measurement systems laboratory

massachusetts institute of technology. cambridge. massachusetts 02139

TE-54

An Analysis of the Accuracy of a Parameter Optimization Technique

by Yoram Baram

January, 1974

TE-54

AN ANALYSIS OF THE ACCURACY OF A PARAMETER OPTIMIZATION TECHNIQUE

by

Yoram Baram

APPROVED: W. Markey

Director
Measurement Systems Laboratory

AN ANALYSIS OF THE ACCURACY OF A PARAMETER OPTIMIZATION TECHNIQUE

by

Yoram Baram

B.S., Technion, Israel Institute of Technology
(1972)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
January, 1974

Signature of Author Y. Baram
Department of Aeronautics
and Astronautics, January, 1974

Certified by
Thesis Supervisor

Accepted by
Chairman, Departmental

AN ANALYSIS OF THE ACCURACY OF A PARAMETER OPTIMIZATION TECHNIQUE

By

Yoram Baram

Submitted to the Department of Aeronautics and Astronautics in January 1974 in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

The design of high order control systems by parameter optimization usually involves a great deal of computation. Round-off errors in machine computation may lead to inaccurate results that in many cases may prevent the continuation of the optimization procedure. Some of the numerical operations involved in a currently used optimization technique are discussed and analyzed with special attention to the numerical accuracy. Alternative methods are suggested for more accurate and cost effective solutions.

Thesis Supervisor: H. Philip Whitaker

Title: Professor of Aeronautics and Astronautics

ACKNOWLEDGEMENTS

The author would like to express his sincere thanks to Professor H. Philip Whitaker who,as thesis supervisor, provided motivation and continual constructive criticism throughout this thesis effort.

The author would also like to take this oppurtunity to thank his wife, Susan, for her advice, encouragement and patience during the course of his studies.

This thesis was prepared under DSR project 73598 sponsored by the Edwards Flight Research Center of the National Aeronautics and Space Administration through Grant NGL 22-009-207.

TABLE OF CONTENTS

LIST OF SYMBOLS

Symbol	Definition
A	System coefficient matrix
a,b,c	System transfer function coefficients
D	Difference matrix
E_i	i'th order system coefficient matrix
F,G,H,J	System state matrices
f	Scaling factor
G(s)	Transfer function
I	Identity matrix
k	Model numerator order
k_j	Constant in sytem polynomial coefficient
ℓ	Model order or number of design parameters
$\underline{\ell}$,$\underline{m}$	Laplace variable polynomial vectors
$\underline{\ell}^*$	Transfer function vector
m	Order of transfer function numerator
n	Order of system
P	Design parameter
PI	Performance index
Q,W	Weighting matrices
s	Laplace variable
u	System input
X	System state matrix
X_0	System imitial condition matrix

$\underline{x}$	System state vector
$\underline{x}_0$	System initial state vector
$\underline{x}_i$	i'th column of X matrix
$\underline{x}_{0_i}$	i'th column of X_0 matrix
y	System output
α, β	Model transfer function coefficients

CHAPTER 1

BACKGROUND AND PROBLEM IDENTIFICATION

1.1 The Design of a Control System by Parameter Optimization

The design problem that raised the need for this work is the one of finding the optimal set of prespecified parameters of a fixed configuration control system. A criterion for the design can be derived by some evaluation of the system response to some standard input. If the system's transient response to a step input is weighted by some criterion and integrated along all time, a performance index can be defined, the minimization of which gives the optimal design.

In the case of a linear system, whose state vector satisfies the equation:

$$\dot{\underline{x}}(t) = A\underline{x}(t) \qquad \underline{x}(0) = \underline{x}_0 \tag{1.1}$$

such a performance index can be given the quadratic form:

$$PI = \underline{x}_0^T \, w \, \underline{x}_0 + \int_0^\infty \underline{x}^T \, Q \, \underline{x} \, dt \tag{1.2}$$

where $\underline{x}$ is the transient state vector, $\underline{x}_0$ is the initial transient state vector and W and Q are positive semi-definite weighting matrices.

The integral expression in (1.2) can be presented more conveniently by:

$$\int_0^\infty \underline{x}^T Q\underline{x}\, dt = tr\ [Q \cdot X] \qquad (1.3)$$

where tr denotes the trace of the matrix, and where:

$$X = \int_0^\infty \underline{x}\, \underline{x}^T\, dt \qquad (1.4)$$

Using equation (1.1):

$$\frac{d}{dt}(\underline{x}\, \underline{x}^T) = \underline{x}\, \dot{\underline{x}}^T + \dot{\underline{x}}\, \underline{x}^T = \underline{x}\, \underline{x}^T A^T + A\underline{x}\, \underline{x}^T \qquad (1.5)$$

and integrating both sides along all time for a time invariant system gives:

$$\underline{x}\, \underline{x}^T\ (t=\infty) - \underline{x}\, \underline{x}^T\ (t=0) = XA^T + AX$$

For a stable linear system, the transient response vanishes as $t \to \infty$, and we get that the state matrix X must satisfy the equation:

$$AX + XA^T = -X_0 \qquad (1.6)$$

where:

$$X_0 = \underline{x}_0\, \underline{x}_0^T \qquad (1.7)$$

1.2 The Model Performance Index

The Model Performance Index was first suggested by Rediess [11] and was improved by Palsson [10] for application in an automatic optimization algorithm. The theory is developed in these two references and will not be described here. A brief outline of some results, applicable in an optimization procedure is necessary, however, for the under-

standing of some computation operations analysed in the following chapter.

It is desired to approximate the step input response of a system whose transfer function is:

$$G_s(s) = \frac{b_m s^m + \ldots\ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0} \qquad (1.8)$$

to that of a model whose transfer function is:

$$G_m(s) = \frac{\beta_k s^k + \ldots\ldots + \beta_1 s + \beta_0}{s^\ell + \alpha_{\ell-1}s^{\ell-1} + \ldots + \alpha_1 s + \alpha_0} \qquad (1.9)$$

The model is required to satisfy the condition:

$\ell - k \leq n - m$

A new transfer function that has no zeros, and whose denominator is the cascaded system denominator and model numerator, is formed:

$$G(s) = \frac{a_0 \beta_0}{(\beta_k s^k + \ldots\ldots + \beta_1 s + \beta_0)(s^n + a_{n-1}s^{n-1} + \ldots\ldots + a_1 s + a_0)} \qquad (1.10)$$

with the initial conditions:

$$x_{1_0} = \frac{b_0}{a_0}$$

$$x_{(i+1)_0} = 0 \qquad 1 \leq i < n-m$$

$$x_{(i+1)_0} = b_{n-i} - \sum_{j=n-m}^{i-1} a_{n-i+j}\, x_{(j+1)_0} \qquad n-m \leq i < n \qquad (1.11)$$

