A PARAMETER-ESTIMATION ALGORITHM FOR SMALL DIGITAL COMPUTERS

ROBERT JAMES TAPP

B.Sc., University of Victoria, 1969

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

in the Department
of
Electrical Engineering

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

July, 1972

ABSTRACT

An algorithm is developed for performing parameter estimation on a small-size digital computer. First principles of matrix algebra are used to derive a sequential estimator which computes an estimate of a general parameter array $\underline{A}$ from an array of measurements $\underline{Z} = \underline{H}\,\underline{A} + \underline{V}$ where $\underline{V}$ is a matrix of zero-mean noise terms. At every stage a new row is adjoined to each of $\underline{Z}$, $\underline{H}$ and $\underline{V}$ and a new estimate of $\underline{A}$ is calculated recursively, with any one of three well-known filtering processes available from the same basic set of recursive equations: a least-squares filter to minimize $J = \frac{1}{2}$ trace $(\underline{Z} - \underline{H}\,\hat{\underline{A}})(\underline{Z} - \underline{H}\,\hat{\underline{A}})'$, a maximum-likelihood filter to maximize $p_{\underline{Z}|\underline{A}}(\underline{Z}|\hat{\underline{A}})$ or a maximum-a-posteriori filter to maximize $p_{\underline{A}|\underline{Z}}(\hat{\underline{A}}|\underline{Z})$. Provision is made for starting the filter either with a-priori means and variances of the parameters or with a deterministic "minimum-norm" composition based on the first s measurement rows, s being the number of rows in the parameter array.

The algorithm is applied to the problem of identifying the parameters of a discrete model for a linear time-invariant control system directly from sequential observations of the inputs and outputs. Results from computer tests are used to demonstrate properties of the algorithm and the important computer programs are included, along with suggestions for further applications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I should like to express my gratitude to the National Research Council for the financial support afforded me in the form of a Post-graduate Scholarship (1969-70) and a Post-graduate Bursary (1970-71), and also to the following individuals: my supervisor, Dr. E.V. Bohn, for his patient guidance; Dr. R. Donaldson for his helpful suggestions; Dr. Graham Qualtrough for his useful ideas; Mr. Dave Holmes for his assistance with the computer work; and my good friend and fellow student, Mr. John Wong, for his advice and encouragement.

R. Tapp,

July, 1972.

I. Introduction

When it is necessary to estimate important parameters of a system from measurements of system variables, the choice of an optimal mathematical procedure depends on the amount of statistical information available concerning the system and measurement process. Unfortunately, not enough information is available in many practical situations to permit using well-known estimators like the Kalman filter, nor is it obvious how these procedures can be adapted for simpler problems. Kishi [8], Sage [13], Young [17] and other authors have indicated how classical least-squares filtering can be useful because of its validity in the absence of statistical information and its similarities with more sophisticated methods, but very little has been written in the way of a unified and complete theory of practical least-squares filtering. Greville [3] presents a derivation of least-squares curve fitting which is mathematically rigorous but unnecessarily complicated by the use of generalized-inverse theory and not directly applicable to the problem of parameter estimation. In an attempt to apply it to the estimation problem, Kishi [8] loses some of the mathematical rigour and neglects some important practical considerations. Young [17] and Sinha and Pille [15] have contributed accurate but very simplified descriptions of the method.

There is considerable advantage to be gained by using a classical least-squares estimator as the basis for on-line filtering algorithms because it is straightforward to implement, valid under most conditions and easily modified for a-priori statistical information. It is the purpose of this thesis to develop a complete theory for least-squares filtering, leading to an algorithm that can be programmed on a small digital computer and to considerations of how the algorithm can be extended for a number of practical situations. The mathematical approach used by Greville [2,3] was chosen as the most suitable on which to base the derivations for general least-squares filtering equations, although his use of Penrose's pseudo-inverse theory [11, 12] has been abandoned in favour of a more straightforward approach which employs only first principles of matrix algebra. To include the statistical maximum-likelihood and Bayesian filters, some simple modifications of the equations are considered.

In this thesis all symbols representing vectors and matrices are underscored, with upper-case letters denoting matrices and lower-case letters denoting column-vectors wherever possible. A symbol followed by a prime indicates the transpose of the corresponding matrix or column-vector (example: $\underline{A}'$). Where dimensions of a matrix or vector are given, they are enclosed in parentheses following the symbol (example: $\underline{B}(m \times n)$). The identity matrix is represented by the symbol "$\underline{I}$" and matrix inverses are denoted by the superscript "-1". The symbol for the statistical expected-value operator is "ε".

II. Least-Squares Filtering

An arbitrary but very general representation of the relation between a collection of measurements of system variables and the basic parameters of the system is

$$\underline{Z} = \underline{H}\,\underline{A} + \underline{V} \tag{1}$$

where $\underline{Z}$ is an array containing all the measured data, $\underline{A}$ is the array of unknown fixed parameters, $\underline{H}$ is the matrix representing the defined relationship between the quantities measured and the parameters, and $\underline{V}$ is an array of measurement noise terms. In a simple example of a body moving with a constant velocity v, it is desired to estimate the velocity and initial position s_o of the body from measurements of its position s at known times t. The parameters s_o and v are defined by the equation

$$s = s_o + v\,t$$

If the position is measured at times t_1, t_2 and t_3 and values $\bar{s}_1$, $\bar{s}_2$ and $\bar{s}_3$ are obtained, then a representation corresponding to equation (1) would be

$$\begin{bmatrix} \bar{s}_1 \\ \bar{s}_2 \\ \bar{s}_3 \end{bmatrix} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ 1 & t_3 \end{bmatrix} \begin{bmatrix} s_o \\ v \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}$$

where n_1, n_2 and n_3 are measurement noise terms.

The classical method of least squares assumes that for zero-mean noise the estimate $\hat{\underline{A}}$ of the parameter array $\underline{A}$ should result in a minimum of the sum of the squares of the elements of the matrix $(\underline{Z} - \underline{H}\hat{\underline{A}})$. This corresponds to minimizing the cost function

$$J = \frac{1}{2} \text{ trace } (\underline{Z} - \underline{H}\hat{\underline{A}})(\underline{Z} - \underline{H}\hat{\underline{A}})' \quad (2)$$

If the rows of $\underline{Z}$ are labelled successively $\underline{z}_1', \underline{z}_2', \underline{z}_3', \ldots$, and the rows of $\underline{H}$ are similarly labelled $\underline{h}_1', \underline{h}_2', \underline{h}_3', \ldots$, then the cost function can be written

$$J = \frac{1}{2}\sum_i (\underline{z}_i' - \underline{h}_i'\hat{\underline{A}})(\underline{z}_i' - \underline{h}_i'\hat{\underline{A}})' \quad (3)$$

For a minimum the derivative with respect to $\hat{\underline{A}}$ must be zero:

$$-\sum_i \underline{h}_i(\underline{z}_i' - \underline{h}_i'\hat{\underline{A}}) = \underline{0}$$

$$\sum_i \underline{h}_i\underline{z}_i' = \sum_i \underline{h}_i\underline{h}_i'\hat{\underline{A}}$$

$$\underline{H}'\underline{Z} = \underline{H}'\underline{H}\hat{\underline{A}} \quad (4)$$

If the number of rows in $\underline{H}$ is greater than or equal to the number of columns and the columns are linearly independent then the column vector $\underline{H}\underline{u}$, which is a linear combination of the columns of $\underline{H}$, is non-zero for all non-zero $\underline{u}$. Therefore $\underline{u}'\underline{H}'\underline{H}\underline{u}$ is positive for all non-zero $\underline{u}$ which means that $\underline{H}'\underline{H}$ is positive definite and hence non-singular. (4) then gives the unique solution

$$\underline{\hat{A}} = (\underline{H}'\underline{H})^{-1}\underline{H}'\underline{Z} \tag{5}$$

If the number of rows in $\underline{H}$ is less than or equal to the number of columns and the rows are linearly independent then the row vector $\underline{u}'\underline{H}$, which is a linear combination of the rows of $\underline{H}$, is non-zero for all non-zero $\underline{u}$. Thus $\underline{u}'\underline{H}\,\underline{H}'\underline{u}$ is positive for all non-zero $\underline{u}$ and $\underline{H}\,\underline{H}'$ is positive definite and therefore nonsingular. Pre-multiplying both sides of (4) by $\underline{H}$ gives

$$\underline{H}\,\underline{H}'\underline{Z} = \underline{H}\,\underline{H}'\underline{H}\,\underline{\hat{A}}$$

$$\underline{Z} = \underline{H}\,\underline{\hat{A}} \tag{6}$$

Except for the case where $\underline{H}$ is square, this equation does not have a unique solution, but although no unique solution can be defined on the basis of the least-squares criterion alone it will nevertheless be desirable to define some arbitrary solution. The most logical choice is that least-squares solution which has a minimum "norm" and is found by minimizing the cost function

$$J_n = \frac{1}{2}\,\text{trace}\,(\underline{\hat{A}}\,\underline{\hat{A}}') \tag{7}$$

subject to equation (6). Using Lagrange's method of undetermined multipliers, an augmented cost function is defined:

$$J_a = \text{trace}\left[\frac{1}{2}\,\underline{\hat{A}}\,\underline{\hat{A}}' + \underline{\lambda}(\underline{Z} - \underline{H}\,\underline{\hat{A}})\right] \tag{8}$$

where $\underline{\lambda}$ is the array of undetermined multipliers. Now

$$\frac{\partial J_a}{\partial \underline{A}} = \underline{\hat{A}} - \underline{H}'\underline{\lambda}' = \underline{0}$$

$$\underline{\hat{A}} = \underline{H}'\underline{\lambda}' \tag{9}$$

Using (9) in (6),

$$\underline{Z} = \underline{H}\,\underline{H}'\underline{\lambda}'$$

$$\underline{\lambda}' = (\underline{H}\,\underline{H}')^{-1}\,\underline{Z} \tag{10}$$

Using (10) in (9),

$$\underline{\hat{A}} = \underline{H}'(\underline{H}\,\underline{H}')^{-1}\,\underline{Z} \tag{11}$$

This equation will define the least-squares estimate of $\underline{A}$ whenever the number of rows of $\underline{H}$ is less than or equal to its number of columns and the rows are linearly independent.

For the many applications where the observations are not available all at once but are received sequentially in time, it is desirable to have a recursive relation which will provide parameter estimates at every stage by updating prior estimates as each new set or block of data arrives. The addition of more data to the $\underline{Z}$ matrix will require that elements be added to the $\underline{H}$ matrix and since the dimensions of $\underline{H}$ will be changing at every stage it is important to establish which of equations (5) and (11) should be used to determine the estimate at each stage.

If the parameter matrix $\underline{A}$ is to have fixed dimensions, labelled (s x r), then equation (1) shows that $\underline{H}$ must always have s columns and $\underline{Z}$ must always have r columns. Thus in this scheme, elements adjoined to the $\underline{H}$ and $\underline{Z}$ matrices at sequential stages must take the form of additional rows. If q is the number of rows adjoined to each of $\underline{Z}$ and $\underline{H}$ at every estimation stage, then the total number of rows in each matrix is kq where k is the number of the current estimation stage. To summarize the dimension labels, (1) can be re-written

$$\underline{Z}(kq\times r) = \underline{H}(kq\times s)\,\underline{A}(s\times r) + \underline{V}(kq\times r) \tag{12}$$

Now, using (5) and (11), the least-squares estimate for $\underline{A}$ at stage k is defined by

$$\hat{\underline{A}}_k = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}\,\underline{Z}_k\,, \qquad kq \leqq s \tag{13}$$

$$\hat{\underline{A}}_k = (\underline{H}_k'\underline{H}_k)^{-1}\,\underline{H}_k'\,\underline{Z}_k\,, \qquad kq \geqq s \tag{14}$$

where $\underline{H}_k$ and $\underline{Z}_k$ are the matrices $\underline{H}$ and $\underline{Z}$ at stage k. If $q \geqq s$ then (14) will apply for all values of k, but if $q < s$ then (13) will apply until k exceeds $\frac{s}{q}$ and (14) will apply for all further stages. In designing a general recursive relation for (13) and (14), advantage can be taken of the fact that both solutions would apply for a stage k where $kq = s$, provided the rows of $\underline{H}$ are linearly independent. $\underline{H}_k$ would be square and nonsingular and (13) and (14) would reduce to

$$\hat{\underline{A}}_k = \underline{H}_k^{-1}\,\underline{Z}_k\,, \qquad kq = s \tag{15}$$

Thus if the number of rows adjoined to $\underline{H}_k$ at each stage (q) is a factor of its number of columns (s) then there will be a stage where kq = s such that both (13) and (14) are valid and the final solution from the recursive form of (13) can be used as the starting value for the recursive form of (14).

To obtain the recursive forms for equations (13) and (14) it is convenient to introduce new symbols $\underline{G}_k$ and $\underline{J}_k$ defined by

$$\underline{G}_k = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}, \quad kq \leqq s \tag{16}$$

$$\underline{J}_k = (\underline{H}_k'\underline{H}_k)^{-1}\underline{H}_k', \quad kq \geqq s \tag{17}$$

In the theory of generalized inverses $\underline{G}_k$ would be called the right generalized inverse or right pseudo-inverse of $\underline{H}_k$ and $\underline{J}_k$ would be called the left generalized inverse or left pseudo-inverse of $\underline{H}_k$. The matrices $\underline{Z}_k$, $\underline{H}_k$, $\underline{G}_k$ and $\underline{J}_k$ are partitioned as follows:

$$\underline{Z}_k(kq\times r) = \begin{bmatrix} \underline{Z}_{k-1}([k-1]q\times r) \\ \cdots\cdots\cdots\cdots \\ \underline{Z}_k^*(q\times r) \end{bmatrix} \tag{18}$$

$$\underline{H}_k(kq\times s) = \begin{bmatrix} \underline{H}_{k-1}([k-1]q\times s) \\ \cdots\cdots\cdots\cdots \\ \underline{H}_k^*(q\times s) \end{bmatrix} \tag{19}$$

$$\underline{G}_k(s\times kq) = \begin{bmatrix} \underline{F}_k(s\times[k-1]q) & \vdots & \underline{E}_k(s\times q) \end{bmatrix} \tag{20}$$

$$\underline{J}_k(s \times kq) = \left[\underline{D}_k(s \times [k-1]\, q) \;\vdots\; \underline{B}_k(s \times q)\right] \tag{21}$$

Equation (13) can now be written

$$\hat{\underline{A}}_k = \underline{G}_k\underline{Z}_k = \underline{F}_k\underline{Z}_{k-1} + \underline{E}_k\underline{Z}_k^{*}, \qquad kq \leqq s \tag{22}$$

To solve for $\underline{F}_k$ and $\underline{E}_k$, define the matrix

$$\underline{Q}_k = \underline{G}_k\underline{H}_k = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}\,\underline{H}_k = \underline{F}_k\underline{H}_{k-1} + \underline{E}_k\underline{H}_k^{*} \tag{23}$$

Post-multiplying by $\underline{H}_k'$ gives

$$\underline{H}_k' = \underline{F}_k\underline{H}_{k-1}\underline{H}_k' + \underline{E}_k\underline{H}_k^{*}\,\underline{H}_k' \tag{24}$$

Using (19), this can be written as two equations:

$$\underline{H}_{k-1}' = \underline{F}_k\underline{H}_{k-1}\underline{H}_{k-1}' + \underline{E}_k\underline{H}_k^{*}\,\underline{H}_{k-1}' \tag{25}$$

$$\underline{H}_k^{*\prime} = \underline{F}_k\underline{H}_{k-1}\underline{H}_k^{*\prime} + \underline{E}_k\underline{H}_k^{*}\,\underline{H}_k^{*\prime} \tag{26}$$

From (25)

$$\underline{F}_k = \underline{H}_{k-1}'(\underline{H}_{k-1}\underline{H}_{k-1}')^{-1} - \underline{E}_k\underline{H}_k^{*}\,\underline{H}_{k-1}'(\underline{H}_{k-1}\underline{H}_{k-1}')^{-1} \tag{27}$$

Substituting this into (22) gives

$$\hat{\underline{A}}_k = \hat{\underline{A}}_{k-1} - \underline{E}_k\underline{H}_k^{*}\hat{\underline{A}}_{k-1} + \underline{E}_k\underline{Z}_k^{*}, \qquad kq \leqq s$$

$$\hat{\underline{A}}_k = \hat{\underline{A}}_{k-1} + \underline{E}_k(\underline{Z}_k^{*} - \underline{H}_k^{*}\hat{\underline{A}}_{k-1}), \qquad kq \leqq s \tag{28}$$

and into (23) gives

$$\underline{Q}_k = \underline{Q}_{k-1} - \underline{E}_k \underline{H}_k^* \underline{Q}_{k-1} + \underline{E}_k \underline{H}_k^*$$

$$\underline{Q}_k = \underline{Q}_{k-1} + \underline{E}_k \underline{H}_k^* (\underline{I} - \underline{Q}_{k-1}) \tag{29}$$

and into (26) gives

$$\underline{H}_k^{*\prime} = \underline{Q}_{k-1} \underline{H}_k^{*\prime} - \underline{E}_k \underline{H}_k^* \underline{Q}_{k-1} \underline{H}_k^{*\prime} + \underline{E}_k \underline{H}_k^* \underline{H}_k^{*\prime}$$

$$\underline{E}_k = (\underline{I} - \underline{Q}_{k-1}) \underline{H}_k^{*\prime} \left[\underline{H}_k^* (\underline{I} - \underline{Q}_{k-1}) \underline{H}_k^{*\prime} \right]^{-1} \tag{30}$$

Equations (28), (29) and (30) constitute the recursive relation which corresponds to equation (13). It may be verified from these equations that the correct starting values for $\underline{\hat{A}}$ and $\underline{Q}$ are zero, for then

$$\underline{E}_1 = \underline{H}_1^{*\prime} (\underline{H}_1^* \underline{H}_1^{*\prime})^{-1}$$

$$\underline{\hat{A}}_1 = \underline{H}_1^{*\prime} (\underline{H}_1^* \underline{H}_1^{*\prime})^{-1} \underline{Z}_1^*$$

$$\underline{Q}_1 = \underline{H}_1^{*\prime} (\underline{H}_1^* \underline{H}_1^{*\prime})^{-1} \underline{H}_1^*$$

which are consistent with the definitions of $\underline{\hat{A}}_k$ and $\underline{Q}_k$ in (13) and (23).

Using (18) and (21), equation (14) can be written

$$\hat{\underline{A}}_k = \underline{J}_k \underline{Z}_k = \underline{D}_k \underline{Z}_{k-1} + \underline{B}_k \underline{Z}_k^* , \quad kq \geqq s \tag{31}$$

To solve for $\underline{D}_k$ and $\underline{B}_k$, begin by forming the product $\underline{H}_k \underline{J}_k$ using (17), (19) and (21):

$$\underline{H}_k \underline{J}_k = \underline{H}_k (\underline{H}_k' \underline{H}_k)^{-1} \underline{H}_k' = \left[\begin{array}{c:c} \underline{H}_{k-1} \underline{D}_k & \underline{H}_{k-1} \underline{B}_k \\ \hdashline \underline{H}_k^* \underline{D}_k & \underline{H}_k^* \underline{B}_k \end{array} \right] \tag{32}$$

Pre-multiplying by $\underline{H}_k'$ gives

$$\underline{H}_k' = \underline{H}_k' \left[\begin{array}{c:c} \underline{H}_{k-1} \underline{D}_k & \underline{H}_{k-1} \underline{B}_k \\ \hdashline \underline{H}_k^* \underline{D}_k & \underline{H}_k^* \underline{B}_k \end{array} \right] \tag{33}$$

Using (19) this can be written as the two equations

$$\underline{H}_{k-1}' = \underline{H}_{k-1}' \underline{H}_{k-1} \underline{D}_k + \underline{H}_k^{*'} \underline{H}_k^* \underline{D}_k \tag{34}$$

$$\underline{H}_k^{*'} = \underline{H}_{k-1}' \underline{H}_{k-1} \underline{B}_k + \underline{H}_k^{*'} \underline{H}_k^* \underline{B}_k \tag{35}$$

From (34)

$$\underline{D}_k = (\underline{H}_{k-1}' \underline{H}_{k-1} + \underline{H}_k^{*'} \underline{H}_k^*)^{-1} \underline{H}_{k-1}' \tag{36}$$

and from (35)

$$\underline{B}_k = (\underline{H}_{k-1}' \underline{H}_{k-1} + \underline{H}_k^{*'} \underline{H}_k^*)^{-1} \underline{H}_k^{*'} \tag{37}$$

If a new matrix is defined by

$$\underline{P}_k = \underline{J}_k\underline{J}_k' = (\underline{H}_k'\underline{H}_k)^{-1}\underline{H}_k'\underline{H}_k(\underline{H}_k'\underline{H}_k)^{-1} = (\underline{H}_k'\underline{H}_k)^{-1}$$

$$= (\underline{H}_{k-1}'\underline{H}_{k-1} + \underline{H}_k^{*\prime}\underline{H}_k^{*})^{-1} \qquad (38)$$

then (36) and (37) can be written as

$$\underline{D}_k = \underline{P}_k\underline{H}_{k-1}' \qquad (39)$$

$$\underline{B}_k = \underline{P}_k\underline{H}_k^{*\prime} \qquad (40)$$

From (38)

$$\underline{P}_k^{-1} = \underline{H}_{k-1}'\underline{H}_{k-1} + \underline{H}_k^{*\prime}\underline{H}_k^{*}$$

$$= \underline{P}_{k-1}^{-1} + \underline{H}_k^{*\prime}\underline{H}_k^{*} \qquad (41)$$

Pre-multiplying by $\underline{P}_k$ and post-multiplying by $\underline{P}_{k-1}$ gives

$$\underline{P}_{k-1} = \underline{P}_k + \underline{P}_k\underline{H}_k^{*\prime}\underline{H}_k^{*}\underline{P}_{k-1} \qquad (42)$$

Using (40) this becomes

$$\underline{P}_{k-1} = \underline{P}_k + \underline{B}_k\underline{H}_k^{*}\underline{P}_{k-1}$$

$$\underline{P}_k = \underline{P}_{k-1} - \underline{B}_k\underline{H}_k^{*}\underline{P}_{k-1} \qquad (43)$$

and using this result in (39) gives

$$\underline{D}_k = \underline{P}_{k-1}\underline{H}'_{k-1} - \underline{B}_k\underline{H}^*_k\underline{P}_{k-1}\underline{H}'_{k-1} \tag{44}$$

and in (40) gives

$$\underline{B}_k = \underline{P}_{k-1}\underline{H}^{*'}_k - \underline{B}_k\underline{H}^*_k\underline{P}_{k-1}\underline{H}^{*'}_k$$

$$\underline{B}_k = \underline{P}_{k-1}\underline{H}^{*'}_k(\underline{I} + \underline{H}^*_k\underline{P}_{k-1}\underline{H}^{*'}_k)^{-1} \tag{45}$$

