August 1973

To the Graduate Council:

I am submitting herewith a thesis written by Carl A. Nehmer entitled "Realization of Sequential Least Squares Parameter Estimation on a Minicomputer." I recommend that it be accepted for nine quarter hours of credit in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Walter L. Green
Major Professor

We have read this thesis
and recommend its acceptance:

L. H. Turner

J. C. Bunge

Accepted for the Council:

Hilton A. Smith
Vice Chancellor for
Graduate Studies and Research

REALIZATION OF SEQUENTIAL LEAST SQUARES PARAMETER ESTIMATION ON A MINICOMPUTER

A Thesis

Presented to

the Graduate Council of

The University of Tennessee

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Carl A. Nehmer

December 1973

ACKNOWLEDGMENTS

For his aid, encouragement, and constructive criticism in the preparation of this thesis, the author is indebted to Dr. Walter L. Green, Associate Professor of Electrical Engineering.

Appreciation is also expressed to Mrs. Linda G. Hoffman for her patience and care in typing the manuscript.

A sincere thanks to AC Spark Plug Division, General Motors Corporation for the fellowship which made possible this thesis and permitted the author to complete the requirements for his Master's degree.

1112975

ABSTRACT

This thesis presents a practical realization of system parameter estimation on a minicomputer (PDP-11/20). An assembly language program, utilizing the sequential least squares algorithm, is developed and used to demonstrate the feasibility of employing small computers for system identification. Execution time for each iterative estimation is approximately 15 milliseconds. A special set of floating point, arithmetic subroutines were developed to achieve this low execution time.

Data for parameter estimation is collected in real-time by means of another program developed for that purpose. This assembly language program offers complete flexibility in specifying sampling rates, data collection rates, input perturbations, and hard-copy teletype outputs.

A relation is derived that transforms the discrete model matrices obtained from the parameter estimation program to a continuous model matrix representation. The algorithm is limited to second-order systems.

Several numerical examples demonstrating the capabilities of the software package are presented.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION AND BACKGROUND	1
	Introduction	1
	Background	1
	Classical Control Theory Modeling	1
	Modern Control Theory Modeling	2
	System Identification	2
	Model Structure	2
	Model Order	3
	Parameter Estimation Techniques	4
	The Least Squares Method	4
	Generalized Least Squares	6
	The Maximum Likelihood Method	7
	Prior Knowledge Fitting	7
	Correlation Analysis	8
	Analog Applications	9
	Selection of an Identification Scheme	9
	Thesis Objective	10
II.	ANALYTICAL DEVELOPMENT	11
	Sampled-Data Systems Representation	11
	Sequential Least Squares Parameter Estimation	14
	Estimation Bias	20
	Input-Output Models	23

CHAPTER PAGE

Identification of Nonlinear Systems. 24

Linearity-in-the-Parameters. 24

Transformations. 24

Polynomial Approximations. 26

III. IMPLEMENTATION OF LEAST SQUARES PARAMETER ESTIMATION 28

Real-Time Data Acquisition 28

Parameter Estimation 33

Matrix and Pointer Initializations 33

Matrix Manipulations 34

Output of Results. 36

Floating Point Subroutines 36

Execution Times. 37

Descriptions of Floating Point Subroutines 38

NRAD 38

MULT 40

DIVD 42

ADDA 44

MATX 46

NEGT 48

DBWD, DTOA, and DECA 48

IV. ANALYSIS AND APPLICATION OF PARAMETER ESTIMATION 52

Physical Significance of the Parameter Estimates 52

Example IV.1 53

CHAPTER PAGE

Criteria for Judging Model "Goodness". 54

Evaluation in Discrete Time. 55

Evaluation in Continuous Time. 56

Application of Parameter Estimation. 60

Simulated Data Example 60

Real-Time Data Examples - Laboratory Setup 63

Parameter Estimation of a Type 1, Second-Order System. . . 65

Parameter Estimation of a Type 1, Second-Order System with a Zero. 68

Summary. 72

V. CONCLUSIONS. 74

LIST OF REFERENCES 76

APPENDIXES 79

A. THE PDP-11/20 MINICOMPUTER AND ITS PERIPHERALS 80

B. SAMPLED-DATA SYSTEMS MONITORING PROGRAM. 84

C. ASSEMBLY LANGUAGE LISTING FOR SAMPLED-DATA SYSTEMS MONITORING PROGRAM 95

D. ASSEMBLY LANGUAGE LISTING FOR SEQUENTIAL LEAST SQUARES PARAMETER ESTIMATION 109

VITA 122

LIST OF TABLES

TABLE		PAGE
II.1.	Classification of Systems by the Concept of Linearity -in-the-Parameters	25
III.1.	Number of Arithmetic Operations Required for Parameter Estimation per Iteration	39
III.2.	Floating Point Subroutine Execution Times for m = 1 and n = 2.	39
B.1.	Input/Output Commands for the Sampled-Data Systems Monitoring Program (SDSMP)	86
B.2.	Subroutines for the Sampled-Data Systems Monitoring Program.	94

LIST OF FIGURES

FIGURE		PAGE
I.1.	Simple System Model.	5
II.1.	Estimation Bias Model.	22
III.1.	Interaction of Computer Software, Peripherals, and External Influences.	29
III.2.	Flow Chart for the Clock Service Routine	32
III.3.	Flow Chart for Parameter Estimation.	35
III.4.	Floating Point Number Format	37
III.5.	Flow Chart for MULT Floating Point Subroutine.	41
III.6.	Flow Chart for DIVD Floating Point Subroutine.	43
III.7.	Flow Chart for ADDA Floating Point Subroutine.	45
III.8.	Flow Chart for MATX Floating Point Subroutine.	47
III.9.	Flow Chart for Floating Point to ASCII Conversion and Output	49
IV.1.	Sampled-Data System Example.	53
IV.2.	CSMP Simulated System Response for $G(s) = \frac{12.09}{s(s + 1.56)}$. . . .	62
IV.3.	Response Generated from the Parameter Estimates, Equation (4.25).	64
IV.4.	Analog Manifold Configuration for $G(s) = \frac{24.18}{s(s + 1.67)}$	66
IV.5.	Actual System Response for $G(s) = \frac{24.18}{s(s + 1.67)}$.	67
IV.6.	Response Generated from the Parameter Estimates, Equation (4.28).	69
IV.7.	Analog Manifold Configuration for $G(s) = \frac{10.75(s + 5.00)}{s(s + 1.67)}$ . .	70

FIGURE PAGE

IV.8. Actual System Response for $G(s) = \frac{10.75(s + 5.00)}{s(s + 1.67)}$ 71

IV.9. Response Generated from the Parameter Estimates, Equation (4.31). 73

CHAPTER I

INTRODUCTION AND BACKGROUND

I. INTRODUCTION

Although system modeling and identification is generally the first problem to overcome in the design of a control scheme, chronologically it has been the last one in control theory to be thoroughly explored. In recent years much attention has been given to this problem[1] with various methods and techniques being proposed and analyzed. Numerous papers have been presented dealing with computer implementation of these various methods in either on-line or off-line situations. However, the software development for achieving identification has for the most part given major emphasis to high level languages such as Fortran. This thesis presents an application of the minicomputer, programmed in assembly language, to the system identification problem. The following paragraphs present a brief outline of the system identification problem and some of the more common identification techniques.

II. BACKGROUND

Classical Control Theory Modeling

One of the major factors which contributes to the success of the frequency-domain techniques of "classical" control theory is that the design methods are complemented by adequate system identification techniques. Graphical analysis of step and impulse responses in the time

domain and frequency analysis generally supply sufficient information for designing a control scheme. These techniques also make it possible to determine a reasonable approximation of the system transfer function. They are limited in use however by their need for specialized inputs under specific operating conditions and their very conceptual nature subject to human errors in graphical interpolation.

Modern Control Theory Modeling

The models used in "modern" control theory are with a few exceptions parametric models in terms of state equations. The need to determine such models from experimental data has given rise to new and more powerful techniques for system identification, vis-à-vis parameter estimation.

III. SYSTEM IDENTIFICATION

The system identification problem can be divided into two steps. The first step, a qualitative operation, is that of classifying the system to be identified as to the type and order of the function relating the input to the output. The second step is making a numerical estimate of the undefined parameters of the selected system structure that optimizes some cost criterion.

Model Structure

The principal distinction between systems is made on the basis of linearity. A system is linear if it satisfies the principle of

superposition. A distinction must likewise be made between stationary and nonstationary systems, the latter being systems whose parameters vary with time. Systems may be considered stationary if their parameters vary slowly in comparison with the time required for adequate identification. The uncorrelated or correlated noise characteristics of a system must also be considered. The choice of a particular class of systems will have a direct bearing on the selection of an identification scheme, the computational effort involved, the possibility of a unique solution, and the usefulness of the subsequent estimations. Unfortunately the choice of model structure has no direct analytical solution and hence must be left to the expertise of the system analyist in interpreting whatever empirical information is available.

Model Order

The determination of the order of a system is only slightly less subjective. Åström[2] presents a statistical method (F-test) in which different order models are tested so as to determine the model order that best fits a given data set. This procedure is time consuming, limited to off-line identification, and does not necessarily assure a model of lowest order.[3] Three other methods, the enhanced determinant ratio, the enhanced residual mean square, and the likelihood ratio, are presented by Woodside[4] and appear feasible for noise levels up to 10%. These methods, especially the second, require considerably less computational effort while still generating reliable order approximations. An additional consideration in selecting the order of a system

is the purpose of the identification. In the case of designing a stable regulator, a fairly crude, low-order model might be sufficient. Here, engineering judgement plays an important role.

IV. PARAMETER ESTIMATION TECHNIQUES

A variety of techniques are available for estimating the parameters of linear systems. Frequently these techniques can also be applied to nonlinear systems if the type of nonlinearity is a priori available. Nonlinear systems, as they relate to the identification problem, are discussed in Chapter II. The following paragraphs present an informal discussion of some of the methods for parameter estimation. A system configuration of utmost simplicity will suffice for the purposes of this discussion.

Consider the system of Figure I.1 where an estimate $\hat{b}$ is to be made for the parameter b. This estimate must be obtained from a finite number of observable signal pairs, u(k) and y(k) where

$$y(k) = bu(k) + n(k), \tag{1.1}$$

and where the expected value of the noise n(k) is zero.

The Least Squares Method

Using the least squares method, $\hat{b}$ is selected so as to minimize the cost function defined by

$$J(\hat{b}) = \sum_{k}[y(k) - \hat{b}u(k)]^2 . \tag{1.2}$$

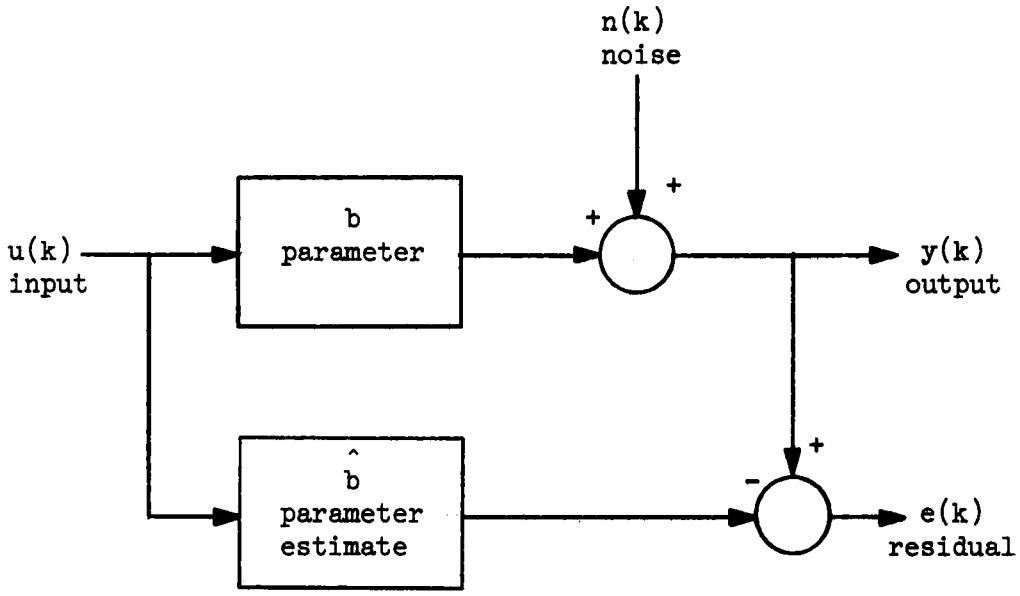

Figure I.1. Simple system model.

A minimum is obtained when the partial derivative of $J(\hat{b})$ with respect to the parameter estimate $\hat{b}$ is obtained, set equal to zero, and the resultant relation solved for $\hat{b}$. This procedure yields

$$\hat{b} = \frac{\sum_k u(k)y(k)}{\sum_k u(k)u(k)} . \tag{1.3}$$

The estimate $\hat{b}$ will be unbiased only if the sequence of residuals given by

$$e(k) = y(k) - \hat{b}u(k) \tag{1.4}$$

is uncorrelated. The problem of correlated residuals will be discussed in Chapter II.

Generalized Least Squares

One method to overcome the difficulty of correlated residuals is the method of generalized least squares.[5] Referring again to Equation (1.1), assume n(k) is a sequence of correlated random variables and that the correlations of the residuals are known and can be represented by

$$n(k) = QV(k) \tag{1.5}$$

where V(k) is a sequence of uncorrelated random variables and Q is a pulse transfer function. Defining new input and output variables

$$\begin{aligned} u^*(k) &= Q^{-1}u(k), \\ \text{and } y^*(k) &= Q^{-1}y(k) \end{aligned} \tag{1.6}$$

results in the ordinary least squares problems described above where Q^{-1} acts as a prefilter. The correlation function of the residuals and consequently the pulse transfer function is seldom known in practice although iterative procedures have been proposed for its determination.(5)

The Maximum Likelihood Method

Another procedure for treating correlated residuals is the maximum likelihood method.(6) In this case one assumes the sequence n(k) to be normal and uncorrelated. The likelihood function, a function of the unknown parameters in the model, is interpreted as the probability that any given model could have produced the observed signal. The particular model corresponding to the maximum of that function is the one of interest. A "hill-climbing" scheme such as the Newton-Raphson algorithm is employed to find this maximum. A filter function similar to Q^{-1} of the generalized least squares (GLS) method is also utilized. Unlike the GLS method, Q^{-1} is altered at each iteration interval by evaluating the first- and second-order partial derivatives of a loss function.

Prior Knowledge Fitting

The prior knowledge fitting method(7) assumes that the noise sequence n(k) is some ergodic random process with zero mean value which is independent of the input sequence u(k). This fact is considered as the only "prior knowledge" available about the system and

can be expressed by

$$E\{n(k)|u(k), u(k-1), \ldots, u(k-N)\} = \sum_{i=0}^{N} c_i u(k-i). \tag{1.7}$$

where the regression coefficients c_i must be zero for all i. A least squares estimate $\hat{c}_i$ of c_i, a function of β, is obtained by minimizing the sum of squared residuals,

$$e(k) = \sum_{i=0}^{N} \hat{c}_i u(k-i) - [y(k) - bu(k)]. \tag{1.8}$$

The estimate $\hat{b}$ of b is then found by minimizing the cost function

$$V = \sum_{i=0}^{N} [\hat{c}_i]^2 \quad , \tag{1.9}$$

by the least squares method.

Correlation Analysis

Correlation analysis[8] techniques give a non-parametric representation of a system. For the system of Equation (1.1)

$$R_{uy}(T) = \sum_{i=0}^{\infty} h(i) R_u(T - i) \tag{1.10}$$

holds provided u(k) and n(k) are uncorrelated, stationary, ergodic random processes. $R_{uy}(T)$ is the cross-correlation function of the input and the output, and $R_u(T)$ is the auto-correlation function of the output. It is possible to estimate the impulse function, h(t), from Equation (1.10), particularly if $R_u(T)$ has a simple form. The simplest form would occur if u(k) were white noise. A pseudo-

random binary sequence[9] gives an excellent approximation of white noise and is widely used as the input perturbation in identification problems regardless of the identification method employed.

Analog Applications

The techniques briefly described above are not limited solely to digital applications. Analog identification schemes using correlation analysis and other statistical techniques have been applied with moderate success.

V. SELECTION OF AN IDENTIFICATION SCHEME

When formulating an identification problem it is important to keep in mind the purpose of the identification. In control problems the primary goal is generally to design a control scheme. Often the identification problem is of a "diagnostic" nature, e.g. rate coefficients in chemical reactions, heat transfer coefficients in industrial processes, and pressure-flow dynamics of the cardiovascular system.

In order to assess the merits of a proposed identification scheme it is necessary to consider a number of related aspects. Several points to consider are:[9]

(1) The usefulness of the selected model structure.

(2) The complexity of equipment required to estimate the model parameters.

(3) The amount of time and data which is required to achieve the desired confidence in the model parameters.

(4) The types and nature of test signals which are to be used, and the extent to which they are likely to disturb normal operation of the system.

(5) The "a priori" information requirements of the particular scheme.

All of these factors interact and so must be evaluated subject to engineering judgements.

VI. THESIS OBJECTIVE

For the purposes of this thesis the sequential version of the least squares (L.S.) method of identification was selected on the basis of the following considerations: (a) the L.S. method does not require "a priori" knowledge about the system disturbances; (b) a minimum amount of computational effort is required for each estimation; (c) the method is well suited for on-line techniques; and (d) least squares convergence is assumed at each estimation step. Factors (b), (c), and (d) are of prime interest as it is the purpose of this thesis to develop a scheme which can be used in real-time simultaneous identification-control problems.

The thesis includes, in Chapter II, a mathematical background of sequential least-squares parameter estimation and associated subjects. Chapter III gives a discussion of the software developed to implement the algorithm while Chapter IV presents application examples. In the final chapter a brief summary of the work is presented and suggested areas for future development are discussed.

CHAPTER II

ANALYTICAL DEVELOPMENT

In this chapter the state transition matrix method of formulating the sampled-data system parameter matrices from the known continuous model representation[10] is reviewed. The derivation of the sequential regression least squares method of parameter estimation for a linear dynamic system is presented.[11] The estimation bias problems associated with this method are discussed along with the extension of the state model to an input-output model. Various aspects of applying this method to the nonlinear case are considered.

I. SAMPLED-DATA SYSTEMS REPRESENTATION

Continuous linear dynamic systems may be described by the state equation

$$\dot{\underline{x}}(t) = \alpha \underline{x}(t) + \beta \underline{u}(t) + \underline{n}(t), \tag{2.1}$$

where $\dot{\underline{x}}(t)$ represents the time derivative of the n-dimensional state vector $\underline{x}(t)$, $\underline{u}(t)$ is the m-dimensional deterministic input vector, $\underline{n}(t)$ is the n-dimensional measurement noise vector, and α and β are respectively the nxn and nxm parameter matrices. Underscoring of a variable denotes a column vector.

Consider the time response of the system of Equation (2.1) when the control signal $\underline{u}(t)$ is subjected to a sampling process. From

Schultz and Melsa[10], the solution is given by

$$\underline{x}(t) = \Phi(t)\underline{x}(0) + \int_0^t \Phi(t - \tau)\beta\underline{u}_s(\tau)d\tau \tag{2.2}$$

where $\Phi(t) = \exp(\alpha t)$ is the state transition matrix (STM). Employing Equation (2.2) and recognizing that the sampled input $\underline{u}_s(t)$ can be represented by

$$\underline{u}_s(t) = \underline{u}(kT) \quad \text{for } kT \leq t < (k + 1)T, \quad k = 0, 1, 2, \cdots \tag{2.3}$$

results in

$$\underline{x}(t) = \Phi(t)\underline{x}(0) + \Phi(t) \sum_{j=0}^{k-1} \left[\int_{jT}^{(j+1)T} \Phi(-\tau)\beta\underline{u}(jT)d\tau \right] + \int_{kT}^{t} \Phi(t - \tau)\beta\underline{u}(kT)d\tau \ . \tag{2.4}$$

For notational convenience let

$$\zeta = \sum_{j=0}^{k-1} \left[\int_{jT}^{(j+1)T} \Phi(-\tau)\beta\underline{u}(jT)d\tau \right] \ . \tag{2.5}$$

The discrete system formulation requires knowledge of the states only at the sampling times, say $t = kT$ and $t = (k+1)T$. Using the notation of Equation (2.5), Equation (2.4) can be written as

$$x(kT) = \Phi(kT)\underline{x}(0) + \Phi(kT)\zeta \tag{2.6}$$

and

$$\underline{x}(\overline{k+1}T) = \Phi(\overline{k+1}T)\underline{x}(0) + \Phi(\overline{k+1}T)\zeta + \int_{kT}^{(k+n)T} \Phi(\overline{k+1}T - \tau)\beta\underline{u}(kT)d\tau \ . \tag{2.7}$$

Solving Equation (2.6) for $\underline{x}(0)$ and substituting into Equation (2.7) gives

$$\underline{x}(\overline{k+1}T) = \Phi(T)\underline{x}(kT) + \int_{kT}^{(k+1)T} \Phi(\overline{k+1}T - \tau)\beta\underline{u}(kT)d\tau \ . \tag{2.8}$$

Making the substitution

$$\tau = (k+1)T - \lambda \ , \tag{2.9}$$

yields

$$\underline{x}(\overline{k+1}T) = \Phi(T)\underline{x}(kT) + [\int_{0}^{T} \Phi(\lambda)\beta d\lambda]\underline{u}(kT), \tag{2.10}$$

or more simply

$$\underline{x}(k+1) = A\underline{x}(k) + B\underline{u}(k) \tag{2.11}$$

where

$$A = \Phi(T) = \exp(\alpha T), \tag{2.12}$$

$$B = \int_{0}^{T} \Phi(\lambda)\beta d\lambda \ . \tag{2.13}$$

For convenience let $\underline{x}(k)$ imply $\underline{x}(kT)$, the state vector at the k'th sampling interval, and $\underline{u}(k)$ imply $\underline{u}(kT)$, the input vector at the same sampling instance.

The inverse problem of reconstructing the continuous model parameters from the discrete model parameters is slightly more involved. First, recall from z-transform theory that a given discrete transfer function is not unique.[12] However, the time function obtained by the inverse of the transform will give the correct values at the sampling instants.

With this fact under consideration solve Equations (2.12) and (2.13) for α and β in terms of A and B. The obvious solution for α is

$$\alpha = \frac{1}{T}\ell n(A). \tag{2.14}$$

Solving the integral of Equation (2.13) using the definition for $\Phi(T)$ gives

$$B = \alpha^{-1}[\exp(\alpha T) - I]\beta \ . \tag{2.15}$$

Consequently β can be expressed as

$$\beta = \frac{1}{T}\ell n(A)[A - I]^{-1}B. \tag{2.16}$$

II. SEQUENTIAL LEAST SQUARES PARAMETER ESTIMATION

The derivation of the algorithms for sequential least squares parameter estimation follows. Consider again Equation (2.11) and recall that A and B are respectively the nxn and nxm parameter matrices to be identified. Define now

$$\underline{w}(k) = [\underline{x}(k)^T \ ; \ \underline{u}(k)^T]^T \ , \tag{2.17}$$

and

$$G = [A;B] = \begin{bmatrix} a_{11} \cdots a_{1n} ; b_{11} \cdots b_{1m} \\ \vdots \\ a_{n1} \quad a_{nn} ; b_{n1} \quad b_{nm} \end{bmatrix} \tag{2.18}$$

Equation (2.11) may now be written as

$$\underline{x}(k+1) = G\underline{w}(k) \ . \tag{2.19}$$

By defining

$$g_i^T = [a_{i1} \cdots a_{in} ; b_{i1} \cdots b_{im}] \ , \tag{2.20}$$

the i'th row of Equation (2.19) may be written as

$$x_i(k+1) = g_i^T \underline{w}(k) \ , \tag{2.21}$$

where $x_i(k+1)$ is recognized as a scalar quantity.

Let $\hat{g}_i$ represent an estimate of the true g_i. The estimate $\hat{x}_1(k+1)$ of $x_i(k+1)$ can then be expressed as

$$\hat{x}_i(k+1) = \hat{g}_i^T \underline{w}(k). \tag{2.22}$$

Employing a least squares regression technique, the identification problem becomes one of making an estimate $\hat{g}_{i,r}$ based on r + 1 measurements of $\underline{x}(k)$ and r measurements of $\underline{u}(k)$ such that the cost index

$$J_{i,r} = \sum_{k=1}^{r} q(k)[x_i(k+1) - \hat{g}_{i,r}\underline{w}(k)^T]^2 \tag{2.23}$$

is minimized. The notation, $\hat{g}_{i,r}$, denotes the r'th estimate of the

i'th row of the parameter matrix G. The inclusion of a nonunity, arbitrary weighting coefficient, q(k), may serve to weight more heavily measurements of greater confidence. The least squares regression estimate of g_i is given by

$$\frac{\partial J_{i,r}}{\partial \hat{g}_{i,r}} = 0 = -2 \sum_{k=1}^{r} q(k)\underline{w}(k)[x_i(k+1) - \underline{w}(k)^T \hat{g}_{i,r}] , \tag{2.24}$$

or

$$\sum_{k=1}^{r} q(k)\underline{w}(k)\underline{w}(k)^T \hat{g}_{i,r} = \sum_{k=1}^{r} q(k)x_i(k+1)\underline{w}(k) . \tag{2.25}$$

Define

$$P_r^{-1} = \sum_{k=1}^{r} q(k)\underline{w}(k)\underline{w}(k)^T . \tag{2.26}$$

This matrix will be nonsingular if and only if $r \geq n+m$ where n and m are the dimensions of $\underline{x}(k)$ and $\underline{u}(k)$ respectively. Equation (2.25) becomes

$$P_r^{-1}\hat{g}_{i,r} = \sum_{k=1}^{r} q(k)x_i(k+1)\underline{w}(k) , \tag{2.27}$$

or

$$\hat{g}_{i,r} = P_r \sum_{k=1}^{r} q(k)x_i(k+1)\underline{w}(k) . \tag{2.28}$$

This last expression is the ordinary linear regression estimate of g_i. To obtain the estimate sequentially, Equation (2.27) is rewritten as

$$P_r^{-1}\hat{g}_{i,r} = \sum_{k=1}^{r-1} q(k)x_i(k+1)\underline{w}(k) + q(r)x_i(r+1)\underline{w}(r) . \tag{2.29}$$

By replacing r with r-1 in Equation (2.25) and making appropriate substitutions in Equation (2.29) yields

$$P_r^{-1}\hat{g}_{i,r} = \sum_{k=1}^{r-1} q(k)\underline{w}(k)\underline{w}(k)^T\hat{g}_{i,r-1} + q(k)x_i(r+1)\underline{w}(r). \tag{2.30}$$

Adding and subtracting $q(r)w(r)w(r)^T\hat{g}_{i,r-1}$ on the right hand side gives

$$\begin{aligned} P_r^{-1}\hat{g}_{i,r} &= \sum_{k=1}^{r-1} q(k)\underline{w}(k)\underline{w}(k)^T\hat{g}_{i,r-1} + q(r)\underline{w}(r)[x_i(r+1) - \underline{w}(r)^T\hat{g}_{i,r-1}] \\ &\quad + g(r)\underline{w}(r)\underline{w}(r)^T\hat{g}_{i,r-1} \\ &= \sum_{k=1}^{r} q(k)\underline{w}(k)\underline{w}(k)^T\hat{g}_{i,r-1} + q(r)\underline{w}(r)[x_i(r+1) - \underline{w}(r)^T\hat{g}_{i,r-1}]. \end{aligned} \tag{2.31}$$

Using the definition of P_r^{-1} reduces Equation (2.31) to

$$P_r^{-1}\hat{g}_{i,r} = P_r^{-1}\hat{g}_{i,r-1} + q(r)\underline{w}(r)[x_i(r+1) - \underline{w}(r)^T\hat{g}_{i,r-1}] \tag{2.32}$$

which yields

$$\hat{g}_{i,r} = \hat{g}_{i,r-1} + P_r q(r)\underline{w}(r)[x_i(r+1) - \underline{w}(r)^T\hat{g}_{i,r-1}]. \tag{2.33}$$

Provided P_r can be derived sequentially, this last expression gives a method by which $\hat{g}_{i,r}$ can be derived sequentially from the previous estimate of g_i, the measurements $x_i(r+1)$ and $\underline{w}(r)$, and the weight $q(r)$.