The Model Performance Index is defined as:

$$PI = \int_0^\infty \underline{x}^T Q \underline{x} \, dt \tag{1.12}$$

where:

$$Q = \frac{\hat{\alpha}\,\hat{\alpha}^T}{\alpha_0 \alpha_0} \tag{1.13}$$

$\hat{\alpha}$ is the model characteristic coefficients vector presented in the n'th dimension state space:

$$\hat{\alpha}^T = [\alpha_0, \ldots \alpha_{\ell-1}, 1, 0, 0 \ldots 0] \tag{1.14}$$

$\underline{x}$ in the equation (1.12) is the transient state vector of the system represented by equation (1.10) in response to the initial conditions(1.11). By use of expressions derived in the previous section, the performance index can be expressed as:

$$PI = tr[Q \cdot X] \tag{1.15}$$

X must satisfy the matrix equation

$$AX + XA^T = -X_0 \tag{1.6}$$

where:

$$X_0 = \underline{x}_0 \underline{x}_0^T \tag{1.7}$$

1.3 The Solution of the State Matrix X

The solution of equation (1.6) for the matrix X is a major concern of this thesis. Palsson [10] has shown that when the system matrix A is written in the phase variable form:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0.....0 \\ 0 & 0 & 1 & 0.....0 \\ \vdots & & & \\ -a_0 & -a_1 & \cdots\cdots\cdots & a_{n-1} \end{bmatrix} \qquad (1.16)$$

the first column of the state matrix X, $\underline{x}_1$, can be computed by:

$$\underline{x}_1 = -E_n^{-1} \sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}} \qquad (1.17)$$

where $\underline{x}_{0_{n-i}}$ is the (n-i)'th column of the matrix X_0 and the matrices E_i are defined by the relationship:

$$E_i = -AE_{i-1} + a_{n-i} I \qquad 1 \leq i \leq n \qquad (1.18)$$

with

$$E_0 = I$$

The remaining columns of the matrix X are computed by the relationship:

$$\underline{x}_i = -A\underline{x}_{i-1} + \underline{x}_{0_i} \qquad (1.19)$$

The computation of these expressions is discussed in detail in Chapter 2.

1.4 The System Transfer Function Coefficients

A method for obtaining the transfer function of a comlicated (multi-loop with a multi-input multi-output controlled member) system given by its components was developed by Whitaker [13], Griffen [4] and Beyers [1]. It consists of

the following generalized steps:

1. Obtaining the system state equations in the form:

$$\dot{\underline{x}} = F\underline{x} + G\underline{u}$$

$$\underline{y} = H\underline{x} + J\underline{u} \qquad (1.20)$$

where $\underline{x}$ is the state vector, $\underline{u}$ is the input vector, $\underline{y}$ is the output vector and F, G, H, and J are properly dimensional matrices.

2. Obtaining the transfer function matrix by:

$$\underline{y} = [H[sI-F]^{-1}G + J]\ \underline{u} \qquad (1.21)$$

This method is used to obtain the system coefficients which make up the matrices A and X_0 in equation (1.6). Although the accuracy of the computation at this stage is not analysed in detail in this work, it should be pointed out that the large number of arithmetic operations done here especially for high order systems, has been found to be a source of significant numerical errors. The accuracy of the computed performance index cannot be any better than that introduced in the matrices A and X_0.

The error in the system coefficients may affect the minimum search procedure mainly in two ways:

1. When the performance index first increases in a step, it is normally assumed that the minimum point lies somewhere along this step. In the presence of significant inaccuracies such an interpretation of the performance index increase may be false. Taking a shorter step

in this case may even give further increase of the performance index (this problem is later referred to as the "performance index increase in half a step").

2. In the current optimization program used at the Measurement Systems Laboratory at M.I.T. and applying the methods discussed in this chapter, the gradient method is used for the search of the optimum. The derivatives of the system coefficients with respect to the design parameters are used to produce the gradient at each step. These derivatives, which are computed by incrementation of the parameters and division of the corresponding changes in the coefficients by the incremented parameters values, are very sensitive to the accuracy in the computed parameters. In other words, if the error in the computed coefficients is of the order of the expected incremental change in the coefficients, then the resulting derivatives and gradient values would be completely incorrect.

These two problems are considered again in Chapter 4 where an alternative method is discussed.

1.5 Scaling the System Coefficients

A large number of operations involving matrices and vectors take place in the solution of equation (1.6).In some cases significant inaccuracies may occur when elements appear in certain arrangements of magnitude in matrices and vectors.

These cases are discussed in Chapter 2. Scaling these matrices and vectors so as to rearrange the relative magnitudes of the elements may improve the results significantly. Since all the elements in equation (1.6) consist of system coefficients, the desired scaling of matrices and vectors can be obtained in most cases by scaling the system coefficients.

One way of scaling the coefficients is changing the time scale. Since the units of the coefficients are powers of time, changing the time scale can change the magnitude of the coefficients in the desirable way. This is illustrated in the following:

Characteristic coefficients before scaling:

$$a_0, a_1 \quad \ldots . a_{n-1}, 1$$

Characteristic coefficients after time scaling:

$$f^n a_0, f^{n-1} a_1 \ldots . f a_{n-1}, 1$$

where f is the scaling factor. By selecting a value for f, the coefficients can be ordered in an ascending or a descending order of magnitudes as desired. This type of scaling will be referred to later as time scaling.

Another way of scaling the system coefficients is adding a pair of a pole and a zero of the same value (or at the same location in the complex plane) which, of course, does not change the characteristics of the control system itself. The relations between the characteristic coefficients

and the poles are as follows:

a_0 = product of all poles

a_1 = sum of all possible products of n-1 out of n poles

a_2 = sum of all possible products of n-2 out of n poles

.

.

