing, pp. 833-840, Dec. 1982.

62 Kurmaresan, R. "On the Zero of the Linear Prediction-Error Filter for Deterministic Signals," IEEE Trans. on Acoustics, Speech, and Signal Processing, pp. 217-220, Feb. 1983.

63 Kumaresan, K., D. W. Tufts, and L. L. Scharf. "A Prony Method for Noisy Data : Choosing the Signal Components and Selecting the Order in Exponential Signal Models," Proc. IEEE, pp. 230-233, Feb. 1984.

64 Kumaresan, K., L. L. Scharf, and A. K. Shaw. "An Algorithm for Pole-Zero Modeling and Spectral Analysis," IEEE Trans. on Acoustics, Speech, and Signal Processing, pp. 637-640, June 1986.

65 Trudnowski, D. J., J. R. Smith, T. A. Short, and D. A. Pierre. "An Application of Prony Method in PSS Design for Multimachine Systems," IEEE Winter Power Meeting, Feb. 1990 (pending).

66 Gauss, K. F. Theory of the Motion of the Heavenly Bodies. Dover, New York, 1963.

67 Pierre, D. A. "A Perspective on Adaptive Control of Power Systems," IEEE Winter Power Meeting Paper No. 86WM090-5, Feb. 1986, and IEEE Trans. on Power Systems, Vol. PWRS-2, pp. 387-396, May 1987.

68 Pierre, D. A. "Enhanced LQ Controllers and Adaptive Control," Electronics Research Laboratory Report No. 687, Nov. 1987, and Proceedings, IEEE Conf. on Decision and Control, pp. 782-785, Dec. 1988.

69 Pierre, D. A. "Power System Damping: A Simulation Program and Enhanced LQ Self-Tuning Strategies," Proceedings, International Workshop on Adaptive Control for Industrial Use, Alberta, Canada, pp. 73-92, June 1988.

70 Pierre, D. A. and J. R. Smith. "Alternative Adaptive Control Methods for Power Systems," Proceedings, IASTED Conference on High Technology in the Power Industry, pp. 162-167, August 1986.

71 Smith, J. R., D. A. Pierre, D. A. Rudberg, and R. M. Johnson. "Robust VAR Unit Control Strategies to Enhance Damping of Power System Oscillations," Proceedings, IASTED Conference on High Technology in the Power Industry, pp. 238-243, March 1988.

72 Smith, J. R., D. A. Pierre, D. A. Rudberg, I. Sadighi, A. P. Johnson, and J. F. Hauer. "An Enhanced LQ Adaptive VAR Unit Controller for Power System Damping," IEEE Power Engineering Society Summer Meeting, Paper No. 88SM692-6, July 1988, and IEEE Trans. on Power Systems, Vol. 4, pp. 443-451, May 1989.

73 Smith, J. R., D. A. Pierre, I. Sadighi, M. H. Nehrir, and J. F. Hauer. "A Supplementary Adaptive VAR Unit Controller for Power System Damping," IEEE Power Engineering Society Winter Meeting, Paper No. 89WM197-5, Feb. 1989, and IEEE Trans. on Power Systems, Vol. 4, pp. 1017-1023, August, 1989.

74 Smith, J. R. "Robust VAR Unit Control Strategies for Damping of Power System Oscillations," Ph.D. Thesis, Montana State University, Bozeman, MT, July 1988.

75 Astrom, K. J. "Lectures on the Identification Problem - The Least-Squares Method," Report 6086, Lund Institute of Technology, Division of Automatic Control, 1968.

76 Soderstrom, T., L. Ljung, and I. Gustavsson. "A Comparative Study of Recursive Identification Methods," Report 7427, Dept. of Automatic Control, Lund Institute of Thechnology, 1974.

77 Isermann, R. and U. Baur. "Two-Step Process Identification With Correlation Analysis and Least-Squares Parameter Estimation," Tran. ASME, Journal of Dynamic Systems, Measurement, and Control (ser. G), Vol. 96, pp. 425-432, 1974.

78 Astrom, K. J. "Maximum Likelihood and Prediction Error Methods," Automatica, Vol. 16, pp. 551-574, 1980.

79 Soderstrom, T. "On the Uniqueness of Maximum Likelihood Identification," Automatica, Vol. 11, pp. 193-197, 1975.

80 Fuhrt, B. P. "New Estimator for the Identification of Dynamic Processes," IBK Report, Institute Boris Kidric Vinca, Belgrade, Yugoslavia, 1973.

81 Hastings-James, R. and M. W. Sage. "Recursive Generalized Least-Squares Procedure for On-line Identification of Process Parameters," IEE Proc., Vol. 116, pp. 2057-2062, 1969.

82 Gertler, J. and Cs. Banyasz. "A Recursive (on-line) Maximum Likelihood Identification Method," IEEE Trans. Automatic Control, Vol. AC-19, pp. 816-820, 1974.

83 Fortescue, T. R., L. S. Kershenbawn, and B. E. Ydsti. "Implementation of Self-Tuning Regulators With Variable Forgetting Factors," Automatica, Vol. 17, pp. 831-835, 1981.

84 Goodwin, G. C., D. J. Hill, and M. Palaniswami. "A Perspective on Convergence of Adaptive Control Algorithms," Automatica, Vol. 20, pp. 519-531, 1984.

85 Sadighi, I. "System Identification Methods for Adaptive Control," Proposal for Ph. D. Research, Electrical Engineering Department, Montana State University, August 1988.

86 Harris, C. J. and S. A. Billings (editors). Self-Tuning and Adaptive Control : Theory and Applications, Institution of Electrical Engineering, London, 1981.

87 Clark, D. W. "Some Implementation Considerations of Self-Tuning Controllers," pp. 81-110 of Numerical Techniques for Stochastic Systems, edited by A. Archett and M. Cugiani, North Holland Publishing Co., New York, 1980.

88 Bierman, G. J. "Numerical Experience with Modern Estimation Algorithm," Proceedings, 24th IEEE Conf. on Decision and Control, Ft. Lauderdale, Florida, pp. 1896-1901, Dec. 1985.

89 Astrom, K. J. and B. Wittenmark. Computer Control Systems Theory and Design, Prentice-Hall, Englewood Cliffs, NJ, 1984.

90 Astrom, K. J. and T. Bohlin. "Numerical Identification of linear Dynamic System From Normal Operating Records," Proceedings of the IFAC Symposium on Self-Adaptive Systems, Teddington, England, 1965.

91 Nobel, B. Applied Linear Algebra, Prentice-Hall, Englewood Cliffes, NJ, 1969.

92 Astrom, K. J. "Adaptive Feedback Control," Proceedings of the IEEE, Vol. 75, No. 2, pp. 185-217, Feb. 1987.

93 Pierre, D. A., D. A. Rudberg, R. M. Johnson, J. R. Smith, I. Sadighi, and A. P. Johnson. "Adaptive Control in Power Systems," Montana : Electronics Research Laboratory Report No. 487, Montana State University, Sept. 1987.

94 Ogata, K. Discrete-Time Control Systems, Prentice-Hall, Inc., Englewood Cliffes, NJ, 1987.

95 Peebles, P. Z. Probability, Random Variables, and Random Signal Principles, McGraw-Hill, Inc., New York, NY, 1987.

96 Walpole, R. E., R. H. Myers. Probability and Statistics for Engineering and Scientists, Macmillan Publishing Co., Inc., New York, NY, 1978.

APPENDICES

APPENDIX A - THE GENERATE SUBROUTINE

In this appendix we present a FORTRAN 77 program for generating input and output signals based on a given model. This program prompts the user for variety of information which it needs to generate the input and output values, also it produces a file which contains the generated values.

```
C
C
C  PROGRAM TO GENERATE A SEQUENCE OF INPUTS AND OUTPUTS WITH
C  APPROPRIATE NOISE ADDITION TO THE OUTPUT, FOR IDENTIFICATION
C  PURPOSES.
C
C
C            +------------+
C   E(T) OR  |  C(Z**-1)  |  V(T) OR VNEW
C   ------->|  --------  |-----------------|
C   ENEW     |  D(Z**-1)  |                 |
C            +------------+                 |
C                                           |
C                                           |
C            +------------+                 |  +
C   U(T) OR  |  B(Z**-1)  |   ACT. Y(T)   + \/    Y(T) OR YNEW(T)
C   ------->|  --------  |---------------->O-------------------->
C   UNEW     |  A(Z**-1)  |
C            +------------+
C
C
C
C                                +================+
C          B(Z**-1)              | C(Z**-1)       |
C   Y(T) = --------  U(T)  +     | --------  E(T) |
C          A(Z**-1)              | D(Z**-1)       |
C                                +================+
C                                |<---- V(T) --->|
C
C  WHERE :
C              Y(T)   : OUTPUT OF THE SYSTEM AT TIME T.
C              U(T)   : INPUT OF THE SYSTEM AT TIME T.
C              E(T)   : GUASSIAN WHITE NOISE AT TIME T.
C              V(T)   : DISTURBANCES AT TIME T.
C       A(Z**-1)   : 1 + A(1)Z**-1 + ... + A(NA)Z**-NA
C       B(Z**-1)   :     B(1)Z**-1 + ... + B(NB)Z**-NB
C       C(Z**-1)   : 1 + C(1)Z**-1 + ... + C(NC)Z**-NC
C       D(Z**-1)   : 1 + D(1)Z**-1 + ... + D(ND)Z**-ND
C
      IMPLICIT NONE
      REAL*8 A,B,C,D,UNEW,YNEW,ENEW,VNEW,ED1,ED2,ACTY
      REAL*8 FI,FIV,SAMPLE,W,AA,BB,WB,ED,UD,UD1,UD2,FII
      REAL*8 TEMPU,TEMPY,TEMPV,TEMPE,TEMPEV,T,SIGN,TIN,TFI
      REAL*8 GARBAGE,ADD_C
      INTEGER NA,NB,NC,ND,LOOP,N,I1,I2,DATAFILE,WHERE_TO_READ
      INTEGER I,IASK,ITERATION,IICON,IE,IU,INTERVAL
      CHARACTER*20 IN_FILE,DATA_FILE
      DIMENSION A(100),B(100),C(100),D(100)
      DIMENSION FI(100),FIV(100),FII(100)
      DATA I1,I2/9887,12945/
C.....................................................
C     PRINT SOME INFORMATION ABOUT THIS PROGRAM.
C.....
      CALL CLS
      WRITE(*,*) '+-----------------------------------------------+'
      WRITE(*,*) '| PROGRAM TO GENERATE A SEQUENCE OF INPUT AND |'
      WRITE(*,*) '| OUTPUT WITH ADDITION OF NOISE TERM FOR      |'
      WRITE(*,*) '| IDENTIFICATION PURPOSES.                    |'
      WRITE(*,*) '+-----------------------------------------------+'
      WRITE(*,*) ' '
```

Figure 117. The GENERATE subroutine.

```
      WRITE(*,*) '
      WRITE(*,*) '                                  +================+'
      WRITE(*,*) '          B(Z**-1)                | C(Z**-1)       |'
      WRITE(*,*) ' Y(T) = -------- U(T) +           | -------- E(T)  |'
      WRITE(*,*) '          A(Z**-1)                | D(Z**-1)       |'
      WRITE(*,*) '                                  +================+'
      WRITE(*,*) '                                  |<---- V(T) --->|'
      WRITE(*,*) '  WHERE :'
      WRITE(*,*) '       Y(T)   : OUTPUT OF THE SYSTEM AT TIME T.'
      WRITE(*,*) '       U(T)   : INPUT OF THE SYSTEM AT TIME T.'
      WRITE(*,*) '       E(T)   : GUASSIAN WHITE NOISE AT TIME T.'
      WRITE(*,*) '       V(T)   : DISTURBANCES AT TIME T.'
      WRITE(*,*) '  A(Z**-1)    : 1 + A(1)Z**-1 + ... + A(NA)Z**-NA'
      WRITE(*,*) '  B(Z**-1)    :     B(1)Z**-1 + ... + B(NB)Z**-NB'
      WRITE(*,*) '  C(Z**-1)    : 1 + C(1)Z**-1 + ... + C(NC)Z**-NC'
      WRITE(*,*) '  D(Z**-1)    : 1 + D(1)Z**-1 + ... + D(ND)Z**-ND'
      WRITE(*,*) '  ..........................................'
C.....................
C
C  GET THE INPUTS.
C.....
      WRITE(*,*) 'INPUT THE NUMBER OF A COEFFICIENTS (NA).:'
      READ (*,*) NA
      WRITE(*,*) 'INPUT THE NUMBER OF B COEFFICIENTS (NB).:'
      READ (*,*) NB
      WRITE(*,*) 'INPUT THE NUMBER OF C COEFFICIENTS (NC).:'
      READ (*,*) NC
      WRITE(*,*) 'INPUT THE NUMBER OF D COEFFICIENTS (ND).'
      READ (*,*) ND
      WHERE_TO_READ=0
      WRITE(*,*) 'READ THESE COEFFICIENTS FROM A DATA FILE ?'
      WRITE(*,*) '1 => YES'
      WRITE(*,*) '0 => NO'
      WRITE(*,*) 'WHICH ONE ? ( 1 OR 0 )'
      READ (*,*) WHERE_TO_READ
      IF (WHERE_TO_READ .EQ. 1) THEN
            WRITE(*,*) 'INPUT THE NAME OF THE DATA FILE.'
            READ (*,24) DATA_FILE
            OPEN(2,FILE=DATA_FILE,STATUS='OLD')
      END IF
      CALL CLS
      WRITE(*,*) '.....................................'
      WRITE(*,*) 'INPUT THE NAME OF OUTPUT DATA FILE.'
      READ (*,24) IN_FILE
24    FORMAT(A)
      WRITE(*,*) '.....................................'
1     WRITE(*,*) ' '
      WRITE(*,*) 'THE OUTPUT DATA FILE MAY HAVE ONE OF THE'
      WRITE(*,*) 'FOLLOWING FORMATS.'
      WRITE(*,*) 'WHICH ONE DO YOU WANT?(1 OR 2)'
      WRITE(*,*) '   1 => TIME,Y,U'
      WRITE(*,*) '   2 => TIME,Y,U,V'
      WRITE(*,*) '   3 => TIME,Y,NOISE FREE Y (ACTUAL Y),U,V'
      WRITE(*,*) 'NOTE THAT, Y = ACTUAL Y + DISTURBANCE (V(T))'
      READ (*,*) DATAFILE
      IF((DATAFILE .LT. 1) .OR. (DATAFILE .GT.3)) GOTO 1
      CALL CLS
3     WRITE(*,*) 'DO YOU WANT THE E (NOISE) TERM BE :'
      WRITE(*,*) '   1 => NO NOISE.'
```

Figure 117. Continued.

```
      WRITE(*,*) '   2 => ED*GAUSSIAN WHITE NOISE. (0,ED)'
      WRITE(*,*) '   3 => ED*EXP(A*T)*SIN(W*T)'
      WRITE(*,*) '   4 => ED*EXP(A*T)*SIN(W*T)*GAUSSIAN WHITE NOISE.'
      WRITE(*,*) '   5 => ED1*EXP(A*T)*SIN(W*T) + ED2*GAUSSIAN
     &                WHITE NOISE.'
      WRITE(*,*) 'NOTE : ED, ED1 AND ED2 ARE CONSTANT MULTIPLIERS.'
      WRITE(*,*) '   WHICH ONE ? (1,2,3 OR 4)'
      READ (*,*) IE
      IF ((IE .LT. 1) .OR. (IE .GT. 5)) GOTO 3
      IF(IE .GT. 2) THEN
             IF(IE .EQ. 5) THEN
                    WRITE(*,*) 'ED1 = ?'
                    READ (*,*) ED1
                    WRITE(*,*) 'ED2 = ?'
                          READ (*,*) ED2
             ELSE
                    WRITE(*,*) 'ED= ?'
                    READ (*,*) ED
             END IF
             WRITE(*,*) 'A = ?'
             READ (*,*) AA
             WRITE(*,*) 'W = ?'
             READ (*,*) W
      END IF
      IF(IE .EQ. 2) THEN
             WRITE(*,*) 'ED= ?'
             READ (*,*) ED
      END IF
      CALL CLS
7       WRITE(*,*) 'THE INPUT TO THE SYSTEM IS :'
      WRITE(*,*) '1=> U = B (CONSTANT)'
      WRITE(*,*) '2=> U = (+ OR -)B'
      WRITE(*,*) '3=> U = UD*EXP(B*T)*SIN(WB*T) + (+ OR -)C'
      WRITE(*,*) '4=> U = UD*EXP(B*T)*SIN(WB*T)*NOISE + (+ OR -)C'
      WRITE(*,*) '5=> U = UD1*EXP(B*T)*SIN(WB*T) + UD2*NOISE +
     &                    (+ OR -)C'
      WRITE(*,*) 'NOTE : UD, UD1 AND UD2 ARE CONSTANT MULTIPLIERS.'
      WRITE(*,*) 'THE NOISE IS A GUSSIAN WHITE NOISE.'
      WRITE(*,*) 'AND THE SIGN(+ OR -)WILL BE CHOSEN AT RANDOM.'
      WRITE(*,*) 'WHICH ONE ? (1,2,3,4,5 OR 6)'
      READ (*,*) IU
      IF((IU .LT. 1) .OR. (IU .GT. 5)) GOTO 7
      WRITE(*,*) 'B = ?'
      READ (*,*) BB
      IF(IU .GT. 2) THEN
             WRITE(*,*) 'C = ?'
             READ (*,*) ADD_C
      END IF
      IF(IU .GT. 1) THEN
          IF(((IU .GT. 2) .AND. (ADD_C .NE. 0.0)) .OR.
     &       (IU .EQ. 2)) THEN
            WRITE(*,*) 'NOTE : THE SIGN OF B OR C WILL BE CHOSEN
     &      AS FOLLOW;'
            WRITE(*,*) 'THE SIGN OF B OR C WILL KEEP CONSTANT OVER'
            WRITE(*,*) 'SOME NUMBER OF ITERATIONS, THEN A RANDOM'
            WRITE(*,*) 'NUMBER WILL BE GENERATED BETWEEN 0 AND 1.'
            WRITE(*,*) 'THE SIGN OF B OR C WILL BE CHANGED IF THIS
     &                  RANDOM NUMBER'
```