To obtain the r'th estimate, $\hat{g}_{i,r}$, it is necessary to have the (r+1)'th measurement of x_i available.

Rewrite Equation (2.26) as

$$P_r^{-1} = \sum_{k=1}^{r-1} q(k)\underline{w}(k)\underline{w}(k)^T + q(r)\underline{w}(r)\underline{w}(r)^T$$

$$= P_{r-1}^{-1} + q(r)\underline{w}(r)\underline{w}(r)^T . \qquad (2.34)$$

This expression for P_r^{-1} appears to require the inversion of the matrix P_r. To circumvent this drawback, the matrix inversion lemma[13] is employed to facilitate a recursive derivation of P_r. Define

$$\underline{H}(r) = \sqrt{q(r)}\ \underline{w}(r) \qquad (2.35)$$

such that

$$\underline{H}(r)\underline{H}(r)^T = q(r)\underline{w}(r)\underline{w}(r)^T. \qquad (2.36)$$

Pre-multiplying Equation (2.34) by P_r and post-multiplying by P_{r-1} yields

$$P_{r-1} = P_r + P_r\underline{H}(r)\underline{H}(r)^T P_{r-1}. \qquad (2.37)$$

Further post-multiplication by $\underline{H}(r)$ gives

$$P_{r-1}\underline{H}(r) = P_r\underline{H}(r)[1 + \underline{H}(r)^T P_{r-1}\underline{H}(r)] \qquad (2.38)$$

where it is noted that $[1 + \underline{H}(r)^T P_{r-1}\underline{H}(r)]$ is a scalar. Post-multiplying by $[1 + \underline{H}(r)^T P_{r-1}\underline{H}(r)]^{-1}\underline{H}(r)P_{r-1}$ and substituting for

$P_r\underline{H}(r)\underline{H}(r)^T P_{r-1}$ from Equation (2.37) yields

$$P_{r-1}\underline{H}(r)[1 + \underline{H}(r)^T P_{r-1}\underline{H}(r)]^{-1}\underline{H}(r)P_{r-1} = P_{r-1} - P_r \ , \qquad (2.39)$$

or

$$P_r = P_{r-1} - P_{r-1}\underline{H}(r)[1 + \underline{H}(r)^T P_{r-1}\underline{H}(r)]^{-1}\underline{H}(r)^T P_{r-1} \ . \qquad (2.40)$$

Recognizing once again that $[1 + \underline{H}(r)^T P_{r-1}\underline{H}(r)]^{-1}$ is a scalar, it can be seen that no matrix inversion is involved in deriving P_r by the recursive relation of Equation (2.40) provided an initial estimate, P_o, is available.

The initial estimate, P_o, may be arbitrary[14]; however, for a near optimal solution of $\hat{g}_i$, an initial estimate due to Lee[15] is advantageous. Consider Equation (2.34) for the case where $r = 1$[†], that is

$$P_1^{-1} = P_o^{-1} + q(1)\underline{w}(1)\underline{w}(1)^T \ . \qquad (2.41)$$

By the definition of P_r^{-1}, Equation (2.26), this holds true only for

$$P_o^{-1} = [0] \ , \qquad (2.42)$$

where [0] denotes the null matrix. Consequently, a near optimal

---

[†]Let r = 1 denote the first possible trial, i.e. r = n+m. To satisfy this condition, assume that measurements of $\underline{w}(k)$ are available for negative time.

initial estimate is

$$P_o = \frac{1}{\varepsilon} I \Big|_{\varepsilon \to 0} \tag{2.43}$$

where I is the identity matrix and $1/\varepsilon$ is a scalar. An initial estimate for $\hat{g}_{i,o}$, obtained by utilizing the above results, is given by

$$\hat{g}_{i,o} = [0] \ . \tag{2.44}$$

III. ESTIMATION BIAS

As mentioned in Chapter I, the estimates $\hat{g}_{i,r}$ will be unbiased only if the sequence of residuals, $[x_i(r+1) - \underline{w}(r)^t \hat{g}_i(r-1)]$, given by Equation (2.33) is uncorrelated. For an example where this assumption does not hold true, consider the following noise-free first order system,

$$z(k+1) = az(k) + bu(k) \ , \tag{2.45}$$

$$x(k) = z(k) + n(k) \ . \tag{2.46}$$

For this case x(k) is the observed value of z(k) and n(k) is the independent measurement errors (additive noise). Equation (2.45) can then be written as

$$x(k+1) = ax(k) + bu(k) + n(k+1) - an(k). \tag{2.47}$$

The terms n(k+1) and an(k) show that this system has correlated

residuals and thus an estimation bias. The reason for this bias can best be indicated by referring to Figure II.1. The estimate results from a minimization of the loss function,

$$J = \sum_{k=1}^{r} [e(k)]^2 , \tag{2.48}$$

which is given by

$$\frac{\partial J}{\partial \hat{b}} = \sum_{k=1}^{r} e(k)u(k-1) = 0 , \tag{2.49}$$

and

$$\frac{\partial J}{\partial \hat{a}} = \sum_{k=1}^{r} e(k)x(k-1) = 0 , \tag{2.50}$$

where $\hat{a}$ and $\hat{b}$ are respectively the least squares estimate of a and b. In Equation (2.49) the additive noise n(k) is present only in e(k), not in u(k), and consequently does not affect the expectation of $\hat{b}$. In Equation (2.50) the additive noise is present in both e(k) and x(k) and hence leads to a term

$$\sum_{k=1}^{r} [n(k-1)]^2 \tag{2.51}$$

which does cause a bias of $\hat{a}$.

The generalized least squares and maximum likelihood methods of Chapter I are designed to compensate for this problem. The price of the improved accuracy, however, is added computation effort. In all methods, computer round-off errors and quantization errors may seriously impair the achievable accuracy of even the most sophisticated

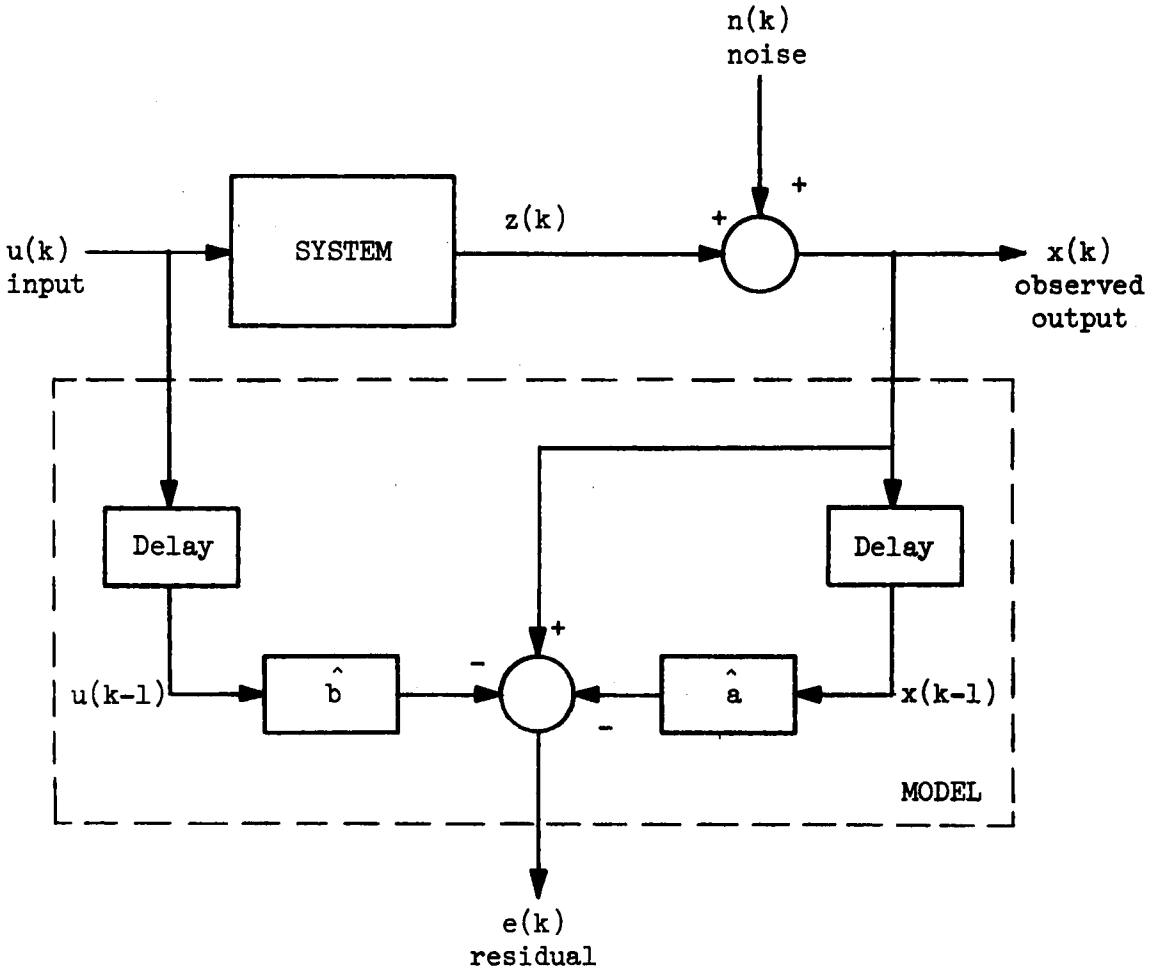

Figure II.1. Estimation bias model.

methods. These factors must be reviewed in light of the identification objective before any one method can be pronounced "better" than another.

IV. INPUT-OUTPUT MODELS

The sequential least squares parameter estimation procedure described earlier is concerned with a state-space formulation of the system. When suitable states of the system are inaccessible, a parametric model may be useful. A canonical form of a linear, time invarient, discrete-time model with one input and one output is given by

$$y(k) = a_1 y(k-1) + \cdots + a_n y(k-n) + [b_1 u(k-1) + \cdots + b_m u(k-m)] \tag{2.52}$$

where u(k-1) ... u(k-m) is the input and y(k) is the output. By redefining $\underline{w}(k)$ of Equation (2.17) as

$$\underline{w}(k) = [y(k-1) \cdots y(k-n); u(k-1) \cdots u(k-m)] , \tag{2.53}$$

and g^T of Equation (2.20) as

$$g^T = [a_1 \cdots a_n; b_1 \cdots b_m] , \tag{2.54}$$

Equation (2.22) becomes

$$\hat{y}(k) = \hat{g}^T \underline{w}(k) . \tag{2.55}$$

Equations (2.33) and (2.40) can then be applied without alteration to the input-output model identification problem.

V. IDENTIFICATION OF NONLINEAR SYSTEMS

Linearity-in-the-Parameters

The classification of a system as nonlinear by the law of superposition does not necessarily render it incapable of being identified by linear estimation techniques. The more important criterion with respect to parameter estimation is linearity-in-the-parameters.[1] This concept is concerned with the relationship between the output variables and the parameters. To illustrate that nonlinear by the law of superposition does not necessarily imply nonlinear-in-the-parameters and vice versa, consider the examples of Table II.1. A system is assumed with input signal u and output signal y where e denotes the error between system and model output and w is a dummy variable. The model is selected to demonstrate the different possibilities.

Transformations

The nonlinear estimation problem is greatly simplified if a transformation of the variables can be made that will create a model that is linear-in-the-parameters. In such a case, ordinary least squares estimation can be employed. The examples below illustrate this. The symbol " $\Rightarrow$ " denotes the transformation.

TABLE II.1

CLASSIFICATION OF SYSTEMS BY THE
CONCEPT OF LINEARITY-IN-THE-PARAMETERS

System	Dynamic Behavior	MODEL -in-the-parameters Linear	Nonlinear
$\dot{y} + ay = u$	linear	$e = \dot{y} + \alpha y - u$	$e = y - w$ $\dot{w} + \alpha w = u$
$\dot{y} + ay^3 = u$	nonlinear	$e = \dot{y} + \alpha y^3 - u$	$e = y - w$ $\dot{w} + \alpha w^3 = u$

Source: K. J. Åström, "System Identification," Preprints of the 2nd Prague IFAC Symposium, p. 9, June, 1970.

Reciprocal transformation:

$$z = \frac{\alpha_2 x_2 + x_1}{\alpha_1 x_1 x_2} \quad \Rightarrow \quad z = \beta_1 u_1 + \beta_2 u_2$$

where

$$\frac{1}{x_1} = u_1; \ \frac{1}{x_2} = u_2; \ \frac{\alpha 2}{\alpha_1} = \beta_1; \ \frac{1}{\alpha_1} = \beta_2 .$$

Logarithmic transformation:

$$z = c x_1^{\alpha_1} x_2^{\alpha_2} \quad \Rightarrow \quad y = \beta_o + \beta_1 u_1 + \beta_2 u_2$$

where

$$\log z = y; \ \log x_1 = u_1; \ \log x_2 = u_2;$$

$$\log c = \beta_o; \ \alpha_1 = \beta_1; \ \alpha_2 = \beta_2 .$$

The transformations illustrated above require a priori information about the type of nonlinearity involved. Assumptions of this nature are not unrealistic. The relationship between certain variables in a process is often known from theoretical considerations. Often a priori information about the operating conditions of the nonlinear system may make a linear approximation possible. If the changes about an "operating point" are considered to be the variables then the system can be characterized by a linear relation.

Polynomial Approximations

A polynomial approximation of a nonlinear system may be required when insufficient information is available for making a

suitable transformation or when a transformation is not feasible. Orthogonal polynomial expressions are especially attractive for approximating nonlinear systems whose characteristics are unknown. The coefficients of the polynomial approximation equation are independent of the highest power of the polynomial equation that is considered. Hence several order models may be checked for the best fit without recalculating the coefficients of lower order. The Chebyshev polynomials possess an added property that make them particularly attractive. The approximation errors for a given set of data oscillate between two limits that are almost equal. Other expansions, such as the Volterra series[11], or the Laguerre function series in Hilbert space[19] also are used for modeling.

The following chapter gives the flow charts and the algorithums for implementing practical software for sequential parameter estimation as described above.

CHAPTER III

IMPLEMENTATION OF LEAST SQUARES PARAMETER ESTIMATION

In this chapter it is shown how sequential least squares parameter estimation is realized on Digital Equipment Corporation's PDP-11/20 minicomputer. A detailed analysis of the software employed is given. A description of the PDP-11/20 and the peripherals[16] utilized can be found in Appendix A.

The software developed to achieve identification can be divided into three major areas as given below.

1. Development of a versital software package for the acquisition of data.
2. Implementation of the sequential least squares algorithm including the necessary initializations and transfers of data.
3. Development of fast, floating point, arithmetic subroutines to achieve the necessary precision.

The interaction of the software programs, the PDP-11/20 peripherals, and the external influences is depicted in Figure III.1.

I. REAL-TIME DATA ACQUISITION

The executive program of this software package, called the Sampled-Data Systems Monitoring Program (SDSMP), oversees the different operations involved in controlling and investigating a sampled-data

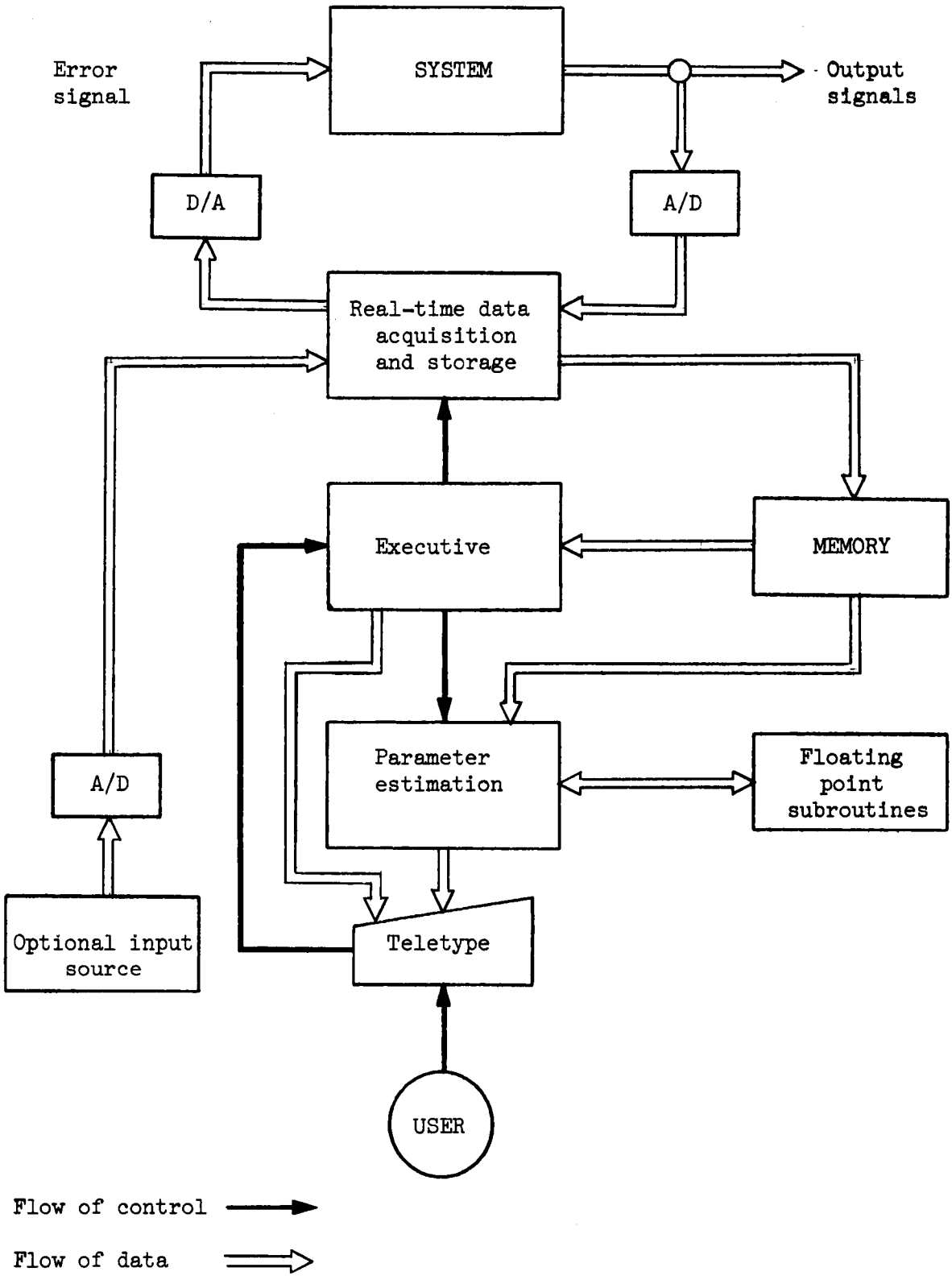

Figure III.1. Interaction of computer software, peripherals, and external influences.

system in real-time. Of foremost importance is the real-time acquisition of information from a dynamic system, the storage of data, and returning the necessary information to the system for control purposes. The section of SDSMP that performs this function is the service routine for the programmable real-time clock and is described below. The structure and features of the program that are not directly related to the identification problem will be discussed in Appendix B.

The parameter estimation algorithm demands that the data be collected at evenly spaced intervals of time over a range of dynamic operation. Similarly, to be consistent with z-transform theory, the time intervals between applications of control information to the system must also be constant. To accomplish this, a programmable real-time clock was utilized. The clock provides repeated interrupts of a program at selected time intervals. An interrupt causes a program to be temporarily terminated while the service routine for the interrupting device is executed. An interrupt can occur only if the priority of the interrupting device is higher than that of the program being executed. By selecting priorities for the programs lower than that of the clock, a constant interrupt interval can be assured.

The interrupt interval is a constant multiple of a preselected clock rate. A clock rate of 100 KHz is used for SDSMP. The program user selects a suitable multiple (BUF2). The procedure for changing this constant and others is described in Appendix B. A dilemma arises in that the sampling interval desired for control purposes and the in-

terval desired for data storage are generally different and frequently not even integer multiples of one another. This problem is solved by a pair of counters in the clock service routine.

The clock service routine is executed at every interrupt interval. Referring to the flowchart in Figure III.2, the output signal (OUTPT) and an auxiliary signal (XAD1) are obtained through an analog-to-digital (A/D) convertor and temporarily stored. A "flag" (OISI) is then checked to determine the source of the input signal. Two options exist: (1) a step-input of user specified amplitude, or (2) a signal from any signal generating device connected to the A/D convertor. If OISI does not equal zero then the latter situation is present. In such a case an A/D conversion is executed and the temporary input location (INPUT) updated. The error signal is next calculated by

$$\text{CERR} = \text{INPUT} - \text{OUTPT}. \tag{3.1}$$

Another flag is interrogated to determine if the calculated error (CERR) should be applied during this interrupt. The flag (TSR2) is a counter which determines the sampling interval of the system. If upon being decremented, TSR2 equals zero, the calculated error is stored in the applied error storage location (ERROR). CERR is then arithmetically shifted left to compensate for the fact that A/D conversion is 11-bit bipolar for $\pm$ 1Ø volts full scale whereas digital-to-analog (D/A) conversion is 12-bit bipolar for the same scale. The corrected error is applied to the system through a D/A convertor and

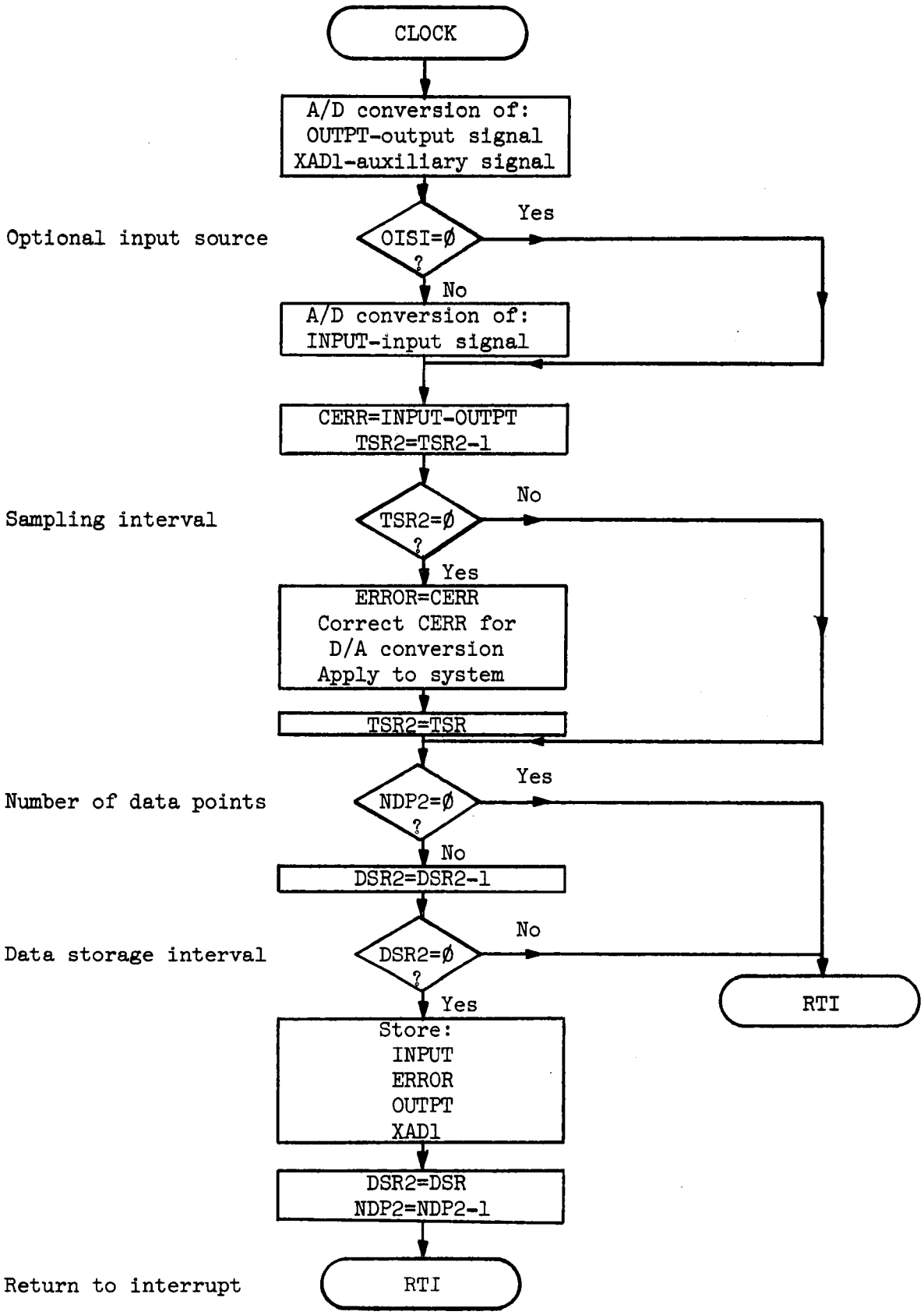

Figure III.2. Flow chart for the clock service routine.

the sampling rate counter, TSR2, is reset.