.

a_{n-1}=sum of all poles

It follows that adding a pole-zero pair near the origin, for example, would decrease a_0, not affecting a_{n-1} significantly, since:

$$a_0 = p_1 \cdot p_2 \cdots\cdots\cdots p_n \cdot \varepsilon$$

and

$$a_{n-1} = p_1 + p_2 + \cdots\cdots\cdots p_n + \varepsilon$$

where $p_1 \ldots p_n$ are the original system poles, and ε is the additional scaling pole (assuming that ε is not of the magnitude order of the other poles). There is freedom in the location of the pole-zero pair and the corresponding effect on the coefficients. This type of scaling will be referred to later as pole-zero scaling. There are some operations in the solution of equation (1.6) (as discussed in sections 2.2 and 2.3) where time scaling of the coefficients cannot improve the accuracy, but pole-zero scaling may be used effectively for this purpose. The disadvantage of the pole-zero scaling is that it increases the order of the system. The

trade-off between the positive effect of scaling and the negative effect of increased order on the accuracy must then be considered.

1.6 Severe Inaccuracies Occurance

There were some typical cases in which severe inaccuracies occurred, preventing the completion of the optimum search procedure. It is impossible, however, to point out one general reason for the occurance of the problem. Severe inaccuracies result in phenomena like negative performance index values, the "performance index increase in half a step", large values of performance index when small ones are expected, and other results that just do not make sense.

The problem has occurred in cases where the complex frequency range of the system modes of response was large, or in other words, the system poles were far apart. One such example is the presence of the phugoid mode in the equations of a controlled aircraft. This mode has relatively low frequency and small magnitude. When the "short period approximation" was exercised and the phugoid mode was eliminated, the severe inaccuracies in the computation disappeared. The addition of a remote pole to a system may also cause the appearance of severe inaccurracies in the computation. In such cases the problem can be avoided by elimination of low residue modes. But this is not always the case. In many situations low residue modes are not easy to identify and to isolate,

especially in the process of optimization, and it is not always clear that such elimination would solve the inaccuracy problem. Thus, rather than treating the different cases separately, the approach to the problem was by investigation of the numerical operations trying to identify the potential sources of inaccuracies and to suggest adequate solutions.

CHAPTER 2

THE NUMERICAL COMPUTATION

2.1 Introduction

The method proposed by Palsson for the solution of the matrix equation (1.6) was presented in section 1.3. In this chapter each step of the solution procedure is discussed in terms of the explicit algebraic expressions and the arithmetic operations, with special attention to the numerical accuracy obtained in these operations. Some parts of the computation are discussed only for the purpose of shedding some light on the various stages of the procedure, so that sources of significant inaccuracies that may occur can be spotted more easily.

2.2 The Computation of the Matrices E_i $1 \leq i \leq n$

The matrices E_i are obtained by the relationship:

$$E_i = -AE_{i-1} + A_{n-1}I \qquad 1 \leq i \leq n \tag{1.18}$$

with

$$E_0 = I$$

This part of the computation can be best investigated by considering the algebraic expressions of the elements of these matrices. The first few matrices are given in Appendix A so

that the nature of these and of higher order E_i matrices can be understood.

From the explicit expression of the E_i matrices, it can be learned that the elements at lower positions in the columns contain higher powers of the system characteristic coefficients. The arrangement of the coefficients in the matrices elements may affect the accuracy of the computation in two ways:

1. inaccurate computation of the matrices elements;
2. ill conditioning of the matrix E_n with respect to its inversion in a later stage of the procedure.

To understand the first possibility it should be realized that in the computation of low position elements of the E_i matrices, small and large numbers are normally added together with considerable loss of low position digits. Notice, for example, that the element:

$$E_5(5,5) = 2a_{n-5} + 2a_{n-3}a_{n-2} - 2a_{n-4}a_{n-1} + 2a_{n-2}^2 a_{n-1} + 6a_{n-3}a_{n-1}^2 - 6a_{n-2}a_{n-1}^3 + 2a_{n-1}^5$$

would be normally dominated by the higher powers of a_{n-1}, unless some scaling of the coefficients is done. Notice that time scaling of the system coefficients would have no effect on the accuracy of this computation, since such scaling is equivilant to multiplying the matrix element by a factor, leaving the relative magnitudes of the addents unchanged.

Using a scaling factor f gives, for example:

$$E_{5_{scaled}}(5,5) = 2(f^5 a_{n-5}) + 2(f^3 a_{n-3} f^2 q_{n-2}) - 2(f^4 a_{n-4} f a_{n-1}) + 2((f^2 a_{n-2})^2 f a_{n-1}) + 6(f^3 a_{n-3}(f a_{n-1})^2) - 6(f^2 a_{n-2}(f a_{n-1})^3) + 2(f a_{n-1})^5$$

$$= f^5 E_{5_{unscaled}}(5,5)$$

Pole-zero scaling can be used effectively to control the accuracy of this computation. Such scaling, as was discussed in Section 1.5, can be used to magnify the value of a_0 and other low order coefficients with respect to a_{n-1} without changing the value of a_{n-1} significantly. Scaling would be done to make the addents close enough in magnitude so that loss of low position digits is minimized.

The inversion of the matrix E_n is discussed later in this chapter. It is pointed out that ill conditioning of a matrix with respect to its inversion occurs when the elements on the diagonal are significantly smaller than elements to their right in rows. It is evident from the E_i matrices (consider E_5, for example) that such situations can be controlled by either time scaling or pole-zero scaling of the system coefficients. A situation where elements on the right of the rows are larger than those on the left can be reversed by such scaling.

One may argue that changing the order of the operations in the computation of the E_i matrices so as to sum small numbers first and then add the larger numbers, minimizing loss

of digits, would be useful. It can be verified, however, that in general these operations are indeed done in such order, as each E_i matrix is computed from the previous one.

2.3 The Computation of $\sum_{i=1}^{n} E_{i-1}\underline{x}_{0_{n-i}}$

$\underline{x}_{0_{n-i}}$ in the expression $\sum_{i=1}^{n} E_{i-1}\underline{x}_{0_{n-i}}$ is the (n-i)'th column of the initial conditions matrix:

$$X_0 = \underline{x}_0\underline{x}_0^T = \begin{bmatrix} x_{0_1}x_{0_1} & x_{0_1}x_{0_2} & \cdots & x_{0_1}x_{0_n} \\ x_{0_2}x_{0_1} & & & \\ \vdots & & & \\ x_{0_n}x_{0_1} & x_{0_n}x_{0_2} & \cdots & x_{0_n}x_{0_n} \end{bmatrix} \quad (2.1)$$

