ON-LINE SYSTEM IDENTIFICATION IN REAL TIME

USING A MINICOMPUTER

ON-LINE SYSTEM IDENTIFICATION IN REAL TIME USING A MINICOMPUTER

By

M. Y. TANG, B. Eng.

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Engineering

McMaster University

April 1975

MASTER OF ENGINEERING (1975)
(Electrical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario.

TITLE: On-Line System Identification in Real Time Using Minicomputer

AUTHOR: M. Y. Tang, B. Eng. (McMaster University)

SUPERVISOR: N. K. Sinha, B.Sc. Eng. (Banaras)
Ph.D. (University of Manchester)

NUMBER OF PAGES: vii: 109.

SCOPE AND CONTENTS: By making use of measurements of the input and output of an unknown system, the characterizing parameters are found by using matrix pseudoinverse method. For noise-corrupted measurements, least squares estimation of the parameters are obtained.

Stochastic approximation is employed to improve the estimation.

Both methods are tested on-line in real time using the PDP-11/45 minicomputer while the system is simulated on the TR-20 analog computer.

ACKNOWLEDGEMENTS

I would like to thank Dr. N.K. Sinha for his encouragement and assistance during the course of this work and in the preparation of this thesis.

I would also like to thank my colleagues especially Mr. A. Sen for many valuable discussions. A special gratitude is due to Mr. H. de Bruin who designed the interface panel used in the experiments.

Awards from National Research Council in the form of a bursary and a scholarship are much appreciated. Financial assistance from McMaster University in the form of a scholarship and teaching assistantship is also gratefully acknowledged.

Finally, I would like to thank Miss W. Lee for typing this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

The terms system identification and process parameter estimation have been used interchangeably by many people. It is the process of obtaining the characteristic parameters of a system by artificially exciting the system with some specially selected signals. Depending on the situation and the need, a great variety of techniques have been developed.

The rapid development in the area of system identification can be attributed to many roots. Prominent among them is that there is a real need to plan and devise better process control. In order to do so, there is the inevitable prerequisite of a good knowledge of the process at hand. Due to the different conditions under which the process will operate, an accurate calculation beforehand is very difficult or sometimes impossible. To complicate the problem further, in the real world environment, we always have random disturbances generally called noise in the system. This naturally makes the characterization of the system even more difficult.

The techniques of system identification lend themselves to a host of applications. The study of the dynamics of many high performance systems such as space vehicles is another example. However, aside from the fact that there exists a need to utilize these techniques in various situations, the availability of the necessary tools is also an important factor. On the theorectical side, we see tremendous advances in control theory. Also of vital importance is the phenomenal development of computer hardware technology and software. Nowadays, computers are fast enough to accommodate complex calculations in relatively complicated algorithms in much less time than a decade ago. Yet, they are cheap enough to be considered for implementation in many industrial processes. Parallel developments are also found in pheripheral devices for data acquisition, interfacing and data transmission. The dramatic reduction in size and weight further facilitate their usage in many more situations.

The present study represents an attempt tending to the goal of utilising the modern computer technology in system identification in a noisy environment.

In the remaining parts of this thesis, the various aspects of implementing two on-line system identification algorithms based on matrix pseudoinverse will be presented. Chapter two is primarily concerned with the definition and classification of different methods of system identification. Ways of constructing a convenient model are also discussed. Chapter three reviews the basics of matrix pseudoinverse together with its application to develop an on-line identification algorithm. Before leaving for the discussion of a second algorithm, we will study methods for the determination of the order of the model of the system. Chapter four begins with an investigation of the effects of measurement noise on estimation and then proposes a method using stochastic approximation to improve the estimation. All the practical aspects of the experiments are presented in chapter five. Results of the experiments with tables and plots are presented in the next chapter. Chapter seven is the final conclusion of the whole work with an appendix giving the program listing.

CHAPTER TWO

SYSTEM IDENTIFICATION

2.1 Definition of System Identification

While there is not yet a unique definition for system identification agreed upon by everybody, we shall adopt the one suggested by Zadeh [1]:

> "Identification is the determination, on the basis of input and output, of a system within a specified class of systems, to which the system under test is equivalent"

Before we proceed, clarification of some of the terms used in the above definition is in order. Systems here refer to the mathematical models of physical systems. It is the first step in the formulation of an identification problem and indeed of any analysis. Modelling of a system generally refers to its representation by a pulse transfer function or by the state variables. Identification, therefore, is the determination of the coefficients in either representation. The meaning of equivalence can be better understood by referring to another concept called identifiability. Astrom and Bohlin [2] suggested that a system is identifiable if the estimates of its parameters are consistent. A necessary condition is that the information matrix be positive definite. When two models are identifiable and result in the same consistent estimates, they are said to be equivalent.

2.2 Classification of Identification Methods

The problem of parameter identification can be viewed as an optimization problem. The solution to this problem consists of finding the extremum of a certain specified loss function as a function of the parameters to be identified.

Computationally, identification methods can be divided into two categories, namely on-line methods and off-line methods.

Off-line method is a one-shot technique where the loss function is an explicit mathematical relation between the measurements and the estimates of interest. It is also known as the open loop method. The solution is available after a fixed finite number of elementary operations. But processing of the data can only start after all measurements are completed. In general, algorithms belonging to this category require considerable computer memory for the processing and storage of the data. Many methods making use of auto- and cross-correlation [3] belong to this category.

On-line methods, on the other hand, are closed loop methods. An iterative scheme whereby the estimation of parameters are being continuously updated as new measurements are made. These intermediate estimates are approximate solutions only. The final solution is approached asymptotically and is therefore, in principle, available only after an infinite number of elementary operations. The objective function in this case is an implicit function of the parameters and some form of model-adjustment strategy is employed to search for the extremum of the objective function. For example, the matrix pseudoinverse method being investigated in the present study adopts the squares of the residual errors of the system dynamic equation as the objective function. This function is being minimized through an iterative scheme .

The stochastic approximation method can also be formulated in an iterative form for on-line application. This method is being used to identify the noise model in the present study. Both methods will be described in greater detail later.

The on-line methods have the obvious advantage over the off-line methods in that intermediate results are available for use during the identification process if desired. This type of estimation will also be able to respond to a change in the system dynamics. When applied in real time, however, they pose more problems in their implementation. A useful on-line method must have the important property that it be computationally efficient. The reason is obvious because we have to complete all necessary calculations in updating the estimates before the next set of new measurements can be made. An inefficient algorithm will impose severe constraints on the choice of sampling frequencies. Moreover, in an industrial setting, identification is only part of the control loop. The computer is likely to be used for other purposes as well. Thus, the ease of implementation, and hence the feasibility for realistic application, depends heavily on the computational simplicity.

2.3 Formulation of an Identification Problem

The formulation of an identification problem begins with the modelling of the system. Preferably, the model chosen should have the following properties:-

(a) It is based on the input-output measurements and does not depend on other measurements that might be difficult or impossible to be made directly;

(b) It must assume a simple form with the smallest possible number of parameters to be identified and can readily be used for controller design;

(c) It should be able to accommodate the stochastic behaviour of the system due to random disturbances.

Several basic assumptions are also made in the modelling which would greatly reduce the amount of work without severely limiting the usefulness. We shall assume that the system is linear or can be adequately approximated by a linearised model. Further the system is assumed to be time-invariant. If the system is only slowly time-varying, the changes can be reflected on the estimates obtained by on-line algorithms. Finally, the physical system is assumed to be of finite order.

There are two convenient types of models available, namely the state-variable model and the pulse transfer function model.

The state-variable model of a linear, finite dimensional and time-invariant discrete time system is given by

$$\underset{\sim}{x}(k+1) = A\underset{\sim}{x}(k) + Bu(k) \tag{2.1}$$

$$y(k) = C\underset{\sim}{x}(k) \tag{2.2}$$

where $k \geq o$ is an integer,

$\underset{\sim}{x}$ is the state vector of dimension m,

A is a mxm matrix,

B is a mx1 vector,

C is a 1xm vector,

u is the input,

and y is the output.

The parameters are the elements in matrices A, B and C. Gupta [4] has investigated this type of model in detail applied to system identification.

The transfer function counterpart of equations (2.1) and (2.2) are as follows:-

$$H(z^{-1}) = \frac{C(z^{-1})}{R(z^{-1})}$$

$$= \frac{a_0 + a_1 z^{-1} + \ldots + a_m z^{-m}}{1 + b_1 z^{-1} + \ldots + b_n z^{-n}} \tag{2.3}$$

where $C(z^{-1})$ is a polynomial for the output of order m

$R(z^{-1})$ is a polynomial for the input of order n

The parameter vector for identification can be defined as

$$\underset{\sim}{\theta}^T = [a_0 a_1 \ldots a_m b_1 b_2 \ldots b_n] \tag{2.4}$$

where superscript T denotes matrix transpose.

The transfer function model is more suitable for the equation-error approach since it is a direct relation between the input and output. For this reason, it is chosen for the present study.

In discrete time, equation (2.3) can be written as

$$c_i = \sum_{j=0}^{m} a_j \, r_{i-j} - \sum_{j=1}^{n} b_j \, c_{i-j} \tag{2.5}$$

where $c_i = c(iT)$, output of system at $t = iT$

$r_i = r(iT)$, input of system at $t = iT$

T = sampling period

i = integer

In matrix notation, letting i range from 1 to same integer k, we can again rewrite (2.5) as

$$A_k' \, \underset{\sim}{\ell} = \underset{\sim}{c}_k \tag{2.6}$$

where

$$A_k' = \begin{bmatrix} r_0 & r_{-1} & \cdots & r_{1-m} & -c_{-1} & -c_{-2} & \cdots & -c_{1-n} \\ r_1 & r_0 & \cdots & r_{2-m} & -c_0 & -c_{-1} & \cdots & -c_{2-n} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ r_{k-1} & r_{k-2} & \cdots & r_{k-m} & -c_{k-2} & -c_{k-3} & \cdots & -c_{k-n} \end{bmatrix} \tag{2.7}$$

and

$$\underset{\sim}{c}_k^T = [c_1 \; c_2 \; \cdots \; c_k] \tag{2.8}$$

Equation (2.6) can be solved analytically if m+n+1 pairs of r_i and c_i are known exactly, assuming known initial conditions, or twice as many pairs of measurements if initial conditions are unknown.

However, the measurements are usually contaminated with noise and we have the input-output measurements as follows:

$$u_i = r_i + w_i \tag{2.9}$$

$$y_i = c_i + v_i \tag{2.10}$$

where $\{w_i\}$ and $\{v_i\}$ are noise sequences for input and output respectively.

Therefore, equations (2.6), (2.7) and (2.8) will be replaced by

$$A_k \hat{\underset{\sim}{\phi}} = \underset{\sim}{y}_k \tag{2.11}$$

where

$$A_k = \begin{bmatrix} u_o & u_{-1} & \cdots & u_{1-m} & -y_{-1} & \cdots & y_{1-n} \\ u_1 & u_0 & \cdots & u_{2-m} & -y_0 & \cdots & y_{2-n} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ u_{k-1} & u_{k-2} & \cdots & u_{k-m} & -y_{k-2} & \cdots & y_{k-n} \end{bmatrix} \tag{2.12}$$

$$\underset{\sim}{y}_k^T = [y_1 \; y_2 \; \cdots \; y_k] \tag{2.13}$$

and the parameter vector

$$\hat{\underset{\sim}{\phi}}^T = [\hat{a}_0 \hat{a}_1 \; \cdots \; \hat{a}_m \hat{b}_1 \hat{b}_2 \; \cdots \; \hat{b}_n] \tag{2.14}$$

In subsequent chapters, an algorithm will be developed to solve $\hat{\underset{\sim}{\theta}}$ in an iterative manner suitable for on-line application based on the noise-corrupted input-output measurements. The error of estimation $\underset{\sim}{E}_R$ is defined as the difference between the estimated vector $\hat{\underset{\sim}{\theta}}$ and the true parameter vector $\underset{\sim}{\theta}$,

$$\underset{\sim}{E}_R = \underset{\sim}{\theta} - \hat{\underset{\sim}{\theta}} \tag{2.15}$$

Very often, it is convenient to express the error as a scalar quantity which can be normalised for easy comparison. Thus, we define the normalised error of the estimated parameter vector as

$$e_R = \frac{||\underset{\sim}{\theta} - \hat{\underset{\sim}{\theta}}||^2}{||\underset{\sim}{\theta}||^2} \tag{2.16}$$

where $||\ \ ||$ denotes the norm.

Pictorially, the error due to an incorrect estimation of the parameter vector can be depicted in figure 2.1.

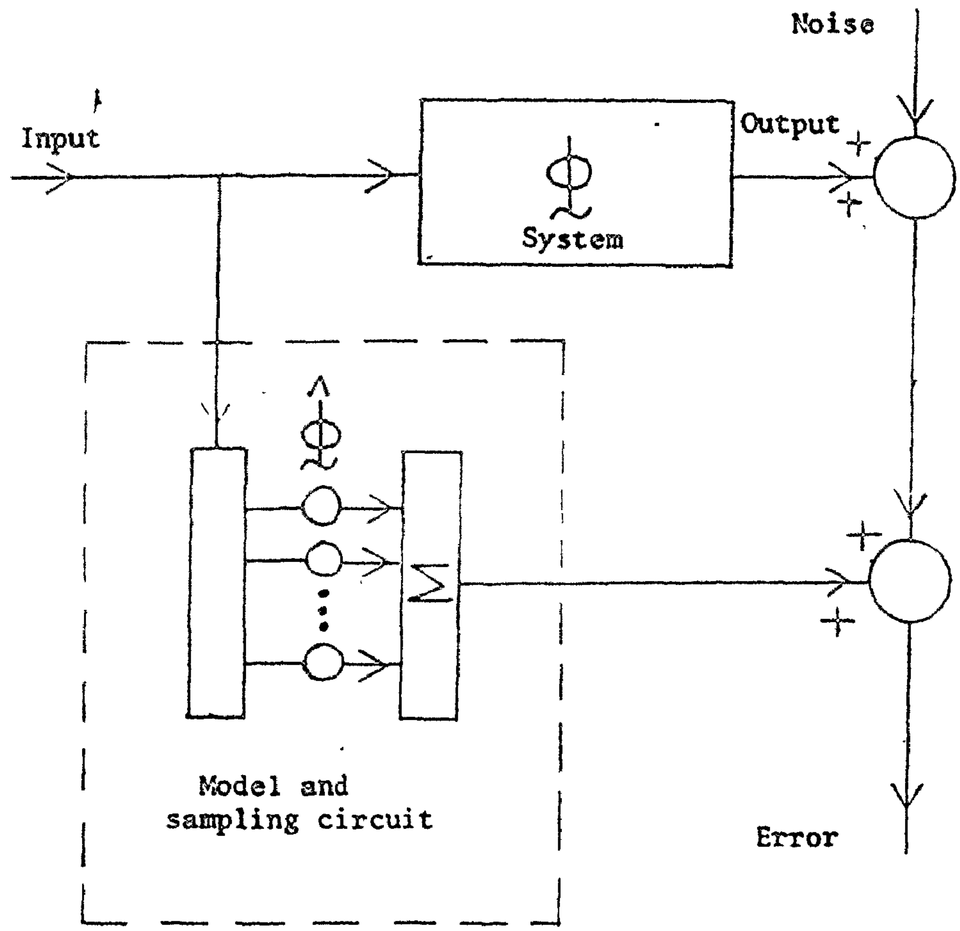


Figure 2.1 Error in Estimation

CHAPTER THREE

PSEUDOINVERSE APPLIED TO SYSTEM IDENTIFICATION

3.1 Definition of Pseudoinverse

In solving the system of linear equations

$$A\underset{\sim}{x} = \underset{\sim}{y}$$

we have

$$\underset{\sim}{x} = A^{-1}\underset{\sim}{y}$$

where A^{-1} is called the inverse matrix of matrix A such that A must be a square non-singular matrix with the property that

$$AA^{-1} = A^{-1}A = I$$

Penrose [5] generalized the idea of matrix inverse to include cases where A is rectangular and gave it the name generalized inverse or pseudo-inverse of a matrix. Greville [6] and others also have investigated its properties and applications. One way of defining [6] the pseudoinverse is as follows:

Let A be a matrix of dimension mxn with rank equal to r. It can be factorized into two matrices B and C such that

$$A = BC \qquad (3.1)$$

where B is a mxr matrix of rank r

C is a rxn matrix of rank r

$r \leq m$, n integers

The factorization can be found by first selecting B such that its columns are the linearly independent columns of A. Since A is of rank r, the dimension of B must be mxr. Then C is chosen such that it will satisfy equation (3.1).

The pseudoinverse of A is then given by

$$A^{+} = C^{T}[CC^{T}]^{-1}\,[B^{T}B]^{-1}\,B^{T} \quad \text{if } A \neq 0$$
$$= A^{T} \quad \text{if } A = 0 \tag{3.2}$$

It can be proved [7] that there always exists a unique pseudoinverse A^{+} as defined above for any matrix A.

In the special case when n=r, (3.1) reduces to

$$A = BI$$

and (3.2) is simplified to

$$A_L^{+} = [B^{T}B]^{-1}\,B^{T}$$
$$= [A^{T}A]^{-1}\,A^{T} \tag{3.3}$$

where A_L^{T} is called the left pseudoinverse of A with the rows in the row space of A^{T} such that $A_L^{+}\,A = I$

Example:

Let $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 2 & 1 \end{bmatrix}$

Here $m=3$, $n=r=2$

Using (3.3) $A_L^+ = [A^TA]^{-1} A^T$

$$= \frac{1}{21} \begin{bmatrix} 5 & -4 & 8 \\ -2 & 10 & 1 \end{bmatrix}$$

$$A_L^+ A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Similarly, when m=r, (3.1) reduces to

$$A = IC$$

and (3.2) becomes

$$A_R^+ = C^T [CC^T]^{-1}$$

$$= A^T [AA^T]^{-1} \qquad (3.4)$$

where A_R^+ is called the right pseudoinverse of A with the columns in the column space of A^T such that $AA_R^T = I$

Example:

Let $A = [1 \quad 2]$

Here $m=r=1$, $n=2$

Using (3.4) $A_R^T = [A^TA]^{-1} A^T$

$$= \frac{1}{5} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$AA_R^+ = 1 = I$$

Comparing equations (3.2), (3.3) and (3.4), we can deduce that

$$A^+ = A_L^+ A_R^+ \qquad (3.5)$$

3.2 Properties of the Pseudoinverse

Before going on to consider the use of pseudoinverse for system identification, we have to look more closely into its properties. Many of these properties will be used in the derivation of the identification algorithm

For every real mxn matrix A, there exists a unique real pseudo inverse A^+ as defined earlier which will satisfy the following identities

$$
\begin{aligned}
A^+ AA^+ &= A^+ \\
A A^+ A &= A \\
[AA^+]^T &= AA^+ \\
[A^+A]^T &= A^+A
\end{aligned}
\qquad (3.6)
$$

where the superscript T denotes transpose of a matrix.

In fact, the above identities are used by Penrose [4] to define pseudoinverse in his original paper. They form a set of necessary and sufficient conditions to prove the existence and uniqueness of the pseudo-inverse.

Other properties are summarized below [8]:-

1. $A^{++} = A$
2. $A^{T+} = A^{+T}$
3. $A^+ = A^{-1}$ if A is square and nonsingular
4. $[\lambda A]^+ = \lambda^+ A^+$, λ scalar

$$\text{and } \lambda^{+} = \frac{1}{\lambda} \text{ for } \lambda \neq 0$$

$$\lambda^{+} = 0 \text{ for } \lambda = 0$$

5. $[A^T A]^{+} = A^{+} A^{T+}$

$[AA^T]^{+} = A^{+T} A^{+}$

in general $[AB]^{+} \neq B^{+} A^{+}$

6. The ranks of A, $A^T A$, A, $A^{+}A$ are all equal to the trace of $A^{+}A$

3.3 Application to the Solution of A System of Linear Equations

In this section, we shall prove an important theorem which serves to illustrate the crucial value of matrix pseudoinverse in the solution of a system of linear equations. This in turn plays an important role in the development of the algorithm for system identification.

Recall the definition of the pseudoinverse given in section 3.1 that

$$\text{if} \quad A = BC \tag{3.1}$$

$$\text{then } A^{+} = C^T [CC^T]^{-1} [B^T B]^{-1} B^T \tag{3.2}$$

Theorem [9]:

With the pseudoinverse A^{+} of matrix A defined as in equations (3.1) and (3.2), the solution of the system of linear equations

$$\chi = A\underset{\sim}{x} \tag{3.7}$$

which will minimize

(a) the sum of squares of the residuals $\underset{\sim}{e}^T \underset{\sim}{e}$

$$\text{where } \underset{\sim}{e} = \chi - A\underset{\sim}{x} \tag{3.8}$$

(b) the sum of squares of the unknown $\chi^T \chi$ is given by

$$\chi = A^{+}\chi \tag{3.9}$$

Proof: Let $S = (\chi - A\chi)^T (\chi - A\chi)$

$$= \chi^T\chi - \chi^T A^T \chi - \chi^T A \chi + \chi^T A^T A \chi$$

minimizing S

$$\frac{\partial S}{\partial x_j} = 0 \quad , \quad j = 1,2,3 \ldots n$$

we have $A^T A\chi = A^T \chi$ (3.10)

Now, substitute equation (3.1) into (3.10), we have

$$C^T [B^T B] C\chi = C^T B^T \chi \tag{3.11}$$

Multiply both sides by $[B^T B]^{-1} [CC^T]^{-1}$ and rearranging, we get

$$x = C^T [CC^T]^{-1} [B^T B]^{-1} B^T \chi$$

$$= A^{+} y \tag{3.12}$$

by applying equation (3.2). Hence the assertions (a) and (b) follow.

In other words, the solution given by $\chi = A^{+} \chi$ has the important consequence that it is also the optimal solution in the sense that the square of the residual error is minimized. The identification algorithm developed according to this same principle will likewise yield optimal estimates in the same sense.

3.4 Recursive Identification Algorithm

Recall that we have formulated the parameter estimation problem in section 2.3 as

$$A_k \hat{\underset{\sim}{\phi}}_k = \underset{\sim}{y}_k \tag{2.11}$$

Applying what we have just developed in the preceding section, we have at the kth iteration

$$\hat{\underset{\sim}{\phi}}_k = A_k^+ \underset{\sim}{y}_k \tag{3.13}$$

with $\hat{\underset{\sim}{\phi}}_k$ as the optimal estimation of $\underset{\sim}{\phi}_k$.

If $\hat{\underset{\sim}{\phi}}_k$ has p=m+n+1 elements as defined in section 2.3, we may form the following special cases:

(a) For $k \leq p$,

$$\hat{\underset{\sim}{\phi}}_k = A_k^T [A_k A_k^T]^{-1} \underset{\sim}{y}_k \tag{3.14}$$

where $A_k^T [A_k A_k^T]^{-1}$ is the right pseudoinverse of A_k. It gives the so called minimum norm solution of $\hat{\underset{\sim}{\phi}}_k$.

(b) For $k > p$

$$\hat{\underset{\sim}{\phi}}_k = [A_k^T A_k]^{-1} A_k^T \underset{\sim}{y}_k \tag{3.15}$$

where $[A_k^T A_k]^{-1} A_k^T$ is the left pseudoinverse of A_k. It gives the least squares solution of $\hat{\underset{\sim}{\phi}}_k$.

In both cases, A_k is assumed to have full rank. This condition can be guaranteed if one of the following conditions [10] is imposed on the input sequence r(iT):-

1. r(iT) is a sequence composed of n discrete Fourier components and all natural modes are present in the output sequence C_i.

2. $r_i = 0$ for $k < n$

   $= 1$ for $k \geq n$

3. r(iT) is a random signal.

More will be said about the input signal for the experiments in chapter five.