Using (44) in (31)

$$\hat{\underline{A}}_k = \underline{P}_{k-1}\underline{H}'_{k-1}\underline{Z}_{k-1} - \underline{B}_k\underline{H}^*_k\underline{P}_{k-1}\underline{H}'_{k-1}\underline{Z}_{k-1} + \underline{B}_k\underline{Z}_k, \quad kq \geqq s$$

Since $\underline{P}_{k-1} = (\underline{H}'_{k-1}\underline{H}_{k-1})^{-1}$, the last equation becomes

$$\hat{\underline{A}}_k = \hat{\underline{A}}_{k-1} + \underline{B}_k(\underline{Z}^*_k - \underline{H}^*_k\hat{\underline{A}}_{k-1}), \quad kq \geqq s \tag{46}$$

Equations (43), (45) and (46) provide the recursive relation corresponding to equation (14) and can be started by applying (14) and (38) directly to the first stage k such that $kq \geqq s$, which will require inversion of at least an $s \times s$ matrix. Since matrix inversion requires fairly complex programming on a small computer, it is perhaps better to arrange that the starting value for (46) be taken from the last solution of (28) at a stage k where $kq = s$, as described earlier. Similarly a recursive relation can be found which will provide a starting value for $\underline{P}_k$ in (43) when $kq = s$. From (38) the definition of $\underline{P}_k$ is

$$\underline{P}_k = \underline{J}_k\underline{J}'_k$$

and from equations (16) and (17)

$$\underline{G}_k = \underline{J}_k = \underline{H}_k^{-1}, \quad kq = s$$

Therefore

$$\underline{P}_k = \underline{J}_k\underline{J}_k' = \underline{G}_k\underline{G}_k', \quad kq = s \tag{47}$$

Thus at a stage k where $kq = s$ it is possible to obtain the starting value for $\underline{P}_k$ from a recursive relation for

$$\underline{R}_k = \underline{G}_k\underline{G}_k' = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}(\underline{H}_k\underline{H}_k')^{-1}\underline{H}_k \tag{48}$$

Using (20) this can be written

$$\underline{R}_k = \underline{F}_k\underline{F}_k' + \underline{E}_k\underline{E}_k' \tag{49}$$

From (27)

$$\underline{F}_k = (\underline{I} - \underline{E}_k\underline{H}_k^*)\underline{H}_{k-1}'(\underline{H}_{k-1}\underline{H}_{k-1}')^{-1} \tag{50}$$

Substituting this into (49) gives

$$\underline{R}_k = (\underline{I} - \underline{E}_k\underline{H}_k^*)\underline{H}_{k-1}'(\underline{H}_{k-1}\underline{H}_{k-1}')^{-1}(\underline{H}_{k-1}\underline{H}_{k-1}')^{-1}\underline{H}_{k-1}$$

$$\times (\underline{I} - \underline{E}_k\underline{H}_k^*)' + \underline{E}_k\underline{E}_k'$$

$$= (\underline{I} - \underline{E}_k\underline{H}_k^*)\underline{R}_{k-1}(\underline{I} - \underline{E}_k\underline{H}_k^*)' + \underline{E}_k\underline{E}_k'$$

$$\underline{R}_k = \underline{R}_{k-1} - \underline{R}_{k-1}\underline{H}_k^{*'}\underline{E}_k' - \underline{E}_k\underline{H}_k^{*}\underline{R}_{k-1}$$

$$+ \underline{E}_k\underline{H}_k^{*}\underline{R}_{k-1}\underline{H}_k^{*'}\underline{E}_k' + \underline{E}_k\underline{E}_k' \qquad (51)$$

which is in a convenient form to be calculated in conjunction with (28), (29) and (30).

The final general algorithm for least-squares estimation of the parameter matrix $\underline{A}$ would therefore use equations (28), (29), (30) and (51) for all estimation stages k such that $kq \leqq s$ and for all subsequent stages would use equations (43), (45) and (46) beginning with the values of $\hat{\underline{A}}_k$ and $\underline{P}_k$ given by (28) and (51) at a stage k where $kq = s$.

The calculations involved in these equations are easily performed on a small computer, apart from the following inverses which appear in (30) and (45) respectively:

$$\left[\underline{H}_k^{*}(\underline{I} - \underline{Q}_{k-1})\underline{H}_k^{*'}\right]^{-1} \qquad\qquad (\underline{I} + \underline{H}_k^{*}\underline{P}_{k-1}\underline{H}_k^{*'})^{-1}$$

As shown in (19) the dimension of $\underline{H}_k^{*}$ is $q \times s$ which indicates that both of the matrices being inverted above have dimension $q \times q$, q being the number of rows adjoined to $\underline{Z}$ and $\underline{H}$ at each estimation stage. Thus by choosing $q = 1$, both inverses will involve scalars and the necessary computer programming will be vastly simplified. The number of rows adjoined at each estimation stage need have no effect on the number of rows adjoined at each measurement stage because the measured rows can be stored and adjoined in the estimation algorithm one at a time. Selecting $q = 1$ also has the advantage that q will always be a divisor of s, the number of columns in $\underline{H}$, which is the requirement for proper linking of the two sets of equations as previously explained.

When $q = 1$, the matrices $\underline{Z}_k^*$ and $\underline{H}_k^*$ degenerate to row vectors and $\underline{E}_k$ and $\underline{B}_k$ degenerate to column vectors. For this reason it is desirable to change the notation and replace

$\underline{Z}_k^*$ by $\underline{z}_k'$ $\quad$ $\underline{E}_k$ by $\underline{e}_k$

$\underline{H}_k^*$ by $\underline{h}_k'$ $\quad$ $\underline{B}_k$ by $\underline{b}_k$

Equation (30) now becomes

$$\underline{e}_k = (\underline{I} - \underline{Q}_{k-1})\underline{h}_k \left[\underline{h}_k'(\underline{I} - \underline{Q}_{k-1})\underline{h}_k\right]^{-1} \tag{52}$$

If the column vector $(\underline{I} - \underline{Q}_{k-1})\underline{h}_k$ in this equation is given the symbol $\underline{c}_k$,

$$\underline{c}_k = (\underline{I} - \underline{Q}_{k-1})\underline{h}_k \tag{53}$$

then from the definition of $\underline{Q}_k$ in equation (23), which was

$$\underline{Q}_k = \underline{G}_k\underline{H}_k = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}\,\underline{H}_k$$

it can be seen that

$$\underline{c}_k' = \underline{h}_k'(\underline{I} - \underline{Q}_{k-1})' = \underline{h}_k'(\underline{I} - \underline{Q}_{k-1}) \tag{54}$$

and

$$\underline{c}_k'\underline{c}_k = \underline{h}_k'(\underline{I} - \underline{Q}_{k-1})(\underline{I} - \underline{Q}_{k-1})\underline{h}_k$$

$$= \underline{h}_k'(\underline{I} - \underline{Q}_{k-1} - \underline{Q}_{k-1} + \underline{Q}_{k-1}\underline{Q}_{k-1})\underline{h}_k$$

$$= \underline{h}_k'(\underline{I} - \underline{Q}_{k-1} - \underline{Q}_{k-1} + \underline{Q}_{k-1})\underline{h}_k$$

$$= \underline{h}_k'(\underline{I} - \underline{Q}_{k-1})\underline{h}_k \tag{55}$$

so that equation (52) can now be written

$$\underline{e}_k = \underline{c}_k(\underline{c}_k'\underline{c}_k)^{-1} \tag{56}$$

and equation (29) now becomes

$$\underline{Q}_k = \underline{Q}_{k-1} + \underline{e}_k\underline{h}_k'(\underline{I} - \underline{Q}_{k-1}) = \underline{Q}_{k-1} + \underline{e}_k\underline{c}_k' \tag{57}$$

Following is a summary of the major equations and their starting values for the simplified algorithm where $q = 1$:

$$k \leqq s \begin{cases} \underline{c}_k = (\underline{I} - \underline{Q}_{k-1})\underline{h}_k & (53) \\ \underline{e}_k = \underline{c}_k(\underline{c}_k'\underline{c}_k)^{-1} & (56) \\ \underline{Q}_k = \underline{Q}_{k-1} + \underline{e}_k\underline{c}_k', \quad \underline{Q}_k = \underline{0} \text{ at } k = 0 & (57) \end{cases}$$

$$
k \leqq s \begin{cases}
\underline{R}_k = \underline{R}_{k-1} - \underline{R}_{k-1}\underline{h}_k\underline{e}_k' - \underline{e}_k\underline{h}_k'\underline{R}_{k-1} + \underline{e}_k\underline{h}_k'\underline{R}_{k-1}\underline{h}_k\underline{e}_k' + \underline{e}_k\underline{e}_k' , \\
\qquad \underline{R}_k = \underline{0} \text{ at } k = 0 & (58) \\
\hat{\underline{A}}_k = \hat{\underline{A}}_{k-1} + \underline{e}_k(\underline{z}_k' - \underline{h}_k'\hat{\underline{A}}_{k-1}) , \quad \hat{\underline{A}}_k = \underline{0} \text{ at } k = 0 & (59)
\end{cases}
$$

$$
k > s \begin{cases}
\underline{b}_k = \underline{P}_{k-1}\underline{h}_k(1 + \underline{h}_k'\underline{P}_{k-1}\underline{h}_k)^{-1} & (60) \\
\underline{P}_k = \underline{P}_{k-1} - \underline{b}_k\underline{h}_k'\underline{P}_{k-1} , \quad \underline{P}_k = \underline{R}_k \text{ at } k = s & (61) \\
\hat{\underline{A}}_k = \hat{\underline{A}}_{k-1} + \underline{b}_k(\underline{z}_k' - \underline{h}_k'\hat{\underline{A}}_{k-1}) ,
\end{cases}
$$

$$
\hat{\underline{A}}_k = \hat{\underline{A}}_k \text{ from (59) at } k = s \qquad (62)
$$

It has already been shown that when the rows of the matrix $\underline{H}_k$ are linearly independent, the product $\underline{H}_k\underline{H}_k'$ is nonsingular and the matrix

$$
\underline{Q}_k = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}\underline{H}_k
$$

is a left identity for the matrix $\underline{H}_k'$ because

$$
\underline{Q}_k\underline{H}_k' = \underline{H}_k'(\underline{H}_k\underline{H}_k')^{-1}\underline{H}_k\underline{H}_k' = \underline{H}_k'
$$

Similarly $\underline{Q}_k$ is a left identity for any other matrix whose columns lie in the transposed row space of $\underline{H}_k$. Thus if a new row $\underline{h}_k'$ is adjoined to the $\underline{H}$ matrix and is a linear combination of the previous k-1 rows, then the vector $\underline{h}_k$ will lie in the transposed row space of $\underline{H}_{k-1}$ and equation (53) will give

$$\underline{c}_k = (\underline{I} - \underline{Q}_{k-1})\underline{h}_k = \underline{0} \tag{63}$$

Since the recursive least-squares procedure requires that $\underline{H}$ have maximum rank at every stage and in particular that the rows be linearly independent for the first s stages, $\underline{c}_k$, being the first calculation involving a new row of $\underline{H}$, is an extremely useful indicator of this condition. Depending on the process involved, a measurement which would make the rows of $\underline{H}$ linearly dependent can be rejected in favour of a new measurement or the entire process can be re-started with a minimum of wasted time.

Although at this point all the essential equations for a least-squares filtering algorithm have been developed, a preliminary comparison with statistical methods will lead to minor improvements which make the algorithm much more useful. In the next chapter will be presented a derivation of the statistical maximum-likelihood filter which parallels that of the least-squares filter in this chapter. Chapter IV will then describe the complete mechanics of the final computational algorithm which was used in the research project outlined in Chapter V.

III. Maximum-Likelihood Filtering

A maximum-likelihood procedure gives the optimum minimum-variance parameter estimate when no a-priori statistical information is available concerning the parameters and the noise terms affecting the measurements are zero-mean independent white-Gaussian random variables of known variance.

The development of the maximum-likelihood filtering equations in this chapter follows closely that of the least-squares filter in the previous chapter in order that similarities between the two methods will be apparent. This should facilitate explanation of the general-purpose computational algorithm to be presented in Chapter IV.

As in Chapter II, equation (1) will be the arbitrary representation of the measurement process, where as before the matrix $\underline{H}$ is assumed to have maximum rank. The optimum estimate $\hat{\underline{A}}$ of the parameters $\underline{A}$ is chosen so that the probability density of each measured quantity conditional on $\underline{A} = \hat{\underline{A}}$ has a maximum at the observed value of the measured quantity. The probability density function involved is often given the name "likelihood function" and since the noise terms are statistically independent the likelihood function for a row i of measurements is the product of their individual likelihood functions, which are Gaussian:

$$p_{\underline{z}_i|\underline{A}} = \frac{1}{(2\pi)^{r/2}\, s_i^{r/2}} \exp\left[-\frac{1}{2s_i}\,(\underline{z}_i' - \underline{h}_i'\underline{A})(\underline{z}_i' - \underline{h}_i'\underline{A})'\right] \tag{64}$$

where r is the number of measurements in a row or the number of columns in the measurement array and it has been assumed that the noise on each measurement of a row has the same variance s_i. This latter assumption results in a great simplification to the derivation which follows and does not seriously limit the usefulness of the equations, because measurements having different noise-variances can always be located in separate rows. A product of the likelihood functions of all k rows gives the likelihood function for the entire measurement set at stage k:

$$p_{\underline{Z}|\underline{A}} = \frac{1}{(2\pi)^{kr/2}(\det \underline{S})^{r/2}} \exp\left[-\frac{1}{2}\sum_i s_i^{-1}(\underline{z}_i' - \underline{h}_i'\underline{A})(\underline{z}_i' - \underline{h}_i'\underline{A})'\right] \quad (65)$$

where $\underline{S} = \begin{bmatrix} s_1 & 0 & 0 & \cdot & \cdot & \cdot \\ 0 & s_2 & 0 & \cdot & \cdot & \cdot \\ 0 & 0 & s_3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$, a positive-definite matrix.

Maximizing this likelihood function is equivalent to maximizing its logarithm:

$$\log p_{\underline{Z}|\underline{A}} = -\frac{1}{2}\sum_i s_i^{-1} (\underline{z}_i' - \underline{h}_i'\underline{A})(\underline{z}_i' - \underline{h}_i'\underline{A})' - \frac{kr}{2} \log (2\pi)$$

$$- \frac{r}{2} \log (\det S) \qquad (66)$$

and a maximum results when the derivative with respect to $\underline{A}$ is zero:

$$\sum_i s_i^{-1} \underline{h}_i (\underline{z}_i' - \underline{h}_i' \hat{\underline{A}}) = \underline{0}$$

$$\underline{H}'\underline{S}^{-1}\underline{Z} = \underline{H}'\underline{S}^{-1}\underline{H}\hat{\underline{A}} \tag{67}$$

When $\underline{H}$ has fewer rows than columns, $(\underline{H}\underline{H}')^{-1}$ exists and the last equation reduces to

$$\underline{Z} = \underline{H}\hat{\underline{A}} \tag{68}$$

This is identical to the least-squares result of equation (6) and the minimum-norm estimates for the two methods are the same:

$$\hat{\underline{A}} = \underline{H}'(\underline{H}\underline{H}')^{-1}\underline{Z} \tag{69}$$

When $\underline{H}$ has more rows than columns, $\underline{H}'\underline{H}$ is positive definite and so is $\underline{H}'\underline{S}^{-1}\underline{H}$. Thus the maximum-likelihood estimate from equation (67) is

$$\hat{\underline{A}} = (\underline{H}'\underline{S}^{-1}\underline{H})^{-1}\underline{H}'\underline{S}^{-1}\underline{Z} \tag{70}$$

which is the least-squares solution of (5) weighted by the inverse of the noise variance matrix $\underline{S}$.

A recursive relation for equation (70), unlike the method of least squares, cannot theoretically be started using the minimum-norm result of (69) at stage $k = s$ because (69) does not contain the information regarding the noise variance for stages $k \leqq s$ that is required by (70). This problem will be discussed later. It is first necessary to obtain a recursive form for (70).

Following a procedure similar to that used for the recursive form of (5), the maximum-likelihood estimate at a stage k is written as

$$\hat{\underline{A}}_k = \underline{J}_k \underline{Z}_k \tag{71}$$

where $\underline{J}_k$ is now defined by

$$\underline{J}_k = (\underline{H}_k' \underline{S}_k^{-1} \underline{H}_k)^{-1} \underline{H}_k' \underline{S}_k^{-1} \tag{72}$$

To make the equations compatible with the algorithm derived near the end of Chapter II, one row only will be adjoined to each of $\underline{H}_k$ and $\underline{Z}_k$ at every stage, and the following partitionings are valid:

$$\underline{Z}_k(k \times r) = \begin{bmatrix} \underline{Z}_{k-1}([k-1] \times r) \\ \cdots\cdots\cdots \\ \underline{z}_k'(1 \times r) \end{bmatrix} \tag{73}$$

$$\underline{H}_k(k \times s) = \begin{bmatrix} \underline{H}_{k-1}([k-1] \times s) \\ \cdots\cdots\cdots \\ \underline{h}_k'(1 \times s) \end{bmatrix} \tag{74}$$

$$\underline{J}_k(s \times k) = \begin{bmatrix} \underline{D}_k(s \times [k-1]) & \vdots & \underline{b}_k(s \times 1) \end{bmatrix} \tag{75}$$

$$\underline{S}_k^{-1} = \begin{bmatrix} \underline{S}_{k-1}^{-1} & \vdots & 0 \\ \cdots\cdots & \vdots & \cdots\cdots \\ 0 & \vdots & s_k^{-1} \end{bmatrix} \tag{76}$$

Now

$$\underline{H}_k\underline{J}_k = \underline{H}_k(\underline{H}_k'\underline{S}_k^{-1}\underline{H}_k)^{-1}\underline{H}_k'\underline{S}_k^{-1}$$

$$= \begin{bmatrix} \underline{H}_{k-1}\underline{D}_k & \vdots & \underline{H}_{k-1}\underline{b}_k \\ \cdots & & \cdots \\ \underline{h}_k'\underline{D}_k & \vdots & \underline{h}_k'\underline{b}_k \end{bmatrix} \quad (77)$$

Pre-multiplying by $\underline{H}_k'\underline{S}_k^{-1}$,

$$\underline{H}_k'\underline{S}_k^{-1} = \underline{H}_k'\underline{S}_k^{-1} \begin{bmatrix} \underline{H}_{k-1}\underline{D}_k & \vdots & \underline{H}_{k-1}\underline{b}_k \\ \cdots & & \cdots \\ \underline{h}_k'\underline{D}_k & \vdots & \underline{h}_k'\underline{b}_k \end{bmatrix} \quad (78)$$

Using (74) and (76) this can be written as the two equations

$$\underline{H}_{k-1}'\underline{S}_{k-1}^{-1} = \underline{H}_{k-1}'\underline{S}_{k-1}^{-1}\underline{H}_{k-1}\underline{D}_k + \underline{h}_k s_k^{-1}\underline{h}_k'\underline{D}_k \quad (79)$$

$$\underline{h}_k s_k^{-1} = \underline{H}_{k-1}'\underline{S}_{k-1}^{-1}\underline{H}_{k-1}\underline{b}_k + \underline{h}_k s_k^{-1}\underline{h}_k'\underline{b}_k \quad (80)$$

From (79)

$$\underline{D}_k = (\underline{H}_{k-1}'\underline{S}_{k-1}^{-1}\underline{H}_{k-1} + \underline{h}_k s_k^{-1}\underline{h}_k')^{-1}\underline{H}_{k-1}'\underline{S}_{k-1}^{-1} \quad (81)$$

From (80)

$$\underline{b}_k = (\underline{H}_{k-1}'\underline{S}_{k-1}^{-1}\underline{H}_{k-1} + \underline{h}_k s_k^{-1}\underline{h}_k')^{-1}\underline{h}_k s_k^{-1} \quad (82)$$

Defining

$$\underline{P}_k = (\underline{H}_k'\underline{S}_k^{-1}\underline{H}_k)^{-1} = (\underline{H}_{k-1}'\underline{S}_{k-1}^{-1}\underline{H}_{k-1} + \underline{h}_k s_k^{-1}\underline{h}_k')^{-1} \quad (83)$$

(81) and (82) become

$$\underline{D}_k = \underline{P}_k \underline{H}_{k-1}' \underline{S}_{k-1}^{-1} \tag{84}$$

$$\underline{b}_k = \underline{P}_k \underline{h}_k s_k^{-1} \tag{85}$$

From (83)

$$\underline{P}_k^{-1} = \underline{P}_{k-1}^{-1} + \underline{h}_k s_k^{-1} \underline{h}_k' \tag{86}$$

Pre-multiplying by $\underline{P}_k$ and post-multiplying by $\underline{P}_{k-1}$,

$$\underline{P}_{k-1} = \underline{P}_k + \underline{P}_k \underline{h}_k s_k^{-1} \underline{h}_k' \underline{P}_{k-1} \tag{87}$$

Using (85) this becomes

$$\underline{P}_k = \underline{P}_{k-1} - \underline{b}_k \underline{h}_k' \underline{P}_{k-1} \tag{88}$$

and using this result in (84) gives

$$\underline{D}_k = \underline{P}_{k-1} \underline{H}_{k-1}' \underline{S}_{k-1}^{-1} - \underline{b}_k \underline{h}_k' \underline{P}_{k-1} \underline{H}_{k-1}' \underline{S}_{k-1}^{-1} \tag{89}$$

and in (85),

$$\underline{b}_k = \underline{P}_{k-1} \underline{h}_k s_k^{-1} - \underline{b}_k \underline{h}_k' \underline{P}_{k-1} \underline{h}_k s_k^{-1}$$

$$\underline{b}_k = \underline{P}_{k-1} \underline{h}_k (s_k + \underline{h}_k' \underline{P}_{k-1} \underline{h}_k)^{-1} \tag{90}$$

Using (73), (75) and (89), equation (77) can be written

$$\hat{\underline{A}}_k = \hat{\underline{A}}_{k-1} + \underline{b}_k (\underline{z}_k' - \underline{h}_k' \hat{\underline{A}}_{k-1}) \tag{91}$$

Comparing equations (90), (88) and (91) with (60), (61) and (62) respectively, it is seen that the maximum-likelihood filtering equations are identical to the least-squares filtering equations except that the "1" in equation (60) has been replaced by the variance term s_k in equation (90). In other words the maximum-likelihood filter degenerates to a least-squares filter if $s_k = 1$.