The initial conditions are computed by equations (1.11) as explicit functions of the system coefficients. Consider the case: n=8, m=5, ℓ=3, then:

$$x_{0_1} = -\frac{b_0}{a_0}$$

$$x_{0_2} = 0$$

$$x_{0_3} = 0$$

$$x_{0_4} = b_5$$

$$x_{0_5} = b_4$$

$$x_{0_6} = b_3 - a_6b_5 - a_7b_4$$

$$x_{0_7} = b_2 - a_5b_5 - a_6b_4 - a_7b_3 + a_7a_6b_5 + a_7a_7b_4$$

$$x_{0_8} = b_1 - a_4b_5 - a_5b_4 - a_6b_3 + a_6a_6b_5 + a_6a_7b_4$$
$$- a_7b_2 + a_7a_5b_5 + a_7a_6b_4 + a_7a_7b_3 - a_7a_7a_6b_5$$
$$- a_7a_7a_7b_4 \qquad (2.2)$$

The higher order initial conditions involve higher order products of the system's coefficients, and normally they are significantly larger than the lower order ones. Computation of high order initial conditions involves a great deal of multiplications and additions, and single precision is inadequate if accurate results are desired. Since each one of the initial conditions depends linearly on one b coefficient, it can be argued that inaccurate computation would result in initial conditions corresponding to another set of b coefficients, or, in other words, to another system. This may cause problems like the "performance index increase in half a step", and inaccurate computation of the gradient of the performance index with respect to the design parameters, that were discussed in Chapter 1. The accuracy of x_{0_1} is of special importance. Since in the optimization algorithm a system static sensitivity of 1 is assumed, the value of x_{0_1} must be 1. Even a slight error in this value would result in a steady state difference between the model's and the system's responses, which, at least theoretically, should

produce an infinite performance index. This point was not considered in the original program where the value of x_{0_1} was computed in 3 steps, which in single precision floating point arithmetic resulted in 2 incorrect digits out of 8. This error has lead to "awkward" values of performance index. When the value of x_{0_1}, was fixed at -1 (instead of the unnecessary computation) sufficiently accurate results were obtained.

The error in x_{0_1} had a major effect on the computed performance index in the case of one system, but had no significant effect in the case of another system of the same order. This is explained by the "weighting" of the elements x_{0_i} x_{0_j} of the X_0 matrix by the E_i matrices, first in the sum $\sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}}$ and then in the product $E_n^{-1} \sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}}$. As was shown in 2.2 the lower order E_i matrices contain many zero elements. It is easy to realize that in the vector $\sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}}$ the first element is mostly affected by products of x_{0_1}. Elements at the lower positions in the vector are less affected by x_{0_i}. This effect depends, of course, on the relative magnitudes of the elements of E_i, or the weighting imposed by these matrices. Then, in the product $E_n^{-1} \sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}}$, the contribution of the elements of the vector $\sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}}$ depends on the weighting by E^{-1}. When the first elements in the first row of E^{-1} are considerably

larger than other elements, the error in x_0 is weighted most heavily in the first element of the product vector x_1. This indeed was the case when the problem was first encountered. Fixing the value of x_{0_1} at -1 resulted in a significant change in X(1,1) ($\underline{x}_1(1)$), the other elements remaining practically unchanged, which in turn gave the desired correction in the computed performance index.

The algebraic expressions of the elements of $E_{i-1}\underline{X}_{0_{n-i}}$ are extensive and giving them here in detail seems unnecessary (besides, they can be easily obtained from the given E_i and X_0 matrices). It should be noted, however, that this part of the computation involves a great deal of arithmetic operations and may be a source of severe inaccuracies. If these inaccuracies are found to influence the computation significantly, rearranging the operations so that small and large numbers are summed in an optimal order (to minimize loss of low-position digits) should improve the results. If this is done in the computation of the product vectors $E_{i-1}\underline{X}_{0_{n-i}}$ only, the additional storage required is negligable (n words), but the additional check operations may be time consuming. The alternative of extended precision computation has the disadvantage of having to use a different programming language (see 2.4) if precision higher than double is desired. The cost of interfacing languages in a program is considerably high, and the trade-off between the different possibilities must be considered.

Considering the corresponding expressions in the matrices E_i and X_0, it can be verified that time scaling of the system's coefficients is equivilant to multiplying the results by a factor, and thus, has no effect on the accuracy in the computation of the vectors $E_{i-1}\underline{x}_{0_{n-i}}$ or of the sum $\sum_{i=1}^{n} E_{i-1}\ \underline{x}_{0_{n-i}}$ (see a verification of a similar argument in 2.2). Pole-zero scaling can be used effectively to arrange this computation so that small numbers are summed first and then added to the larger numbers. This can be done by placing the pole-zero pair on the left half of the real axis far from the origin which would make a_0 appreciably smaller than a_{n-1}. Notice, however, that such scaling may have an undesirable effect on the E matrix, increasing the elements to the right of the diagonal in the rows. As is discussed in section 2.4 this may "ill condition" the matrix with respect to inversion.

2.4 The Computation of $\underline{x}_1$ and Matrix Inversion

$\underline{x}_1$ is the first column of the state matrix X. It satisfies the equation:

$$E_n\underline{x}_1 = \sum_{i=1}^{n} E_{i-1}\ \underline{x}_{0_{n-i}} \qquad (2.3)$$

or

$$\underline{x}_1 = E_n^{-1} \sum_{i=1}^{n} E_{i-1}\ \underline{x}_{0_{n-i}} \qquad (1.17)$$

This section is mostly concerned with the numerical errors

arising in the inversion of the matrix E_n for the solution of $\underline{x}_1$.

A popular method for machine inversion of matrices is the Gauss-Jordan method. This method is described in detail in references [5], [6], [7], and [12] and will not be presented here. It is , however, the matrix inversion method referred to in the following discussion.

A significant loss of accuracy may occur when elements on the diagonal of the matrix to be inverted are considerably smaller than elements to their right in the rows. Such situations may be avoided by reordering the rows of the matrix so as to place the larger elements on the diagonal. Such reordering may require extensive check and bookkeeping operations for high order matrix inversion. In the case of the E_n matrix such a solution is inadequate in general, since elements in all the rows appear in a similar arrangement of relative magnitudes (see Appendix A).

The usefulness of scaling a matrix by a linear transformation before scaling is questionable. The scaling and rescaling transformations may cause additional errors. Multiplying the matrix elements by powers of 10 when floating point arithmetic is used may result in a matrix that is isomorphic to the original. Indeed, attempts to use such scaling by a transformation matrix did not improve the inversion accuracy.

Scaling the system characteristic coefficients is, as

discussed in section 2.2, a useful tool for initially arranging the elements of the E_n matrix in a desirable order of magnitudes. This is illustrated in an example case in Chapter 3.