Figure 117. Continued.

```
        WRITE(*,*) 'IS BETWEEN 0.5 AND 1 OTHERWISE THE SIGN
&                   WILL'
        WRITE(*,*) 'KEEP THE SAME.'
        WRITE(*,*) 'INPUT THE NUMBER OF ITERATIONS FOR THIS
&                   PURPOSE ?'
         READ (*,*) INTERVAL
      END IF
  END IF
  IF((IU .GE. 3) .AND. (IU .LE. 5)) THEN
        IF(IU .EQ. 5) THEN
              WRITE(*,*) 'UD1 = ?'
              READ (*,*) UD1
              WRITE(*,*) 'UD2 = ?'
              READ (*,*) UD2
        ELSE
              WRITE(*,*) 'UD = ?'
              READ (*,*) UD
        END IF
        WRITE(*,*) 'WB = ?'
        READ (*,*) WB
  END IF
  IF(NA .NE. 0) THEN
        CALL CLS
        IF (WHERE_TO_READ .EQ. 1) THEN
              WRITE(*,*) 'READING THE A COEFFICIENTS..'
        ELSE
              WRITE(*,*) 'INPUT THE A COEFFICIENTS.'
        END IF
        DO I =1,NA
              IF (WHERE_TO_READ .EQ. 1) THEN
                    READ(2,*) A(I)
                    WRITE(*,*) A(I)
              ELSE
                    WRITE(*,*) 'A(',I,'  ) = ?'
                    READ (*,*) A(I)
              END IF
         END DO
  END IF
  IF(NB .NE. 0) THEN
        CALL CLS
        IF (WHERE_TO_READ .EQ. 1) THEN
              WRITE(*,*) 'READING THE B COEFFICIENTS..'
        ELSE
              WRITE(*,*) 'INPUT THE B COEFFICIENTS.'
        END IF
        DO I =1,NB
              IF (WHERE_TO_READ .EQ. 1) THEN
                    READ(2,*) B(I)
                    WRITE(*,*) B(I)
              ELSE
                    WRITE(*,*) 'B(',I,'  ) = ?'
                    READ (*,*) B(I)
              END IF
         END DO
  END IF
  IF(NC .NE. 0) THEN
        CALL CLS
        IF (WHERE_TO_READ .EQ. 1) THEN
              WRITE(*,*) 'READING THE C COEFFICIENTS..'
```

Figure 117. Continued.

```
                ELSE
                        WRITE(*,*) 'INPUT THE C COEFFICIENTS.'
                END IF
                DO I =1,NC
                        IF (WHERE_TO_READ .EQ. 1) THEN
                                READ(2,*) C(I)
                                WRITE(*,*) C(I)
                        ELSE
                                WRITE(*,*) 'C(',I,'  ) = ?'
                                READ (*,*) C(I)
                        END IF
                 END DO
        END IF
        IF(ND .NE. 0) THEN
                CALL CLS
                IF (WHERE_TO_READ .EQ. 1) THEN
                        WRITE(*,*) 'READING THE D COEFFICIENTS..'
                ELSE
                        WRITE(*,*) 'INPUT THE D COEFFICIENTS.'
                END IF
                DO I =1,ND
                        IF (WHERE_TO_READ .EQ. 1) THEN
                                READ(2,*) D(I)
                                WRITE(*,*) D(I)
                        ELSE
                                WRITE(*,*) 'D(',I,'  ) = ?'
                                READ (*,*) D(I)
                        END IF
                 END DO
        END IF
        CALL CLS
        WRITE(*,*) 'INPUT THE STARTING TIME.'
        READ (*,*) TIN
        WRITE(*,*) '.............................'
        WRITE(*,*) 'INPUT THE FINAL TIME.'
        READ (*,*) TFI
        WRITE(*,*) '.............................'
        WRITE(*,*) 'INPUT THE SAMPLING RATE (T).'
        WRITE(*,*) 'NOTE : T = T + T'
        READ (*,*) SAMPLE
        CALL CLS
C.....................................................
C  OPEN THE OUTPUT DATA FILE.
C.....
        OPEN(1,FILE=IN_FILE,STATUS='NEW')
        WRITE(*,*) 'THE FILE ',IN_FILE,' HAS BEEN OPENED FOR OUTPUT.'
C.......................
C
C  INITIALIZATIONS.
C.....
        DO I=1,(NA+NB+NC)
                FI(I) = 0.D0
                FIV(I) = 0.D0
                FII(I) = 0.D0
        END DO
        UNEW  = 0.0
        TEMPU = BB
        YNEW  = 0.0
```

Figure 117. Continued.

```
      ENEW  = 0.0
      VNEW  = 0.0
      T     = 0.0
      IICON = 0
C.....................................
C  START THE LOOP.
C.....
       DO 1000 T=TIN, TFI, SAMPLE
C.........................................
C
C  CALCULATE THE NEW Y, U, V AND E VALUES.
C.....
C.........................................
C
C  CALL GGNML SUBROUTINE TO FIND E AND U VALUES AT TIME T.
C.....
            SIGN = RAN(I1,I2)
            IF(IU .GT. 1) THEN
                  IF(IICON .GE. INTERVAL) THEN
                        IF(SIGN .GT. 0.5) THEN
                              TEMPU = -TEMPU
                              ADD_C = -ADD_C
                        END IF
                        IICON = 0
                  END IF
                  IICON = IICON + 1
            END IF
            CALL GGNML(TEMPE)
            IF(IU.EQ.3)TEMPU=UD*EXP(BB*T)*SIN(WB*T)+ADD_C
            IF(IU.EQ.4)TEMPU=UD*EXP(BB*T)*SIN(WB*T)*TEMPE+ADD_C
            IF(IU.EQ.5)TEMPU=UD1*EXP(BB*T)*SIN(WB*T)+UD2*TEMPE+
     &                      ADD_C
            CALL GGNML(TEMPE)
            IF(IE.EQ.1)TEMPE=0.0
            IF(IE.EQ.3)TEMPE=ED*EXP(AA*T)*SIN(W*T)
            IF(IE.EQ.4)TEMPE=ED*EXP(AA*T)*SIN(W*T)*TEMPE
            IF(IE.EQ.5)TEMPE=ED1*EXP(AA*T)*SIN(W*T) + ED2*TEMPE
            IF(IE.EQ.2)TEMPE=ED*TEMPE
C.....................................................
C
C  FIND THE VALUE OF V AT TIME T.
C  V(T) = -D(1)V(T-1) - D(2)V(T-2) -...+ E(T) + C(1)E(T-1)+...
C.....
            TEMPV = TEMPE
            DO 25 I = 1,ND
                  TEMPV = TEMPV + D(I)*FIV(I)
 25         CONTINUE
            DO 26 I = ND+1,ND+NC
                  TEMPV = TEMPV + C(I-ND)*FIV(I)
 26         CONTINUE
C.....................................................
C
C  FIND THE VALUE OF Y AND NOISE FREE Y AT TIME T.
C  Y(T) = -A(1)Y(T-1) - A(2)Y(T-2) - ... + B(1)U(T-1) + ... + V(T)
C         +A(1)V(T-1) + A(2)V(T-2) + ...
C.....
            TEMPY = 0.0
            ACTY = 0.0
```

Figure 117. Continued.

```
                DO 30 I =1,NA
                       ACTY = ACTY + A(I)*FII(I)
                       TEMPY = TEMPY + A(I)*FI(I)
   30                CONTINUE
                  DO 40 I =NA+1,NB+NA
                       ACTY = ACTY + B(I-NA)*FII(I)
                         TEMPY = TEMPY + B(I-NA)*FI(I)
   40                CONTINUE
                  TEMPEV = TEMPV
                  DO 41 I = (NA+NB+1),(NA+NB+NA)
                       TEMPEV = TEMPEV + A(I-NA-NB)*FI(I)
   41                CONTINUE
                YNEW = TEMPY + TEMPEV
                UNEW = TEMPU
                ENEW = TEMPE
                VNEW = TEMPV
C.....
C  UPDATE THE FI,FII AND FIV VECTORS.
C.....
                N = NA+NB+NA
                DO 900 I =1,N-1
                       FI(N+1-I) = FI(N-I)
 900               CONTINUE
                FI(1) = -YNEW
                FI(NA+1) = UNEW
                FI(NA+NB+1) = VNEW
                N = NC+ND
                DO 910 I =1,N
                       FIV(N+1-I) = FIV(N-I)
 910               CONTINUE
                FIV(1) = -VNEW
                FIV(ND+1) = ENEW
                DO 911 I =1,(NA+NB)
                       FII(NA+NB+1-I) = FII(NA+NB-I)
 911               CONTINUE
                FII(1) = -ACTY
                FII(NA+1) = UNEW
C.....
C  WRITE THE RESULTS OUT.
C.....
                IF(DATAFILE .EQ. 1) THEN
                       WRITE(1,111) T,YNEW,UNEW
 111                       FORMAT(3(F15.7,2X))
                END IF
                IF(DATAFILE .EQ. 2) THEN
                       WRITE(1,112) T,YNEW,UNEW,TEMPEV
 112                       FORMAT(4(F15.7,2X))
                END IF
                IF(DATAFILE .EQ. 3) THEN
                       WRITE(1,113) T,YNEW,ACTY,UNEW,TEMPEV
 113                       FORMAT(5(F15.7,2X))
                END IF
 1000         CONTINUE
        WRITE(*,*) ' '
        WRITE(*,*) ' '
        WRITE(*,*) '                                    +--------------+'
```

Figure 117. Continued.

```
      WRITE(*,*) '                                    |   ALL DONE    |'
      WRITE(*,*) '                                    +---------------+'
      WRITE(*,*) ' '
      WRITE(*,*) ' '
       END
C
C
C  GGNML
C
C  TO GENERATE WHITE NOISE SEQUENCE WITH (0,1).
C
C
      SUBROUTINE GGNML(GNOISE)
      IMPLICIT REAL*8 (A-H), REAL*8(O-Z)
      IMPLICIT INTEGER*2 (I-N)
      DATA I1,I2/9887,12945/
      GNOISE = 0.D0
      DO 10 I =1,12
            GNOISE = GNOISE + RAN(I1,I2)
10       CONTINUE
       GNOISE = GNOISE-6.0
      RETURN
      END
```

Figure 117. Continued.

APPENDIX B - RLS USING UD FACTORIZATION

In this appendix we present a FORTRAN 77 subroutine for least-squares estimation based on UD factorization, as described by Bierman [13].

```
C
C  UUD
C
C  LEAST-SQUARES ESTIMATION OF THE PARAMETERS, USING UD
C  FACTORIZATION.
C
C  INPUT :
C          U        : NEW INPUT MEASUREMENT.
C          Y        : NEW OUTPUT MEASUREMENT.
C          XLAMBDA  : FORGETTING FACTOR.
C          N        : I.E. N = NA + NB + NC
C          NA       : NUMBER OF A COEFFICIENTS.
C          NB       : NUMBER OF B COEFFICIENTS.
C          THETA    : A PRIORI ESTIMATE OF THE PARAMETERS.
C          ISTART   : 0 => INITIALIZE ALL THE VARIABLES. THIS IS
C                          THE FIRST PASS THROUGH THIS SUBROUTINE.
C                     1 => CONTINUE WITH ESTIMATION.
C                     2 => STOP ESTIMATION.
C          PDIAG    : INITIAL VALUE FOR DIAGONAL TERMS OF THE
C                     COVARIANCE MATRIX.
C
C  OUTPUT :
C          THETA       : UPDATED VERSION OF THE ESTIMATES.
C          TRACE       : TRACE OF THE COVARIANCE MATRIX.
C
C
      SUBROUTINE UUD(U,Y,XLAMBDA,N,NA,NB,THETA,ISTART,PDIAG,
     $                    TRACE,TT,EEE,ACTY)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER*4 (I-N)
      DIMENSION THETA(100),FI(100),DIAG(100),OFFDIAG(100)
      DIMENSION XK(100),FII(100)
      IF ( ISTART .EQ. 2 ) GOTO 1010
      IF ( ISTART .EQ. 0 ) THEN
C........
C  INITIALIZATION.
C.....
            NUP = N*(N-1)/2
            DO 100 I =1,NUP
                  THETA(I)    = 0.D0
                  FI(I)       = 0.D0
                  FII(I)      = 0.D0
                  OFFDIAG(I)  = 0.D0
C
C        LET DIAG VECTOR BE A BIG NUMBER WHICH MEANS THE INITIAL
C        VALUES ARE NOT ACCURATE.
C
                  DIAG(I)     = PDIAG
 100           CONTINUE
            IF(ISTART .EQ. 0)  ISTART = 1
      END IF
C........
CESTIMATE THE ERROR IN ESTIMATING THE Y(T) AND ESTIMATE OF Y (ESTY)
C.....
      ESTY = 0.0
      DO 1333 I=1,N
            ESTY = ESTY + FII(I)*THETA(I)
```

Figure 118. The subroutine for RLS using UD factorization.

```
 1333        CONTINUE
         ERROR = Y - ESTY
C.........
C  START THE PROCEDURE.
C.....
         XPERR = ERROR
C.........
C  CALCULATE GAIN AND COVARIANCE USING U-D METHOD.
C.....
         FJ = FII(1)
         VJ = DIAG(1)*FJ
         XK(1) = VJ
         ALPHAJ = 1.0 + VJ*FJ
         DIAG(1) = DIAG(1)/ALPHAJ/XLAMBDA
         IF ( N .GT. 1 ) THEN
                KF = 0
                KU = 0
                DO 20 J =2,N
                        FJ = FII(J)
                        DO 30 I =1,J-1
                                KF = KF+1
                                FJ = FJ + FII(I)*OFFDIAG(KF)
 30                           CONTINUE
                        VJ = FJ*DIAG(J)
                        XK(J) = VJ
                        AJLAST = ALPHAJ
                        ALPHAJ = AJLAST + VJ*FJ
                        DIAG(J) = DIAG(J)*AJLAST/(ALPHAJ*XLAMBDA)
                        PJ = -FJ/AJLAST
                        DO 40 I =1,J-1
                                KU = KU + 1
                                W = OFFDIAG(KU) + XK(I)*PJ
                                XK(I) = XK(I) + OFFDIAG(KU)*VJ
                                OFFDIAG(KU) = W
 40                           CONTINUE
 20                  CONTINUE
         ENDIF
C.........
C  UPDATE PARAMETER ESTIMATES.
C.....
        DO 50 I =1,N
                THETA(I) = THETA(I) + XPERR*XK(I)/ALPHAJ
 50         CONTINUE
C.........
C  FIND THE TRACE OF COVARIANCE MATRIX.
        TRACE = 0.D0
        DO 12 I = 1,N
                TRACE = TRACE + DIAG(I)
                DO 1212 J = I+1,N
                        K = J*(J-3)/2+1+I
                        TRACE = TRACE + DIAG(J)*OFFDIAG(K)*OFFDIAG(K)
 1212                CONTINUE
 12         CONTINUE
C.........
C  FIND AN ESTIMATE TO YU (ESTYU).
       ESTYU=0.D0
       DO 864 I=1 ,NA+NB
```

Figure 118. Continued.

```
                ESTYU = ESTYU + FI(I)*THETA(I)
 864          CONTINUE
C.........
C  ESTIMATE THE DISTURBANCE, I.E. EST. DISTURB = Y(T) - ESTYU(T)
C......
       DISTURB = Y - ESTYU
C.........
C  WRITE THE RESULTS TO THE DATA FILE.
C......
       WRITE(7,*) TT,ESTYU,U
       WRITE(2,121) TT,Y,ESTY,ACTY,ESTYU,U,TRACE
 121   FORMAT(7(E12.5,2X))
       WRITE(6,122) TT,YDC,UDC,XLAMBDA
 122   FORMAT(4(E12.5,2X))
       WRITE (3,51) TT,(THETA(I), I=1,NA)
       WRITE (4,51) TT,(THETA(I), I=NA+1,NA+NB)
       WRITE (5,51) TT,(THETA(I), I=NA+NB+1,N)
 51       FORMAT(6(E12.5,2X))
C.........
C  UDATE THE FI AND FII OR DATA VECTOR.
C.......
       DO 60 I =1,(NA+NB-1)
              FI(NA+NB+1-I) = FI(NA+NB-I)
 60    CONTINUE
       FI(1) = -ESTYU
        FI(NA+1) = U
       DO 61 I =1,N-1
              FII(N+1-I) = FII(N-I)
 61    CONTINUE
       FII(1) = -Y
       FII(NA+1) = U
       FII(NA+NB+1) = ERROR
C.........
C  RETURN TO CALLING PROGRAM.
C......
 1010       RETURN
       END
```

Figure 118. Continued.

APPENDIX C - RLS USING SQRT FACTORIZATION

In this appendix we present a FORTRAN 77 subroutine for least-squares estimation based on SQRT factorization as given by Bierman [13].

```
C
C  SQRTF
C
C  PETERKA'S SQUARE-ROOT ALGORITHM.
C  THE SYSTEM CAN BE REPRESENTED AS:
C
C  Y(K)+A(1)Y(K-1)+...+A(NA)Y(K-NA) = B(1)U(K-1)+...+B(NB)
C                                     U(K-NB)+E(K)+
C                                     C(1)E(K-1)+...+C(NC)E(K-NC)
C  INPUT :
C         U        : NEW INPUT MEASUREMENT.
C         Y        : NEW OUTPUT MEASUREMENT.
C         XLAMBDA  : FORGETTING FACTOR.
C         NA       : NUMBER OF A COEFFICIENTS.
C         NB       : NUMBER OF B COEFFICIENTS.
C         N        : I.E. N = NA + NB.  NUMBER OF PARAMETERS TO BE
C                    ESTIMATED.
C         ISTART   : 0 => INITIALIZE ALL THE VARIABLES. THIS IS
C                         THE FIRST PASS THROUGH THIS SUBROUTINE.
C                    1 => CONTINUE WITH ESTIMATION.
C                    2 => STOPE ESTIMATION.
C         FI       : DATA VECTOR. ( THE PREVIOUS MEASUREMENTS OF
C                    U AND Y)
C         THETA    : A PRIORI ESTIMATE OF THE PARAMETERS.
C         SQRTP    : SQUARE-ROOT OF COVARIANCE MATRIX, ELEMENTS STORED
C         IN A VECTOR (1,1),(1,2),(2,2),(1,3),(2,3),(3,3),..
C
C  OUTPUT :
C         THETA    : UPDATED VERSION OF THE ESTIMATES.
C         FI       : UPDATED VERSION OF DATA VECTOR.
C         SQRTP    : UPDATED VERSION OF SQUARE-ROOT OF THE
C                    COVARIANCE MATRIX.
C
      SUBROUTINE SQRTF (UU,Y,XLAMBDA,N,NA,NB,ISTART,THETA,FI,SQRTP)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION THETA(100),FI(100),SQRTP(500),XK(50),U(20,20)
      DIMENSION UT(20,20),P(20,20),PFI(20)
      IF (ISTART .EQ. 2) THEN
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  RETURN TO THE CALLING PROGRAM.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
            RETURN
      END IF
      IF (ISTART .EQ. 0) THEN
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  INITIALIZATION.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
            DO 100 I =1,N
                  THETA(I)    = 0.0
                  FI(I)       = 0.0
                  XK(I)       = 0.0
100               CONTINUE
```

Figure 119. The subroutine for RLS using SQRT factorization.