The transformation of equation (3.13) into an iterative formula has been considered by Wells [11] and Sinha and Pille [12]. The information matrix A_{k+1} is considered to be formed in the following manner:

$$A_{k+1} \triangleq \begin{bmatrix} A_k \\ a^T_{k+1} \end{bmatrix} \tag{3.16}$$

where $$a^T_{k+1} = [u_{k+1}\ u_k\ \cdots\ u_{k-m}\ y_k\ y_{k-1}\ \cdots\ y_{k-n}] \tag{3.17}$$

a row vector containing the latest set of measurements.

Similarly, the output vector χ_{k+1} is of the form

$$\chi_{k+1} = \begin{bmatrix} \chi_k \\ y_{k+1} \end{bmatrix} \tag{3.18}$$

where y_{k+1} is the latest measurement of the output of the system at the (k+1)th instant corresponding to the input u_{k+1}.

The result is that, when a new pair of input-output measurements is made, a new row is added to the information matrix A_{k+1} and a new element is added to the output vector χ_{k+1}.

With these arrangements, we are now in a position to derive the iterative algorithm [12]. The major results for th kth iteration are summarized below:

Let p be the dimension of vector $\hat{\phi}_k$

For $k \leq p$ (minimum norm solution)

$$\hat{\phi}_{k+1} = \hat{\phi}_k + \frac{Q_{k+1}\, a_{k+1}}{a_{k+1}\, Q_k\, a_{k+1}} (y_{k+1} - a^T_{k+1}\, \hat{\phi}_k) \tag{3.19}$$

where

$$Q_{k+1} = Q_k - \frac{[Q_k\, a_{k+1}]^T\, [Q_k\, a_{k+1}]^T}{a^T_{k+1}\, Q_k\, a_{k+1}} \tag{3.20}$$

and

$$P_{k+1} = P_k + \frac{[Q_k\, a_{k+1}]\, [Q_k\, a_{k+1}]^T\, [1 + a^T_{k+1}\, P_k\, a_{k+1}]}{[a^T_{k+1}\, Q_k\, a_{k+1}]^2}$$

$$- \frac{[P_k\, a_{k+1}]\, [Q_k\, a_{k+1}]^T + [Q_k\, a_{k+1}]\, [P_k\, a_{k+1}]^T}{a^T_{k+1}\, Q_k\, a_{k+1}} \tag{3.21}$$

withe the initial conditions

$$Q_o = I \quad , \quad P_o = 0 \text{ and } \hat{\phi}_o = 0 \tag{3.22}$$

For $k > P$ (least squares estimation)

$$\hat{\phi}_{k+1} = \hat{\phi}_k + \frac{[P\, a_{k+1}]\, [y_{k+1} - a^T_{k+1}\, \hat{\phi}_k]}{1 + a^T_{k+1}\, P_k\, a_{k+1}} \tag{3.23}$$

where $$P_{k+1} = P_k - \frac{[P_k\, a_{k+1}]\, [P_k\, a_{k+1}]^T}{1 + a^T_{k+1}\, P_k\, a_{k+1}} \tag{3.24}$$

The matrices P_k and Q_k are defined as

$$P_k = A_k^{+} A_k^{+T} \tag{3.25}$$

$$Q_k = I - A_k^{+} A_k \tag{3.26}$$

Thus they are both symmetric. The dimensions of matrices P_k and Q_k are both pxp while that of the vectors $\hat{\phi}_k$ and a_k are both p. The storage requirement is minimal. Further, only a total of

$$N = 3p^2 + 4p$$

multiplications are required to calculate equations (3.23) and (3.24) so that this algorithm is regarded as computationally efficient.

The results of the experiments applying this algorithm to the identification of a second order system will be described in greater details in chapter six. Meanwhile, making use of some of the definitions just presented, we shall discuss methods to determine the order of the model.

3.5 Determination of the order of the System

The algorithm discussed in the previous section assumes that we know the value of

$$p = m+n+1 \tag{3.27}$$

where m and n are the number of coefficients in the numerator and denominator of equation (2.3) respectively. Naturally, the values of m and n depend on the order of the model used for identification.

One method to determine the order of the model is to assume certain low values for m and n in equation (3.27) and proceed with the estimation of $\hat{\phi}$. Then, increase m and n by one and repeat the estimation procedures. The step responses of the last two trials are compared. If they are the same or sufficiently close, the latter model is of unnecessary high order. Otherwise, the trial process is carried on until we meet such a requirement. This is admittedly a very crude trial and error approach. We might have problems if the noise level is high.

The following method is a much more systematic approach proposed by Sinha and Pille [12]. It is based on the following theorem.

Theorem:

Consider a system transfer function

$$H(z) = \frac{P(z^{-1})}{Q(z^{-1})} \tag{3.28}$$

where $P(z^{-1})$ and $Q(z^{-1})$ are polynomials of order m and n respectively.

Assume that, in the model, the order of both the numerator and denominator is N which may be chosen arbitrary large. Let

$$q = \mathrm{tr}\, Q_k \tag{3.29}$$

where

$$Q_k = I - A_k^{+} A_k \tag{3.30}$$

which can be obtained iteratively from equation (3.20). If k is incremented from 1 to M ($M \leq 2N$) until q becomes a constant, the true order of the system is given by

$$n = N - q \tag{3.31}$$

Proof:

$$\begin{aligned} \text{Rank } A_k &= tr(A_k^+ A_k) \\ &= 2N - tr\ Q_k \qquad (3.32) \\ &= 2N - q \end{aligned}$$

The maximum rank of A_k is N+n, since A_k has 2n + (N-n) degrees of freedom. Thus, when A_k has attained a maximum rank, q becomes a constant, and

$$N+n = 2N - q$$

$$\text{or} \qquad n = N - q$$

In the event when the order m < n, the algorithm should yield zero for the estimates of $\hat{a}_{m+1}$, $\hat{a}_{m+2}$, . . . $\hat{a}_n$ etc. in equation (2.14) and the order of the system is apparent.

However, the above derivation assumed the effect of noise to be insignificant so that q would approach a constant value. When the noise level is high, we might have difficulty in applying this method.

A third method to the same end takes the presence of noise into account. It is worthwhile to note that, in a noisy environment, several independent methods are sometimes necessary since each method has its own limitations due to the assumptions made in each of them. If they all give the same result, we can be confident that it is the correct value.

The system dynamic equation defined in section 2.3 is

$$\underset{\sim}{Y}_k = A_k \, \hat{\underset{\sim}{\theta}}_k \tag{2.11}$$

where $\underset{\sim}{Y}_k$ = output measurement vector with noise

A_k = information matrix with noisy measurements

$\hat{\underset{\sim}{\theta}}_k$ = estimation vector at the kth instant

$$= [\hat{a}_0 \; \hat{a}_1 \; . \; . \; . \; \hat{a}_m \; \hat{b}_1 \; \hat{b}_2 \; . \; . \; . \; \hat{b}_n]^T \tag{2.14}$$

Let the error function be defined as

$$v = \frac{1}{k} \underset{\sim}{e}^T \underset{\sim}{e} \tag{3.33}$$

According to Van den Boom and Van den Enden [13], assume the noise is white and taking the probability limit of v, we have

$$\underset{k\to\infty}{\text{plim}} \, [v] = \underset{k\to\infty}{\text{plim}} \, [\frac{1}{k} \underset{\sim}{e}^T \underset{\sim}{e}] \tag{3.34}$$

Let $\hat{N}$ be an estimation of the true order N of the system, this leads to the following consideration by taking into account the asymptotic properties of the estimates $\hat{\underset{\sim}{\theta}}_k$.

$$\underset{k\to\infty}{\text{plim}} \, [\hat{a}_i] \begin{cases} \neq a_i & \text{if } \hat{N} < N \quad \text{due to trunction effect} \\ = a_i & \text{if } \hat{N} = N \\ \left.\begin{array}{ll} = a_i, & i \leq N \\ = 0 & i > N \end{array}\right\} & \text{if } \hat{N} > N \end{cases} \tag{3.35}$$

$$\underset{k\to\infty}{\text{plim}}\,[\hat{b}_i] \begin{cases} \neq b_i & \text{if } \hat{N} < N \quad \text{due to truncation effect} \\ = b_i & \text{if } \hat{N} = N \\ = b_i & i \leq N \\ = 0 & i > N \end{cases} \Big\} \text{ if } \hat{N} > N \tag{3.36}$$

These conditions result in

$$\text{plim}\,[\frac{1}{k}\, e^T\, e] \begin{cases} >0 & \text{for } \hat{N} < N \\ =0 & \text{for } \hat{N} \geq N \end{cases} \tag{3.37}$$

An important consequence is apparent in that there is a marked change in the behaviour of the error function v when $\hat{N} = N$. Pictorially, its behavious is shown in figure 3.1.

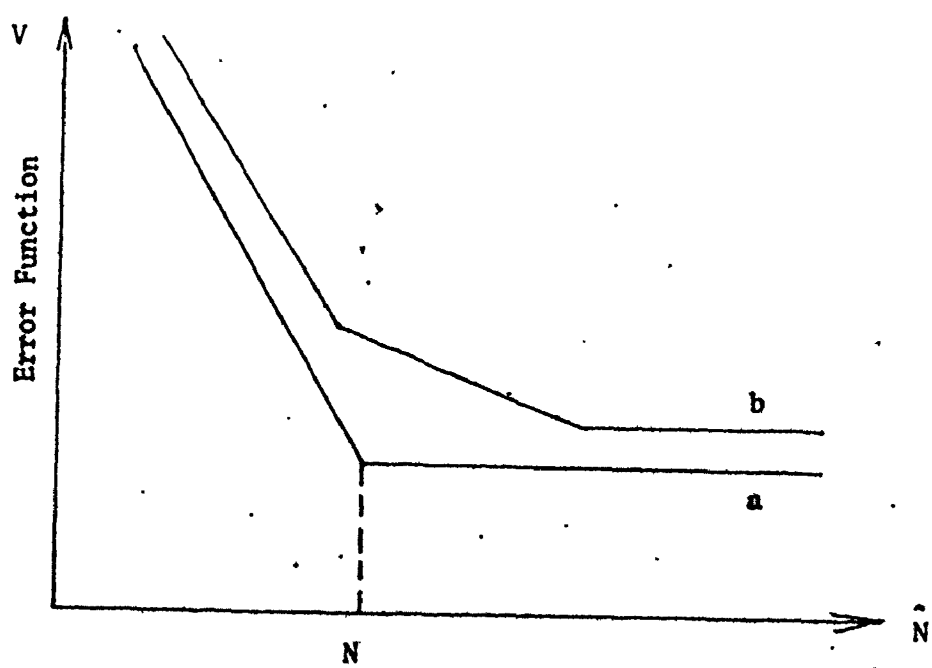


Figure 3.1 Error Function V versus Estimated Order $\hat{N}$

CHAPTER FOUR

STOCHASTIC APPROXIMATION FOR THE REMOVAL OF BIAS

4.1 Effect of Measurement Noise on the Estimation

Again refer to the basic dynamic equation

$$A_k \hat{\theta}_k = \chi_k \tag{2.11}$$

where the solution is given by

$$\hat{\theta}_k = A_k^{+} \chi_k \tag{3.13}$$

Matrix A_k is the information matrix containing the input-output measurements. As noted earlier, these measurements are contaminated with noise. We shall analyze the effect of noise on the final estimates.

Let matrix A_k be decomposed into two component matrices as follows (dropping the subscript k):

$$A = B + N \tag{4.1}$$

where B is the noise-free component of A and N is the noise matrix.

Similarly, the output can be decomposed to

$$\chi = \xi + \chi \tag{4.2}$$

where ξ is the true output and χ is the output noise.

As shown in the previous chapter, for least squares estimation, the pseudoinverse A^+ is given by

$$A^+ = [A^T A]^{-1} A^T$$

Substituting equation (4.1)

$$A^+ = [(B+N)^T (B+N)]^{-1} [B+N]^T$$

$$= [B^+B + B^TN + N^TB + N^TN]^{-1} [B+N]^T \qquad (4.3)$$

If the noise is white and hence uncorrelated with the input and output, we have

$$B^TN \rightarrow 0 \qquad\qquad N^TB \rightarrow 0$$

Then, equation (4.3) becomes

$$A^+ = [B^TB + N^TN]^{-1} [B+N]^T \qquad (4.4)$$

Using the identity [14]

$$[B^TB + N^TN]^{-1} = [B^TB]^{-1} [I + (N^TN)^{-1} (B^TB)]^{-1} [B+N]^T \qquad (4.5)$$

equation (4.4) becomes

$$A^+ = [I-Z] B^+ + [I-Z] [B^TB]^{-1} N^T \qquad (4.6)$$

where

$$Z \triangleq [I + (N^TN)^{-1} (B^TB)]^{-1}$$

$$= [N^TN + B^TB]^{-1} [N^TN] \qquad (4.7)$$

Substituting (4.6) and (4.7) into (4.1), we get .

$$\hat{\underset{\sim}{\phi}} = [I-Z]\, B^{+}\, \underset{\sim}{\xi} + [I-Z]\, [B^T B]^{-1}\, N^T\, \underset{\sim}{\xi}$$

$$= [I-Z]\, B^{+}\, \underset{\sim}{\chi} + [I-Z]\, [B^T B]^{-1}\, N^T\, \underset{\sim}{\chi} \qquad (4.8)$$

For uncorrelated white noise

$$N^T\, \underset{\sim}{\xi} \rightarrow 0$$

$$B^{+}\, \underset{\sim}{\chi} \rightarrow 0$$

$$N^{+}\, \underset{\sim}{\chi} \rightarrow 0$$

Hence, equation (4.8) becomes

$$\hat{\underset{\sim}{\phi}} = [I-Z]\, B^{+}\, \underset{\sim}{\xi}$$

$$= [I-Z]\, \underset{\sim}{\phi} \qquad (4.9)$$

Since $\underset{\sim}{\phi} = B^{+}\, \underset{\sim}{\xi}$

Thus $\underset{\sim}{\phi} - \hat{\underset{\sim}{\phi}} = [N^T N + B^T B]^{-1}\, [B^T N] \qquad (4.11)$

$$\neq 0 \quad \text{if } N \neq 0$$

The difference given by equation (4.11) is called the bias of the estimation due to the presence of measurement noise.

4.2 Removal of Bias by Filtering

The result in the previous section holds true only for white uncorrelated additive noise. In practice, the noise is coloured. Coloured noise can be modelled as the output resulting from passing white noise through a finite order transfer function. Recognizing this fact, we can devise a method to find out this noise model. From this model, we can find its inverse with which a filter is constructed. With this filter, we can remove the noise component from the measurements by passing them through this filter.

This approach is not unlike the Wiener-Hopf filtering method. In order to achieve this, we have to first identify the noise model and secondly develop a filtering mechanism. All these have to be integrated with the pseudoinverse algorithm we have just discussed. Further it must be an iterative algorithm so that it can be applied on-line. Finally, the amount of extra computation involved should be as little as possible in order to end up with a still efficient algorithm.

Sen and Sinha [14] have proposed a scheme by applying stochastic approximation to find the noise filter working in parallel with the pseudo-inverse algorithm. The derivation of this algorithm will be given in a later section while we pause to introduce the basic principles of stochastic approximation.

4.3 Stochastic Approximation

Stochastic approximation may be regarded as a scheme for successive approximation of a sought quantity when the observations involve random errors due to the stochastic nature of the problem. It has the following advantages:

(a) Only a small interval of data needs processing.

(b) Only simple computations are required.

(c) A priori knowledge of the process statistics is not required, nor is the functional relationship between the desired parameters and the observed data. The only requirements are that it satisfies certain regularity conditions and that a unique solution exists.

Many major contributions are made to the area by various people. A comprehensive survey paper by Sakrison [15] gives a good general picture of various aspects of the subject.

First, let us look at the Robbins-Monro approach [16] which is the statistical analogue of the simple gradient method for finding the root of the equation

$$h(x) = 0 \tag{4.12}$$

which is

$$x_{i+1} = x_i - K_i h(x_i) \tag{4.13}$$

where $\{K_i\}$ is a sequence of real numbers which must satisfy certain conditions to ensure that the algorithm will converge.

When there is additive random noise, h(x) becomes

$$Z(x_i) = h(x_i) + v_i \tag{4.14}$$

where $\{v_i\}$ is a zero mean noise sequence.

Making use of the fact that the expectation of $Z(x_i)$ is $h(x_i)$, equation (4.13) may be modified to

$$x_{i+1} = x_i - K_i\, Z(x_i) \tag{4.15}$$

Robbins and Monro showed that equation (4.15) will converge if the following conditions are met:

$$\lim_{i \to \infty} K_i = 0 \quad ,$$

$$\sum_{i=1}^{\infty} K_i = \infty \quad , \qquad (4.16)$$

and $$\sum_{i=1}^{\infty} K_i^2 < \infty$$

A simple sequence which meets these requirements is

$$K_i = \frac{\alpha}{\beta + i} \qquad (4.17)$$

where α and β are positive constants. Also, it is required that $h(x)$ be bounded on either side of a true solution by straight lines, such that it is not possible to overshoot the solution x which cannot be corrected by a K_i satisfying equation (4.17).

Kiefer and Wolfowitz [17] extended the method to find the extremum of an unknown unimodal regression function $\theta(u)$. This approach is the exact analogue of the gradient approach in the deterministic optimization procedure which yields

$$U_{i+1} = U_i - K_i \frac{d\ \theta(U_i)}{d\ U_i} \qquad (4.18)$$

The stochastic counterpart is

$$U_{i+1} = U_i - K_i \frac{d\ \eta(U_i)}{d\ U_i} \qquad (4.19)$$

where $\eta(u) = \theta(u) + \xi$

and ξ is the random noise component.

Since the differentiation in equation (4.19) does not exist in general, one may use the following approximation

$$\frac{d\ \eta(U_i)}{d\ U_i} \simeq \frac{\eta(U_i+\Delta U_i) - \eta(U_i-\Delta U_i)}{2\Delta U_i} \tag{4.20}$$

Convergence is guaranteed if the following conditions are satisfied:

$$\begin{aligned} &\lim_{i\to\infty} K_i = 0 \ , \\ &\lim_{i\to\infty} \Delta U_i = 0 \ , \\ &\sum_{i=1}^{\infty} K_i = \infty \ , \\ &\sum_{i=1}^{\infty} K_i^2 < \infty \ , \\ \text{and} \quad &\sum_{i=1}^{\infty} \left[\frac{k_i}{\Delta U_i}\right]^2 < \infty \end{aligned} \tag{4.21}$$

A basic idea [15] of stochastic approximation is that a stochastic counterpart exists for any deterministic algorithm. Fu et al [20], Sinha and Griscik [18] and Kwanty [19] have proposed specific formulae to implement the above ideas.

4.4. Formulation of the Noise Model

Let the system dynamic equation be represented in the following form:

$$[1 + A(z^{-1})]y_i = [b_o + B(z^{-1})]U_i + e_i \tag{4.22}$$

where $e_i = [1 + A(z^{-1})]n_i$, the residual error (4.23)

$\{n_i\}$ is a zero mean random noise sequence

$$A(z^{-1}) = a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}$$

$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}$$

To guarantee stability of the process, the roots of $[1 + A(z^{-1})]$ are assumed to lie inside the unit circle.

We assume that the noise sequence $\{n_i\}$ can be described as a filtering of a well behaved, zero mean white noise signal ξ_i i.e.

$$n_i + \sum_{j=1}^{p} d_j \; n_{i-j} = \xi_i + \sum_{i=1}^{q} g_i \; \xi_{i-j}$$

or equivalently,

$$n_i = \frac{1 + G(z^{-1})}{1 + D(z^{-1})} \xi_i \tag{4.24}$$

where $D(z^{-1}) = d_1 z^{-1} + d_2 z^{-2} + \ldots + d_p z^{-p}$

$$G(z^{-1}) = g_1 z^{-1} + g_2 z^{-2} + \ldots + g_q z^{-q}$$

Again, the roots of $[1 + D(z^{-1})]$ are assumed to lie within the unit circle.

Combining (4.23) and (4.24), we have

$$e_i = \frac{[1 + A(z^{-1})]\,[1 + G(z^{-1})]}{[1 + D(z^{-1})]}\,\xi_i \qquad (4.25)$$

The above residual error sequence $\{e_i\}$ is now approximated by a low order linear process. The filter involved is the inverse of this process. This is similar to the method of "pre-whitening" used in spectral density estimation. Two possible processes are suitable for this purpose. They are the moving average process of the form

$$e_i = \xi_i + \sum_{r=1}^{p} m_r z^{-r}\,\xi_i \qquad (4.26)$$

and the autoregressive process of the form

$$e_i + \sum_{r=1}^{p} f_r z^{-r}\,e_i = \xi_i \qquad (4.27)$$

These processes are duals of each other as a moving average process filtered by an autoregressive filter becomes a white noise or vice versa.

In the present study, an autoregressive model is chosen. In particular, if we have cascaded filters of the type

$$\hat{e}_i' = [1 + \sum_{j=1}^{p} f_j z^{-j}]\,e_i' \qquad (4.28)$$

we can approximate the true process to any degree of accuracy by choosing an appropriate sequence $\{f_i\}$. Using this principle, equation (4.25) can therefore be approximated by

$$e_i = \frac{\xi_i}{[1 + F_s(z^{-1})]} \tag{4.29}$$

where $$F_s(z^{-1}) = f_1 z^{-1} + f_2 z^{-2} + \ldots + f_s z^{-s}$$

This implies that

$$[1 + F_s(z^{-1})] = [1 + D(z^{-1})]\ [1 + A(z^{-1})]^{-1}\ [1 + G(z^{-1})]^{-1}$$

This is true if sufficient number of terms of the filter on the left hand side are used.

Substitute (4.29) into (4.22), we have

$$[1 + A(z^{-1})]\ y_i = [b_o + B(z^{-1})]\ u_i + \frac{\xi_i}{[1 + F_s(z^{-1})]}$$

Multiplying throughout by $[1 + F_s(z^{-1})]$, we get

$$[1 + A(z^{-1})]\ [1 + F_s(z^{-1})]\ y_i = [b_o + B(z^{-1})]\ [1 + F_s(z^{-1})] u_i + \xi_i$$

or

$$[1 + A(z^{-1})]\ \overline{y}_i = [b_o + B(z^{-1})]\ \overline{u}_i + \xi_i \tag{4.30}$$

where $$\overline{y}_i = (1 + f_1 z^{-1} + f_2 z^{-2} + \ldots + f_s z^{-s})\ y_i$$

$$\overline{u}_i = (1 + f_1 z^{-1} + f_2 z^{-2} + \ldots + f_s z^{-s})\ U_i \tag{4.31}$$

ξ_i is a white noise sequence.

If the filter is known, the measurements y_i and u_i will be filtered in such a manner as in equation (4.31) to obtain the filtered input-output pair $\bar{u}_i$ and $\bar{y}_i$. These filtered quantities are then used in the algorithm to calculate the estimates. Comparing equations (4.22) and (4.31), the residual error sequence has now been changed to a white noise sequence. The final estimates will therefore be unbiased because the white residual error is not correlated with the input and output.

The next logical step is to find the parameters of the filter $F_s(z^{-1})$. This is where stochastic approximation comes into the picture and is the subject of discussion in the next section.