The situation of diagonal elements that are considerably smaller than elements to their right may arise at a later stage of the inversion procedure. How the pre-conditioning of the matrix should be done to avoid such cases is not easy to determine, since the relation between the elements of the resulting matrix and the elements of the original matrix is not a simple one. One way to handle such situations is suggested by Hellerman [5, p.57]. It consists of using the largest element in the corresponding column rather than the diagonal as a pivot element, while recording the place of this element. Then, since the rows and the columns of the inverse matrix appear scrambled, rearranging them in the right order (the Gauss-Jordan method is referred to as the inversion method).

A number of measures and criteria for the inaccuracy in matrix inversion and linear equations solution procedures can be found in the literature. They all fail to give an exact measure of the inaccuracy (to indicate how many digits in a result number are incorrect) and, at best, give an indication that some significant inaccuracy has occured in the computation. One such check often used is obtained from the elements of the difference matrix:

$$D = A \cdot A^{-1} - I$$

In our case of interest this check is hardly satisfactory. An investigation has shown that when the inverse of E_n contains elements considerably different in order of magnitude, the small elements are computed with large errors. In the computation of the product $E_n^{-1} \sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}}$ the contribution of these elements is negligible, and even relatively large errors in them would have no effect on the final result. Yet, when the product $E_n E_n^{-1}$ is computed for the check, these elements are multiplied by large elements of the matrix E_n, and the errors in them may produce appreciably large elements in the check matrix D. On the other hand, errors in the large elements of E^{-1}, which can affect the computation of $\underline{x}_1$ significantly, may not be detected by the check at all. Thus, such checks can be used to obtain a general idea about the accuracy of the computation but not as a measure of it, especially not as a basis for comparison between numerical results.

Absolute large values of the matrix elements (not only their relative magnitude with respect to each other) have been found to affect the accuracy of the matrix inversion. Severe inaccuracies have occured in some practical cases when the matrix determinant was very large. When the elements of the matrix were divided by a certain factor before and after the inversion, accurate inversion was obtained in these cases.

A different approach to the solution of the vector $\underline{x}_1$

in the matrix equation:

$$E_n \underline{x}_1 = \sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}} \qquad (2.3)$$

is usually referred to as the indirect or the iterative approach. The Gauss-Seidel method is a popular iterative method used for machine solution of systems of linear equations. It is described along with some other such methods in references [6] and [7], where it is also pointed out that these methods may or may not converge on the solution depending on the numerical values involved, even in the absence of round-off errors.

If a sufficiently accurate computation of $\underline{x}_1$ cannot be obtained by scaling the E_n matrix, an extended precesion version of the matrix inversion algorithm may be used. Since IBM/370 FORTRAN does not have a built-in extended precision facility, the MINV program from the IBM PL/I Scientific Subroutine Package [8] has been modified to invert a matrix in quadruple precision. This program can be called by the FORTRAN optimization program whenever an extended precision inversion is desired. The modified program and the interface control cards are given in Appendix B.

2.5 The Remaining Columns of the Matrix X

After the first column of the state matrix X is computed, the remaining columns are computed by equation (1.19):

$$\underline{x}_i = -A\underline{x}_{i-1} + \underline{x}_{0_i}$$

e.g.

$$\underline{x}_2 = \begin{bmatrix} 0 & -1 & 0 & 0 & \dots & 0 \\ 0 & 0 & -1 & 0 & \dots & 0 \\ \vdots & & & & & \\ a_0 & a_1 & \dots\dots\dots & & & a_{n-1} \end{bmatrix} \underline{x}_1 + \begin{bmatrix} x_{0_1} x_{0_1} \\ x_{0_2} x_{0_1} \\ \vdots \\ x_{0_n} x_{0_1} \end{bmatrix}$$

Note that except for the last element in each column x_i all the elements are obtained by shifting the elements of the previous column upwards with an inverted sign and adding an element from the previously computed X_0 matrix. Both operations do not produce any numerical error. In Figure 2.1 the elements of an 8'th order state matrix X that are computed by shifting the elements of the first column and adding an element from the X_0 matrix are denoted by x, and elements that involve multiplication of the system's coefficients by a previous column are denoted by *. The computation of the * elements in this stage from previously computed values may add to their inaccuracy.

$$\begin{bmatrix} x & x & x & x & x & x & x & x & * \\ x & x & x & x & x & x & x & * & * \\ x & x & x & x & x & x & * & * & * \\ x & x & x & x & x & * & * & * & * \\ x & x & x & x & * & * & * & * & * \\ x & x & x & * & * & * & * & * & * \\ x & x & * & * & * & * & * & * & * \\ x & * & * & * & * & * & * & * & * \end{bmatrix}$$

Figure 2.1 - Additional Inaccuracy in the Computation of the Remaining Columns of an 8'th order X Mateix

In the case of the Model Performance Index only the elements

that are both in the first ℓ+1 rows and in the first ℓ+1 columns (ℓ being the model's order) take part in the computation of the performance index (e.g. in the case of an 8'th order system and a 3'rd order model only the elements in the 4 x 4 square in Figure 2.1 affect the result). It follows that for most practical cases this part of the computation does not introduce any numerical inaccuracy at all.

2.6 The Computation of the Performance Index

The performance index is computed by equation (1.15):

$$PI = tr\ [Q \cdot X]$$

where Q is the weighting matrix in the performance index and X is the state matrix whose computation was discussed before. Except for the case of very high order systems and when the correct value of PI is close to zero, this part of the computation should not introduce additional severe inaccuracy. If the expected value of PI is small, then accurate low-position digits must be obtained, and single precision may not be sufficient. Also when the order of the system is high, the large number of multiplications and additions may result in significant loss of digits. Note that in the case of the Model Performance Index the elements in the matrix Q that are both not in the first ℓ+1 rows and not in the first ℓ+1 columns are all zeros, and thus, the amount of computation in this stage depends only on the order of the model.

CHAPTER 3

AN EXAMPLE CASE

3.1 Introduction

Because of the nature of the problem the numerical example cannot be of a simple, low order system. A real inaccuracy problem encountered in a control system design is discussed in order to demonstrate some of the inaccuracy problems and their solutions as described in the previous chapters.

3.2 The Example System and the Inaccuracy Problem

The example control system is given by its mathematical block diagram in Figure 3.1. The numerical values are given in Figure 3.2(a) in the format of the System Description Program[12]. The rows of the SYS and the SIG matrices correspond to the blocks in the block diagram. The last two rows of these matrices represent the model zeros that are cascaded to the system to use the Model Performance Index. The controlled member state matrices and the design parameters location and initial values are also given. The computed values of the over-all transfer function characteristic coefficients (ACOF(I)), the numerator coefficients (BCOF(I)) and the initial condition vector are listed in Figure 3.2(b).