A third flag (NDP2) is checked to determine if a data collection is in progress. NDP2 is preset to the number of sets of data points requested at the start of a data collection sequence. It is decremented after each set is stored and will equal zero if the collection is complete. If a collection is in progress, then another flag (DSR2) is interrogated to determine if data should be stored during this interval. This flag is the counter that determines the data storage interval of the system, independent of the system sampling interval. Accordingly, the following signals are stored for possible future use: (1) INPUT, the present input, (2) ERROR, the most recent applied error, (3) OUTPT, the present output; and (4) XAD1, the present auxiliarly signal. The data storage rate counter is reset and NDP2 is decremented. The computer then returns to the interrupted program and resumes execution.

II. PARAMETER ESTIMATION

The parameter estimation program is divided into three parts: (1) the matrix and pointer initializations, (2) the conversion of A/D data at each iteration and the subsequent matrix manipulations, and (3) the output of the final results along with the number of iterations.

Matrix and Pointer Initializations

The parameter estimation program can run congruently only with the clock service routine of the SDSMP program. The other routines

utilize the same registers as the parameter estimation program and consequently will cause disastrous errors if allowed to function. To alleviate this problem, the priority of the parameter estimation program is set higher than the other routines with the exception of the clock service routine. It is also necessary to terminate all data collection before the parameter estimation program can be executed.

Two matrix initializations are required. First, the parameter matrix must be set equal to zero according to Equation (2.44). Secondly, the initial estimate of the P matrix must satisfy Equation (2.43). This requires that an identity matrix be multiplied by a constant. The pointers, TX1, TX2, and TIN, are initialized with the starting adresses of the OUTPT table, the XAD1 table, and the INPUT table respectively. The iterations counter (R) is set to zero.

Matrix Manipulations

Using the pointers set above, the A/D data corresponding to the R'th iteration is obtained, converted to floating point normalized form, and placed in the state and input vector, HRØ, and the next-state vector, XR1-XR2. The pointers are updated for the next iteration.

Referring to the flowchart in Figure III.3, the sequence of matrix operations for the parameter estimation program is selected so as to minimize computational effort and storage requirements. Whenever possible the symmetry or equivalence of the various matrices is employed. An example of this is the symmetrical matrix P(R-1). Equation (2.40) demands the series of matrix multiplications

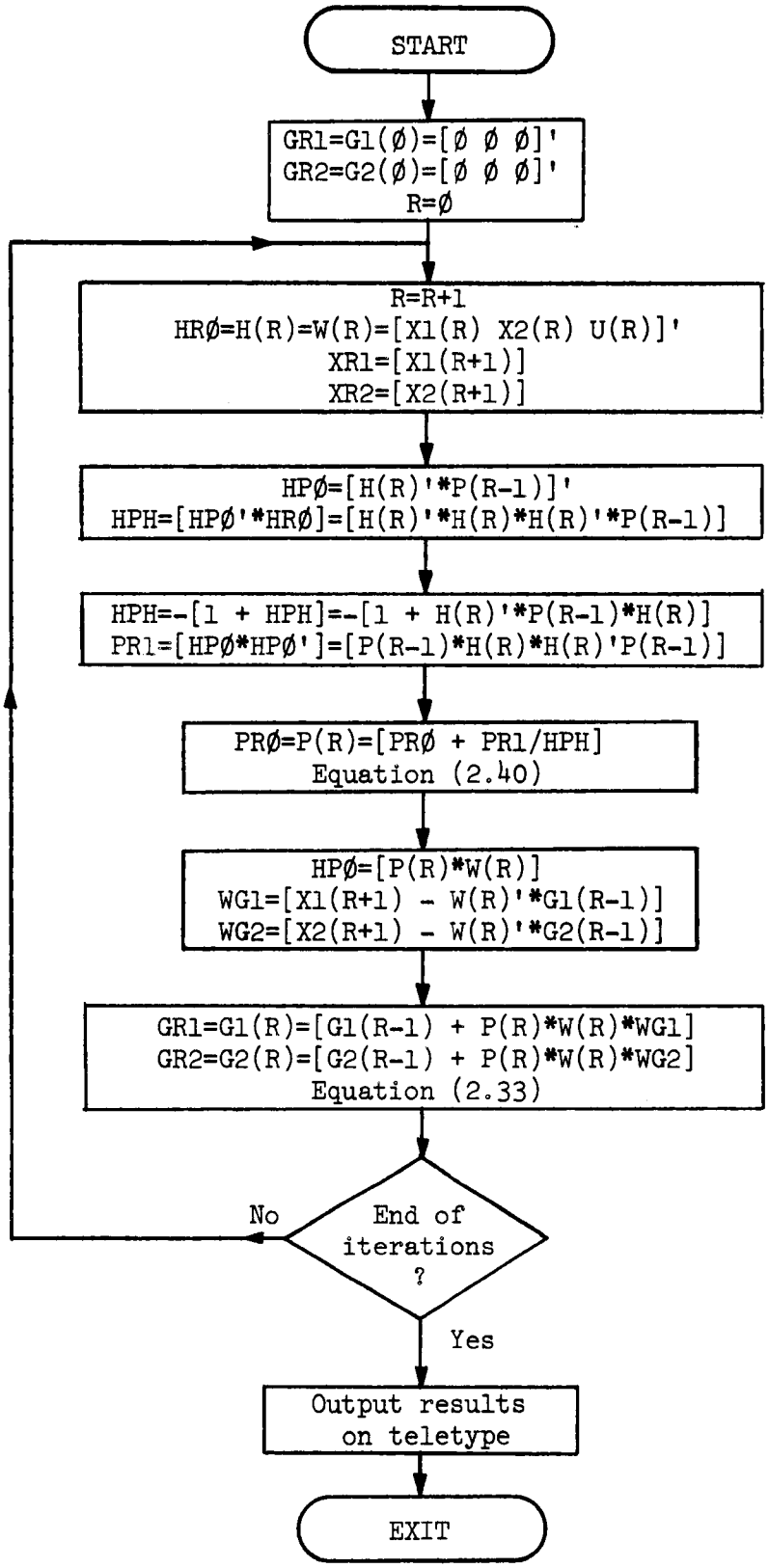

Figure III.3. Flow chart for parameter estimation.

$$H(R)'*P(R-1)*H(R), \tag{3.2}$$

and

$$P(R-1)*H(R)*H(R)'*P(R-1) \tag{3.3}$$

where P(R-1) is a pxp matrix and H(R) is a p-dimensional vector. An asterisk denotes multiplication and an apostrophe indicates the transpose of a matrix. The common term in Equations (3.2) and (3.3) is recognized and duly employed. Furthermore, due to the symmetry of P(R-1),

$$[P(R-1)*H(R)]' = [H(R)'*P(R-1)]. \tag{3.4}$$

These and other equivalences are utilized to substantially reduce the average total execution time.

Output of Results

The final values of the parameters are output on the teletype after all calculations are complete. The number of requested iterations is also printed out. Format of the output is consistent with the definition of the matrices A and B given by Equation (2.11). Control is relinquished to SDSMP at the completion of the output.

III. FLOATING POINT SUBROUTINES

The complexity of the calculations, the wide variation in the magnitudes of numbers and the precision required by the sequential,

least squares algorithm demands that the computations be performed using floating point arithmetic. The floating point format utilized is as appears in Figure III.4.

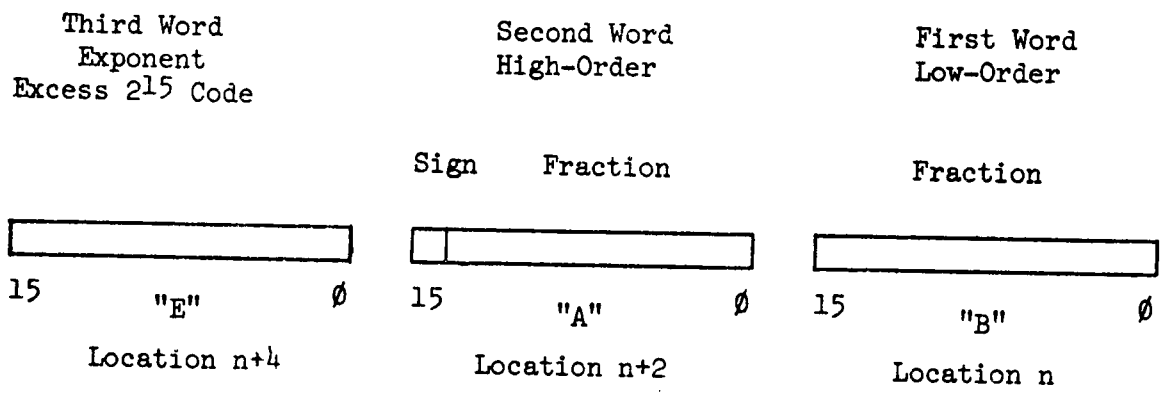

Figure III.4. Floating point number format

Execution Times

The objective of a short execution time requires that a special set of floating point subroutines be developed. The standard floating point package available from Digital Equipment Corporation (DEC) for use on the PDP-11/20 does not utilize the hardware available and hence the execution times are excessively long. As the subroutines of the standard package are also general purpose in nature, a savings in time is realized by creating special purpose routines. One additional requirement is that the subroutines must be able to endure an interruption, subject to certain restrictions, without loss of information.

The number of arithmethic operations required for a single iteration of Equations (2.33) and (2.40) is given in Table III.1. For the general case of m = 1 and n = 2, the execution times for DEC's subroutines and the ones developed are given in Table III.2. All execution times are based on the average value as given by DEC[20].

The five-fold savings in time without loss of precision is achieved by judicious use of the Extended Arithmetic Element[17]. The subroutines available from DEC use software exclusively. Depending upon the addressing mode, DEC's subroutines may also require up to 300 μsec additional time for each of the 115 operations. Another savings in time is realized by recognizing that a minimum of 4p+1 of the additions require only a minimum execution time which for DEC is 186 μsec and for the subroutine developed is 25 μsec.

Descriptions of Floating Point Subroutines

The following paragraphs describe the function of each subroutine, the necessary formatting, and the techniques used to implement each algorithm. Flowcharts accompany most of the descriptions. A documented assembly language listing of the subroutines is found in Appendix D.

NRAD. Converts an A/D word to floating point format. The word is scaled by a factor of 1.25/128 to achieve the correct floating point representation. The source pointer, R4, contains the address of the A/D word. The destimation pointer, R5, contains the address of the location in which the results are to be deposited. Upon exit from the

TABLE III.1

NUMBER OF ARITHMETIC OPERATIONS REQUIRED
FOR PARAMETER ESTIMATION PER ITERATION

Operation	Number
Addition	$n + p(2n+1) + 4p^2$
Multiplication	$p(2n+1) + 3p^2$
Division	p^2
Negation	$2n + 1$
Normalization	$2n + m$

TABLE III.2

FLOATING POINT SUBROUTINE EXECUTION TIMES
FOR m = 1 AND n = 2

Operation	No.	* DEC = (μsec)	* Thesis = (μsec)
Addition	53	* 364 = 19,282	* 135 = 7155
Multiplication	43	* 810 = 34,830	* 135 = 5805
Division	9	*1770 = 15,930	* 115 = 1035
Negation	5	* 26 = 130	* 17 = 85
Normalization	5	* 384 = 1920	* 50 = 250
Total	115	72,092 μs	14,330 μs

program, R4 points to the next A/D word, and R5 is advanced the equivalent of one floating point number i.e. three words. The A/D word is not disturbed by the conversion. If the A/D word is equal to zero then a floating point zero is generated which is defined as three words of zero.

MULT. Multiplies two floating point numbers together without altering their values. The product is stored at a third location specified by the destination pointer. The two source pointers are advanced to the beginning of the next floating point word-set. The destination pointer points to the first word of the product upon exit from the subroutine.

Referring to the flowchart in Figure III.5, the destination pointer is first saved on the "stack". The address of the Multiplier Quotient (MQ), the low-order data register of the Extended Arithmetic Element (EAE), is placed in the register RØ. A savings in time of 1.2 μsec per instruction, and a saving in memory of one word per instruction is realized by using RØ as a pointer to the addresses in the EAE. This mode of addressing is also used in the DIVD and ADDA subroutines. The product of the two high-order words is formed and then the exponents of the two sources are checked. If either is zero, a product of zero is formed and the subroutine is exited. If neither is zero, the sum of the exponents is formed and corrected to compensate for the summing process.

The two secondary products, i.e. the product of the high-order

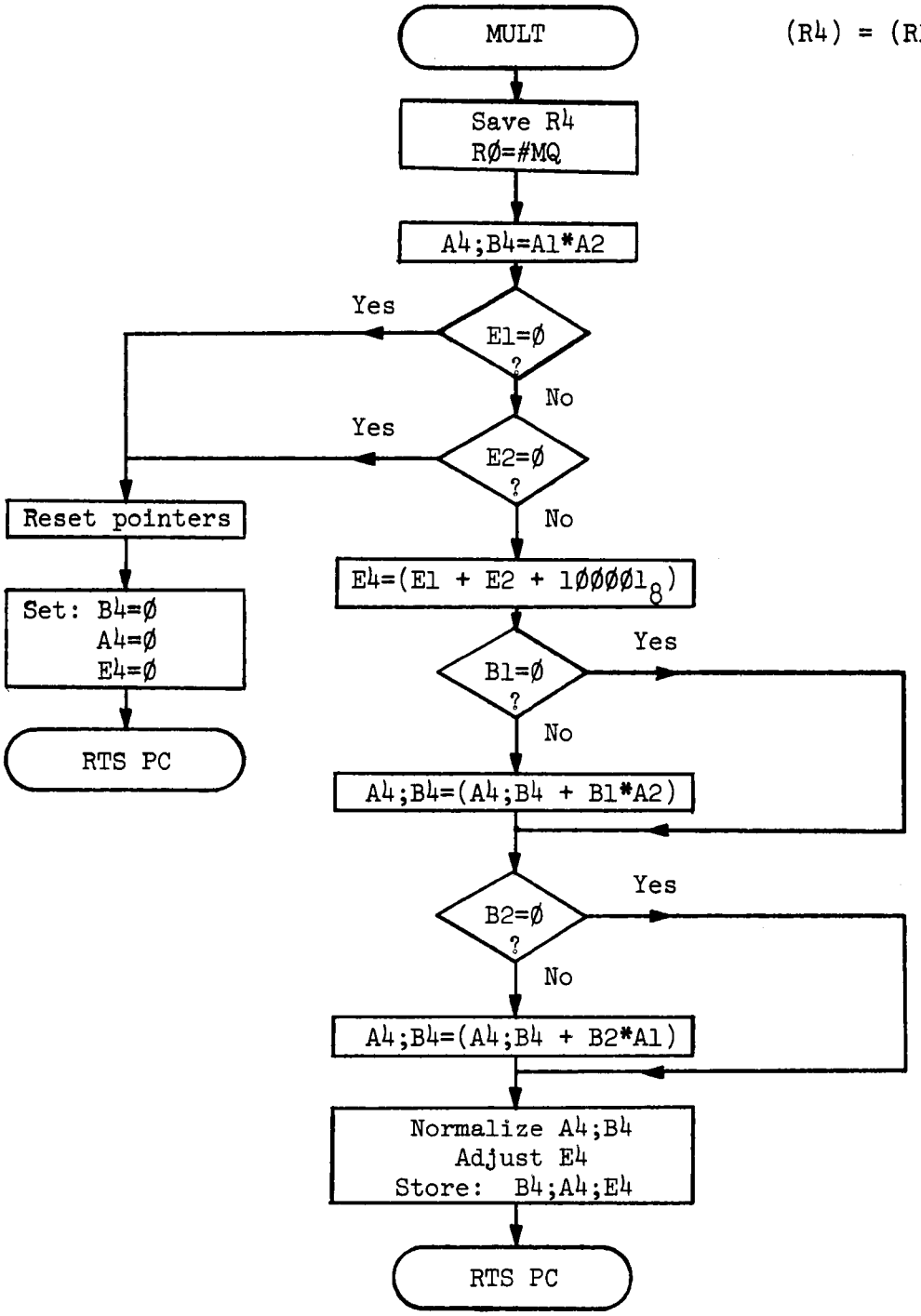

Figure III.5. Flow chart for MULT floating point subroutine.

word of one source and the low-order word of the other, are formed and added to the low-order word of the primary product. The multiplications are circumvented if a low-order word is zero. A logical shift right must be performed on the low-order word before the secondary product can be formed. This is to prevent the occurrence of an erroneous change in sign. A logical left shift is executed after the product is formed to compensate for this algorithm adjustment. The final product is normalized and stored. The exponent is adjusted to correct for any shifts during normalization.

DIVD. Divides one floating point word by the high-order word of another floating point word. The quotient replaces the dividend and the pointer is advanced. The divisor and the divisor pointer are unaltered by the operation. See Figure III.6 for the flowchart.

The high- and low-order words of the dividend are loaded into the data registers of the EAE. An arithmetic shift of two places to the right is performed to eliminate the problems that might otherwise occur. Very little loss in accuracy is incurred. At this point the exponent of the dividend is checked. If it is zero then all that is required is an adjustment of the quotient pointer before exiting. Otherwise, the primary quotient is formed by dividing the adjusted high- and low-order words by the high-order word of the divisor. A flag is set and the subroutine exited if the divisor equals zero. The new exponent is formed by the difference of the dividend exponent and the divisor exponent plus an algorithm adjustment.

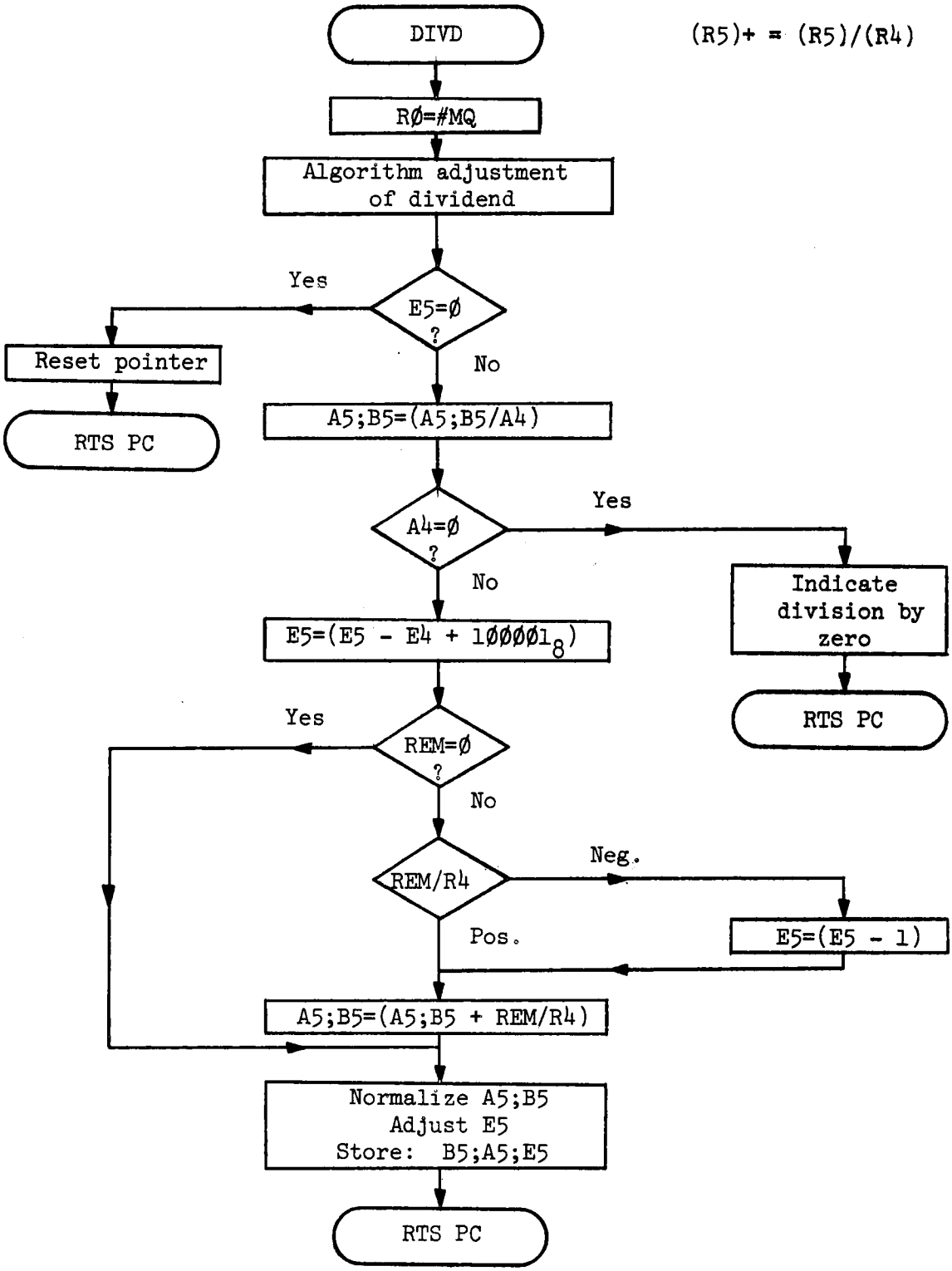

Figure III.6. Flow chart for DIVD floating point subroutine.

A remainder of zero from the above division eliminates the next several steps. The remainder is shifted into position and then divided by the high-order word of the divisor. This secondary quotient is added to the primary quotient. If necessary, the exponent is adjusted. The result is normalized, and stored. An adjustment of the exponent is made to account for any shift during the normalization process.

ADDA. Adds two floating point numbers. The sum replaces the second addend. Neither register pointer is advanced. The first addend may be altered by the subroutine.

Referring to the flowchart in Figure III.7, the first operation is to check the exponents of both addends. If either is zero, then the sum is simply the other addend and the subroutine is quickly exited. With nonzero exponents, the next step is to set up the EAE address pointer as in the MULT subroutine. The difference between the two exponents is then calculated for purposes of the algorithm. In performing a floating point addition, the exponents of the two numbers must be identical. To accomplish this the fraction of the addend with the more negative exponent is arithmetically shifted right the number of places corresponding to the absolute difference in the two exponents. This aligns the bits of a particular weight in one addend with the bits of equal weight in the other addend.

The low-order words of the two numbers are summed and the carry bit added to the high order word of the destination number. If an

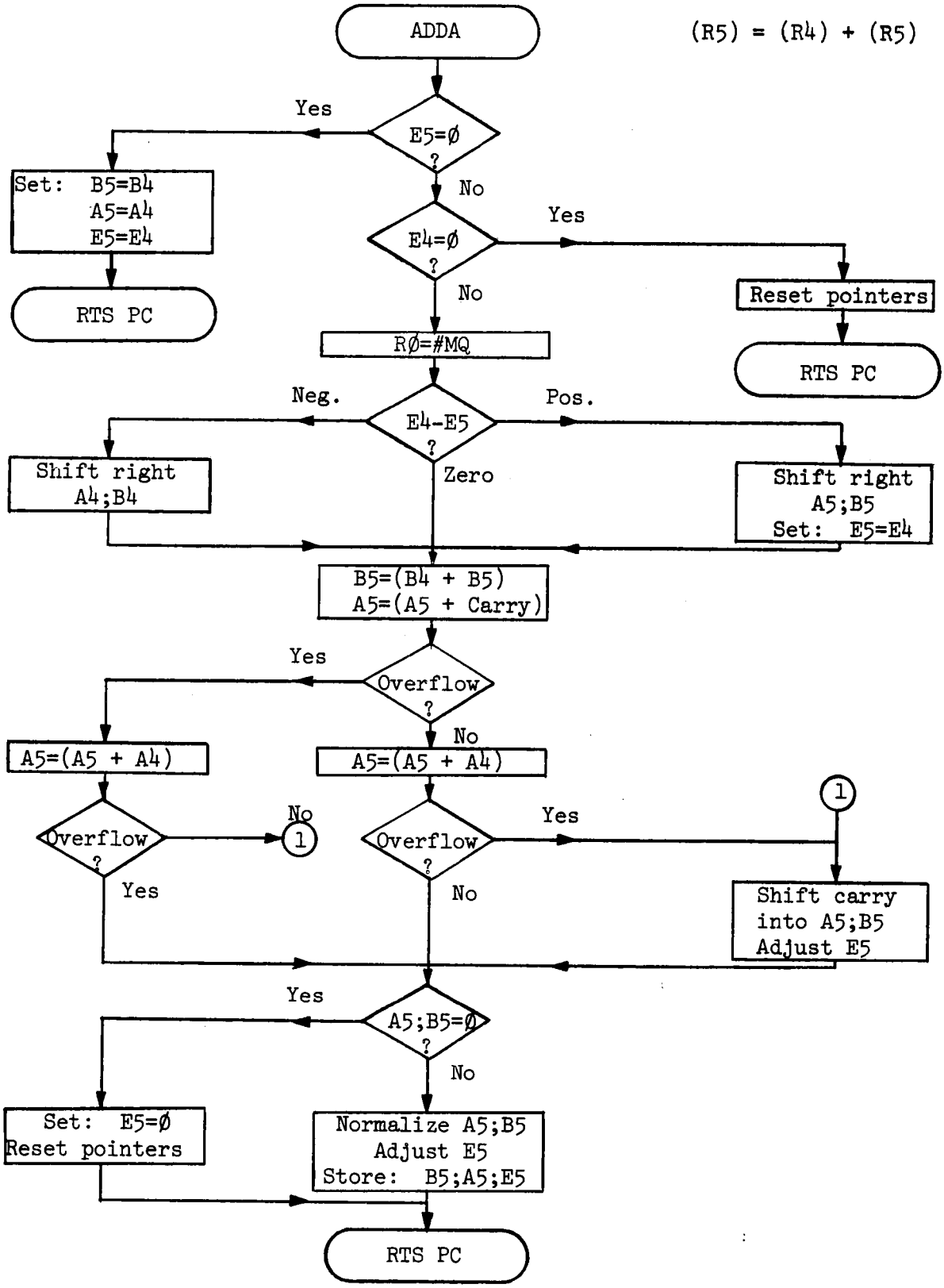

Figure III.7. Flow chart for ADDA floating point subroutine.

overflow occurs, the sum of the high-order fractions are summed and tested to determine if a carry bit should be shifted back into the fraction. A non-zero fraction is normalized, the exponent adjusted accordingly, and the results stored. A zero sum requires that the exponent be set to zero and the pointers adjusted.

MATX. Forms the product of two matrices and stores the resultant matrix at a third location. Matrix dimensions are transferred through three fixed locations (M, N, and P) according to

$$[M*P] = [M*N][N*P]. \tag{3.5}$$

The elements of the first matrix must be prearranged serially by rows and the second matrix serially by columns. The elements of the resultant matrix are arranged serially by rows. The pointer register of the resultant matrix is set to the beginning of that matrix; the pointer registers of the other two matrices are unspecified at the end of the subroutine.