4.5 Application of Stochastic Approximation

At the kth iteration, the residual error is represented by

$$\hat{e}_k = F^T E_k + W_k \tag{4.32}$$

where $F^T = [f_1 \; f_2 \; . \; . \; . \; f_s]^T$, the filter parameter vector

$E_k = [-\hat{e}_{k-1} \; -\hat{e}_{k-2} \; . \; . \; . \; -\hat{e}_{k-s}]^T$, the error vector

W_k is a white noise sequence

s is the order of the noise filter

To obtain F, the stochastic approximation method of the form proposed by Kwanty [19] is employed, i.e.

$$\hat{F}_{k+1} = \hat{F}_k - \frac{\gamma}{k+1} \frac{[F_k^T E_k - \hat{e}_k] \, E_k}{||E_k||^2} \tag{4.33}$$

where γ is a positive gain constant chosen large enough to guarantee convergence.

$$F_k = \text{kth estimate of } F$$

$$= [\hat{f}_1(k)\ \hat{f}_2(k)\ .\ .\ .\ \hat{f}_s(k)] \qquad (4.34)$$

Using the estimated filter $\hat{F}_k$, the input and output measurements u_k and y_k respectively are filtered to obtain

$$\bar{u}_k = u_k + \sum_{i=1}^{s} \hat{f}_i\ u_{k-i}$$
$$\bar{y}_k = y_k + \sum_{i=1}^{s} \hat{f}_i\ y_{k-i} \qquad (4.35)$$

These filtered quantities are used in the updating of the information matrix A_k in the pseudoinverse algorithm. Since $\hat{F}_k$ is only an approximation to the true F, using $\hat{F}_k$ as filter will not remove all the bias but only part of it.

The results of experiments on a second order system are presented in chapter six.

CHAPTER FIVE

ANALOG SIMULATION AND HARDWARE

5.1 Introduction

The two algorithms we have just discussed are implemented and tested in real time as applied to the identification of a one-input one-output second order system. In the present chapter, a detailed description of the simulation and hardware will be provided.

Figure 5.1 shows the general layout of the hybrid set up of the experiment. It consists of three different types of equipments. The first one is the TR-20 analog computer. The integrators, summing operational amplifiers and the potentiometers on it provide the simulation of the system to be identified as well as the pseudorandom analog input signal. The second piece of major equipment is the PDP-11/45 minicomputer. This computer is of recent design with a unibus. It allows us to address any peripheral device as convenient as any other memory locations and thus facilitate the data acquisition procedures. The memory size is 20K with both the fixed-head and moving-head disks. It is also equipped with a hardwired floating-point arithmetic processor and a real time clock. Loading and running of programs can be done easily through the system monitor which is a software package provided by the manufacturer. The analog and digital computers are coupled together by the interface panel in between. Mounted on this panel are the sampling devices and control circuits to co-ordinate the sampling process.

Control of the analog computer is a manual mechanical switch. However, the program provides a feature to co-ordinate the on-off switching of the analog computer and the starting of the identification algorithm. This is done by having the computer to repeatedly check the data buffer of one channel at the beginning. If the analog computer is off, this data buffer is zero. As soon as a certain threshold value is being detected in this data buffer, it is understood by the program that the analog computer has been switched on and it will proceed with the rest of the program. The non-zero threshold value is necessary because of the presence of a small amount of noise in the system. Experience shows that about 0.075 volt is enough.

The control of program running itself is by means of the switch register console on the computer and the keyboard. Output device can either be the cathode ray screen or the teletype printer. Since each device works on a different speed, precise timing is necessary. This can be accomplished by utilizing the priority interrupt structure of the digital computer.

For experimental purposes, an external noise generator is installed to provide noise at different power levels. The noise is artificially introduced into the output terminal of the system. For all practical purposes, we can assume that this noise source gives us white guassian noise with a very wide spectrum.

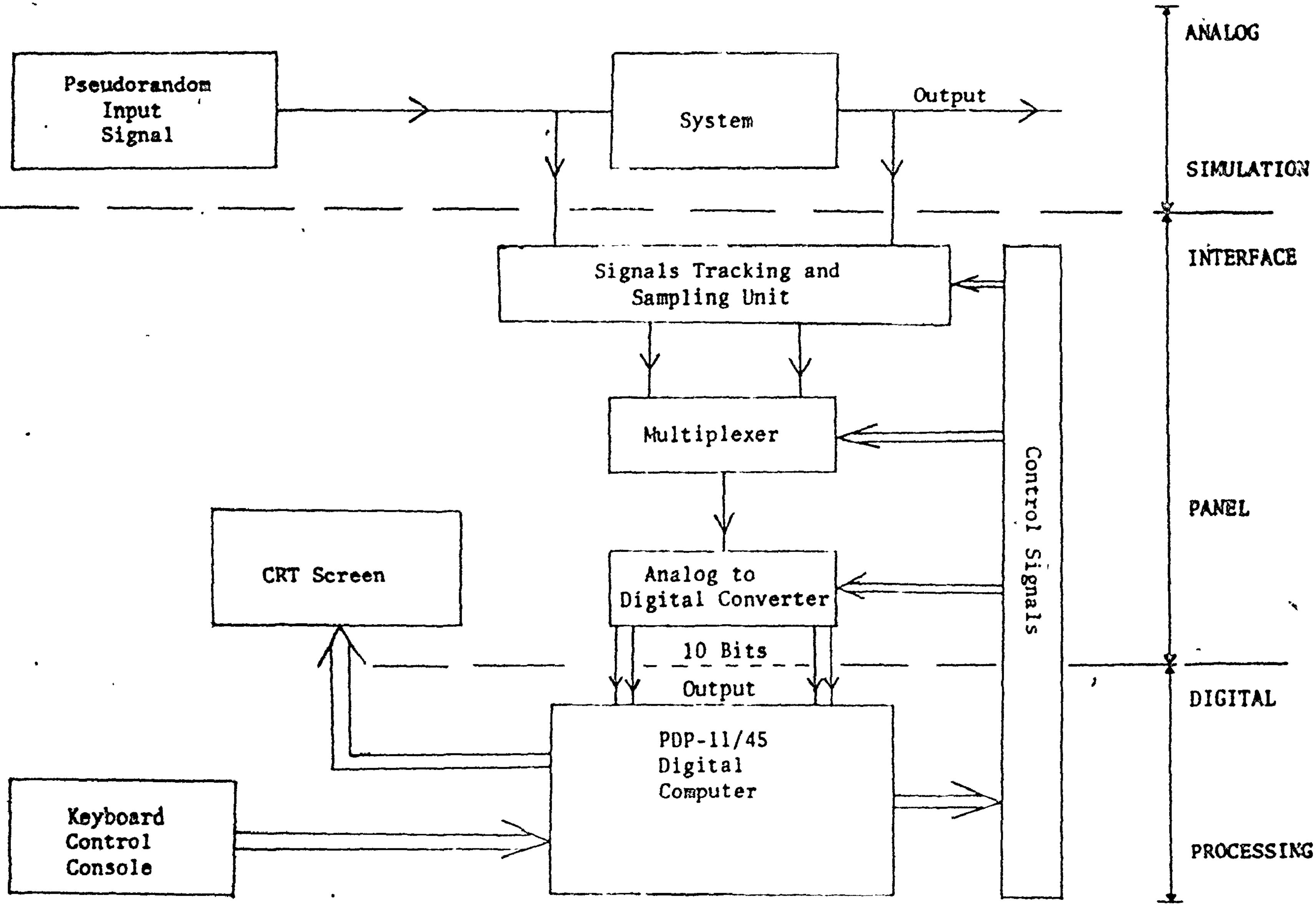


Figure 5.1 General Schematic of Experiment

5.2 Interface Hardware

The interfacing panel has been designed to handle two channels of input signals. The circuit diagram is shown in figure 5.2a.

The two channels of incoming signals are connected to a signal sampling unit which consists of two sample-and-hold modules so that simultaneous sampling of both channels is possible. This is necessary to obtain corresponding input and output measurements at the same time instant.

There is only one analog-to-digital converter capable of converting one voltage at a time. The multiplexer is situated in front of it to act as a switch. It can be programmed in such a way that different channels are presented to the input of the analog-to-digital converter individually in a pre-determined order. The binary coded digital outputs from the analog-to-digital converter are fed into a data buffer register which can be directly accessed by the digital computer to facilitate a data transfer into the core memory.

The sequence of actions of the sampling process are co-ordinated by programmed control signals together with hardwired control logic circuits. The circuit diagram of the control logic is shown in figure 5.2b.

The control bits from the channel selector register are gated to control the switching of the multiplexer through two flip-flops. The time delay circuit is used to delay the conversion trigger pulse so that the analog-to-digital converter is triggered to start the conversion just after the selected channel has been switched to the input of the converter by the multiplexer. The delay introduced in the path travelled by the bus ready pulse is about fifty nanoseconds. The required delay for the trigger is then equal to this length of time plus the time for the analog voltage to settle in the multiplex output.

When the conversion is complete, it is signalled to the computer through a change in the logic level of the end-of-conversion output in the analog-to-digital converter. Now, the computer can transfer the data into core. After reading in the data, the computer is ready for another cycle of action.

The control of sampling frequency is done by setting an external frequency control switch on the interface panel. A detailed circuit diagram of the clock frequency generator is shown in figure 5.2c. The different frequencies are generated by allowing the clock pulses originated from the crystal clock inside the computer to pass through a different number of decade counters depending on the setting of the switch. There are a total of fifteen choices. At the lower end, we can either select one, two or five hertz. By bypassing one decade counter, we can generate pulses ten times faster. There are altogether four decade counters so that we can step up the above frequencies by four folds. However, for the purpose of process identification, we seldomly need such high frequencies. In the program written, the user can step down the sampling frequency set on the switch by any integral number of times by entering an integer constant from the keyboard. The program will make use of the integer supplied to determine the number of times it will loop through a delay loop in it.

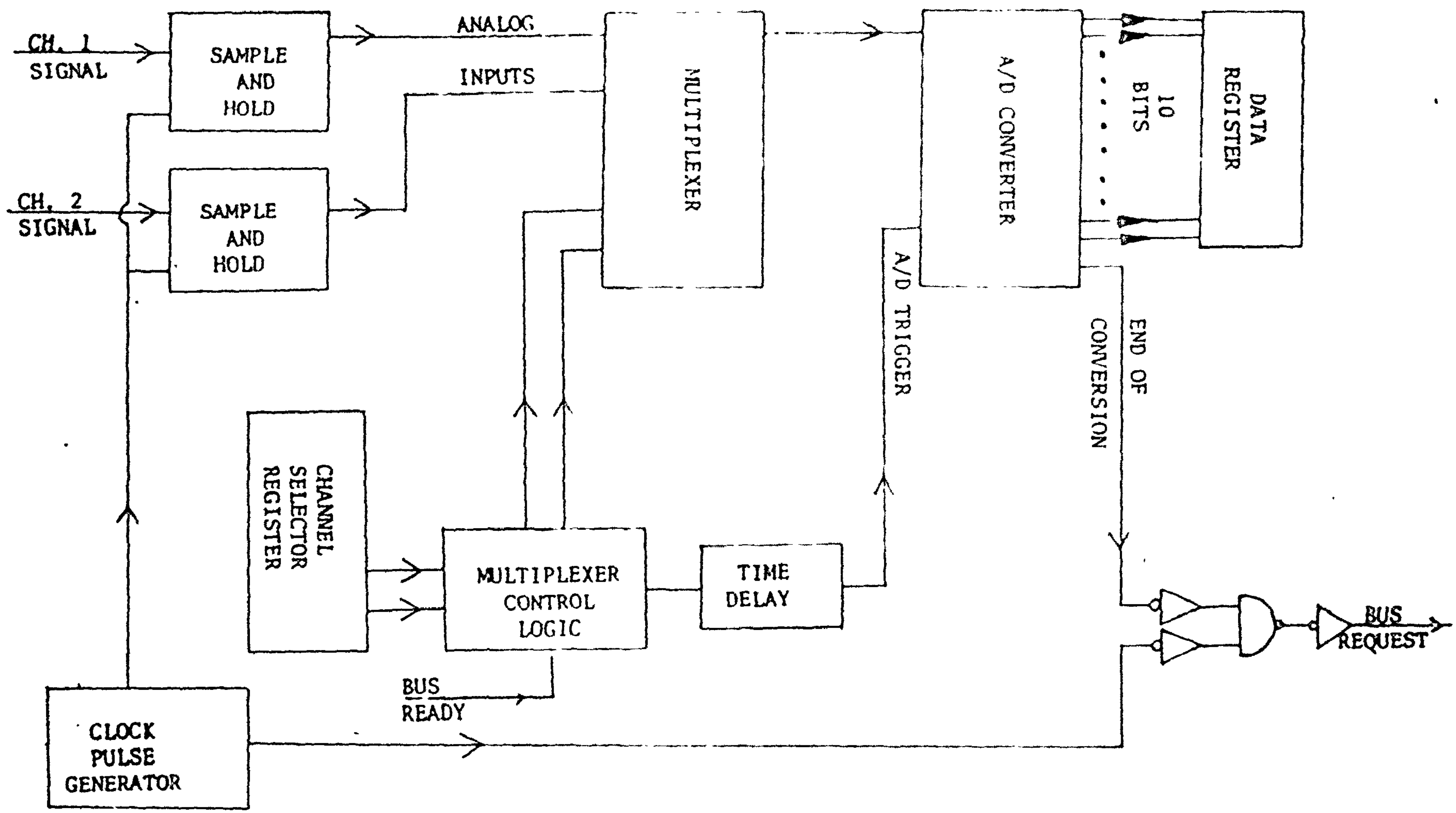


Figure 5.2a Circuit Diagram of Interface Panel

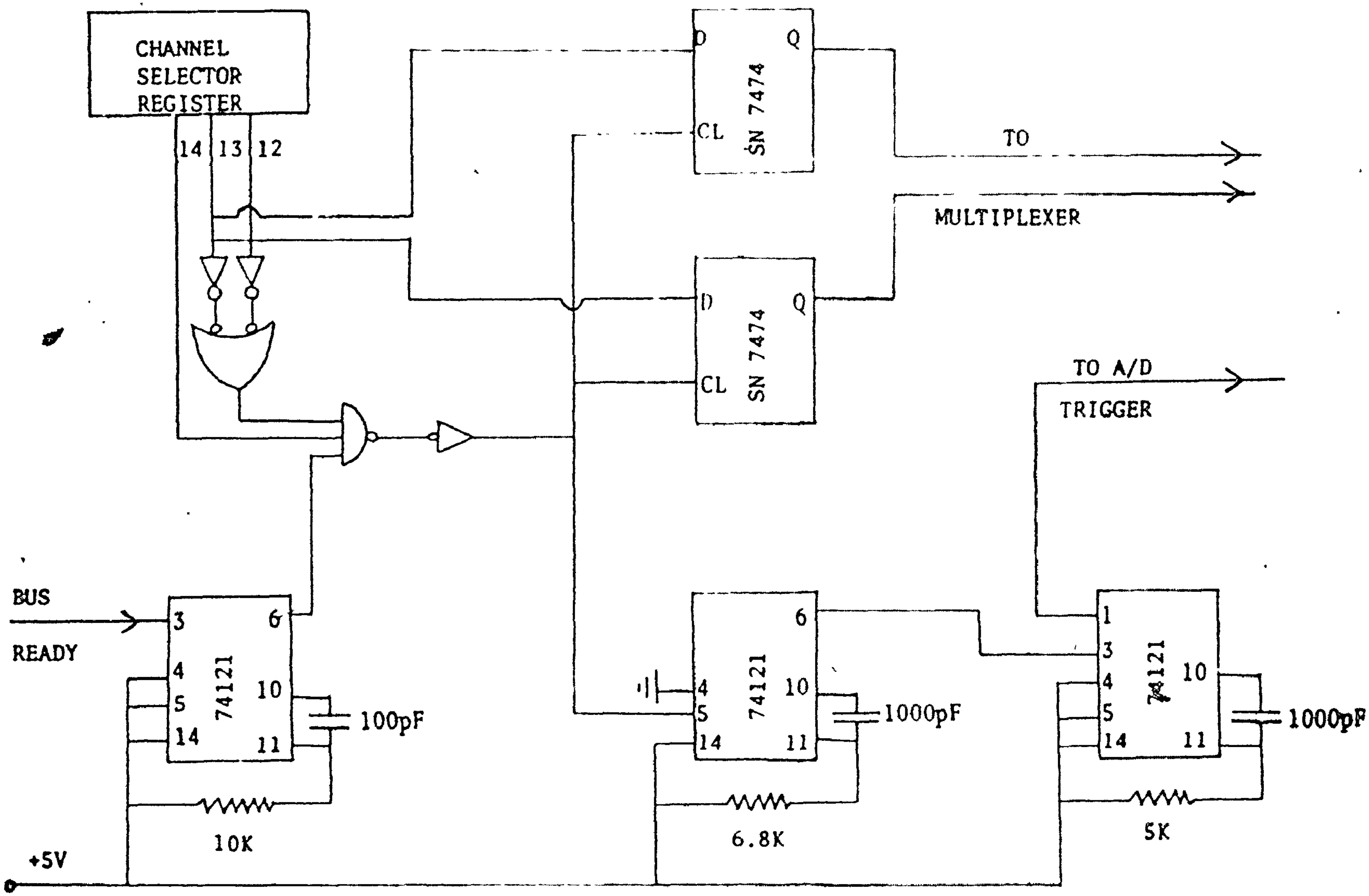


Figure 5.2b Circuit Diagram of Control Logic

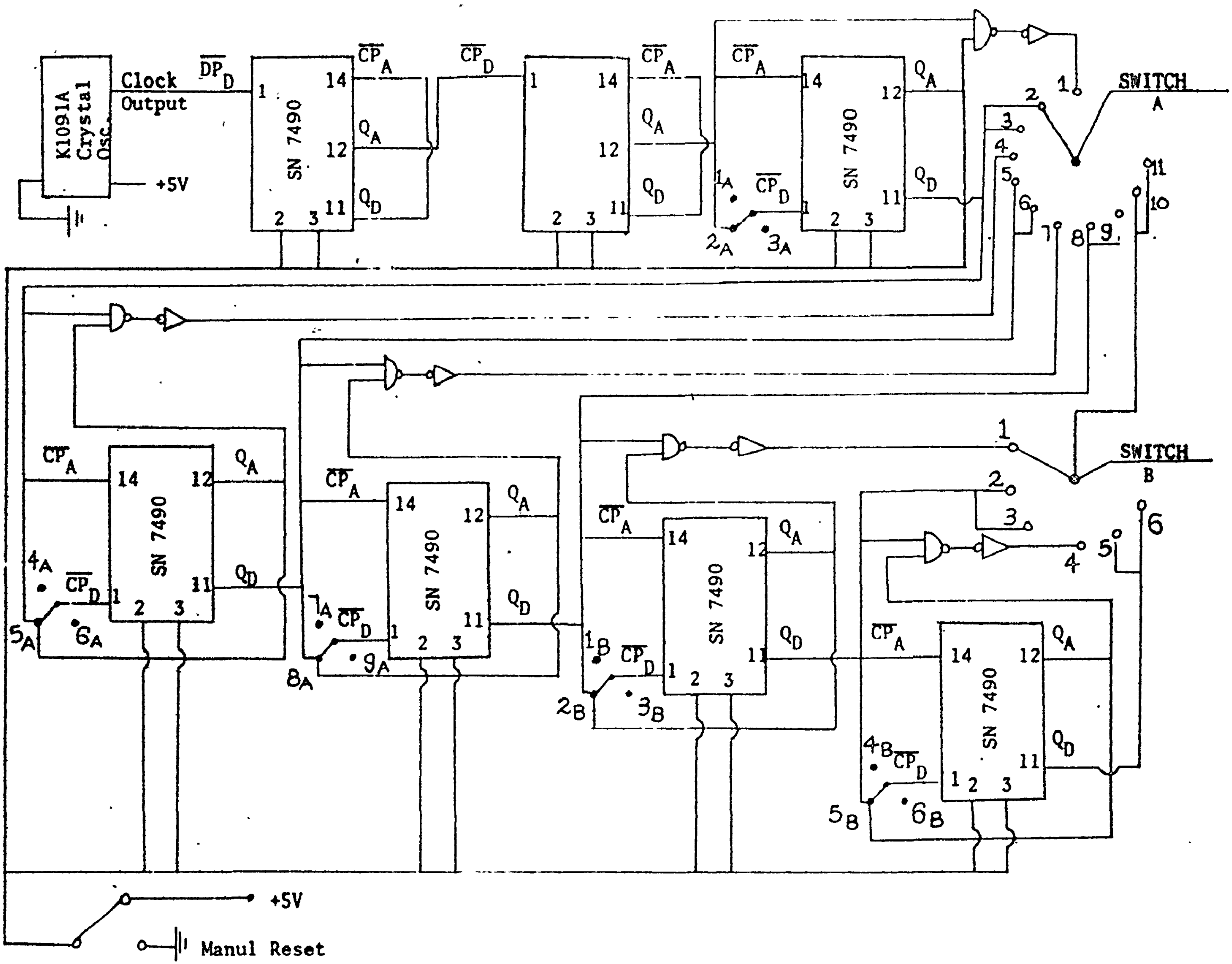


Figure 5.2c Circuit Diagram of Clock Pulse Generator

Referring back to figure 5.2a, the clock pulse is gated with the end-of-conversion output and is also fed into the sample-and-hold modules. The fomer connection is used for producing a bus request which is primarily responsible for the generation of an interrupt. The latter connection is used to switch the sample-and-hold modules to either one of two different modes of operation depending on the need. In the track mode, the sample-and-hold will be tracking the voltage level of the input signal. The second mode is the hold mode during which the voltage is being held at the level just before the switching signal comes in. Since both sample-and-hold modules are wired together, they go to the hold mode at the same time and hence simultaneous sampling of both channels. These samples will eventually be transmitted to the analog-to-digital converter for conversion. The digital outputs are received by a data buffer for final transfer to the computer.

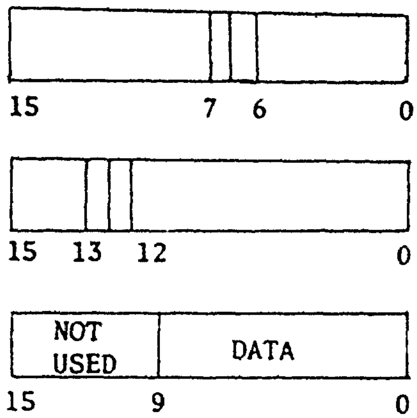
Having briefly reviewed the functions of each piece of hardware in the interface panel, we shall describe the sequence of operations in the next section.

5.3 Operating Sequence

The whole interrupt sequence is effected by three sources of control working together, namely the external clock, the control logic circuits and the software.

There are three specialized registers in the computer central to the whole operation (see figure 5.3). The first one is the control-and-status register (ADCSR) at location 177520. Bit six is the interrupt enable bit and has to be set by the program to initiate the interrupt sequence. Bit seven is the bus request bit set by the external circuits. When it is set, an interrupt will be generated. Depending on the priority of this interrupt request and that of the task being processed at that time, the computer will determine whether to honour the request immediately or wait until the higher priority jobs have been completed. The priority of the interrupt request is of course chosen by the programmer. The second register is the channel selector register (CHNSLR) at location 177522. Since there are two channels, we need a two-bit binary code to represent them. Bits twelve and thirteen of this register are used for this purpose. They decide which channel is being selected for conversion. The third register is the data buffer register (DATBUF). This is simply a data logging register serving as a temporary storage for the outputs of the analog-to-digital converter.