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  LET SQRTP VECTOR BE A LARGE NUMBER WHICH MEANS THAT THE
C  INITIAL GUESS IS NOT GOOD AT ALL.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
               DO 111 I =1,500
                     SQRTP(I) = 1000000.0
 111               CONTINUE
               ISTART = 1
               TRACE = 0.0
       END IF
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  ESTIMATE THE ERROR. I.E. ERROR = ACT.Y - EST.Y
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       ERROR = 0.0
       DO 11 I =1,N
             ERROR = ERROR + FI(I)*THETA(I)
 11       CONTINUE
       ERROR = Y - ERROR
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  START THE PROCEDURE.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       XPERR = ERROR
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  CALCULATE KALMAN GAIN VECTOR (XK) AND SQUARE-ROOT OF COVARIANCE
C  MATRIX BY PETERKA'S SQUARE-ROOT ALGORITHM.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       SIG = XLAMBDA
       SIGSQ = XLAMBDA*XLAMBDA
       IJ = 0
       JI = 0
       DO 20 J =1,N
             J1 = J-1
             FJ = 0.0
             DO 30 I = 1,J
                   JI = JI+1
                   FJ = FJ+SQRTP(JI)*FI(I)
 30          CONTINUE
             A = SIG/XLAMBDA
             B = FJ/SIGSQ
             SIGSQ = SIGSQ+FJ*FJ
             SIG = SQRT(SIGSQ)
             A = A/SIG
             XK(J) = SQRTP(JI)*FJ
             SQRTP(JI) = A*SQRTP(JI)
             IF(J1 .GT. 0) THEN
                   DO 40 I=1,J1
                         IJ = IJ+1
                         SQP = SQRTP(IJ)
                         SQRTP(IJ) = A*(SQP-B*XK(I))
                         XK(I) = XK(I) + SQP*FJ
 40                CONTINUE
```

Figure 119. Continued.

```
                END IF
                IJ = IJ+1
 20         CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  UPDATE THE PARAMETER ESTIMATES.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        DO 109 I =1,N
                THETA(I) = THETA(I) + XPERR*XK(I)/SIGSQ
 109        CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UPDATE THE FI VECTOR OR DATA VECTOR.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        DO 110 I =1,N-1
                FI(N+1-I) = FI(N-I)
 110        CONTINUE
        FI(1) = -Y
        FI(NA+1) = UU
        FI(NA+NB+1) = ERROR
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  RETURN TO THE CALLING PROGRAM.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        RETURN
        END
```

Figure 119. Continued.

APPENDIX D - THE ILS SUBROUTINE

In this appendix we present a FORTRAN 77 subroutine for least-squares estimation based on the ILS algorithm as described by Sripada and Fisher [8].

```
C
C  ILS
C
CIMPROVED LEAST SQUARE ALGORITHM. THE SYSTEM CAN BE REPRESENTED AS:
C
CY(K)+A(1)Y(K-1)+...+A(NA)Y(K-NA)=B(1)U(K-1)+...+B(NB)U(K-NB)+E(K)
C                                  C(1)E(K-1)+...+C(NC)E(K-NC)
C
C  INPUT :
C         UU       : NEW INPUT MEASUREMENT.
C         Y        : NEW OUTPUT MEASUREMENT.
C         XLAMBDA  : FORGETTING FACTOR.
C         NA       : NUMBER OF A COEFFICIENTS.
C         NB       : NUMBER OF B COEFFICIENTS.
C         NC       : NUMBER OF C COEFFICIENTS.
C         N        : I.E. N = NA + NB + NC.  NUMBER OF PARAMETERS
C                    TO BE ESTIMATED.
C         THETA    : A PRIOR ESTIMATE OF THE PARAMETERS.
C         ISTART   : 0 => INITIALIZE ALL THE VARIABLES. THIS IS
C                         THE FIRST PASS THROUGH THIS SUBROUTINE.
C                    1 => CONTINUE WITH ESTIMATION.
C                    2 => STOP ESTIMATION.
C
C  OUTPUT :
C         THETA    : UPDATED VERSION OF THE ESTIMATES.
C         FI       : UPDATED VERSION OF DATA VECTOR.
C
C  OTHER VARIABLES USED :
C         FI         : DATA VECTOR. ( THE PREVIOUS MEASUREMENTS OF
C                      U AND Y )
C         P          : COVARIANCE MATRIX.
C         PINV       : INVERSE OF COVARIANCE MATRIX.
C         S,SNEW     : SCALING MATRICES.
C         SINV,SNEWINV
C                    : INVERSE OF SCALING MATRICES.
C         U          : SQUARE ROOTE OF P.
C         UT         : TRANSPOSE OF U.
C         PFI        : (S*P*S)(SINV*FI)
C         CON        : INITIAL VALUE FOR CONDITION NUMBER.
C         PDIAG      : INITIAL VALUE FOR DIAGONAL TERMS OF P MATRIX.
C                      (I.E. PDIAG = 1.0)
C
*****************************************************************
*
*              PROCESS REPRESENTATION
*              ----------------------
*
*      THE PROCESS CAN BE REPRESENTED IN THE FORM :
*
*    Y(T) = THETA * PHI(T) + ETA(T)
*    WHERE:                                          T
*       THETAT(T)   := (A1,A2,....,AN,B1,B2,....,BM)
*       FI(T)       := (-Y(T-1),-Y(T-2),....,-Y(T-N),U(T-TAU-1),
*                                                     T
*                      U(T-TAU-2),.........,U(T-TAU-M))
*
```

Figure 120. The ILS subroutine..

```
*****************************************************************
*****************************************************************
*         THIS ALGORITHM HAS THE FOLLOWING PROPERTIES :
*         ---------------------------------------------
*      1) ON/OFF CRITERION TO PREVENT PARAMETER DRIFT DURING PERIODS
*         OF LOW EXCITATION.
*      2) A VARIABLE FORGETING FACTOR WHICH MAINTAINS THE TRACE OF
*         COVARIANCE MATRIX AT A USER SPECIFIED VALUE.
*      3) DATA PREPROCESSING AND NORMALIZATION TO IMPROVE NUMERICAL
*         ACCURACY.
*      4) SCALING OF THE REGRESSOR VECTOR TO MINIMIZE THE CONDITION
*         NUMBER OF THE COVARIANCE MATRIX.
*      5) INDEPENDENT ESTIMATION OF THE MEAN VALUES OF THE I/O DATA
*         WHICH CAN BE USED TO ELIMINATE ERRORS DUE TO D.C. BIAS OR
*         SLOWLY DRIFTING ELEMENTS IN THE REGRESSOR VECTOR.
*
*****************************************************************
      SUBROUTINE ILS(UU,Y,XLAMBDA,N,NA,NB,ISTART,THETA,CON,PDIAG)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION P(20,20),S(20),SINV(20),U(20,20),UT(20,20)
      DIMENSION PINV(20,20),PP(20,20),PPINV(20,20),PFI(20)
      DIMENSION SNEW(20),SNEWINV(20),THETA(20),FI(20)
      IF (ISTART .EQ. 2) THEN
            RETURN
      END IF
      IF (ISTART .EQ. 0) THEN
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  INITIALIZATION.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
            DO 100 I =1,N
                  THETA(I)    = 0.0
                  FI(I)       = 0.0
                  S(I)        = 1.0
                  SINV(I)     = 1.0
                  SNEW(I)     = 1.0
                  SNEWINV(I)  = 1.0
                  PFI(I)      = 0.0
 100              CONTINUE
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  LET P = I*PDIAG   AND  PINV = P ** -1.0.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
            DO 111 I =1,N
                  DO 112 J =1,N
                        P(I,J)    = 0.0
                        PINV(I,J) = 0.0
 112                  CONTINUE
                  P(I,I) = PDIAG
                  PINV(I,I) = 1.0/PDIAG
 111              CONTINUE
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  FIND THE U OR SQUARE ROOT OF P.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 120. Continued.

```
                CALL UUT(P,N,U)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  FIND THE INITIAL SCALING MATRIX AND INVERSE OF IT.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                DO 113 I=1,N
                        SUM =0.0
                        DO 114 J=I,N
                                SUM = SUM + DABS(U(I,J))
 114                        CONTINUE
                        S(I) = 1.0/SUM
                        SINV(I) = SUM
 113                CONTINUE
                YDC=0.0
                UDC=0.0
                ISTART = 1
                FINORM = 1.0
        END IF
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  ESTIMATE THE D.C. BIAS FROM THE MEASUREMENTS.
C  HIGH-PASS FILTERING.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        YDC = XLAMBDA*YDC + (1.0 - XLAMBDA)*Y
        UDC = XLAMBDA*UDC + (1.0 - XLAMBDA)*UU
        Y = Y - YDC
        UU = UU - UDC
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  ESTIMATE THE ERROR. I.E. ERROR = ACT.Y - EST.Y
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ERROR = 0.0
        DO 11 I =1,N
                ERROR = ERROR + FI(I)*THETA(I)
 11        CONTINUE
        ERROR = Y - ERROR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  NORMALIZATION.
C
C  THE ELEMENTS IN REGRESSOR VECTOR FI(T) , DATA VECTOR , ARE
C  NORMALIZED BY N(T) := MAX(1.0 , ||FI(T)||) SO THAT ALL THE
C  ELEMENTS OF FI(T) := FI(T)/N(T) ARE <= 1.0. ALSO Y IS NORMALIZED
C  BY N(T).
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        FINORM = 1.0
        DO 21 I =1,N
                FITEMP = DABS(FI(I))
                IF(FITEMP .GT. FINORM) THEN
                        FINORM = FITEMP
                END IF
 21        CONTINUE
        IF(FINORM .GT. 1.0) THEN
                Y = Y/FINORM
                DO 22 I=1,N
```

Figure 120. Continued.

```
                    FI(I)=FI(I)/FINORM
 22                CONTINUE
       END IF
C*************************************************************
C  DATA SCALING.
C
C  WHEN SCALING IS USED IN RECURSIVE LEAST SQUARES ESTIMATION,
C  THE REGRESSOR VECTOR FI, P AND PINV ARE TRANSFORMED BY A
C  DIAGONAL SCALING MATRIX, S, BEFORE USE IN THE ESTIMATOR.
C
C  FI(T)        = SINV(T-1)*FI(T)
C  P(T-1)       = S(T-1)*P(T-1)*S(T-1)
C  PINV(T-1)    = SINV(T-1)*PINV(T-1)*SINV(T-1)
C*************************************************************
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  SCALE THE FI VECTOR.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       DO 53 I =1,N
             FI(I) = SINV(I)*FI(I)
 53       CONTINUE
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  SCALE THE COVARIANCE AND INVERSE COVARIANCE MATRICES.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       CALL VAMB(S,P,N,P)
       CALL MAVB(P,S,N,P)
       CALL VAMB(SINV,PINV,N,PINV)
       CALL MAVB(PINV,SINV,N,PINV)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  FIND THE NEW FORGETTING FACTOR.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       DO 63 I =1,N
             PFI(I)=0.0
             DO 64 J=1,N
                   PFI(I) = PFI(I) + P(I,J)*FI(J)
 64             CONTINUE
 63       CONTINUE
       R=1.0
       DO 65 I =1,N
             R = R + FI(I)*PFI(I)
 65       CONTINUE
       TRACEP = 0.0
       DO 66 I=1,N
             TRACEP = TRACEP + P(I,I)
 66       CONTINUE
       CALL NORM2(PFI,N,XNORM)
       XLAMBDA = 1.0 - (R - SQRT(R*R - 4.0*XNORM*XNORM/TRACEP))/2.0
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  FIND THE CONDITION NUMBER. I.E. C(P)=|| P || || PINV ||
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 120. Continued.

```
        CALL INFNORM(P,N,PNORM)
        CALL INFNORM(PINV,N,PINVNORM)
        COND = PNORM * PINVNORM
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  IF THE CONDITION NUMBER IS > CON, SOME USERS SPECIFIED VALUE THEN
C  STOP THE ESTIMATION.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       IF(COND .GT. CON) THEN
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  STOP THE ESTIMATION.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
              GOTO 1234
       END IF
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  UNSCALE THE FI VECTOR.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       DO 9998 I =1,N
              FI(I) = S(I) * FI(I)
 9998        CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  FIND THE ERROR BY USING THE NORMALIZED VALUES FOR FI AND Y.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       XPERR = Y
       DO 10 I =1,N
              XPERR = XPERR-THETA(I)*FI(I)
 10       CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  SCALE THE FI VECTOR.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       DO 9999 I =1,N
              FI(I) = SINV(I) * FI(I)
 9999        CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  FIND THE PP AND PPINV MATRICES.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       DIVD = 0.0
       DO 9100 I=1,N
              DIVD = DIVD + PFI(I)*FI(I)
 9100        CONTINUE
       DIVD = DIVD + XLAMBDA
       DO 9110 I =1,N
              DO 9111 J =1,N
                     PP(I,J) = -(PFI(I)*FI(J))/DIVD
 9111               CONTINUE
 9110        CONTINUE
       DO 9120 I =1,N
              PP(I,I) = 1.0 + PP(I,I)
```

Figure 120. Continued.

```
                DO 9130 J =1,N
                      PP(I,J) = PP(I,J)/XLAMBDA
 9130                CONTINUE
 9120          CONTINUE
               CALL EMPROD(PP,P,N,N,N,PP)
               DO 8301 I =1,N
                     DO 8302 J =1,N
                           PPINV(I,J) = XLAMBDA*PINV(I,J) + FI(I)*FI(J)
 8302                CONTINUE
 8301          CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  FIND THE CONDITION NUMBER. I.E. C{PP} = || PP || || PPINV ||
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
               CALL INFNORM(PP,N,PNORM)
               CALL INFNORM(PPINV,N,PINVNORM)
               COND = PNORM * PINVNORM
               IFLAG = 0
               IF(COND .GT. CON) THEN
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  IF THE CONDITION NUMBER IS > CON THEN FIND A NEW SCALING MATRIX.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  FIND THE U, SQUARE ROOT, OF PP MATRIX.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
               CALL UUT(PP,N,U)
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  OBTAIN THE NEW SCALING MATRIX.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
               DO 81 I=1,N
                           SUM = 0.0
                           DO 82 J =I,N
                                 SUM = SUM + DABS(U(I,J))
 82                           CONTINUE
                           SNEW(I) = 1.0/SUM
                           SNEWINV(I) = SUM
 81                    CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  SCALE THE PP AND PPINV MATRICES.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                     CALL VAMB(SNEW,PP,N,PP)
                     CALL MAVB(PP,SNEW,N,PP)
                     CALL VAMB(SNEWINV,PPINV,N,PPINV)
                     CALL MAVB(PPINV,SNEWINV,N,PPINV)
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  FIND THE CONDITION NUMBER. I.E. C{PP} = || PP || || PPINV ||
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                     CALL INFNORM(PP,N,PNORM)
```

Figure 120. Continued.

```
              CALL INFNORM(PPINV,N,PINVNORM)
              COND = PNORM * PINVNORM
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  IF THE CONDITION NUMBER IS > CON THEN STOP THE ESTIMATION.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
              IF(COND .GT. CON) THEN
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  STOP THE ESTIMATION.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UNSCALE THE P MATRIX, THEN FIND THE UNSCALED U MATRIX.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1234                    CALL VAMB(SINV,P,N,P)
                      CALL MAVB(P,SINV,N,P)
                      CALL UUT(P,N,U)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UNSCALE THE FI VECTOR AND UNNORMALIZED THE FI AND Y VALUES.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                      DO 101 I=1,N
                             FI(I) = S(I)*FI(I)
 101                      CONTINUE
                      IF(FINORM .GT. 1.0) THEN
                             Y = Y*FINORM
                             DO 102 I=1,N
                                    FI(I) = FI(I)*FINORM
 102                              CONTINUE
                      END IF
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UPDATE THE FI OR DATA VECTOR.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                      DO 9876 I=1,N-1
                             FI(N+1-I) = FI(N-I)
 9876                     CONTINUE
                      FI(1) = -Y
                      FI(NA+1) = UU
                      FI(NA+NB+1) = ERROR
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  RETURN TO CALLING PROGRAM.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                      RETURN
              END IF
              IFLAG = 1
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UNSCALE THE FI VECTOR.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 120. Continued.

```
                DO 5000 I =1,N
                      FI(I) = S(I) * FI(I)
 5000                CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  UPDATE THE SCALING MATRIX.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                DO 103 I =1,N
                      S(I) = S(I) * SNEW(I)
                      SINV(I) = 1.0/S(I)
 103                CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  SCALE THE FI VECTOR.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                DO 5001 I =1 ,N
                      FI(I) = SINV(I) * FI(I)
 5001                CONTINUE
        END IF
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UPDATE THE P AND PINV MATRICES.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        DO 94 I =1,N
                DO 95 J=1,N
                      P(I,J) = PP(I,J)
                      PINV(I,J) = PPINV(I,J)
 95                CONTINUE
 94         CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UPDATE THE PARAMETER ESTIMATES.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        CALL VAMB(SINV,P,N,P)
        IFLAG = 0
        DO 9001 I =1,N
                PFI(I) = 0.0
                DO 9002 J =1,N
                      PFI(I) = PFI(I) + P(I,J)*FI(J)
 9002                CONTINUE
 9001         CONTINUE
        DO 9003 I =1,N
                THETA(I) = THETA(I) + PFI(I)*XPERR
 9003         CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UNSCALE THE P AND PINV MATRICES.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        CALL VAMB(SINV,P,N,P)
        CALL MAVB(P,SINV,N,P)
        CALL VAMB(S,PINV,N,PINV)
        CALL MAVB(PINV,S,N,PINV)
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
```

Figure 120. Continued.

```
C  UNSCALE THE FI VECTOR.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      DO 6000 I =1,N
            FI(I) = S(I)*FI(I)
 6000    CONTINUE
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  RESTORE Y AND FI VECTOR TO THEIR UNNORMALIZED VALUES.
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IF(FINORM .GT. 1.0) THEN
            Y = Y*FINORM
            DO 31 I=1,N
                  FI(I) = FI(I)*FINORM
 31               CONTINUE
      END IF
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  UPDATE THE FI VECTOR.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      DO 110 I =1,N-1
            FI(N+1-I) = FI(N-I)
 110    CONTINUE
      FI(1) = -Y
      FI(NA+1) = UU
      FI(NA+NB+1) = ERROR
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C  RETURN TO THE CALLING PROGRAM.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       RETURN
      END
```

Figure 120. Continued.