Referring to Figure III.8, the matrix dimensions are stored for future reference and the number of bytes comprising a row is calculated. A pointer to the temporary storage location is set and the starting addresses of two of the matrices are stored. The first operation of the subroutine is to zero the memory location corresponding to the next element of the resultant matrix to be calculated. That element is then calculated by a succession of floating point multiplications and additions. The subroutines MULT and ADDA are specifically designed

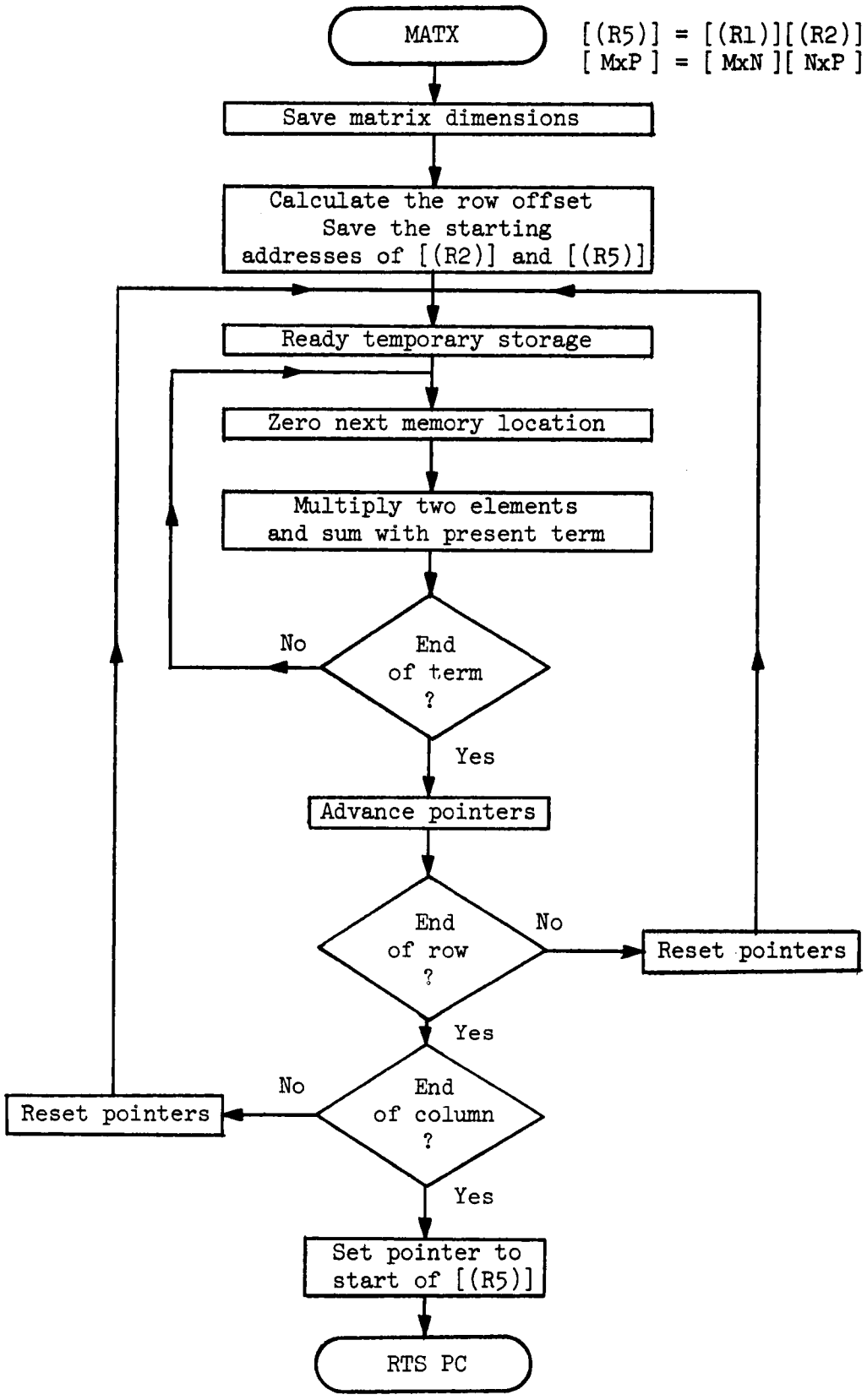

Figure III.8. Flow chart for MATX floating point subroutine.

to be used with MATX. A series of flags and counters determine when each element, row, and column of the new matrix is complete. Pointer adjustments are made corresponding to the state of the operation. The end of the columns indicate the completion of the matrix multiply. The destination pointer is set to the beginning of the resultant matrix and the subroutine exited.

NEGT. Negates a floating point number. The result replaces the original number. The pointer register is unaltered by the operation.

DBWD, DTOA, and DECA. A set of three routines for outputting the decimal approximation of a binary floating point number. DBWD converts the floating point number to two 16-bit words with the decimal point assumed between the words. DTOA makes a decimal approximation of the two words and converts it to ASCII code using the subroutine DECA. DECA outputs the results on the teletype.

The three routines are flowcharted in Figure III.9. The floating point word is first converted to a two word representation with the decimal point assumed between the words. This is done by performing an algorithm adjustment of the exponent. The result is checked and further alterations made if the exponent is negative. The fraction of the floating point word is then shifted the number of spaces indicated by the adjusted exponent to form the double word representation. Control is passed to the DTOA routine which searches for the best decimal approximation using the DECA subroutine. DTOA successively

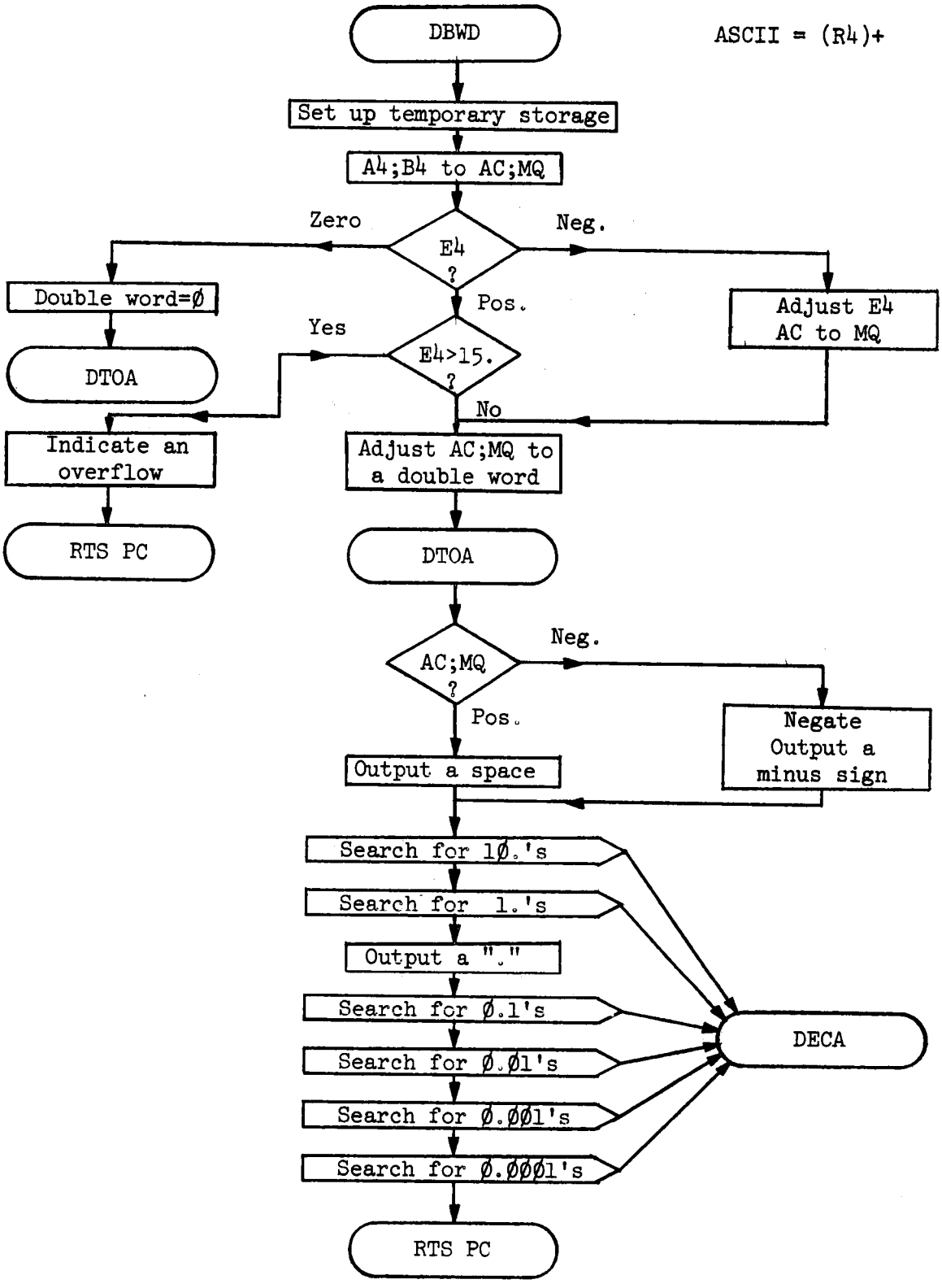

Figure III.9. Flow chart for floating point to ASCII conversion and output.

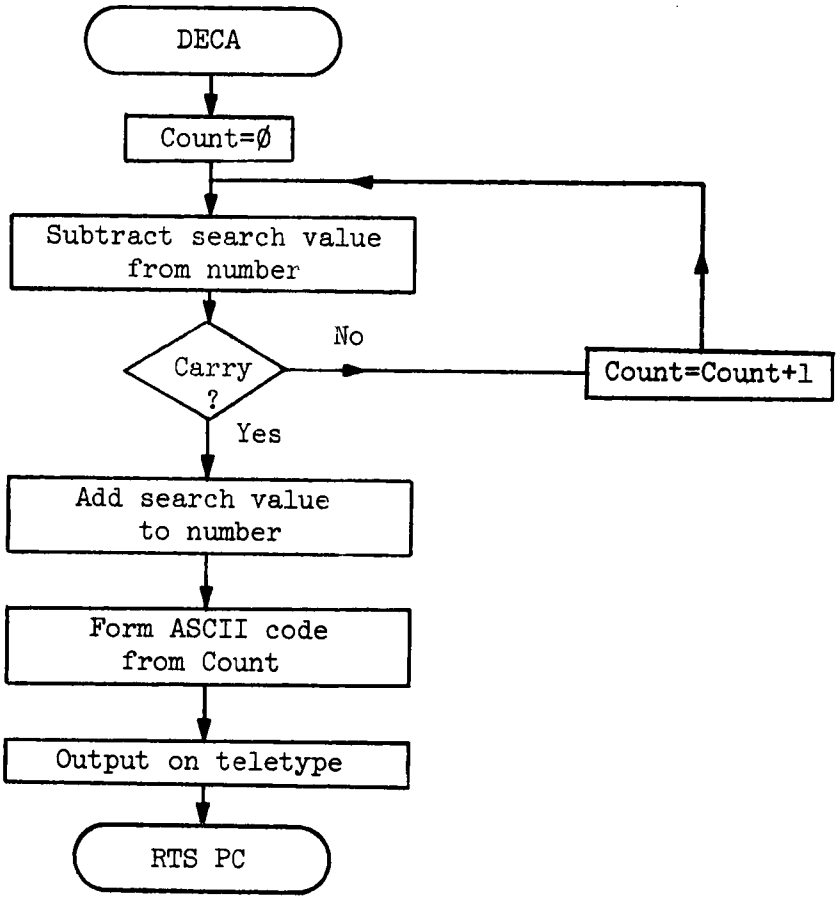

Figure III.9 (continued.

sets up the appropriate registers with the decimal search value. DECA repeatedly subtracts the search value from the word under conversion. When a carry is detected, the search is terminated, the search value added back to the word, and the count value converted to ASCII code. The result is printed by the teletype and control returned to DTOA. The routine makes an approximation to four decimal places.

Numerical examples using the software developed are presented in the following chapter.

CHAPTER IV

ANALYSIS AND APPLICATION OF PARAMETER ESTIMATION

This chapter examines the physical significance of the parameter estimates. Several criteria for judging the "goodness" of an estimate are evaluated and applied to experimental results. A simple method for transforming the discrete system model to a continuous system model is developed for the special case of a second-order systems. Three applications of parameter estimation are presented and the results evaluated.

I. PHYSICAL SIGNIFICANCE OF THE PARAMETER ESTIMATES

Assuming that "good" estimates of the parameters can be obtained using the software developed, the question arises as to the physical significance of these estimates. For the purpose of this and subsequent discussions, consider a single-input, second-order system where both states of the system are available and where one of the states is the output of the system. Referring to Equation (2.11), this special case is represented by

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(k), \qquad (4.1)$$

where a_{ij} and b_i are the parameter estimates. Recall that $x_i(k)$ implies $x_i(kT)$, the i'th state at the k'th sampling interval, T.

State $x_1(k)$ is arbitrarily assumed to be the system output.

Inspection of Equation (4.1) reveals that knowledge of a_{ij} and b_i (i=1.2; j=1,2), the states, and the input of the system at time kT makes possible the prediction of the system states at time (k+1)T. Assuming that M future values of the input, u(k), are available, recursive application of Equation (4.1) will yield the system states for M sampling intervals into the future. This can best be seen by the example below.

Example IV.1

Consider the system shown in Figure IV.1, where the input is sampled at t = 0, 1, 2, ..., i.e. T = 1 second. Find the time response for t = 1 and 2 if the system is initially at states $x_1(0) = 0$, $x_2(0) = 0.5$, and is subjected to a unit step at t = 0, i.e. u(t) = 1 for $t \geq 0$.

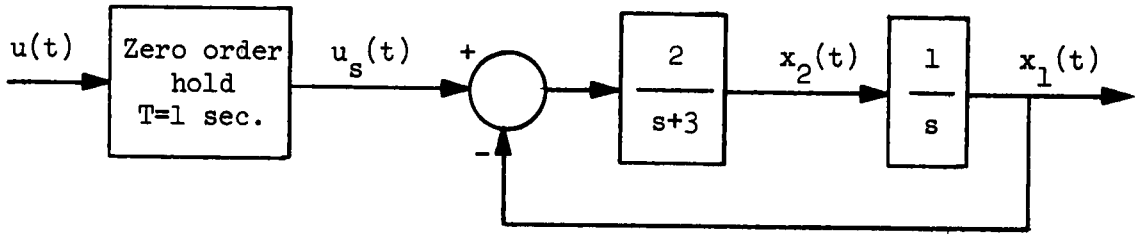

Figure IV.1. Sampled-data system example.

The continuous system model as denoted in Equation (2.1) can be expressed by

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u(t). \qquad (4.2)$$

Direct application of Equations (2.12) and (2.13) for T = 1 second yields the discrete model

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} .60042 & .23254 \\ -.46508 & -.09720 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} -.39958 \\ .46508 \end{bmatrix} u(k). \qquad (4.3)$$

For the intial conditions and the input specified in the problem, the system response is directly calculated as below:

$$\begin{bmatrix} x_1(1) \\ x_2(1) \end{bmatrix} = \begin{bmatrix} .600 & .233 \\ -.465 & -.097 \end{bmatrix} \begin{bmatrix} .000 \\ .500 \end{bmatrix} + \begin{bmatrix} .400 \\ .465 \end{bmatrix} 1.0 = \begin{bmatrix} .516 \\ .416 \end{bmatrix};$$

$$\begin{bmatrix} x_1(2) \\ x_2(2) \end{bmatrix} = \begin{bmatrix} .600 & .233 \\ -.465 & -.097 \end{bmatrix} \begin{bmatrix} .516 \\ .416 \end{bmatrix} + \begin{bmatrix} .400 \\ .465 \end{bmatrix} 1.0 = \begin{bmatrix} .806 \\ .185 \end{bmatrix}.$$

In a similiar fashion $\underline{x}(3)$, $\underline{x}(4)$, ... can be determined. The parameter matrices remain constant (provided the system is time invariant and linear) for any input and any initial state.

II. CRITERIA FOR JUDGING MODEL "GOODNESS"

Several approaches for evaluating the confidence of the parameter estimates are discussed in the following paragraphs. A simple method for converting a discrete model to a continuous model representation is also given.

Evaluation in Discrete Time

The most obvious way to evaluate the estimated system model is to use the procedure described above for generating the system response from the system model. The actual response and the simulated response are then compared, and a subjective judgement made as to the "goodness" of the model. The comparsion can be based on the respective maximums, minimums, overshoot, settling time, etc. A more analytical approach to the problem is to use the criterion of least squares.

The derivation in Chapter II of the sequential least squares parameter estimation algorithm is based on the minization of the cost index given by Equation (2.23). This cost index is the square of the difference between the true value and the predicted value of the system state, or more simply stated, the square of the residual. By summing the squares of the residuals between the actual and the simulated responses over a finite number of points, an analytical comparison can be made. Some guide must now be established for determining an acceptable value for the "sum of squared residuals." To evaluate this sum, consider the system configuration for parameter estimation given in Figure III.2, page 32. All data used for parameter estimation must first be quantized by analog-to-digital convertors. The A/D convertor utilized has a full scale range from -10 volts to +10 volts, which is converted into an eleven bit digital signal. This results in a minimum quantization step of approximately .01 volts. If the simulated response deviates from the actual response by an

average of not more one or two quantization steps, then the estimated model should be more than adequate for any subsequent digital applications. This is the major criterion used for evaluating the experimental results of this thesis.

Evaluation in Continuous Time

Regardless of the "goodness" of the estimated discrete model, its use is restricted to the system sampling interval in effect at the time the estimate was calculated. A transformation to the continuous time domain would greatly enhance the usefulness of the estimate and permit the application of "classical" control techniques. Equations (2.14) and (2.16) present a transformation that requires obtaining the natural logrithm of a matrix. The logrithm of a matrix involves finding its eigenvalues and the corresponding eigenvectors. A matrix is created with the eigenvalues along the major diagonal. It is a simple procedure then to obtain the logrithm of the eigenvalues (as if they are scalar quanties). The resultant matrix is post-multiplied by the matrix composed of the eigenvectors and premultiplied by the inverse of that same matrix.[20] A simple alternative transformation for the special case of a second-order system is developed below.

Consider again the definition for the A matrix of the discrete model given by Equation (2.12) as

$$A = \Phi(T) = \exp(\alpha T). \tag{4.4}$$

Recall that $\Phi(T)$ represents the state transition matrix evaluated

at t = T, the sampling interval. A more useful definition of the state transition matrix and therefore and A matrix is given by

$$A = \Phi(T) = \mathcal{L}^{-1}\{[sI - \alpha]^{-1}\}\Big|_{t=T} , \tag{4.5}$$

where $\mathcal{L}^{-1}$ denotes the inverse Laplace transform of $\{\cdot\}$. Let the continuous model matrix, α, be represented in phase variable form such that

$$\alpha = \begin{bmatrix} 0 & 1 \\ -\alpha_0 & -\alpha_1 \end{bmatrix} , \tag{4.6}$$

Where α_1 and α_0 are the coefficients of the continuous system characteristic equation,

$$s^2 + \alpha_1 s + \alpha_0 = 0 . \tag{4.7}$$

Equation (4.5) can now be expressed as

$$A = \mathcal{L}^{-1}\left\{ \frac{1}{s^2 + \alpha_1 s + \alpha_0} \begin{bmatrix} s+\alpha_1 & 1 \\ -\alpha_0 & s \end{bmatrix} \right\} \Bigg|_{t=T} . \tag{4.8}$$

Completing the square of the characteristic equation yields

$$s^2 + \alpha_1 s + \alpha_0 = [s + \frac{\alpha_1}{2}]^2 + [\sqrt{\alpha_0 - (\frac{\alpha_1}{2})^2}\,]^2 . \tag{4.9}$$

For notational convenience let

$$\gamma = \sqrt{\alpha_0 - (\frac{\alpha_1}{2})^2} \ . \tag{4.10}$$

Equation (4.8) becomes

$$A = \mathcal{L}^{-1}\left\{\begin{bmatrix} \frac{s + \frac{\alpha_1}{2} + \frac{\alpha_1}{2}}{(s + \frac{\alpha_1}{2})^2 + \gamma^2} & \frac{1}{(s + \frac{\alpha_1}{2})^2 + \gamma^2} \\ \frac{-\alpha_0}{(s + \frac{\alpha_1}{2})^2 + \gamma^2} & \frac{s + \frac{\alpha_1}{2} - \frac{\alpha_1}{2}}{(s + \frac{\alpha_1}{2})^2 + \gamma^2} \end{bmatrix}\right\}\Bigg|_{t=T} , \tag{4.11}$$

which transforms into

$$A = \begin{bmatrix} \exp(-\frac{\alpha_1 T}{2})[\cos(\gamma T) + \frac{\alpha_1}{2\gamma}\sin(\gamma T)] \ ; \ \frac{1}{\gamma}\exp(-\frac{\alpha_1 T}{2})\sin(\gamma T) \\ -\frac{\alpha_0}{\gamma}\exp(-\frac{\alpha_1 T}{2})\sin(\gamma T) \ ; \ \exp(-\frac{\alpha_1 T}{2})[\cos(\gamma T) - \frac{\alpha_1}{2}\sin(\gamma T)] \end{bmatrix} . \tag{4.12}$$

Equating corresponding matrix elements gives

$$a_{11} = \frac{1}{\gamma}\exp(-\frac{\alpha_1 T}{2})\sin(\gamma T)[\gamma\cot(\gamma T) + \frac{\alpha_1}{2}] \ , \tag{4.13}$$

$$a_{12} = \frac{1}{\gamma}\exp(-\frac{\alpha_1 T}{2})\sin(\gamma T) \ , \tag{4.14}$$

$$a_{21} = \frac{1}{\gamma}\exp(-\frac{\alpha_1 T}{2})\sin(\gamma T)[-\alpha_0] \ , \tag{4.15}$$

$$a_{22} = \frac{1}{\gamma}\exp(-\frac{\alpha_1 T}{2})\sin(\gamma T)[\gamma\cot(\gamma T) - \frac{\alpha_1}{2}] \ , \tag{4.16}$$

or more simply

$$a_{11} = a_{12}[\gamma\cot(\gamma T) + \frac{\alpha_1}{2}] , \tag{4.17}$$

$$a_{21} = a_{12}[-\alpha_0] , \tag{4.18}$$

$$a_{22} = a_{12}[\gamma\cot(\gamma T) - \frac{\alpha_1}{2}] . \tag{4.19}$$

Solving for α_0 in Equation (4.18) yields

$$\alpha_0 = - \frac{a_{21}}{a_{12}} . \tag{4.20}$$

Equations (4.17) and (4.19) combine such that

$$\alpha_1 = \frac{(a_{11} - a_{22})}{a_{12}} . \tag{4.21}$$

The lengthy procedure for determining the logrithm of a matrix is thus circumvented for second-order systems. A transformation for the discrete matrix B is obtained by using the above results in Equation (2.16). This yields

$$\beta = \alpha[A - I]^{-1}B . \tag{4.22}$$

The derivation is based on a sampling configuration synonymous with Figure IV.1, page 53. For systems with the zero order hold located elsewhere, the transformation holds true only if the sampling interval

is small with respect to the system time constants. This is the assumption made by this thesis in applying the transform to the experimental results. One additional consideration is that the derivation appears to require that the time intervals of the data from which the model is obtained equal the system sampling interval. However, empirical results indicate that the data can be taken at any time interval and still yield correct transformations.

III. APPLICATION OF PARAMETER ESTIMATION

Applications presented in this section are of two categories: (1) estimations based on data obtained using the IBM Continuous Systems Modeling Program (CSMP) to simulate a continuous system; (2) estimations based on data collected in real-time using the software developed. The application to the CSMP simulated data will be presented first to verify the discrete to continuous model transformation, and to demonstrate the capabilities of the program when operating on "noise-free" data. This will be followed by applications of the program to two different systems operating under control of the Sampled-Data Systems Monitoring Program.

Simulated Data Example

The system considered has unity negative feedback with a plant transfer function given by

$$G(s) = \frac{12.09}{s(s + 1.56)} \quad . \tag{4.23}$$

Consequently, its characteristic equation is

$$s^2 + 1.56s + 12.09 = 0 \ . \qquad (4.24)$$

Forty-One data triplets (the input and two states) spaced at .100 second intervals were obtained from the continuous system simulation using CSMP. The parameter estimation program requires that the data for estimation be in analog-to-digital word format. The CSMP data was quantized and stored in the appropriate memory locations. The quantized data used for parameter estimation is given in Figure IV.2. The plot represents the response of the system to a 0→3 volt step input at time zero.

Forty iterations of the parameter estimation program produced the following discrete model matrices:

$$A = \begin{bmatrix} .94129 & .09058 \\ -1.09501 & .80221 \end{bmatrix} \qquad B = \begin{bmatrix} .05851 \\ 1.09386 \end{bmatrix} . \qquad (4.25)$$

Using the transformations of Equations (4.20), (4.21), and (4.22) yields the continuous model matrices:

$$\alpha = \begin{bmatrix} 0 & 1 \\ -12.089 & -1.535 \end{bmatrix} \qquad \beta = \begin{bmatrix} .000 \\ 12.075 \end{bmatrix} . \qquad (4.26)$$

Comparing the parameters in the bottom row of the α matrix with the coefficients of the characteristic equation, Equation (4.24), verifies the transform relationships.