A timing diagram is shown in figure 5.4. There are six curves in the figure. The clock pulse curve determines the mode of the sample-and-hold modules. They are in track mode when the clock pulse is HI and in hold mode when the clock pulse is LO. The end-of-conversion (EOC) curve is normally at LO level except when conversion is taking place. The bus request curve is formed by ANDing the logical compliments of the first two curves. If the interrupt enable bit in the ADCSR is set, an interrupt will be generated whenever there is a positive logical transition from LO to HI in the bus request curve. In this situation, we have the sample-and-hold modules holding the signals and the previous conversion has been completed. Priority permitting and depending on the contents of a two-word interrupt vector at locations 110 and 112, the computer will honour the request by



Control and Status Register (ADCSR)
Address = 177520
Bit 6 -- set for interrupt enable
Bit 7 -- set for bus request

Channel Selector Register (CHNSLR)
Address = 177522
Bit 12 -- channel 1 selected if set
Bit 13 -- channel 2 selected if set

Data Register (DATREG)
Address = 177524
10 bits two's complement format

Figure 5.3 Special Registers

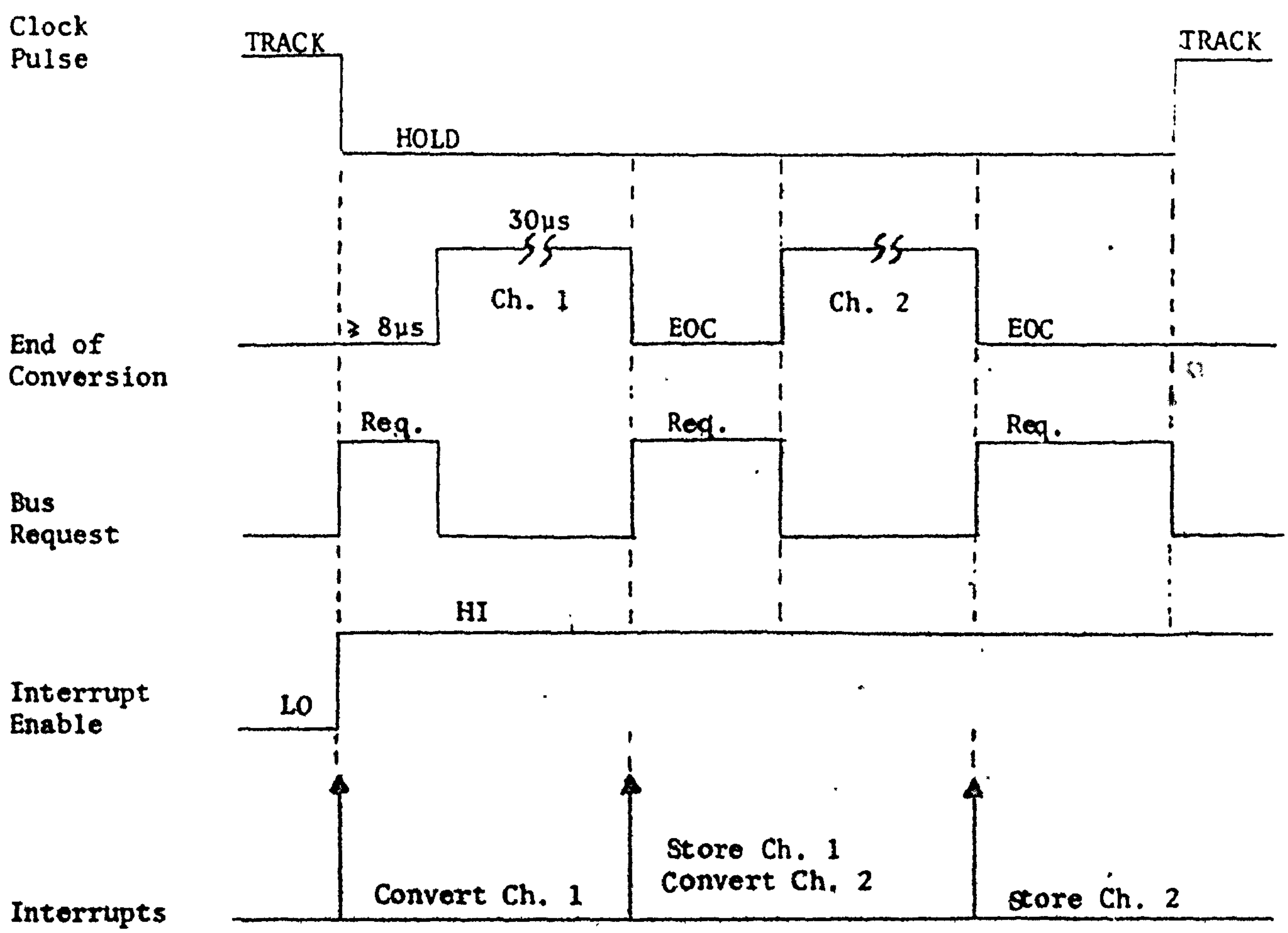


Figure 5.4 Timing Diagram

executing the subroutine pointed to by the first word of the vector with a priority according to the second word of the vector. The type of task performed is under program control. The program can change the contents of the vector to assign different tasks to different interrupt requests As indicated in figure 5.4, the first interrupt service subroutine commands the interface panel to convert the first channel. The next one is for storing the data of the first channel in an assigned location in core followed by a command to convert the next channel. Finally, the third one is to store the data from channel two.

After servicing both channels, the cycle can be made to repeat itself for any desired length of time by simply keeping the interrupt enable bit in ADCSR set. If no further sampling is desired, the interrupt enable bit is cleared to inhibit further action.

5.4 Pseudorandom Input Signal

In section 3.4, we have noted that in order to make the assumption about the rank of the information matrix, the input signal has to satisfy one of several conditions. One such condition is to excite the system by a random signal. In practice, true white noise signal cannot be realized physically. However, we can synthesize pseudorandom signals that would still satisfy our purpose. What we really need is a random sequence during the finite time interval when the identification process is taking place. We have at least two convenient methods at our disposal. The first method is to use a pseudorandom binary sequence (PRBS) generated from switch registers.

In our present study, we use the technique of summing many sinusoids with randomly selected phase angles to produce a pseudorandom signal. The use of sinusoids for this purpose has the important advantage that we can obtain whatever spectral density we desire. This is important because in identification problems, we can excite every mode of the system dynamics by a proper choice of the component sinusoids.

The basic principle can be understood by referring to figure 5.5 which represents a certain specified spectral density distribution of signal y(t). The curve is divided into 2m parts of equal area. We can now replace the whole spectrum by pairs of impulse functions in both positive and negative frequencies as shown in figure 5.6. The frequency of each impulse function corresponds to the centre frequency of each of the 2m partitions. The magnitudes of the impulses are all equal to the area of each portion they replace. The pair of positive and negative spectra constitutes one sinusoid of randomly selected phase angles.That is, the approximation of the pseudorandom signal will be

$$y(t) = \sqrt{A} \sum_{k=1}^{m} \sin (w_k k + \phi_k)$$

where $A = \frac{2}{\pi}$ x area of each partition

ϕ = randomly sected phase angle

If m is increased to infinity, we would obtain a truely random signal that matches the specified spectral density exactly.

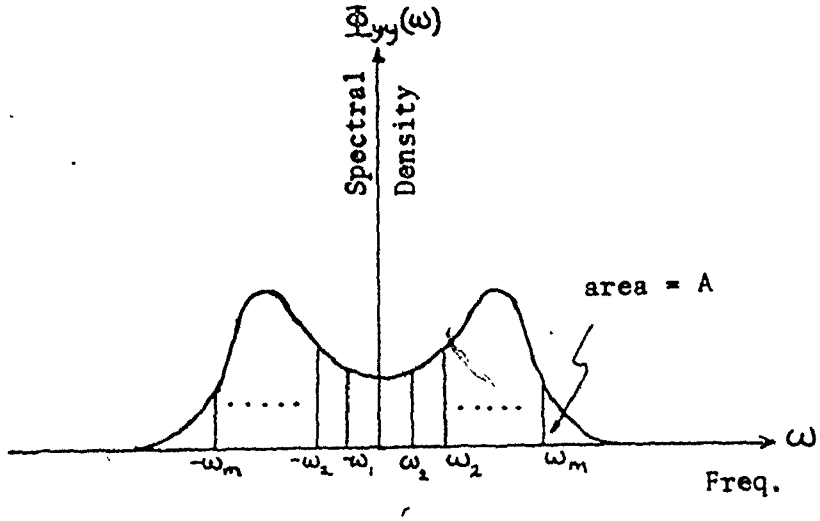


Figure 5.5 Spectral Density of y(t)

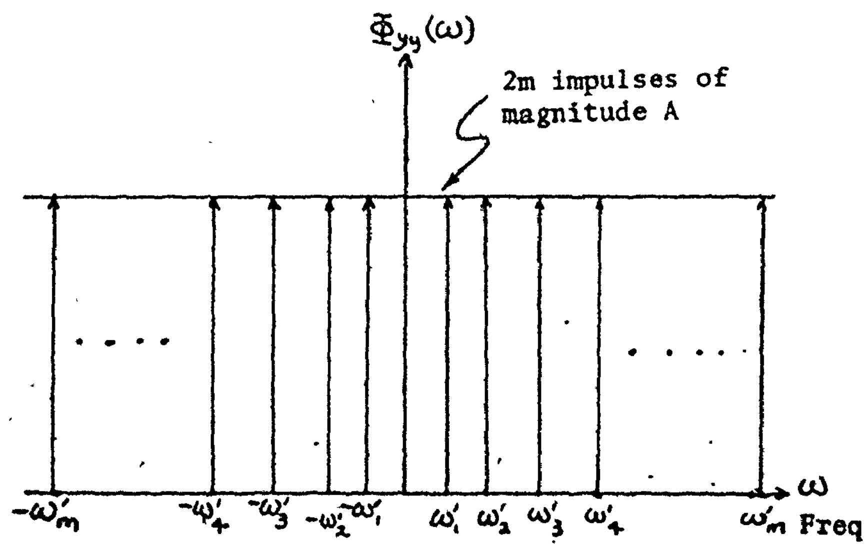


Figure 5.6 Equivalent Impulse Functions

5.5 Analog Simulations

Analog simulations of the system to be identified and the input signal we have just discussed are done on the TR-20 analog computer. A complete circuit diagram is shown in figure 5.7.

Simulation of the one-input one-output second order system is relatively straight forward requiring only two integrators and two summing amplifiers in addition to the potentiometers.

The input signal is constructed by superposing several sinusoids. Each sinusoid requires two integrators and an inverter. The choice of frequency for the construction of the pseudorandom signal must be such that they are not integrally related. This is to avoid pattern repetition by eliminating the presence of subharmonics. Fortunately, this can be easily satisfied when we use analog simulation.

Note that even though the phase angles are randomly selected, these sinusoids become deterministic signals once they are fixed. The resulting signal using a finite number of these components is therefore also deterministic and has a certain finite repetition frequency. This frequency is equal to the least common multiple of all the component frequencies and can be made small by proper adjustment. Again, using analog simulation, it poses no serious problem. Experience shows that at least five sinusoids are needed for our purpose.

The fact that we have a deterministic input signal is in effect an advantage in the experimental work because this signal is also repeatable.

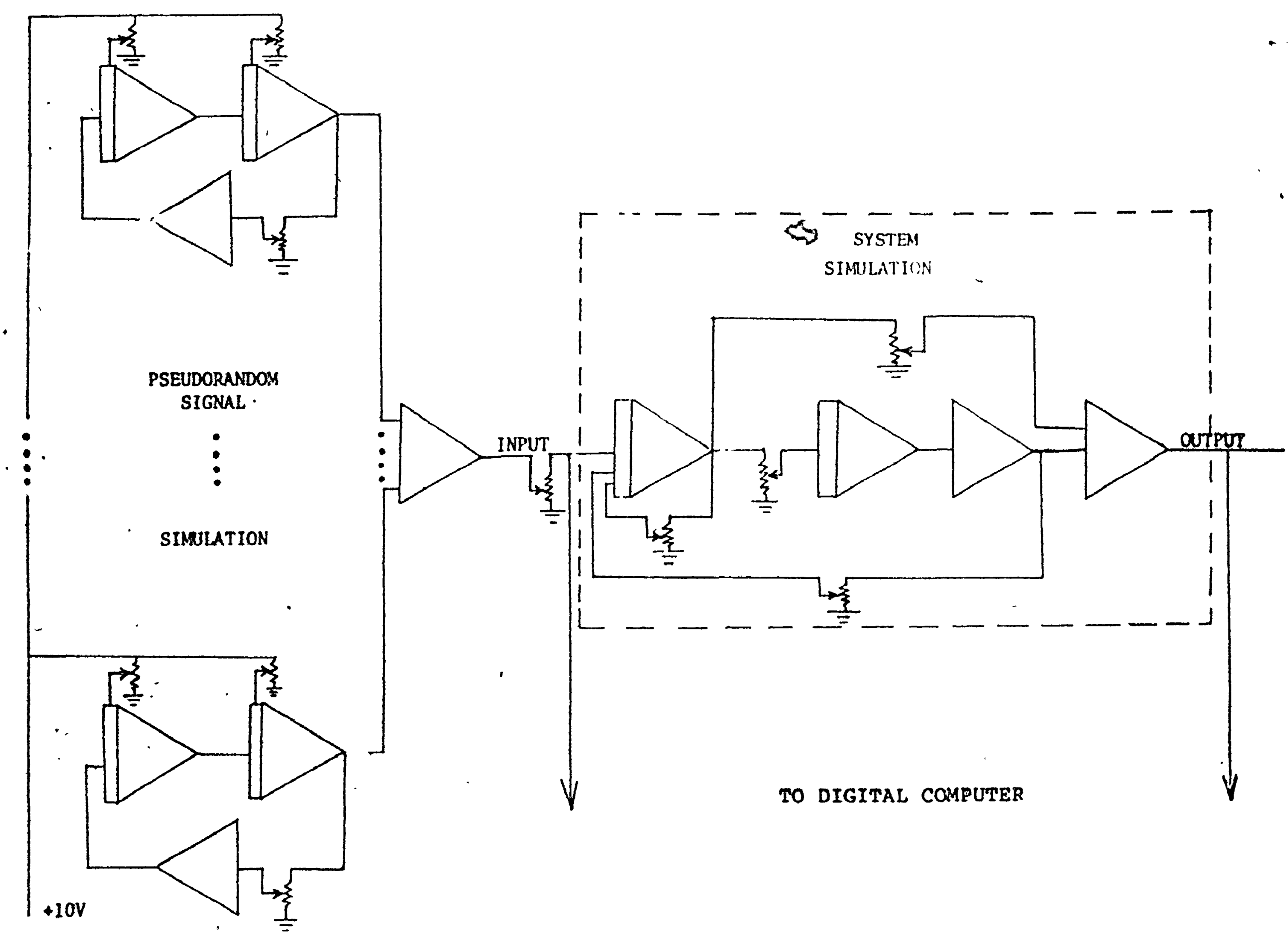


Figure 5.7 Analog Simulation

A repeatable input signal gives the same output signal every time. We can make comparisons of effects due to different noise levels. We can also compare different algorithms under essentially identical conditions. Should this input signal be truely random, we would have to make many runs of the same test in order to arrive at a statistical result.

5.6 Error Analysis

There are two main sources of errors in the system we have just described. Firstly, there are the random disturbances within the system that are completely unpredictable. This kind of disturbances are usually assumed to be white gaussian and are difficult to assess.

The second source of errors comes from the non-ideal components of the instruments used in the experiments. They are of systematic in nature. The following is a brief summary of these systematic errors:-

(A) Analog unit (non-ideal operational amplifiers)

(i) drift

(ii) zero off-set

(iii)phase shifts

(B) Multiplexer

(i) zero off-set

(ii) non-infinite backward resistance

(iii)non-zero forward resistance

(iv) cross-talk due to imperfect isolation between channels

(C) Sampler

(i) finite aperture time

(ii) time delay due to finite tracking time and switching

(iii) uncertainty in synchronization of simultaneous sampling

(iv) due to finite word length

Temperature dependence of many electronic components also may introduce errors. However, the electronic components available nowadays are quite reliable and a 0.01 percent full scale deviation can usually be obtained. This is not at all severe in our present application. Since a 10-bit converter is used, it is accurate up to about ±0.01 volt. Again, it will not cause severe degradation in the results.

CHAPTER SIX

SOFTWARE AND RESULTS

6.1 Introduction

This is a user-oriented program designed for convenient application to identify a second order one-input one-output system. It can easily be modified for higher order systems. The main bulk of the program is written in the MACRO-11 assembly language for the PDP-11/45 minicomputer. Compared with compiler language programs, the assembly language programs have the advantage of using less core memory and require less computation time since a lot of overheads can be eliminated. There is however one FORTRAN subroutine. This is being used for conducting the initial interactive dialog to obtain some essential data. It does not affect the efficiency of the identification algorithms because the dialog is being carried out at the beginning of the program before sampling is started. But it offers the convenience of flexible formats for the data to be read in from the keyboard.

As far as the size of the program is concerned, the assembly language portion of the program which consists of all computational and input/output aspects of both algorithms requires only 7632 words of core. The FORTRAN subroutine requires 1722 words of core. There is one common data block between the assembly language program and the FORTRAN subroutine occupying only 24 words of core. Modification of this program for higher order systems does not significantly change the numbers just quoted. If the output device is not fast enough to empty the output buffer for the intermediate estimates, storage area has to be provided for storing all these intermediate results. The size of this temporary storage depends on the number of samples we want to take. Two words of storage are necessary to store one parameter value for each iteration.

6.2 Organization of the Program

A flowchart of the program is shown in figure 6.1 giving a general picture of its organization. The program is loaded into core by the system monitor in the usual manner. When it starts to run, it will first jump to the subroutine that conducts the initial dialog with the user.

During the interactive dialog, the user is asked to select either one of two algorithms i.e. the algorithm using matrix pseudoinverse only and the algorithm with filter. In the case of the second algorithm, the user has to supply a gain factor for use in the iterative stochastic approximation formula. He also has to specify when the filtering should begin. Since a third order filter is used and thirteen previous error terms are needed for updating the filter parameters, the filtering can only start after at least fourteen samples have been taken. Should the user direct the filtering to start at a even later time, the program would still update the noise filter after fourteen samples so that a more accurate filter would be available at that time.

We can also specify the maximum number of samples to be taken. However, the user reserves the right to abort the program any time during run time. This is done simply by raising bit 0 in the switch register console on the computer. The program checks this bit every time it enters a service

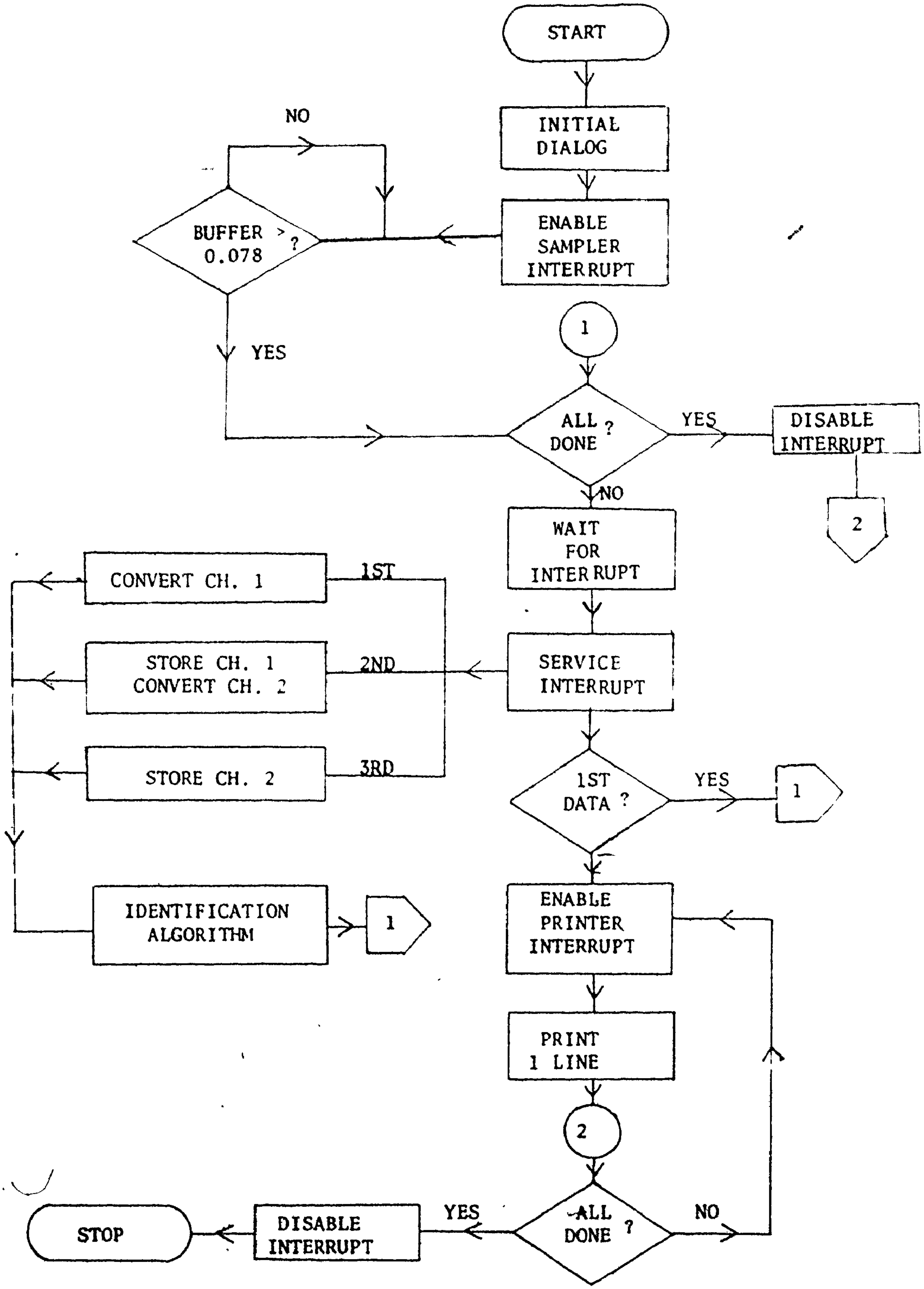


Figure 6.1 Flowchart of Program

subroutine. It would immediately clear all interrupts and then exit from the program to return to the system monitor. From thereon, the user can restart the program.

As we have noted in chapter five, the lowest sampling frequency available on the selector switch is one hertz. Here, we can enter an integer that will be used by the program as a multiplying factor of the sampling period. We can therefore greatly increase the number of choices in sampling frequency.

After completing the initial dialog, it will jump into a loop waiting for the analog computer to be switched on. The identification will start after the analog computer has been started.

Results of each iteration are printed out as soon as they are ready. A sample print-out of a typical run can be found on figure 6.2. Because of the different speeds of the many devices and the need to sample at equal intervals, priorities have been assigned to the different interrupt driven service subroutines. The sampling service subroutines have the highest priority (priority 5) because samples need to be taken at precise instances. The data acquired through sampling also need to be transferred into core from the data buffer before the next data comes in. By arranging the priority of the printing interrupt (priority 4) below that of the sampling and analysis but above that of the processor (priority 3), they can swap control of the computer until the prescribed maximum number of iterations has been reached without interfering with each other.

Figure 6.3 tabulates the functions of the major subroutines together

```
DO YOU WANT FILTERING ?
IF YES,TYPE 1;  IF NO,TYPE Ø

1
WHEN DO YOU WANT FILTERING TO START ?  [I3]

Ø5Ø
ENTER THE GAIN TERM FOR STOCHASTIC APPROXIMATION. [F5.1]

1.Ø
ENTER THE MULT. FACTOR FOR THE SAMPLING PERIOD.  [I2]

Ø1
ENTER THE MAX. NO. OF SAMPLES YOU WANT. [I3]

3ØØ


THANK YOU.  TO START,STRIKE ANY KEY
```

	PHI1	PHI2	PHI3	PHI4	PHI5
1	2.22E-Ø1	Ø.ØØE-ØØ	Ø.ØØE-ØØ	Ø.ØØE-ØØ	Ø.ØØE-ØØ
2	2.21E-Ø1	5.89E-Ø2	Ø.ØØE-ØØ	-1.33E ØØ	Ø.ØØE-ØØ
3	2.21E-Ø1	5.9ØE-Ø2	-1.61E-Ø1	-1.33E ØØ	4.48E-Ø1
4	2.21E-Ø1	5.9ØE-Ø2	-1.62E-Ø1	-1.33E ØØ	4.52E-Ø1
5	2.21E-Ø1	5.9ØE-Ø2	-1.61E-Ø1	-1.33E ØØ	4.51E-Ø1

Figure 6.2 Sample Print-out of the Results

Figure 6.3 Table of Subroutines

Subroutine Name	Functions and calling conventions
START	Main Program.
DIALOG	Obtains information from user through a series of questions.
ANAL	Dispatch subroutine for data analysis, stores the resulting estimations into buffers.
METHOD	Minimum norm and least squares algorithm for pseudoinvers.
STAPR	Stochastic approximation.
FILTER	Filtering of measurements.
CONVSN	Conversion of A/D outputs into floating point format. Calling convention: MOV DATA, R3 JSR R5, CONVSN Result is on top two words of the stack
CLOCK	Enable the sampling interrupt.
AD	Gives command to convert channel 1; interrupt driven.
STR1	Stores data of channel 1; gives command to convert channel 2, interrupt driven.
STR2	Stores data of channel 2; jumps to subroutine CONVSN: interrupt driven.
DELAY	Dispatch subroutine for analysis of data; stores resulting estimates in buffers.
DELAY	Modifies sampling period by an integer multiplicative factor $k \geq 1$.
PRINT	Enables the printing interrupt; monitors the progress of sampling, data analysis and printing.
IO	ASCII conversion of results before transferring to the printing buffers.
IOP	Generates a 3-digit ASCII coded line counter in ascending order.
PRN	Printing subroutine; interrupt driven.