APPENDIX E - THE CLS SUBROUTINE

In this appendix we present a FORTRAN 77 subroutine for least-squares estimation based on the CLS algorithm as described in this work.

```
C.....
C  [------------------------------------------------------------]
C  CLS
C
C  LEAST-SQUARES ESTIMATION OF THE PARAMETERS, USING UD
C  FACTORIZATION.  THE PLANT CAN BE REPRESENTED BY THE FOLLOWING
C  EQUATION :
C
C  INPUT :
C         T         : TIME.
C         Y         : NEW OUTPUT MEASUREMENT.
C         ACTY      : XHAT VALUE IF KNOWN. (THIS IS FOR
C                     COMPARISON PURPOSES ONLY.)
C         U         : NEW INPUT MEASUREMENT.
C         NOISE     : NOISE, V(T). (THIS IS FOR COMPARISON
C                     PURPOSES ONLY.)
C         START     : 0 => INITIALIZE ALL THE VARIABLES. THIS IS
C                          THE FIRST PASS THROUGH THIS SUBROUTINE.
C                     1 => CONTINUE WITH ESTIMATION.
C                     2 => ONLY UPDATE THE DATA VECTOR.
C         LAMBDA    : FORGETTING FACTOR. ( LOOK AT FORGET. )
C         IDIAGP    : INITIAL VALUE FOR DIAGONAL TERMS OF COVARIANCE
C                     MATRIX.
C         N         : I.E. N = NA + NB + NC
C         NA        : NUMBER OF A COEFFICIENTS.
C         NB        : NUMBER OF B COEFFICIENTS.
C         NC        : NUMBER OF C COEFFICIENTS.
C         FORGET    : FLAG FOR FORGETTING FACTOR.
C                     0 => CONSTANT.
C                     1 => VARIABLE FORGETTING FACTOR SUCH THAT
C                          IT KEEPS THE TRACE OF COVARIANCE MATRIX
C                          CONSTANT.
C                     2 => VARIABLE FORGETTING FACTOR SUCH THAT ITS
C                          MAGNITUDE DECREASES WHEN PARAMETER(S) IS/ARE
C                          TIME VARING.
C         DELTA     : FLAG ON INPUT AND OUTPUT MEASUREMENTS.
C                     0 => USE INPUT AND OUTPUT MEASUREMENTS.
C                     1 => USE DELTA INPUT AND DELTA OUTPUT.
C                     WHERE  DELTA X(T) = X(T) - X(T-1).
C         PTRACE    : FLAG FOR COVARIANCE MATRIX.
C                     0 => INITIALIZE THE COVARIANCE MATRIX WHEN
C                          ITS TRACE IS TOO BIG OR TOO SMALL.
C                     1 => DON'T INITIALIZE THE COVARIANCE MATRIX
C                          AT ANY TIME EXCEPT WHEN START=0.
C         DCVALUES: FLAG FOR D.C. CALCULATIONS.
C                     0 => ELIMINATES THE D.C. VALUES FROM INPUT AND
C                          OUTPUT MEASUREMENTS.  THIS WILL BE DONE
C                          BY HIGH PASS FILTERING OF THESE
C                          MEASUREMENTS.
C                     1 => DON'T ELIMINATE THE D.C. VALUES FROM THE
C                          MEASUREMENTS.
C         NORMALIZE:FLAG FOR NORMALIZATION CALCULATIONS.
C                   0 => NORMALIZE THE DATA VECTOR.
C                   1 => DON'T NORMALIZE THE DATA VECTOR.
C         THETA     : A PRIOR ESTIMATE OF THE PARAMETERS.
C         DOWRITE   : FLAG FOR PRINTING PURPOSES.
C                     0 => PRINT THE RESULTS TO THE DATA FILES.
C                     1 => DON'T PRINT THE RESULTS TO THE DATA
```

Figure 121. The CLS subroutine.

```
C                          FILES.
C          WTOL      : TOLERANCE FOR RATIO RATIO TEST.
C                      0.0 MEANS THE TEST IS SATISFIED IF
C                           |(RATIO RATIO) - 1.0| = 0.0.
C                      0.1 MEANS THE TEST IS SATISFIED IF
C                           |(RATIO RATIO) - 1.0| <= 0.1
C                      AND SO ON.
C          WLENGTH   : LENGTH OF THE WINDOW FOR THE RATIO RATIO TEST.
C          UPPER_TRACE : UPPER LIMIT ON THE TRACE OF THE COVARIANCE
C                        MATRIX, USED FOR RESETTING PURPOSES.
C          LOWER_TRACE : LOWER LIMIT ON THE TRACE OF THE COVARIANCE
C                        MATRIX, USED FOR RESETTING PURPOSES.
C          FILTER_VALUE: THIS VALUE WILL BE USED TO LOW PASS FILTER
C                        THE THETA VECTOR (UNKNOWN COEFFICIENTS).
C          RATIO       : |ERROR/Y|, USED FOR CHECKING BEFORE UPDATING
C                        DIAGONAL TERMS OF THE COVARIANCE MATRIX IN THE
C                        CASE OF VARYING DYNAMICS.
C   OUTPUT :
C          THETA       : UPDATED VERSION OF ESTIMATE VECTOR.
C          TRACE       : TRACE OF THE COVARIANCE MATRIX.
C          LAMBDA      : UPDATED VERSION OF FORGETTING FACTOR IF
C                        FORGET IS 1 OR 2.  ( LOOK AT FORGET
C                        VARIABLE.)
C   [-------------------------------------------------------------]
C........
      SUBROUTINE CLS (T,Y,ACTY,U,NOISE,START,LAMBDA,IDIAGP,N,NA,
     &                          NB,NC,FORGET,DELTA,PTRACE,DCVALUES,
     &                          NORMALIZE,THETA,TRACE,DOWRITE,WTOL,
     &                          WLENGTH,UPPER_TRACE,LOWER_TRACE,
     &                          FILTER_VALUE,RATIO)
        IMPLICIT NONE
        REAL*8     T,Y,ACTY,U,NOISE,LAMBDA,IDIAGP,THETA,TRACE
        REAL*8     FI,OFFDIAG,DIAG,YDC,UDC,OLDU,OLDY,TEMPU,TEMPY
        REAL*8     NORM,ERROR,GAIN,DTHETA,THETADC,ATHETADC,XHAT
        REAL*8     CORRECT,P,SERROR2,SERROR,VAR_ERROR,SIGMA,JUNK
        REAL*8     FIXHAT,OLDERROR,RATIO,EHAT,VHAT,FIEVHAT,NERROR
        REAL*8     OLD_TRACE,OLD_RATIO,CHECK_RATIO,NEW_RATIO
        REAL*8     ATIEH_OFF,ATIEH_DIAG,UPPER_TRACE,LOWER_TRACE
        REAL*8     FILTER_VALUE,WTOL,PFI,XNORM,ORG_LAMBDA
        REAL*8     KEEP_OFF,KEEP_DIAG,ALL_WINDOW,SECOND_TEST
        INTEGER*4  START,N,NA,NB,NC,FORGET,DELTA,PTRACE
        INTEGER*4  I,NUP,DCVALUES,NORMALIZE,CHECK,UPDATE,J,K
        INTEGER*4  ADD_TO_P,DOWRITE,TALIEH,AMIR,ATIEH_WINDOW
        INTEGER*4  ATIEH_FLAG,WLENGTH,KEEP_FLAG
        DIMENSION  THETA(500),DIAG(500),OFFDIAG(500),FI(500)
        DIMENSION  GAIN(500),DTHETA(500),THETADC(500),ATHETADC(500)
        DIMENSION  CORRECT(500),P(500,500),FIXHAT(500),FIEVHAT(500)
        DIMENSION  ATIEH_OFF(500),ATIEH_DIAG(500),PFI(500)
        DIMENSION  KEEP_OFF(500),KEEP_DIAG(500),ALL_WINDOW(500)
C.....
C  INITIALIZATION.
C.....
       IF (START .EQ. 0) THEN
             NUP = N*(N-1)/2
             DO 10 I=1, NUP
                   THETA(I)              = 0.D0
```

Figure 121. Continued.

```
                    FI(I)                = 0.D0
                    FIXHAT(I)            = 0.D0
                    FIEVHAT(I)           = 0.D0
                    OFFDIAG(I)           = 0.D0
                    ATIEH_OFF(I)         = 0.D0
                    GAIN(I)              = 0.D0
                    DTHETA(I)            = 0.D0
                    THETADC(I)           = 0.D0
                    ATHETADC(I)          = 0.D0
                    CORRECT(I)           = 0.D0
                    DIAG(I)              = IDIAGP
                    ATIEH_DIAG(I)        = IDIAGP
                    DO 11 J=1,NUP
                           P(I,J)        = 0.D0
                           IF (I .EQ. J) THEN
                                  P(I,J)= IDIAGP
                           END IF
11                      CONTINUE
10                CONTINUE
                  YDC          = 0.D0
                  UDC          = 0.D0
                  XHAT         = 0.D0
                  START        = 1
                  OLDU         = 0.D0
                  OLDY         = 0.D0
                  TRACE        = 0.D0
                  TEMPU        = 0.D0
                  TEMPY        = 0.D0
                  NORM         = 0.D0
                  ERROR        = 0.D0
                  SERROR2      = 0.D0
                  SERROR       = 0.D0
                  VAR_ERROR    =0.D0
                  SIGMA        = 0.D0
                  JUNK         = 0.D0
                  ADD_TO_P     = 0
                  NERROR       = 0.D0
                  TALIEH       = 0
                  AMIR         = 0
                  OLDERROR     =0.D0
                  OLD_TRACE    = 1.D0
                  OLD_RATIO    = 1.D0
                  NEW_RATIO    = 1.D0
                  CHECK_RATIO= 0.D0
                  ATIEH_FLAG = 0
                  ATIEH_WINDOW= 0
                  ORG_LAMBDA = LAMBDA
                  KEEP_FLAG  = 0
                  FILTER_VALUE= 0.5
                  SECOND_TEST = LAMBDA**WLENGTH
                  SECOND_TEST=SECOND_TEST+SECOND_TEST*2.0*WLENGTH/100.0
            END IF
            IF (AMIR .GT. 5*N) THEN
                  AMIR=100
                  FILTER_VALUE = LAMBDA
            ELSE
                  AMIR = AMIR+1
                  FILTER_VALUE = 0.5*LAMBDA
            END IF
```

Figure 121. Continued.

```
C.....
C  IF DELTA IS EQUAL TO 1 THEN USE DELTA INPUT AND DELTA OUTPUT
C  VALUES.
C.....
       IF (DELTA .EQ. 1) THEN
              TEMPU = U
              TEMPY = Y
              U     = TEMPU - OLDU
              Y     = TEMPY - OLDY
       END IF
C.....
C  ELIMINATE THE D.C. VALUES FROM THE MEASUREMENTS.
C.....
       IF (DCVALUES .EQ. 0) THEN
              YDC = LAMBDA*YDC + (1.D0 - LAMBDA)*Y
              UDC = LAMBDA*UDC + (1.D0 - LAMBDA)*U
              Y   = Y - YDC
              U   = U - UDC
       END IF
C.....
C  IF START EQUAL TO 2 THEN UPDATE THE DATA VECTOR AND RETURN
C  TO THE CALLING PROGRAM.
C.....
       IF (START .EQ. 2) THEN
             GOTO 85
       END IF
C.....
C  FIND THE ERROR IN ESTIMATING THE Y.
C.....
       ERROR = Y
       DO 40 I=1,N
             ERROR = ERROR - FI(I)*THETA(I)
 40      CONTINUE
C.....
C  NORMALIZATION.
C  THE ELEMENTS IN REGRESSOR VECTOR FI, DATA VECTOR, ARE
C  NORMALIZED BY MAX(|FI|,Y) SO THAT ALL THE
C  ELEMENTS OF FI ARE <= 1.D0.  ALSO Y AND U ARE NORMALIZED
C  BY NORM.
C.....
       IF (NORMALIZE .EQ. 0) THEN
            NORM = DABS(Y)
            DO 20 I=1,N
                  IF (DABS(FI(I)) .GT. NORM) THEN
                        NORM = DABS(FI(I))
                  END IF
 20            CONTINUE
             IF(NORM .GT. 0.D0) THEN
                  Y = Y/NORM
                  DO 30 I=1,N
                        FI(I) = FI(I)/NORM
```

Figure 121. Continued.

```
 30                          CONTINUE
                END IF
        END IF
C.....
C  FIND THE NEW FORGETTING FACTOR (LAMBDA).
C.....
C  FOR CONSTANT TRACE.
C.....
         IF (FORGET .EQ. 1) THEN
                CALL UD_TO_POS(P,N,ATIEH_OFF,ATIEH_DIAG)
                TRACE = 0.D0
                DO I=1,N
                        PFI(I) = 0.D0
                        DO J=1,N
                                PFI(I) = PFI(I) + P(I,J)*FI(J)
                        END DO
                        TRACE = TRACE + P(I,I)
                END DO
                LAMBDA = 1.D0
                DO I=1,N
                        LAMBDA = LAMBDA + FI(I)*PFI(I)
                END DO
                CALL NORM2 (PFI,N,XNORM)
                IF (TRACE .NE. 0.D0) THEN
                        LAMBDA = 1.D0 -(LAMBDA - DSQRT(LAMBDA*LAMBDA -
     &                                 4.D0*XNORM*XNORM/TRACE))/2.D0
                ELSE
                        LAMBDA = ORG_LAMBDA
                END IF
        END IF
C.....
C  FOR TIME VARING PARAMETERS.
C  FORTESCUE ET AL. 1981, AUTOMATICA VOL. 17 NO. 6 PG. 831, 1981
        IF (FORGET .EQ. 2) THEN
                NERROR = NERROR + 1.D0
                SERROR2 = SERROR2 + (ERROR*ERROR)
                SERROR = SERROR + ERROR
                VAR_ERROR = (SERROR2 - (SERROR*SERROR)/NERROR)/NERROR
                SIGMA = 1000.D0*VAR_ERROR
                JUNK = 0.D0
                DO 45 I=1,N
                        JUNK = JUNK + GAIN(I)*FI(I)
 45                CONTINUE
                IF (ERROR .EQ. 0.D0) THEN
                        ERROR = 0.00000001
                END IF
                JUNK = (JUNK + 1.D0)/(ERROR*ERROR)
                JUNK = JUNK*SIGMA
                IF (JUNK .LE. 0.D0) THEN
                        JUNK = 33.34
                END IF
                LAMBDA = 1.D0 - 1.D0/JUNK
                IF (LAMBDA .GT. 0.99) THEN
                        LAMBDA = 0.99
                END IF
        END IF
```

Figure 121. Continued.

```
C  CALL THE UD SUBROUTINE TO UPDATE THE COVARIANCE MATRIX AND
C  THE GAIN VECTOR.  ALSO UPDATE THE ESTIMATE VECTOR (THETA).
C.....
       CALL UD (GAIN,ERROR,ATIEH_DIAG,ATIEH_OFF,LAMBDA,N,FI,AMIR,
     &          FORGET)
       IF (KEEP_FLAG .EQ. 0) THEN
              DO I=1,(N*(N-1)/2)
                     KEEP_OFF(I) = ATIEH_OFF(I)
              END DO
              DO I=1,N
                     KEEP_DIAG(I) = ATIEH_DIAG(I)
              END DO
       END IF
C.....
C  FIND THE TRACE OF THE COVARIANCE MATRIX.
       TRACE = 0.D0
       DO 50 I=1,N
              TRACE = TRACE + ATIEH_DIAG(I)
              DO 60 J = I+1, N
                     K = J*(J-3)/2+1+I
                     TRACE = TRACE + ATIEH_DIAG(J)*
    &                                ATIEH_OFF(K)*ATIEH_OFF(K)
 60                CONTINUE
 50       CONTINUE
C......
C  FIND THE NEW_RATIO OF TRACES.
       IF (FORGET .EQ. 0) THEN
              NEW_RATIO = 1000.D0
              IF (TRACE .NE. 0.D0) NEW_RATIO = OLD_TRACE/TRACE
              IF ((TRACE .EQ. 0.D0) .AND. (OLD_TRACE .EQ. 0.D0)) THEN
                     NEW_RATIO = 1.D0
              END IF
C......
C  RICHNESS TEST. RATIO-RATIO TEST.
C......
              CHECK_RATIO = OLD_RATIO/NEW_RATIO
              IF (DABS(CHECK_RATIO - 1.D0) .LT.  WTOL) THEN
                     ALL_WINDOW(WLENGTH-ATIEH_WINDOW) = TRACE
                     ATIEH_WINDOW = ATIEH_WINDOW + 1
                     IF (ATIEH_WINDOW .GE. WLENGTH) THEN
C......
C  CHECK TO MAKE SURE THAT IT IS NOT A CONSTANT TRACE.
C......
                        IF((DABS(ALL_WINDOW(WLENGTH)/TRACE).LE.
     &                           SECOND_TEST) .OR.
     &                           (ATIEH_FLAG.EQ.1)) THEN
                            ATIEH_WINDOW = WLENGTH
                            IF (ATIEH_FLAG .EQ. 0) THEN
                                   DO I=1, (N*(N-1)/2)
                                   OFFDIAG(I) = ATIEH_OFF(I)
                             END DO
                                   DO I=1, N
                                   DIAG(I) = ATIEH_DIAG(I)
                            END DO
                            ATIEH_FLAG = 1
```