A test of the "goodness" of the discrete model estimate is made using the criterion of least squares. The parameter matrices of

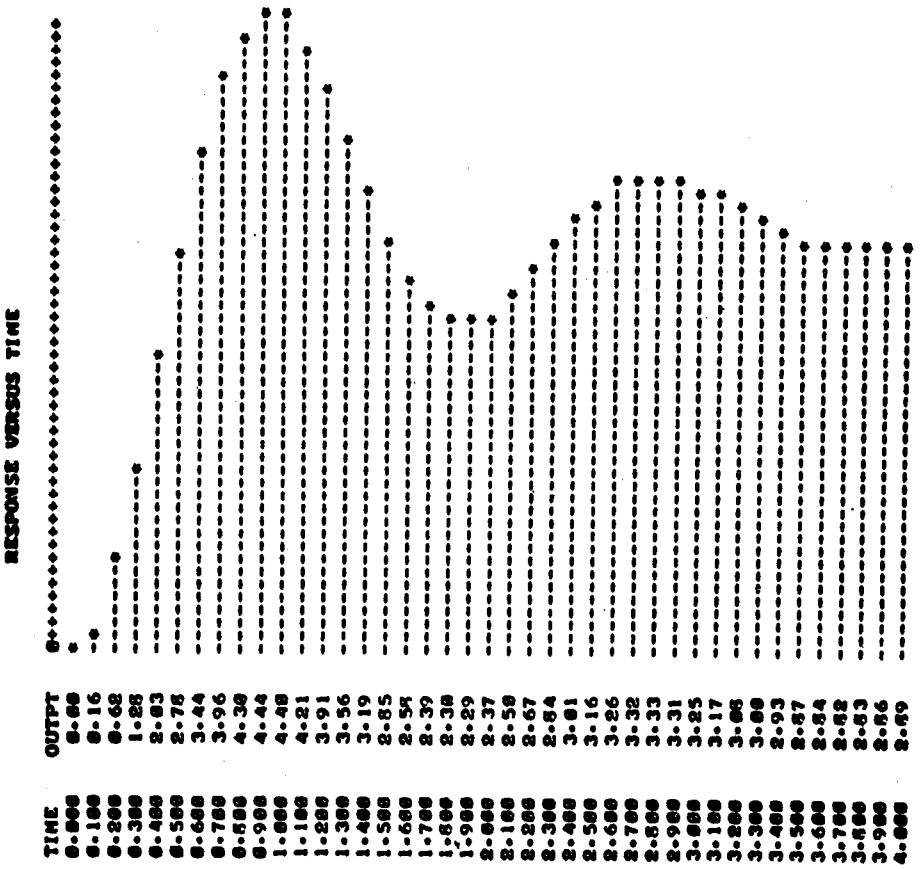

Figure IV.2. CSMP simulation system response for

$$G(s) = \frac{12.09}{s(s + 1.56)} .$$

Equation (4.25) were used to recursively generate the system response as described earlier in this chapter. The results appear in Figure IV.3. The residual at each time interval was squared and then summed with the previous residuals. The total sum of squared residuals based on forty -one points is .0027, or an average error of .0081 per data point; a deviation that is less than one quantization step in magnitude.

Real-Time Data Examples - Laboratory Setup

The schematic diagram of the laboratory arrangement for real -time data acquisition is shown in Figure III.1, page 29. The block labeled SYSTEM is an analog simulation of the system. A Model 194 operational amplifier manifold, manufactured by Analog Devices, Inc., is used to "build-up" the system transfer function. The analog manifold is a chassis with five operational amplifiers, and numerous female receptacals for mounting external circuit elements. The analog simulation configurations used for parameter estimation are described separately for each example.

The real-time data for the following examples is obtained using the Sampled-Data Systems Monitoring Program. For both examples, a sampling interval of two milliseconds was selected. This sampling interval is "short" with respect to the system time constants, hence, the transform techniques of Equations (4.20), (4.21), and (4.22) can be utilized. Three sources of measurement error are associated with the real-time data collection: (1) signal noise; (2) A/D conversion errors; and (3) A/D conversion quantization effect. An additional

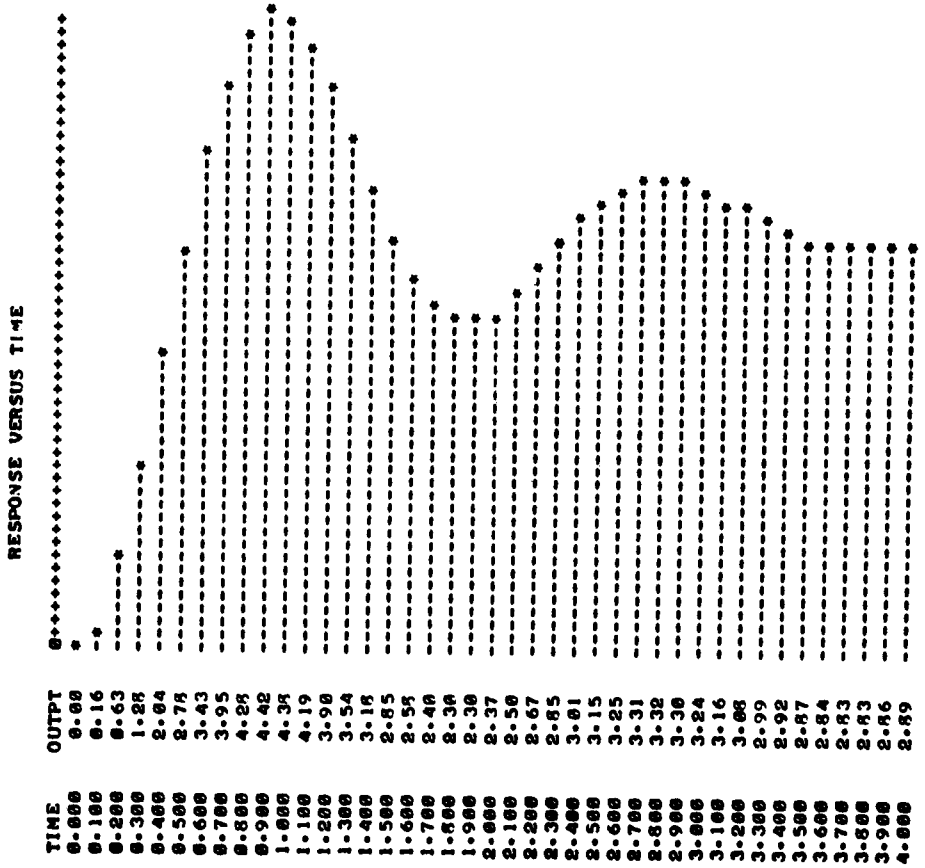

Figure IV.3. Response generated from the parameter estimates, Equation (4.25).

error is incurred whenever the continuous signal exceeds the maximum range of the convertor. The input perturbation must be selected to avoid this situation if an accurate estimation is to be made. The estimations are made using the same data triplet described for the CSMP simulation. Actual hardware connections to the analog system are shown in the respective analog configuration diagrams.

Parameter Estimation of a Type 1, Second-Order System

The system considered has unity negative feedback with a plant transfer function given by

$$G(s) = \frac{24.18}{s(s + 1.67)} \quad . \tag{4.27}$$

The transfer function is derived theoretically from the component values used in the analog manifold setup shown in Figure IV.4. The closed-loop response of the discrete system (T = 2 msec), along with the output data values used for estimation, is shown in Figure IV.5. The system input perturbation is a 0→2 volt step initiated at time zero. Data values are specified for a storage interval of .050 seconds. This interval is arbitrarily selected, based on prior observations, to supply an accurate "picture" of the system response.

Fifty iterations of the parameter estimation program produced the following discrete model matrices:

$$A = \begin{bmatrix} .96618 & .04615 \\ -1.18233 & .89023 \end{bmatrix} \qquad B = \begin{bmatrix} .03363 \\ 1.17502 \end{bmatrix} \quad . \tag{4.28}$$

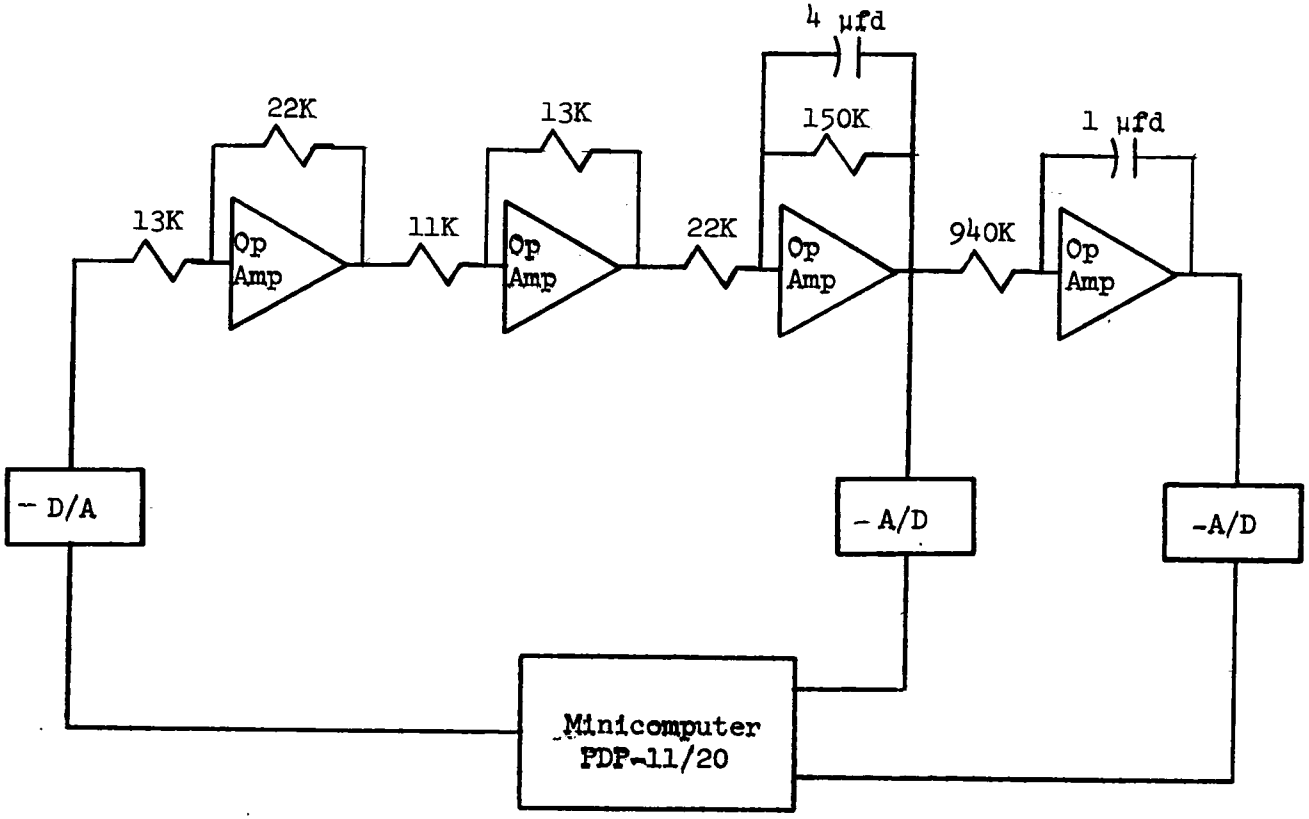

Figure IV.4. Analog manifold configuration for $G(s) = \frac{24.18}{s(s + 1.67)}$.

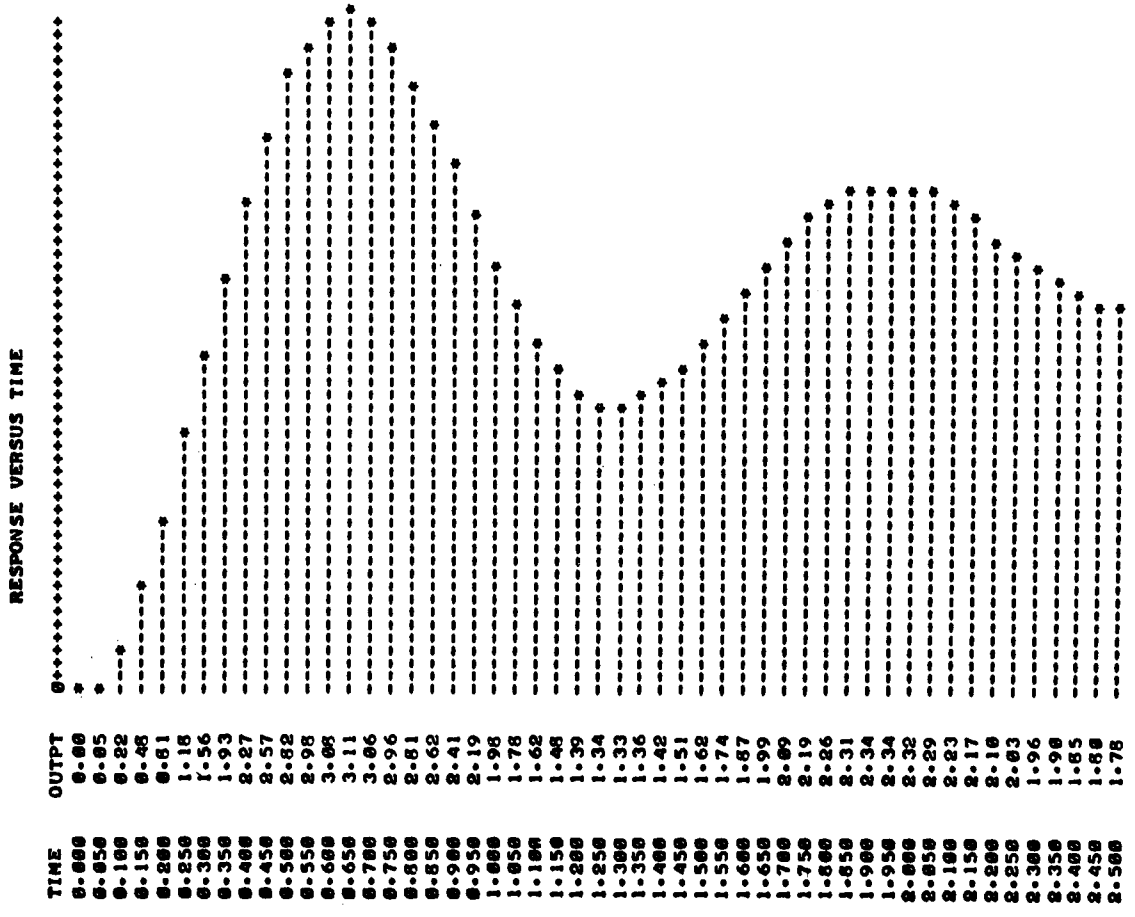

Figure IV.5. Actual system response for $G(s) = \frac{24.18}{s(s + 1.67)}$.

Using the discrete-to-continuous model transformations yields the continuous model matrices:

$$\alpha = \begin{bmatrix} 0 & 1 \\ -25.619 & -1.646 \end{bmatrix} \qquad \beta = \begin{bmatrix} .000 \\ 25.461 \end{bmatrix} . \tag{4.29}$$

The signs of several of the matrix elements were changed to account for the signal inversions introduced by the operational amplifiers. The matrices of Equation (4.28) were used to generate the system response in Figure IV.6. The total sum of squared residuals based on fifty-one points is .0078, or an average error in estimation of .0124 per data point. Based on the minimum A/D convertor quanitization step of .01 volts, this estimate is considered excellent.

Parameter Estimation of a Type 1, Second-Order System with a Zero

The system considered has unity negative feedback with a plant transfer function given by

$$G(s) = \frac{10.75(s + 5.00)}{s(s + 1.67)} . \tag{4.30}$$

The analog manifold realization is shown in Figure IV.7. Figure IV.8 shows the closed-loop response of the system (T = 2 msec) to a step input of 0→1.8 volts. Estimation data is specified at intervals of .050 seconds.

Fifty iterations of the parameter estimation program produced the following discrete model matrices:

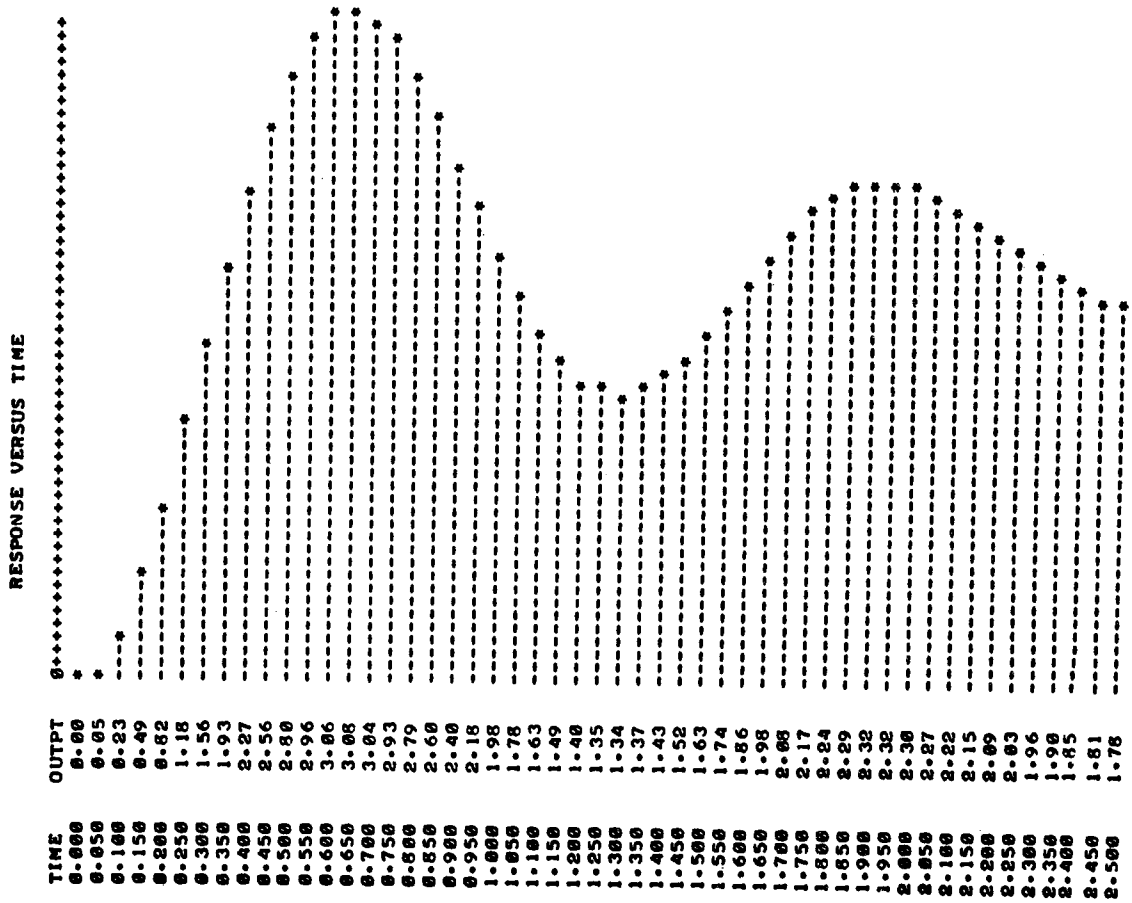

Figure IV.6. Response generated from the parameter estimates, Equation (4.28).

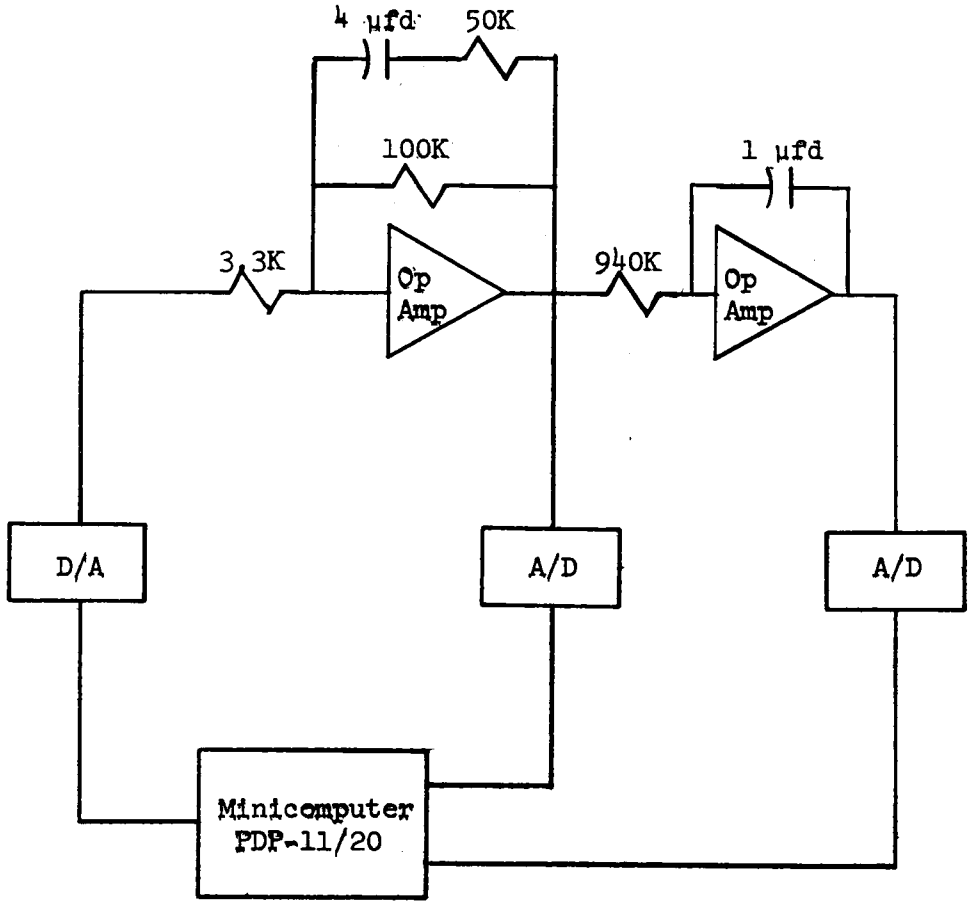

Figure IV.7. Analog manifold configuration for $G(s) = \frac{10.75(s + 5.00)}{s(s + 1.67)}$.

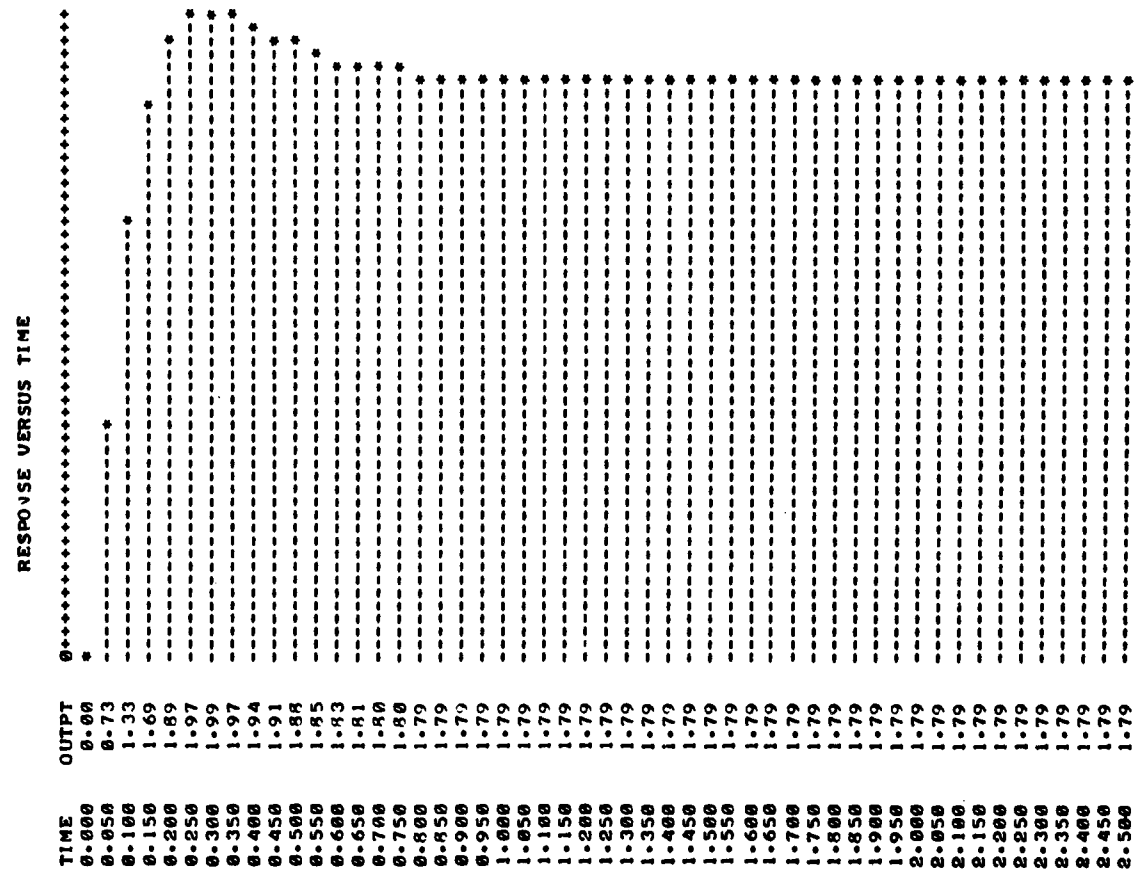

Figure IV.8. Actual system response for $G(s) = \frac{10.75(s + 5.00)}{s(s + 1.67)}$.

$$A = \begin{bmatrix} .60851 & .02155 \\ -5.24592 & .36267 \end{bmatrix} \qquad B = \begin{bmatrix} .39603 \\ 5.28031 \end{bmatrix} . \tag{4.31}$$

The transform method of Equations (2.14) and (2.16) yielded the continuous model matrices:

$$\alpha = \begin{bmatrix} -10.974 & 1 \\ -62.877 & 0 \end{bmatrix} \qquad \beta = \begin{bmatrix} 11.056 \\ 63.510 \end{bmatrix} . \tag{4.32}$$

The generated system response, Figure IV.9, resulted in a total sum of squared residuals, based on fifty-one points, of .0738, or an average error in estimation of .0380 per data point. Possible causes for this larger deviation are: (1) saturation of the A/D convertor at the first stage amplifier; (2) an insufficient number of data points in the dynamic range of operation; (3) increased A/D convertor errors with the faster system; and (4) a more complex system.

IV. SUMMARY

The results of this chapter demonstrate that parameter estimation using a minicomputer is feasible. Excellent approximations of the system response can be generated from the parameter estimates as shown in Figures IV.3, IV.6, and IV.9, pages 64, 69, and 73. At present the program accomodates only the second-order state model system, but can easily be extended to higher order systems and input-output models.