Figure 6.3 cont'd.

Mathematical Subroutines	
MULFP ADDFP DIVFP SUBFP	Floating point multiplication, addition or subtraction of A and B. Calling convention: JSR R5, XXXFP .WORD A, B, C Result is in C
MMUL MADD MSUB	Matrix multiplication, addition or subtraction of A and B. Calling convention: JSR R5, MXXX .WORD A, B, C .WORD ROW, COL Result is in C.
DSC MSC	Matrix division or multiplication by a scaler. Calling conventions: JSR R5, XSC .WORD A, B, SC .WORD RWO, COL Result is in B.
MTRN	Matrix transposition. Calling convention: JSR R5, MTRN .WORD A, AT .WORD ROW, COL Result is in AT.
SHIFT	Shifts all elements of vector A by one position downwards. Calling convention: JSR R5, SHIFT .WORDS A, N

with their calling conventions whenever they are appropriate. This program is adaptable to computers without a hardwired floating point processor without making any changes. Macro definitions are liberally used in the mathematical subroutines to facilitate easy checking. The complete heavily commented program listing is in the appendix.

6.3 Results

The results are based on experiments identifying the second order one-input one-output system given by

$$H(s) = \frac{0.04+0.28s}{0.04+0.4s+s^2} \tag{6.1}$$

According to Sinha [21], the sampled-data eqnivalent of equation (5.1) can be obtained by the bilinear transformation

$$s = \frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})} \tag{6.2}$$

subject to the condition that

$$p_k T \leq 0.5 \tag{6.3}$$

where T = sampling period

p_k = system poles.

The system represented by equation (6.1) has poles

$$p_1 = p_2 = 0.2 \tag{6.4}$$

Therefore, the transformation given by (6.2) is valid if T=2 seconds is used.

Applying (6.2) to (6.1), we have

$$H(z) = \frac{0.222+0.0556z^{-1}-0.167z^{-2}}{1-1.333z^{-1}+0.444z^{-2}} \qquad (6.5)$$

However, due to the presence of a small amount of disturbance in the system even when no external noise is added, the experimental results of the coefficients of equation (5.5) are found to be

$$\phi^T = [0.221 \quad 0.059 \quad 0.162 \quad -1.33 \quad 0.452] \qquad (6.6)$$

Values in equation (6.6) are being used as a reference for subsequent comparisons.

Whenever it is appropriate, experimental results from both algorithms are presented together. Both algorithms are tested under different amount of externally introduced white noise into the output of the simulated system. For the second algorithm using combined matrix pseudoinverse and stochastic approximation, a third order noise filter has been used.

Figure 6.4 summarizes the estimates and the resulting normalized errors for both algorithms after three hundred iterations. We can conclude from these results that the error of estimation is directly dependent on the level of noise present. It is also observed that the second algorithm gives better estimates in all instances. The extent of improvement, however, varies.

Sample plots are shown in figures 6.5 and 6.6 displaying the behavious of the error in the estimates during the course of identification for the pseudoinverse algorithm and the one with filtering respectively. Note that in figure 6.6, filtering starts only after fifty iterations in order to avoid the initial region with large fluctuations in the estimation.

Despite the good convergence properties clearly exhibited, there always exists a bias in the estimation as expected because the residuals are correlated. Considerable amount of the bias is successfully removed by the filtering method.

It is of interest to recall that parameter estimation problems can be considered as optimization problems. In our case, the objective for minimization is the residual error sequence. This is our criterion to define the "goodness" of estimation. There are other criteria that can be used. For example, the time domain errors of step response or the impulse response are both valid criteria to evaluate estimations. Since there is no direct functional relationship among different criteria, a good estimation according to one criterion does not necessary imply that it is also good when judged by other criteria. A case in point to illustrate this fact is to compare the step response and impulse response of the tests we have performed.

The two plots figure 6.7 show the step response and impulse response respectively of the two algorithms at a noise ratio of 25%. From the table of figure 6.4, we notice the big different in the values of the parameters. We also notice that, the first algorithms has a normalised error of 88.72% compared with the much smaller 11.38% in the second algorithm. The difference of almost seven hundred percents is also show graphically in figure 6.6b. Despite all these dramatic differences, a reference to figure 6.7 shows that their responses are not too far apart considering what we have seen from the difference in parameter values. The table in figure 6.8 tabulates the corresponding mean square errors in step and impulse response of all the tests we have performed. There is no direct correlation between this set of values and the normalised errors. While filtering unfailingly reduces the parameter error, no such conclusion can be drawn about the errors in step response and impulse response.

As far as computation time is concerned, no more than 0.05 second per iteration is required for the matrix pseudoinverse algorithm. Twice as much time is required for the second algorithm i.e. 0.1 second.

	NOISE RATIO (%)	ESTIMATION $\hat{\phi}_1$	$\hat{\phi}_2$	$\hat{\phi}_3$	$\hat{\phi}_4$	$\hat{\phi}_5$	NORMALIZED ERROR
PSEUDOINVERSE	5	0.2188	0.2596	0.04155	-0.4156	-0.07847	0.5487
	10	0.2176	0.2928	0.07486	-0.2561	-0.1682	0.8035
	25	0.2137	0.3011	0.08354	-0.1934	-0.1879	0.8872
	50	0.2076	0.2992	0.08611	-0.1603	-0.1546	0.9044
	75	0.2025	0.2976	0.08969	-0.1305	-0.1081	0.9130
	100	0.1982	0.2961	0.09082	-0.1056	-0.0654	0.9199
WITH STOCHASTIC APPROX.	5	0.2126	0.1407	-0.07118	-0.9829	0.2575	0.08469
	10	0.1835	0.1252	-0.04103	-0.9497	0.2355	0.1035
	25	0.2167	0.1960	-0.06142	-0.9101	0.2858	0.1137
	50	0.1878	0.1088	-0.04592	-0.5546	-0.05058	0.4255
	75	0.1973	0.06805	0.03884	-0.5079	0.02452	0.4387
	100	0.1841	0.01295	0.02091	-0.4835	0.005042	0.4716

Figure 6.4 Results in Parameter Estimation and the Normalized Errors

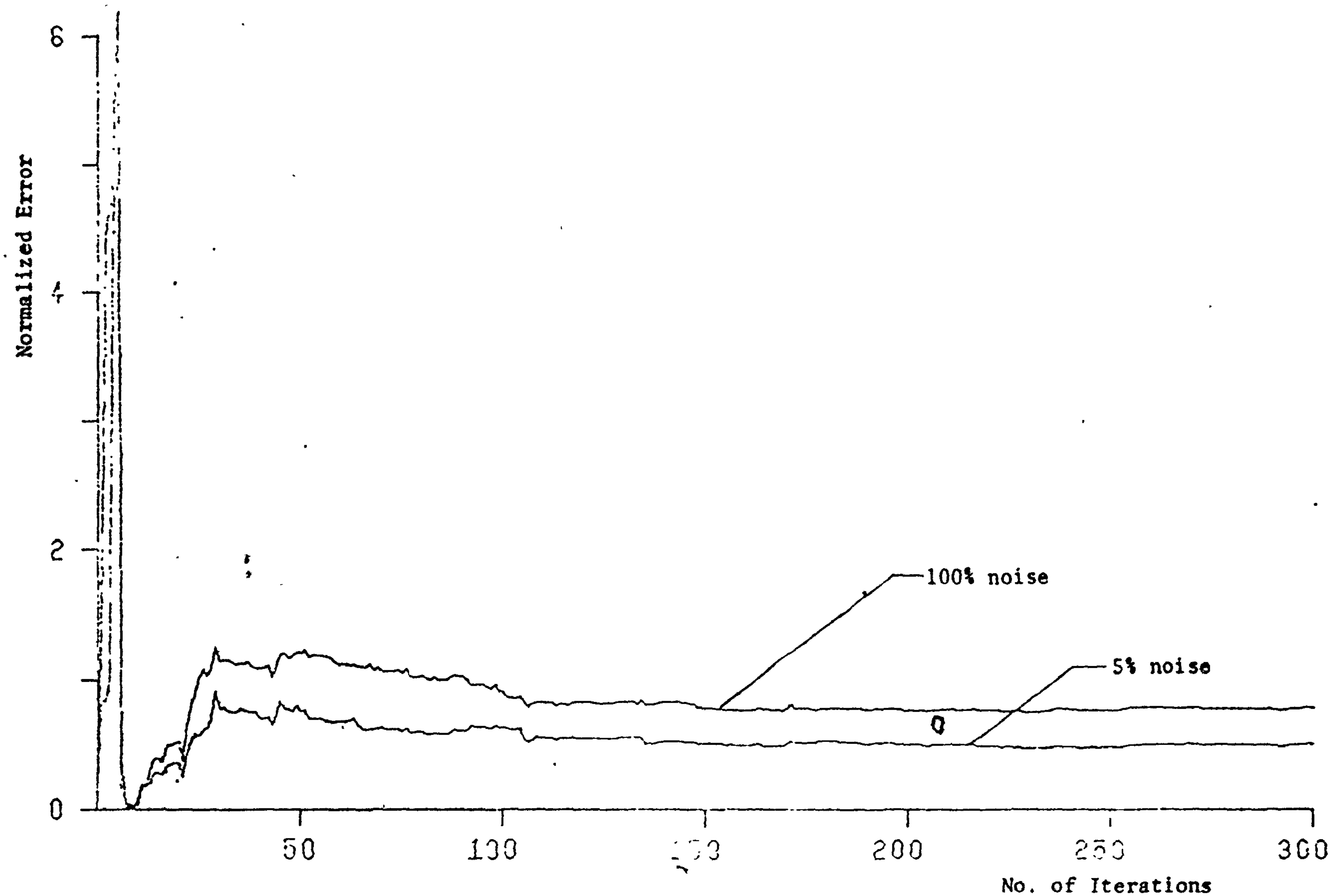


Figure 6.5 Pseudoinverse Algorithm

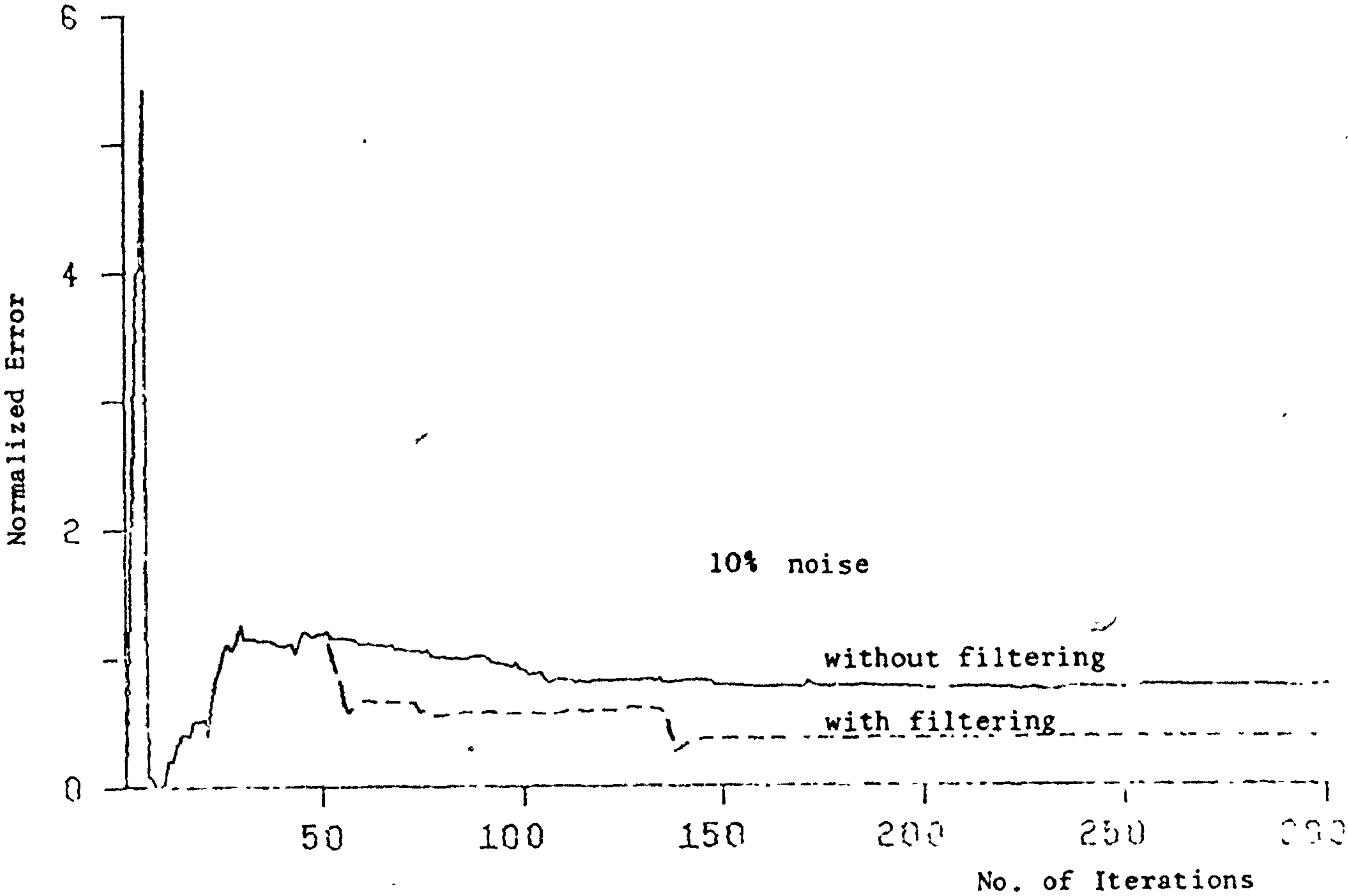
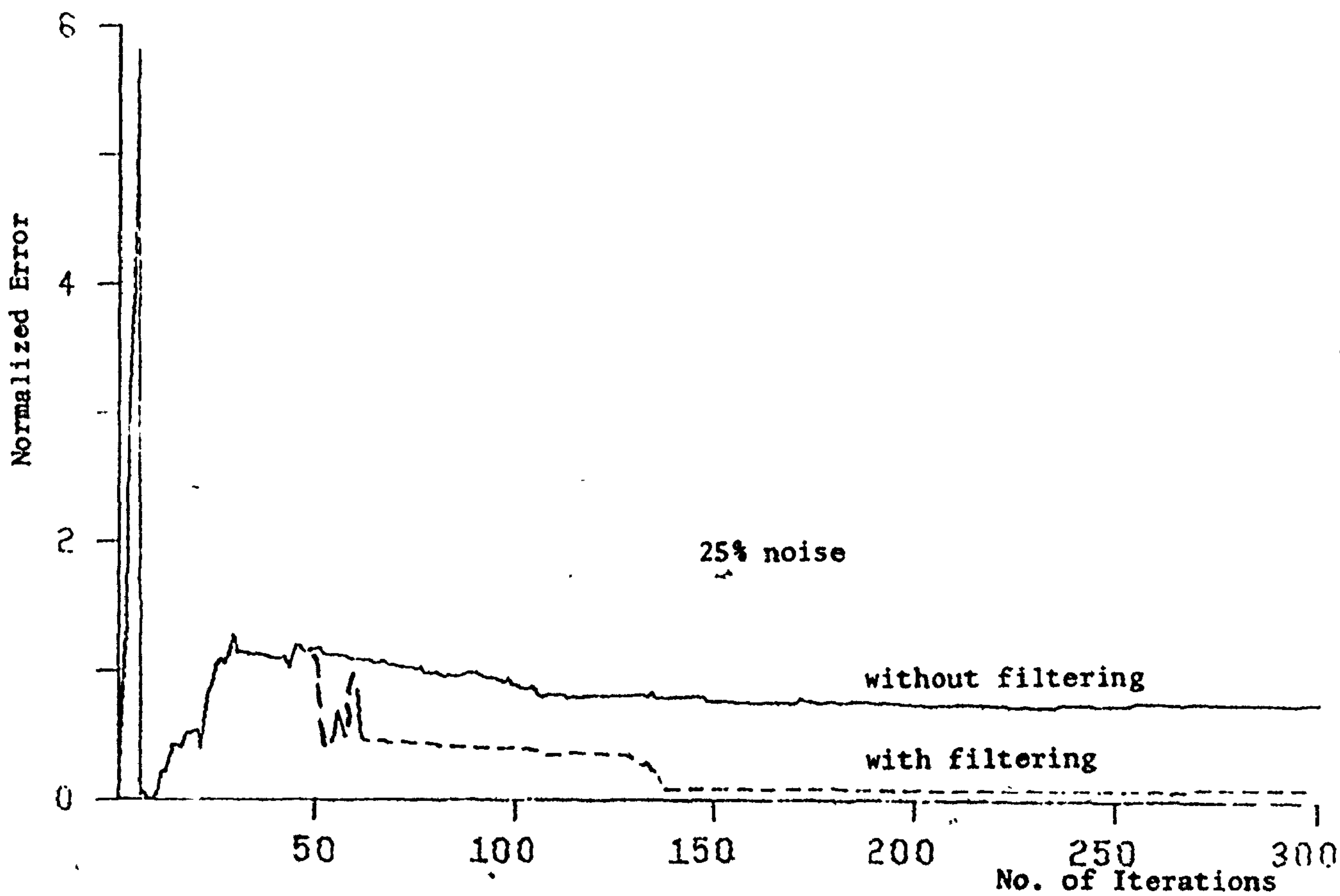


Figure 6.6a Stochastic Approx. Filtering



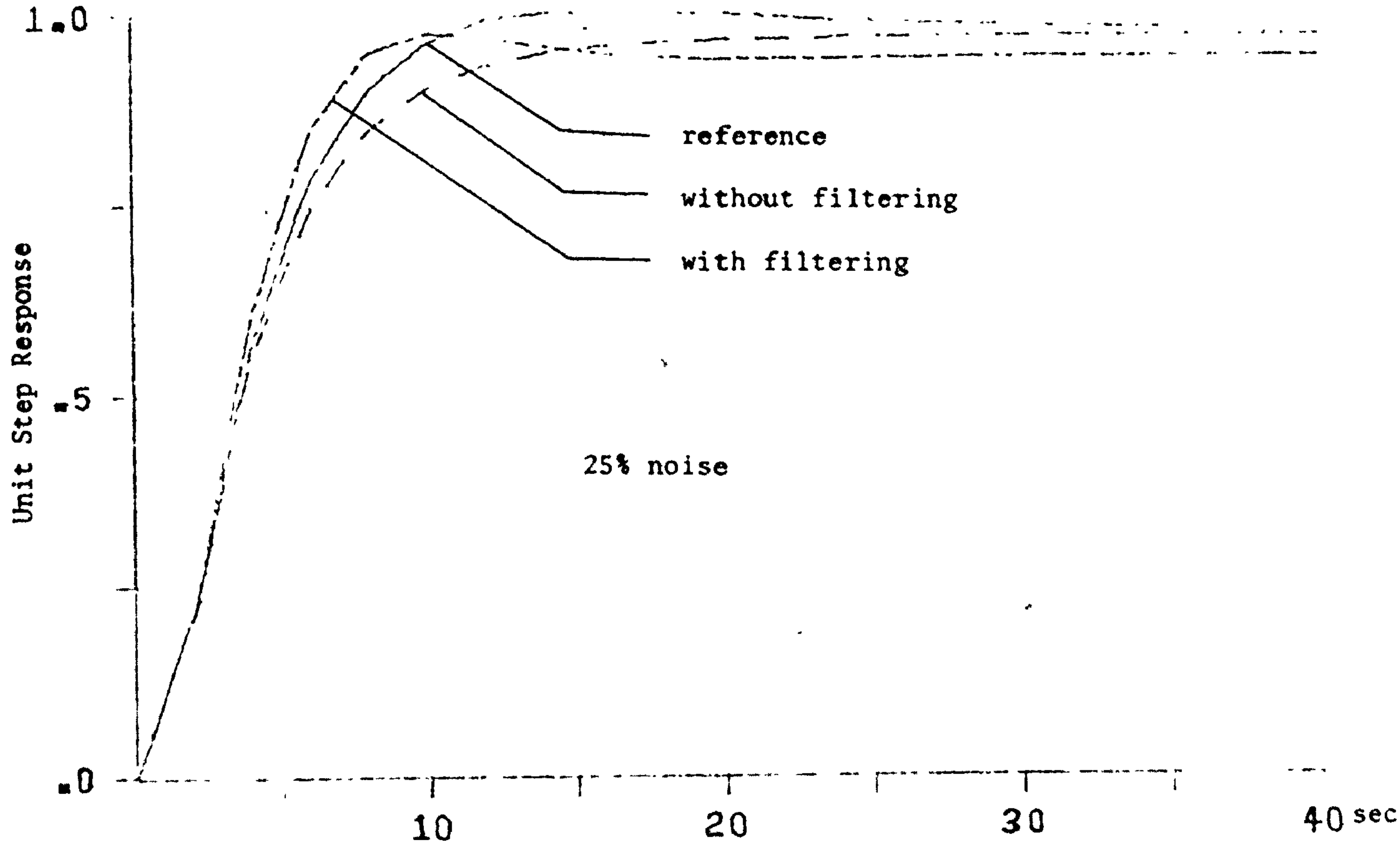


Figure 6.7a Comparison of Step Responses

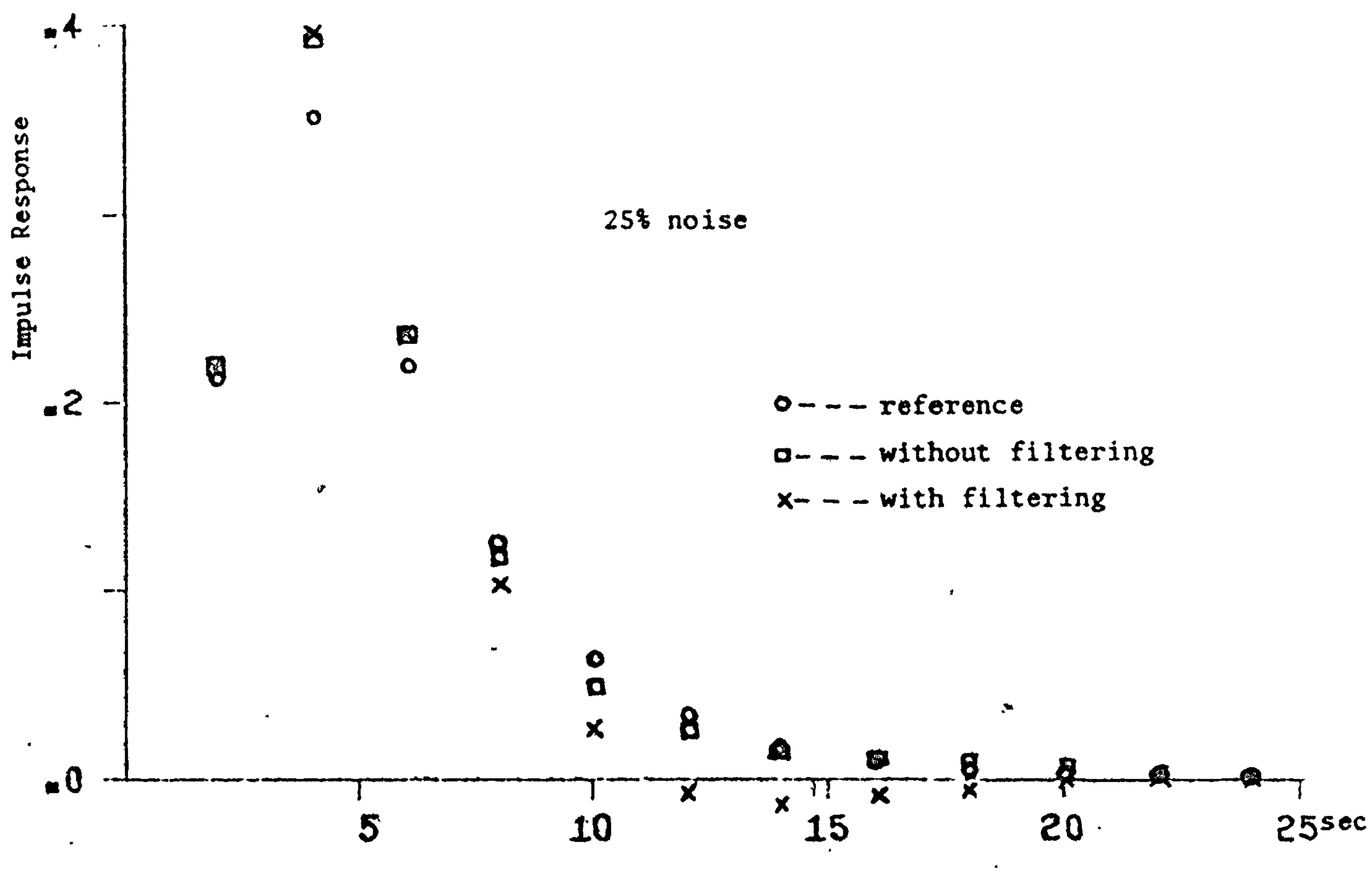


Figure 6.7b Comparison of Impulse Responses

	NOISE RATIO (%)	MEAN SQUARE ERROR: STEP RESPONSE ($x10^{-2}$)	MEAN SQUARE ERROR: IMPULSE RESPONSE ($x10^{-4}$)
PSEUDOINVERSE	5	0.1122	0.2367
	10	0.07293	0.2759
	25	0.1108	0.5449
	50	1.3490	2.1173
	75	3.8398	5.0463
	100	6.6374	8.4712
WITH STOCHASTIC APPROX.	5	0.1186	0.2367
	10	0.4288	0.2318
	25	0.2245	3.0333
	50	11.7404	20.811
	75	14.086	24.0609
	100	28.6127	46.859

Figure 6.8 Mean Square Errors in Step Responses and Impulse Responses

CHAPTER SEVEN

CONCLUSIONS

The theories presented in this thesis result in two algorithms for on-line system identification. The first one is the matrix pseudoinverse algorithm. A fundamental property of this method is that the estimates are optimal in the sense that the residual error is minimized.