Figure 121. Continued.

```
                                ELSE
                                        DO I=1, (N*(N-1)/2)
                                        ATIEH_OFF(I) = OFFDIAG(I)
                                    END DO
                                        DO I=1, N
                                        ATIEH_DIAG(I) = DIAG(I)
                                    END DO
                                END IF
                                GOTO 1234
                            ELSE
                                ATIEH_WINDOW = 0
                                DO I=1,N
                                  THETA(I) =THETA(I)+ GAIN(I)*ERROR
                                END DO
                            END IF
                        END IF
                ELSE
                        ATIEH_WINDOW = 0
                        IF (ATIEH_FLAG .EQ. 1) THEN
                            ATIEH_FLAG = 0
                                DO I=1, (N*(N-1)/2)
                                ATIEH_OFF(I) = OFFDIAG(I)
                            END DO
                                DO I=1, N
                                ATIEH_DIAG(I) = DIAG(I)
                            END DO
                            CALL UD (GAIN,ERROR,ATIEH_DIAG,ATIEH_OFF,
     &                                  LAMBDA,N,FI,AMIR,FORGET)
                        END IF
                END IF
        END IF
        IF ((ATIEH_WINDOW .LT. WLENGTH) .AND. (KEEP_FLAG .EQ. 0)) THEN
                DO 41 I=1,N
                    THETA(I) = THETA(I) + GAIN(I)*ERROR
 41             CONTINUE
         END IF
C....
C  IF TRACE < LOWER_TRACE OR TRACE > UPPER_TRACE THEN HOLD THE
C  COVARIANCE MATRIX CONSTANT.
C.....
        IF ((PTRACE .EQ. 0) .AND. (AMIR .GE. 100)) THEN
                IF ((TRACE .LT. LOWER_TRACE) .OR.
     &              (TRACE .GT. UPPER_TRACE)) THEN
                    KEEP_FLAG=1
                    DO I=1,(N*(N-1)/2)
                            ATIEH_OFF(I) = KEEP_OFF(I)
                    END DO
                    DO I=1,N
                            ATIEH_DIAG(I) = KEEP_DIAG(I)
                    END DO
                ELSE
                        KEEP_FLAG = 0
                        AMIR = 0
                END IF
        END IF
C.....
C  FIND DELTA THETA AND ALSO LOW PASS THE THETA ESTIMATES.
C.....
        IF (FORGET .EQ. 0) THEN
```

Figure 121. Continued.

```
                ADD_TO_P = 0
                DO 42 I=1,N
                       DTHETA(I)   = GAIN(I)*ERROR
                       THETADC(I)  = FILTER_VALUE*THETADC(I) +
     &                                (1.D0 - FILTER_VALUE)*GAIN(I)*ERROR
                       ATHETADC(I)= FILTER_VALUE*ATHETADC(I)+
     &                                (1.D0 - FILTER_VALUE)*DABS(GAIN(I)*
     &                               ERROR)
                       IF (DABS(ATHETADC(I)) .GT. 0.D0) THEN
                          CORRECT(I) = (DABS(DTHETA(I)/ATHETADC(I)))
                       ELSE
                          CORRECT(I) = 1.D0
                       END IF
                       IF (CORRECT(I) .GT. 1.D0) THEN
                             ADD_TO_P = ADD_TO_P + 1
                       END IF
 42                CONTINUE
C.....
C  IF ADD_TO_P FLAG IS > N/2 THEN MODIFY THE DIAGONAL TERMS OF THE
C  COVARIANCE MATRIX.
C.....
               IF (DABS(ERROR)/(DABS(Y)+0.000001) .GT. RATIO) THEN
                     IF ((ADD_TO_P.GE.N/2) .AND. (AMIR.GE.100)) THEN
                           TALIEH=0
                           DO 43 I=1,(NA+NB)
                             IF (CORRECT(I) .GT. 1.D0) THEN
                               IF ((OLDERROR*FI(I)) .NE. 0.D0) THEN
                                  IF (DABS(OLDERROR) .LT.
     &                                DABS(ERROR)) THEN
                                     JUNK = DABS(DTHETA(I)/(OLDERROR*
     &                                         FI(I)))
                                   ATIEH_DIAG(I,I) = JUNK
                                   KEEP_FLAG = 0
                                        END IF
                               END IF
                               UPDATE = 0
                             END IF
 43                            CONTINUE
                           ERROR = OLDERROR
                     ELSE
                           TALIEH = TALIEH + 1
                     END IF
               END IF
               OLDERROR=ERROR
       END IF
C.....
C  UNNORMALIZE THE DATA VECTOR,Y AND U.
C.....
 1234       IF ((NORMALIZE .EQ. 0) .AND. (NORM .GT. 0.D0)) THEN
              Y = Y*NORM
              DO 80 I=1,N
                    FI(I) = FI(I)*NORM
 80              CONTINUE
       END IF
C.....
C  FIND THE XHAT (ESTIMATE OF THE ACTUAL OUTPUT.)
```

Figure 121. Continued.

```
 85     XHAT = 0.D0
        DO 93 I=1, (NA+NB)
               XHAT = XHAT + THETA(I)*FIXHAT(I)
 93     CONTINUE
C.....
C  FIND ESTIMATE OF VHAT AND EHAT.
        VHAT=Y-XHAT
        EHAT=VHAT
        DO I=1,NA
               EHAT = EHAT + THETA(I)*FIEVHAT(I)
        END DO
        DO I=1,NC
               EHAT = EHAT - THETA(NA+NB+I)*FIEVHAT(NA+I)
        END DO
C.....
C  UPDATE THE FI AND FIXHAT VECTORS.
        DO 90 I=1,N-1
               FI(N+1-I) = FI(N-I)
               FIXHAT(N+1-I) = FIXHAT(N-I)
 90        CONTINUE
        FI(1)             = -Y
        FIXHAT(1)         = -XHAT
        IF (NA .GT. 0) THEN
               FI(NA+1)             = U
               FIXHAT(NA+1)         = U
        END IF
        IF (NC .GT. 0) THEN
               FI(NA+NB+1)          = EHAT
               DO I=1,(NA+NC)
                      FIEVHAT(NA+NC+1-I) = FIEVHAT(NA+NC-I)
               ENDDO
               FIEVHAT(1)     = VHAT
               FIEVHAT(NA+1) = EHAT
        END IF
C.....
C  WRITE THE RESULTS TO THE DATA FILES.
C.....
        IF (DOWRITE .EQ. 0) THEN
               WRITE(2,*) T,(THETA(I), I=1,NA)
               WRITE(3,*) T,(THETA(I), I=NA+1,NA+NB)
               WRITE(4,*) T,(THETA(I), I=NA+NB+1,N)
               WRITE(7,*) T,TRACE,OLD_TRACE,NEW_RATIO,OLD_RATIO,
     &                      CHECK_RATIO
               WRITE(5,1010) T,YDC,UDC,LAMBDA,ADD_TO_P,ERROR,EHAT
               WRITE(6,1000) T,Y,(Y-ERROR),ACTY,XHAT,NOISE,VHAT,
     &                       U,TRACE
 1000          FORMAT(9(E12.5,2X))
 1010          FORMAT(4(E12.5,2X),I2)
        END IF
C.....
C  IF WE ARE USING THE DELTA VALUES THEN KEEP THE U AND Y VALUES
C  FOR NEXT ITERATION.
        IF (DELTA .EQ. 1) THEN
               OLDU = TEMPU
               OLDY = TEMPY
        END IF
```

Figure 121. Continued.

```
C  KEEP THE TRACE AND OLD_RATIO VALUES FOR NEXT TIME.
C.....
       OLD_TRACE = TRACE
       OLD_RATIO = NEW_RATIO
C.....
C  RETURN TO THE CALLING PROGRAM.
       RETURN
       END
C........
C  [-------------------------------------------------------------]
C         UD
C
C  PURPOSE :
C         TO UPDATE THE COVARIANCE MATRIX AND GAIN VECTOR USING
C  U-D FACTORIZATION METHOD.
C
C  INPUT :
C         ERROR    : ESTIMATE OF THE ERROR. (ERROR = Y(T) - EST. Y(T))
C         DIAG     : VECTOR STORED DIAGONAL MATRIX ( D ).
C         OFFDIAG  : VECTOR STORED UPPER DIAGONAL MATRIX ( U ).
C         XLAMBDA  : FORGETTING FACTOR (LAMBDA).
C         N        : ORDER OF D AND U MATRICES.
C         FII      : DATA VECTOR.
C
C  OUTPUT :
C         GAIN     : UPDATED VERSION OF THE GAIN VECTOR.
C         DIAG     : UPDATED VERSION OF THE D MATRIX.
C         OFFDIAG  : UPDATED VERSION OF THE U MATRIX.
C
C
C  NOTE THAT :
C                          T
C  COVARIANCE MATRIX = UDU
C  [-------------------------------------------------------------]
C........
       SUBROUTINE UD  (XK,XPERR,DIAG,OFFDIAG,XLAMBDA,N,FII,AMIR,
     &                 FORGET)
       IMPLICIT NONE
       REAL*8          XK,XPERR,DIAG,OFFDIAG,XLAMBDA,FII,FJ,VJ
       REAL*8          ALPHAJ,AIRAJ,AJLAST,PJ,W
       INTEGER*4       N,KF,KU,J,I,AMIR,FORGET
       DIMENSION       XK(500),DIAG(500),OFFDIAG(500),FII(500)
C........
C  START THE PROCEDURE TO UPDATE THE COVARIANCE MATRIX AND THE GAIN
C  VECTOR USING U-D FACTORIZATION METHOD.
C.....
       FJ = FII(1)
       VJ = DIAG(1)*FJ
       XK(1) = VJ
       ALPHAJ = 1.D0 + VJ*FJ
       AIRAJ = 1.D0/(ALPHAJ*XLAMBDA)
       DIAG(1) = DIAG(1)*AIRAJ
       IF ( N .GT. 1 ) THEN
              KF = 0
              KU = 0
              DO 20 J =2,N
```

Figure 121. Continued.

```
                        FJ = FII(J)
                        DO 30 I =1,J-1
                              KF = KF+1
                               FJ = FJ + FII(I)*OFFDIAG(KF)
 30                        CONTINUE
                        VJ = FJ*DIAG(J)
                        XK(J) = VJ
                        AJLAST = ALPHAJ
                        ALPHAJ = AJLAST + VJ*FJ
                        AIRAJ = AJLAST/(ALPHAJ*XLAMBDA)
                        DIAG(J) = DIAG(J)*AIRAJ
                        PJ = -FJ/AJLAST
                        DO 40 I =1,J-1
                              KU = KU + 1
                              W = OFFDIAG(KU) + XK(I)*PJ
                              XK(I) = XK(I) + OFFDIAG(KU)*VJ
                              OFFDIAG(KU) = W
 40                        CONTINUE
 20                CONTINUE
       END IF
C.....
C  FIND THE GAIN VECTOR.
C.....
       DO 50 I=1,N
             XK(I) = XK(I)/ALPHAJ
 50       CONTINUE
C.....
C  RETURN TO THE CALLING PROGRAM.
C.....
       RETURN
       END
```

Figure 121. Continued.

APPENDIX F - THE IVAML SUBROUTINE

In this appendix we present a FORTRAN 77 subroutine for instrumental variable approximate maximum likelihood estimation as described by Young and Jakeman [30].

```
C
C
C  IVAML
C
C  *********      NOTE : THIS ROUTINE USES UD TO UPDATE THE P.
C  PURPOSE :
C                 TO ESTIMATE THE COEFFICIENTS OF A, B, C AND D
C                 POLYNOMIALS IN THE FOLLOWING BLOCK DIAGRAM, BY
C                 USING THE REFINED IVAML METHOD.
C
C                 IV        :       INSTRUMENTAL VARIABLE.
C                 AML       :        APPROXIMATE MAXIMUM LIKELIHOOD.
C
C                 IV METHOD USED TO ESTIMATE THE DYNAMIC PART OF
C                 THE SYSTEM. (I.E. A AND B).
C
C                 AML METHOD USED TO ESTIMATE THE DISTURBANCES V(T).
C                 (I.E. C AND D).
C
C
C
C           +------------+
C   E(T) OR |  C(Z**-1)  |  V(T) OR VNEW
C   ------->|  --------  |-----------------|
C   ENEW    |  D(Z**-1)  |                 |
C           +------------+                 |
C                                          |
C           +------------+                 | +
C   U(T) OR |  B(Z**-1)  |   ACT. Y(T)    + \/    Y(T) OR YNEW(T)
C   ------->|  --------  |---------------->O--------------------->
C   UNEW    |  A(Z**-1)  |
C           +------------+
C
C
C
C                                +================+
C          B(Z**-1)              | C(Z**-1)       |
C   Y(T) = -------- U(T) +       | -------- E(T)  |
C          A(Z**-1)              | D(Z**-1)       |
C                                +================+
C                                |<---- V(T) --->|
C
C  WHERE :
C              Y(T)   : OUTPUT OF THE SYSTEM AT TIME T.
C              U(T)   : INPUT OF THE SYSTEM AT TIME T.
C              E(T)   : INPUT TO THE MODEL OF DISTURBANCE.
C              V(T)   : DISTURBANCES AT TIME T.
C       A(Z**-1)   : 1 + A(1)Z**-1 + ... + A(NA)Z**-NA
C       B(Z**-1)   :     B(1)Z**-1 + ... + B(NB)Z**-NB
C       C(Z**-1)   : 1 + C(1)Z**-1 + ... + C(NC)Z**-NC
C       D(Z**-1)   : 1 + D(1)Z**-1 + ... + D(ND)Z**-ND
C
C
C
C**************************
C
```

Figure 122. FORTRAN subroutine for IVAML identification.

```
C  INPUT :
C         THETA   : VECTOR OF ORDER (NA+NB) CONTAINING THE
C                   ESTIMATE OF COEFFICIENTS IN A AND B
C                   POLYNOMIALS.
C
C         THETAD  : VECTOR OF ORDER (NC+ND) CONTAINING THE
C                   ESTIMATE OF COEFFICIENTS IN C AND D
C                   POLYNOMIALS.
C
C         Y       : NEW MEASUREMENT OF THE SYSTEM OUTPUT.
C
C         U       : NEW MEASUREMENT OF THE SYSTEM INPUT.
C
C         NA      : THE NUMBER OF COEFFICIENTS IN A POLYNOMIAL.
C
C         NB      : THE NUMBER OF COEFFICIENTS IN B POLYNOMIAL.
C
C         NC      : THE NUMBER OF COEFFICIENTS IN C POLYNOMIAL.
C
C         ND      : THE NUMBER OF COEFFICIENTS IN D POLYNOMIAL.
C
C         LAMBDA  : THE FORGETTING FACTOR.
C
C         DIAGP   : SCALAR PARAMETER USED TO GIVE COVRIANCE
C                   MATRICES (FOR BOTH PROCEDURES) AN INITIAL
C                   VALUE.
C
C         START   : FLG TO BE USED FOR STARTING THE RECURSION :
C                   IF  START = 0 THEN APPROPRIATE VALUES ARE
C                                 FIRST SET. THEN THE PARAMETERS
C                                ARE UPDATED USING THE AVAILABLE
C                                DATA Y AND U.
C                   IF  START = 1 THEN ALL PARAMETERS ARE UPDATED.
C  OUTPUT :
C         THETA       : UPDATED ESTIMATE OF A AND B POLYNOMIALS.
C         THETAD      : UPDATED ESTIMATE OF C AND D POLYNOMIALS.
C         TRACE_AB: TRACE OF COVARIANCE MATRIX FOR IV PROCEDURE.
C         TRACE_CD: TRACE OF COVARIANCE MATRIX FOR AML PROCEDURE.
C
      SUBROUTINE IVAML(THETA,THETAD,Y,U,NA,NB,NC,ND,
     &                 LAMBDA,TRACE_AB,TRACE_CD,DIAGP,START,
     &                     T,ACTV,ACTY,ST,AMIRY)
      IMPLICIT NONE
      REAL*8 THETA,THETAD,Y,U,LAMBDA,TRACE_AB,TRACE_CD,DIAGP,VNEW
      REAL*8 PA,PB,PC,PD,PCA,W,FI,ZETA,FIZETA,FIFI,TEMP1,TEMP2,ETA
      REAL*8 FIETA,FIETAF,FIYF,THETACA,DIAG,TEMPM1,ETAF,FID,PSI,
     &       IRTEMP
      REAL*8 FIPSI,YF,ERROR,ERRORD,TLAMBDA,V,E,PPC,PPA,TSTABD,ENEW,T
      REAL*8 TSTAB,AMY,OFFDIAG,OFFDIAGD,DIAGD,ITEMP1,DIAGPD
      REAL*8 ACTV,ACTY,AMIRY
      INTEGER*4 I,J,N,NA,NB,NC,ND,START,NCA,IST,ST
      DIMENSION THETA(70),THETAD(70),TSTAB(70),TSTABD(70)
      DIMENSION PA(70),PB(70),PC(70),PD(70),PCA(70),PPA(70)
      DIMENSION FI(70),ZETA(70),FIZETA(70,70),FIFI(70,70),PPC(70)
```