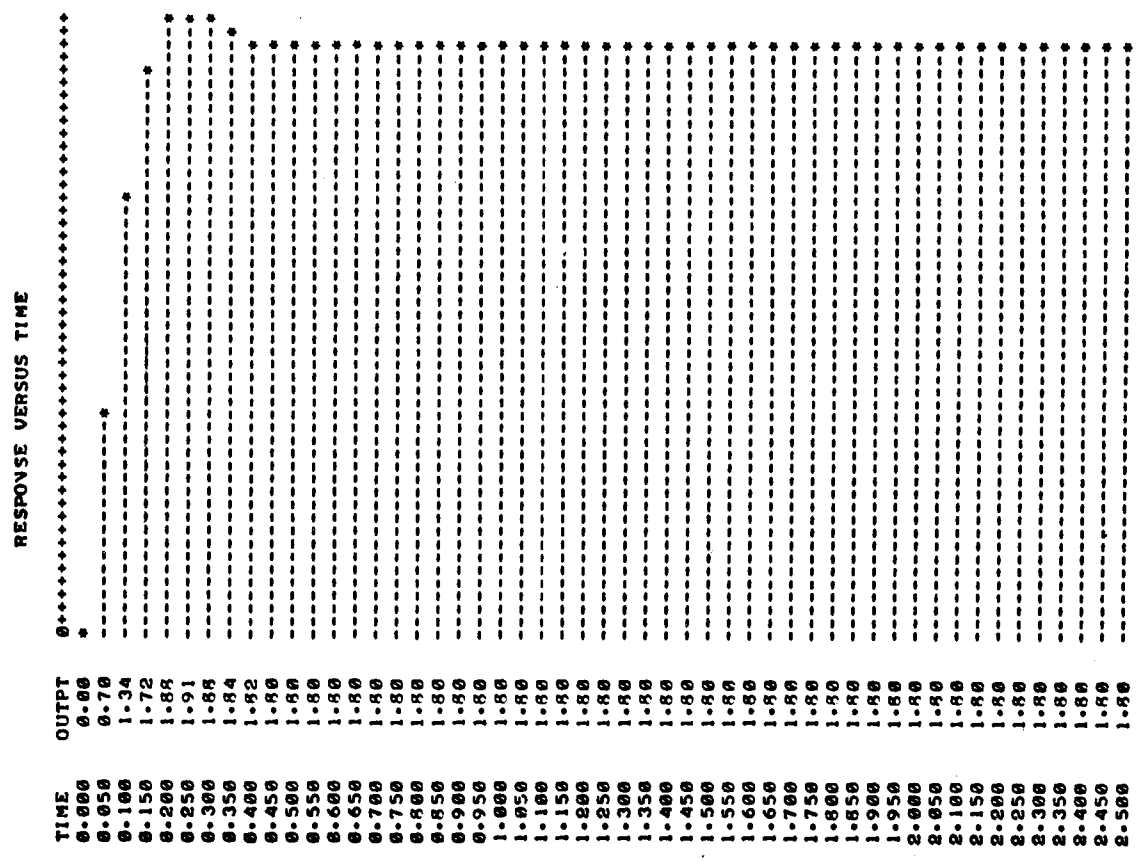

Figure IV.9. Response generated from the parameter estimates, Equation (4.31).

CHAPTER V

CONCLUSIONS

An assembly language program has been developed for realizing parameter estimation on a minicomputer (Digital Equipment Corporation's PDP-11/20). The algorithm employed is that of sequential least squares parameter estimation. A derivation of the algorithm is included along with a description of the software developed. Several other methods for identification are reviewed.

Example problems using both simulated and actual data are presented. As is expected, the method worked well on the simulated data. The "goodness" of the estimates based on actual data, however, deteriorated with the complexity of the system. This is an indication that some of the a priori assumptions are contradicted by the experimental data, e.g. zero mean noise. Another limiting factor is the capability of the analog-to-digital convertors to make consistently accurate conversions.

In the example problems, the parameter estimation program was used in a situation that is considered to be off-line, i.e. the calculations are made after all measurements are complete. The program was developed, however, to be used in on-line applications, e.g. simultaneous identification-control. It was for this reason that a set of fast floating point subroutines were required and hence developed. These special subroutines make possible the short execution time of

15 milliseconds per iterative estimation.

Two other assembly language programs were written for the purposes of this thesis. The first, as described in Appendix B, is a general purpose software package offering the user complete flexibility in specifying inputs, sampling rates, and teletype printer-plot outputs. Several of these printer-plots are used in Chapter IV. The second program, not described in this thesis, was used to generate data from the parameter estimation matrices and calculate the sum of squared residuals.

Further work should be oriented toward appling the parameter estimation program to the identification-control problem. Also, additional work needs to be done in the areas of sampling (data) interval selection, determining the optimum input for identification, and investigating the capabilities and limitations of the program.

The parameter estimation program is written for the special case of the second-order system where both states are obtainable. The program should be generalized to accomodate higher order systems and to provide input-output model descriptions.

Finally, the relationships between the individual parameter estimation matrix elements as a function of the sampling interval should be explored along with the problem of relating those elements to the continuous-time transfer function coefficients.

LIST OF REFERENCES

LIST OF REFERENCES

1. K. J. Åström and P. Eykhoff, "System Identification," Preprints of the 2nd Prague IFAC Symposium, Identification and Process Parameter Estimation, Survey Paper, (June 1970), paper 0.1.

2. K. J. Åström, "On the Achievable Accuracy in Identification Problems," Proceedings IFAC Symposium, Indentification in Automatic Control Systems, Prague, (1967), paper 1.8.

3. I. Gustavsson, "Comparison of Different Methods for Identification of Linear Models for Industrial Processes," Preprints of the 2nd Prague IFAC Symposium, Identification and Process Parameter Estimation, (June 1970), paper 11.4.

4. C. M. Woodside, "Estimation of the Order of Linear Systems," Preprints of the 2nd Prague IFAC Symposium, Indentification and Process Parameter Estimation, (June 1970), paper 1.7.

5. D. W. Clarke, "Generalized Least Squares Estimation of the Parameters of a Dynamic Model," Proceedings IFAC Symposium Indentification in Automatic Control Systems, Prague, (1967) paper 3.17.

6. K. J. Åström and T. Bohlin, "Numerical Identification of Linear Dynamic Systems from Normal Operating Records," Proceedings IFAC Symposium, Self-Adaptive Control Systems, Teddington, (1965).

7. V. Peterka and K. Smuk, "On-Line Estimation of Dynamic Model Parameters from Input-Output Data," Proceedings Fourth IFAC Symposium, Warsaw, (1969), paper 26.1

8. G. M. Jenkins and D. G. Watts, Spectral Analysis and its Applications, San Fransisco: Holden-Day, (1968).

9. D. Bell and A. W. J. Griffin (Editors), Modern Control Theory and Computing, London: McGraw-Hill, (1969).

10. D. G. Schultz and J. L. Melsa, State Functions and Linear Control Systems, New York: McGraw-Hill, (1967).

11. D. Graupe, Identification of Systems, New York: Van Nostrand Rienhold Co., (1972).

12. R. G. Brown and J. W. Nilsson, Introduction to Linear Systems Analysis, New York: John Wiley and Sons, Inc., (1962).

13. A. P. Sage, Optimum Systems Design, Englewood, N. J.: Prentice Hall, (1968).

14. A. E. Albert and L. A. Gardner, Stochastic Approximation and Nonlinear Regression, Cambridge, Mass.: M.I.T. Press, (1967).

15. R. C. K. Lee, Optimal Estimation, Identification, and Control, Cambridge, Mass.: M.I.T. Press, (1964).

16. PDP-11 Peripherals and Interfacing Handbook, Digital Equipment Corporation, Maynard, Mass., (1971).

17. PDP-11 Processor Handbook, Digital Equipment Corporation, Maynard, Mass., (1971).

18. PDP-11 Paper Tape Software Programming Handbook, Digital Equipment Corporation, Maynard, Mass., (1972).

19. M. Cuenod and A. P. Sage, "Comparison of Some Methods Used for Process Identification," Automatica, Vol. 4., pp. 235-269, (1968).

20. P. E. Vowels, "On the Logarithm, Circular and Hyperbolic Functions of a Matrix," Matrix and Tensor Quarterly, Vol. 17, No. 2, (December 1966), pp 60-62.

APPENDIXES

APPENDIX A

THE PDP-11/20 MINICOMPUTER AND ITS PERIPHERALS

The following sections give a general description of the Digital Equipment Corporation (DEC) PDP-11/20 minicomputer and its peripherals. For a more detailed description, see references No. 16 and No. 17 given in List of References.

A minicomputer is somewhat arbitrarily defined to be a general-purpose, programmable, digital computer that sells in its basic form for under $20,000. Its popularity is largely a reflection of the growth in use of computers dedicated to a specific task. Digital process control is one example of its many applications.

I. THE PDP-11/20

The PDP-11/20 is a 16-bit, general-purpose, parallel logic computer. All communication between system components is done on a single, bidirectional, high-speed bus called the Unibus. This permits the addressing of device registers as memory, hence all PDP-11/20 instructions are available for input/output (I/O) operations. The maximum transfer rate on the Unibus is 2,500,000 16-bit words per second.

Peripheral devices operate on a multi-line priority interrupt system. Polling of devices is eliminated since the interrupt hardware selects and begins executing the appropriate service routine automatically. A device's service routine priority may be set independent of the hardware

interrupt priority.

Most efficient use of the minicomputer can be realized when it is programmed in assembly language. The advantage of using assembly language is fast execution time, efficient utilization of memory, and access to the entire instruction repertoire of the computer. These benefits are especially pronounced when working with real-time control applications.

II. PERIPHERALS

Extended Arithmetic Element

The extended arithmetic element (KE-11A) is an option which performs multiplication, division, multiple position shifts, and normalization significantly faster than software routines. It connects directly to the Unibus and is programmed as a peripheral. The KE-11A performs the following operations:

Multiply. Two 16-bit numbers are multiplied to give a 32-bit product.

Divide. A 32-bit dividend is divided by a 16-bit divisor to give a 16-bit quotient and a 16-bit remainder. The sign of the remainder is always the same as the sign of the dividend, unless the remainder is zero. An overflow is indicated if more than 16-bits would be required to express the quotient.

Multiple Shifts. A 32-bit number is shifted either left or right the number of places specified by a count. A positive count generates a

left shift; a negative count generates a right shift. There are two types of shift operations. In a logical shift, zeros always fill the vacated positions. An arithmetic shift has two modes of operation. When shifting left, zeros fill the vacated positions and the most significant bit of the number never changes. When shifting right, the most significant bit is replicated.

Normalize. A 32-bit number is arithmetically shifted left until the two most significant bits are different. A count is kept of the number of positions shifted. For the three exceptions to the rule, see reference No. 17.

Programmable Real-Time Clock

The programmable real-time clock (KW11-P) provides scheduled real-time interrupts at specified intervals. Two crystal-controlled clock rates of 100 KHz and 10 KHz are available along with 60 Hz line frequency and an external clock input. The clock rate and the mode of operation are selected by setting specific bits in the control and status register. In the repeated interrupt mode, a 16-bit synchronous binary counter is decremented at the programmed clock rate until a count of zero is obtained. At this time an interrupt occurs. The counter is automatically reloaded from a 16-bit data buffer and the count resumed while the service routine is being executed.

Analog-to-Digital Conversion

The analog-to-digital (A/D) convertor (AD01-D) employed is a

multiplexed four-channel data acquisition device. It provides 11-bit bipolar digitization of the analog signal at its input. Conversion time is approximately 35 μsec with an aperature of 100 nanoseconds. Full scale range is ± 10.0 volts.

Conversion may be initiated under program control by setting bit Ø in the A/D status register. Bit 7 is automatically set upon completion of conversion. The digitized analog voltage is made available to the program through a 16-bit data register. Overload protection is provided up to ± 20 volts.

Digital-to-Analog Conversion

The digital-to-analog (D/A) convertor is a peripheral device capable of converting a 12-bit, 2's complement digital word into the equivalent analog voltage. Full scale output voltage is trimpot adjustable from ± 1 volt to ± 10 volts. The D/A conversion is initiated by loading a D/A buffer register with the desired value. Settling time is a maximum of 20 μsec for a full scale step change.

Teletype

The standard ASR 33 teletype can be used to type in or print out information at a rate of up to ten characters per second. It is also equipped with a low speed paper tape punch and reader which operates at the same rate. Data input may be accomplished under either program or interrupt control. Data output is achieved similarly.

APPENDIX B

SAMPLED-DATA SYSTEMS MONITORING PROGRAM

The Sampled-Data Systems Monitoring Program (SDSMP) enables a user to investigate various aspects of digital control without a formal knowledge of the computer's assembly language. The package has been written to accept a set of 16 single-character commands which give the user complete flexibility in specifying inputs, sampling rates, and teletype printer-plot outputs. The assembly language listing with documentation is included in Appendix C.

The main features of SDSMP are: (1) the availability upon request of a hard copy of virtually any signal in the system; (2) an optional input source mode, whereby any signal generating source may be used as the system's input perturbation; (3) immediate response to any keyboard command. This includes altering operating conditions during printouts without disturbing the printout.

I. OPERATING PROCEDURE AND SYSTEM COMMANDS

Initial Operating Procedure

Hardware Connections. Four hardware connections are required before operation can begin. They are: (1) connect the system output to A/D Channel No. Ø; (2) any signal the user selects may be connected to A/D Channel No. 1; (3) the optional input source, if desired, is connected to A/D Channel No. 2; and (4) D/A Channel No. Ø is con-

nected to the input of the system. The signal generated on this line is the system error given by

$$\text{ERROR} = \text{INPUT} - \text{OUTPUT}. \tag{B.1}$$

Loading. Enter the address of the absolute loader (77500_8) into the switch register. Load the binary tape for SDSMP into the high-speed reader, move the HALT switch to the ENABLE position, and then depress the START switch. The binary tape will be read into memory.

Starting. SDSMP is automatically started upon completion of the loading process. Should the program be halted for any reason during operation, it may be restarted at address 1000_8.

Initial Dialogue. Upon starting, the program will print out a listing of the available commands. After completing this printout, the program indicates its readiness to accept commands by printing an asterist(*). Several different commands may be issued during the listing without disruption. The sequence may be terminated at any time by typing a K or one of several other commands. For further details see the description of the individual commands in Table B.1.

Initial Operating Conditions. The set of conditions present when the program is started are summarized as follows: (1) a Ø.Ø volt amplitude step is applied to the system; (2) the interrupt interval is set at one millisecond; (3) the sampling interval counter is set to ten giving an effective sampling interval of ten msec; (4) the effective data storage interval is 100 msec; and (5) the number of data points is ini-

TABLE B.1

INPUT/OUTPUT COMMANDS FOR THE SAMPLED-DATA SYSTEMS MONITORING PROGRAM (SDSMP)

Command	Format	Description
A	A±XXX↓	Applies a step input of amplitude ±XXX to the system upon receipt of a carriage return. SDSMP interprets the character string as a decimal number with an implied decimal point two places to the left of the carriage return (↓). Examples: A5ØØ↓=5.ØØ volts; A-7↓=-Ø.Ø7 volts; AØ↓=Ø.Ø volts. Use of a plus sign (+) is optional. The amplitude is assumed positive unless preceded by a minus sign (-). Execution of this command will immediately terminate any OIS.
B	B	Negates the present system input. Terminates OIS if in progress and uses the negative of the last value of the OIS as the new input amplitude.
C	C	Optional Input Source (OIS). Interprets the signal appearing at A/D Channel #2 as the system input. Any signal generating device may be used provided the signal does not exceed the range of the A/D convertor (-1Ø.Ø volts to 9.99Ø2 volts).
D	D±XXX↓	Data storage request. Terminates OIS. Applies a step input of amplitude ±XXX (See A command for a description of the format) to the system and stores the number of data points specified by the N command at the interval specified by the S command.
F	F	Optional input source data storage request. Store data points per the D command with the exception that the collection is done in the OIS mode of operation. The F command assures that the system is operating in OIS mode before storage begins.

TABLE B.1 (continued)

Command	Format	Description
N	NXXX↓	Designates the number of data points to be stored and/or output. SDSMP interprets XXX as any integer number with the stipulation that it be less than or equal to 1Ø1.
P	P	Executes the Sequential Least Squares Parameter Estimation (SLSPE) program. Terminates all data storage and data output. Prevents the execution of any command during its execution. Makes the number of iterations designated by the Q command, and then outputs that number and the final estimated values of the parameters. Relinquishes control to SDSMP after the output sequence. The physical system does remain under control of SDSMP during SLSPE.
Q	QXXX↓	Designates the number of iterations to be made by the SLSPE program. XXX is any integer number less than the integer number specified by the N command.
R	RXXXXX↓	Interrupt interval of the computer. SDSMP relinquishes control to the clock service routine every R/1ØØ msec. A/D conversions are made and the sampling interval counter and data storage counters are interrogated. Appropriate action is taken. XXXXX is any integer number greater than 35, the maximum execution time of the clock service routine.
T	TXXXXX↓	System sampling interval. A new system error signal is calculated and applied once every T*R/1ØØ msec. The error signal is calculated by ERROR = INPUT - OUTPUT. XXXXX is any integer number.
S	SXXXXX↓	Data storage interval. Data is stored per the D or F commands once every S*R/1ØØ msec. The S command does not initiate data storage. XXXXX is any integer number.

TABLE B.1 (continued)

Command	Format	Description
I	I	Printer-plots the input signal as stored by the D or F commands. The number of points output is determined by the latest N command. This number must be less than or equal to the number of points stored. Issuance of the command terminates any output sequence and/or data collection. Plots the data values versus time. Prints the data values, the time scale, and the system sampling interval.
E	E	Printer-plots the applied error signal. See I command for further details.
O	O	Printer-plots the system's output signal. See I command for further details.
X	X	Printer-plots any signal appearing at A/D Channel #1. See I command for further details.
K	K	Terminates any output sequence. Will also terminate the input command with arguments before they are executed.

COMMENTS: The A, B, C, N, Q, R, T, and S commands may be issued at any time without altering an output sequence. The K command when used to terminate an A, N, Q, R, T, or S command before execution also will not affect the output. All other commands, including illegal characters, will terminate an output sequence immediately. An illegal character is any character used as a command that is not designated as a legal command. Receipt of such a character will elicit a question mark (?) from SDSMP. Likewise any character following an A, N, Q, R, T, or S command other than the sign or a decimal number will have the same effect as the K command and elicit the same response. SDSMP indicates its readiness to accept commands by printing an asterisk(*).

tially set at forty-one.

Input/Output Commands

Table B.1 summarizes the various commands made available by SDSMP. The command character, its format, and a brief description are given. An X when used as a format character denotes any decimal number. A "↓" denotes a user invoked carriage return. SDSMP will always indicate its readiness to accept another command by printing an asterisk (*).

II. PROGRAM DESCRIPTION

The Sampled-Data Systems Monitoring Program operates strictly on a priority interrupt basis. The execution of any section of the program, with the exception of the initializations, is always in response to an interrupt from the real-time clock, the teletype keyboard, or the teletype printer. The clock service routine is described in Chapter III. The teletype keyboard (TTY-IN) service routines, the teletype printer (TTY-OUT) service routine, and the associated subroutines are described in the following paragraphs. For a more detailed picture of how the various functions are performed, see the documented assembly language listing in Appendix C.

TTY-OUT Service Routine

In response to a TTY-OUT interrupt, the TTY-OUT service routine disables further TTY-OUT interrupts, and signals all other service

routines that an output sequence is in progress. The listing of the available Input/Output (I/O) commands is printed by the teletype. Upon completion, the flag indicating an "output in progress" is cleared and the service routine exited via the standard command terminator.

TTY-IN, Executive

All I/O commands are identified by the TTY-IN Executive, and action taken accordingly. The Executive is divided into five sections: (1) identification and execution of those commands not pre-empting any output sequence; (2) termination of any output sequence in progress; (3) identification and execution of any remaining legal command; (4) a termination of any illegal command; and (5) the standard command terminator, consisting of a carriage return, line feed, and an asterisk.

Command indentification is accomplished by a series of comparisons between the ASCII code for the command character and the character received from the teletype keyboard. When a match is found, the command is either immediately executed or, in the case of all commands requiring an argument, the TTY-IN interrupt trap vector is pointed toward another TTY-IN service routine. If the keyboard character is not a command that can be executed during an output sequence for reasons of conflict in register usage, then any output sequence in progress is terminated. This is accomplished by first determining where in the output sequence program the interrupt occurred. If it occurred during a subroutine then a pseudo return from subroutine is executed along with the pseudo return from interrupt. In this manner, all further

execution of that output program is terminated until called again.

Should any keyboard character remain unidentified throughout the command character search, then that keyboard character is deamed not legitimate. A question mark is output on the teletype and the standard command terminator executed.

TTY-IN, Data Changer

Any command requiring a numerical argument initializes a set of pointers, and redirects the TTY-IN interrupt trap vector to the Data Changer service routine. Here a string of numeric characters, optionally preceded by a sign and terminated by a carriage return, are converted to binary numbers and placed in the appropriate memory location. Should any character other than the numbers zero through nine and a plus or minus sign be received by this service routine, the routine is immediately exited via the illegal character sequence of the TTY-IN Executive. The Data Changer also makes provisions for converting a binary word to an A/D word and initiating data collection.

Data Storage and Data Output

The ability to store data in real-time from a dynamic system and then make available a hard-copy of that data to the user is the prime motivation for SDSMP. The techniques employed to accomplish this are discussed below.

Data Storage. The storage of data from a system may be accomplished using either a step input of user specified amplitude or any

signal generating device as the system perturbation. A special feature of the step input is that it is initiated from the present state of the system to the amplitude specified. This may be in either a positive or a negative direction. The step input mode of operation utilizes the Data Changer routine as described above. Either data storage request initializes the registers to be used as pointers to the data tables. The status quo interrupt interval, sampling interval, and storage interval are stored for later use in a data output.

Data Output. Printer-plots of any one of four different signals are available upon request. The choices are described in Table B.1, page 86. The first step in all cases is to indicate an output in progress and to lower the priority of the output program. The lowering of the priority level permits certain commands to be executed while the output is taking place. Any data collection in progress is terminated due to conflicts in register usage. A series of short routines are executed to determine the maximum and minimum data values, their absolute difference, the quanitization step for the plot based on this difference, and the data storage interval in milliseconds. The universal portion of the printer-plot header is output on the teletype. A search is made to determine which one of the four signals is to be plotted. The data column header is appropriately printed.

The next step is to print the ordinate axis. The axis indicates the negative and positive portions of the subsequent plot and the approximate location of zero. The plot of the data values is made and

their corresponding magnitudes and storage times are printed. The printer-plot is terminated with a printout of the system's sampling interval.

Subroutines

Four subroutines were developed explicitly for SDSMP. Their purpose and features are summarized in Table B.2.

TABLE B.2

SUBROUTINES FOR THE SAMPLED-DATA
SYSTEMS MONITORING PROGRAM

Name	Purpose
PRINT	Outputs a string of ASCII characters on the teletype. The starting address of the string is stored in one register and the terminating address is contained in another.
OCDIG OCTS	Outputs on the teletype, the sign and magnitude of the octal number stored in a specified register. The output format is as specified by the condition code designated by the main program. One format is for time values specified in milliseconds and output in seconds, and the other is for data values specified in hundredths of a volt and output in volts.
PLOT	Outputs a series of dashes corresponding to the magnitude of the number stored in the specified register. The plot quantization step and minimum valve must be prespecified.