The second algorithm is an extension of the first one aiming at further improving the estimation by removing the bias in the estimates due to measurement noise. The mechanism employed is the introduction of a filter obtained from the properties of the noise present in the system. The estimation of the noise properties and hence the construction of the filter is itself an on-line estimation process being carried out in parallel with the pseudoinverse method. The tool to this end is the use of stochastic approximation which is also computationally simple. As a result, incorporation of the stochastic approximation into the original scheme does not result in serious degradation in efficiency.

The upgrading of the program to accommodate higher order systems can be easily done by simply expanding the data block to provide enough room as a working area. All other instructions remain unchanged. However, high order models are usually not necessary in many engineering applications. Very complex systems can sometimes be approximated by second or third order models. For example, Sinha and Bereznai[22] have modelled the dynamics of the nuclear reactor power generating station in Douglas Point by a second order model. This second order model which is being updated on-line is subsequently used for controller design. This example may serve as a justification for choosing only a second order example in the present study. Also of importance as a reminder, simulation of the system on the analog computer bearsclose resemblance to the actual situations where identification might be employed. In the industrial setting, transducers are used to make such measurements as temperature and speed in terms of electrical voltages. These voltages are being sampled just as we have done in the present study.

We can see from the results that the convergence property of both algorithms are clearly and nicely exhibited even in high noise environment. The accuracy in estimation are quite naturally deteriorating with the increase in the amount of noise. Introduction of filtering, however, greatly enhances this accuracy at the expense of a relatively small amount of computation time.

However, there remains room for further improvement. A common problem in applying stochastic approximation is the difficulty in choosing an appropriate gain factor that would best suit a particular situation. No systematic method yet exists. It would be a worthwhile research area to devise an iterative scheme by which an optimal prediction of this gain factor can be done on-line together with what we have already developed. A point of caution is in order. We have to watch out for the amount of extra computation thereby introduced. A time consuming method would destroy the major appeal of our present approach which is computational simplicity.