The starting values for $\underline{P}_k$ and $\hat{\underline{A}}_k$ can be obtained by a direct application of (70) and (83) to the first s measurement stages, which would require inversion of an $s \times s$ matrix. However, since starting values constitute a-priori known statistics of the parameters it is instructive instead to compare the recursive maximum-likelihood filter with a similar filter that is based on such statistics. In the maximum-a-posteriori (MAP) filter, $\underline{A}$ has a normal or Gaussian probability distribution, $\hat{\underline{A}}_o$ is its expected value and $\underline{P}_o$ is a diagonal matrix such that the ith element on its main diagonal is the variance of every parameter in the ith row of $\underline{A}$. To obtain this filter it is necessary to maximize the so-called a-posteriori density $p_{\underline{A}|\underline{Z}}$ which is related to the likelihood density $p_{\underline{Z}|\underline{A}}$ by the Bayes rule:

$$p_{\underline{A}|\underline{Z}} = \frac{p_{\underline{Z}|\underline{A}}\, p_{\underline{A}}}{p_{\underline{Z}}} \tag{92}$$

This is equivalent to finding a maximum of its logarithm:

$$\log p_{\underline{A}|\underline{Z}} = \log p_{\underline{Z}|\underline{A}} + \log p_{\underline{A}} - \log p_{\underline{Z}} \tag{93}$$

Differentiating with respect to $\underline{A}$ results in

$$\frac{d}{d\underline{A}} \log p_{\underline{A}|\underline{Z}} = \underline{H}'\underline{S}^{-1}(\underline{Z} - \underline{H}\,\underline{A}) + \frac{d}{d\underline{A}} \log p_{\underline{A}} \tag{94}$$

where $\frac{d}{d\underline{A}} \log p_{\underline{Z}}$ is zero because $p_{\underline{Z}}$ is not a function of $\underline{A}$. The a-priori density $p_{\underline{A}}$ can be written

$$p_{\underline{A}} = \frac{1}{(2\pi)^{rs/2}(\det P_o)^{r/2}} \exp\left[-\frac{1}{2}\sum_{i,j} \underline{P}_{o_{ii}}^{-1} (\underline{A}_{ij} - \hat{\underline{A}}_{o_{ij}})^2\right] \tag{95}$$

and then

$$\frac{d}{d\underline{A}} \log p_{\underline{A}} = -\underline{P}_o^{-1}(\underline{A} - \hat{\underline{A}}_o) \tag{96}$$

The maximum a-posteriori density occurs when $\frac{d}{d\underline{A}} \log p_{\underline{A}|\underline{Z}}$ is zero:

$$\underline{H}'\underline{S}^{-1}(\underline{Z} - \underline{H}\,\hat{\underline{A}}) - \underline{P}_o^{-1}(\hat{\underline{A}} - \hat{\underline{A}}_o) = \underline{0}$$

$$\hat{\underline{A}} = \underline{P}(\underline{H}'\underline{S}^{-1}\underline{Z} + \underline{P}_o^{-1}\hat{\underline{A}}_o) \tag{97}$$

where

$$\underline{P} = (\underline{H}'\underline{S}^{-1}\underline{H} + \underline{P}_o^{-1})^{-1} \tag{98}$$

Comparison with equation (86) shows that the MAP estimate will in fact be generated by the recursive maximum-likelihood equations when $\hat{\underline{A}}_o$ and $\underline{P}_o$ are the expected value and variance, respectively, of the parameters. The resulting filter is actually a special case of the well-known discrete Kalman filter but has limited application possibilities because of the need for accurate a-priori statistics of the parameters and because of the restriction that parameters in the same row of $\underline{A}$ must have the same variance. However, the fact that the maximum-likelihood estimate of (70) and (83) is generated by the same recursive relations as the MAP estimate of (97) and (98) and that the MAP estimate degenerates to the maximum-likelihood estimate as $\underline{P}_o$ becomes infinite, indicates that the maximum-likelihood filter can be started with $\hat{\underline{A}}_o$ equal to zero and $\underline{P}_o$ a diagonal matrix with large elements on the main diagonal. In this way the recursive maximum-likelihood filter would presuppose $\underline{A}$ to have zero expected value and very large variance, which is consistent with a total lack of a-priori statistics.

IV. General Computational Algorithm

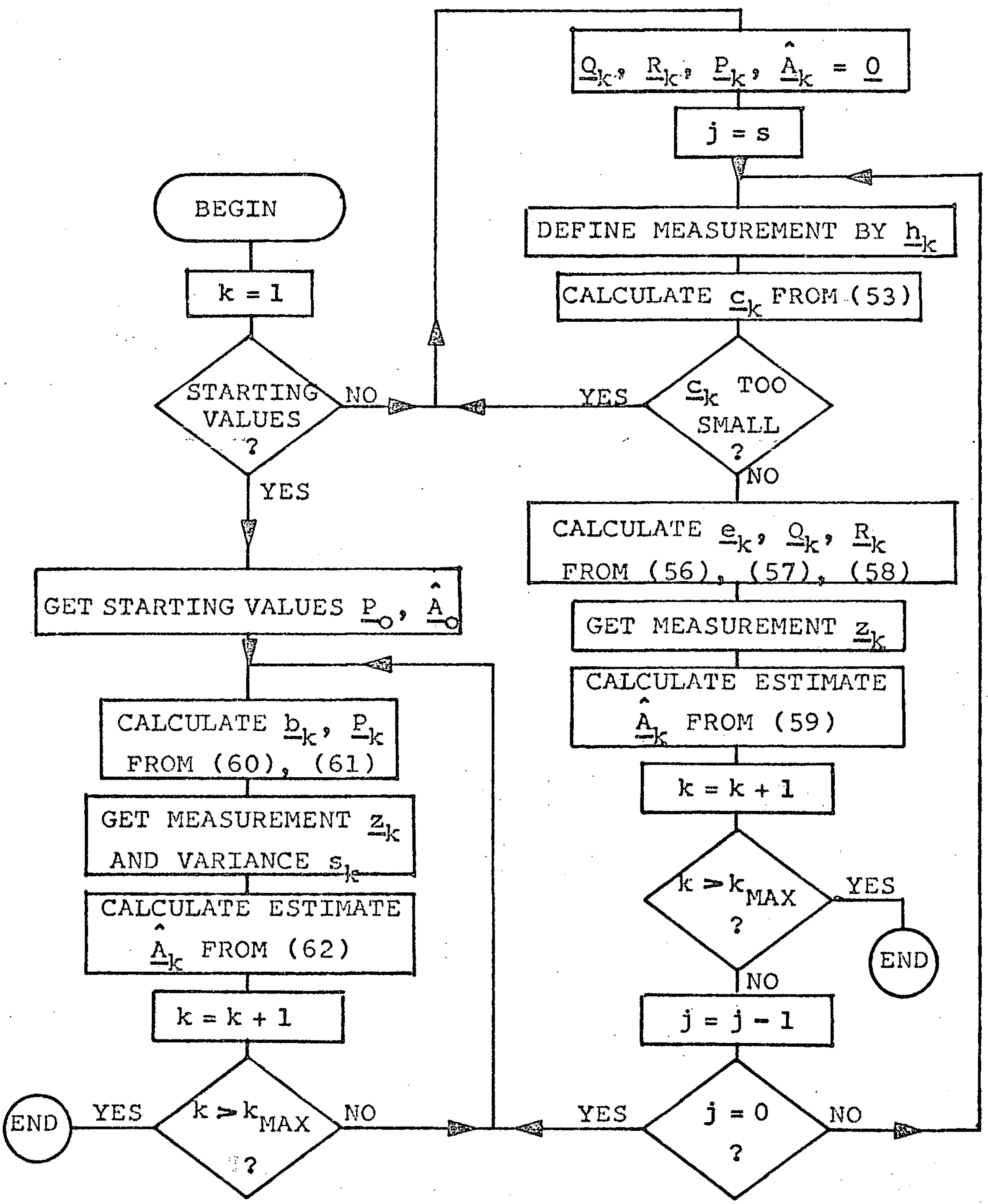
It is now apparent that with very minor alterations the basic least-squares recursive equations of Chapter II (53-62) can perform either maximum-likelihood or Bayesian (maximum-a-posteriori) filtering. By substituting the noise variance s_k in place of the "1" in equation (60) and replacing the minimum-norm equations (53-59) with initial values $\hat{\underline{A}}_o$ zero and $\underline{P}_o$ very large, a maximum-likelihood filter results. A Bayesian filter is produced by using the expected value and variance of $\underline{A}$ for $\hat{\underline{A}}_o$ and $\underline{P}_o$ respectively in the maximum-likelihood filter. The following table summarizes the differences:

TABLE I - Essential Differences of Least-Squares, Maximum-Likelihood and Bayesian MAP Filters

Filter	s_k	Initial Values
Least-squares	1	minimum-norm composition using equations (53-59)
Maximum-likelihood	noise variance	$\hat{\underline{A}}_o$ zero, $\underline{P}_o$ very large
Bayesian MAP	noise variance	$\hat{\underline{A}}_o$ = expected value of $\underline{A}$ $\underline{P}_o$ = variance of $\underline{A}$

On the next page is presented a general computational algorithm which allows for any of the combinations in the above table. It also allows for unclassified combinations such as one in which the noise variance term is "1" and the initial values are $\hat{\underline{A}}_o$ zero and $\underline{P}_o$ large. This is effectively

FIGURE 1 - General Computational Algorithm for Estimation



a least-squares filter which is begun in the same way as the maximum-likelihood filter, eliminating the need for the "minimum-norm" equations (53) and (56-59). Tests of this filter are described in example 1 of Chapter VI.

In the general algorithm, linear dependence of the rows of $\underline{H}$ in the first s stages of the least-squares filter results in a value of $\underline{c}_k$ which is near zero, re-initializing the entire process. How close to zero $\underline{c}_k$ must come in order for this to occur is a difficult matter to define and depends among other things on the precision of the calculations. While exact linear dependence would theoretically make $\underline{c}_k$ exactly zero, the value determined by the computer will normally contain errors due to truncation and thus be slightly different from zero. In any event, cases of near linear dependence can produce inaccurate estimates, so it is probably best to require that $\underline{c}_k$ remain reasonably large. This can be done by defining a threshold value and causing re-initialization if the magnitude of $\underline{c}_k$ falls less than this threshold during the first s stages.

In choosing which of the various filtering procedures to use it is important to know how the errors of the estimates are expected to compare. A useful matrix which gives an estimate of the error is the error covariance matrix, hereby defined as

$$\text{cov}\,(\hat{\underline{A}}) = \varepsilon(\hat{\underline{A}} - \underline{A})(\hat{\underline{A}} - \underline{A})' \qquad (99)$$

The trace of this matrix is equivalent to the expected value of the sum of the squares of the error matrix $(\underline{A} - \hat{\underline{A}})$.

Using equations (5) and (1) it can be seen that the error covariance of the least-squares filter after s stages is given by

$$\text{cov}\ (\hat{\underline{A}}_{LS}) = (\underline{H}'\underline{H})^{-1}\underline{H}'\ \varepsilon(\underline{V}\underline{V}')\underline{H}(\underline{H}'\underline{H})^{-1} \tag{100}$$

$\varepsilon(\underline{V}\underline{V}')$ can readily be shown to be a diagonal matrix such that the ith element on its main diagonal is the sum of the variances of the measurements in the ith row of $\underline{Z}$.

The least-squares estimate is always unbiased. That is,

$$\varepsilon(\underline{A} - \hat{\underline{A}}_{LS}) = \underline{0} \tag{101}$$

The maximum-likelihood estimate (70) has an error covariance given by

$$\text{cov}\ (\hat{\underline{A}}_{ML}) = (\underline{H}'\underline{S}^{-1}\underline{H})^{-1}\underline{H}'\underline{S}^{-1}\ \varepsilon(\underline{V}\underline{V}')\ \underline{S}^{-1}\underline{H}(\underline{H}'\underline{S}^{-1}\underline{H})^{-1} \tag{102}$$

Because the maximum-likelihood filter requires that measurements in the same row of $\underline{Z}$ have the same variance and since there are r measurements in each row, it is apparent that

$$\varepsilon(\underline{V}\underline{V}') = r \times \underline{S} \tag{103}$$

where $\underline{S}$ is the measurement-noise variance matrix as defined in (65). Therefore, when all measurements in the same row of

$\underline{Z}$ have the same variance, the error covariances of the least-squares and maximum-likelihood estimates become

$$\text{cov}\ (\hat{\underline{A}}_{LS}) = r \times (\underline{H}'\underline{H})^{-1}\ \underline{H}'\underline{S}\ \underline{H}(\underline{H}'\underline{H})^{-1} \tag{104}$$

$$\text{cov}\ (\hat{\underline{A}}_{ML}) = r \times (\underline{H}'\underline{S}^{-1}\ \underline{H}) \tag{105}$$

The definition of the $\underline{P}$-matrix for the maximum-likelihood filter as given in equation (83) shows that the error covariance of the maximum-likelihood filter is given simply by

$$\text{cov}\ (\hat{\underline{A}}_{ML}) = r \times \underline{P} \tag{106}$$

Using the matrix inequality

$$\underline{M}'\underline{M} \geqq (\underline{N}'\underline{M})'(\underline{N}'\underline{N})^{-1}(\underline{N}'\underline{M}) \tag{107}$$

(see Sage and Melsa [14], p. 246) where $\underline{M}$ and $\underline{N}$ are any two $k \times s$ matrices with $k \geqq s$ and $\underline{N}$ of rank s, and making the substitutions

$$\underline{M} = \underline{S}^{-\frac{1}{2}}\ \underline{H}(\underline{H}'\underline{H})^{-1} \tag{108}$$

$$\underline{N} = \underline{S}^{\frac{1}{2}}\ \underline{H} \tag{109}$$

it is easily shown that

$$\text{cov}\ (\hat{\underline{A}}_{ML}) \leqq \text{cov}\ (\hat{\underline{A}}_{LS}) \tag{110}$$

That is, the maximum-likelihood filter, when applicable, gives an estimate which is as good as or better than that of the least-squares filter.

Like the least-squares estimate, the maximum-likelihood estimate is unbiased:

$$\varepsilon(\underline{\hat{A}}_{ML} - \underline{A}) = \underline{0} \tag{111}$$

The error covariance of the Bayesian MAP estimate, as defined by equation (97), is

$$\text{cov}\,(\underline{\hat{A}}_{MAP}) = \underline{P}\,(\underline{H}'\underline{S}^{-1}\,\varepsilon(\underline{V}\,\underline{V}')\,\underline{S}^{-1}\,\underline{H} + \underline{P}_o^{-1}\,\varepsilon(\underline{A}\,\underline{A}')\,\underline{P}_o^{-1})\,\underline{P}' \tag{112}$$

Since all measurements in the same row of $\underline{Z}$ must have the same noise variance and the probability distributions of all parameters in the same row of $\underline{A}$ must have the same variance,

$$\varepsilon(\underline{V}\,\underline{V}') = r \times \underline{S} \tag{113}$$

$$\varepsilon(\underline{A}\,\underline{A}') = r \times \underline{P}_o \tag{114}$$

where r is the number of elements in each row of $\underline{Z}$ and $\underline{A}$. The error covariance therefore becomes

$$\text{cov}\,(\underline{\hat{A}}_{MAP}) = r \times \underline{P}\,(\underline{H}'\underline{S}^{-1}\,\underline{H} + \underline{P}_o^{-1})\,\underline{P} = r \times \underline{P} \tag{115}$$

which can be readily determined from the $\underline{P}$-matrix. Comparing the values of $\underline{P}$ for the maximum-likelihood and MAP estimates, it is obvious that the MAP estimate, where applicable, has an error covariance which is less than or equal to that of either the maximum-likelihood or least-squares estimates. In fact, the MAP estimate, when valid, is known to have the least error covariance of any known estimate. Even when the restriction is removed that the noise and parameters be Gaussian the MAP filter still provides the best estimate of all linear filters. The noise must still be random with zero-mean and known variance and the expected value and variance of the parameters must still be known. The filter is then usually called a linear-minimum-variance filter.

In addition to the fact that all parameters in the same row of $\underline{A}$ must have the same variance, the MAP estimate has another major disadvantage. If incorrect prior expected values and variances are used the estimates will be biased, with the bias at a stage k given by

$$\varepsilon(\hat{\underline{A}}_k - \underline{A}) = \underline{P}_k(\underline{H}_k'\underline{S}_k^{-1}\underline{H}_k\varepsilon(\underline{A}) + \underline{P}_o^{-1}\hat{\underline{A}}_o - \underline{P}_k^{-1}\varepsilon(\underline{A}))$$

$$= \underline{P}_k(\underline{P}_o^{-1}\hat{\underline{A}}_o - \underline{P}_o^{-1}\varepsilon(\underline{A}))$$

$$= (\underline{I} + \underline{H}_k'\underline{S}_k^{-1}\underline{H}_k\underline{P}_o)^{-1}(\hat{\underline{A}}_o - \varepsilon(\underline{A}))$$

$$= (\underline{I} + \underline{P}_o\sum_{i=1}^{k}\underline{h}_i s_i^{-1}\underline{h}_i')^{-1}(\hat{\underline{A}}_o - \varepsilon(\underline{A})) \qquad (116)$$

The bias is most noticeable for smaller values of k and decreases as k increases. It is also smaller for higher values of the initial variance $\underline{P}_o$ and approaches zero as $\underline{P}_o$ becomes infinite, the estimate then becoming a maximum-likelihood estimate.

In conclusion it may be said that among the three filters, least-squares, maximum-likelihood and Bayesian MAP, the more extensive the a-priori statistical knowledge of the parameters and measurement noise, the lower is the covariance of estimation error.

V. Identification of A Linear Stationary Process

The computational algorithm of the previous chapter can be used to estimate the parameters of a discrete model for a linear time-invariant process. If measurements of the system variables are available at uniformly-spaced intervals of time, it is possible to develop a model of the form

$$\underline{x}_k = \underline{\phi}\,\underline{x}_{k-1} + \underline{\Delta}\,\underline{u}_{k-1} \qquad (117)$$

where $\underline{x}_k$ is an n-dimensional vector composed of the system outputs at stage k, $\underline{u}_k$ is an m-dimensional vector composed of the system inputs at stage k and $\underline{\phi}$ and $\underline{\Delta}$ are matrices composed of the constant parameters describing the process. $\underline{x}_k$ is called the state of the system at stage k, $\underline{u}_k$ the control and $\underline{\phi}$ and $\underline{\Delta}$ the state-transition and state-driving matrices respectively. Transposing both sides of the last equation results in

$$\underline{x}_k' = \underline{x}_{k-1}'\,\underline{\phi}' + \underline{u}_{k-1}'\,\underline{\Delta}' \qquad (118)$$

Because of measurement noise, there will be differences between the observed values of the variables and their true values. Therefore it is convenient to distinguish the observed values with a superscribed bar as follows:

$$\bar{\underline{x}}_k = \underline{x}_k + \underline{\mu}_k \qquad (119)$$

$$\bar{\underline{u}}_k = \underline{u}_k + \underline{\omega}_k \qquad (120)$$

where $\underline{\mu}_k$ and $\underline{\omega}_k$ are vectors comprised of the noise terms. Combining these two equations with (118) yields the relation between the parameters and the observed values:

$$\underline{\bar{x}}_k' = \underline{\bar{x}}_{k-1}' \underline{\emptyset}' + \underline{\bar{u}}_{k-1}' \underline{\Delta}' + \underline{\mu}_k' - \underline{\mu}_{k-1}' \underline{\emptyset}' - \underline{\omega}_{k-1}' \underline{\Delta}'$$

or

$$\underline{\bar{x}}_k' = \begin{bmatrix} \underline{\bar{x}}_{k-1}' & \vdots & \underline{\bar{u}}_{k-1}' \end{bmatrix} \begin{bmatrix} \underline{\emptyset}' \\ \cdots \\ \underline{\Delta}' \end{bmatrix} + \underline{\mu}_k' - \underline{\mu}_{k-1}' \underline{\emptyset}' - \underline{\omega}_{k-1}' \underline{\Delta}' \qquad (121)$$

If the measured vectors $\underline{\bar{x}}_k'$, $k = 1, 2, 3, \ldots$ become the successive rows of the $\underline{Z}$ matrix in the computational algorithm:

$$\underline{z}_k' = \underline{\bar{x}}_k' \quad (m) \qquad (122)$$

and the corresponding prior measurements become the successive rows of the $\underline{H}$ matrix:

$$\underline{h}_k' = \begin{bmatrix} \underline{\bar{x}}_{k-1}' & \vdots & \underline{\bar{u}}_{k-1}' \end{bmatrix} \quad (n+m) \qquad (123)$$

then in accordance with the representation of equation (1) the unknown parameter will be

$$\underline{A} = \begin{bmatrix} \underline{\emptyset}' \\ \cdots \\ \underline{\Delta}' \end{bmatrix} \qquad (124)$$

and the sequential noise vectors will be

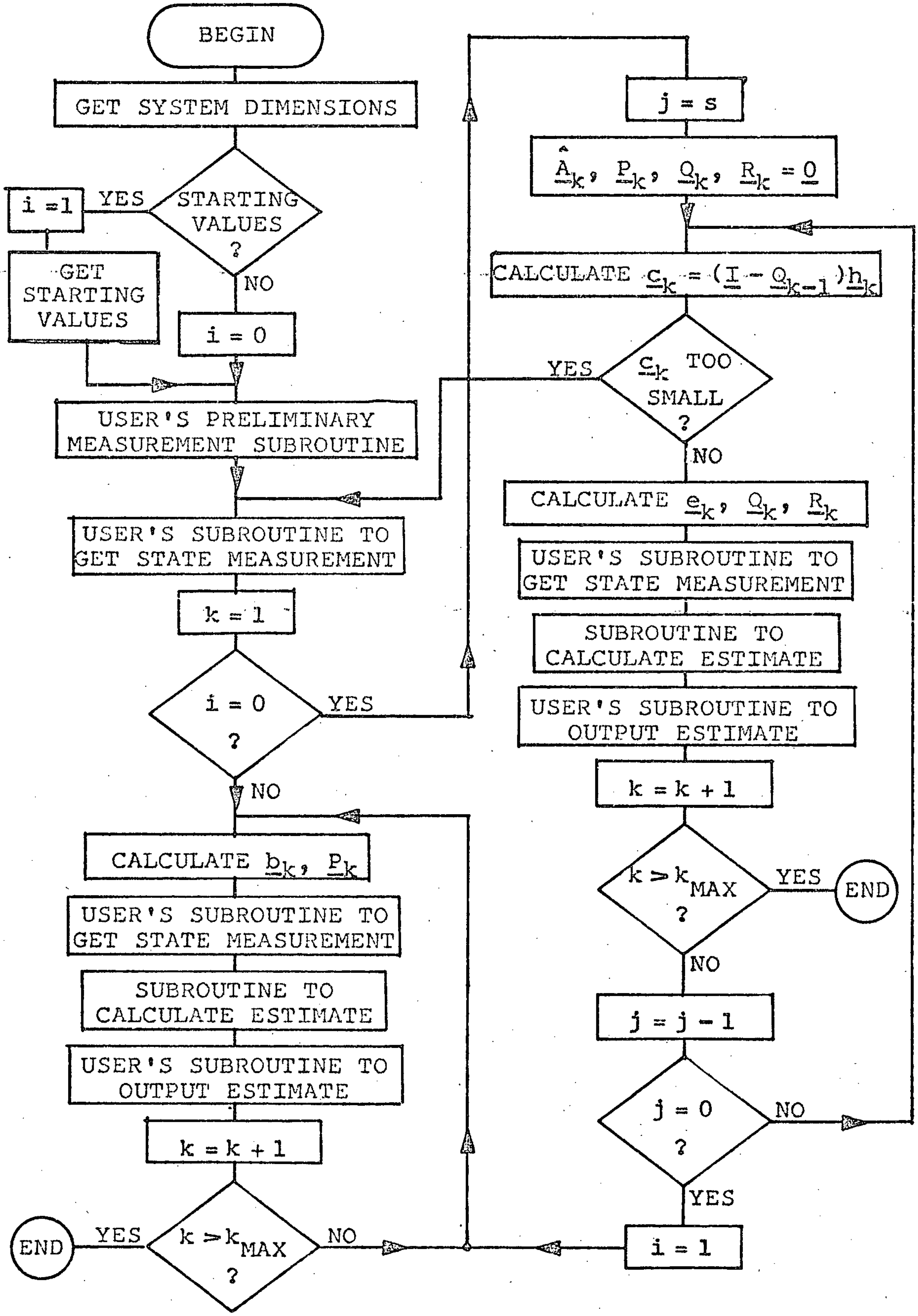
$$\underline{v}_k' = \underline{\mu}_k' - \underline{\mu}_{k-1}'\underline{\emptyset}' - \underline{\omega}_{k-1}'\underline{\Delta}' \qquad (125)$$

In other words, use of the $\underline{z}_k'$ and $\underline{h}_k'$ vectors as defined by (122) and (123) in the general computational algorithm of Chapter IV will produce an estimate of the matrix defined by (124).