APPENDIX C

ASSEMBLY LANGUAGE LISTING FOR SAMPLED-DATA SYSTEMS MONITORING PROGRAM

The assembly language program used for data acquisition is given in the following pages. For an explanation of its purpose and uses, see Appendix B.

```
;SAMPLED-DATA-SYSTEMS MONITORING PROGRAM
;
;ELECTRICAL ENGINEERING DEPARTMENT
;     UNIVERSITY OF TENNESSEE
;
;PROGRAMMER:  CARL A. NEHMER
;DATE:  JULY 1973
;
;ASSIGN NAMES TO REGISTERS AND BUFFERS
        RØ=%Ø
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        SP=%6
        PC=%7
        PS=177776
        CRS=172540
        BUF=CRS+2
        DABØ=176760
        DAB1=DABØ+2
        ADSR=176770
        ADBR=ADSR+2
        DIV=177300
        AC=DIV+2
        MQ=DIV+4
        MUL=DIV+6
        TKS=177560
        TKB=TKS+2
        TPS=TKS+4
        TPB=TKS+6

;SET UP VECTOR ADDRESSES
.=6Ø            ;TTY-IN VECTOR ADDRESS
TEXEC           ;SERVICE ROUTINE
24Ø             ;PRIORITY 5
.=64            ;TTY-OUT VECTOR ADDRESS
INITL           ;SERVICE ROUTINE
14Ø             ;PRIORITY 3
.=1Ø4           ;REAL-TIME PROGRAMMABLE CLOCK
CLOCK           ;SERVICE ROUTINE
34Ø             ;PRIORITY 7

;INITIALIZE REGISTERS, BUFFERS, AND COUNTERS
.=1ØØØ                  ;STARTING ADDRESS
START:  MOV #.,SP       ;SET UP STACK POINTER
        CLR INPUT       ;INPUT AMPLITUDE
        CLR NDP2        ;NUMBER OF DATA POINTS COUNTER
        CLR SIGN        ;INPUT SIGN INDICATOR
        CLR ECHO        ;OUTPUT IN PROGRESS INDICATOR
        CLR OISI        ;OPTIONAL INPUT SOURCE INDICATOR
```

```
        MOV #1ØØØ,ADSR  ;SET UP A/D STATUS REGISTER
        MOV #1ØØ.,BUF   ;PROGRAMMABLE CLOCK DATA BUFFER
        MOV #1ØØ.,#BUF2 ;INTERRUPT RATE STORAGE
        BUF2=.-2
        MOV #1Ø.,#TSR   ;SAMPLING RATE
        TSR=.-2
        MOV #1,TSR2     ;SAMPLING RATE COUNTER
        MOV #1Ø.,#RSQ   ;NUMBER OF P.E. ITERATIONS
        RSQ=.-2
        MOV #111,CRS    ;CLOCK STATUS REGISTER
        MOV #1ØØ.,#DSR  ;DATA STORAGE RATE
        DSR=.-2
        MOV #41.,#NDP   ;NUMBER OF DATA POINTS
        NDP=.-2
        MOV #1ØØ,TKS    ;TTY-IN STATUS REGISTER
        MOV #1ØØ,TPS    ;TTY-OUT STATUS REGISTER

LOOP:   BR LOOP         ;WAIT FOR INTERRUPTS HERE

;SERVICE ROUTINE:  TTY-OUT, INITIAL DIALOGUE
INITL:  CLR TPS         ;DISABLE TTY-OUT INTERRUPT
        INC ECHO        ;OUTPUT-IN-PROGRESS INDICATOR
        MOV #MS1,R1     ;START OF MESSAGE
        MOV #MS2,R2     ;END OF MESSAGE
        JSR PC,PRINT
        CLR ECHO
        JMP NEXT

;SERVICE ROUTINE:  CLOCK
CLOCK:  MOV #Ø,ADSR     ;A/D CHANNEL #Ø
        TSTB ADSR       ;DETERMINE THE VALUE
        BPL .-4         ;OF THE SYSTEM OUTPUT
        MOV ADBR,#OUTPT ;AND STORE
        OUTPT=.-2
        MOV #4ØØ,ADSR   ;A/D CHANNEL #1
        TSTB ADSR       ;DETERMINE THE VALUE
        BPL .-4         ;OF THE SIGNAL AT XAD1
        MOV ADBR,#XAD1  ;AND STORE
        XAD1=.-2
        TST #OISI       ;OPTIONAL INPUT SIGNAL?
        OISI=.-2
        BEQ CLOC1       ;NO
        MOV #1ØØØ,ADSR  ;YES, A/D CHANNEL #2
        TSTB ADSR       ;DETERMINE THE VALUE
        BPL .-4         ;OF THE INPUT SOURCE
        MOV ADBR,#INPUT ;AND STORE
        INPUT=.-2
CLOC1:  MOV INPUT,CERR  ;OBTAIN THE INPUT
        SUB OUTPT,CERR  ;CALCULATE THE ERROR
        DEC #TSR2       ;APPLY ERROR THIS INTERRUPT?
        TSR2=.-2
        BNE DSRQ        ;NO
```

```
        MOV CERR,#ERROR ;YES, STORE APPLIED ERROR
        ERROR=.-2
        ASL #CERR       ;D/A CONVERSION IS 12-BIT BIPOLAR
        CERR=.-2
        MOV CERR,DABØ   ;APPLY ERROR TO SYSTEM
        MOV TSR,TSR2    ;RESET SAMPLING RATE COUNTER
DSRQ:   TST #NDP2       ;END OF DATA COLLECTION?
        NDP2=.-2
        BNE .+4
        RTI             ;YES
        DEC #DSR2       ;NO, COLLECT THIS INTERVAL?
        DSR2=.-2
        BEQ .+4
        RTI             ;NO
        MOV INPUT,(RØ)+ ;YES, STORE INPUT
        MOV ERROR,(R1)+ ;STORE ERROR
        MOV OUTPT,(R2)+ ;STORE OUTPUT
        MOV XAD1,(R3)+  ;STORE XAD1
        MOV DSR,DSR2    ;RESET STORAGE RATE COUNTER
        DEC NDP2
        RTI

;SERVICE ROUTINE:  TTY-IN, EXECUTIVE
TEXEC:  MOVB TKB,R4     ;USE R4 TO SAVE TIME AND SPACE
        BIC #1776ØØ,R4  ;GET RID OF TRASH
        TST #ECHO       ;OUTPUT IN PROGRESS?
        ECHO=.-2
        BNE .+12.       ;YES, DO NOT ECHO
        TSTB TPS        ;NO, ECHO TELETYPE
        BPL .-4
        MOVB R4,TPB
        CMPB #1Ø2,R4    ;B, NEGATE INPUT?
        BNE .+12.       ;NO
        CLR OISI        ;YES, TERMINATE OPTIONAL INPUT
        NEG INPUT       ;NEGATE PRESENT INPUT
        BB NEXT
        CMPB #1Ø3,R4    ;C, OPTIONAL INPUT REQUEST?
        BNE .+8.        ;NO
        INC OISI        ;YES, SET OISI FLAG
        BR NEXT
        CMPB #1Ø1,R4    ;A, INPUT AMPLITUDE?
        BNE .+6
        JMP AMPLC
        CMPB #122,R4    ;R, INTERRUPT RATE?
        BNE .+6
        JMP TIMEC
        CMPB #124,R4    ;T, SAMPLING RATE?
        BNE .+6
        JMP SAMPR
        CMPB #123,R4    ;S, DATA STORAGE RATE?
        BNE .+6
        JMP DSINT
```

```
        CMPB #116,R4      ;N, NUMBER OF DATA POINTS?
        BNE .+6
        JMP NUMDP
        CMPB #121,R4      ;Q, NUMBER OF P.E. ITERATIONS
        BNE .+6
        JMP PEITR

;ALL COMMANDS THAT FOLLOW MUST TERMINATE ANY
;    OUTPUT SEQUENCE BEFORE EXECUTION
        TST ECHO          ;OUTPUT IN PROGRESS?
        BEQ TEXED         ;NO, CONTINUE COMMAND SEARCH
        CLR ECHO          ;YES, TERMINATE THE OUTPUT BY
        TSTB TPS          ;FIXING UP THE STACK POINTER
        BPL .-4
        MOVB R4,TPB       ;ECHO THE TELETYPE FIRST
        CMP #PRINT,(SP)   ;WAS A SUBROUTINE INTERRUPTED?
        BGT TEXED-2       ;NO, PSEUDO RTI ONLY
        CMP #OCTS,(SP)    ;YES, DID IT OCCUR DURING OCTS?
        BGT .+4           ;NO, PSEUDO RTS PC ONLY ONCE
        TST (SP)+         ;YES, PSEUDO RTS PC TWICE
        TST (SP)+         ;PSEUDO RTS PC
        CMP (SP)+,(SP)+   ;PSEUDO RTI
TEXED:  CMPB #1Ø4,R4      ;D, DATA STORAGE REQUEST?
        BNE .+6
        JMP DSREQ
        CMPB #1Ø6,R4      ;F, OIS DATA STORAGE REQUEST?
        BNE .+1Ø.         ;NO
        INC OISI          ;YES, ASSURE OPTIONAL INPUT OPERATION
        JMP DSREQ+4       ;ENTER DSREQ AFTER FIRST INSTRUCTION
        CMPB #111,R4      ;I, INPUT PLOT?
        BNE .+1Ø.         ;NO
        MOV #TINP,R3      ;YES, SET TABLE POINTER
        JMP DOREQ
        CMPB #1Ø5,R4      ;E, ERROR PLOT?
        BNE .+1Ø.
        MOV #TERR,R3
        JMP DOREQ
        CMPB #117,R4      ;O, OUTPUT PLOT?
        BNE .+1Ø.
        MOV #TOUT,R3
        JMP DOREQ
        CMPB #13Ø,R4      ;X, XAD1 PLOT?
        BNE .+1Ø.
        MOV #TAD1,R3
        JMP DOREQ
        CMPB #12Ø,R4      ;P, PARAMETER ESTIMATION
        BNE .+6
        JMP 7ØØØ          ;ADDRESS OF P.E. PROGRAM

;ANY CHARACTER MAKING IT THIS FAR IS NOT LEGITIMATE
KILL:   MOV #TEXEC,6Ø     ;RESET TTY-IN TRAP VECTOR
        TST ECHO          ;OUTPUT IN PROGRESS?
```

```
        BEQ .+4             ;NO
        RTI                 ;YES
        TSTB TPS            ;INDICATE A TYPOGRAPHICAL ERROR
        BPL .-4
        MOVB #77,TPB        ;QUESTION MARK
NEXT:   TST ECHO            ;OUTPUT IN PROGRESS?
        BEQ .+4             ;NO
        RTI                 ;YES
        TSTB TPS
        BPL .-4
        MOVB #15,TPB        ;CARRIAGE RETURN
        TSTB TPS
        BPL .-4
        MOVB #12,TPB        ;LINE FEED
        TSTB TPS
        BPL .-4
        MOVB #52,TPB        ;ASTERISK
        RTI

;INPUT AMPLITUDE
AMPLC:  MOV #DATA,6Ø        ;POINT TTY TRAP VECTOR
        MOV #INPUT,DATL     ;DATA DESTINATION POINTER
        CLR DATS            ;TEMPORARY DATA STORAGE
        MOV #1,DATP         ;DATA SERVICE ROUTINE POINTER
        RTI

;INTERRUPT RATE
TIMEC:  MOV #DATA,6Ø        ;POINT TTY TRAP VECTOR
        MOV #BUF,DATL
        CLR DATS
        MOV #4,DATP
        RTI

;SAMPLING RATE
SAMPR:  MOV #DATA,6Ø        ;POINT TTY TRAP VECTOR
        MOV #TSR,DATL
        CLR DATS
        CLR DATP
        RTI

;DATA STORAGE RATE
DSINT:  MOV #DATA,6Ø        ;POINT TTY TRAP VECTOR
        MOV #DSR,DATL
        CLR DATS
        CLR DATP
        RTI

;NUMBER OF DATA POINTS
NUMDP:  MOV #DATA,6Ø        ;POINT TTY TRAP VECTOR
        MOV #NDP,DATL
        CLR DATS
        CLR DATP
        RTI
```

```
;NUMBER OF P.E. ITERATIONS
PEITR:  MOV #DATA,6Ø     ;POINT TTY TRAP VECTOR
        MOV #RSQ,DATL
        CLR DATS
        CLR DATP
        RTI

;DATA STORAGE REQUEST
DSREQ:  CLR OISI         ;TERMINATE OPTIONAL INPUT
        MOV #TINP,RØ     ;"INPUT" TABLE
        MOV #TERR,R1     ;"ERROR" TABLE
        MOV #TOUT,R2     ;"OUTPUT" TABLE
        MOV #TAD1,R3     ;"XAD1" TABLE
        MOV #2,DSR2      ;INITIAL DATA SAMPLING RATE
        MOV TSR,#TSR3    ;SAMPLING RATE OF DSREQ
        TSR3=.-2
        MOV DSR,#DSR3    ;DATA STORAGE RATE OF DSREQ
        DSR3=.-2
        MOV BUF2,#BUF3   ;INTERRUPT RATE OF DSREQ
        BUF3=.-2
        TST OISI         ;OPTIONAL INPUT SOURCE?
        BEQ .+1Ø.        ;NO
        MOV NDP,NDP2     ;YES, START COLLECTION OF DATA
        BR NEXT
        MOV #DATA,6Ø     ;POINT TTY TRAP VECTOR
        MOV #INPUT,DATL ;SET THE STAGE FOR A CHANGE
        CLR DATS         ;IN INPUT AMPLITUDE AND
        MOV #3,DATP      ;START OF DATA STORAGE
        RTI

;SERVICE ROUTINE:  TTY-IN, DATA CHANGER
;    DATA INPUT AND UPDATING UNDER CONDITIONS SET:
;        DATP BIT Ø SET FOR D/A CONVERSION
;        DATP BIT 1 SET FOR DATA OUTPUT REQUEST
;        DATP BIT 2 SET FOR INTERRUPT RATE
DATA:   MOVB TKB,R4      ;R4 SAVES TIME AND SPACE
        BIC #1776ØØ,R4   ;GET RID OF TRASH
        CMPB #15,R4      ;<CR>, END OF DATA?
        BEQ DAT1         ;YES
        CMPB #113,R4     ;K, ERROR IN DATA?
        BEQ DAT4         ;YES
        TST ECHO         ;OUTPUT IN PROGRESS?
        BNE .+12.        ;YES, DO NOT ECHO
        TSTB TPS         ;NO, ECHO TELETYPE
        BPL .-4
        MOVB R4,TPB
        CMPB #55,R4      ;-, MINUS SIGN?
        BNE .+8.         ;NO
        INC #SIGN        ;YES, INDICATE AS SUCH
        SIGN=.-2
        RTI
```

```
        CMPB #53,R4         ;+, PLUS SIGN?
        BNE .+4             ;NO
        RTI                 ;YES, IGNORE IT
        CMPB #60,R4         ;LOOK FOR A 0
        BGT DAT4            ;ERROR IF SMALLER THAN 0
        CMPB #71,R4         ;LOOK FOR A 9
        BLT DAT4            ;ERROR IF LARGER THAN 9
        BIC #177760,R4      ;CONVERT TO BINARY
        MOV DATS,@#MQ       ;UPDATE TEMPORARY
        MOV #10.,@#MUL      ;STORAGE LOCATION
        MOV @#MQ,DATS
        ADD R4,#DATS        ;ADD NEW DATA
        DATS=.-2
        RTI
DAT1:   TST SIGN            ;NEGATE DATA?
        BEQ .+6             ;NO
        NEG DATS            ;YES
        ROR #DATP           ;INPUT AMPLITUDE CHANGE?
        DATP=.-2
        BCC DAT2            ;NO
        MOV DATS,@#MQ       ;YES, SCALE FOR D/A CONVERSION
        MOV #128.,@#MUL
        MOV #125.,@#DIV
        MOV @#MQ,DATS
        CLR OISI            ;TERMINATE OPTIONAL INPUT
DAT2:   MOV DATS,@DATL      ;CHANGE DATA
        MOV #TEXEC,60       ;REPOINT TTY TRAP VECTOR
        CLR SIGN            ;CLEAR SIGN FOR NEXT TIME
        ROR DATP            ;DATA OUTPUT REQUEST?
        BCC DAT3            ;NO
        MOV NDP,NDP2        ;YES, START COLLECTION OF DATA
        JMP NEXT
DAT3:   ROR DATP            ;INTERRUPT RATE CHANGE?
        BCC .+8.            ;NO
        MOV DATS,BUF2       ;YES, STORE INTERRUPT RATE
        JMP NEXT
DAT4:   JMP KILL            ;INPUT DATA ERROR
DATL:   0                   ;ADDRESS POINTER

;DATA OUTPUT
;    R3 CONTAINS DATA ADDRESS
DOREQ:  INC ECHO            ;INDICATE OUTPUT IN PROGRESS
        CLR NDP2            ;TERMINATE DATA STORAGE
        MOV #140,PS         ;PRIORITY 3
        MOV R3,R1           ;TABLE POINTER
        MOV NDP,R2          ;NUMBER OF OUTPUT POINTS
        MOV (R1),MINN       ;INITIALIZATIONS
        MOV (R1)+,MAXN

;DETERMINE MAXIMUM AND MINIMUM VALUES
MAG1:   DEC R2              ;END OF DATA?
        BEQ MAG3            ;YES
        MOV (R1)+,R5        ;NO, OBTAIN NEXT POINT
```

```
        CMP R5,MAXN       ;NEW MAXIMUM?
        BMI MAG2          ;NO
        MOV R5,#MAXN      ;YES, STORE
        MAXN=.-2
        BR MAG1
MAG2:   CMP R5,MINN       ;NEW MINIMUM?
        BPL MAG1          ;NO
        MOV R5,#MINN      ;YES, STORE
        MINN=.-2
        BR MAG1
MAG3:   MOV MAXN,MAGN
        SUB MINN,#MAGN    ;ABSOLUTE DIFFERENCE
        MAGN=.-2

;DETERMINE QUANTIZATION STEP
        MOV MAGN,@#MQ     ;MAXIMUM PLOT MAGNITUDE
        MOV #5Ø.,@#DIV    ;DIVIDE BY PLOT WIDTH
        MOV @#MQ,#STEP    ;PLOT QUANTIZATION STEP
        STEP=.-2
        BNE .+6           ;CHECK FOR ZERO STEP
        INC STEP          ;IF SO, DO SOMETHING ABOUT IT

;DETERMINE DATA STORAGE RATE IN MILLI-SEC
        MOV DSR3,@#MQ
        MOV BUF3,@#MUL
        MOV #1ØØ.,@#DIV
        MOV @#MQ,#MSEC
        MSEC=.-2

;OUTPUT HEADER
        MOV #MS2,R1
        MOV #MS3,R2
        JSR PC,PRINT      ;UNIVERSAL PORTION OF HEADER
        CMP #TINP,R3      ;"INPUT" PLOT?
        BNE EPLT          ;NO, CHECK NEXT POSSIBILITY
        MOV #MS3,R1       ;YES, SET UP MESSAGE POINTERS
        MOV #MS4,R2
        BR AXIS
EPLT:   CMP #TERR,R3      ;"ERROR" PLOT?
        BNE OPLT          ;NO
        MOV #MS4,R1       ;YES
        MOV #MS5,R2
        BR AXIS
OPLT:   CMP #TOUT,R3      ;"OUTPUT" PLOT?
        BNE XPLT          ;NO, IT MUST BE XAD1 THEN
        MOV #MS5,R1       ;YES
        MOV #MS6,R2
        BR AXIS
XPLT:   MOV #MS6,R1       ;"XAD1" PLOT
        MOV #MS7,R2
AXIS:   JSR PC,PRINT      ;PRINT TYPE OF PLOT
```

```
;PLOT AN ORDINATE AXIS
        CLR R5          ;PLOT MAGNITUDE REGISTER
        CMP STEP,MINN   ;POSITIVE MINIMUM > STEP?
        BMI AXI2        ;YES, PLOT +'S ONLY
        TST MAXN        ;NO, PLOT A ZERO?
        BPL .+6         ;YES
        ADD MAXN,R5     ;NO, PLOT -'S ONLY
        JSR PC,PLOT
        ADD STEP,MAXN   ;MAXIMUM > -STEP?
        BMI AXI3        ;NO, ALL DONE WITH AXIS
        SUB STEP,MAXN   ;YES, CORRECT MAXN
        TSTB TPS
        BPL .-4
        MOVB #6Ø,TPB    ;PRINT A Ø
        SUB STEP,R5     ;GET READY FOR +'S
        ADD MAXN,R5
AXI1:   SUB STEP,R5     ;PLOT +'S
        BLT AXI3        ;END OF AXIS PLOT.
        TSTB TPS
        BPL .-4
        MOVB #53,TPB    ;PLUS SIGN
        BR AXI1
AXI2:   MOV MAGN,R5     ;GET READY FOR A +'S ONLY PLOT
        BR AXI1
AXI3:   MOV #MS7,R1
        MOV #MS8,R2
        JSR PC,PRINT    ;CR AND LF

;OUTPUT DATA AND TIME VALUES
        CLR #TIME       ;INITIALIZATIONS
        TIME=.-2
        MOV NDP,#NDP3
        NDP3=.-2
        CLR #COND
        COND=.-2
PRPLT:  INC COND        ;SET CONDITION CODE FOR TIME
        MOV TIME,R5
        JSR PC,OCDIG    ;PRINT TIME VALUE
        ADD MSEC,TIME   ;UPDATE TIME VALUE FOR NEXT TIME
        MOV #MS8,R1
        MOV #MS9,R2
        JSR PC,PRINT    ;THREE SPACES
        CLR COND        ;SET CONDITION CODE FOR DATA
        MOV (R3),R5     ;FETCH DATA VALUE
        JSR PC,OCDIG    ;PRINT DATA VALUE
        MOV #MS8,R1
        MOV #MS9,R2
        JSR PC,PRINT    ;THREE MORE SPACES
        MOV (R3)+,R5
        JSR PC,PLOT     ;PLOT DATA VALUE
        MOV #MS9,R1
        MOV #MSA,R2
```

```
          JSR PC,PRINT      ;ASTERISK, CR, AND LF
          DEC NDP3          ;END OF PLOT?
          BNE PRPLT         ;NO

;TERMINATE PLOT AND OUTPUT SAMPLE RATE
PPEND:    MOV #MSA,R1
          MOV #MSB,R2
          JSR PC,PRINT      ;"SAMPLING INTERVAL-"
          MOV TSR3,@#MQ     ;DETERMINE SAMPLE RATE
          MOV BUF3,@#MUL
          MOV #1ØØ.,@#DIV
          MOV @#MQ,R5
          INC COND          ;SET CONDITION CODE FOR TIME
          JSR PC,OCDIG      ;OUTPUT VALUE
          MOV #MSB,R1
          MOV #MEB,R2
          JSR PC,PRINT      ;TERMINATION CHARACTERS
          CLR ECHO          ;END OF THIS OUTPUT SEQUENCE
          JMP NEXT

;SUBROUTINE:  PRINT
;    PRINTS A STRING OF ASCII CHARACTERS
;    STARTING AT R1 AND TERMINATING WITH R2
PRINT:    TSTB TPS
          BPL .-4
          MOVB (R1)+,TPB
          CMP R1,R2
          BNE PRINT
          RTS PC

;SUBROUTINE:  OCDIG
;    OUTPUTS SIGN AND MAGNITUDE OF OCTAL NUMBER
;    CONTAINED IN R5 UNDER CONDITIONS SET:
;         COND=Ø FOR DATA
;         COND>Ø FOR TIME
OCDIG:    CLR SUPZ          ;SUPPRESS LEADING ZEROES
          TST COND
          BNE TNTH          ;TIME
          MOV R5,@#MQ       ;DATA, SCALE A/D NUMBER
          MOV #125.,@#MUL
          MOV #128.,@#DIV
          MOV @#MQ,R5
          BPL PLUS          ;POSITIVE NUMBER
          TSTB TPS          ;NEGATIVE NUMBER
          BPL .-4
          MOVB #55,TPB      ;OUTPUT MINUS SIGN
          NEG R5            ;OBTAIN MAGNITUDE
          BR THOU
PLUS:     TSTB TPS
          BPL .-4
          MOVB #4Ø,TPB      ;SPACE
          BR THOU
```

```
TNTH:   MOV #1ØØØØ.,R1
        JSR PC,OCTS
        INC SUPZ        ;CEASE ZERO SUPPRESSION FOR TIME
THOU:   MOV #1ØØØ.,R1
        JSR PC,OCTS
        INC SUPZ        ;CEASE FOR DATA HERE
        TST COND
        BEQ HUND        ;DATA
        TSTB TPS        ;TIME
        BPL .-4
        MOVB #56,TPB    ;DECIMAL POINT FOR TIME
HUND:   MOV #1ØØ.,R1
        JSR PC,OCTS
        TST COND
        BNE TENS        ;TIME
        TSTB TPS        ;DATA
        BPL .-4
        MOVB #56,TPB    ;DECIMAL POINT FOR DATA
TENS:   MOV #1Ø.,R1
        JSR PC,OCTS
        MOV #1,R1
        JSR PC,OCTS
        RTS PC

;SUBROUTINE:  PLOT OUTPUT
;   R5 CONTAINS NUMBER TO BE PLOTTED
PLOT:   SUB MINN,R5     ;OBTAIN PLOT MAGNITUDE
        SUB STEP,R5
        BGT .+4
        RTS PC
        TSTB TPS
        BPL .-4
        MOVB #55,TPB    ;DASH
        BR PLOT+4

;SUBROUTINE:  OCTS
;   OCTDIG SERVICE ROUTINE
OCTS:   CLR R2          ;COUNTER
        SUB R1,R5
        BLO .+6
        INC R2
        BR OCTS+2
        ADD R1,R5
        TST R2          ;R2=Ø.Ø?
        BNE OCT1        ;NO
        TST #SUPZ       ;YES, SUPPRESS ZERO?
        SUPZ=.-2
        BNE OCT1        ;NO
        SUB #2Ø,R2      ;YES, FORM ASCII SPACE
        BR .+6          ;KEEP SUPPRESSING ZEROES
```

```
OCT1:   INC SUPZ        ;CEASE ZERO SUPPRESSION
        ADD #6Ø,R2
        TSTB TPS
        BPL .-4
        MOV R2,TPB      ;OUTPUT CHARACTER
        RTS PC

;MESSAGES
MS1=.
.BYTE 15,12,12,12
.ASCII /A-INPUT AMPLITUDE, TERM OIS (A5ØØ=5.ØØ VOLTS)/
.BYTE 15,12
.ASCII /B-NEGATE INPUT, TERMINATE OIS/
.BYTE 15,12
.ASCII /C-OPTIONAL INPUT SOURCE (OIS)/
.BYTE 15,12
.ASCII /D-DATA STORAGE REQUEST, TERM OIS (STEP INPUT)/
.BYTE 15,12
.ASCII /F-OPTIONAL INPUT DATA STORAGE REQUEST/
.BYTE 15,12
.ASCII /N-NUMBER OF DATA POINTS/
.BYTE 15,12
.ASCII /P-PARAMETER ESTIMATION/
.BYTE 15,12
.ASCII /Q-NUMBER OF P.E. ITERATIONS/
.BYTE 15,12
.ASCII $R-INTERRUPT RATE (R/1ØØ MSEC)$
.BYTE 15,12
.ASCII $T-SAMPLING RATE (T*R/1ØØ MSEC)$
.BYTE 15,12
.ASCII $S-DATA STORAGE RATE (S*R/1ØØ MSEC)$
.BYTE 15,12
.ASCII /I-PLOT "INPUT"/
.BYTE 15,12
.ASCII /E-PLOT "ERROR"/
.BYTE 15,12
.ASCII /O-PLOT "OUTPUT"/
.BYTE 15,12
.ASCII /X-PLOT "XAD1"/
.BYTE 15,12
.ASCII /K-KILL INPUT OR OUTPUT SEQUENCE/
.BYTE 15,12,12,12
MS2=.
.BYTE 15,12,12,12,12,12
.ASCII /                        RESPONSE VERSUS TIME/
.BYTE 15,12,12
.ASCII / TIME    /
MS3=.
.ASCII / INPUT   /
MS4=.
.ASCII / ERROR   /
```

```
MS5=.
.ASCII /OUTPUT    /
MS6=.
.ASCII / XAD1     /
MS7=.
.BYTE 15,12
MS8=.
.BYTE 4Ø,4Ø,4Ø
MS9=.
.BYTE 52,15,12
MSA=.
.BYTE 12
.ASCII /SAMPLING INTERVAL-/
MSB=.
.ASCII / SECONDS/
.BYTE 15,12,12,12,12
MEB=.
.EVEN

;STORAGE LOCATIONS
TINP=.              ;INPUT TABLE
.=.+2Ø2.
TERR=.              ;ERROR TABLE
.=.+2Ø2.
TOUT=.              ;OUTPUT TABLE
.=.+2Ø2.
TAD1=.              ;XAD1 TABLE
.=.+2Ø2.
.END START
```