APPENDIX

Program Listing

```
                      .TITLE IDENT
               .GLOBL DIALOG
               .GLOBL $ADR,$SBR,$MLR,$DVR,$POLSH
        000000 R0=%0
        000001 R1=%1
        000002 R2=%2
        000003 R3=%3
        000004 R4=%4
        000005 R5=%5
        000006 SP=%6
        000007 PC=%7
               ;
               .SBTTL MACRO DEFINITIONS
               ;
               .MACRO MMUL A,B,C,ROWA,COLA,COLB
                      JSR         R5,MMUL
                      .WORD       A,B,C
                      .WORD       ROWA,COLA,COLB
                       .ENDM
               .MACRO MADD A,B,C,ROW,COL
                      JSR         R5,MADD
                      .WORD       A,B,C
                      .WORD       ROW,COL
                      .ENDM
               .MACRO MSUB A,B,C,ROW,COL
                      JSR         R5,MSUB
                      .WORD       A,B,C
                      .WORD       ROW,COL
                      .ENDM
               .MACRO MSC A,B,SC,ROW,COL
                      JSR         R5,MSC
                      .WORD       A,B,SC
                      .WORD       ROW,COL
                      .ENDM
               .MACRO DSC A,B,SC,ROW,COL
                      JSR         R5,DSC
                      .WORD       A,B,SC
                      .WORD       ROW,COL
                      .ENDM
               .MACRO MTRN A,AT,ROW,COL
                      JSR         R5,MTRN
                      .WORD       A,AT
                      .WORD       ROW,COL
                      .ENDM
               .MACRO SAVE RA,RB,RC,RD,RE,RF
                      MOV         RA,-(SP)
                      .IIF DF RB,         MOV      RB,-(SP)
                      .IIF DF RC,         MOV      RC,-(SP)
                      .IIF DF RD,         MOV      RD,-(SP)
                      .IIF DF RE,         MOV      RE,-(SP)
                      .IIF DF RF,         MOV      RF,-(SP)
                      .ENDM
```

```
        .MACRO UNSAVE RA,RB,RC,RD,RE,RF
               .IIF DF RF,      MOV       (SP)+,RF
               .IIF DF RE,      MOV       (SP)+,RE
               .IIF DF RD,      MOV       (SP)+,RD
               .IIF DF RC,      MOV       (SP)+,RC
               .IIF DF RB,      MOV       (SP)+,RB
               MOV        (SP)+,RA
               .ENDM
        .MACRO POPF A
               MOV        (SP)+,A
               MOV        (SP)+,A+2
               .ENDM
        .MACRO PUSHF A
               MOV        A+2,-(SP)
               MOV        A,-(SP)
               .ENDM
        .MACRO FMUL A,B,C
               JSR        R5,MULFP
               .WORD      A,B,C
               .ENDM
        .MACRO FDIV A,B,C
               JSR        R5,DIVFP
               .WORD      A,B,C
               .ENDM
        .MACRO FADD A,B,C
               JSR        R5,ADDFP
               .WORD      A,B,C
               .ENDM
        .MACRO FSUB A,B,C
               JSR        R5,SUBFP
               .WORD      A,B,C
               .ENDM
        .MACRO MOVF A,B
               MOV        A,B
               MOV        A+2,B+2
               .ENDM
        .MACRO FSHIFT A,NUM
               JSR        R5,SHIFT
               .WORD      A,NUM
               .ENDM
```

```
                     .SBTTL MAIN PROGRAM
                     ;
                     ;
                     ;
                     ;        THIS IS THE MAIN PROGRAM
                     ;
                     ;
                     ;        IT FIRST JUMPS TO SUBROUTINE DIALOG TO
                     ;        OBTAIN THE FOLLOWING PARAMETERS :-
                     ;
                     ;        IFTR--  - TO SEE IF FILTERING IS DESIRED
                     ;        IFTRST-- IF SO,WHEN SHOULD IT BEGIN
                     ;        GAIN--   AND WHAT WOULD BE THE VALUE OF
                     ;         THE GAIN FACTOR FOR STOCH. APPROX.
                     ;        IDELAY-- A PARAMETER TO MODIFY THE SAMPLING
                     ;         FREQUENCY
                     ;        MAXAD--- MAX. NO. OF SAMPLES TO BE TAKEN
                     ;
                     ;        NEXT, SUBROUTINE CLOCK INITIATES THE
                     ;        SAMPLING PROCESS AND PROCEED WITH THE
                     ;        ALGORITHM SELECTED I.E.
                     ;        ALGORITHM 1 -- MATRIX PSEUDOINVERSE
                     ;        ALGORITHM 2 -- COMBINED PSEUDOINVERSE
                     ;               AND STOCHASTIC APPROX.
                     ;
                     ;        SUBROUTINE PRINT MONITORS THE PROGRESS OF
                     ;        THE PROGRAM AND PRINTS OUT THE ESTIMATES
                     ;        WHEN THEY ARE READY
                     ;
                     ;
                     ;
                     ;
00000 004567 START:   JSR     R5,DIALOG;      INITIAL DIALOGE
      003000G
00004 004567          JSR     R5,CLOCK;       STARTS SAMPLING
      003532
00010 000001          WAIT
00012 004567          JSR     R5,PRINT;       STARTS PRINTING
      004620
00016 000005          RESET
00020 000000          HALT
00022 104060 1$:      EMT     60
                     ;
                     ;
                     ;
      000000'.CSECT FTRN
                     ;
                     ;DATA OBTAINED FROM INITIALIZATION SUBROUTINE
                     ;
00000 000000 IFTR:    .WORD 0
00002 000000 IFTRST:  .WORD 0
00004 000000 IDELAY:  .WORD 0
00006 000000 MAXAD:   .WORD 0
00010        GAIN:    .BLKW 2.
```

```
        000024 '.CSECT
                .SBTTL PSEUDOINVERSE ALGORITHM
                ;
                ;
                ;
                ;       COMMON SECTION
                ;       CAL. ATPHI,Y-ATPHI,PA,ATPA,1+ATPA
                ;
000024          METHOD: MMUL    AT,PHI,DUM1,ONE,FIVE,ONE
00044                   FSUB    Y,DUM1,DUM1
00056                   MMUL    P,A,PA,FIVE,FIVE,ONE
00076                   MMUL    AT,PA,DUM2,ONE,FIVE,ONE
00116                   FADD    F.ONE,DUM2,DUM2
00130   026727          CMP     AN.K,#4;        >5 ITERATIONS ?
        004500
        000004
00136   003156          BGT     PSEUDO
                ;
                ;       MINIMUM NORM SECTION
                ;
00140                   MMUL    Q,A,QA,FIVE,FIVE,ONE
00160                   MMUL    AT,QA,DUM3,ONE,FIVE,ONE
00200                   DSC     QA,T5.1,DUM3,FIVE,ONE
                ;
                ;       UPDATE Q
                ;
00216                   MMUL    T5.1,QAT,T5.5,FIVE,ONE,FIVE
00236                   MSUB    Q,T5.5,Q,FIVE,FIVE
                ;
                ;       UPDATE PHI
                ;
00254                   MSC     T5.1,T5.1,DUM1,FIVE,ONE
00272                   MADD    PHI,T5.1,PHI,FIVE,ONE
                ;
                ;       UPDATE P
                ;
00310                   DSC     T5.5,T5.5,DUM3,FIVE,FIVE
00326                   MSC     T5.5,T5.5,DUM2,FIVE,FIVE
00344                   MADD    P,T5.5,P,FIVE,FIVE
00362                   MMUL    PA,QAT,PAQAT,FIVE,ONE,FIVE
00402                   MTRN    PAQAT,T5.5,FIVE,FIVE
00416                   MADD    PAQAT,T5.5,T5.5,FIVE,FIVE
00434                   DSC     T5.5,T5.5,DUM3,FIVE,FIVE
00452                   MSUB    P,T5.5,P,FIVE,FIVE
00470   000167          JMP     FINSH
        000126
```

```
                    ;
                    ;
                    ;        LEAST SQUARES SECTION (K>P)
                    ;
                    ;
                    ;        UPDATE P
                    ;
00474               PSEUDO: MMUL     PA,PAT,T5.5,FIVE,ONE,FIVE
00514                       DSC      T5.5,T5.5,DUM2,FIVE,FIVE
00532                       MSUB     P,T5.5,P,FIVE,FIVE
                    ;
                    ;        UPDATE PHI
                    ;
00550                       MSC      PA,T5.1,DUM1,FIVE,ONE
00566                       DSC      T5.1,T5.1,DUM2,FIVE,ONE
00604                       MADD     PHI,T5.1,PHI,FIVE,ONE
                    ;
                    ;        ALGORITHM 1 OR 2
                    ;
00622 022767 FINSH:  CMP      #0,IFTR;        NEED FILTERING ?
      000000
      000000
00632 001402         BEQ      1$;             NO,SKIP
00632 004567         JSR      R5,STAPR;  STOCHASTIC APPRO.ALGO.
      001576
00636 000205 1$:     RTS      R5
```

```
                       .SBTTL SUBROUTINE UPDATE
                       ;
                       ;        UPDATE--SUBROUTINE TO UPDATE THE
                       ;        INFORM. MATRIX AS NEW SAMPLES COME IN
                       ;        IT ALSO JUMPS TO FILTERING IF NEEDED
                       ;
                       ;
000640                 UPDATE:
000640 026767                   CMP     IFTRST,AN.K;    START FILTERING ?
       000002
       003766
00646  002002                   BGE     21$;            NO,SKIP
00650  004567                   JSR     R5,FILTER;       YES
       002574
00654                  21$:      FSHIFT A,N3;           UPDATING OF
00664                           FSHIFT  A+14,N2;        INFORMATION
00674                           MOVF    U,A;            MATRIX
00710                           MOVF    YOLD,A+14;       HERE
00724                           MOVF    Y,YOLD
00740  022767                   CMP     #0,YOLD;        CHECK IF
       000000
       001026
00746  001005                   BNE     10$;            YOLD=0
00750  022767                   CMP     #0,YOLD+2
       000000
       001020
00756  001413                   BEQ     2$
00760  000407                   BR      1$
00762  005767          10$:      TST    YOLD
       001006
00766  100004                   BPL     1$
00770  042767                   BIC     #100000,YOLD
       100000
       000776
00776  000403                   BR      2$
01000  052767          1$:       BIS    #100000,YOLD
       100000
       000766
01006  000205          2$:       RTS    R5
                       ;
                       ;
                       ;        DATA BLOCK FOR BOTH ALGORITHMS
                       ;
                       ;
01010                  A:
01010                  AT:      .BLKW 10.
01034                  P:       .BLKW 50.
01200                  PA:
01200                  PAT:     .BLKW 10.
```

```
01224 040200 Q:       .FLT2 1.E0
01226 000000
                     .BLKW 10.
01254 040200          .FLT2 1.E0
01256 000000
                     .BLKW 10.
01304 040200          .FLT2 1.E0
01306 000000
                     .BLKW 10.
01334 040200          .FLT2 1.E0
01336 000000
                     .BLKW 10.
01364 040200          .FLT2 1.E0
01366 000000
01370        QA:
01370        QAT:     .BLKW 10.
01414        PHI:     .BLKW 10.
01440        PAQAT:   .BLKW 50.
01604        T5.5:    .BLKW 50.
01750        T5.1:    .BLKW 10.
01774        YOLD:    .BLKW 2.
02000        DUM1:    .BLKW 2.
02004        DUM2:    .BLKW 2.
02010        DUM3:    .BLKW 2.
02014 000001 ONE:     .WORD 1
02016 000002 TWO:     .WORD 2
02020 000003 THREE:   .WORD 3
02022 000005 FIVE:    .WORD 5
02024 000002 N2:      .WORD 2
02026 000003 N3:      .WORD 3
02030 000012 N10:     .WORD 10.
02032 000014 N12:     .WORD 12.
02034 000015 N13:     .WORD 13.
02036 000017 N15:     .WORD 15.
02040 041710 N100:    .FLT2 100.
02042 000000
02044        INDEX:   .BLKW 2.
02050        COUNT:   .BLKW 2.
02054        F:       .BLKW 6.
02070        FU:      .BLKW 24.
02150        FUU:     .BLKW 6.
02164        FY:      .BLKW 24.
02244        FYY:     .BLKW 6.
02260        EV:      .BLKW 20.
02330        EVV:     .BLKW 6.
02344        EV2:     .BLKW 20.
02414        EV22:    .BLKW 6.
02430 040200 F.ONE:   .FLT2 1.E0
02432 000000
```

```
                    .SBTTL STOCHASTIC APPROXIMATION ALGORITHM
                    ;
                    ;
                    ;         STAPR-- THIS SUBROUTINE UPDATES THE
                    ;         PARAMETERS OF THE NOISE FILTER VECTOR
                    ;         F(3) BY ITERATING TEN TIMES PER PASS
                    ; FU=VECTOR WITH 15 PREVIOUS VALUES OF U
                    ; FY=VECTOR WITH 15 PREVIOUS VALUES OF Y
                    ; FUU=VECTOR WITH 3 PREVIOUS VALUES OF U
                    ; FYY=VECTOR WITH 3 PREVIOUS VALUES OF Y
                    ;
                    ;
02434               STAPR:  FADD    COUNT,F.ONE,COUNT;    K=K+1
02446                       FSHIFT  FU,N15;               UPDATE FU & FY
02456                       FSHIFT  FY,N15;               EACH STORING
02466                       MOVF    U,FU;                 15 PREVIOUS
02502                       MOVF    Y,FY;                 U & Y VALUES RESP.
02516 022767                CMP     #17,AN.K;              > 15 SAMPLES ?
      000017
      002110
02524 002402                BLT     1$;                   YES
02526 000167                JMP     STA.GO;               NO,SKIP
      000714
                    ;
                    ; CAL. ERROR VECTOR [EV] & [EV]**2
                    ;
02532 016700        1$:     MOV     N13,R0;
      177276
02536 016701                MOV     N12,R1
      177270
02542 006301                ASL     R1
02544 006301                ASL     R1
02546               STA.E:  MMUL    FUU,PHI,DUM1,ONE,THREE,ONE
02566                       MMUL    FYY+4,PHI+14,DUM2,ONE,TWO,ONE
02606                       FSUB    FYY,DUM1,DUM1
02620                       FADD    DUM1,DUM2,DUM1
02632 016761                MOV     DUM1,EV(R1)
      177142
      002260'
02640 016761                MOV     DUM1+2,EV+2(R1)
      177136
      002262'
02646                       FMUL    DUM1,DUM1,DUM1
02660 016761                MOV     DUM1,EV2(R1)
      177114
      002344'
02666 016761                MOV     DUM1+2,EV2+2(R1)
      177110
      002346'
02674 024141                CMP     -(R1),-(R1)
02676                       PUSHF   FUU+10
```

```
02706                  PUSHF     FYY+10
02716                  FSHIFT    FU,N15
02726                  FSHIFT    FY,N15
02736                  POPF      FU
02746                  POPF      FY
02756 005300           DEC       R0
02760 020027           CMP       R0,#0
      000000
02764 001402           BEQ       11$
02766 000167           JMP       STA.E
      177554
                ;
                ;RESTORE FU,FY
                ;
02772           11$:    PUSHF    FUU+10
03002                  PUSHF     FYY+10
03012                  FSHIFT    FU,N15
03022                  FSHIFT    FY,N15
03032                  POPF      FU
03042                  POPF      FU
03052                  FSHIFT    FU,N15
03062                  FSHIFT    FY,N15
                ;
                ;KWANTY FORMULA FOR STOCHASTIC APPROX.
                ;
03072 016700           MOV       N10,R0
      176732
03076           STA.F:  FADD     INDEX,F.ONE,INDEX
03110                  FADD      INDEX,COUNT,DUM1
03122                  FDIV      GAIN,DUM1,DUM1
03134                  MMUL      EVV,F,DUM2,ONE,THREE,ONE
03154                  FADD      EVV-4,DUM2,DUM2
03166                  FADD      EV22,EV22+4,DUM3
03200                  FADD      EV22+10,DUM3,DUM3
03212                  FDIV      DUM2,DUM3,DUM2
03224                  FMUL      DUM1,DUM2,DUM1
03236                  MOVF      DUM1,DUM2
03252 005767           TST       DUM2;          IF DUM1>100.
      176524
03256 100003           BPL       1$;            SKIP
03260 042767           BIC       #100000,DUM2
      100000
      176516
03266           1$:     FSUB     N100,DUM2,DUM2
03300 005767           TST       DUM2
      176500
03304 100436           BMI       STA.A
```

```
                            ;
                            ;NEW (F) CALCULATED HERE
                            ;
   03306                            FMUL      DUM1,EVV+10,DUM3
   03320                            FMUL      DUM1,EVV+4,DUM2
   03332                            FMUL      DUM1,EVV,DUM1
   03344                            FSUB      F,DUM1,F
   03356                            FSUB      F+4,DUM2,F+4
   03370                            FSUB      F+10,DUM3,F+10
   03402                    STA.A:  FSHIFT    EV,N13
   03412                            FSHIFT    EV2,N13
   03422 005300                     DEC       R0
   03424 020027                     CMP       R0,#0
         000000
   03430 001402                     BEQ       11$
   03432 000167                     JMP       STA.F
         177440
   03436 005067             11$:    CLR       INDEX
         176402
   03442 005067                     CLR       INDEX+2
         176400
   03446 000205             STA.GO: RTS       R5
                            ;
                            ;SUBROUTINE FILTER FITERS INCOMING DATA
                            ;
   3450                     FILTER: SAVE      R5
   3452                             MMUL      F,FU,DUM1,ONE,THREE,ONE
   3472                             MMUL      F,FY,DUM2,ONE,THREE,ONE
   3512                             FADD      U,DUM1,U
   3524                             FADD      Y,DUM2,Y
   3536                             UNSAVE R5
   3540 000205                      RTS       R5
```

```
                     .SBTTL SAMPLING
                     ;
                     ;
              177564 TPS=177564; PUNCH STATUS REG
              177566 TPB=177566; PUNCH BUFFER REG
              177520 ADCSR=177520;   ADDR.OF CONTROL & STATUS REG
              177522 CHNSLR=177522;  ADDR. OF CHANNEL SELECTOR REG
              177524 DATREG=177524;  ADDR. OF DATA BUFFER REG.
                     ;
                     ;CLOCK---SUBROUTINE TO START SAMPLING PROCESS
                     ;
       03542 005037  CLOCK:  CLR     @#TPS;  INITIALIZE SAMPLING & PUNCH
             177564
       03546 005037          CLR     @#ADCSR;  CONTROL &STATUS REGS.
             177520
       03552 012737          MOV     #AD,@#110;  SET UP INT. VECTOR
             003632'
             000110
       03560 012737          MOV     #240,@#112;  FOR SAMPLING; PRTY.=5
             000240
             000112
       03566 013767          MOV     @#64,C.V1;  SAVE OLD CONTENTS OF
             000064
             000032
       03574 013767          MOV     @#66,C.V1+2;  PRINTER INTR. VECTOR
             000066
             000026
       03602 012737          MOV     #IO,@#64;  INTR. VECTOR FOR
             004772'
             000064
       03610 012737          MOV     #4200,@#66;  PRINTING;PRTY 4
             004200
             000066
                                     ;  REG. SET NO. 2
       03616 012737          MOV     #100,@#ADCSR;   HERE WE GO BABY
             000100
             177520
       03624 000205          RTS     R5
       03626         C.V1:   .BLKW 2.
                     ;
                     ;
                     ;AD-----SERVICE ROUTINE FOR THE FIRST A/D
                     ;       INTERUPT TO CONVERT CH. 1
                     ;
       03632 000235  AD:     SPL     5;  MASK OUT PRINTING INTERUPT
       03634 032737          BIT     #1,@#177570;     ABORT PROGRAM ?
             000001
             177570
       03642 001405          BEQ     2$;              NO,SKIP
       03644 005037          CLR     @#ADCSR;         YES,QUIT
             177520
       03650 005267          INC     QUIT;            SET FLAG
             000050
```

```
03654 000002            RTI
03656 012737 2$:         MOV       #10000,@#CHNSLR;   CONVERT CH.1
      010000
      177522
03664 012737            MOV       #STR1,@#110;      SERVE STR1 NEXT
      003732'
      000110
03672 022767            CMP       #0,AD.CHK;        CHECK PT. CLR ?
      000000
      000026
03700 001010            BNE       1$;              NO,SKIP
03702 005267            INC       AD.A;    INC COUNT OF SAMPLES
      000022
03706 026767            CMP       AD.A,MAXAD;       =MAX. COUNT?
      000016
      000006'
03714 003402            BLE       1$;              NO,SKIP
03716 005037            CLR       @#ADCSR;         YES,FINISH
      177520
03722 000002 1$:         RTI
03724 000000 QUIT:      .WORD 0
03726 000001 AD.CHK:    .WORD 1
03730 000000 AD.A:      .WORD 0
                     ;
                     ;STR1--- SERVICE ROUTINE FOR THE 2ND A/D INTERUPT
                     ;        TO STORE CH.1 DATA  & CONVERT CH.2
                     ;
03732 000235 STR1:      SPL       5;  MASK OUT PRINTING INTERPT
03734 013767            MOV       @#DATREG,BUF1;  STORE CH.1 DATA
      177524
      000262
03742 032737            BIT       #1,@#177570;     ABORT PROGRAM ?
      000001
      177570
03750 001405            BEQ       4$;              NO,SKIP
03752 005037            CLR       @#ADCSR;         YES,QUIT
      177520
03756 005267            INC       QUIT;            SET FLAG
      177742
03762 000002            RTI
03764 022767 4$:         CMP       #0,AD.CHK;        CHECK PT. CLR?
      000000
      177734
03772 001416            BEQ       1$;              YES,SKIP
03774 012737            MOV       #AD,@#110;        NO
      003632'
      000110
04002 016700            MOV       BUF1,R0;         TEST ABS(BUF1)
      000216
04006 005700            TST       R0
04010 100001            BPL       2$
04012 005400            NEG       R0
```

```
04014 020027 2$:     CMP     R0,#7;            >0.078 VOLT ?
      000007
04020 101402         BLOS    3$;               NO, SKIP
04022 005067         CLR     AD.CHK;           CLR CHECK PT.
      177700
04026 000002 3$:     RTI
04030 012737 1$:     MOV     #20000,@#CHNSLR;  CONVERT CH.2
      020000
      177522
04036 012737         MOV     #STR2,@#110;       SERVE STR2 NEXT
      024046
      000110
04044 000002         RTI
                     ;
                     ;STR2----SERVICE ROUTINE TO STORE CH.2 DATA
                     ;        AND PROCESS THEM IN THE SELECTED METHOD
                     ;
04046 000235 STR2:   SPL     5;  MASK OUT PRINTING INTERUPT
04050 013767         MOV     @#DATREG,BUF2;  STORE CH.2 DATA
      177524
      000150
04056 032737         BIT     #1,@#177570;      ABORT PROGRAM ?
      000001
      177570
04064 001405         BEQ     1$;               NO,SKIP
04066 005037         CLR     @#ADCSR;          YES,QUIT
      177520
04072 005267         INC     QUIT;             SET FLAG
      177626
04076 000002         RTI
04100 022767 1$:     CMP     #0,IDELAY;         NEED DELAY ?
      000000
      000004
04106 001004         BNE     2$
04110 012737         MOV     #AD,@#110;         NO,SERVE AD NEXT
      003632
      000110
04116 000403         BR      3$
04120 012737 2$:     MOV     #DELAY,@#110;    YES,SERVE DELAY NEXT
      004232
      000110
04126 022767 3$:     CMP     #0,ANFLG;  FINISH LAST ANALYSIS ?
      000000
      000074
04134 001402         BEQ     4$
04136 000236         SPL     6
04140 104060         EMT     60;               IF NO,ERROR EXIT
04142 005267 4$:     INC     ANFLG;            SET FLAG
      000062
04146 016703         MOV     BUF1,R3;          CONVERT CH.1
      000052
04152 004567         JSR     R5,CONVSN;         TO FLT.PT.
      000132
```

```
04156 012667          MOV       (SF)+,U;          STORE IN U
      000032
04162 012667          MOV       (SP)+,U+2
      000030
04166 016703          MOV       BUF2,R3;          CONVERT CH.2
      000034
04172 004567          JSR       R5,CONVSN;         TO FLT. PT.
      000112
04176 012667          MOV       (SP)+,Y;          STORE IN Y
      000016
 4202 012667          MOV       (SP)+,Y+2
      000014
 4206 004567          JSR       R5,ANAL;          GO TO PROCESS DATA
```

```
IDENT   MACRO VR05A 01-JAN-72 05:23 PAGE 6+
SAMPLING

      000270
 4212 000002          RTI
               ;
 4214          U:      .BLKW 2.
 4220          Y:      .BLKW 2.
 4224 000000 BUF1:     .WORD 0
 4226 000000 BUF2:     .WORD 0
 4230 000000 ANFLG:    .WORD 0
               ;
               ;DELAY----SERVICE ROUTINE AS A DELAY LOOP
               ;         TO MODIFY THE SAMPLING FREQ.
               ;
 4232 000235 DELAY:    SPL      5;  MASK OUT PRINTING INTERUPT
 4234 032737          BIT       #1,@#177570;       ABORT PROGRAM ?
      000001
      177570
 4242 001405          BEQ       2$;               NO,SKIP
 4244 005037          CLR       @#ADCSR;          YES,QUIT
      177520
 4250 005267          INC       QUIT;             SET FLAG
      177450
 4254 000002          RTI
 4256 005267 2$:       INC      DE.D;             INC DELAY COUNT
      000024
 4262 026767          CMP       DE.D,IDELAY;       ENOUGH ?
      000020
      000004
 4270 002405          BLT       1$;               NO HURRY, SON
 4272 012737           MOV      #AD,@#110;         BACK TO CH.1
      003632
      000110
 4300 005067          CLR       DE.D;             RESET DE.D
      000002
 4304 000002 1$:       RTI
 4306 000000 DE.D:     .WORD 0
```

```
                      .SBTTL DECODING OF OUTPUT OF A/D CONVERTER
                      ;
                      ;       SUBROUTINE TO CONVERT A BINARY CODED OUTPUT
                      ;       FROM A/D CONVERTER INTO NORMALISED
                      ;       FLOATING POINT FORMAT
                      ;
4310 012667 CONVSN: MOV       (SP)+,TMPSP;      SAVE STACK
     000154
4314 005703         TST       R3;              POSITIVE?
4316 100004         BPL       1$;              YES,SKIP
4320 012767         MOV       #100000,SIGN;     SET SIGN BIT
     100000
     000152
4326 005403         NEG       R3;              GET 2'S COMPLIMENT
4330 010301 1$:     MOV       R3,R1;           SAVE ON R1
4332 022703         CMP       #0,R3;           ZERO ?
     000000
4336 001436         BEQ       3$
4340 006303         ASL       R3;
4342 006303         ASL       R3;
4344 060103         ADD       R1,R3;           *5
4346 006303 2$:     ASL       R3;              SHIFT LEFT
4350 105367         DECB      EXP;             DECREMENT EXPONENT
     000120
4354 005703         TST       R3;              MSB SET ?
4356 100373         BPL       2$;              NO,GO BACK
4360 042703         BIC       #100000,R3;       CLEAR MSB
     100000
4364 110367         MOVB      R3,TEMBUF+3;      STORE 2ND WORD
     000105
4370 016746         MOV       TEMBUF+2,-(SP); PUSH ON STACK
     000100
4374 000303         SWAB      R3;              GET 2ND BYTE
4376 110367         MOVB      R3,TEMBUF;        STORE MANTISA
     000070
4402 000367         SWAB      EXP;
     000070
4406 006067         ROR       EXP;             ALIGN EXPONENT BYTE
     000064
4412 056767         BIS       EXP,TEMBUF;       COPY EXPONENT
     000060
     000052
4420 056767         BIS       SIGN,TEMBUF;      COPY SIGN
     000054
     000044
4426 016746         MOV       TEMBUF,-(SP);     PUSH 1ST WORD
     000040
4432 000402         BR        4$;              ON STACK
```

```
4434 005046 3$:     CLR     -(SP);          FLOATING ZERO
4436 005046         CLR     -(SP)
4440 005067 4$:     CLR     TEMBUF;         READY TO QUIT
     000026
4444 005067         CLR     TEMBUF+2
     000024
4450 005067         CLR     SIGN
     000024
4454 012767         MOV     #000210,EXP;    RESTORE INITIAL EXP
     000210
     000014
4462 016746         MOV     TMPSP,-(SP);    RESTORE STACK
     000002
4466 000205         RTS     R5
4470 000000 TMPSP:  .WORD 0
4472        TEMBUF: .BLKW 2.
4476    210 EXP:    .BYTE 210,0
4477    000
4500 000000 SIGN:   .WORD 0
```

```
                       .SBTTL DATA ANALYSIS
                       ;
                       ;ANAL---SUBROUTINE TO JUMP TO IDENT. ALGORITHM
                       ;       AND STORE RESULTS IN DATA BUFFER
                       ;
                       ;
004502 004567 ANAL:    JSR     R5,UPDATE;  UPDATE INFORM. MATRIX
       174132
004506 004567          JSR     R5,METHOD;      GO TO ALGORITHM
       173312
004512 016700          MOV     AN.K1,R0;       LOAD POINTER
       000114
04516 016760           MOV     PHI,FI1(R0);    STORE RESULTS
       174672
       000000'
04524 016760           MOV     PHI+2,FI1+2(R0)
       174666
       000002'
04532 016760           MOV     PHI+4,FI2(R0)
       174662
       003720'
04540 016760           MOV     PHI+6,FI2+2(R0)
       174656
       003722'
04546 016760           MOV     PHI+10,FI3(R0)
       174652
       007640'
04554 016760           MOV     PHI+12,FI3+2(R0)
       174646
       007642'
04562 016760           MOV     PHI+14,FI4(R0)
       174642
       013560'
04570 016760           MOV     PHI+16,FI4+2(R0)
       174636
       013562'
04576 016760           MOV     PHI+20,FI5(R0)
       174632
       017500'
04604 016760           MOV     PHI+22,FI5+2(R0)
       174626
       017502'
04612 022020           CMP     (R0)+,(R0)+;    INC R0 BY 4
04614 010067           MOV     R0,AN.K1;       STORE POINTER
       000012
04620 005267           INC     AN.K;   INC. ITERATION COUNT
       000010
04624 005067           CLR     ANFLG;          CLR FLAG
       177400
04630 000205           RTS     R5
04632 000000 AN.K1:    .WORD 0
04634 000000 AN.K:     .WORD 0
```

```
                      .SBTTL SUBROUTINES FOR PRINTING
                      ;
                      ;PRINT-----SUBROUTINE TO MONITOR PROGRESS OF
                      ;          ANALYSIS AND PRINTING
                      ;
 004636 032737 PRINT:  BIT     #1,@#177570;    ABORT PROGRAM ?
        000001
        177570
 004644 001401         BEQ     2$;            NO,GO AHEAD
 004646 000426         BR      5$;            YES,QUIT
 004650 026767 2$:     CMP     IO.K,AN.K;  IS PRINTING LAGGING ?
        000114
        177756
 04656 002402          BLT     3$;            IF YES, GO AHEAD
 04660 000001          WAIT;          IF NO,WAIT AND
 04662 000765          BR      PRINT;         CHECK AGAIN, SON
 04664 012737 3$:      MOV     #100,@#TPS; PRINT NEXT CHARACTER
        000100
        177564
 04672 000001 4$:      WAIT
 04674 022767          CMP     #0,PR.FI;       FINISH 1 LINE ?
        000000
        000064
 04702 001373          BNE     4$;            NO,CHECK AGAIN
 04704 005267          INC     IO.K;          INC I/O COUNT
        000060
 04710 005267          INC     PR.FI;         RESET FLAG
        000052
 04714 026767          CMP     IO.K,MAXAD;     ALL DONE ?
        000050
        000006
 04722 002745          BLT     PRINT;         NO,CARRY ON SON
 04724 005037 5$:      CLR     @#ADCSR
        177520
 04730 012700          MOV     #77777,R0
        077777
 04734 016737          MOV     C.V1,@#64;  RESTORE INT. VECTOR
        176666
        000064
 04742 016737          MOV     C.V1+2,@#66
        176662
        000066
 04750 012737          MOV     #100,@#TPS
        000100
        177564
 04756 000005          RESET
 04760 000000          HALT
 04762 104060          EMT     60
 04764 000205          RTS     R5
 04766 000001 PR.FI:   .WORD 1
 04770 000000 IO.K:    .WORD 0
```

```
                     ;
                     ;ECO----MACRO DEFINITION TO FETCH A FLOATING
                     ;       WORD AT NUM TO BE CONVERTED TO 9-BYTES
                     ;       ASCII CODES STARTING FROM ASCII
                     ;
                     .MACRO ECO ASCII,NUM
                            MOV     #ASCII,-(SP)
                            MOV     #11,-(SP)
                            MOV     #2,-(SP)
                            MOV     #1,-(SP)
                            MOV     NUM+2(R2),-(SP)
                            MOV     NUM(R2),-(SP)
                            JSR     PC,$ECO
                            .ENDM
                     .GLOBL $ECO
                     ;
                     ;
                     ;IO----SERVICE SUBROUTINE TO DO ASCII CONVERSION
                     ;      AND FILL OUTPUT PRINTER BUFFER
                     ;
04772 016702 IO:     MOV     IO.K,R2;        R2=POINTER
      177772
04776 006302         ASL     R2
05000 006302         ASL     R2
05002 010267         MOV     R2,IO.R2
      000346
05006                ECO $FI1,FI1;           CONVERT 1ST WORD
05042 016702         MOV     IO.R2,R2
      000306
05046                ECO $FI2,FI2;           CONVERT 2ND WORD
05102 016702         MOV     IO.R2,R2
      000246
05106                ECO $FI3,FI3;           CONVERT 3RD WORD
05142 016702         MOV     IO.R2,R2
      000206
05146                ECO $FI4,FI4;           CONVERT 4TH WORD
05202 016702         MOV     IO.R2,R2
      000146
05206                ECO $FI5,FI5;           CONVERT 5TH WORD
05242 012703         MOV     #BUFST,R3
      005470'
05246 004567         JSR     R5,IOF;         3-DIGIT LINE NO.
      000106
05252 012737         MOV     #PRN,@#64
      005266'
      000064
05260 012767         MOV     #BUFST,PONTR
      005470'
      000070
05266 032737 PRN:    BIT     #1,@#177570;    ABORT PROGRAM ?
      000001
      177570
05274 001403         BEQ     1$;             NO,GO AHEAD
```

```
05276 005037              CLR     @#ADCSR;          YES,QUIT
      177520
05302 000423              BR      2$
05304 105767 1$:          TSTB    TPS;              PRINTER READY ?
      177564
05310 100366              BPL     PRN
05312 016700              MOV     PONTR,R0;         YES,LOAD POINTER
      000040
05316 112037              MOVB    (R0)+,@#TPB;      LOAD PRINTER BUFFER
      177566
05322 010067              MOV     R0,PONTR
      000030
05326 020027              CMP     R0,#BUFEND;       FINISH 1 LINE ?
      005571
05332 001007              BNE     2$
05334 005037              CLR     @#TPS;            YES
      177564
05340 005067              CLR     PR.F1;            CLR FLAG
      177422
05344 012737              MOV     #IO,@#64;         RESTORE VECTOR
      004772
      000064
05352 000002 2$:          RTI
05354 000000 IO.R2:       .WORD 0
05356 000000 PONTR:       .WORD 0
             ;
             ;ICF--- SUBROUTINE TO FORM 3-DIGIT ASCII CODE
             ;       IN ASCENDING ORDER EACH TIME IT IS CALLED
             ;       ASCII BUFFER MUST FIRST BE INITIALISED
             ;       BY A NO.
             ;
05360 126327 ICF:         CMPB    2(R3),#71;        DIGIT 3=9 ?
      000002
      000071
05366 001403              BEQ     1$
05370 105263              INCB    2(R3);            NO,INC BY 1
      000002
05374 000434              BR      5$
05376 112763 1$:          MOVB    #60,2(R3);        DIGIT 3=0
      000060
      000002
05404 126327              CMPB    1(R3),#60;   DIGIT 2=0 OR SPACE ?
      000001
      000060
05412 003004              BGT     2$
05414 112763              MOVB    #61,1(R3);        DIGIT 2=1
      000061
      000001
5422 000421               BR      5$
5424 126327 2$:           CMPB    1(R3),#71;        DIGIT 2=9 ?
     000001
     000071
5432 001403               BEQ     3$
```

```
5434 105263          INCB     1(R3);            NO,INC BY 1
     000001
5440 000412          BR       5$
5442 112763 3$:       MOVB    #60,1(R3);        DIGIT 2=0
     000060
     000001
5450 121327          CMPB     (R3),#40;         DIGIT 1=SPACE ?
     000040
5454 001003          BNE      4$
5456 112713          MOVB     #61,(R3);         DIGIT 1=1
     000261
5462 000401          BR       5$
5464 105213 4$:       INCB    (R3);            DIGIT 1=1
5466 000205 5$:       RTS     R5
                   ;
                   ;OUTPUT ASCII BUFFER FOR 5 PARAMETERS
                   ;
5470               BUFST:
5470    040               .BYTE 40,40,60,40,40,40
5471    040
5472    060
5473    040
5474    040
5475    040
5476               $FI1: .BLKB 11
5507    040               .BYTE 40,40,40
5510    040
5511    040
5512               $FI2: .BLKB 11
5523    040               .BYTE 40,40,40
5524    040
5525    040
5526               $FI3: .BLKB 11
5537    040               .BYTE 40,40,40
5540    040
5541    040
5542               $FI4: .BLKB 11
5553    040               .BYTE 40,40,40
5554    040
5555    040
5556               $FI5: .BLKB 11
5567    015               .BYTE 15,12
5570    012
        005571'BUFEND=.
                   .EVEN
                   ;
                   ;DATA BUFFER FOR 5 PARAMETERS IN 2-WORD F.P.
                   ;500 EACH
        000000'           .CSECT DATA
0000               FI1:    .BLKW 1000.
3720               FI2:    .BLKW 1000.
7640               FI3:    .BLKW 1000.
3560               FI4:    .BLKW 1000.
7500               FI5:    .BLKW 1000.
```

```
        005572'          .CSECT
                  .SBTTL ARITHMATICAL SUBROUTINES
                  ;
                  ;
                  ; SUBROUTINE MULFP => A*B=C
                  ;
5572              MULFP:  SAVE R0,R1,R2
5600 012500              MOV       (R5)+,R0;          ADDR.OF
5602 012501              MOV       (R5)+,R1;          A,B &C
5604 012502              MOV       (R5)+,R2
5606                     SAVE      R5
5610 016046              MOV       2(R0),-(SP)
     000002
5614 011046              MOV       (R0),-(SP)
5616 016146              MOV       2(R1),-(SP)
     000002
5622 011146              MOV       (R1),-(SP)
5624 004467              JSR       R4,$POLSH
     000000G
5630 000000G             .WORD     $MLR
5632 005634'             .WORD     .+2
5634 012612              MOV       (SP)+,(R2);        RESULT
5636 012662              MOV       (SP)+,2(R2)
     000002
5642                     UNSAVE R0,R1,R2,R5
5652 000205              RTS       R5
                  ;
                  ; SUBROUTINE ADDFP => A+B=C
                  ;
5654              ADDFP:  SAVE R0,R1,R2
5662 012500              MOV       (R5)+,R0;          ADDR.OF
5664 012501              MOV       (R5)+,R1;          A,B &C
5666 012502              MOV       (R5)+,R2
5670                     SAVE      R5
5672 016046              MOV       2(R0),-(SP)
     000002
5676 011046              MOV       (R0),-(SP)
5700 016146              MOV       2(R1),-(SP)
     000002
5704 011146              MOV       (R1),-(SP)
5706 004467              JSR       R4,$POLSH
     000000G
5712 000000G             .WORD     $ADR
5714 005716'             .WORD     .+2
5716 012612              MOV       (SP)+,(R2);        RESULT
5720 012662              MOV       (SP)+,2(R2)
     000002
5724                     UNSAVE R0,R1,R2,R5
5734 000205              RTS       R5
```

```
                     ;
                     ; SUBROUTINE DIVFP => A/B=C
                     ;
5736                 DIVFP:  SAVE R0,R1,R2
5744 012500                 MOV      (R5)+,R0;          ADDR.OF
5746 012501                 MOV      (R5)+,R1;          A,B &C
5750 012502                 MOV      (R5)+,R2
5752                        SAVE     R5
5754 016046                 MOV      2(R0),-(SP)
     000002
5760 011046                 MOV      (R0),-(SP)
5762 016146                 MOV      2(R1),-(SP)
     000002
5766 011146                 MOV      (R1),-(SP)
5770 004467                 JSR      R4,$POLSH
     000000G
5774 000000G                .WORD    $DVR
5776 006000'                .WORD    .+2
6000 012612                 MOV      (SP)+,(R2);        RESULT
6002 012662                 MOV      (SP)+,2(R2)
     000002
6006                        UNSAVE R0,R1,R2,R5
6016 000205                 RTS      R5
                     ;
                     ; SUBROUTINE SUBFP => A-B=C
                     ;
6020                 SUBFP:  SAVE R0,R1,R2
6026 012500                 MOV      (R5)+,R0;          ADDR.OF
6030 012501                 MOV      (R5)+,R1;          A,B &C
6032 012502                 MOV      (R5)+,R2
6034                        SAVE     R5
6036 016046                 MOV      2(R0),-(SP)
     000002
6042 011046                 MOV      (R0),-(SP)
6044 016146                 MOV      2(R1),-(SP)
     000002
6050 011146                 MOV      (R1),-(SP)
6052 004467                 JSR      R4,$POLSH
     000000G
6056 000000G                .WORD    $SBR
6060 006062'                .WORD    .+2
6062 012612                 MOV      (SP)+,(R2);        RESULT
6064 012662                 MOV      (SP)+,2(R2)
     000002
6070                        UNSAVE R0,R1,R2,R5
6100 000205                 RTS      R5
```

```
                   ;
                   ;MATRIX MULTIPLY [A]*[B]=[C]
                   ;
6102               MMUL:   SAVE R0,R1,R2,R3,R4
6114 012567                MOV     (R5)+,A.MUL;     ADDR OF A
     000376
6120 012567                MOV     (R5)+,B.MUL;     B AND C
     000374
6124 012567                MOV     (R5)+,C.MUL
     000372
6130 013500                MOV     @(R5)+,R0;       VALUES OF
6132 013501                MOV     @(R5)+,R1;       ROWA, COLA
6134 013503                MOV     @(R5)+,R3;       AND COLB
6136                       SAVE R5
6140 006300                ASL     R0
6142 006300                ASL     R0;             4*ROWA
6144 010067                MOV     R0,NXTA;        NEXT ELEM. OF A
     000354
6150 070300                MUL     R0,R3;          4*ROWA*COLB =
6152 066703                ADD     C.MUL,R3;        BASE +
     000344
6156 010367                MOV     R3,NO.ELM;       NO. OF ELEM.IN WORDS
     000352
6162 010102                MOV     R1,R2
6164 070100                MUL     R0,R1;          4*ROWA*COLA
6166 024141                CMP     -(R1),-(R1);     -4 =>
6170 010167                MOV     R1,NXTCOL;       SWITCH COL IN B
     000332
6174 066767                ADD     A.MUL,NXTCOL;    +BASE
     000316
     000324
6202 160001                SUB     R0,R1;           -4*ROWA =>
6204 010167                MOV     R1,NXTROW;       SWITCH ROW IN A
     000320
6210 005302                DEC     R2;             COLA-1
6212 010267                MOV     R2,COLA.1
     000320
6216 006302                ASL     R2
6220 006302                ASL     R2;             4*(COLA-1)=>
6222 010267                MOV     R2,SAMCOL;       SAME COL IN B
     000304
6226 016700                MOV     A.MUL,R0;        BASE ADDR OF A
     000264
6232 016701                MOV     B.MUL,R1;        BASE ADDR OF B
     000262
6236 016703                MOV     C.MUL,R3;        BASE ADDR OF C
     000260
6242 016702                MOV     COLA.1,R2;       NO. OF ADDITIONS
     000270
6246               1$:     SAVE R0,R1,R2,R3
6256 016046                MOV     2(R0),-(SP)
     000002
```

```
6262 011046          MOV     (R0),-(SP)
6264 016146          MOV     2(R1),-(SP)
     000002
6270 011146          MOV     (R1),-(SP)
6272 004467          JSR     R4,$POLSH
     000000G
6276 000000G         .WORD $MLR
6300 006302'         .WORD .+2
6302                 POPF TMPMUL
6312                 UNSAVE R0,R1,R2,R3
6322 022702          CMP     #0,R2;          A = COL VECTOR ?
     000000
6326 001437          BEQ     3$;             YES,SKIP
6330 066700 2$:       ADD    NXTA,R0; NXT ELM. OF A,SAME ROW
     000170
6334 022121          CMP     (R1)+,(R1)+;     NEXT ELM. OF B
6336                 SAVE R0,R1,R2,R3
6346                 PUSHF TMPMUL;    PREVIOUS PARTIAL SUM
6356 016046          MOV     2(R0),-(SP)
     000002
6362 011046          MOV     (R0),-(SP)
6364 016146          MOV     2(R1),-(SP)
     000002
6370 011146          MOV     (R1),-(SP)
6372 004467          JSR     R4,$POLSH
     000000G
6376 000000G         .WORD $MLR
6400 000000G         .WORD $ADR
6402 006404'         .WORD .+2
6404                 POPF TMPMUL
6414                 UNSAVE R0,R1,R2,R3
6424 077237          SOB     R2,2$
6426 016702 3$:       MOV    COLA.1,R2;       RESTORE R2
     000104
6432 016723          MOV     TMPMUL,(R3)+; STORE RESULT
     000102
6436 016723          MOV     TMPMUL+2,(R3)+;  IN C
     000100
6442 026703          CMP     NO.ELM,R3;       DONE ?
     000066
6446 001414          BEQ     10$;            YES,QUIT
6450 020067          CMP     R0,NXTCOL;       NEXT COL OF B?
     000052
6454 001405          BEQ     4$;             YES,SKIP
6456 166701          SUB     SAMCOL,R1;       NO,SAME COL
     000050
6462 166700          SUB     NXTROW,R0;       NEXT ROW OF A
     000042
6466 000667          BR      1$;             CARRY ON, SON
6470 016700 4$:       MOV    A.MUL,R0;        BACK TO ROW 1 OF A
     000022
6474 022121          CMP     (R1)+,(R1)+;     NEXT COL OF B
6476 000663          BR      1$;             CARRY ON, SON
```

```
6500        10$:
6500                UNSAVE R0,R1,R2,R3,R4,R5
6514 000205         RTS       R5
6516 000000 A.MUL:  0
6520 000000 B.MUL:  0
6522 000000 C.MUL:  0
6524 000000 NXTA:  0
6526 000000 NXTCOL: 0
6530 000000 NXTROW: 0
6532 000000 SAMCOL: 0
6534 000000 NO.ELM: 0
6536 000000 COLA.1: 0
6540        TMPMUL: .BLKW 2.
            ;
            ;MATRIX ADDITION [A]+[B]=[C]
            ;
6544        MADD:    SAVE R0,R1,R2,R3,R4
6556 012500         MOV       (R5)+,R0;         ADDR OF A
6560 012502         MOV       (R5)+,R2;         B AND C
6562 012503         MOV       (R5)+,R3
6564 013501         MOV       @(R5)+,R1;        NO. OF ROW
6566 070135         MUL       @(R5)+,R1;        ROW*COL
6570                SAVE R5
6572 006301         ASL       R1
6574 006301         ASL       R1;              4*ROW*COL
6576 060001         ADD       R0,R1;           +BASE
6600        1$:      SAVE R0,R1,R2,R3
6610 016046         MOV       2(R0),-(SP)
     000002
6614 011046         MOV       (R0),-(SP)
6616 016246         MOV       2(R2),-(SP)
     000002
6622 011246         MOV       (R2),-(SP)
6624 004467         JSR       R4,$POLSH
     000000G
6630 000000G        .WORD $ADR
6632 006634'        .WORD .+2
6634                POPF TMPADD
6644                UNSAVE R0,R1,R2,R3
6654 016723         MOV       TMPADD,(R3)+; STORE RESULT
     000032
6660 016723         MOV       TMPADD+2,(R3)+;  IN C
     000030
6664 022020         CMP       (R0)+,(R0)+;      INC R0,R2
6666 022222         CMP       (R2)+,(R2)+;      BY 4
6670 020001         CMP       R0,R1;           DONE ?
6672 001342         BNE       1$;              NO,GO BACK
6674                UNSAVE R0,R1,R2,R3,R4,R5
6710 000205         RTS       R5
6712        TMPADD: .BLKW 2.
```

```
                     ;
                     ;MATRIX SUBTRACTION [A]-[B]=[C]
                     ;
6716                 MSUB:   SAVE R0,R1,R2,R3,R4
6730 012500                  MOV     (R5)+,R0;       ADDR OF A
6732 012502                  MOV     (R5)+,R2;       B AND C
6734 012503                  MOV     (R5)+,R3
6736 013501                  MOV     @(R5)+,R1;      NO. OF ROW
6740 070135                  MUL     @(R5)+,R1;      ROW*COL
6742                         SAVE R5
6744 006301                  ASL     R1
6746 006301                  ASL     R1;             4*ROW*COL
6750 060001                  ADD     R0,R1;          +BASE
6752                 1$:     SAVE R0,R1,R2,R3
6762 016046                  MOV     2(R0),-(SP)
     000002
6766 011046                  MOV     (R0),-(SP)
6770 016246                  MOV     2(R2),-(SP)
     000002
6774 011246                  MOV     (R2),-(SP)
6776 004467                  JSR     R4,$POLSH
     000000G
7002 000000G                 .WORD $SBR
7004 007006'                 .WORD .+2
7006                         POPF TMPSUB
7016                         UNSAVE R0,R1,R2,R3
7026 016723                  MOV     TMPSUB,(R3)+; STORE RESULT
     000032
7032 016723                  MOV     TMPSUB+2,(R3)+;  IN C
     000030
7036 022020                  CMP     (R0)+,(R0)+;    INC R0,R2
7040 022222                  CMP     (R2)+,(R2)+;    BY 4
7042 020001                  CMP     R0,R1;          DONE ?
7044 001342                  BNE     1$;             NO,GO BACK
7046                         UNSAVE R0,R1,R2,R3,R4,R5
7062 000205                  RTS     R5
7064                 TMPSUB: .BLKW 2.
                     ;
                     ;MATRIX SCALAR MULTIPLICATION [A]*SC=[B]
                     ;
7070                 MSC:    SAVE R0,R1,R2,R3,R4
7102 012500                  MOV     (R5)+,R0;       ADDR OF A
7104 012501                  MOV     (R5)+,R1;       ADDR OF B
7106 012502                  MOV     (R5)+,R2;       ADDR.OF SCALER
7110 013503                  MOV     @(R5)+,R3;      NO. OF ROW
7112 070335                  MUL     @(R5)+,R3;      ROW*COL
7114                         SAVE R5
7116 006303                  ASL     R3
7120 006303                  ASL     R3;             4*ROW*COL
7122 060003                  ADD     R0,R3;          +BASE
7124                 1$:     SAVE R0,R1,R2,R3
7134 016046                  MOV     2(R0),-(SP)
     000002
```

```
7140 011046          MOV     (R0),-(SP)
7142 016246          MOV     2(R2),-(SP)
     000002
7146 011246          MOV     (R2),-(SP)
7150 004467          JSR     R4,$POLSH
     000000G
7154 000000G         .WORD $MLR
7156 007160'         .WORD .+2
7160                 POPF TMPMSC
7170                 UNSAVE R0,R1,R2,R3
7200 016721          MOV     TMPMSC,(R1)+; STORE RESULT
     000030
7204 016721          MOV     TMPMSC+2,(R1)+;  IN B
     000026
7210 022020          CMP     (R0)+,(R0)+;    INC R0 BY 4
7212 020003          CMP     R0,R3;  DONE ?
7214 001343          BNE     1$
7216                 UNSAVE R0,R1,R2,R3,R4,R5
7232 000205          RTS     R5
7234         TMPMSC: .BLKW 2.
             ;
             ;MATRIX SCALAR DIVISION [A]/SC=[B]
             ;
[illegible]  DSC:    SAVE R0,R1,R2,R3,R4
7252 012500          MOV     (R5)+,R0;          ADDR OF A
7254 012501          MOV     (R5)+,R1;          ADDR OF B
7256 012502          MOV     (R5)+,R2;          ADDR.OF SCALER
7260 013503          MOV     @(R5)+,R3;         NO. OF ROW
7262 070335          MUL     @(R5)+,R3;         ROW*COL
7264                 SAVE R5
7266 006303          ASL     R3
7270 006303          ASL     R3;               4*ROW*COL
7272 060003          ADD     R0,R3;            +BASE
7274         1$:     SAVE R0,R1,R2,R3
7304 016046          MOV     2(R0),-(SP)
     000002
7310 011046          MOV     (R0),-(SP)
7312 016246          MOV     2(R2),-(SP)
     000002
7316 011246          MOV     (R2),-(SP)
7320 004467          JSR     R4,$POLSH
     000000G
7324 000000G         .WORD $DVR
7326 007330'         .WORD .+2
7330                 POPF TMPDSC
7340                 UNSAVE R0,R1,R2,R3
7350 016721          MOV     TMPDSC,(R1)+; STORE RESULT
     000030
7354 016721          MOV     TMPDSC+2,(R1)+;  IN B
     000026
7360 022020          CMP     (R0)+,(R0)+;    INC R0 BY 4
7362 020003          CMP     R0,R3;          DONE ?
7364 001343          BNE     1$
7366                 UNSAVE R0,R1,R2,R3,R4,R5
7402 000205          RTS     R5
7404         TMPDSC: .BLKW 2.
```

```
                    ;
                    ;MATRIX TRANSPOSITION
                    ;
7410                MTRN:   SAVE R0,R1,R2,R3,R4
7422 012567                 MOV     (R5)+,A.TRN;     ADDR OF A
     000120
7426 012567                 MOV     (R5)+,AT.TRN;    ADDR OF AT
     000116
7432 013500                 MOV     @(R5)+,R0;       NO. OF ROW
7434 013502                 MOV     @(R5)+,R2;       NO. OF COL
7436                        SAVE R5
7440 010267                 MOV     R2,CO.TRN;       R2=C
     000106
7444 010005                 MOV     R0,R5;          R0=R
7446 070502                 MUL     R2,R5;          R5=RC
7450 010503                 MOV     R5,R3
7452 006303                 ASL     R3
7454 006303                 ASL     R3
7456 024343                 CMP     -(R3),-(R3);     R3=4RC-4
7460 010304                 MOV     R3,R4;          R4=4RC-4
7462 010001                 MOV     R0,R1
7464 006301                 ASL     R1
7466 006301                 ASL     R1;             R1=4R
7470 066703                 ADD     A.TRN,R3;        R3=4RC-4+BASE
     000052
7474 016346 1$:             MOV     2(R3),-(SP); GET ELEMENTS
     000002
7500 011346                 MOV     (R3),-(SP); IN REVERSE ORDER
7502 160103                 SUB     R1,R3;  NEXT HIGHER ELEM.
7504 077205                 SOB     R2,1$;          FINISH 1 COL ?
7506 016702                 MOV     CO.TRN,R2;       YES,RESTORE R2
     000040
7512 060403                 ADD     R4,R3;          NEXT COL
7514 077011                 SOB     R0,1$;          DONE ?
7516 016700                 MOV     AT.TRN,R0
     000026
7522 012620 2$:             MOV     (SP)+,(R0)+; STORE TRANSPOSED
7524 012620                 MOV     (SP)+,(R0)+;  RESULTS
7526 077503                 SOB     R5,2$;          DONE ?
7530                        UNSAVE R0,R1,R2,R3,R4,R5
7544 000205                 RTS R5
7546 000000 A.TRN: 0
7550 000000 AT.TRN: 0
7552 000000 CO.TRN: 0
```

```
                       ;
                       ;SHIFT---SUBROUTINE THAT SHIFTS ALL NUM ELEMENTS
                       ;         OF VECTOR A 1 PLACE DOWN
                       ;
     7554              SHIFT:  SAVE R1
     7556 012567               MOV       (R5)+,S.A;               ADDR OF A
          000046
     7562 013501               MOV       @(R5)+,R1;               VALUE OF NUM
     7564                      SAVE      R5
     7566 005301               DEC       R1;             4*(NUM-1)
     7570 006301                ASL      R1;             +BASE ADDR
     7572 006301               ASL       R1
     7574 066701               ADD       S.A,R1
          000030
     7600 016111  1$:           MOV      -4(R1),(R1)
          177774
     7604 016161               MOV       -2(R1),2(R1)
          177776
          000002
     7612 024141               CMP       -(R1),-(R1)
     7614 020167               CMP       R1,S.A
          000010
     7620 001367               BNE       1$
     7622                      UNSAVE R1,R5
     7626 000205               RTS       R5
     7630 000000 S.A:           .WORD 0
          000000'              .END START
```