Figure 122. Continued.

```
      DIMENSION TEMP1(70),TEMP2(70),ETA(70),FIETA(70,70)
      DIMENSION FIETAF(70,70),FIYF(70),THETACA(70),DIAG(70)
      DIMENSION TEMPM1(70,70),ETAF(70),OFFDIAGD(70),OFFDIAG(70)
      DIMENSION FID(70),PSI(70),FIPSI(70,70),DIAGD(70),ITEMP1(70)
C.....
C  IF START=0 THEN DO THE INITIALIZATION FIRST.
C.....
      IF (START .EQ. 0) THEN
            DIAGPD = DIAGP*1000000.0
            DO 1010 I =1, 50
                  OFFDIAG(I)        = 0.0
                  OFFDIAGD(I)       = 0.0
1010        CONTINUE
            DO 1 I =1,50
                  PA(I)             = 0.0
                  PB(I)             = 0.0
                  PC(I)             = 0.0
                  PD(I)             = 0.0
                  PCA(I)            = 0.0
                  PPA(I)            = 0.0
                  PPC(I)            = 0.0
                  THETA(I)          = 0.0
                  FI(I)             = 0.0
                  ZETA(I)           = 0.0
                  ITEMP1(I)         = 0.0
                  TEMP1(I)          = 0.0
                  TEMP2(I)          = 0.0
                  ETA(I)            = 0.0
                  FIYF(I)           = 0.0
                  THETACA(I)        = 0.0
                  ETAF(I)           = 0.0
                  THETAD(I)         = 0.0
                  FID(I)            = 0.0
                  PSI(I)            = 0.0
                  TSTABD(I)         = 0.0
                  TSTAB(I)          = 0.0
                  DO 2 J =1,50
                        FIZETA(I,J)       = 0.0
                        FIFI (I,J)        = 0.0
                        FIETA (I,J)       = 0.0
                        FIETAF(I,J)       = 0.0
                        TEMPM1(I,J)       = 0.0
                        FIPSI(I,J)        = 0.0
2                     CONTINUE
                  DIAG(I)                 = DIAGP
                  DIAGD(I)                = DIAGPD
1             CONTINUE
            TRACE_AB = 0.0
            TRACE_CD = 0.0
            AMY      = 1.0
            IST      = 0.0
C.....
C  OPEN THE DATA FILES FOR INPUTS AND OUTPUTS.
C.....
      OPEN(2,FILE='LQGIVUD.DAT',STATUS='NEW')
      OPEN(3,FILE='LQGAIVUD.DAT',STATUS='NEW')
      OPEN(4,FILE='LQGBIVUD.DAT',STATUS='NEW')
```

Figure 122. Continued.

```
      OPEN(5,FILE='LQGCIVUD.DAT',STATUS='NEW')
      OPEN(6,FILE='LQGDIVUD.DAT',STATUS='NEW')
      WRITE(2,*) 'IVUD.DAT Y,ACT.Y,W,V,ACTV,U,TR. OF P,TR. OF DP'
      WRITE(2,1001) 8
      WRITE(3,*) 'AIVUD.DAT   A COEFFICIENTS. A(1),A(2),...'
      WRITE(3,1001) NA
      WRITE(4,*) 'BIVUD.DAT   B COEFFICIENTS. B(1),B(2),...'
      WRITE(4,1001) NB
      WRITE(5,*) 'CIVUD.DAT   C COEFFICIENTS. C(1),C(2),...'
      WRITE(5,1001) NC
      WRITE(6,*) 'DIVUD.DAT   D COEFFICIENTS. D(1),D(2),...'
       WRITE(6,1001) ND
 1001      FORMAT(/,I4)
      END IF
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C                          IV       METHOD
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
      DO 333 I =1,NA
            PPA(I) = PA(NA+1-I)
 333     CONTINUE
      DO 334 I =1,NC
            PPC(I) = PC(NC+1-I)
 334     CONTINUE
      PPA(NA+1) = 1.0
      PPC(NC+1) = 1.0
      CALL MULT(NC,NA,PPC,PPA,NCA,PCA)
      DO 335 I =1,INT((NCA+1)/2)
            IRTEMP = PCA(I)
            PCA(I) = PCA(NCA+2-I)
            PCA(NCA+2-I) = IRTEMP
 335     CONTINUE
      DO 336 I=1,NCA
            PCA(I) = PCA(I+1)
 336     CONTINUE
C.............................
C
C                FIND W.
C.....
      N=NA+NB
      CALL EMVV(THETA,FI,N,W)
C.....................................
C
C  UPDATE FIZETA AND FIETAF MATRICES.
C.....
      DO 70 I =1,N
            DO 80 J =NCA,2,-1
                  FIZETA(I,J) = FIZETA(I,J-1)
                  FIETAF(I,J) = FIETAF(I,J-1)
 80               CONTINUE
 70     CONTINUE
      DO 90 I =1,N
            FIZETA(I,1) = -ZETA(I)
            FIETAF(I,1) = -ETAF(I)
 90     CONTINUE
C.................................................
```

Figure 122. Continued.

```
C  FIND ZETA VECTOR, THE INSTRUMENTAL VARIABLE VECTOR.
C.....
       CALL EMVPROD(FIZETA,PCA,N,NCA,TEMP1)
       CALL EMVPROD(FIFI,PD,N,ND,TEMP2)
       DO 10 I =1,N
              ZETA(I) = TEMP1(I) + TEMP2(I) + FI(I)
 10       CONTINUE
C.....................................
C                 FIND ETAF.
C.....
       CALL EMVPROD(FIETAF,PCA,N,NCA,TEMP1)
       CALL EMVPROD(FIETA,PD,N,ND,TEMP2)
       DO 20 I=1,N
              ETAF(I) = TEMP1(I) + TEMP2(I) + ETA(I)
 20       CONTINUE
C.....................................
C                 FIND YF.
C.....
       DO 140 I=1,NCA
              THETACA(I) = PCA(I)
 140       CONTINUE
       N = NA+NC+ND
       CALL EMVV(FIYF,THETACA,N,YF)
       YF = YF + Y
C.....................................
C  FIND THE ERROR IN THE ESTIMATION.
C.....
       CALL EMVV(ETAF,THETA,N,ERROR)
       ERROR = YF-ERROR
C.....................................
C  UPDATE THETA AND P.  OR CALL CLS TO UPDATE THETA AND P.
C.....
       N = NA+NB
       CALL UUD1(LAMBDA,N,ZETA,DIAG,OFFDIAG,ITEMP1,
     +           TRACE_AB,DIAGP)
C.....................................
C  UPDATE FIYF VECTOR.
C.....
       N = NA+NC+ND
       DO 130 I =1,N-1
              FIYF(N+1-I) = FIYF(N-I)
 130       CONTINUE
       FIYF(1) = -YF
       FIYF(NA+NC+1) = Y
C.....................................
C
C  UPDATE FIFI AND FIETA VECTORS.
C
C.....
       N=NA+NB
       DO 100 I =1,N
              DO 110 J =ND,2,-1
                     FIFI(I,J) = FIFI(I,J-1)
```

Figure 122. Continued.

```
                    FIETA(I,J) = FIETA(I,J-1)
 110              CONTINUE
 100        CONTINUE
          DO 120 I =1,N
             FIFI(I,1) = FI(I)
             FIETA(I,1) = ETA(I)
 120        CONTINUE
C.....................................
C  UPDATE FI AND ETA VECTORS.
C.....
          DO 60 I =1,N-1
             FI(N+1-I) = FI(N-I)
             ETA(N+1-I)= ETA(N-I)
 60        CONTINUE
          FI(1) = -W
          FI(NA+1) = U
          ETA(1) = -Y
          ETA(NA+1) = U
C.....................................
C                  ESTIMATE V.
C.....
          V = Y - W
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C                            AML        METHOD
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C.............................
C                  FIND E.
C.....
          N = NC+ND
          CALL EMVV(THETAD,FID,N,E)
          E = V - E
C.........................
C  UPDATE FIPSI MATRIX.
C.....
          DO 210 I =1,(NC+ND)
             DO 220 J =NC,2,-1
                FIPSI(I,J) = FIPSI(I,J-1)
 220            CONTINUE
 210        CONTINUE
          DO 230 I =1,(NC+ND)
             FIPSI(I,1) = -PSI(I)
 230        CONTINUE
C.....................................
C
C                FIND PSI.
C.....
          CALL EMVPROD(FIPSI,PC,N,NC,TEMP1)
          DO 162 I =1, N
             PSI(I) = TEMP1(I) + FID(I)
 162        CONTINUE
C.........................
C
```

Figure 122. Continued.

```
C  UPDATE THETAD AND DP. OR CALL CLS TO UPDATE THETAD AND DP.
C.....
       CALL UUD1(LAMBDA,(NC+ND),PSI,DIAGD,OFFDIAGD,TEMP1,
     +              TRACE_CD,DIAGPD)
C.....................................
C  FIND THE ERROR IN ESTIMATION.
C.....
       CALL EMVV(FID,THETAD,N,ERRORD)
       ERRORD = V - ERRORD
C.................................................
C  TEST FOR STABILITY OF AC(Z) POLYNOMIAL.
C.....
       IF((ST .EQ. 1) .OR. (ST .EQ. 3)) THEN
             AMY = 1.0
 199             DO 173 I =1,N
                    TSTABD(I) = THETAD(I) + TEMP1(I)*ERRORD*AMY
 173             CONTINUE
              DO 137 I =1,(NA+NB)
                    TSTAB(I) = THETA(I) + ITEMP1(I)*ERROR*AMY
 137             CONTINUE
C.....................................
C  FIND THE COEFFICIENTS FOR FILTER CA.
C.....
             DO 3 I =1,NA
                    PPA(I) = TSTAB(NA+1-I)
 3             CONTINUE
             DO 4 I =1,NC
                    PPC(I) = TSTABD(NC+1+ND-I)
 4             CONTINUE
             PPA(NA+1) = 1.0
             PPC(NC+1) = 1.0
             CALL MULT(NC,NA,PPC,PPA,NCA,PCA)
             DO 975 I=1,INT((NCA+1)/2)
                    IRTEMP = PCA(I)
                    PCA(I) = PCA(NCA+2-I)
                    PCA(NCA+2-I) = IRTEMP
 975             CONTINUE
             CALL NSTABL(PCA,NCA,IST)
             DO 5 I =1,NCA
                    PCA(I) = PCA(I+1)
 5             CONTINUE
             IF(IST .EQ. 0) GOTO 198
             IF(AMY .LE. 0.001) THEN
                    AMY = 0.0
                    GOTO 199
             END IF
             AMY = AMY/2.0
             GOTO 199
       END IF
C.....................................
C  UPDATE THE UNKNOWN VECTORS.
C.....
 198       DO 179 I =1,(ND+NC)
              THETAD(I) = THETAD(I) + TEMP1(I)*ERRORD*AMY
 179       CONTINUE
 33       DO 35 I =1,(NA+NB)
```

Figure 122. Continued.

```
                THETA(I) = THETA(I) + ITEMP1(I)*ERROR*AMY
 35         CONTINUE
C.........................
C  UPDATE PA AND PB.
C.....
          DO 160 I =1,NA
                PA(I) = THETA(I)
 160        CONTINUE
         DO 170 I =(NA+1),(NA+NB)
                PB(I-NA) = THETA(I)
 170        CONTINUE
C.....................................
C                  UPDATE FID.
C.....
         N=NC+ND
          DO 180 I =1,N-1
                FID(N+1-I) = FID(N-I)
 180        CONTINUE
         FID(1) = -V
         FID(ND+1) = E
C.....................................
C  UPDATE PC, PD AND THETACA.
C.....
         DO 190 I =1,ND
                PD(I) = THETAD(I)
 190        CONTINUE
         DO 150 I =1,ND
                THETACA(I+NCA) = PD(I)
 150        CONTINUE
C  ..............................................
         DO 200 I =ND+1, N
                PC(I-ND) = THETAD(I)
 200        CONTINUE
C...............................
C
C  WRITE THE RESULTS OUT.
C
C.....
         WRITE(2,*) T,Y,ACTY,W,V,ACTV,U,TRACE_AB,TRACE_CD
         WRITE(3,*) T,(THETA(I), I=1,NA)
         WRITE(4,*) T,(THETA(I), I=NA+1,(NA+NB))
         WRITE(6,*) T,(THETAD(I), I=1,ND)
         WRITE(5,*) T,(THETAD(I), I=ND+1,NC+ND)
         AMIRY=W
C..............
C..............
C..............
         RETURN
         END
C_____________________________________________________________________
C
C  NSTABL
C
C  TEST FOR STABILITY.
```

Figure 122. Continued.

```
C
C  REFERENCE V. KUCERA : DISCRITE LINEAR CONTROL, 1980, P. 153
C
C  INPUT :
C         A        : VECTOR OF ORDER N+1 CORRESPONDING TO THE
C                    POLYNOMIAL :
C                    A(Z) = A(1)*Z**N + A(2)*Z**(N-1) + ...
C                         + A(N+1)
C         N        : ORDER OF THE POLYNOMIAL.
C                    NOTE THAT THE N A LEAST SHOULD BE EQUAL TO 0.
C  OUTPUT:
C         IST      : INTEGER AT RETURN SHOWING THE STABILITY OF
C                    A(Z).
C                    IF IST = 0 THEN A(Z) HAS ALL ZEROS STRICLY
C                               INSIDE THE UNIT CIRCLE.
C                    IF IST = 1 THEN A(Z) HAS AT LEAST ONE ZERO
C                               ON OR OUTSIDE THE UNIT CIRCLE.
C
      SUBROUTINE        NSTABL(A,N,IST)
      IMPLICIT NONE
      REAL*8 A,W,AL
      INTEGER*4 N,IST,N1,I,K,NK1,J,NK
      DIMENSION A(70),W(70)
      IST = 1
      N1 = N + 1
      DO 1 I=1,N1
            W(I) = A(I)
            W(N1+I) = 0.0
1       CONTINUE
      K = 0
10       IF(K .EQ. N) GOTO 99
      NK1 = N-K+1
      DO 11 J =1,NK1
            W(N1+J) = W(NK1-J+1)
11       CONTINUE
      IF(W(N1+NK1) .EQ. 0.0) GOTO 98
      AL = W(NK1)/W(N1+NK1)
      IF(ABS(AL) .GE. 1.0) GOTO 98
      NK = N-K
      DO 12 J=1,NK
            W(J) = W(J) - AL*W(N1+J)
12       CONTINUE
      K=K+1
      GOTO 10
98       RETURN
99       IST = 0
      RETURN
      END
C
C  UUD1
C
C  LEAST-SQUARES ESTIMATION OF THE PARAMETERS, USING UD
C  FACTORIZATION.
C
```

Figure 122. Continued.

```
C  INPUT :
C    XLAMBDA   : FORGETTING FACTOR.
C    N         : I.E. N = NA + NB + NC
C    FII       : DATA VECTOR.
C    DIAG      : DIAGONAL TERMS OF P (THE COVARIANCE MATRIX).
C    OFFDIAG   : OFF DIAGONAL TERMS OF P.
C    PDIAG     : INITIAL VALUE OF THE DIAGONAL TERMS OF P.
C                NOTE THAT THIS VALUE WILL BE USED TO REINITIALIZE
C                THE COVARIANCE MATRIX IF THE TRACE OF P IS TO
C                LARGE OR TO SMALL.
C  OUTPUT :
C    DIAG      : UPDATED VERSION OF THE DIAGONAL TERMS OF P.
C    OFFDIAG   : UPDATED VERSION OF THE OFF DIAGONAL TERMS OF P.
C    XK        : NEW ESTIMATOR GAIN VECTOR.
C    TRACE     : TRACE OF THE COVARIANCE MATRIX.
C-------------------------------------------------------------------------
      SUBROUTINE UUD1(XLAMBDA,N,FII,DIAG,OFFDIAG,XK,
     +                TRACE,PDIAG)
      IMPLICIT NONE
      REAL*8        XLAMBDA,FII,DIAG,OFFDIAG,XK,TRACE,PDIAG
      REAL*8        FJ,VJ,ALPHAJ,AJLAST,PJ,W
      INTEGER*4        N,KF,KU,J,I,K,NUP
      DIMENSION DIAG(70),OFFDIAG(70),XK(70),FII(70)
C.......
C  CALCULATE GAIN AND COVARIANCE USING U-D METHOD.
C.....
      FJ = FII(1)
      VJ = DIAG(1)*FJ
      XK(1) = VJ
      ALPHAJ = 1.0 + VJ*FJ
      DIAG(1) = DIAG(1)/ALPHAJ/XLAMBDA
      IF ( N .GT. 1 ) THEN
            KF = 0
            KU = 0
            DO 20 J =2,N
                  FJ = FII(J)
                  DO 30 I =1,J-1
                        KF = KF+1
                        FJ = FJ + FII(I)*OFFDIAG(KF)
30                   CONTINUE
                  VJ = FJ*DIAG(J)
                  XK(J) = VJ
                  AJLAST = ALPHAJ
                  ALPHAJ = AJLAST + VJ*FJ
                  DIAG(J) = DIAG(J)*AJLAST/(ALPHAJ*XLAMBDA)
                  PJ = -FJ/AJLAST
                  DO 40 I =1,J-1
                        KU = KU + 1
                        W = OFFDIAG(KU) + XK(I)*PJ
                        XK(I) = XK(I) + OFFDIAG(KU)*VJ
                        OFFDIAG(KU) = W
40                   CONTINUE
20             CONTINUE
      ENDIF
      DO 50 I =1,N
            XK(I) = XK(I)/ALPHAJ
50       CONTINUE
```

Figure 122. Continued.

```
C.....
C  FIND THE TRACE OF COVARIANCE MATRIX.
C.....
       TRACE = 0.0
       DO 12 I = 1,N
              TRACE = TRACE + DIAG(I)
              DO 1212 J = I+1,N
                     K = J*(J-3)/2+1+I
                     TRACE = TRACE + DIAG(J)*OFFDIAG(K)*OFFDIAG(K)
 1212               CONTINUE
 12        CONTINUE
C.....
C  RETURN TO CALLING PROGRAM.
C.....
 1010        RETURN
       END
```

Figure 122. Continued.