If the components of the noise sequences $\underline{\mu}_k$ and $\underline{\omega}_k$ have zero means and are Gaussian, white and independent, then the least-squares filter of Chapter II applies because the overall observation noise $\underline{v}_k$ defined by (125) has zero-mean components. The maximum-likelihood and Bayesian filters, however, are not strictly valid as they have been derived in Chapter III, because the components of $\underline{v}_k$ are not likely to be independent or white. None the less it would seem logical that the hierarchy among the three filters should still exist because of the differing degrees of a-priori information utilized. Thus, although methods exist by which the maximum-likelihood and Bayesian filters can be made optimal (see, for example, Sage and Melsa [14], Chapter 8), they involve such extensive complication of the algorithm that it is convenient in this application to merely ignore the fact that the noise components may be non-white or statistically dependent.

The computational algorithm as applied to the system-identification problem is represented in the flow-chart on the following page. All of the experimental tests described in the next chapter were made using this algorithm on either the

FIGURE 2 - System Identification Algorithm



I.B.M. 360-67 or the Data General Corporation "Nova" digital computer. In the appendix are included the complete programs for the Nova version of the algorithm. Comparison with the flow-chart of Chapter IV will show that the identification algorithm is basically the same except for the addition of certain specialized subroutines for handling the input and output data. These data subroutines can be changed to suit any particular application. There is a preliminary measurement subroutine which is provided in case there are any tasks associated with the measurement process which must be performed before entering the identification cycle. For example, should it be necessary to take samples of the system at a rate faster than the computation cycle would allow, the measurements may all be made in advance and stored by this subroutine. Then on each cycle is a subroutine to get the state measurement from the appropriate source and another to output the calculated estimate.

The algorithm was programmed on the system-360 to allow for more sophisticated analysis using artificial models of known statistics.

In the Nova programs, all the initializing procedures are controlled by the operator using the teletype keyboard in a conversational manner. The system dimensions can be set to estimate any matrix $\underline{A}$ up to a dimension of 8 x 8 and the calculations are performed in floating-point arithmetic that is based on a 24-bit mantissa with sign and 7-bit exponent using the standard basic floating-point software provided with the computer.

Although in the flow-chart the k-counters are separate from the subroutines, in the Nova programs the job of counting stages has been left to the user-supplied subroutines. This allows for counting either at the point where data comes in or at the point of outputting the estimate, whichever is more suitable to the particular application. Also left to the user subroutines is the task of determining the sampling times, which, in the case where data is obtained from an actual system using an analogue-to-digital converter, could require an external real-time clock connected to the input/output bus of the computer.

The programs are thus very versatile, with the permanent software performing identification only, and the user-supplied subroutines having full control over the rest of the process.

VI. Examples

All the examples described in this chapter were used to test the computer programs for the system-identification algorithm and to study various properties of the algorithm. Sequential values of the state were generated for the algorithm in each case using known values of $\underline{\phi}$ and $\underline{\Delta}$ and the estimates were then compared with these known values.

Example 1:

The model parameters were

$$\underline{\phi} = \begin{bmatrix} 0.995 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.5 \\ 0.0 & -1.13 & 0.9 \end{bmatrix} \qquad \underline{\Delta} = \begin{bmatrix} 0.0 \\ 0.0 \\ 1.25 \end{bmatrix}$$

with initial state and control

$$\underline{x}_o = \begin{bmatrix} 0.0 \\ 1.5 \\ 3.95 \end{bmatrix} \qquad u = 1$$

The control was left constant throughout the process, resulting in an open-loop response corresponding to the three curves of Figure 3. The curves are shown to be continuous because it has been assumed that in practice the measurements would result from uniform sampling of this continuous system.

The simulation was performed using the Nova programs and there was no measurement noise added to the model. Table II

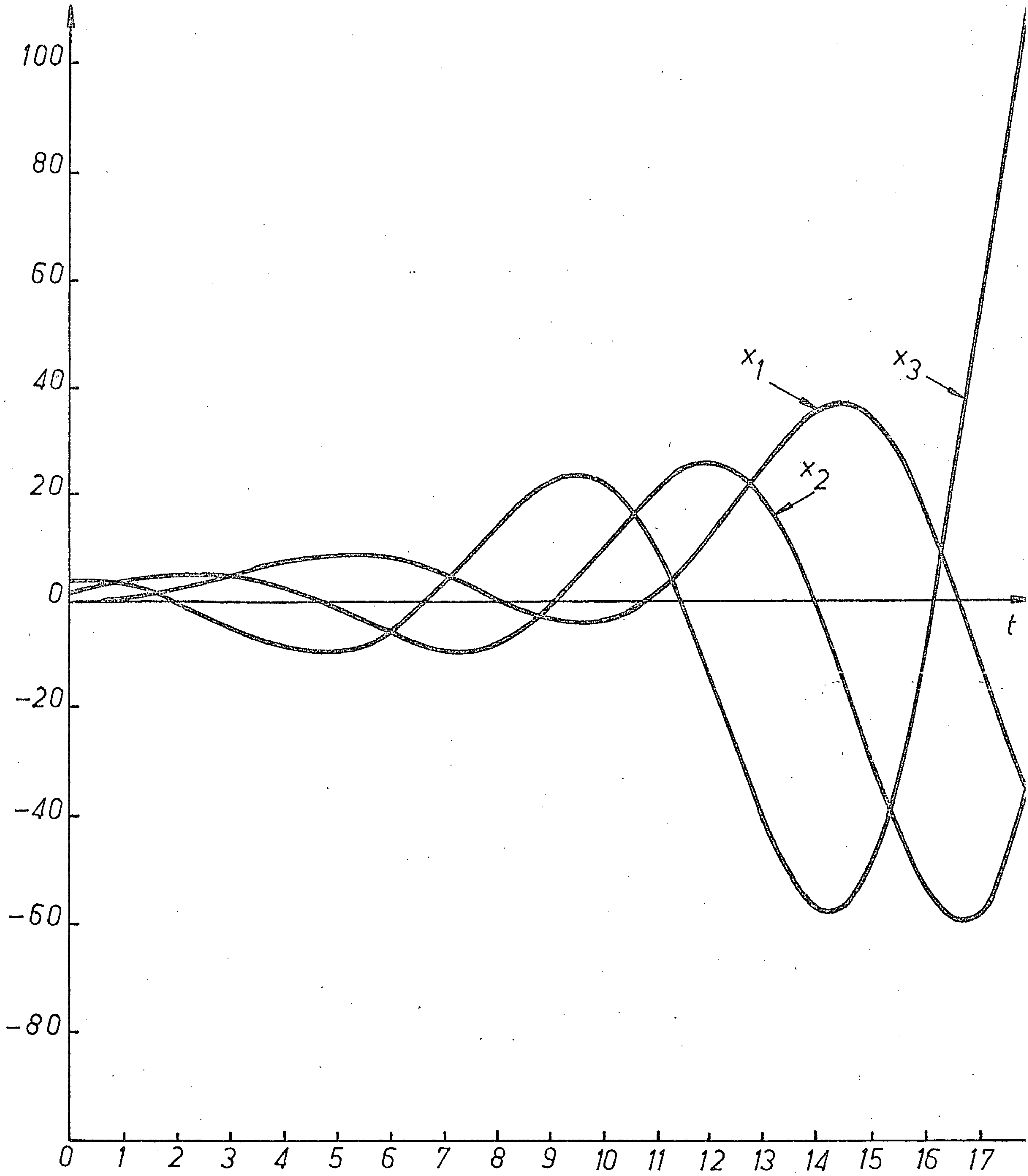


Figure 3 - Time response of continuous system corresponding to Example 1.

TABLE II - Stage-wise Errors of Least-Squares Filter (Example 1)

Figures in each column represent errors at successive stages.

Stage	"minimum norm"	$\underline{P}_0 = 10 \times \underline{I}$	$\underline{P}_0 = 10^2 \times \underline{I}$	$\underline{P}_0 = 10^3 \times \underline{I}$	$\underline{P}_0 = 10^5 \times \underline{I}$	$\underline{P}_0 = 10^7 \times \underline{I}$	$\underline{P}_0 = 10^{16} \times \underline{I}$
1	+.4956009E+01	+.4956044E+01	+.4956009E+01	+.4956009E+01	+.4956011E+01	+.4956009E+01	+.4956009E+01
2	+.2193356E+01	+.2197610E+01	+.2193400E+01	+.2193356E+01	+.2193354E+01	+.2193356E+01	+.2193356E+01
3	+.7292265E+00	+.1245663E+01	+.7476390E+00	+.7294487E+00	+.7292240E+00	+.7292260E+00	+.7292259E+00
4	+.3315584E-08	+.8763182E+00	+.7286461E+00	+.2720855E+00	+.2375113E-03	+.6260436E-05	+.8251297E-05
5	+.9723287E-08	+.8960244E+00	+.4370673E+00	+.2738907E-01	+.8037871E-05	+.3811357E-06	+.2695410E-06
6	+.1147506E-07	+.8040013E+00	+.1357909E+00	+.2873371E-02	+.8270537E-06	+.6168898E-07	+.1110659E-07
7	+.1032982E-08	+.5403006E+00	+.3114320E-01	+.4229341E-03	+.1521597E-06	+.1694626E-07	+.1082285E-07
8	+.6331324E-08	+.2660201E+00	+.7173627E-02	+.8230054E-04	+.4096006E-07	+.2481867E-08	+.3086397E-07
9	+.1174260E-07	+.1069880E+00	+.1860306E-02	+.1993976E-04	+.1178606E-07	+.1211445E-09	+.2235941E-08
10	+.1237370E-07	+.4123187E-01	+.5695862E-03	+.5937541E-05	+.3443537E-08	+.3527151E-09	+.2407231E-08
11	+.1113249E-07	+.1753404E-01	+.2159944E-03	+.2214477E-05	+.5014855E-09	+.6378680E-09	+.4007216E-08
12	+.8576017E-08	+.9148615E-02	+.1058297E-03	+.1067259E-05	+.2560867E-10	+.1017675E-08	+.2451576E-08
13	+.5819554E-08	+.5936951E-02	+.6645077E-04	+.6555694E-06	+.9559106E-09	+.1640657E-08	+.2018661E-08
14	+.4588361E-08	+.4574932E-02	+.5036433E-04	+.4924871E-06	+.3086121E-08	+.2250708E-08	+.1039153E-08
15	+.4319357E-08	+.3887912E-02	+.4249783E-04	+.4198029E-06	+.1098628E-07	+.1374427E-07	+.1718590E-09
16	+.4324768E-08	+.3433026E-02	+.3737925E-04	+.3787415E-06	+.1721376E-09	+.2004652E-08	+.4943047E-08
17	+.4286413E-08	+.3033789E-02	+.3298429E-04	+.3446833E-06	+.3334956E-08	+.2594733E-08	+.2070682E-06
18	+.4066631E-08	+.2629245E-02	+.2855703E-04	+.3090746E-06	+.7399028E-08	+.3726839E-08	+.5057110E-08
19	+.3959783E-08	+.2224612E-02	+.2403805E-04	+.2663924E-06	+.3632550E-07	+.1630333E-07	+.1006140E-07

shows the actual estimation error as defined by

$$\text{Error }(\hat{\underline{A}}_k) = \text{trace }(\underline{A} - \hat{\underline{A}}_k)(\underline{A} - \hat{\underline{A}}_k)' \qquad (126)$$

computed on the Nova at sequential sampling times for the identification algorithm when used as a least-squares filter with a "minimum norm" composition at the start and also when started with $\hat{\underline{A}}_o = \underline{0}$ and $\underline{P}_o$ equal to various scalar multiples of the identity matrix $\underline{I}$. It can be seen that when $\underline{P}_o$ is fairly large, estimates can be obtained which are as good as, and at some stages marginally better than those obtained when the "minimum norm" procedure is used. Here the filtering problem is a deterministic one because there is no a priori information and the estimates should be based solely on the noise-free observations. $\underline{P}_o$ must therefore be made large to give minimal weighting of the initial estimates $\hat{\underline{A}}_o$. The resulting filter is then a good approximation to the purely deterministic least-squares filter of Chapter II, with much less computational requirements. However, the pure least-squares filter is subject to minimum initial bias and with it a better estimate results after fewer measurements. Specifically, the estimation error at the fourth stage in this example was lowest with the pure least-squares filter, the estimates being

$$\hat{\underline{\phi}}_4 = \begin{bmatrix} .9949869 & .5000086 & -.0000036 \\ .0000224 & .9999967 & .5000123 \\ -.0000045 & -1.130009 & .8999898 \end{bmatrix} \qquad \hat{\underline{\Delta}}_4 = \begin{bmatrix} .0000246 \\ -.0000396 \\ 1.249999 \end{bmatrix}$$

Example 2:

The model was

$$\underline{\phi} = \begin{bmatrix} 0.995 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.5 \\ 0.0 & -1.13 & -0.9 \end{bmatrix} \qquad \underline{\Delta} = \begin{bmatrix} 0.0 \\ 0.0 \\ 1.25 \end{bmatrix}$$

with initial state and initial control

$$\underline{x}_o = \begin{bmatrix} 0.0 \\ 1.5 \\ -1.45 \end{bmatrix} \qquad u_o = 1$$

and no simulated measurement noise.

In generating the remaining states, the control was left equal to u_o until stage 1, after which each state was determined using a control chosen to minimize the estimated simple performance function

$$\hat{J}_{k+1} = \hat{\underline{x}}'_{k+1} \underline{W}_1 \hat{\underline{x}}_{k+1} + \underline{u}'_k \underline{W}_2 \underline{u}_k$$

where $\underline{W}_1$ and $\underline{W}_2$ are weighting matrices chosen for stability purposes and $\hat{\underline{x}}_{k+1}$ is defined by the equation

$$\hat{\underline{x}}_{k+1} = \hat{\underline{\phi}}_k \underline{x}_k + \hat{\underline{\Delta}}_k \underline{u}_k$$

In this example it is possible to have

$$\underline{W}_1 = \underline{I} \quad \text{and} \quad \underline{W}_2 = 1$$

Setting the derivative of $\hat{J}_{k+1}$ with respect to u_k equal to zero gives

$$\underline{\hat{\Delta}}_k' \underline{\hat{x}}_{k+1} + u_k = 0$$

$$u_k = -(\underline{\hat{\Delta}}_k' \underline{\hat{\Delta}}_k + 1)^{-1} \underline{\hat{\Delta}}_k' \underline{\hat{\phi}}_k \underline{x}_k$$

Figure 4 shows the variation of the resulting performance function

$$J(t) = \underline{x}_k' \underline{x}_k + u_{k-1}^2 \quad , \quad t_k \leqq t < t_{k+1}$$

for the open-loop case where the control was left equal to u_o for all stages and for two cases where u_k was calculated, beginning with u_2, based on estimates from the least-squares filter using different starting procedures. All calculations were performed on the Nova.

This method might be useful for simple combined identification and control of an actual continuous system by calculating a sub-optimal control based on the discrete model.

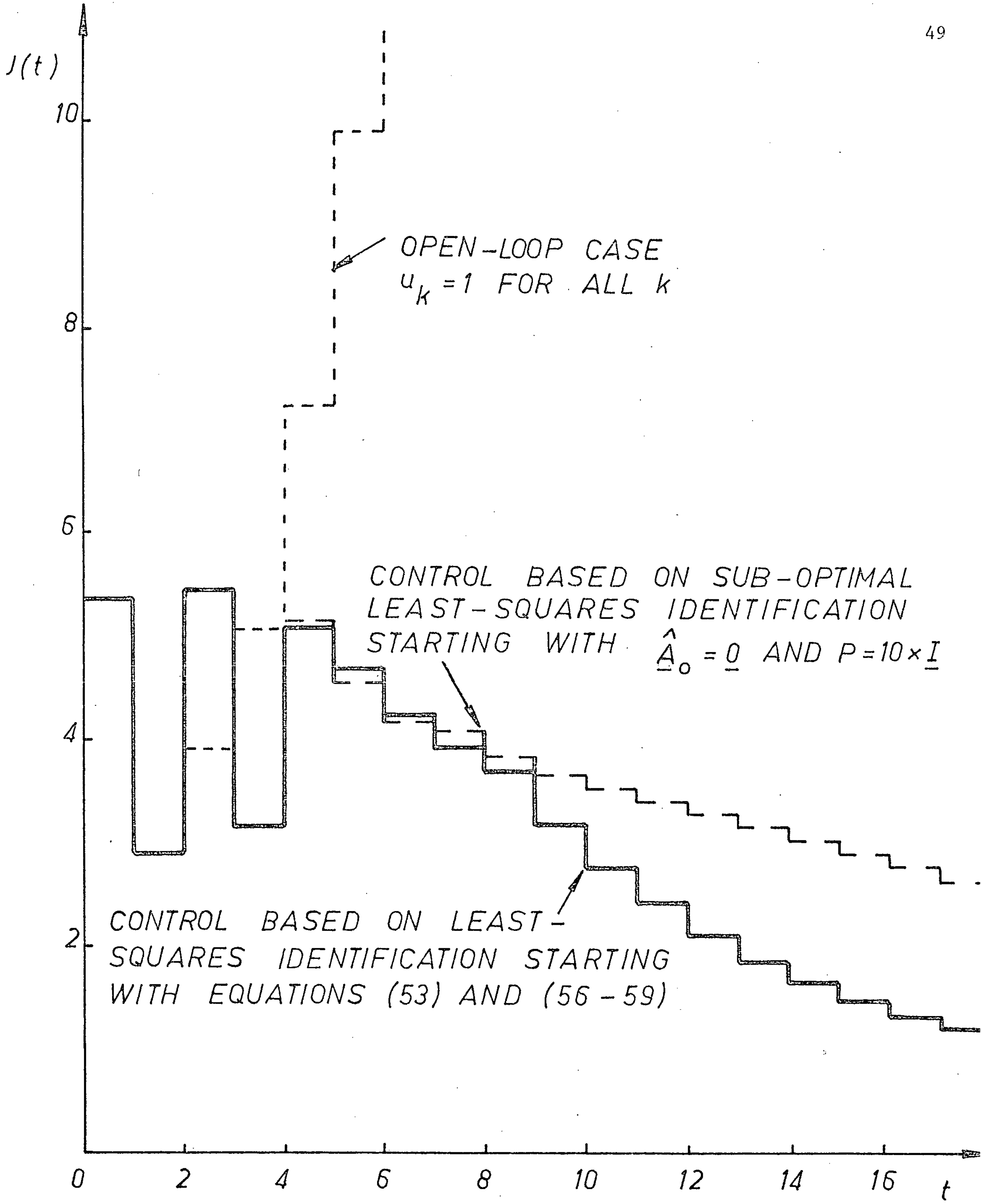


Figure 4 - Plot of performance functions for example 2.

$$J(t) = \underline{x}_k' \underline{x}_k + u_{k-1}^2, \qquad t_k \leq t < t_{k+1}$$

Example 3:

The model was

$$\underline{\emptyset} = \begin{bmatrix} 0.0 & 0.0 & 0.9 \\ 2.0 & 0.0 & 0.0 \\ 0.0 & 0.7 & 0.0 \end{bmatrix} \qquad \underline{\Delta} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

with initial state and control

$$\underline{x}_o = \begin{bmatrix} 2.7 \\ 10.0 \\ 4.9 \end{bmatrix} \qquad u = 0.0$$

The simulation was performed on the system-360 with random noise of normal distribution added to the state and control measurements. The standard deviation of the noise was 0.7, the variance 0.49.

The identification algorithm was used as a "best case" of the Bayesian MAP filter, with $\hat{\underline{\emptyset}}_o = \underline{\emptyset}$ and $\hat{\underline{\Delta}}_o = \underline{\Delta}$. s_k at every stage was set equal to the noise variance, 0.49. While in example 1 the initial estimate was inaccurate and the measurements were exact, in this example the initial estimates are exact and the measurements are noisy. Figure 5 shows the computed estimation errors as defined by (126) at each stage. As expected, the results are opposite to those of example 1, with a lower $\underline{P}_o$ now giving the better estimates because of increased weighting of $\hat{\underline{A}}_o$.