The transfer function coefficients are time scaled by a factor of two.

Significantly inaccurate computation was experienced in attempting to optimize this system. The numerical values of the matrix E_n, the determinant of this matrix and the state matrix X corresponding to the initial values of the design parameters are given in Figure 3.3 for comparison with other cases. The computed performance index in this case was 0.052 while the correct value is 0.26, as was verified by integration methods. Notice that the computed value of the first element of the matrix X is 0.55. The correct value of this element was found to be 0.75. The other elements of the matrix X are sufficiently accurate.

3.3 The Practical Solution of the Inaccuracy Problem

In order to obtain more accurate results in the computation of the example case the value of the first initial condition was fixed at -1, and since the value of the E_n matrix determinant was very large, all the matrix elements were devided by 10^5 before the matrix inversion. The resulting E_n and X matrices are given in Figure 3.4. The given elements of the E_n matrix are before reduction for comparison with other cases and the determinant was computed after reduction. As can be seen in the figure, the computed value of the first element in the matrix X is now 0.75 while the other elements are considerably closer to the ones before the correction.

The resulting performance index value was 0.26, differing by less than one percent from the value obtained by integration. Similar agreement between the performance index values computed by integration and the ones computed by the method discussed in this work was experienced in the following steps of the optimization. It is interesting to note that the elements at lower positions in the columns (or the rows) of the X matrix are closer to the correct values, which is in agreement with the error weighting effect, discussed in section 2.3

The correction of the initial condition has, of course, no effect on the matrix E_n. This correction affects the sum $\sum_{i=1}^{n} E_{i-1}\underline{x}_{0_{n-i}}$ while the reduction of the matrix E_n elements affects the matrix inversion, so that the first column of the matrix X which is equal to $E_n^{-1} \sum_{i=1}^{n} E_{i-1}\underline{x}_{0_{n-i}}$ is affected by both corrections. Indeed, it was found that both corrections were necessary in this case. When only x_{0_1} was corrected and the reduction of the E_n matrix elements was not exercised, negative performance index values were obtained at later steps of the optimization.

3.4 Scaling the Transfer Function Coefficients

As was discussed in the previous chapters scaling the transfer function coefficients may be useful in conditioning matrices and vectors for more accurate computation. Consider the case where a pole -300 was added to the example system presented in section 3.2. The optimization program current-

ly used at the Measurement Systems Laboratory at M.I.T. reduces the system coefficeints by time scaling to prevent large values of the E_n matrix elements which are undesirable as discussed previously. The scaling that was done in the remote pole case is shown in Figure 3.5. This scaling has resulted in an unfavorable conditioning of the matrix E_n where some of the diagonal elements are smaller than elements to their right, as shown in Figure 3.6.

When the scaling factor was inverted as shown in Figure 3.7 and the increase of the E_n matrix elements was handled by dividing them by 10^{15}, the resulting condition of the E_n matrix was a desirable one, as the elements on the diagonal dominate in magnitude the elements on their right. This is shown in Figure 3.8. The resulting error in the computed performance index was in this case 30% of its value.

It is not argued that changing the condition of the matrix E_n would usually improve the accuracy of the computation. In some cases it was found to have no effect at all. The possibility to control the condition of the matrix by such scaling should be noted and used when necessary. However the problem of large values of the E_n matrix elements should be treated directly by dividing these elements by a factor and not by scaling. Scaling should be used on a selective basis to condition matrices and vectors for specific operations like the matrix inversion. The establishment of more specific criteria for the usefulness of such scaling in different cases would provide the control system designer means to obtain more accurate computation in many practical cases.