APPENDIX G - THE LQR SUBROUTINE

In this appendix we present a FORTRAN 77 subroutine which can be used to generate a control action based on linear quadratic regulator.

```
C.....................................................
C  LQG.FOR
C
C  PURPOSE:
C         THIS SUBROUTINE MAKES A DECISION ON THE TYPE OF CONTROL
C         ACTION AND THEN CREATS THE CONTROL ACTION, I.E.
C
C                 UOPT = PROB                    IF IPROB = 0
C                 UOPT = -K*X(K)                 IF IPROB = 1
C
C         WHERE K IS FOUND BY MINIMIZING THE FOLLOWING COST
C         FUNCTION;
C                     INF.    T                 T
C                 J = SUM [ X  (K) Q X(K) + U  (K) W U(K)]
C                     K=0
C
C         FOR THE SYSTEM
C
C                 X(K+1) = G X(K) + H U(K)
C
C  INPUT:
C         G        :    NXN NONSINGULAR MATRIX.
C         H        :    NXR MATRIX.
C         Q        :    NXN POSITIVE DEFINITE OR POSITIVE
C                       SEMIDEFINITE HERMITIAN MATRIX (OR
C                       REAL SYMMETRIC MATRIX).
C         W        :    RXR POSITIVE DEFINITE HERMITIAN MATRIX
C                       (OR REAL SYMMETRIC MATRIX).
C         THETA    :    COEFFICIENTS OF THE MODEL. (I.E. A'S AND B'S)
C         U        :    CONTROL ACTION AT TIME K.
C         Y        :    MEASURED PLANT OUTPUT AT TIME K.
C         NA       :    NUMBER OF A COEFFICIENTS IN THE PLANT MODEL.
C         NB       :    NUMBER OF B COEFFICIENTS IN THE PLANT MODEL.
C         R        :    DIMENSION OF W MATRIX.
C         N        :    DIMENSION OF G MATRIX.
C         IPROB    :    FLAG FOR ADAPTIVE CONTROL GENERATION.
C                       0 => UOPT = PROB
C                       1 => UOPT = -K*X(K)
C         PROB     :    PROBING SIGNAL TO BE USED.  THIS IS USUALLY
C                       REQUIRED FOR SHORT PERIOD OF TIME AT THE
C                       STARTUP OF IDENTIFICATION.  IN THIS PERIOD
C                       NO OPTIMAL CONTROL WILL BE CALCULATED.
C                       THE SIGN OF PROB WILL BE DETERMINED AT
C                       RANDOM.
C  OUTPUT:
C         UOPT     :    THE NEXT CONTROL ACTION.
C************
C  NOTE:  THE PROB ALSO WILL BE USED FOR LIMITING THE CONTROL ACTION
C         WHEN IT IS CALCULATED FROM;
C                UOPT = -K*X(K)
C         THIS MEANS THAT,
C                -PROB <= UOPT <=PROB
C
```

Figure 123. The LQR subroutine..

```
      SUBROUTINE LQG(G,H,Q,W,THETA,U,Y,NA,NB,R,N,IPROB,PROB,UOPT,T)
      IMPLICIT NONE
      REAL*8          G,H,Q,W,THETA,U,Y,PROB,UOPT,JUNK,X,K,T
      INTEGER*4       NA,NB,N,R,IPROB,I,J
      DIMENSION       G(70,70),H(70,70),Q(70,70),W(70,70),THETA(70)
      DIMENSION       X(70),K(70,70)
C
C  CALL THE SSQOC SUBROUTINE TO CALCULATE THE GAIN VECTOR K.
      CALL SSQOC(K,Q,W,G,H,N,R,T)
C
C  CALL THE OBSERVER TO UPDATE THE STATES OF THE SYSTEM.
      CALL OBSERVER(THETA,U,Y,NA,NB,X,T)
C
C  IF IPROB = 0 THEN GENERATE THE PROBING SIGNAL.
      IF (IPROB .EQ. 0) THEN
            CALL GGNML(JUNK)
            IF (JUNK .LT. 0.D0) PROB=PROB*(-1.D0)
            UOPT = PROB
            GOTO 1000
      END IF
C
C  FIND THE UOPT.
      UOPT = 0.D0
      DO I=1,N
            UOPT = UOPT - K(1,I)*X(I)
      END DO
C
C  LIMIT THE UOPT SIGNAL BY USING THE MAGNITUDE OF PROB.
C
      IF (UOPT .GT. DABS(PROB)) UOPT = DABS(PROB)
      IF (UOPT .LT. (-1.D0 * DABS(PROB))) UOPT = -1.D0 * DABS(PROB)
C
C  RETURN TO THE CALLING PROGRAM.
C
 1000       CONTINUE
      RETURN
      END
C-----------------------------------------------
C  OBSERVER.FOR
C
C  PURPOSE:
C         TO ESTIMATE, OR RECREATE THE STATES OF THE SYSTEM.
C
C         X(K+1) = G X(K) + H U(K)
C  INPUT:
C         THETA    : COEFFICIENTS OF THE MODEL. (I.E. A' AND B'S)
C         U        : CONTROL ACTION AT TIME K.
C         Y        : PLANT OUTPUT.
C         NA       : NUMBER OF A COEFFICIENTS IN THE PLANT MODEL.
C         NB       : NUMBER OF B COEFFICIENTS IN THE PLANT MODEL.
C         X        : OBSERVERS STATES AT TIME K.
C
```

Figure 123. Continued.

```
C  OUTPUT:
C         X       :       OBSERVERS STATES AT TIME K+1.
C-----------------------------------------------
        SUBROUTINE OBSERVER(THETA,U,Y,NA,NB,X,T)
        IMPLICIT NONE
        REAL*8          THETA,U,Y,X,A,B,ERROR,OLD_Y,T
        INTEGER*4       I,J,NA,NB,N
        DIMENSION       THETA(70),X(70),A(70),B(70)
C
C  INITIALIZATION
C
        IF (NA .GT. NB) THEN
              N=NA
        ELSE
              N=NB
        END IF
        DO I=1,N
              A(I)=0.D0
              B(I)=0.D0
        END DO
        DO I=1,NA
              A(I)=THETA(I)
        END DO
        DO I=1,NB
              B(I)=THETA(I+NA)
        END DO
C
C  FIND THE OBSERVER ERROR
C
        OLD_Y = X(1)
        ERROR = Y - OLD_Y
C
C  ESTIMATE THE OBSERVERS STATES FOR TIME K+1.
C
        DO I=1,N-1
              X(I) = -A(I)*OLD_Y + X(I+1) + B(I)*U - A(I)*ERROR
        END DO
        X(N) = -A(N)*OLD_Y + B(N)*U - A(N)*ERROR
C
C  RETURN TO THE CALLING PROGRAM.
C
        RETURN
        END
C-------------------------------------------------------------
C  SSQOC.FOR
C
C  PURPOSE:
C         THIS FUNCTION COMPUTES THE OPTIMUM STEADY-STATE GAIN MATRIX
C         K WHICH MINIMIZES THE QUADRATIC COST FUNCTION GIVEN BY
C
C                     INF.    T                T
C                J = SUM [ X (K) Q X(K) + U (K) W U(K) ]
C                     K=0
C
C         FOR THE SYSTEM
```

Figure 123. Continued.

```
C                X(K+1) = G X(K) + H U(K)
C
C          WITH
C
C                U(K) = -K(K)X(K)
C
C          WHERE
C
C                X(K)     : STATE VECTOR (N-VECTOR)
C                U(K)     : CONTROL VECTOR (R-VECTOR)
C                G        : NXN NONSINGULAR MATRIX
C                H        : NXR MATRIX
C                Q        : NXN POSITIVE DEFINITE OR POSITIVE
C                           SEMIDEFINITE HERMITIAN MATRIX (OR
C                           REAL SYMMETRIC MATRIX)
C                W        : RXR POSITIVE DEFINITE HERMITIAN MATRIX
C                           (OR REAL SYMMETRIC MATRIX)
C-----------------------------------
      SUBROUTINE SSQOC(K,Q,W,G,H,N,R,T)
      IMPLICIT NONE
      REAL*8              P,Q,G,GT,H,HT,W,JUNK2,JUNK3,HTP,HTPH,WHTPH
      REAL*8              HTPG,GTP,GTPH,K,GTPG,DETER,T
      INTEGER*4        I,J,N,R
      DIMENSION        P(70,70),Q(70,70),G(70,70),GT(70,70),H(70,70)
      DIMENSION        HT(70,70),W(70,70),JUNK2(70,70)
      DIMENSION        JUNK3(70,70),HTP(70,70),HTPH(70,70)
      DIMENSION        WHTPH(70,70)
      DIMENSION        HTPG(70,70),GTP(70,70),GTPH(70,70),K(70,70)
      DIMENSION        GTPG(70,70)
C
C  FIND THE TRANSPOSES FIRST.
      CALL EMTRAN(G,N,N,GT)
      CALL EMTRAN(H,N,R,HT)
C
C  ITERATE THE RICCATI MATRIX.
      DO I=1,20
            CALL EMPROD(HT,P,R,N,N,HTP)
            CALL EMPROD(HTP,H,R,N,R,HTPH)
            CALL EMADD(HTPH,W,R,R,WHTPH)
            CALL FULPIVOT(WHTPH,R,WHTPH,DETER)
            CALL EMPROD(HTP,G,R,N,N,HTPG)
            CALL EMPROD(GT,P,N,N,N,GTP)
            CALL EMPROD(GTP,H,N,N,R,GTPH)
            CALL EMPROD(GTPH,WHTPH,N,R,R,JUNK3)
            CALL EMPROD(JUNK3,HTPG,N,R,N,JUNK3)
            CALL EMPROD(GTP,G,N,N,N,GTPG)
            CALL EMADD(Q,GTPG,N,N,JUNK2)
            CALL EMSUB(JUNK2,JUNK3,N,N,P)
       END DO
C  FIND THE GAIN VECTOR K.
C
      CALL EMPROD(WHTPH,HTPG,R,R,N,K)
      RETURN
      END
```

Figure 123. Continued.

APPENDIX H - THE POLE SUBROUTINE

In this appendix we present a FORTRAN 77 subroutine which can be used to generate a control action based on pole placement.

```
C........
C  [-------------------------------------------------------------]
C  GAIN
C
C  PURPOSE :
C         TO DETERMINE THE GAIN BY USING ACKERMANN'S FORMULA.
C
C         K = [0 0 ... 0 [][H | G*H | ... | (G**(N-1))*H]**(-1)*W(G)
C
C  WHERE
C         W(G) = (G**N) + A(1)*(G**(N-1)) + ... +
C                A(N-1)*G + A(NA)*I
C
C  WHERE
C
C         A()      : ARE THE DESIRED COEFFICIENTS OF THE CLOSED-
C                    LOOP CHARACTERESTIC EQUATION.
C         I        : IDENTITY MATRIX.
C  [-------------------------------------------------------------]
C........
        SUBROUTINE GAIN(OLD_GAIN,G,H,POLES,G_HK,N)
        IMPLICIT NONE
        REAL*8           G,G_HK,W,T,Q,H,C,OLD_GAIN,GAIN_VECTOR
        REAL*8           R,U,HH,P,POLES,X,DETER
        INTEGER*4        N,N1,I,J,N2,M,MYI,MYJ,K
        DIMENSION        G(70,70),G_HK(70,70),W(70,70),R(70,70)
        DIMENSION        T(70,70)
        DIMENSION        Q(70,70),H(70,70),C(70,70),GAIN_VECTOR(70,70)
        DIMENSION        U(70,70)
        DIMENSION        HH(70,70),P(70),POLES(70,70),OLD_GAIN(70,70)
C
C  INITIALIZATION.
        N1 = 1
        N2 = N+1
        DO 10 I=1,N
              DO 20 J=1,N
                    G_HK(I,J) = 0.0
                    W(I,J) = 0.0
                    R(I,J) = 0.0
                    T(I,J) = 0.0
                    Q(I,J) = 0.0
 20                   CONTINUE
              C(1,I) = 0.0
              GAIN_VECTOR(1,I) = 0.0
              R(I,I) = 1.0
              U(I,1) = 0.0
              HH(1,I) = 0.0
              P(I)    = 0.0
 10        CONTINUE
C
C  START THE PROCEDURE.
        DO 100 I=1,N
              J=I+1
              P(I) = -1.0*POLES(1,J)
 100       CONTINUE
        DO 110 I=1,N
```

Figure 124. The POLE subroutine.

```
              M = N - (I-1)
              CALL EMPROD(R,H,N,N,N1,U)
              X = P(M)
              DO 120 MYI=1,N
                     DO 130 MYJ=1,N
                            W(MYI,MYJ) = X*R(MYI,MYJ)
130                       CONTINUE
120                CONTINUE
              CALL EMADD(T,W,N,N,T)
              DO 140 K=1,N
                     Q(K,I) = U(K,1)
140                CONTINUE
              CALL EMPROD(G,R,N,N,N,W)
              DO 150 MYI=1,N
                     DO 160 MYJ=1,N
                            R(MYI,MYJ) = W(MYI,MYJ)
160                       CONTINUE
150                CONTINUE
110         CONTINUE
        DO 170 I=1,N
              DO 180 J=1,N
                     W(I,J) = R(I,J)
180                CONTINUE
170         CONTINUE
        CALL EMSUB(W,T,N,N,T)
        DETER = 0.0
        CALL FULPIVOT(Q,N,W,DETER)
        HH(1,N) = 1.0
        CALL EMPROD(W,T,N,N,N,Q)
        CALL EMPROD(HH,Q,1,N,N,GAIN_VECTOR)
        CALL EMPROD(H,GAIN_VECTOR,N,N1,N,W)
        CALL EMSUB(G,W,N,N,G_HK)
        DO 200 I=1,N
              OLD_GAIN(1,I) = GAIN_VECTOR(1,I)
200         CONTINUE
        RETURN
        END
```

Figure 124. Continued.

APPENDIX I - THE OTHER SUBROUTINES

In this appendix we present a collection of FORTRAN 77 subroutines which are called by the other subroutines in the previous appendices.

```
*****************************************************************
*       UUT
*
*                                                               T
*       TO FACTOR THE POSITIVE DEFINITE N X N MATRIX A INTO UU  WHERE
*       U IS UPPER TRIANGULAR MATRIX. CHOLESKY FACTORIZATION.
*
*       INPUT :
*              A        : THE MATRIX.
*              N        : DIMENSION OF A AND U MATRICES.
*
*       OUTPUT :                                                T
*              U        : THE FACTOR OF A MATRIX.  NOTE A = U * U.
*****************************************************************
      SUBROUTINE UUT(A,N,U)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION A(20,20),U(20,20),C(20,20)
C
C  STORE A MATRIX INTO C MATRIX.
C
      DO 10 I =1,N
            DO 20 J =1,N
                  C(I,J) = A(I,J)
 20            CONTINUE
 10       CONTINUE
C
C  START THE FACTORIZATION.
C
      DO 5 J =N,2,-1
            U(J,J) = SQRT(C(J,J))
            ALPHA = 1.0/U(J,J)
            DO 5 K =1,J-1
                  U(K,J) = ALPHA * C(K,J)
                  BETHA = U(K,J)
                  DO 5 I =1,K
 5                      C(I,K) = C(I,K) - BETHA*U(I,J)
      U(1,1) = SQRT(C(1,1))
C
C  NOW LET THE LOWER DIAGONAL TERMS OF U BE EQUAL TO ZERO.
C
      DO 30 I =2,N
            DO 40 J =1,(I-1)
                  U(I,J) = 0.0
 40            CONTINUE
 30       CONTINUE
C
C  RETURN TO CALLING PROGRAM.
C
      RETURN
      END
*****************************************************************
*
*       VAMB
*
*       MULTIPLIES A DIAGONAL N X N MATRIX WITH A N X N MATRIX.
```

Figure 125. The OTHER subroutines.

```
*       NOTE : THE DIAGONAL MATRIX IS STORED IN ONE DIMENSIONAL
*       ARRAY.
*************************************************************
      SUBROUTINE VAMB(A,B,N,C)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION A(20),B(20,20),C(20,20),D(20,20)
C  USE THE INTERMEDIATE ARRAY D IN CASE B IS ALSO SPECIFIED AS
C  THE RESULT, I.E., EMTRAN(A,B,N,B) IS THE CALLING STATEMENT.
      DO 10 I =1,N
            DO 20 J =1,N
                  D(I,J) = B(I,J)*A(I)
 20           CONTINUE
 10      CONTINUE
      DO 30 I =1,N
            DO 40 J =1,N
                  C(I,J) = D(I,J)
 40           CONTINUE
 30      CONTINUE
      RETURN
      END
*************************************************************
*       MAVB
*
*       MULTIPLIES N X N MATRIX BY A N X N DIAGONAL MATRIX.
*
*       NOTE : THE DIAGONAL MATRIX IS STORED IN A VECTOR.
*************************************************************
      SUBROUTINE MAVB(A,B,N,C)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION A(20,20),B(20),C(20,20),D(20,20)
C  USE THE INTERMEDIATE ARRAY D IN CASE A IS ALSO SPECIFIED AS
C  THE RESULT, I.E., EMTRAN(A,B,N,A) IS THE CALLING STATEMENT.
      DO 10 I =1,N
            DO 20 J=1,N
                  D(I,J) = A(I,J)*B(J)
 20           CONTINUE
 10      CONTINUE
      DO 30 I=1,N
            DO 40 J=1,N
                  C(I,J) = D(I,J)
 40           CONTINUE
 30      CONTINUE
      RETURN
      END
*************************************************************
*
*       INFNORM
*
*       TO FIND THE NORM INFINITY OF A MATRIX.
*
*       INPUT :
*             A       : THE MATRIX.
*             N       : DIMENSION OF THE A MATRIX. I.E. N X N.
```

Figure 125. Continued.

```
*       OUTPUT :
*               XNORM       : NORM INFINITY OF MATRIX A.
*************************************************************
      SUBROUTINE INFNORM(A,N,XNORM)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION A(20,20)
      XNORM = 0.0
      DO 10 I =1, N
            SUM = 0.0
            DO 20 J =1, N
                  SUM = SUM + DABS(A(I,J))
 20           CONTINUE
            IF (SUM .GT. XNORM) THEN
                XNORM = SUM
            END IF
 10       CONTINUE
      RETURN
      END
*************************************************
*       NORM2
*
*       TO FIND THE NORM 2 OF A VECTOR.
*
*       INPUT :
*               A       : THE VECTOR.
*               N       : NUMBER OF ELEMENTS IN A.
*
*       OUTPUT :
*               XNORM       : NORM 2 OF VECTOR A.
*************************************************
      SUBROUTINE NORM2(A,N,XNORM)
      IMPLICIT REAL*8 (A-H , O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION A(500)
      XNORM = 0.D0
      DO 10 I =1, N
            XNORM = XNORM + A(I)*A(I)
 10       CONTINUE
      XNORM = DSQRT(XNORM)
      RETURN
      END
C.......
C  [-----------------------------------------------------------]
C  SUBROUTINE TO CLEAR THE SCREEN AND LOCATE THE CURSOR AT
C  LOCATION 1,1.
C
C  [-----------------------------------------------------------]
C.......
      SUBROUTINE CLS
      IMPLICIT NONE
      BYTE              ESC/27/
      CHARACTER*1       B(3)/'[','2','J'/
      CHARACTER*1       C(5)/'[','1',';','1','H'/
      INTEGER           I
```