The results of Figure 6 were obtained with this same example and show the effect on the estimation error of using

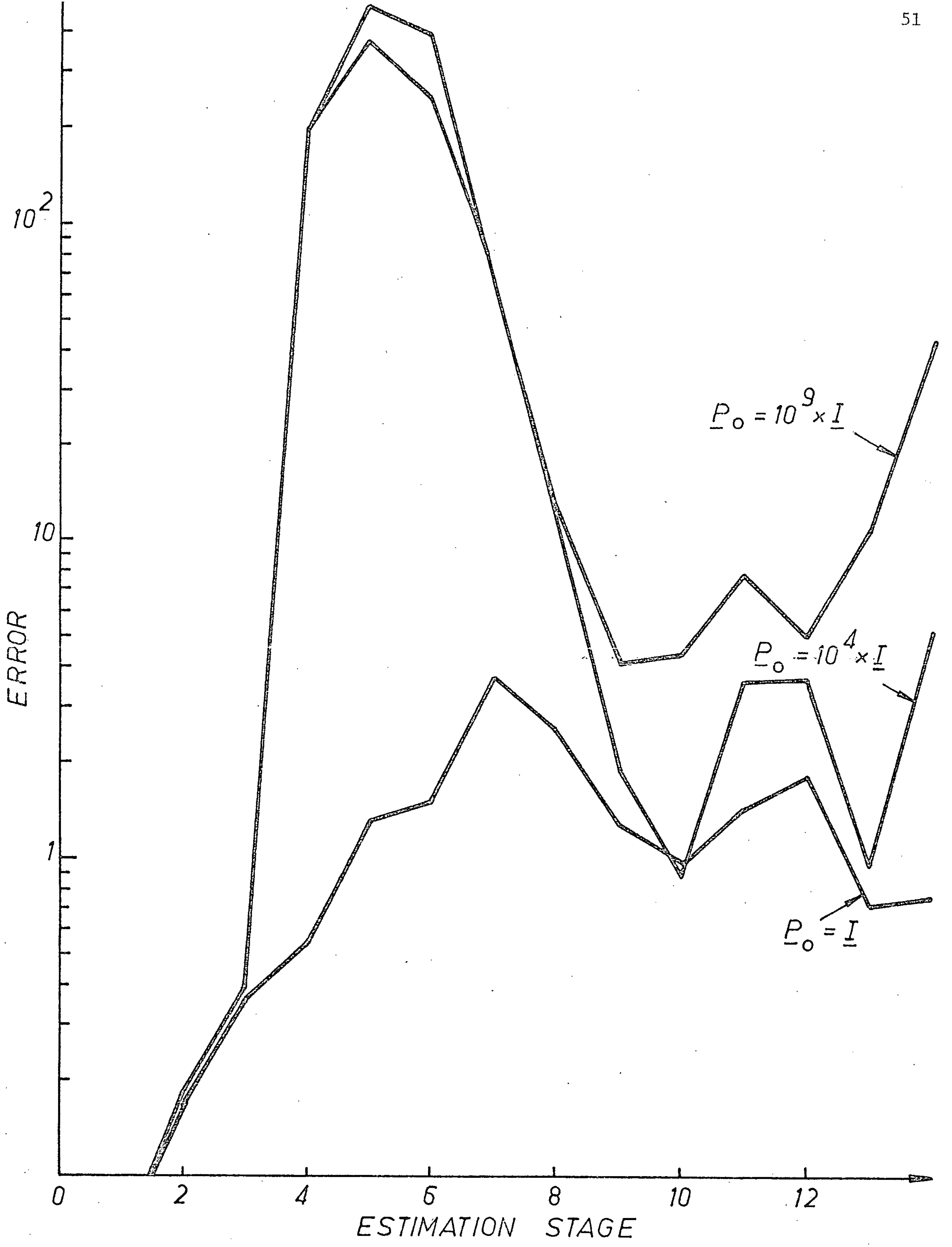


Figure 5 - Plot of estimation errors for example 3. $\hat{\underline{A}}_o = \underline{A}$, s_k = variance.

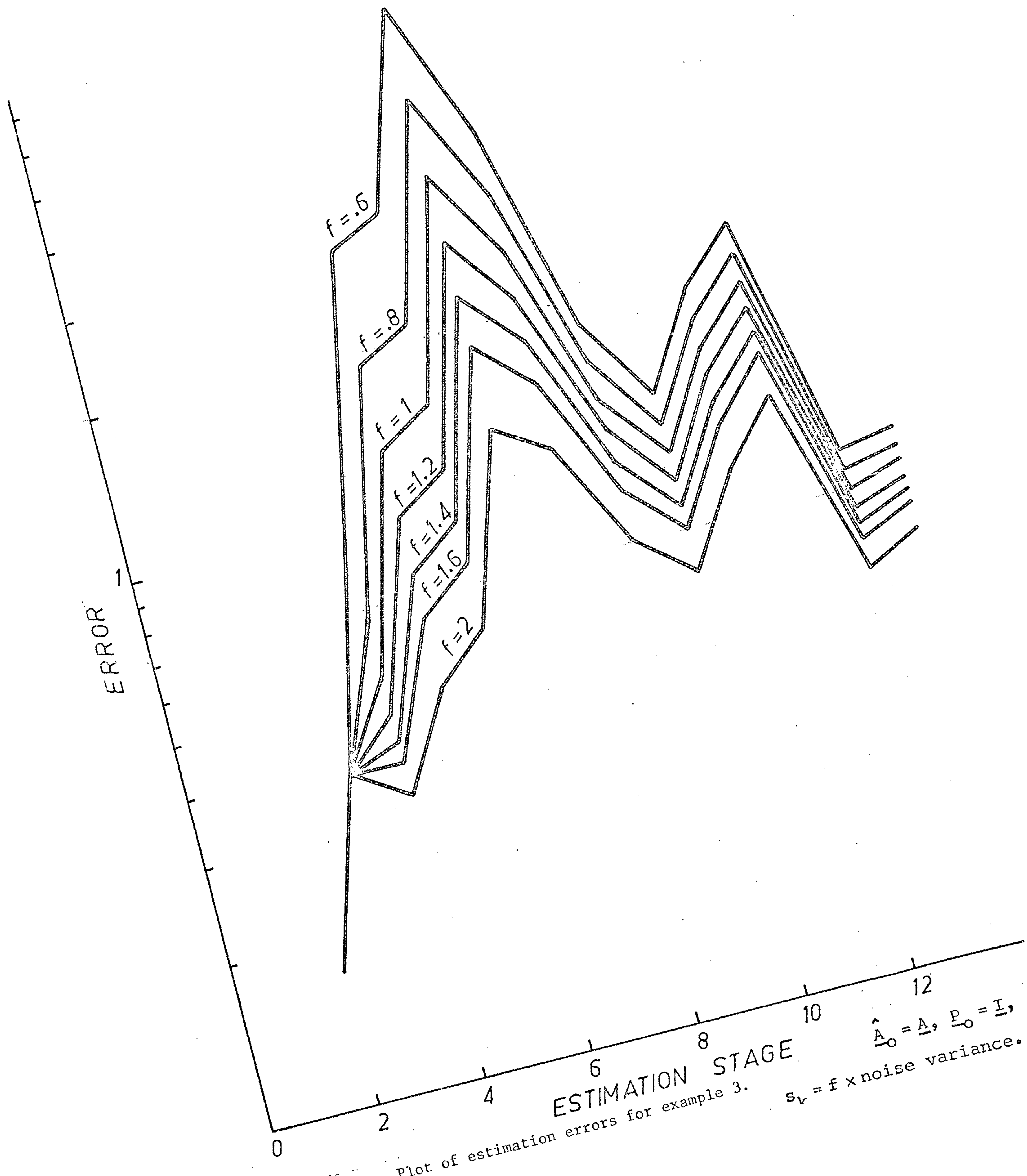


Figure 6 - Plot of estimation errors for example 3. $\hat{\underline{A}}_o = \underline{A}$, $\underline{P}_o = \underline{I}$, $s_\nu = f \times$ noise variance.

different multiples of the noise variance for s_k in the algorithm. When $\underline{P}_o$ is large the effect is not noticeable but when $\underline{P}_o$ is small, increasingly higher multiples give increasingly better estimates in the stages following stage 4. A higher value of s_k provides decreased weighting of the noisy measurements and increased weighting of the good initial estimate. s_k has no noticeable effect on the estimates prior to stage 4.

Figure 7 was obtained by the same procedure, except that the minimum-norm composition was used at the start. As is the case when the filter is started with $\underline{P}_o$ large, there was negligible difference of the errors when values of s_k ranging from 0.5 to 2 times the noise variance were used.

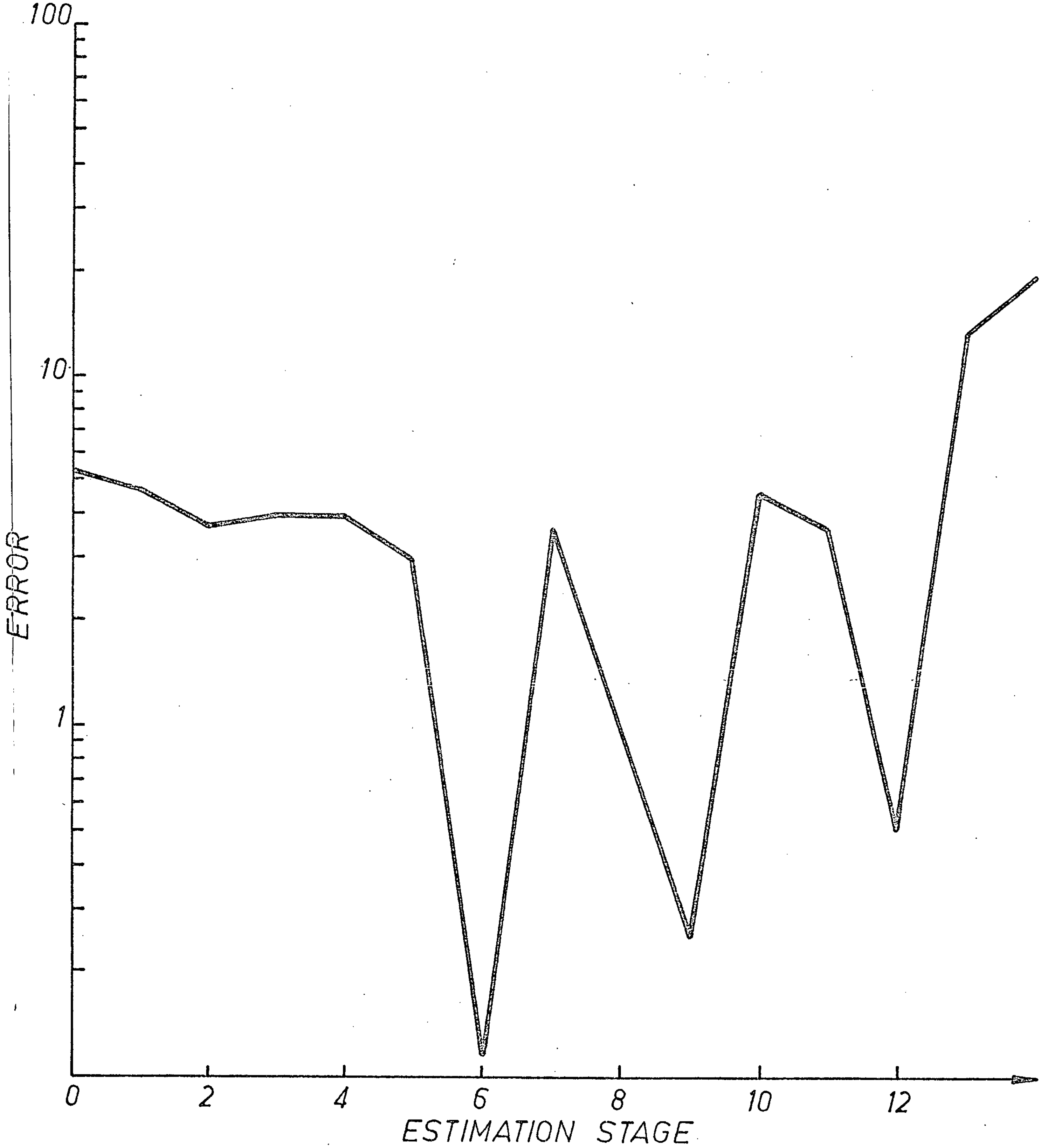


Figure 7 - Plot of estimation errors for example 3. Minimum-norm start.

Example 4:

The model was

$$\underline{\emptyset} = \begin{bmatrix} 0.995 & 0.5 & 0.0 \\ 0.0 & 1.0 & 0.5 \\ 0.0 & -1.13 & 0.9 \end{bmatrix} \qquad \underline{\Delta} = \begin{bmatrix} 0.0 \\ 0.0 \\ 1.25 \end{bmatrix}$$

with initial state and control

$$\underline{x}_o = \begin{bmatrix} 0.0 \\ 1.5 \\ 3.95 \end{bmatrix} \qquad u = 1$$

(see Figure 3 for response curves). The simulation was done on the system-360, introducing Gaussian noise of standard deviation 0.5 and variance 0.25 to the measurements. s_k in the algorithm was set equal to the variance at each stage.

Figure 8 shows the computed estimation errors as defined by (126) at each stage for three different starting procedures: $\hat{\underline{\emptyset}}_o = \underline{\emptyset}$, $\hat{\underline{\Delta}}_o = \underline{\Delta}$, $\underline{P}_o = \underline{I}$ for a "best-case" Bayesian MAP filter; a "minimum-norm" composition for a least-squares or maximum-likelihood filter; $\hat{\underline{\emptyset}}_o = \underline{0}$, $\hat{\underline{\Delta}}_o = \underline{0}$, $\underline{P}_o = 10^6 \times \underline{I}$ for an approximate least-squares or maximum-likelihood filter.

The results still support the hierarchy of filters developed in Chapter IV despite the fact that the overall noise terms defined by (125) are not expected to be statistically independent or white as discussed in Chapter V.

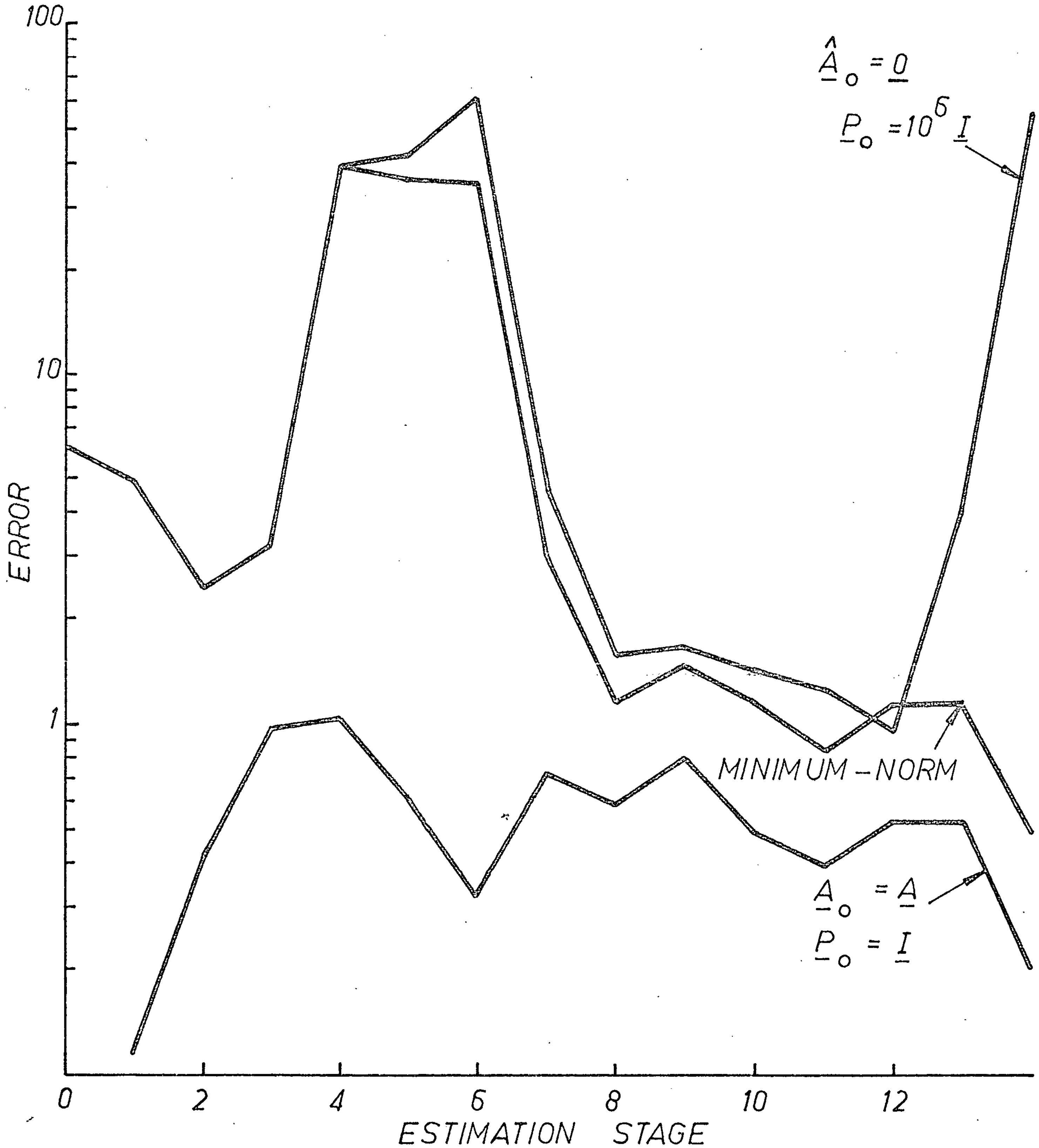


Figure 8 - Plot of errors for estimates in example 4.

VII. Further Applications

What has been presented in this thesis is an algorithm to estimate the parameter matrix $\underline{A}$ in the general measurement process of equation (1):

$$\underline{Z} = \underline{H}\,\underline{A} + \underline{V} \tag{1}$$

While the accompanying computer programs (see Appendix) have been written for the particular system-identification problem of Chapter V, it is a simple matter to adapt them for any measurement process defined by (1). Specifically, the identification problem of Chapter V requires that each row of $\underline{Z}$ ($\underline{z}_k'$) should be taken from the state measurement which will comprise the next row of $\underline{H}$ ($\underline{h}_{k+1}'$), and thus for the computer programs the vectors $\underline{z}$ and $\underline{h}$ can share the same storage locations. In other applications, separate sets of storage locations may be required. Apart from this, the program, when supplied sequentially (via the user's measurement subroutine) with the rows of $\underline{Z}$ and $\underline{H}$ arrays satisfying the relation of equation (1), will generate a sequential estimate of the parameter array $\underline{A}$, subject to the following conditions developed in the previous chapters:

The elements of $\underline{V}$ must have zero expected values. If all elements of the same row of $\underline{V}$ ($\underline{v}_k'$) have the same probability distribution and the variance of their distribution is known, it should be supplied for the value of s_k corresponding to that row (maximum-likelihood filter). If all elements of the same row of $\underline{V}$ do not have the same probability distribution or if the variances of the distributions are not known, s_k should be set equal to 1 at every stage (least-squares filter).

If expected values for the parameters are known then they should be used as the elements of $\hat{\underline{A}}_o$. If in addition the probability distributions of all parameters in the same row of $\underline{A}$ have the same variance and the variances of the distributions of all the parameters are known, then $\underline{P}_o$ should be a diagonal matrix with the ith element on its main diagonal equal to the variance of the distribution of every parameter in the ith row of $\underline{A}$. Otherwise $\underline{P}_o$ should be a diagonal matrix with each element on its main diagonal set large enough to allow for any uncertainty in the corresponding row of $\hat{\underline{A}}_o$.

If expected values for the parameters are not known then no initial values should be supplied for $\hat{\underline{A}}_k$ and $\underline{P}_k$, but equations (53) and (56 - 59) should be used at the start. However, where the increased computational time required by these equations would be prohibitive, a good approximation can be achieved by using initial values $\hat{\underline{A}}_o = \underline{0}$ and $\underline{P}_o$ diagonal with large elements on the main diagonal.

Rapid Identification

A major difficulty with the method of system-identification developed in Chapter V is that the estimated discrete model cannot be accurate unless the rate of sampling the state is high in relation to the rate at which the state varies. At the same time, such rapid sampling can lead to near linear-dependence in the state measurements and consequent ill-conditioning of the $\underline{H}$ matrix, which makes adequate identification impossible. Hanafy and Bohn [4] have suggested augmenting the state measurement at each sampling time with the measured outputs of integrators cascaded to the inputs and outputs of the continuous system. It is claimed that this additional data is effective in overcoming the problem of ill-conditioning. However, the usual treatment becomes cumbersome because the data and parameters must be structured into lengthy vectors in order to fit the form of conventional estimators, which are derived for a measurement process of the type

$$\underline{z} = \underline{H}\,\underline{a} + \underline{v}$$

$\underline{z}$ being the data vector, $\underline{a}$ the parameter vector, $\underline{H}$ the measurement matrix and $\underline{v}$ a vector of noise terms. For the identification problem this results in a large $\underline{H}$ matrix of block-diagonal form and containing many zeros.