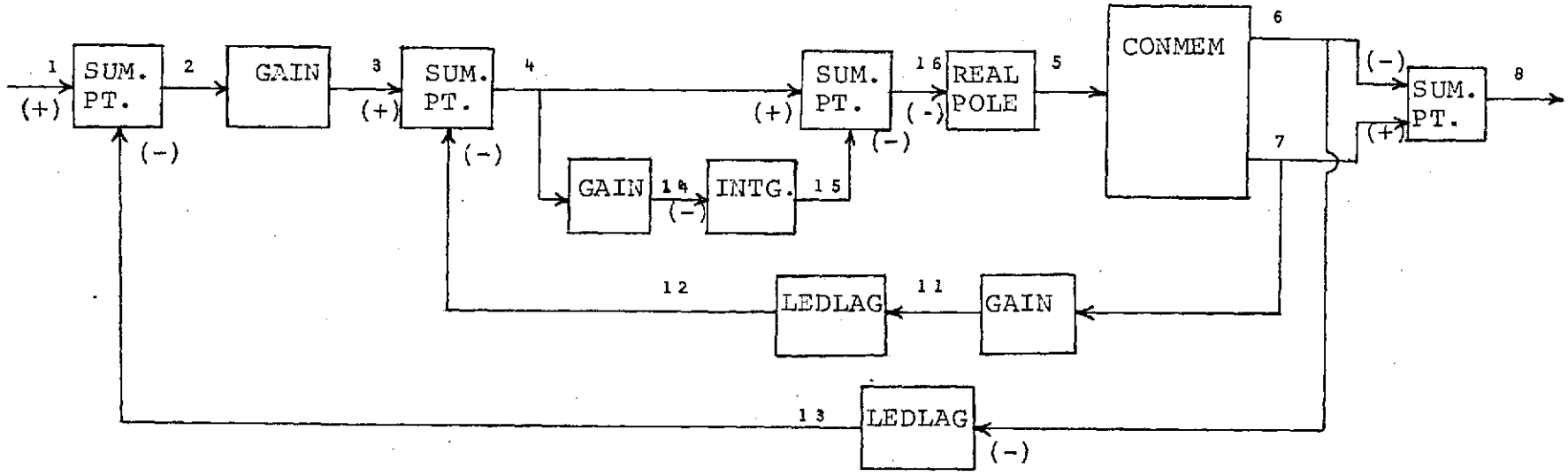


Figure 3.1-A Mathematical Block Diagram and Data Input Definition of the Example Control System

```
SYS MATRIX
      SUMP      0.0          0.0          0.0          0.0
      GAIN      0.14800      0.0          0.0          0.0
      SUMP      0.0          0.0          0.0          0.0
      GAIN      0.75000      0.0          0.0          0.0
      INTG      0.0          0.0          0.0          0.0
      SUMP      0.0          0.0          0.0          0.0
      REAL      0.08000      0.0          0.0          0.0
      CONM      0.0          0.0          0.0          0.0
      SUMP      0.0          0.0          0.0          0.0
      GAIN      0.08100      0.0          0.0          0.0
      LEDL      0.02000      0.25000      0.0          0.0
      LEDL      0.40000      0.50000      0.0          0.0
      REAL      0.34400      0.0          0.0          0.0
      REAL      0.01630      0.0          0.0          0.0

SIG MATRIX
    1  13   2   0   0
    2   0   3   0   0
    3  12   4   0   0
    4   0  14   0   0
    0  14  15   1   0
    4  15  16   0   0
    0  16   5   2   0
    5   0   6   3   4
    7   6   8   0   0
    7   0  11   0   0
   11   0  12   5   0
    0   6  13   8   0
    8   0   9   6   0
    9   0  10   7   0

PAR. NO.    SYS MATRIX LOCATION      PAR. VALUE
   1              2 ,  2                0.148
   2             10 ,  2                0.081
   3             11 ,  3                0.250
   4             12 ,  2                0.400
   5             12 ,  3                0.500

          MIN PARAM INCREMENTS --    .5000E-02   .3000E-02   .2000E-01   .2000E-01   .2000E-01

ICON     5

XCON   ACON MATRIX                                          BCON MATRIX
  3       -.6530        1065.                              -82.00
  4       -.1900E-01    -.1000E-01                         -11.60

OCON     CCON
 6        -.1130E-01     .4690E-02                            2.914
 7         .0            12.43                                .0

ITERATION NUMBER     0

DESIGN PARAMETER VALUES

      0.14800000E+00      0.80999970E-01      0.25000000E+00      0.39999998E+00      0.50000000E+00
```

Figure 3.2(a)-Input Data for the Example System in the System Description Program Format

```
SCALING NECESSARY,  SCALE FACTOR =        2.00

UNSCALED DENOMINATOR COEFFICIENTS
    0.7592E+07    0.2514E+08    0.2189E+08    0.8543E+07    0.1864E+07    0.2278E+06    0.7886E+04    0.1369E+03
    0.1000E+01

UNSCALED NUMERATOR COEFFICIENTS
    0.8708E+07    0.1995E+08    0.1325E+08    0.2916E+07    0.1027E+06    0.9903E+03

SCALED DENOMINATOR COEFFICIENTS
    0.2965E+05    0.1964E+06    0.3421E+06    0.2670E+06    0.1165E+06    0.2847E+05    0.1971E+04    0.6843E+02
    0.1000E+01

SCALED NUMERATOR COEFFICIENTS
    0.3402E+05    0.1559E+06    0.2070E+06    0.9111E+05    0.6418E+04    0.1238E+03

ACOF =  0.2893E+05  0.1932E+06  0.3380E+06  0.2651E+06  0.1161E+06  0.2838E+05  0.1966E+04  0.6833E+02  0.1000E+01

BCOF =  0.2893E+05  0.1326E+06  0.1761E+06  0.7750E+05  0.5459E+04  0.1053E+03


INITIAL CONDITION VECTOR

    -0.99999982E+00       0.0                   0.0                   0.10529182E+03      -0.17354609E+04

    -0.10867187E+05       0.13419100E+07       -0.33179968E+08
```

Figure 3.2(b)-Computed and Scaled Transfer Function Coefficients and Initial Condition Vector

DETERMINANT OF E= 0.15507D+68

E MATRIX

0.0	-0.38640D+06	0.0	-0.53013D+06	0.0	-0.56753D+05	0.0	-0.13666D+03
0.39542D+07	0.26402D+08	0.45800D+08	0.36223D+08	0.15339D+08	0.38779D+07	0.21185D+06	0.93377D+04
-0.27019D+09	-0.18001D+10	-0.31294D+10	-0.24293D+10	-0.10481D+10	-0.24963D+09	-0.14476D+08	-0.42619D+06
0.12332D+11	0.82069D+11	0.14224D+12	0.10984D+12	0.47061D+11	0.11046D+11	0.58805D+09	0.14646D+08
-0.42377D+12	-0.28172D+13	-0.48677D+13	-0.37398D+13	-0.15908D+13	-0.36853D+12	-0.17740D+11	-0.41267D+09
0.11941D+14	0.79304D+14	0.13665D+15	0.10452D+15	0.44181D+14	0.10119D+14	0.44259D+12	0.10457D+11
-0.30258D+15	-0.20084D+16	-0.34549D+16	-0.26352D+16	-0.11098D+16	-0.25256D+15	-0.10434D+14	-0.27195D+12
0.78688D+16	0.52237D+17	0.89961D+17	0.68629D+17	0.28944D+17	0.66071D+16	0.28196D+15	0.81475D+13

X MATRIX

0.55030D+00	-0.50000D+00	-0.52751D+00	-0.39513D-12	0.10774D+03	-0.17355D+04	-0.11130D+05	0.13475D+07
-0.50000D+00	0.52751D+00	0.39513D-12	-0.24472D+01	-0.12101D-11	0.26297D+03	-0.55432D+04	0.19672D+05
-0.52751D+00	-0.39513D-12	0.24472D+01	0.12101D-11	-0.26297D+03	0.55432D+04	-0.19672D+05	-0.26501D+07
0.39513D-12	-0.24472D+01	-0.12101D-11	0.26297D+03	-0.55432D+04	0.19672D+05	0.26501D+07	-0.92176D+08
0.10774D+03	0.12101D-11	-0.26297D+03	-0.55432D+04	0.16306D+06	-0.15059D+07	-0.49116D+08	0.23879D+10
-0.17355D+04	0.26297D+03	0.55432D+04	0.19672D+05	-0.15059D+07	0.30256D+08	-0.59048D+08	-0.16311D+11
-0.11130D+05	-0.55432D+04	-0.19672D+05	0.26501D+07	-0.49116D+08	-0.59048D+08	0.30894D+11	-0.90036D+12
0.13475D+07	0.19672D+05	-0.26501D+07	-0.92176D+08	0.23879D+10	-0.16311D+11	-0.90036D+12	0.37041D+14

Figure 3.3-The Matrices E_n and X Corresponding to Initial Parameter Values of the Example System