SYMBOL TABLE

```
A        001010R          AD        003632R   ADCSR  =  177520
ADDFP    005654R          AD.A      003730R   AD.CHK    003726R
ANAL     004502R          ANFLG     004230R   AN.K      004634R
AN.K1    004632R          AT        001010R   AT.TRN    007550R
A.MUL    006516R          A.TRN     007546R   BUFEND=   005571R
BUFST    005470R          BUF1      004224R   BUF2      004226R
B.MUL    006520R          CHNSLR=   177522    CLOCK     003542R
COLA.1   006536R          CONVSN    004310R   COUNT     002050R
CO.TRN   007552R          C.MUL     006522R   C.V1      003626R
DATREG=  177524           DELAY     004232R   DE.D      004306R
DIALOG=  ****** G         DIVFP     005736R   DSC       007240R
DUM1     002000R          DUM2      002004R   DUM3      002010R
EV       002260R          EVV       002330R   EV2       002344R
EV22     002414R          EXP       004476R   F         002054R
FILTER   003450R          FINSH     000622R   FIVE      002022R
FI1      000000R     003   FI2       003720R     003   FI3      007640R     003
FI4      013560R     003   FI5       017500R     003   FU       002070R
FUU      002150R          FY        002164R   FYY       002244R
F.ONE    002430R          GAIN      000010R     002   IDELAY   000004R     002
IFTR     000002R     002   IFTRST    000002R     002   INDEX    002044R
IO       004772R          IOF       005360R   IO.K      004770R
IO.R2    005354R          MADD      006544R   MAXAD     000006R     002
METHOD   000024R          MMUL      006102R   MSC       007070R
MSUB     006716R          MTRN      007410R   MULFP     005572R
NO.ELM   006534R          NXTA      006524R   NXTCOL    006526R
NXTROW   006530R          N10       002030R   N100      002040R
N12      002032R          N13       002034R   N15       002036R
N2       002024R          N3        002026R   ONE       002014R
P        001034R          PA        001200R   PACAT     001440R
PAT      001200R          PC     =%000007     PHI       001414R
PONTR    005356R          PRINT     004636R   PRN       005266R
PR.FI    004766R          PSEUDO    000474R   Q         001224R
QA       001370R          QAT       001370R   QUIT      003724R
R0     =%000000           R1     =%000001     R2     =%000002
R3     =%000003           R4     =%000004     R5     =%000005
SAMCOL   006532R          SHIFT     007554R   SIGN      004500R
SP     =%000006           STAPR     002434R   START     000000R
STA.A    003402R          STA.E     002546R   STA.F     003076R
STA.GO   003446R          STR1      003732R   STR2      004046R
SUBFP    006020R          S.A       007630R   TEMBUF    004472R
THREE    002020R          TMPADD    006712R   TMPDSC    007404R
TMPMSC   007234R          TMPMUL    006540R   TMPSP     004470R
TMPSUB   007064R          TPB    =  177566    TPS    =  177564
TWO      002016R          T5.1      001750R   T5.5      001604R
U        004214R          UPDATE    000640R   Y         004220R
YOLD     001774R          $ADR   =  ****** G  $DVR   =  ****** G
$ECO   = ****** G         $FI1      005476R   $FI2      005512R
$FI3     005526R          $FI4      005542R   $FI5      005556R
$MLR   = ****** G         $POLSH=   ****** G  $SBR   =  ****** G

. ABS.   000000    000
         007632    001
FTRN     000014    002
DATA     023420    003
```

```
          SUBROUTINE DIALOG
          COMMON /FTRN/ IFTR,IFTRST,IDELAY,MAXAD,GAIN
          CALL SETFIL (5,'A',IERR,'KB')
          PRINT 1
          READ (5,2) IFTR
          IF(IFTR.EQ.0) GOTO 100
          PRINT 3
          READ(5,4) IFTRST
           PRINT 5
          READ(5,6) GAIN
 100       PRINT 7
          READ(5,8) IDELAY
          PRINT 11
          READ(5,12) MAXAD
          PRINT 9
          READ(5,10) ZZZZ
          PRINT 77
 1         FORMAT(' DO YOU WANT FILTERING ?'/,
        *  '   IF YES,TYPE 1;  IF NO,TYPE 0'//)
 2         FORMAT(I1)
 3         FORMAT(' WHEN DO YOU WANT FITERING TO START',
        X  1X,' ? [I3]'//)
 4         FORMAT(I3)
 5         FORMAT(' ENTER THE GAIN TERM FOR STOCHASTIC',
        X  1X,' APPROXIMATION. [F5.1]'//)
 6         FORMAT(F5.1)
 7         FORMAT(' ENTER THE MULT. FACTOR FOR THE',
        X  1X,' SAMPLING PERIOD. [I2]'//)
 8         FORMAT(I2)
 11        FORMAT(' ENTER THE MAX. NO. OF SAMPLES YOU',
        X  1X,' WANT. [I3]'//)
 12        FORMAT(I3)
 9         FORMAT(' THANK YOU.  TO START,STRIKE ANY KEY  '//)
 10        FORMAT(A1)
 77        FORMAT(///,5X,'PHI1',8X,'PHI2',8X,'PHI3',
        X  8X,'PHI4',8X,'PHI5',///)
          END FILE 5
          RETURN
          END
```

REFERENCES

[1] ZADEH, L.A.: "From circuit theory to system theory". Proc. IRE, 1962, pp. 856-865.

[2] ASTROM, K.J. and BOHLIN, J.: "Numerical identification of linear dynamic systems from normal operating records" in Theory of Self-adaptive Control System (Ed. P.H. Hammond). Plenum Press 1966.

[3] ASTROM, K.J, and EYKHOFF, P.: "System Identification - a survey". Automatica, vol. 7, 1971, pp. 123-162.

[4] GUPTA, R.D.: "Parameter Estimation of State Variables Models". Ph.D. Thes, Queen's Univ., Kingston, Ont. 1974.

[5] PENROSE, R.: "On the best approximate solution of matrix linear equation". Proc. Cambridge Philos. Soc., vol. 52, 1956, pp. 17-19.

[6] GREVILLE, T.N.E.: "Some applications of the pseudoinverse of a matrix". SIAM Review, vol. 2, 1960, pp. 15-22.

[7] GREVILLE, T.N.E.: "The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations". SIAM Review, Jan. 1959, pp. 38-43.

[8] ALBERT, A.: "Regression and Moore-Penrose Pseudoinverse". Academic Press, 1972.

[9] NOBLE, B.: "Applied Linear Algebra". Prentice-Hall, 1969, pp. 144-145.

[10] LIFF, A.J.: "System identification in the presence of noise by digital techniques". IEEE Intern. Conv. Record, vol. 14, 1966, pp. 152-166.

[11] WELLS, C.H.: "Minimum norm control of discrete systems". IEEE Intern. Conv. Record, vol. 15, 1967, pp. 55-64.

[12] SINHA, N.K. and PILLE, W.: "On-line parametric estimation using matrix pseudoinverse". Proc. IEE, vol. 118, 1971, pp. 1041-1046.

[13] VAN DEN BOOM, A.J.W. and VAN DEN ENDEN, A.W.M.: "The determination of the order of process and noise dynamics". IFAC Conf. on system identification 1973, Hague, pp. 929-938.

[14] SEN, A. and SINHA, N.K.: "On-line system identification using combined stochastic approximation and pseudoinverse algorithm". IFAC Conf. on system identification 1974, Budapest.

[15] SAKRISON, D.J.: "Stochastic approximation recursive method for solving regression problems" in Advances in Communications Systems, vol. 2, 1966, pp. 51-106.

[16] ROBBINS, H. and MONRO, S.: "A stochastic approximation method". Ann. Math. Statistics, vol. 22, 1951, pp. 400-407.

[17] KIEFER, J. and WOLFOWITZ, J.: "Statistical estimation of the maximum of a regression function". Ann. Math. Statistics, vol. 23, 1952, pp. 462-466.

[18] SINHA, N.K. and GRISCIK, M.P.: "A stochastic approximation method". IEEE Trans. vol. SMC-1, Oct. 1971, pp. 338-344.

[19] KWANTY, H.G.: "A note on stochastic approximation algorithms in system identification". IEEE Trans. AC-17, 1972, pp. 570-572.

[20] FU, K.S., NICOLIC, Z.Z., CHIEN, T.Y. and NEE, W.G.: "On the stochastic approximation and related learning techniques". Purdue Univ. Tech. Rep. TR-EE66, April 1966.

[21] SINHA, N.K.: "Estimation of transfer function of continuous system from sampled data". Proc. IEE, vol. 119, May 1972, pp. 612-614.

[22] SINHA, N.K. and BEREZNAI, G.T.: "Suboptimal control of nuclear reactors using low-order models". Int. J. of Control, vol. 18, 1973, pp. 305-319.