The algorithm of this thesis can be used quite readily for identification with augmented state measurements. At each sampling time, the inputs and outputs of the system are measured and stored, along with the outputs of the successive integrators. A state measurement is processed as one row in the estimation algorithm, followed by the integrator outputs as subsequent rows. Suppose, for example, that each of the system inputs and outputs is passed through two integrators. Evidently the integrals

$$\int_0^t \underline{x}'(t)\,dt \quad \text{and} \quad \int_0^t \left[\int_0^t \underline{x}'(t)\,dt\right] dt$$

will satisfy the same linear differential equation as does $\underline{x}'(t)$, so uniform samples of their outputs should satisfy the same difference equation:

$$\underline{x}'(t_k) = \left[\underline{x}'(t_{k-1}) \;\vdots\; \underline{u}'(t_{k-1})\right] \underline{A}$$

$$\int_0^{t_k} \underline{x}'(t)\,dt = \left[\int_0^{t_{k-1}} \underline{x}'(t)\,dt \;\vdots\; \int_0^{t_{k-1}} \underline{u}'(t)\,dt\right] \underline{A}$$

$$\int_0^{t_k} \left[\int_0^t \underline{x}'(t)\,dt\right] dt = \left[\int_0^{t_{k-1}} \left[\int_0^t \underline{x}'(t)\,dt\right] dt \;\vdots\; \int_0^{t_{k-1}} \left[\int_0^t \underline{u}'(t)\,dt\right] dt\right] \underline{A}$$

The beginning rows of data for the estimation algorithm would therefore be

$$\underline{z}_1' = \overline{\underline{x}'(t_1)} \qquad \underline{h}_1' = \left[\overline{\underline{x}'(t_0)} \;\vdots\; \overline{\underline{u}'(t_0)}\right]$$

$$\underline{z}_2' = \overline{\int_0^{t_1} \underline{x}'(t)\,dt} \qquad \underline{h}_2' = \left[\overline{\int_0^{t_0} \underline{x}'(t)\,dt} \;\vdots\; \overline{\int_0^{t_0} \underline{u}'(t)\,dt} \right]$$

$$\underline{z}_3' = \overline{\int_0^{t_1} \left[\int_0^{t} \underline{x}'(t)\,dt \right] dt} \quad \underline{h}_3' = \left[\overline{\int_0^{t_0} \left[\int_0^{t} \underline{x}'(t)\,dt \right] dt} \;\vdots\; \overline{\int_0^{t_0} \left[\int_0^{t} \underline{u}'(t)\,dt \right] dt} \right]$$

$$\underline{z}_4' = \overline{\underline{x}'(t_2)} \qquad \underline{h}_4' = \left[\overline{\underline{x}'(t_1)} \;\vdots\; \overline{\underline{u}'(t_1)} \right]$$

where the superscribed bars indicate that these are the observed values of the variables concerned. With this procedure the state measurement defining $\underline{z}$ at a given stage does not immediately become the $\underline{h}$ vector for the following stage as it does in the simple identification problem. Therefore separate sets of memory locations are needed for the $\underline{z}$ and $\underline{h}$ vectors, as mentioned at the beginning of this chapter.

Identification of Non-Linear Systems

Both the identification methods discussed thus far have assumed a linear model for the system being measured. However, it is equally possible, within the allowable forms of measurement processes, to assume certain non-linear models. Netravali and de Figueiredo [9] have discussed methods of obtaining regression functions for classes of discrete non-linear systems in which the evolution operators can be represented by algebraic or trigonometric polynomials. Although noise considerations are more involved, the computational requirements are not unlike those of the linear identification problem and are adaptable to the computer programs contained in this thesis. As a very simple example, suppose it is desired to estimate a third-order non-linear algebraic model of the form

$$x_{k+1} = a_0 + a_1 x_k + a_2 x_k^2 + a_3 x_k^3$$

from measurements of the scalar variable x_k, $k = 0,1,2,\ldots$. The estimation algorithm would begin with the following data:

$$\underline{z}_1' = \bar{x}_1 \qquad \underline{h}_1' = \begin{bmatrix} 1 & \bar{x}_0 & \bar{x}_0^2 & \bar{x}_0^3 \end{bmatrix}$$

$$\underline{z}_2' = \bar{x}_2 \qquad \underline{h}_2' = \begin{bmatrix} 1 & \bar{x}_1 & \bar{x}_1^2 & \bar{x}_1^3 \end{bmatrix}$$

where again the superscribed bar is used to denote measured values.

It is useful to assume non-linear models in some cases involving linear systems where not all of the state variables are measured. For example, although Figure 3 in the previous chapter describes a linear system of 3 outputs and 1 input, a model for any one of the outputs, obtained from measurements of that output alone, would have to be non-linear. Of course, non-linear models are not always necessary to reduce the order of a linear system, because many linear systems can be realized in terms of reduced linear models. A further sophistication of the identification algorithm for linear systems could provide for appropriate selection of the measured variables to effect such a reduction.

Time-Varying Parameters

Time-varying parameters can be accomodated by modifying the algorithm so that prior estimates are updated at each stage to allow for expected time-variations during the measurement interval. That is, if the parameter array $\underline{A}$ is known to vary according to the difference equation

$$\underline{A}_{k+1} = \underline{\theta}(k+1, k)\ \underline{A}_k$$

then $\hat{\underline{A}}_{k-1}$ in the algorithm is replaced by its a priori update:

$$\hat{\underline{A}}_{k|k-1} = \underline{\theta}\ \hat{\underline{A}}_{k-1}$$

This is a much-used procedure and forms the basis of the Kalman filter for state estimation. Other methods are available if no model for the parameter variations is known (see, for example, Young [17]).

REFERENCES

[1] Flores, I., The Logic of Computer Arithmetic, Prentice-Hall, Englewood Cliffs, N.J., 1963.

[2] Greville, T.N.E., "The Pseudoinverse of a Rectangular or Singular Matrix and its Application to the Solution of Systems of Linear Equations", SIAM Review, Vol. 1, No. 1, January, 1959.

[3] Greville, T.N.E., "Some Applications of the Pseudoinverse of a Matrix", SIAM Review, Vol. 2, No. 1, January, 1960.

[4] Hanafy, A.A.R. and Bohn, E.V., "Rapid Digital Identification of Linear Systems", Joint Automatic Control Conference, St. Louis, Mo., August, 1971.

[5] Ho, Y.C., "The Method of Least Squares and Optimal Filtering Theory", RAND Corp., Santa Monica, Calif., RM-3329-PR, October, 1963.

[6] Kalman, R.E., "A New Approach to Linear Filtering and Prediction Problems", Trans. ASME, Series D, J. Basic Eng., Vol. 82, March, 1960, pp. 35-45.

[7] Kalman, R.E. and Bertram, J.E., "Control System Analysis and Design Via the 'Second Method' of Liapunov, I. Continuous-Time Systems and II. Discrete-Time Systems", Trans. ASME, Series D, J. Basic Eng., Vol. 82, June, 1960, pp. 371-400.

[8] Kishi, F.H., "On-Line Computer Control Techniques and Their Application to Re-entry Aerospace Vehicle Control", Advances in Control Systems Theory and Application, C.T. Leondes, ed., Vol. 1, Academic Press, New York, 1964.

[9] Netravali, A.N. and de Figueiredo, R.J.P., "On the Identification of Nonlinear Dynamical Systems", IEEE Trans. Automat. Contr., Vol. AC-16, No. 1, February,

1971, pp. 28-36.

[10] Ogata, K., State-Space Analysis of Control Systems, Prentice-Hall, Englewood Cliffs, N.J., 1968.

[11] Penrose, R., "A Generalized Inverse for Matrices", Proc. Cambridge Philos. Soc., Vol. 51, 1955, pp. 406-413.

[12] Penrose, R., "On Best Approximate Solutions of Linear Matrix Equations", Proc. Cambridge Philos. Soc., Vol. 52, 1956.

[13] Sage, A.P., Optimum Systems Control, Prentice-Hall, Englewood Cliffs, N.J., 1968.

[14] Sage, A.P. and Melsa, J.L., Estimation Theory with Applications to Communications and Control, McGraw-Hill, New York, 1971.

[15] Sinha, N.K. and Pille, W., "Online System Identification Using Matrix Pseudoinverse", Electronics Letters, Vol. 6, No. 15, July 23, 1970, pp. 453, 454.

[16] Van Trees, H.L., Detection, Estimation and Modulation Theory, Part I, John Wiley and Sons, New York, 1968.

[17] Young, P.C., "Applying Parameter Estimation to Dynamic Systems-Part I", Control Engineering, Vol. 16, No. 10, October, 1969, pp. 119-125.

APPENDIX

Program-Equivalents of Symbols Appearing in the Text

Programs	Text
A	$\hat{\underline{A}}_k$
B	$\underline{b}_k$, $\underline{e}_k$
C	$\underline{c}_k$
CSQU	$\underline{c}_k'\underline{c}_k$
CTHR	threshold for $\underline{c}_k'\underline{c}_k$
H	$\underline{h}_k$, $\underline{z}_k$
I	i
J	j
KØ	k_{MAX}
P	$\underline{P}_k$, $\underline{R}_k$
Q	$\underline{Q}_k$
R	r, n
RS	rs, n(m+n)
S	s, m+n
SS	s^2, $(m+n)^2$
V	s_k

Memory-Allocation for Identification Programs

Labels apply to main-program assembly only. Locations available for user-written programs are marked by asterisks.

Locations	Label:Content	Use
ØØØØ-ØØØ1		*
ØØØ2		starting address of main program
ØØØ3		*
ØØØ4-ØØØ7		required by floating-pt. interpreter
ØØ1Ø-ØØ37		*
ØØ4Ø-ØØ43		required by floating-pt. interpreter
ØØ44-Ø277		*
Ø3ØØ	KEEP	pointers, indicators
Ø3Ø1	SAVE	pointers, indicators
Ø3Ø2	AMAT	pointers, indicators
Ø3Ø3	AMATØ	pointers, indicators
Ø3Ø4	BMAT	pointers, indicators
Ø3Ø5	BMATØ	pointers, indicators
Ø3Ø6		*
Ø3Ø7	ZERO : Ø	floating-point zero
Ø31Ø	Ø	floating-point zero
Ø311	ONE : 1	one
Ø312	R	no. of columns in parameter array (r)
Ø313	S	number of rows in parameter array (s)
Ø314	RS	product rs
Ø315	SS	product ss
Ø316	I	indicator
Ø317	J	counter
Ø32Ø		*
Ø321	KØ	indicators and counters
Ø322	L	indicators and counters
Ø323	LØ	indicators and counters
Ø324	M	indicators and counters
Ø325	N	indicators and counters
Ø326	NØ	indicators and counters
Ø327		*
Ø33Ø	A : 5ØØ	indirect matrix addresses
Ø331	P : 7ØØ	indirect matrix addresses
Ø332	Q: : 11ØØ	indirect matrix addresses
Ø333	TEMP1: 13ØØ	indirect matrix addresses
Ø334	TEMP2: 15ØØ	indirect matrix addresses
Ø335	TEMP3: 17ØØ	indirect matrix addresses
Ø336	H : 21ØØ	indirect matrix addresses
Ø337	C : 212Ø	indirect matrix addresses

```
Ø34Ø       B:    :  214Ø   }indirect matrix addresses
Ø341       CSQU  :  216Ø   }
Ø342       CTHR  :Ø4Ø42Ø   }threshold for c'_k c_k, initially 1
Ø343                   Ø   }
Ø344                 27ØØ  contains starting adr. of main prog.
Ø345                 3Ø54  contains starting adr. of calc'ns
Ø346       V     :Ø4Ø42Ø   }measurement variance, initially
Ø347                   Ø   }loaded as floating-point "1"

Ø35Ø       MXADD:   223Ø   }
Ø351       MXSUB:   2251   }
Ø352       MXMPY:   2272   }
Ø353       MXDIV:   2355   }
Ø354       MXTR :   2374   }
Ø355       DATRD:   244Ø   }indirect subroutine addresses
Ø356       DATPN:   2473   }
Ø357       DATRC:   2533   }
                           }
Ø36Ø       DIGIT:   2572   }
Ø361       DATWR:   26ØØ   }
Ø362       WRITE:   2646   }
Ø363       INIT            }
Ø364       MEAS            }indirect addresses for user's
Ø365       DATIN           }subroutines
Ø366       DTOUT           }
Ø367                       *

Ø37Ø       STR1 :   217Ø   }
Ø371       STR2 :   2174   }
Ø372       STR3 :   22ØØ   }
Ø373       STR4 :   22Ø4   }indirect addresses for
Ø374       STR5 :   221Ø   }teleprinter message strings
Ø375       STR6 :   2214   }
Ø376       STR7 :   222Ø   }
Ø377       STR8 :   2224   }

Ø4ØØ-Ø477                  floating-pt. interpreter work area
Ø5ØØ-2161                  matrix storage area (see 33Ø-341)
2162-2167                  *
217Ø-2227                  teleprinter message strings
223Ø-2431                  matrix arith. subr. (see 35Ø-354)
2432-2437                  *
244Ø-2666                  I/O subroutines (see 355-362)
2667                       *
27ØØ-34Ø3  BEGIN           main program
34Ø4-5577                  *
56ØØ-6577                  basic floating-pt. interpreter
```

Instructions for Using the Identification Program-Package

First load the program tapes in the following order:

1. Nova Basic Floating-Point Interpreter
2. Data-supply subroutines (INIT, MEAS, DATIN, DTOUT)
3. Identification program-package

The program is self-starting and will begin by printing certain questions which are to be answered by typing numbers into the teletypewriter. Each number will be required in either fixed- or floating-point format. In the case of fixed-point only one decimal digit will be accepted, while floating-point format can be any string of characters in the following order:

1. A + or - sign (optional)
2. A string of decimal digits (optional)
3. A decimal point (optional)
4. A string of decimal digits
5. The letter E, if there is to be an exponent
6. A + or - sign (optional)
7. One or two decimal digits denoting exponent (optional)
8. A "space"

A character typed in error can be deleted with a "rubout". Examples of allowed strings are: 5ØØ_, +5Ø_, +5.E2_, -2.Ø5E-Ø4_, +.3Ø54E-22_, -2EØ3_, where _ denotes a "space".

The questions printed and explanations of the required responses are as follows:

1. "R = ______" : A fixed-point integer from 1 to 8 equalling r, the number of columns in the parameter array, or n, the number of system outputs.

2. "S = ______" : A fixed-point integer from 1 to 8 equalling s, the number of rows in the parameter array, or m+n, where m is the number of system inputs and n is the number of

system outputs.

3. "SAMPLES?" : A number in floating-point format equal to the number of state-samples to be taken.

4. "COPY?   " : A fixed-point integer corresponding to one of the following instructions regarding starting values:

Ø: No starting values are available for A and P.

1: The starting values now in memory are to be used.

2: Copy the starting values from memory onto paper tape.

3: Copy the starting values from memory onto the teleprinter.

4: Read the starting values from the tape in the high-speed reader and enter them into memory (tape must be one which has been produced by response 2).

5: Accept the starting values from the teletype keyboard. (Note that following this response the program will print "PARAMS " after which the elements of the parameter matrix should be typed in floating-point format row by row. A carriage-return and line-feed will occur automatically after each element has been typed and an extra line-feed will occur at the end of each row. When all rows are finished, the program will print "P-MATRIX" and the starting values of the P-matrix elements should then be typed row by row in floating-point format.)

6: Execute the user-written subroutine whose starting address is found in location INIT = 363.

7: Accept new initial values for CTHR and V from the teletype keyboard. (The program will respond by printing "CTHR, V:" after which the values of CTHR and V should be typed in floating-point format one after the other.)

5. "READY?   " : A fixed-point integer corresponding to:

Ø: Return for another pass at question 4.

1: Proceed to execute the identification program.

IMPORTANT: The last response to question 4 must be "Ø" or "1"

Instructions for Writing I/O Subroutines

INIT: This subroutine is called if a "6" is typed in response to the question "COPY?" and allows the user to supply starting values with his own subroutine. Starting address should be stored in location 363.

MEAS: This subroutine is called just before the recursive identification process is started and can be used for such tasks as rapid pre-measuring and storing of data. Its starting address should be entered into location 364.

DATIN: Called each time a new sample of the system outputs is required. Starting address should be loaded into location 365.

DTOUT: Called just after a new estimate of the parameters has been calculated and useful for outputting the parameter matrix. The starting address should be loaded into location 366.

The model estimated by the program will be (see Chapter V)

$$\underline{x}_k(n) = \emptyset(n\times n)\,\underline{x}_{k-1}(n) + \underline{\Delta}(n\times m)\,\underline{u}_{k-1}(m) = \underline{A}'(n\times(n+m))\,\underline{h}_k(n+m)$$

where

$$\underline{h}_k = \begin{bmatrix} \underline{x}_{k-1} \\ \cdots\cdot \\ \underline{u}_{k-1} \end{bmatrix} \qquad \underline{A} = \begin{bmatrix} \underline{\emptyset}' \\ \cdots \\ \underline{\Delta}' \end{bmatrix}$$

The user's data-supply subroutines should store measured values of $\underline{x}_k$ and $\underline{u}_k$ in the $\underline{h}$-vector locations, which begin at the address found in location H = 336. $\underline{x}_k$ is stored first and then $\underline{u}_k$, each element to be written in 32-bit hexadecimal floating-point format occupying 2 consecutive locations as provided by the Nova instruction FFLO. The maximum number of elements in

$\underline{h}$ is 8. The estimated parameter array $\underline{A}$ will be left row by row starting at the address contained in location A = 33Ø, each element in floating-point format and occupying 2 consecutive locations. Output via the teleprinter or paper-tape punch can be achieved using the subroutines which are addressed indirectly through locations DATWR = 361 and DATPN = 356.

Values of the variance term s_k for a maximum-likelihood filter can be entered in floating-point format into locations V = 346 and V + 1 = 347.

The total number of state samples measured or estimation cycles performed is controlled by the user's subroutines, using location K = 32Ø as a counter. The initial count, which is typed in response to the question "SAMPLES?", is found in location KØ = 321.

Dimension parameters typed in response to "R = " and "S = " and the locations where they are stored are:

R = 312	r, n
S = 313	s, m + n
RS = 314	rs, n(m+n)
SS = 315	s^2, $(m+n)^2$

Example:

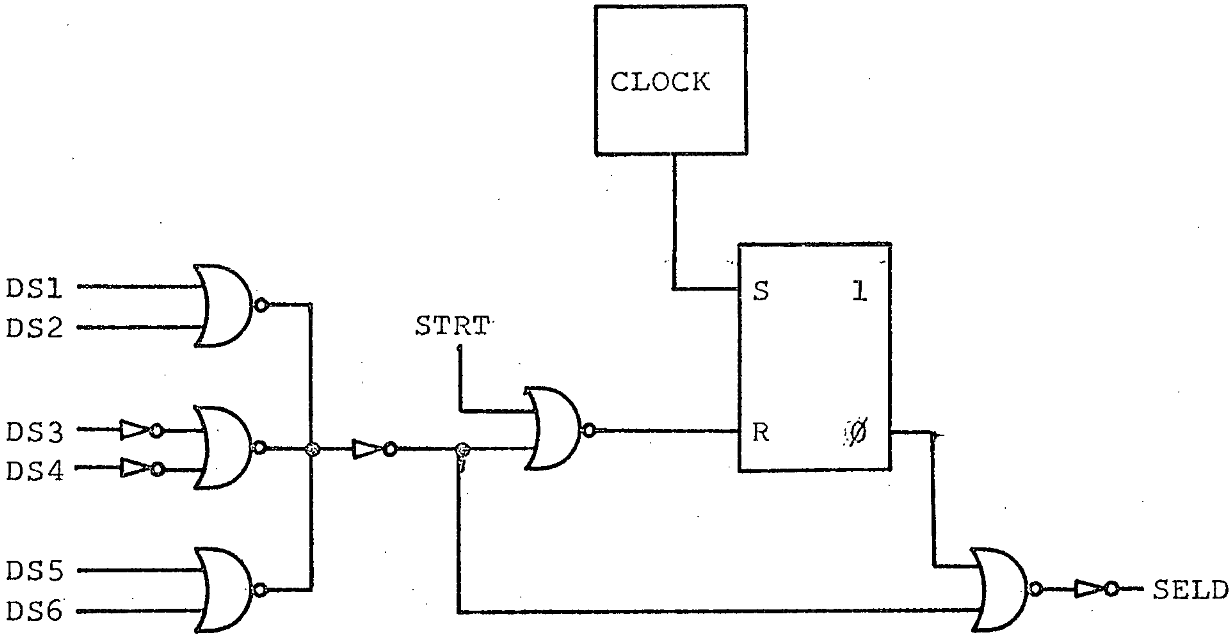
The following set of programs are examples of the subroutines, MEAS, DATIN and DTOUT, required to make and store a rapid set of state-samples of a continuous system via an A/D converter with multiplexed inputs, and under control of an external real-time clock. The stored data is to be processed one row at a time by the identification program, after which the parameter estimate is to be printed, along with a warning in the case of a minimum-norm composition if an insufficient number of linearly independent measurements were available for the parameters to be observable.

```
; DATA-SUPPLY SUBROUTINES FOR A/D CONVERTER

         R          = 312
         S          = 313
         I          = 316
         J          = 317
         K          = 32Ø
         KØ         = 321
         N          = 325
         A          = 33Ø
         H          = 336
         BEGIN      = 344
         START      = 345
         MXTR       = 354
         DATWR      = 361
         WRITE      = 362
         STR5       = 374

         .LOC 364
         MEAS
         DATIN
         DTOUT

         .LOC 341Ø
         111116             ; STRING 11: "INS MEAS"
         123Ø4Ø
         1151Ø5
         1Ø1123

MAX:     2Ø4Ø               ; MAX NO OF STOR LOC AVAILABLE
STORE:   3537               ; IND ADR FOR 1ST STOR LOC
MEAS:    LDA 3, R
         STA 3, N           ; N = R
         LDA 3, KØ
         STA 3, K           ; PRESET SAMPLE COUNTER
         SUB 2, 2           ; CLEAR AC2
         ADD 3, 2
         DSZ N
         JMP .-2            ; AC2 = KR
         LDA 3, MAX         ; AC3 = MAX
         SUBZ# 2, 3, SNC    ; SKIP IF KR NOT EXCEED MAX
         JMP @BEGIN         ; RESTART
         LDA 3, STORE
         STA 3, 21          ; PRESET LOC POINTER
         SUBO 2, 2          ; AC2 = Ø
SMPL:    LDA 3, R
         STA 3, N           ; RESET MEASUREMENT COUNTER
```

```
        DOAC 2, 44         ; SET MUX CHANNEL TO Ø
        NIOS 63            ; ENABLE HARDWARE CLOCK
        SKPDN 63
        JMP .-1            ; WAIT FOR CLOCK
CRRNT:  NIOC 51            ; CLEAR A/D
        NIOS 51            ; START A/D
        SKPDN 51
        JMP..-1            ; WAIT FOR A/D
        DIA Ø, 51          ; GET RESULT
        STA Ø, @21         ; STORE RESULT
        NIOP 44            ; INC MUX CHANNEL
        DSZ N              ; SKIP IF DONE CURRENT SAMPLE
        JMP CRRNT
        DSZ K              ; SKIP IF DONE ALL SAMPLES
        JMP SMPL
        LDA 3, KØ
        STA 3, K
        ISZ K              ; PRESET SAMPLE COUNTER
        LDA 3, STORE
        STA 3, 21
        JMP @START

DATIN:  STA 3, RETURN      ; STORE RETURN ADR
        DSZ K              ; SKIP IF NO MORE DATA
        JMP .+2
        JMP OUT
        LDA 2, H           ; PRESET LOC POINTER
        LDA 3, R           ; AC3 = R
        STA 3, N           ; PRESET MEAS COUNTER
CMPNT:  SUB Ø, Ø           ; ACØ = Ø
        LDA 1, @21         ; GET DATA WORD
        MOVL# 1, 1, SZC    ; SKIP IF NON-NEGATIVE
        COM Ø, Ø           ; ACØ = 177777
        STA Ø, Ø, 2
        STA 1, 1, 2        ; STORE DATUM IN H
        FETR
        FFLO Ø, 2          ; CONVERT TO FP
        FIC2               ; INC LOC POINTER
        FEXT
        DSZ N              ; SKIP IF HAVE ALL COMPS
        JMP CMPNT
        JMP @RETURN        ; RETURN
RETURN: Ø
STR11:  341Ø
OUT:    LDA 3, I
        MOV 3, 3, SZR      ; SKIP IF I = Ø
        JMP PRINT
        LDA 3, J           ; AC3 = J
        SUBZL 1, 1         ; AC1 = 1
        ADCZ# 1, 3, SNC    ; SKIP IF J GREATER THAN 1
        JMP .+3
        LDA 2, STR11
        JSR @WRITE         ; TYPE "INS MEAS"
```

```
PRINT:   LDA 2, STR5
         JSR @WRITE        ; TYPE "PARAMS  "
         LDA Ø, S
         LDA 1, R
         LDA 2, A
         JSR @DATWR        ; PRINT PARAMETERS
         JMP @BEGIN        ; RESTART MAIN PROG

DTOUT:   JMP Ø, 3          ; RETURN

         .END
```