APPENDIX D

ASSEMBLY LANGUAGE LISTING FOR SEQUENTIAL LEAST SQUARES PARAMETER ESTIMATION

The assembly language program used for parameter estimation is given in the following pages. Included also are the special floating point subroutines developed for use by this program. The program and the floating point subroutines are explained in Chapter III. Complementary flowcharts accompany the explanation.

```
;SEQUENTIAL LEAST SQUARES PARAMETER ESTIMATION
;     USING A PDP-11/2Ø MINICOMPUTER
;
;ELECTRICAL ENGINEERING DEPARTMENT
;     UNIVERSITY OF TENNESSEE
;
;PROGRAMMER:  CARL A. NEHMER
;DATE:  JULY 1973
;
;TO BE USED WITH SAMPLED-DATA-SYSTEMS
;     MONITORING PROGRAM
;
;ASSIGN NAMES TO REGISTERS AND BUFFERS
        RØ=%Ø
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        SP=%6
        PC=%7
        PS=177776
        DIV=1773ØØ
        AC=DIV+2
        MQ=DIV+4
        MUL=DIV+6
        SC=DIV+1Ø
        SR=DIV+11
        NOR=DIV+12
        LSH=DIV+14
        ASH=DIV+16
        TPS=177564
        TPB=TPS+2

;INITIALIZATIONS
.=7ØØØ                          ;STARTING ADDRESS
SLSPE:  MOV #24Ø,PS             ;SET PRIORITY AT 5
        CLR NDP2                ;TERMINATE DATA STORAGE
        MOV #3,#Z               ;SECOND-ORDER, SINGLE INPUT
        Z=.-2
        MOV Z,@#MQ
        MOV #6,@#MUL
        MOV @#MQ,RØ             ;STORAGE SPACE FOR CONSTANTS
        MOV RØ,ZZ
        ASR #ZZ                 ;NEEDED LATER
        ZZ=.-2
        MOV #GR1,R1             ;ADDRESS OF CONSTANT MATRICES
        CLR (R1)+
        DEC RØ                  ;GR1=G1(Ø)=[Ø Ø Ø]'
        BNE .-4                 ;GR2=G2(Ø)=[Ø Ø Ø]'
        MOV Z,RØ                ;GENERATE INITIAL P MATRIX
        MOV RØ,R1               ;DIMENSION
        MOV #PRØ,R2             ;MATRIX ADDRESS
```

```
        MOV #EPI,R3       ;FPN(I/E), DIAGONAL MULTIPLIER
SEQ1:   CMP RØ,R1         ;DIAGONAL ELEMENT?
        BEQ SEQ2          ;YES
        CLR (R2)+         ;NO, FLOATING POINT ZERO
        CLR (R2)+
        CLR (R2)+
        BR SEQ3
SEQ2:   MOV (R3),(R2)+    ;ENTER DIAGONAL ELEMENT
        MOV 2(R3),(R2)+
        MOV 4(R3),(R2)+
SEQ3:   DEC RØ            ;END OF ROW?
        BNE SEQ1          ;NO
        MOV Z,RØ          ;YES, RESET ROW COUNTER
        DEC R1            ;END OF COLUMNS AND MATRIX?
        BNE SEQ1          ;NO, PRØ=P(Ø)=1/E[1]
        MOV #TOUT,TX1     ;SET UP TABLE POINTERS:  OUTPUT
        MOV #TAD1,TX2     ;XAD1
        MOV #TINP,TIN     ;INPUT
        CLR R             ;ITERATIONS COUNTER

;BEGIN SEQUENTIAL REGRESSION ANALYSIS
;    OBTAIN AND CONVERT DATA FOR THIS ITERATION
SEQ4:   INC R             ;R'TH ITERATION
        MOV #HRØ,R5       ;STATE AND INPUT VECTOR
        MOV TX1,R4        ;A/D OUTPUT VALUE
        JSR PC,NRAD       ;CONVERT TO FPN
        MOV R4,#TX1       ;UPDATE POINTER
        TX1=.-2           ;X1 POINTER
        MOV TX2,R4        ;R5 SET FROM ABOVE, A/D XAD1 VALUE
        JSR PC,NRAD       ;CONVERT TO FPN
        MOV R4,#TX2       ;UPDATE POINTER
        TX2=.-2           ;X2 POINTER
        MOV TIN,R4        ;R5 SET FROM ABOVE, A/D INPUT VALUE
        JSR PC,NRAD       ;HRØ=H(R)=W(R)=[X1(R) X2(R) U(R)]'
        MOV R4,#TIN       ;UPDATE POINTER
        TIN=.-2           ;INPUT POINTER
        MOV #XR1,R5       ;ADDRESS OF OUTPUT MATRIX
        MOV TX1,R4        ;(R+1)'TH OUTPUT
        JSR PC,NRAD       ;XR1=[X1(R+1)]
        MOV TX2,R4        ;R5 SET FROM ABOVE, (r+1)'TH XAD1
        JSR PC,NRAD       ;XR2=[X2(R+1)]

;BEGIN MATRIX MANIPULATIONS
        MOV #1,#M         ;MATRIX DIMENSIONS
        M=.-2
        MOV Z,#N
        N=.-2
        MOV Z,#P
        P=.-2
        MOV #HRØ,R1       ;MATRIX ADDRESSES
        MOV #PRØ,R2
        MOV #HPØ,R5
        JSR PC,MATX       ;HPØ=[H(R)'*P(R-1)]
```

```
        MOV #1,P          ;M&N SET FROM ABOVE
        MOV #HPØ,R1
        MOV #HRØ,R2
        MOV #HPH,R5
        JSR PC,MATX       ;HPH=[H(R)'*P(R-1)*H(R)]
        MOV #ONE,R3       ;FPN(+1)
        MOV #MATS,R4      ;TEMPORARY STORAGE
        MOV (R3)+,(R4)    ;ADDA OCCASIONALLY DESTROYS (R4)
        MOV (R3)+,2(R4)
        MOV (R3)+,4(R4)
        JSR PC,ADDA       ;HPH=[1+HPH]
        JSR PC,NEGT       ;HPH=-[1+H(R)'*P(R-1)*H(R)]
        MOV Z,M
        MOV #1,N
        MOV Z,P
        MOV #HPØ,R1       ;MATRIX ADDRESSES
        MOV R1,R2         ;BOTH THE SAME
        MOV #PR1,R5
        JSR PC,MATX       ;PR1=[P(R-1)*H(R)*H(R)'*P(R-1)]
        MOV ZZ,R1
        MOV #HPH,R4       ;R5 SET FROM ABOVE
        JSR PC,DIVD
        BVC .+4           ;OVERFLOW?
        RTS PC            ;YES
        DEC R1            ;NO
        BNE .-1Ø.         ;PR1=[PR1]/HPH
        MOV ZZ,R1         ;ADD TWO Z BY Z MATRICES
        MOV #PR1,R4
        MOV #PRØ,R5
SEQ6:   JSR PC,ADDA
        ADD #6,R4
        ADD #6,R5
        DEC R1
        BNE SEQ6          ;PRØ=P(R)=[PRØ+PR1]

;END OF SEQUENTIAL DETERMINATION OF P(R)
;BEGIN UPDATE OF PARAMETER MATRIX
        MOV Z,N           ;M SET FROM ABOVE
        MOV #1,P
        MOV #PRØ,R1
        MOV #HRØ,R2
        MOV #HPØ,R5
        JSR PC,MATX       ;HPØ=[P(R)*W(R)]
        MOV #1,M          ;N&P SET FROM ABOVE
        MOV #HRØ,R1       ;W(R)=H(R) FOR Q(R)=1.Ø
        MOV #GR1,R2
        MOV #WG1,R5
        JSR PC,MATX       ;WG1=[W(R)'*G1(R-1)]
        JSR PC,NEGT
        MOV #XR1,R4
        JSR PC,ADDA       ;WG1=[X1(R+1)-W(R)'*G1(R-1)]
        MOV #HRØ,R1
```

```
        MOV #GR2,R2
        MOV #WG2,R5
        JSR PC,MATX       ;WG2=[W(R)'*G2(R-1)]
        JSR PC,NEGT
        MOV #XR2,R4
        JSR PC,ADDA       ;WG2=[X2(R+1)-W(R)'*G2(R-1)]
        MOV Z,CT
        MOV #WG1,R1
        MOV #HPØ,R2
        MOV #HPH,R4       ;TEMPORARY STORAGE
        MOV #GR1,R5
SEQ7:   JSR PC,MULT
        JSR PC,ADDA
        SUB #6,R1
        ADD #6,R5
        DEC #CT
        CT=.-2
        BNE SEQ7          ;GR1=G1(R)=[G1(R-1)+P(R)*W(R)*WG1]
        MOV Z,CT
        MOV #WG2,R1
        MOV #HPØ,R2
        MOV #GR2,R5       ;R4 SET FROM ABOVE
SEQ8:   JSR PC,MULT
        JSR PC,ADDA
        SUB #6,R1
        ADD #6,R5
        DEC CT
        BNE SEQ8          ;GR2=G2(R)=[G2(R-1)+P(R)*W(R)*WG2]
        CMP RSQ,#R        ;END OF ITERATIONS?
        R=.-2
        BEQ .+6           ;YES
        JMP SEQ4          ;NO

;OUTPUT ESTIMATED VALUES AND NUMBER OF ITERATIONS
        MOV #MSC,R1
        MOV #MSD,R2
        JSR PC,PRINT      ;HEADER
        CLR R1            ;SUPPRESS ZEROES
        MOV #DBWS,R5      ;TEMPORARY STORAGE ADDRESS
        MOV RSQ,(R5)      ;OUTPUT NUMBER OF ITERATIONS
        MOV #1ØØ.,R3      ;RSQ<1ØØØ.
        JSR PC,DECA       ;USE DTOA SERVICE ROUTINE
        MOV #1Ø.,R3
        JSR PC,DECA
        INC R1            ;TERMINATE ZERO SUPPRESS
        MOV #1,R3
        JSR PC,DECA
        MOV #MSD,R1
        MOV #MSE,R2
        JSR PC,PRINT      ;INDENT
        MOV Z,CT          ;COUNTER
        MOV #GR1,R4       ;DATA ADDRESS
```

```
SEQ9:    JSR PC,DBWD       ;OUTPUT VALUES
         DEC CT            ;END OF THIS SET?
         BEQ SEQA          ;YES
         MOV #MSE,R1
         MOV #MSF,R2
         JSR PC,PRINT      ;NO, OUTPUT SPACES
         BR SEQ9
SEQA:    MOV #MSF,R1
         MOV #MSG,R2
         JSR PC,PRINT      ;INDENT SECOND SET OF VALUES
         MOV Z,CT          ;COUNTER
         MOV #GR2,R4       ;DATA ADDRESS
SEQB:    JSR PC,DBWD       ;OUTPUT VALUES
         DEC CT            ;END OF OUTPUT?
         BEQ SEQC          ;YES
         MOV #MSE,R1
         MOV #MSF,R2
         JSR PC,PRINT      ;NO, MORE SPACES
         BR SEQB
SEQC:    MOV #MSG,R1
         MOV #MEG,R2       ;TERMINATION CHARACTERS
         JMP NEXT          ;ALL DONE

;SUBROUTINE:  PRINT
;    SAME AS FOR SDSMP
PRINT:   TSTB TPS
         BPL .-4
         MOVB (R1)+,TPB
         CMP R1,R2
         BNE PRINT
         RTS PC

;FLOATING POINT SUBROUTINES USING
;    THE EXTENDED ARITHMETIC UNIT
;FORMAT: B-LOWER ORDER WORD
;        A-HIGHER ORDER WORD
;        E-EXPONENT, EXCESS 2**15.  FORM

;SUBROUTINE:  NRAD
;    (R5)+=FPN(R4)+
;    FOR USE ONLY ON A/D VALUES
NRAD:    MOV #1ØØØ26,R3    ;SET UP EXPONENT
         MOV (R4)+,@#MQ    ;OBTAIN A/D VALUE
         BEQ NRAZ          ;(R4)=Ø.Ø
         MOV #5,@#MUL      ;ALGORITHM ADJUSTMENT
         INC @#NOR         ;NORMALIZE
         SUB @#NOR,R3      ;ADJUST EXPONENT
         CLR (R5)+         ;CLEAR B
         MOV @#AC,(R5)+    ;AC TO A
```

```
        MOV R3,(R5)+        ;EXP TO E
        RTS PC
NRAZ:   CLR (R5)+           ;A/D VALUE WAS ZERO
        CLR (R5)+
        CLR (R5)+
        RTS PC

;SUBROUTINE:  MULT
;    (R4) = (R1)+ * (R2)+
;    (R1) & (R2) MUST BE FPN
MULT:   MOV R4,-(SP)        ;SAVE DESTINATION ADDRESS
        MOV #MQ,RØ          ;USING RØ AS A POINTER SAVES TIME
        CMP (R1)+,(R2)+     ;INCREMENT R1 & R2
        MOV (R1)+,(RØ)+     ;A1 TO MQ
        MOV (R2)+,(RØ)      ;A2 TO MUL, FORM PRIMARY PRODUCT
        MOV (R1)+,R3        ;E1 TO EXP
        BEQ MULZ            ;E1=Ø.Ø
        TST (R2)
        BEQ MULZ            ;E2=Ø.Ø
        ADD (R2)+,R3        ;EXP=E1+E2
        ADD #1ØØØØ1,R3      ;ALGORITHM ADJUSTMENT
        MOV R3,4(R4)        ;STORE EXP
        MOV -(RØ),R3        ;MQ TO B
        MOV -2(RØ),R4       ;AC TO A, STORE PRODUCT
        MOV -6(R1),(RØ)     ;B1 TO MQ
        BEQ MUL1            ;B1=Ø.Ø
        CLR -(RØ)           ;CLEAR AC
        DEC @#LSH           ;LOGICAL SHIFT RIGHT
        MOV -4(R2),4(RØ)    ;A2 TO MUL, 1ST SECONDARY PRODUCT
        INC @#LSH           ;LOGICAL SHIFT LEFT
        ADD (RØ)+,R3        ;AC + B
        ADC R4              ;CARRY + A, UPDATE PRODUCT
MUL1:   MOV -6(R2),(RØ)     ;B2 TO MQ
        BEQ MUL2            ;B2=Ø.Ø
        CLR -(RØ)           ;CLEAR AC
        DEC @#LSH           ;LOGICAL SHIFT RIGHT
        MOV -4(R1),4(RØ)    ;A1 TO MUL, 2ND SECONDARY PRODUCT
        INC @#LSH           ;LOGICAL SHIFT LEFT
        ADD (RØ)+,R3        ;AC + B
        ADC R4              ;CARRY + A, UPDATE PRODUCT
MUL2:   MOV R3,(RØ)         ;B TO MQ
        MOV R4,-(RØ)        ;A TO AC
        MOV (SP)+,R4        ;RESTORE DESTINATION ADDRESS
        INC @#NOR           ;NORMALIZE PRODUCT
        SUB @#NOR,4(R4)     ;ADJUST EXP
        MOV (RØ)+,2(R4)     ;AC TO A
        MOV (RØ)+,(R4)      ;MQ TO B
        RTS PC
MULZ:   CMP (R2)+,(SP)+     ;INCREMENT R2 AND SP
        CLR (R4)
        CLR 2(R4)
        CLR 4(R4)
        RTS PC
```

```
;SUBROUTINE:   DIVD
;     (R5)+ =  (R5)/(R4)
;     (R4) & (R5) MUST BE FPN
DIVD:   MOV #MQ,RØ       ;USING RØ AS A POINTER SAVES TIME
        MOV (R5)+,(RØ)   ;B5 TO MQ
        MOV (R5)+,-(RØ)  ;A5 TO AC
        MOV #-2,@#ASH    ;ALGORITHM ADJUSTMENT
        MOV (R5),R3      ;E5 TO EXP
        BEQ DIVZ         ;(R5)=Ø.Ø
        MOV 2(R4),-(RØ)  ;A4 TO DIV
        BEQ DIVV         ;DIVISION BY ZERO IS A NO, NO
        SUB 4(R4),R3     ;EXP=E5-E4
        ADD #1ØØØØ1,R3   ;ALGORITHM ADJUSTMENT
        MOV R3,(R5)      ;EXP TO E
        ADD #4,RØ        ;MQ
        MOV (RØ),R3      ;SAVE MQ (A)
        MOV #-16.,@#ASH  ;GET RID OF OLD QUOTIENT
        BEQ DIV2         ;REMAINDER=Ø.Ø
        MOV #15.,@#ASH   ;REMAINDER TO AC (ALMOST)
        MOV 2(R4),-4(RØ);A4 TO DIV
        INC @#ASH        ;ALGORITHM ADJUSTMENT
        MOV R3,-(RØ)     ;A TO AC
        BPL DIV1
        DEC (RØ)         ;ADJUST A IF NEGATIVE
DIV1:   INC @#NOR        ;NORMALIZE
        SUB @#NOR,(R5)+  ;ADJUST EXP
        MOV (RØ)+,-4(R5);AC TO A
        MOV (RØ)+,-6(R5);MQ TO B
        RTS PC
DIV2:   CLR (RØ)         ;CLEAR AC & MQ
        MOV R3,-(RØ)     ;A TO AC
        BR DIV1
DIVZ:   TST (R5)+        ;INCREMENT R5
        RTS PC           ;(R5) IS ZERO
DIVV:   TSTB TPS
        BPL .-4
        MOV #166,TPB     ;V, INDICATE OVERFLOW
        SEV              ;SET OVERFLOW BIT
        RTS PC

;SUBROUTINE:   ADDA
;     (R5) = (R4) + (R5)
;     (R4) & (R5) MUST BE FPN
ADDA:   ADD #4,R4        ;GET TO EXPONENTS
        ADD #4,R5
        TST (R4)
        BEQ ADD7         ;(R4)=Ø.Ø
        TST (R5)
        BEQ ADD6         ;(R5)=Ø.Ø
        MOV #MQ,RØ       ;USING RØ SAVES TIME & SPACE
;MAKE THE TWO NUMBERS COMPATIBLE FOR ADDITION
        MOV (R4),R3      ;E4
        SUB (R5),R3      ;R3=E4-E5
```

```
        BPL ADD1          ;E4>E5
        BEQ ADD2          ;E4=E5
        MOV -4(R4),(RØ)   ;E4<E5,  B4 TO MQ
        MOV -(R4),-(RØ)   ;A4 TO AC
        MOV R3,@#ASH      ;ASR E4-E5
        MOV (RØ)+,(R4)    ;AC TO A4
        MOV (RØ),-(R4)    ;MQ TO B4
        BR ADD2+4
ADD1:   NEG R3
        MOV -4(R5),(RØ)   ;B5 TO MQ
        MOV -(R5),-(RØ)   ;A5 TO AC
        MOV R3,@#ASH      ;ASR E5-E4
        MOV (RØ)+,(R5)    ;AC TO A5
        MOV (RØ),-(R5)    ;MQ TO B5
        MOV (R4),R3       ;EXPONENT
        SUB #4,R4
        BR .+12.
ADD2:   SUB #4,R4
        MOV (R5),R3       ;EXPONENT
        SUB #4,R5
        ADD (R4)+,(R5)+   ;B5=B4+B5
        ADC (R5)
        BVS ADD5          ;JUMP IF V BIT SET
        ADD (R4),(R5)     ;A5=A4+A4
        BVC ADD4          ;IF AN OVERFLOW OCCURS
ADD3:   ROR (R5)          ;THEN SHIFT THE CARRY BIT
        ROR -2(R5)        ;BACK INTO THE NUMBER
        INC R3            ;AND INCREMENT THE EXP
ADD4:   TST (R5)          ;CHECK FOR ZERO SUM
        BNE .+8.
        TST -2(R5)        ;CHECK LOWER ORDER WORD
        BEQ ADD8
        MOV -2(R5),(RØ)   ;NORMALIZE:  B5 TO MQ
        MOV (R5),-(RØ)    ;A5 TO AC
        INC @#NOR
        SUB @#NOR,R3      ;ADJUST EXPONENT
        MOV R3,2(R5)      ;EXP
        MOV (RØ)+,(R5)    ;AC TO A5
        MOV (RØ),-(R5)    ;MQ TO B5
        TST -(R4)         ;DECREMENT R4
        RTS PC
ADD5:   ADD (R4),(R5)     ;ADD HIGH ORDER FRACTIONS
        BCS ADD4          ;IF CARRY NO REAL OVERFLOW
        BR ADD3           ;ELSE SHIFT RIGHT
ADD6:   MOV (R4),(R5)     ;(R5)=Ø.Ø
        MOV -(R4),-(R5)
        MOV -(R4),-(R5)
        RTS PC
ADD7:   SUB #4,R4         ;(R4)=Ø.Ø
        SUB #4,R5
        RTS PC
ADD8:   CLR 2(R5)         ;SUM=Ø.Ø
        CMP -(R4),-(R5)   ;RESET POINTERS
        RTS PC
```

```
;SUBROUTINE:  MATX
;     [(R5)]=[(R1)][(R2)]
;     [M*P]=[M*N][N*P]
;     [(R1)]-ARRANGED SERIALLY BY ROWS
;     [(R2)]-ARRANGED SERIALLY BY COLUMNS
;     [(R5)]-IS ARRANGED SERIALLY BY ROWS
;     ALL NUMBERS MUST BE FPN
;     EXIT WITH R5 AT THE START OF [(R5)]
MATX:    MOV M,M1          ;SAVE MATRIX DIMENSIONS
         MOV N,N1
         MOV P,P1
         MOV #6,@#MQ
         MOV N1,@#MUL
         MOV @#MQ,#N2
         N2=.-2            ;N OFFSET (ONE ROW)
         MOV R2,#P2
         P2=.-2            ;START OF [(R2)]
         MOV R5,#M2
         M2=.-2            ;START OF [(R5)]
         MOV #MATS,R4      ;MATRIX TEMPORARY STORAGE
         ADD #4,R5
MAT1:    CLR (R5)          ;READY MEMORY LOCATION
         CLR -(R5)
         CLR -(R5)
MAT2:    JSR PC,MULT       ;MULTIPLY TWO VALUES
         JSR PC,ADDA       ;AND SUM WITH PRESENT TERM
         DEC #N1           ;END OF TERM?
         N1=.-2
         BNE MAT2          ;NO
         ADD #1Ø.,R5       ;YES, SAVE TERM
         MOV N,N1          ;RESET N COUNTER
         DEC #P1           ;END OF ROW?
         P1=.-2
         BEQ .+8.          ;YES
         SUB N2,R1         ;NO, RESET R1 POINTER
         BR MAT1           :AND RETURN
         DEC #M1           ;END OF MATRIX MULTIPLY?
         M1=.-2
         BNE .+8.
         MOV M2,R5         ;YES,RESET R5 POINTER
         RTS PC
         MOV P2,R2         ;NO, RESET R2 POINTER
         MOV P,P1          ;RESET P1 COUNTER
         BR MAT1           ;FORM NEXT TERM
         MATS=.            ;TEMPORARY STORAGE LOCATION
         .=.+6

;SUBROUTINE:  NEGT
;     (R5) = NEG(R5)
;     EXITS WITH R5 AT START OF (R5)
NEGT:    NEG 2(R5)         ;NEGATE HIGH ORDER WORD
         NEG (R5)          ;NEGATE LOW ORDER WORD
         SBC 2(R5)         ;CORRECT FOR CARRY
         RTS PC
```

```
;FPN TO ASCII CONVERSION ROUTINES
;SUBROUTINE:  DBWD
;     CONVERTS FPN TO DOUBLE WORD INTEGER
;     WITH DECIMAL ASSUMED BETWEEN WORDS
DBWD:    MOV #DBWS,R5     ;TEMPORARY STORAGE
         MOV (R4)+,@#MQ   ;B TO MQ
         MOV (R4)+,@#AC   ;A TO AC
         MOV (R4)+,R3     ;EXPONENT
         BEQ DBW2         ;FPN(R4)=Ø.Ø
         BPL DBW3         ;NEGATIVE EXPONENT
         SUB #1ØØØ17,R3
         BGT DBW4         ;EXPONENT>15., OVERFLOW
DBW1:    MOV R3,@#ASH     ;ASR,ADJUST TO DOUBLE WORD
         MOV @#AC,(R5)    ;STORE IN TEMPORARY LOCATION
         MOV @#MQ,2(R5)
         BR DTOA
DBW2:    CLR (R5)         ;FPN(R4)=Ø.Ø
         CLR 2(R5)
         BR DTOA
DBW3:    ADD #1ØØØØ1.R3   ;NEGATIVE EXPONENT ADJUSTMENT
         MOV @#AC,@#MQ
         BR DBW1
DBW4:    MOV #7,R1        ;OVERFLOW INDICATOR ROUTINE
         TSTB TPS
         BPL .-4
         MOVB #52,TPB     ;ASTERISK
         DEC R1
         BNE DBW4+4
         RTS PC

;DOUBLE WORD TO ASCII CONVERSION
;    DOUBLE WORD < 1ØØ.Ø FOR PRINTOUT
DTOA:    CLR R1           ;ZERO SUPPRESS INDICATOR
         TST (R5)
         BPL DTO1         ;POSITIVE NUMBER
         NEG (R5)         ;NEGATIVE NUMBER
         NEG 2(R5)        ;NEGATE
         SBC (R5)
         TSTB TPS
         BPL .-4
         MOVB #55,TPB     ;MINUS SIGN
         BR DTO2
DTO1:    TSTB TPS
         BPL .-4
         MOVB #4Ø,TPB     ;SPACE FOR POSITIVE NUMBERS
DTO2:    MOV #12,R3       ;1Ø.
         JSR PC,DECA
         INC R1           ;TERMINATE ZERO SUPPRESSION
         MOV #1,R3        ;1.
         JSR PC,DECA
         TSTB TPS
         BPL .-4
         MOVB #56,TPB     ;DECIMAL POINT
```

```
        TST (R5)+       ;INCREMENT R5
        MOV #14631,R3   ;Ø.1=Ø.Ø99 99Ø 844 726 562 5
        JSR PC,DECA
        MOV #122Ø,R3    ;Ø.Ø1=Ø.Ø1Ø ØØ9 765 625
        JSR PC,DECA
        MOV #1Ø2,R3     ;Ø.ØØ1=Ø.ØØ1 ØØ7 Ø8Ø Ø78 125
        JSR PC,DECA
        MOV #7,R3       ;Ø.ØØØ1=Ø.ØØØ 1Ø6 811 523 437 5
        JSR PC,DECA
        RTS PC
        DBWS=.          ;TEMPORARY STORAGE LOCATION
        .=.+6

;SUBROUTINE:  DECA
;    DTOA SERVICE ROUTINE
DECA:   CLR R2          ;COUNTER
        SUB R3,(R5)
        BLO .+6
        INC R2
        BR DECA+2
        ADD R3,(R5)
        TST R2          ;R2=Ø.Ø?
        BNE DEC1        ;NO
        TST R1          ;SUPPRESS ZERO?
        BNE DEC1        ;NO
        SUB #2Ø,R2      ;YES, 6Ø-2Ø=4Ø=ASCII SPACE
        BR .+4
DEC1:   INC R1          ;TERMINATE ZERO SUPPRESSION
        ADD #6Ø,R2      ;FORM ASCII CHARACTER
        TSTB TPS
        BPL .-4
        MOV R2,TPB      ;OUTPUT ASCII CHARACTER
        RTS PC

;MESSAGES
MSC=.
.BYTE 15,12,4Ø,4Ø,4Ø,4Ø
MSD=.
.ASCII /        /
MSE=.
.BYTE 4Ø,4Ø,4Ø,4Ø,4Ø
MSF=.
.BYTE 15,12
.ASCII /                /
MSG=.
.BYTE 15,12
MEG=.
.EVEN
```

```
;STORAGE LOCATIONS
GR1=.
.=.+18.
GR2=.
.=.+18.
EPI=.
.WORD Ø,4ØØØØ,1ØØØØ5      ;(1/E)=16.Ø
WG1=.
.=.+6
WG2=.
.=.+6
HPH=.
.=.+6
HPØ=.
.=.+18.
HRØ=.
.=.+18.
ONE=.
.WORD Ø,4ØØØØ,1ØØØØ1      ;FPN(+1.Ø)
PRØ=.
.=.+54.
PR1=.
.=.+54.
XR1=.
.=.+6
XR2=.
.=.+6

;ADDRESSES FROM SDSMP PROGRAM
.=1Ø72
RSQ=.

.=1336
NDP2=.

.=2Ø22
NEXT=.

.=5346
TINP=.

.=6172
TOUT=.

.=65Ø4
TAD1=.

.END
```

VITA

Carl A. Nehmer was born in Port Huron, Michigan, on March 26, 1950. He attended elementary schools in Croswell, Michigan and was graduated from Croswell-Lexington Community High School in 1968. In August of that year he entered General Motors Institute, Flint, Michigan, under sponsorship of AC Spark Division, General Motors Corporation. At the end of the fourth year he qualified for a General Motors Fellowship under G.M.I.'s Bachelor-Master Plan. He entered the Graduate School at the University of Tennessee in September, 1972, to continue his studies toward a Master of Science degree in Electrical Engineering. He is a member of Tau Beta Pi.