Figure 125. Continued.

```
      WRITE(*,1) ESC,(B(I),I=1,3)
      WRITE(*,1) ESC,(C(I),I=1,5)
 1    FORMAT('+',9A1)
      RETURN
      END
C.....
C  [-------------------------------------------------------------]
C        POS_TO_UD
C
C        PURPOSE:
C              TO FIND THE U-D FACTOR OF ANY POSITIVE
C              DEFINITE MATRIX.
C
C        INPUT :
C              POS      :       INPUT MATRIX.
C              N        :       DIMENSION OF POS MATRIX.
C
C        OUTPUT :
C              U        :       VECTOR STORED UPPER TRIANGULAR
C                               MATRIX.
C              D        :       VECTOR STORED DIAGONAL MATRIX.
C
C
C        NOTE THAT :
C                          T
C              POS = UDU
C
C  [-------------------------------------------------------------]
      SUBROUTINE POS_TO_UD(POS,N,U,D)
      IMPLICIT NONE
      REAL*8          POS,U,D,UD,ALPHA,BETA
      INTEGER*4       N,I,J,K,JJ,NP2,L,KK,KJ,JM1,IJ,IK
      DIMENSION       POS(500,500),U(500),D(500),UD(500)
C.....
      K=0
      DO 10 I=1,N
            DO 20 J=1,I
                  K=K+1
                  UD(K) = POS(J,I)
 20           CONTINUE
 10     CONTINUE
C.....
      JJ=N*(N+1)/2
      NP2=N+2
      DO 50 L=2,N
            J=NP2-L
            ALPHA=0.D0
            IF (UD(JJ) .GE. 0.D0) GOTO 100
            UD(JJ) = 0.D0
100               IF (UD(JJ) .GT. 0.D0) ALPHA = 1.D0/UD(JJ)
            JJ = JJ-J
            KK = 0
            KJ = JJ
            JM1 = J-1
               DO 40 K=1,JM1
```

Figure 125. Continued.

```
                        KJ = KJ+1
                        BETA = UD(KJ)
                        UD(KJ)=ALPHA*UD(KJ)
                        IJ = JJ
                        IK = KK
                        DO 30 I=1,K
                                IK = IK+1
                                IJ = IJ+1
                                UD(IK) = UD(IK) - BETA*UD(IJ)
 30                        CONTINUE
                        KK = KK+K
 40                  CONTINUE
 50          CONTINUE
             IF (UD(1) .LT. 0.D0) THEN
                 UD(1)=0.D0
             END IF
C.......
             KK=0
             DO 60 I=1,N
                 KK=I*(I-1)/2+I
                 D(I) = UD(KK)
 60          CONTINUE
             KK=0
             DO 70 J=2,N
                 DO 80 I=1,J-1
                     KK=KK+1
                     JJ=J*(J-1)/2+I
                     U(KK) = UD(JJ)
 80                  CONTINUE
 70          CONTINUE
             RETURN
             END
C........
C   [-------------------------------------------------------------]
C          UD_TO_POS
C
C          PURPOSE :
C                 TO FIND THE POSITIVE DEFINITE MATRIX FROM
C                 ITS U-D FACTORS.
C
C          INPUT :
C                 U        :       VECTOR STORED UPPER TRIANGULAR
C                                   MATRIX.
C                 D        :       VECTOR STORED DIAGONAL MATRIX.
C                 N        :       DIMENSION OF U AND D MATRICES.
C
C                 POS      :       OUTPUT POSITIVE DEFINITE MATRIX.
C
C          NOTE THAT :
C                              T
C                 POS = UDU
C
C   [-------------------------------------------------------------]
C........
             SUBROUTINE UD_TO_POS(POS,N,U,D)
             IMPLICIT NONE
             REAL*8          POS,U,D,UD,POUT,S,ALPHA
```

Figure 125. Continued.

```
      INTEGER*4        I,J,K,N,IJ,JJ,JJL,II,JM1,IK
      DIMENSION        POS(500,500),U(500),D(500),UD(500),POUT(500)
C.....
      K=0
      DO 10 I=1,N
            K=K+I
            UD(K)=D(I)
 10     CONTINUE
      K=0
      DO 20 J=2,N
            DO 30 I=1,J-1
                  K=K+1
                  IJ=J*(J-1)/2+I
                  UD(IJ)=U(K)
 30           CONTINUE
 20     CONTINUE
C.....
      POUT(1) = UD(1)
      JJ=1
      DO 40 J=2,N
            JJL=JJ
            JJ=JJ+J
            POUT(JJ)=UD(JJ)
            S=POUT(JJ)
            II=0
            JM1=J-1
            DO 50 I=1,JM1
                  II=II+I
                  ALPHA=S*UD(JJL+I)
                  IK=II
                  DO 60 K=I,JM1
                        POUT(IK)=POUT(IK)+ALPHA*UD(JJL+K)
                        IK=IK+K
 60                  CONTINUE
                  POUT(JJL+I)=ALPHA
 50           CONTINUE
 40     CONTINUE
C.....
C
C.....
      DO 200 J=1,N
            DO 300 I=1,J
                  IJ=J*(J-1)/2+I
                  POS(I,J)=POUT(IJ)
                  POS(J,I)=POUT(IJ)
 300          CONTINUE
 200    CONTINUE
C.....
C
C.....
      RETURN
      END
     &
C*****************************************************
C
```

Figure 125. Continued.

```
C  EMVV
C
C  MULTIPLIES TWO VECTORS TOGETHER.
C       T
C  C = A * B.
C
C  INPUT :
C          A        : THE FIRST VECTOR (N X 1).
C          B        : THE SECOND VECTOR(N X 1).
C          N        : DIMENSION OF A AND B VECTORS.
C
C  OUTPUT :
C          C  : RESULT OF THIS MULTIPLICATION.
C*******************************************************
       SUBROUTINE EMVV(A,B,N,C)
       IMPLICIT NONE
       REAL*8        A,B,C
       INTEGER*4        I,N
       DIMENSION        A(70), B(70)
       C = 0.0
       DO 10 I =1, N
             C = C + A(I)*B(I)
 10       CONTINUE
       RETURN
       END
C******************* MULT ********************
C   SUBROUTINE TO MULTIPLY TWO POLYNOMIALS
C   FORMAT ---  N  ORDER OF FIRST POLYNOMIAL
C    M  ORDER OF SECOND POLYNOMIAL
C    A  FIRST POLYNOMIAL IN ASCENDING POWERS
C    B  SECOND POLYNOMIAL IN ASCENDING POWERS
C    K  ORDER OF PRODUCT POLYNOMIAL
C    C  PRODUCT POLYNOMIAL IN ASCENDING POWERS
C
C    FIRST POLYNOMIAL = A(1) + A(2)X + ... + A(N)X**N
C    SECOND POLYNOMIAL= B(1) + B(2)X + ... + B(M)X**M
C    THEREFORE :
C           A = A(1),A(2),...,A(N)
C           B = B(1),B(2),...,B(M)
C           C = C(1),C(2),...,C(N+M)
       SUBROUTINE MULT(N,M,A,B,K,C)
       IMPLICIT REAL*8 (A-H,O-Z)
       IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70),B(70),C(70)
       K = M+N
       DO I=1,K+1
         C(I) = 0.
       END DO
       DO I=1,N+1
         DO J=1,M+1
           C(I+J-1) = C(I+J-1)+A(I)*B(J)
         END DO
       END DO
       END
C*******************************************************
```

Figure 125. Continued.

```
C  EMVVM
C
C  MULTIPLIES TWO VECTORS TO PRODUCE A MATRIX.
C
C       T
C  C(I,J) = A(I)*B(J)
C
C  INPUT :
C    A        : FIRST VECTOR.
C    B        : SECOND VECTOR.
C    N        : DIMENSION OF A,B AND C.
C
C  OUTPUT :
C    C  : THE RESULTING MATRIX.
C*****************************************************
      SUBROUTINE EMVVM(A,B,N,C)
      IMPLICIT NONE
      REAL*8 A,B,C
      INTEGER*4 I,J,N
      DIMENSION A(70),B(70),C(70,70)
      DO 10 I =1,N
            DO 20 J =1,N
                  C(I,J) = A(I)*B(J)
 20            CONTINUE
 10       CONTINUE
      RETURN
      END
C*****************************************************
C  EMVM
C
C  MULTIPLIES A VECTOR BY A MATRIX.
C
C  INPUT :
C    A        : INPUT VECTOR.
C    B        : INPUT MATRIX.
C    N        : COLUMN NUMBER OF B MATRIX.
C    M        : ROW NUMBER OF B MATRIX AND THE
C               DIMENSION OF A VECTOR.
C
C  OUTPUT :
C    C  : THE RESULT OF MULTIPLICATION.
C
C*****************************************************
      SUBROUTINE EMVM(A,B,N,M,C)
      IMPLICIT NONE
      REAL*8 A,B,C
      INTEGER*4 I,J,N,M
      DIMENSION A(70),B(70,70),C(70)
      DO 10 I =1,N
            C(I) = 0.0
            DO 20 J =1,M
                  C(I) = C(I) + A(J)*B(J,I)
 20            CONTINUE
 10       CONTINUE
```

Figure 125. Continued.

```
      RETURN
      END
C*****************************************************
C  SUBROUTINE IDENT GENERATES AN IDENTITY MATRIX
      SUBROUTINE IDENT(A,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
      DIMENSION A(70,70)
      DO I=1,N
          DO J=1,N
              A(I,J) = 0.0
              IF (I .EQ. J)  A(I,I) = 1.0
          END DO
      END DO
      RETURN
      END
C*********************************************
C  SUBROUTINE EMADD ADDS MATRICES:  C = A+B
      SUBROUTINE EMADD(A,B,N,M,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70,70),B(70,70),C(70,70)
      DO I=1,N
           DO J=1,M
              C(I,J) = A(I,J) + B(I,J)
          END DO
      END DO
      RETURN
      END
C*********************************************
C  SUBROUTINE EMSUB SUBTRACTS MATRICES:  C = A - B
      SUBROUTINE EMSUB(A,B,N,M,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70,70),B(70,70),C(70,70)
      DO I=1,N
           DO J=1,M
              C(I,J) = A(I,J) - B(I,J)
          END DO
      END DO
      RETURN
      END
C*****************************************************
C  SUBROUTINE VSUB SUBTRACTS VECTORS:  C = A - B
      SUBROUTINE VSUB(A,B,N,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70),B(70),C(70)
      DO I=1,N
          C(I) = A(I) - B(I)
      END DO
      RETURN
      END
C*********************************************
C  SUBROUTINE EMTRAN TRANSPOSES MATRICES
```

Figure 125. Continued.

```
      SUBROUTINE EMTRAN(A,N,M,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70,70),B(70,70),C(70,70)
C  USE THE INTERMEDIATE ARRAY B IN CASE A IS ALSO SPECIFIED AS
C  THE RESULT, I.E., EMTRAN(A,N,M,A) IS THE CALLING STATEMENT.
      DO I=1,N
          DO J=1,M
            B(J,I) = A(I,J)
          END DO
      END DO
      DO I=1,M
          DO J=1,N
            C(I,J) = B(I,J)
          END DO
      END DO
      RETURN
      END
C*********************************************
C  SUBROUTINE EMPROD MULTIPLIES MATRICES
C  USING CLASSICAL NOTIONS OF INNER PRODUCTS
      SUBROUTINE EMPROD(A,B,N,M,L,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70,70),B(70,70),C(70,70),D(70,70)
C  USE THE ARRAY D IN CASE THE CALLING STATEMENT IS
C  CALL EMPROD(A,B,N,M,L,A).
      DO I=1,N
          DO J=1,L
             D(I,J)=0.
             DO K=1,M
                 D(I,J) = D(I,J) + A(I,K)*B(K,J)
             END DO
          END DO
      END DO
      DO I=1,N
          DO J=1,L
              C(I,J) = D(I,J)
          END DO
      END DO
      RETURN
      END
C*********************************************
C  SUBROUTINE EMVPROD MULTIPLIES A MATRIX AND
C  A VECTOR.
      SUBROUTINE EMVPROD(A,B,N,M,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70,70),B(70),C(70),D(70)
C  USE THE ARRAY D IN CASE THE CALLING STATEMENT IS
C  CALL EMVPROD(A,B,N,M,B).
      DO I=1,N
          D(I)=0.
          DO J=1,M
              D(I) = D(I) + A(I,J)*B(J)
          END DO
```

Figure 125. Continued.

```
        END DO
        DO I=1,N
            C(I) = D(I)
        END DO
        RETURN
        END
C*****************************************************************
C  FULPIVOT INVERTS MATRICES USING A GAUSS-JORDAN PROCEDURE
C  WITH FULL-PIVOTING.
C
C  AA = MATRIX TO BE INVERTED
C  B  = INVERSE OF AA
C  N  = SIZE OF MATRICES AA AND B
        SUBROUTINE FULPIVOT(AA,N,B,DETER)
        IMPLICIT REAL*8 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I-N)
        DIMENSION AA(70,70),A(70,70),B(70,70),JK(70)
C  IF AA IS ONLY 1 X 1 (A SCALAR), INVERT AND RETURN.
        IF (N .EQ. 1) THEN
            IF (AA(1,1) .NE. 0.0) THEN
                B(1,1) = 1.0 / AA(1,1)
            ELSE
                DETER  = 0.0
                B(1,1) = 0.0
            END IF
            RETURN
        END IF
C  COPY AA INTO A WORKING ARRAY.  AT THE SAME TIME, CONSTRUCT AN
C  IDENTITY MATRIX, B, INTO WHICH ROW OPERATIONS ON A WILL BE
C  COPIED, THUS FORMING THE INVERSE.  ALSO CLEAR THE JK ARRAY
C  (WHICH IS USED TO MARK THOSE ROWS THAT HAVE BEEN PROCESSED)
C  TO INDICATE THAT NO ROWS HAVE BEEN PROCESSED.  FINALLY, INITIALIZE
C  THE DETERMINANT VALUE.
        DO I=1,N
            DO J=1,N
                A(I,J) = AA(I,J)
                B(I,J) = 0.0
            END DO
        END DO
        DO I=1,N
            B(I,I) = 1.0
            JK(I)  = 0
        END DO
        DETER = 1.0
C  EXECUTE THE REDUCTION LOOP N TIMES
C
        DO K=1,N
C  FIND THE PIVOT ELEMENT.  IT IS THE LARGEST ELEMENT IN THE ARRAY
C  THAT IS IN THE ROWS AND COLUMNS THAT HAVE NOT BEEN PROCESSED.
C  THE CHECK FOR NON-PROCESSED ROWS IS FOR JK = 0.
C
            COMP = 0.0
            DO J=1,N
                IF (JK(J) .EQ. 0) THEN
                    DO I=1,N
```

Figure 125. Continued.

```
                        IF (JK(I) .EQ. 0) THEN
                            IF (ABS( A(I,J) ) .GT. COMP) THEN
                                COMP = ABS( A(I,J) )
                                II = I
                                JJ = J
                            END IF
                        END IF
                    END DO
                END IF
            END DO
            JK(JJ) = 1
C   UPDATE THE DETERMINANT.
        DETER = DETER * COMP
C   INTERCHANGE ROWS IF NECESSARY.  MAKE THE SAME INTERCHANGES ON A
C   AND B.
            IF (II .NE. JJ) THEN
                DO J=1,N
                    TEMP     = A(II,J)
                    A(II,J) = A(JJ,J)
                    A(JJ,J) = TEMP
                    TEMP     = B(II,J)
                          B(II,J) = B(JJ,J)
                    B(JJ,J) = TEMP
                END DO
            END IF
C   NORMALIZE THE PIVOT ROW.
            IF ( A(JJ,JJ) .EQ. 0.0) THEN
                DETER = 0.0
               DO I=1,N
                   DO J=1,N
                       B(I,J) = 0.0
                   END DO
               END DO
                RETURN
            END IF
            TEMP = 1.0 / A(JJ,JJ)
            DO J=1,N
                A(JJ,J) = TEMP * A(JJ,J)
                B(JJ,J) = TEMP * B(JJ,J)
            END DO
            A(JJ,JJ) = 1.0
C   PERFORM THE ROW OPERATIONS.  RECORD THEM IN BOTH A AND B.
            DO I=1,N
                IF (I .NE. JJ) THEN
                    TEMP = A(I,JJ)
                    DO J=1,N
                        A(I,J) = A(I,J) - TEMP*A(JJ,J)
                        B(I,J) = B(I,J) - TEMP*B(JJ,J)
                    END DO
                END IF
            END DO
        END DO
C   CHECK THE DETERMINANT.  IF IT IS ZERO,
C   RETURN A VALUE OF ZERO FOR THE INVERSE.
C
```

Figure 125. Continued.

```
      IF (DETER .EQ. 0.0) THEN
          DO I=1,N
              DO J=1,N
                  B(I,J) = 0.0
              END DO
          END DO
      END IF
      RETURN
      END
C*********************************************
C
C MATRIX MULTIPLICATION VIA CANNON ALGORITHM
C
C*********************************************
      SUBROUTINE EMPRODB(A,B,N,M,L,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I-N)
       DIMENSION A(70,70),B(70,70),C(70,70)
      DO I=1,N
          DO J=1,L
             C(I,J) = 0.0
          END DO
      END DO
      DO I=1,N
          DO J=1,L
             DO K=1,M
                 IX     = 1 + MOD((I+J-K+M-1),M)
                 C(I,J) = C(I,J) + A(I,IX)*B(IX,J)
             END DO
          END DO
      END DO
      RETURN
      END
```

Figure 125. Continued.