INTERFACE FOR REAL-TIME CLOCK



Identification Programs for Nova Computer

On the following pages are found the assembler listings of the basic identification programs for the Nova computer.

```
                ; MATRIX ARITHMETIC SUBROUTINES

                ; REQUIRE BASIC FLOATING POINT INTERPRETER


      000300            KEEP    = 300
      000301            SAVE    = 301
      000302            AMAT    = 302
      000303            AMAT0   = 303
      000304            BMAT    = 304
      000305            BMAT0   = 305
      000322            L       = 322
      000323            L0      = 323
      000324            M       = 324
      000325            N       = 325
      000326            N0      = 326

      000307            .LOC 307
00307 000000  ZERO:     0
00310 000000            0

      000350            .LOC 350
00350 002230            MXADD           ; THESE ARE THE PAGE-0
00351 002251            MXSUB           ; ADDRESSES IN WHICH THE
00352 002272            MXMPY           ; STARTING ADDRESSES OF
00353 002355            MXDIV           ; THE SUBROUTINES CAN BE
00354 002374            MXTR            ; FOUND.

      002230            .LOC 2230


                ; SUBR TO ADD TWO MATRICES (C = A + B)

                ; ENTER WITH:   LOC N   = NO. OF ELTS IN EACH MATRIX
                ;               AC0     = ADR OF A
                ;               AC1     = ADR OF B
                ;               AC2     = ADR OF C

02230 040302  MXADD:    STA 0, AMAT     ; PRESET A-MATRIX POINTER
02231 044304            STA 1, BMAT     ; PRESET B-MATRIX POINTER
02232 054301            STA 3, SAVE     ; STORE RETURN ADR
02233 006004  XADD:     FETR            ; ENTER FP MODE
02234 022302            FLDA 0, @AMAT   ; GET ELT OF A
02235 026304            FLDA 1, @BMAT   ; GET ELT OF B
02236 123000            FADD 1, 0       ; ADD ELTS
02237 041000            FSTA 0, 0, 2    ; STORE RESULT IN C
02240 104000            FIC2            ; INC C-MATRIX POINTER
02241 100000            FEXT            ; EXIT FP MODE
02242 010302            ISZ AMAT
02243 010302            ISZ AMAT        ; INC A-MATRIX POINTER
02244 010304            ISZ BMAT
02245 010304            ISZ BMAT        ; INC B-MATRIX POINTER
02246 014325            DSZ N           ; SKIP IF ALL ELTS ADDED
02247 000764            JMP XADD        ; ADD NEXT PAIR OF ELTS
02250 002301            JMP @SAVE       ; RETURN


                ; SUBR TO SUBTRACT ONE MATRIX FROM ANOTHER (C = A - B)

                ; ENTER WITH:   LOC N   = NO. OF ELTS IN EACH MATRIX
```

```
                      ;               AC0       = ADR OF A
                      ;               AC1       = ADR OF B
                      ;               AC2       = ADR OF C

02251 040302  MXSUB:  STA 0, AMAT      ; PRESET A-MATRIX POINTER
02252 044304          STA 1, BMAT      ; PRESET B-MATRIX POINTER
02253 054301          STA 3, SAVE      ; STORE RETURN ADR
02254 006004  XSUB:   FETR             ; ENTER FP MODE
02255 022302          FLDA 0, @AMAT    ; GET ELT OF A
02256 026304          FLDA 1, @BMAT    ; GET ELT OF B
02257 122400          FSUB 1, 0        ; SUBTRACT ELT OF B FROM ELT OFA
02260 041000          FSTA 0, 0, 2     ; STORE RESULT IN C
02261 104000          FIC2             ; INC C-MATRIX POINTER
02262 100000          FEXT             ; EXIT FP MODE
02263 010302          ISZ AMAT
02264 010302          ISZ AMAT         ; INC A-MATRIX POINTER
02265 010304          ISZ BMAT
02266 010304          ISZ BMAT         ; INC B-MATRIX POINTER
02267 014325          DSZ N            ; SKIP IF DONE
02270 000764          JMP XSUB         ; DO ANOTHER SUBTRACTION
02271 002301          JMP @SAVE        ; RETURN


                      ; SUBR TO MULTIPLY TWO MATRICES (C = AB)

                      ; ENTER WITH:   LOC L     = NO. OF COLUMNS IN A/ROWS IN B
                      ;               LOC M     = NO. OF ROWS IN A
                      ;               LOC N     = NO. OF COLUMNS IN B
                      ;               AC0       = ADR OF A
                      ;               AC1       = ADR OF B
                      ;               AC2       = ADR OF C

02272 040302  MXMPY:  STA 0, AMAT
02273 040303          STA 0, AMAT0     ; PRESET A-MATRIX POINTERS
02274 044304          STA 1, BMAT
02275 044305          STA 1, BMAT0     ; PRESET B-MATRIX POINTERS
02276 054301          STA 3, SAVE      ; STORE RETURN ADR
02277 034322          LDA 3, L
02300 054323          STA 3, L0        ; LOC L0 = NO. OF COLUMNS IN A
02301 024325          LDA 1, N
02302 044326          STA 1, N0        ; LOC N0 = NO. OF COLUMNS IN B
02303 127000          ADD 1, 1         ; AC1 = TWICE NO. COLS IN B
02304 000420          JMP XMPY+2
02305 034302  MRET:   LDA 3, AMAT
02306 054303          STA 3, AMAT0     ; BEGIN NEXT ROW OF A
02307 034305          LDA 3, BMAT0
02310 054304          STA 3, BMAT      ; GO TO FIRST COLUMN OF B
02311 034326          LDA 3, N0
02312 054325          STA 3, N         ; RESET COLUMN-COUNTER
02313 000407          JMP XMPY
02314 034300  NRET:   LDA 3, KEEP
02315 054304          STA 3, BMAT
02316 010304          ISZ BMAT
02317 010304          ISZ BMAT         ; BEGIN NEXT COLUMN OF B
02320 034303          LDA 3, AMAT0
02321 054302          STA 3, AMAT      ; REPEAT SAME ROW OF A
02322 034323  XMPY:   LDA 3, L0
02323 054322          STA 3, L         ; RESET PRODUCT-COUNTER
02324 034304          LDA 3, BMAT
02325 054300          STA 3, KEEP      ; STORE COLUMN-POINTER
```

```
02326 006004           FETR            ; ENTER FP MODE
02327 030307           FLDA 2, ZERO    ; ZERO CUMULATIVE SUM
02330 022302  LRET:    FLDA 0, @AMAT   ; GET ELT OF A
02331 026304           FLDA 1, @BMAT   ; GET ELT OF B
02332 120100           FMPY 1, 0       ; MULTIPLY ELTS
02333 113000           FADD 0, 2       ; ADD PROD TO CUMULATIVE SUM
02334 100000           FEXT            ; EXIT FP MODE
02335 010302           ISZ AMAT
02336 010302           ISZ AMAT        ; MOVE ALONG ROW OF A
02337 034304           LDA 3, BMAT
02340 137000           ADD 1, 3
02341 054304           STA 3, BMAT     ; MOVE DOWN COLUMN OF B
02342 006004           FETR            ; ENTER FP MODE
02343 014322           FDSZ L          ; SKIP IF ALL PRODUCTS DONE
02344 000764           FJMP LRET       ; FORM NEXT PRODUCT
02345 051000           FSTA 2, 0, 2    ; STORE RESULT IN C
02346 104000           FIC2            ; MOVE ALONG ROW OF C
02347 100000           FEXT            ; EXIT FP MODE
02350 014325           DSZ N           ; SKIP IF DONE ALL COLS OF B
02351 000743           JMP NRET
02352 014324           DSZ M           ; SKIP IF DONE ALL ROWS OF A
02353 000732           JMP MRET
02354 002301           JMP @SAVE       ; RETURN
```

```
                    ; SUBR TO DIVIDE A MATRIX BY A SCALAR (C = A/B)

                    ; ENTER WITH:    LOC N    = NO. OF ELTS IN A
                    ;                AC0      = ADR OF A
                    ;                AC1      = ADR OF B
                    ;                AC2      = ADR OF C

02355 040302  MXDIV:   STA 0, AMAT      ; PRESET A-MATRIX POINTER
02356 044304           STA 1, BMAT      ; PRESET B-MATRIX POINTER
02357 054301           STA 3, SAVE      ; STORE RETURN ADR

02360 006004  XDIV:    FETR             ; ENTER FP MODE
02361 022302           FLDA 0, @AMAT    ; GET ELT OF A
02362 026304           FLDA 1, @BMAT    ; GET ELT OF B
02363 120200           FDIV 1, 0        ; DIVIDE ELT OF A
02364 041000           FSTA 0, 0, 2     ; STORE RESULT IN C
02365 104000           FIC2             ; INC C-MATRIX POINTER
02366 100000           FEXT             ; EXIT FP MODE
02367 010302           ISZ AMAT
02370 010302           ISZ AMAT         ; INC A-MATRIX POINTER
02371 014325           DSZ N            ; SKIP IF ALL ELTS DIVIDED
02372 000766           JMP XDIV
02373 002301           JMP @SAVE        ; RETURN


                    ; SUBR TO TRANSPOSE A SQUARE MATRIX (A = A')

                    ; ENTER WITH:    AC1      = NO. OF ROWS OR COLUMNS OF A
                    ;                AC2      = ADR OF A

02374 044324  MXTR:    STA 1, M
02375 014324           DSZ M            ; LOC M = 1 LESS THAN NO. ROWS
02376 000402           JMP .+2
02377 001400           JMP 0, 3
02400 054301           STA 3, SAVE      ; STORE RETURN ADR
02401 127000           ADD 1, 1         ; AC1 = TWICE NO. OF ROWS
02402 121400           INC 1, 0
02403 101400           INC 0, 0         ; AC0 = TWICE NO. ROWS + 2
02404 000403           JMP .+3
02405 030303  TRRET:   LDA 2, AMAT0
02406 113000           ADD 0, 2         ; MOVE DOWN DIAGONAL
02407 050303           STA 2, AMAT0     ; STORE ELT POINTER
02410 050302           STA 2, AMAT      ; PRESET FIRST ELT POINTER
02411 034324           LDA 3, M
02412 054325           STA 3, N         ; PRESET COUNTER
02413 034302  XTR:     LDA 3, AMAT
02414 137000           ADD 1, 3
02415 054302           STA 3, AMAT      ; SET FIRST ELT-POINTER
02416 006004           FETR             ; ENTER FP MODE
02417 104000           FIC2             ; SET SECOND ELT-POINTER
02420 026302           FLDA 1, @AMAT
02421 031000           FLDA 2, 0, 2     ; GET ELTS
02422 045000           FSTA 1, 0, 2
02423 052302           FSTA 2, @AMAT    ; SWAP ELTS
02424 100000           FEXT             ; EXIT FP MODE
02425 014325           DSZ N            ; SKIP IF DONE ROW
02426 000765           JMP XTR
02427 014324           DSZ M            ; SKIP IF DONE MATRIX
02430 000755           JMP TRRET
```

```
JMP @ SAVE      ; RETURN

.END 7777
```

```
---
AMAT     ØØØ3Ø2
AMATØ    ØØØ3Ø3
BMAT     ØØØ3Ø4
BMATØ    ØØØ3Ø5
KEEP     ØØØ3ØØ
L        ØØØ322
LØ       ØØØ323
LRET     ØØ233Ø
M        ØØØ324
MRET     ØØ23Ø5
MXADD    ØØ223Ø
MXDIV    ØØ2355
MXMPY    ØØ2272
MXSUB    ØØ2251
MXTR     ØØ2374
N        ØØØ325
NØ       ØØØ326
NRET     ØØ2314
SAVE     ØØØ3Ø1
TRRET    ØØ24Ø5
XADD     ØØ2233
XDIV     ØØ236Ø
XMPY     ØØ2322
XSUB     ØØ2254
XTR      ØØ2413
ZERO     ØØØ3Ø7
```

```
                ; INPUT-OUTPUT SUBROUTINES FOR TTY AND PTP

                ; REQUIRE BASIC FLOATING-POINT INTERPRETER

      000040            .LOC 40            ; THESE ARE THE SUBROUTINES
00040 002560            RECV               ; TO BE USED BY FP INSTRUCTIONS
00041 002637            TYPE               ; FDFC AND FFDC, RESPECTIVELY

      000300            KEEP    = 300
      000301            SAVE    = 301
      000324            M       = 324
      000325            N       = 325
      000326            N0      = 326

      000355            .LOC 355           ; THESE ARE THE PAGE-0
00355 002440            DATRD              ; ADDRESSES IN WHICH THE
00356 002473            DATPN              ; STARTING ADDRESSES OF
00357 002533            DATRC              ; THE SUBROUTINES CAN BE
00360 002572            DIGIT              ; FOUND.
00361 002600            DATWR
00362 002646            WRITE

      002440            .LOC 2440


                ; SUBR TO STORE DATA FROM PAPER TAPE

                ; ENTER WITH:   AC1        = NO. OF FLOATING-POINT DATA
                ;               AC2        = STARTING ADR OF STORAGE LOC

02440 127000    DATRD:  ADD 1, 1           ; DOUBLE AC1
02441 044325            STA 1, N           ; N = NO. OF DATA WORDS
02442 050020            STA 2, 20
02443 014020            DSZ 20             ; SET LOC POINTER
02444 054301            STA 3, SAVE        ; STORE RETURN ADR
02445 060112            NIOS PTR           ; READ A LINE FROM TAPE
02446 063612            SKPDN PTR
02447 000777            JMP .-1
02450 060512            DIAS 0, PTR        ; GET RESULT, READ AGAIN
02451 063612            SKPDN PTR
02452 000777            JMP .-1
02453 101005            MOV 0, 0, SNR      ; SKIP IF RESULT NON-ZERO
02454 000774            JMP .-4
02455 004405            JSR READ           ; GET DATA WORD
02456 042020            STA 0, @20         ; STORE DATA WORD
02457 014325            DSZ N              ; SKIP IF ALL WORDS READ
02460 000775            JMP .-3
02461 002301            JMP @SAVE          ; RETURN

02462 064512    READ:   DIAS 1, PTR        ; GET RESULT, READ AGAIN
02463 063612            SKPDN PTR
02464 000777            JMP .-1
02465 125300            MOVS 1, 1          ; LEFT-JUSTIFY IN AC1
02466 060512            DIAS 0, PTR        ; GET RESULT, READ AGAIN
02467 063612            SKPDN PTR
02470 000777            JMP .-1
02471 123000            ADD 1, 0           ; COMBINE HALVES
02472 001400            JMP 0, 3           ; RETURN
```

```
                        ; SUBR TO PUNCH DATA ON PAPER TAPE

                        ; ENTER WITH:   AC1        = NO. OF FLOATING-POINT DATA
                        ;               AC2        = STARTING ADR OF STORAGE LOC

02473 127000            DATPN:  ADD 1, 1           ; DOUBLE AC1
02474 044325                    STA 1, N           ; N = NO. OF DATA WORDS
02475 050020                    STA 2, 20
02476 014020                    DSZ 20             ; SET LOC POINTER
02477 054301                    STA 3, SAVE        ; STORE RETURN ADR
02500 102400                    SUB 0, 0           ; ZERO AC0
02501 152420                    SUBZ 2, 2          ; ZERO AC2, SET CARRY
02502 004421                    JSR PUNCH          ; PUNCH A ZERO
02503 151103                    MOVL 2, 2, SNC     ; COUNT OF 17
02504 000776                    JMP .-2
02505 100000                    COM 0, 0           ; SET AC0
02506 061113                    DOAS 0, PTP        ; PUNCH A 377
02507 063613                    SKPDN PTP
02510 000777                    JMP .-1
02511 022020                    LDA 0, @20         ; GET DATA WORD
02512 004411                    JSR PUNCH          ; PUNCH DATA WORD
02513 014325                    DSZ N              ; SKIP IF ALL WORDS PUNCHED
02514 000775                    JMP .-3
02515 102400                    SUB 0, 0           ; ZERO AC0
02516 152420                    SUBZ 2, 2          ; ZERO AC2, SET CARRY
02517 004404                    JSR PUNCH          ; PUNCH A ZERO
02520 151103                    MOVL 2, 2, SNC     ; COUNT OF 17
02521 000776                    JMP .-2
02522 002301                    JMP @SAVE          ; RETURN

02523 105300            PUNCH:  MOVS 0, 1          ; RIGHT-JUSTIFY FIRST HALF
02524 065113                    DOAS 1, PTP        ; PUNCH FIRST HALF
02525 063613                    SKPDN PTP
02526 000777                    JMP .-1
02527 061113                    DOAS 0, PTP        ; PUNCH SECOND HALF
02530 063613                    SKPDN PTP
02531 000777                    JMP .-1
02532 001400                    JMP 0, 3           ; RETURN


                        ; SUBR TO STORE DATA FROM KEYBOARD

                        ; ENTER WITH:   AC0        = NO. OF ROWS OF FP DATA
                        ;               AC1        = NO. OF COLUMNS
                        ;               AC2        = STARTING ADR OF STORAGE LOC

02533 040324            DATRC:  STA 0, M           ; M = NO. OF ROWS
02534 044326                    STA 1, N0          ; N0 = NO. OF COLUMNS
02535 054301                    STA 3, SAVE        ; STORE RETURN ADR
02536 034326            NXTRW:  LDA 3, N0
02537 054325                    STA 3, N           ; N = NO. OF COLUMNS
02540 020503                    LDA 0, LF
02541 004476                    JSR TYPE           ; LINE-FEED
02542 020502            NXTEL:  LDA 0, CR
02543 004474                    JSR TYPE           ; CARRIAGE-RETURN
02544 020477                    LDA 0, LF
02545 004472                    JSR TYPE           ; LINE-FEED
02546 006004                    FETR               ; ENTER FLOATING-POINT MODE
02547 124000                    FDFC 1             ; ACCEPT DEC NO., CONVERT
```

```
02550 045000           FSTA 1, 0, 2     ; STORE HEXADECIMAL NO.
02551 104000           FIC2             ; INC STORAGE-LOC POINTER
02552 100000           FEXT             ; EXIT FP MODE
02553 014325           DSZ N            ; SKIP IF HAVE ALL ELTS OF ROW
02554 000766           JMP NXTEL
02555 014324           DSZ M            ; SKIP IF HAVE ALL ROWS
02556 000760           JMP NXTRW
02557 002301           JMP @SAVE        ; RETURN

02560 054300  RECV:    STA 3, KEEP      ; STORE RETURN ADR
02561 060110           NIOS TTI         ; ENABLE KEYBOARD
02562 063610           SKPDN TTI
02563 000777           JMP .-1          ; WAIT FOR CHARACTER
02564 060410           DIA 0, TTI       ; GET CHARACTER
02565 024404           LDA 1, MASK      ; AC1 = 177
02566 123400           AND 1, 0         ; MASK TO 7 BITS
02567 004450           JSR TYPE         ; ECHO CHARACTER
02570 002300           JMP @KEEP        ; RETURN
02571 000177  MASK:    177

              ; SUBR TO ACCEPT A DIGIT FROM KEYBOARD

              ; BINARY VALUE OF DIGIT IS LEFT IN AC0

02572 054301  DIGIT:   STA 3, SAVE      ; STORE RETURN ADR
02573 004765           JSR RECV         ; RETURN DIGIT IN AC0
02574 024403           LDA 1, DTMSK     ; AC1 = 17
02575 123400           AND 1, 0         ; MASK TO 4 BITS
02576 002301           JMP @SAVE        ; RETURN
02577 000017  DTMSK:   17
```

```
                ; SUBR TO TYPE DATA ON TELEPRINTER

                ; ENTER WITH:   AC0      = NO. OF ROWS OF FP DATA
                ;               AC1      = NO. OF COLUMNS
                ;               AC2      = FIRST ADR WHERE DATA STORED

02600 040324    DATWR:  STA 0, M           ; LOC M = NO. OF ROWS
02601 044326            STA 1, N0          ; N0 = NO. OF COLUMNS
02602 054301            STA 3, SAVE        ; STORE RETURN ADR
02603 125112    ROW:    MOVL# 1, 1, SZC    ; SKIP IF 2 LINES TYPED
02604 000403            JMP .+3            ; NO LINE-FEED
02605 020436            LDA 0, LF
02606 004431            JSR TYPE           ; LINE-FEED
02607 024427            LDA 1, COLS        ; AC1 = -4
02610 034326            LDA 3, N0
02611 054325            STA 3, N           ; N = NO. OF COLUMNS
02612 020432    LINE:   LDA 0, CR
02613 004424            JSR TYPE           ; CARRIAGE-RETURN
02614 020427            LDA 0, LF
02615 004422            JSR TYPE           ; LINE-FEED
02616 020427    ELT:    LDA 0, SP
02617 004420            JSR TYPE           ; SPACE
02620 004417            JSR TYPE           ; SPACE
02621 006004            FETR               ; ENTER FP MODE
02622 021000            FLDA 0, 0, 2       ; LOAD HEX FP NO. IN FAC0
02623 140000            FFDC 0             ; TYPE NO. IN DEC FP
02624 104000            FIC2               ; INC STORAGE-LOC POINTER
02625 100000            FEXT               ; EXIT FP MODE
02626 014325            DSZ N              ; SKIP IF DONE ALL ELTS OF ROW
02627 000404            JMP .+4
02630 014324            DSZ M              ; SKIP IF DONE ALL ROWS
02631 000752            JMP ROW
02632 002301            JMP @SAVE          ; RETURN
02633 125404            INC 1, 1, SZR      ; SKIP IF FINISHED LINE
02634 000762            JMP ELT
02635 000755            JMP LINE
02636 177774    COLS:   -4

02637 061111    TYPE:   DOAS 0, TTO        ; TYPE CHARACTER
02640 063611            SKPDN TTO
02641 000777            JMP .-1
02642 001400            JMP 0, 3           ; RETURN
02643 000012    LF:     12
02644 000015    CR:     15
02645 000040    SP:     40


                ; SUBR TO TYPE A STRING OF 8 CHARACTERS

                ; ENTER WITH AC2 = STARTING ADR OF STRING

02646 054301    WRITE:  STA 3, SAVE        ; STORE RETURN ADR
02647 020775            LDA 0, CR
02650 004767            JSR TYPE           ; CARRIAGE RETURN
02651 020772            LDA 0, LF
02652 004765            JSR TYPE
02653 004764            JSR TYPE           ; DOUBLE-SPACE
02654 024762            LDA 1, COLS        ; AC1 = -4
02655 021000    CHAR:   LDA 0, 0, 2        ; GET WORD
```

```
02656 101300        MOVS 0, 0       ; SWAP HALVES
02657 101200        MOVR 0, 0       ; SHIFT RIGHT
02660 004757        JSR TYPE        ; PRINT 1ST CHARACTER
02661 021000        LDA 0, 0, 2     ; GET WORD AGAIN
02662 004755        JSR TYPE        ; PRINT 2ND CHARACTER
02663 151400        INC 2, 2        ; INC LOC POINTER
02664 125404        INC 1, 1, SZR   ; SKIP IF DONE
02665 000770        JMP CHAR
02666 002301        JMP @SAVE       ; RETURN

      007777        .END 7777
```

```
---
CHAR    ØØ2655
COLS    ØØ2636
CR      ØØ2644
DATPN   ØØ2473
DATRC   ØØ2533
DATRD   ØØ244Ø
DATWR   ØØ26ØØ
DIGIT   ØØ2572
DTMSK   ØØ2577
ELT     ØØ2616
KEEP    ØØØ3ØØ
LF      ØØ2643
LINE    ØØ2612
M       ØØØ324
MASK    ØØ2571
N       ØØØ325
NØ      ØØØ326
NXTEL   ØØ2542
NXTRW   ØØ2536
PUNCH   ØØ2523
READ    ØØ2462
RECV    ØØ256Ø
ROW     ØØ26Ø3
SAVE    ØØØ3Ø1
SP      ØØ2645
TYPE    ØØ2637
WRITE   ØØ2646
```

```
                    ; RECURSIVE LEAST-SQUARES IDENTIFICATION

                    ; REQUIRES:     BASIC FP INTERPRETER
                    ;               MATRIX ARITHMETIC SUBROUTINES
                    ;               I/O SUBROUTINES FOR TTY, TAPE
                    ;               DATA-SUPPLY PROGRAMMES

                    ; THIS PROGRAMME USES THE MAXIMUM
                    ; NO. OF SYMBOLS ALLOWED BY THE
                    ; RELOCATABLE ASSEMBLER. DO NOT ADD ANY MORE.

      000312              R       = 312
      000313              S       = 313
      000314              RS      = 314
      000315              SS      = 315
      000316              I       = 316
      000317              J       = 317
      000321              K0      = 321
      000322              L       = 322
      000324              M       = 324
      000325              N       = 325
      000350              MXADD   = 350
      000351              MXSUB   = 351
      000352              MXMPY   = 352
      000353              MXDIV   = 353
      000354              MXIR    = 354
      000355              DATRD   = 355
      000356              DATPN   = 356
      000357              DATRC   = 357
      000360              DIGIT   = 360
      000361              DATWR   = 361
      000362              WRITE   = 362
      000363              INIT    = 363
      000364              MEAS    = 364
      000365              DATIN   = 365
      000366              DTOUT   = 366

      000002              .LOC 2
00002 002344              JMP @344

      000007              .LOC 7
00007 000400              400                  ; WORK AREA FOR FP INTERPRETER

      000311              .LOC 311
00311 000001 ONE:         1

      000330              .LOC 330
00330 000500 A:           500                  ; MATRIX ADDRESSES
00331 000700 P:           700
00332 001100 Q:           1100
00333 001300 TEMP1:       1300
00334 001500 TEMP2:       1500
00335 001700 TEMP3:       1700
00336 002100 H:           2100
00337 002120 C:           2120
00340 002140 B:           2140
00341 002160 CSQU:        2160
      000342              .LOC 342
```

```
0002  .MAIN
 00342 040420 CTHR:   040420
 00343 000000         0
       000344         .LOC 344
 00344 002700         BEGIN
 00345 003054         START
 00346 040420 V:      040420
 00347 000000         0
       000370         .LOC 370
 00370 002170 STR1:   2170              ; ADDRESSES OF STRINGS
 00371 002174 STR2:   2174
 00372 002200 STR3:   2200
 00373 002204 STR4:   2204
 00374 002210 STR5:   2210
 00375 002214 STR6:   2214
 00376 002220 STR7:   2220
 00377 002224 STR8:   2224

       002170         .LOC 2170         ; MESSAGE STRINGS IN ASCII
 02170 122040         122040            ; STRING 1: "R =    "
 02171 075040         075040
 02172 040040         040040
 02173 040040         040040
 02174 123040         123040            ; STRING 2: "S =    "
 02175 075040         075040
 02176 040040         040040
 02177 040040         040040
 02200 123101         123101            ; STRING 3: "SAMPLES?"
 02201 115120         115120
 02202 114105         114105
 02203 123077         123077
 02204 103117         103117            ; STRING 4: "COPY?   "
 02205 120131         120131
 02206 077040         077040
 02207 040040         040040
 02210 120101         120101            ; STRING 5: "PARAMS  "
 02211 122101         122101
 02212 115123         115123
 02213 040040         040040
 02214 120055         120055            ; STRING 6: "P-MATRIX"
 02215 115101         115101
 02216 124122         124122
 02217 111130         111130
 02220 122105         122105            ; STRING 7: "READY?  "
 02221 101104         101104
 02222 131077         131077
 02223 040040         040040
 02224 103124         103124            ; STRING 8: "CTHR, V:"
 02225 110122         110122
 02226 054040         054040
 02227 126072         126072


       002700         .LOC 2700         ; PROGRAMME BEGINS

 02700 006305 BEGIN:  FINI
 02701 030370         LDA 2, STR1
 02702 006362         JSR @WRITE        ; PRINT "R = ?"
 02703 006360         JSR @DIGIT        ; GET R
 02704 040312         STA 0, R
```

```
0003  .MAIN
 02705 030371          LDA 2, STR2
 02706 006362          JSR @WRITE      ; PRINT "S = ?"
 02707 006360          JSR @DIGIT      ; GET S
 02710 040313          STA 0, S
 02711 040325          STA 0, N
 02712 126400          SUB 1, 1
 02713 107000          ADD 0, 1
 02714 014325          DSZ N
 02715 000776          JMP .-2
 02716 044315          STA 1, SS       ; SS = SQUARE OF S
 02717 034312          LDA 3, R
 02720 054325          STA 3, N
 02721 126400          SUB 1, 1
 02722 107000          ADD 0, 1
 02723 014325          DSZ N
 02724 000776          JMP .-2
 02725 044314          STA 1, RS       ; LOC RS = PRODUCT OF R AND S
 02726 030372          LDA 2, STR3
 02727 006362          JSR @WRITE      ; PRINT "SAMPLES?"
 02730 102520          SUBZL 0, 0
 02731 126520          SUBZL 1, 1
 02732 030333          LDA 2, TEMP1
 02733 006357          JSR @DATRC      ; GET K0
 02734 030333          LDA 2, TEMP1
 02735 006004          FETR
 02736 075000          FFIX 0, 2
 02737 100000          FEXT
 02740 035001          LDA 3, 1, 2
 02741 054321          STA 3, K0       ; K0 = NO. OF RAPID SAMPLES
 02742 030373 COPY:    LDA 2, STR4
 02743 006362          JSR @WRITE      ; PRINT "COPY?"
 02744 006360          JSR @DIGIT      ; GET I
 02745 040316          STA 0, I
 02746 101005          MOV 0, 0, SNR   ; SKIP IF I = 0
 02747 000476          JMP READY       ; NO STARTING VALUES
 02750 040317          STA 0, J
 02751 014317 OPT1:    DSZ J           ; SAME STARTING VALUES
 02752 000402          JMP OPT2
 02753 000472          JMP READY
 02754 014317 OPT2:    DSZ J           ; PAPER-TAPE COPY
 02755 000407          JMP OPT3
 02756 024314          LDA 1, RS
 02757 030330          LDA 2, A
 02760 006356          JSR @DATPN
 02761 024315          LDA 1, SS
 02762 030331          LDA 2, P
 02763 006356          JSR @DATPN
 02764 014317 OPT3:    DSZ J           ; TELETYPE COPY
 02765 000415          JMP OPT4
 02766 030374          LDA 2, STR5
 02767 006362          JSR @WRITE
 02770 020313          LDA 0, S
 02771 024312          LDA 1, R
 02772 030330          LDA 2, A
 02773 006361          JSR @DATWR
 02774 030375          LDA 2, STR6
 02775 006362          JSR @WRITE
 02776 020313          LDA 0, S
 02777 024313          LDA 1, S
```

```
 03000 030331         LDA 2, P
 03001 006361         JSR @DATWR
 03002 014317 OPT4:   DSZ J           ; STARTING VALUES FROM TAPE
 03003 000427         JMP OPT5
 03004 024314         LDA 1, RS
 03005 030330         LDA 2, A
 03006 006355         JSR @DATRD
 03007 024315         LDA 1, SS
 03010 030331         LDA 2, P
 03011 006355         JSR @DATRD
 03012 014317 OPT5:   DSZ J           ; STARTING VALUES FROM KBD
 03013 000415         JMP OPT6
 03014 030374         LDA 2, STR5
 03015 006362         JSR @WRITE      ; PRINT "PARAMS"
 03016 020313         LDA 0, S
 03017 024312         LDA 1, R
 03020 030330         LDA 2, A
 03021 006357         JSR @DATRC
 03022 030375         LDA 2, STR6
 03023 006362         JSR @WRITE      ; PRINT "P-MATRIX"
 03024 020313         LDA 0, S
 03025 024313         LDA 1, S
 03026 030331         LDA 2, P
 03027 006357         JSR @DATRC
 03030 014317 OPT6:   DSZ J           ; STARTING VALUES SUPPLIED
 03031 000402         JMP .+2         ; BY USER SUBROUTINE
 03032 006363         JSR @INIT
 03033 014317 OPT7:   DSZ J           ; CHANGE CTHR AND V
 03034 000411         JMP READY
 03035 030377         LDA 2, STR8
 03036 006362         JSR @WRITE      ; PRINT "CTHR, V:"
 03037 006004         FETR
 03040 120000         FDFC 0          ; GET CTHR
 03041 124000         FDFC 1          ; GET V
 03042 040342         FSTA 0, CTHR    ; STORE CTHR
 03043 044346         FSTA 1, V       ; STORE V
 03044 100000         FEXT
 03045 030376 READY:  LDA 2, STR7
 03046 006362         JSR @WRITE      ; PRINT "READY?"
 03047 006360         JSR @DIGIT
 03050 101005         MOV 0, 0, SNR
 03051 000671         JMP COPY
 03052 002364         JMP @MEAS       ; USER PROGRAMME
 03053 003260         SEQ2
```

```
↑ 0005  .MAIN
  03054 006365 START:  JSR @DATIN        ; USER PROGRAMME
  03055 034316         LDA 3, I
  03056 175004         MOV 3, 3, SZR     ; SKIP IF I = 0
  03057 002774         JMP @START-1
  03060 034313         LDA 3, S
  03061 054317         STA 3, J          ; SET COUNTER: J = S
  03062 102400         SUB 0, 0          ; A, P, Q = 0
  03063 030333         LDA 2, TEMP1
  03064 034330         LDA 3, A
  03065 041400         STA 0, 0, 3
  03066 175400         INC 3, 3
  03067 172414         SUB# 3, 2, SZR
  03070 000775         JMP .-3
  03071 020332 LOOP:   LDA 0, Q          ; 1. QH
  03072 024336         LDA 1, H
  03073 030337         LDA 2, C
  03074 034313         LDA 3, S
  03075 054322         STA 3, L
  03076 054324         STA 3, M
  03077 034311         LDA 3, ONE
  03100 054325         STA 3, N
  03101 006352         JSR @MXMPY
  03102 020336         LDA 0, H          ; 2. C = H - QH
  03103 024337         LDA 1, C
  03104 030337         LDA 2, C
  03105 034313         LDA 3, S
  03106 054325         STA 3, N
  03107 006351         JSR @MXSUB
  03110 020337         LDA 0, C          ; 3. C'C
  03111 024337         LDA 1, C
  03112 030341         LDA 2, CSQU
  03113 034313         LDA 3, S
  03114 054322         STA 3, L
  03115 034311         LDA 3, ONE
  03116 054324         STA 3, M
  03117 054325         STA 3, N
  03120 006352         JSR @MXMPY
  03121 006004         FETR              ; 4. C TOO SMALL?
  03122 020342         FLDA 0, CTHR
  03123 026341         FLDA 1, @CSQU
  03124 106406         FSUB 0, 1, FSLE
  03125 000403         FJMP .+3
  03126 100000         FEXT
  03127 000725         JMP START
  03130 100000         FEXT
  03131 020337 SEQ1:   LDA 0, C          ; 5. B = C/C'C
  03132 024341         LDA 1, CSQU
  03133 030340         LDA 2, B
  03134 034313         LDA 3, S
  03135 054325         STA 3, N
  03136 006353         JSR @MXDIV
  03137 020336         LDA 0, H          ; 6. HB'
  03140 024340         LDA 1, B
  03141 030333         LDA 2, TEMP1
  03142 034311         LDA 3, ONE
  03143 054322         STA 3, L
  03144 034313         LDA 3, S
  03145 054324         STA 3, M
  03146 054325         STA 3, N
```

```
03147 006352        JSR @MXMPY
03150 020331        LDA 0, P         ; 7. PHB'
03151 024333        LDA 1, TEMP1
03152 030334        LDA 2, TEMP2
03153 034313        LDA 3, S
03154 054322        STA 3, L
03155 054324        STA 3, M
03156 054325        STA 3, N
03157 006352        JSR @MXMPY
03160 020331        LDA 0, P         ; 8. P - PHB'
03161 024334        LDA 1, TEMP2
03162 030331        LDA 2, P
03163 034315        LDA 3, SS
03164 054325        STA 3, N
03165 006351        JSR @MXSUB
03166 024313        LDA 1, S         ; 9. (BH'P)(HB')
03167 030334        LDA 2, TEMP2
03170 006354        JSR @MXTR
03171 020334        LDA 0, TEMP2
03172 024333        LDA 1, TEMP1
03173 030335        LDA 2, TEMP3
03174 034313        LDA 3, S
03175 054322        STA 3, L
03176 054324        STA 3, M
03177 054325        STA 3, N
03200 006352        JSR @MXMPY
03201 020340        LDA 0, B         ; 10. BB'
03202 024340        LDA 1, B
03203 030333        LDA 2, TEMP1
03204 034311        LDA 3, ONE
03205 054322        STA 3, L
03206 034313        LDA 3, S
03207 054324        STA 3, M
03210 054325        STA 3, N
03211 006352        JSR @MXMPY
03212 020331        LDA 0, P         ; 11. (P - PHB') - BH'P
03213 024334        LDA 1, TEMP2
03214 030331        LDA 2, P
03215 034315        LDA 3, SS
03216 054325        STA 3, N
03217 006351        JSR @MXSUB
03220 020331        LDA 0, P         ; 12. (P-PHB'-BH'P)+BH'PHB'
03221 024335        LDA 1, TEMP3
03222 030331        LDA 2, P
03223 034315        LDA 3, SS
03224 054325        STA 3, N
03225 006350        JSR @MXADD
03226 020331        LDA 0, P         ; 13. P=(P-PHB'-BH'P
03227 024333        LDA 1, TEMP1     ;        +BH'PHB')+BB'
03230 030331        LDA 2, P
03231 034315        LDA 3, SS
03232 054325        STA 3, N
03233 006350        JSR @MXADD
03234 020340        LDA 0, B         ; 14. BC'
03235 024337        LDA 1, C
03236 030333        LDA 2, TEMP1
03237 034311        LDA 3, ONE
03240 054322        STA 3, L
03241 034313        LDA 3, S
```

```
03242 054324          STA 3, M
03243 054325          STA 3, N
03244 006352          JSR @MXMPY
03245 020332          LDA 0, Q          ; 15. Q = Q + BC'
03246 024333          LDA 1, TEMP1
03247 030332          LDA 2, Q
03250 034315          LDA 3, SS
03251 054325          STA 3, N
03252 006350          JSR @MXADD
03253 004465          JSR EST
03254 014317          DSZ J
03255 000614          JMP LOOP
03256 034311          LDA 3, ONE
03257 054316          STA 3, I
03260 020331 SEQ2:    LDA 0, P          ; 16. PH
03261 024336          LDA 1, H
03262 030333          LDA 2, TEMP1
03263 034313          LDA 3, S
03264 054322          STA 3, L
03265 054324          STA 3, M
03266 034311          LDA 3, ONE
03267 054325          STA 3, N
03270 006352          JSR @MXMPY
03271 020336          LDA 0, H          ; 17. H'PH
03272 024333          LDA 1, TEMP1
03273 030334          LDA 2, TEMP2
03274 034313          LDA 3, S
03275 054322          STA 3, L
03276 034311          LDA 3, ONE
03277 054324          STA 3, M
03300 054325          STA 3, N
03301 006352          JSR @MXMPY
03302 006004          FETR              ; 18. V + H'PH
03303 020346          FLDA 0, V
03304 026334          FLDA 1, @TEMP2
03305 107000          FADD 0, 1
03306 046334          FSTA 1, @TEMP2
03307 100000          FEXT
03310 020333          LDA 0, TEMP1      ; 19. B = PH/(V+H'PH)
03311 024334          LDA 1, TEMP2
03312 030340          LDA 2, B
03313 034313          LDA 3, S
03314 054325          STA 3, N
03315 006353          JSR @MXDIV
03316 020333          LDA 0, TEMP1      ; 20. (PH)B'
03317 024340          LDA 1, B
03320 030335          LDA 2, TEMP3
03321 034311          LDA 3, ONE
03322 054322          STA 3, L
03323 034313          LDA 3, S
03324 054324          STA 3, M
03325 054325          STA 3, N
03326 006352          JSR @MXMPY
03327 020331          LDA 0, P          ; 21. P = P - PHB'
03330 024335          LDA 1, TEMP3
03331 030331          LDA 2, P
03332 034315          LDA 3, SS
03333 054325          STA 3, N
03334 006351          JSR @MXSUB
```

```
0008  .MAIN
 03335 004403          JSR EST
 03336 002722          JMP SEQ2
 03337 000000          0
 03340 054777 EST:     STA 3, .-1
 03341 022336          LDA 0, H           ; 22. H'A
 03342 024330          LDA 1, A
 03343 030334          LDA 2, TEMP2
 03344 034313          LDA 3, S
 03345 054322          STA 3, L
 03346 034311          LDA 3, ONE
 03347 054324          STA 3, M
 03350 034312          LDA 3, R
 03351 054325          STA 3, N
 03352 006352          JSR @MXMPY
 03353 006365          JSR @DATIN         ; 23. Z' - H'A
 03354 020336          LDA 0, H
 03355 024334          LDA 1, TEMP2
 03356 030335          LDA 2, TEMP3
 03357 034312          LDA 3, R
 03360 054325          STA 3, N
 03361 006351          JSR @MXSUB
 03362 020340          LDA 0, B           ; 24. B(Z'-H'A)
 03363 024335          LDA 1, TEMP3
 03364 030333          LDA 2, TEMP1
 03365 034311          LDA 3, ONE
 03366 054322          STA 3, L
 03367 034313          LDA 3, S
 03370 054324          STA 3, M
 03371 034312          LDA 3, R
 03372 054325          STA 3, N
 03373 006352          JSR @MXMPY
 03374 020330          LDA 0, A           ; 25. A=A+B(Z'-H'A)
 03375 024333          LDA 1, TEMP1
 03376 030330          LDA 2, A
 03377 034314          LDA 3, RS
 03400 054325          STA 3, N
 03401 006350          JSR @MXADD
 03402 006366          JSR @DTOUT         ; USER PROGRAMME
 03403 002734          JMP @EST-1

       002700          .END 2700
```

```
---
  ØØØ9   .MAIN
A        ØØØ33Ø
B        ØØØ34Ø
BEGIN    ØØ27ØØ
C        ØØØ337
COPY     ØØ2742
CSQU     ØØØ341
CTHR     ØØØ342
DATIN    ØØØ365
DATPN    ØØØ356
DATRC    ØØØ357
DATRD    ØØØ355
DATWR    ØØØ361
DIGIT    ØØØ36Ø
DTOUT    ØØØ366
EST      ØØ3342
H        ØØØ336
I        ØØØ316
INIT     ØØØ363
J        ØØØ317
KØ       ØØØ321
L        ØØØ322
LOOP     ØØ3371
M        ØØØ324
MEAS     ØØØ364
MXADD    ØØØ35Ø
MXDIV    ØØØ353
MXMPY    ØØØ352
MXSUB    ØØØ351
MXTR     ØØØ354
N        ØØØ325
ONE      ØØØ311
OPT1     ØØ2751
OPT2     ØØ2754
OPT3     ØØ2764
OPT4     ØØ3ØØ2
OPT5     ØØ3Ø12
OPT6     ØØ3Ø3Ø
OPT7     ØØ3Ø33
P        ØØØ331
Q        ØØØ332
R        ØØØ312
READY    ØØ3Ø45
RS       ØØØ314
S        ØØØ313
SEQ1     ØØ3131
SEQ2     ØØ326Ø
SS       ØØØ315
START    ØØ3Ø54
STR1     ØØØ37Ø
STR2     ØØØ371
STR3     ØØØ372
STR4     ØØØ373
STR5     ØØØ374
STR6     ØØØ375
STR7     ØØØ376
STR8     ØØØ377
TEMP1    ØØØ333
TEMP2    ØØØ334
TEMP3    ØØØ335
```

```
 0010  .MAIN
V      000346
WRITE  000362
```