DETERMINANT OF E= 0.15507D+28

E MATRIX

0.0	-0.38643D+06	0.0	-0.53013D+06	0.0	-0.56753D+05	0.0	-0.13666D+03
0.39542D+07	0.26402D+08	0.45803D+08	0.36223D+08	0.15339D+08	0.38779D+07	0.21185D+06	0.93377D+04
-0.27019D+09	-0.18001D+10	-0.31294D+10	-0.24293D+10	-0.10481D+10	-0.24963D+09	-0.14476D+08	-0.42619D+06
0.12332D+11	0.82069D+11	0.14224D+12	0.10984D+12	0.47061D+11	0.11046D+11	0.58805D+09	0.14646D+08
-0.42377D+12	-0.28172D+13	-0.48677D+13	-0.37398D+13	-0.15908D+13	-0.36853D+12	-0.17740D+11	-0.41267D+09
0.11941D+14	0.79304D+14	0.13665D+15	0.10452D+15	0.44181D+14	0.10119D+14	0.44259D+12	0.10457D+11
-0.30258D+15	-0.20084D+16	-0.34549D+16	-0.26352D+16	-0.11098D+16	-0.25256D+15	-0.10434D+14	-0.27195D+12
0.78688D+16	0.52237D+17	0.89901D+17	0.68629D+17	0.28944D+17	0.66071D+16	0.28196D+15	0.91475D+13

X MATRIX

0.75317D+00	-0.50000D+00	-0.55445D+00	-0.38168D-12	0.10777D+03	-0.17355D+04	-0.11130D+05	0.13474D+07
-0.50000D+00	0.55445D+00	0.38168D-12	-0.24765D+01	-0.11686D-11	0.26306D+03	-0.55432D+04	0.19671D+05
-0.55445D+00	-0.38168D-12	0.24765D+01	0.11686D-11	-0.26306D+03	0.55432D+04	-0.19671D+05	-0.26501D+07
0.38168D-12	-0.24765D+01	-0.11686D-11	0.26306D+03	-0.55432D+04	0.19671D+05	0.26501D+07	-0.92176D+08
0.10777D+03	0.11686D-11	-0.26306D+03	-0.55432D+04	0.16306D+06	-0.15059D+07	-0.49116D+08	0.23879D+10
-0.17355D+04	0.26306D+03	0.55432D+04	0.19671D+05	-0.15059D+07	0.30256D+08	-0.59047D+08	-0.16311D+11
-0.11130D+05	-0.55432D+04	-0.19671D+05	0.26501D+07	-0.49116D+08	-0.59047D+08	0.30893D+11	-0.90035D+12
0.13474D+07	0.19671D+05	-0.26501D+07	-0.92176D+08	0.23879D+10	-0.16311D+11	-0.90035D+12	0.37041D+14

Figure 3.4-Corrected Values of the Matrices E_n and X of the Example Case (the given Elements of E_n are before reduction)

```
SCALING NECESSARY,  SCALE FACTOR =       2.00

UNSCALED DENOMINATOR COEFFICIENTS
    0.2300E+10    0.7626E+10    0.6659E+10    0.2611E+10    0.5734E+09    0.7089E+08    0.2617E+07    0.4934E+05
    0.4399E+03    0.1000E+01

UNSCALED NUMERATOR COEFFICIENTS
    0.2639E+10    0.6046E+10    0.4019E+10    0.9835E+09    0.3112E+08    0.3001E+06

SCALED DENOMINATOR COEFFICIENTS
    0.4493E+07    0.2979E+08    0.5202E+08    0.4079E+08    0.1792E+08    0.4431E+07    0.3272E+06    0.1234E+05
    0.2199E+03    0.1000E+01

SCALED NUMERATOR COEFFICIENTS
    0.5154E+07    0.2362E+08    0.3137E+08    0.1381E+08    0.9724E+06    0.1876E+05

SCALING NECESSARY,  SCALE FACTOR =       4.00

UNSCALED DENOMINATOR COEFFICIENTS
    0.4493E+07    0.2979E+08    0.5202E+08    0.4079E+08    0.1792E+08    0.4431E+07    0.3272E+06    0.1234E+05
    0.2199E+03    0.1000E+01

UNSCALED NUMERATOR COEFFICIENTS
    0.5154E+07    0.2362E+08    0.3137E+08    0.1381E+08    0.9724E+06    0.1876E+05

SCALED DENOMINATOR COEFFICIENTS
    0.8776E+04    0.1164E+06    0.4064E+06    0.6274E+06    0.5600E+06    0.2768E+06    0.4090E+05    0.3085E+04
    0.1100E+03    0.1000E+01

SCALED NUMERATOR COEFFICIENTS
    0.1007E+05    0.9225E+05    0.2451E+06    0.2157E+06    0.3039E+05    0.1172E+04

ACOF =  0.8563E+04  0.1145E+06  0.4016E+06  0.6328E+06  0.5581E+06  0.2760E+06  0.4077E+05  0.3080E+04  0.1099E+03

ACOF =  0.1000E+01

BCOF =  0.8563E+04  0.7847E+05  0.2095E+06  0.1835E+06  0.2585E+05  0.9971E+03
```

Figure 3.5-Scaling the System Coefficients of the Remote Pole Case

DETERMINANT OF E= 0.357870+36

E MATRIX

0.171250+05	0.0	0.303130+06	0.0	0.111620+07	0.0	0.315450+05	0.0	0.219840+03
-0.188240+07	-0.251460+08	-0.882820+08	-0.138320+09	-0.122730+09	-0.595650+08	-0.882360+07	-0.509490+06	-0.241650+05
0.206920+09	0.276410+10	0.967900+10	0.152340+11	0.133490+11	0.654440+10	0.925750+09	0.654570+08	0.206890+07
-0.176460+11	-0.235680+12	-0.824800+12	-0.129440+13	-0.113500+13	-0.555400+12	-0.774800+11	-0.542390+10	-0.161380+09
0.137920+13	0.184190+14	0.644470+14	0.101110+15	0.886340+14	0.433180+14	0.601210+13	0.418570+12	0.122360+11
-0.105190+15	-0.140470+16	-0.491480+16	-0.770960+16	-0.675520+16	-0.330170+16	-0.457580+15	-0.310210+14	-0.931830+12
0.797890+16	0.106550+18	0.372780+18	0.584750+18	0.512350+18	0.250400+18	0.346910+17	0.241210+16	0.706070+14
-0.604590+18	-0.807360+19	-0.282470+20	-0.443080+20	-0.388220+20	-0.189730+20	-0.262840+19	-0.182750+18	-0.534920+16
0.458040+20	0.611660+21	0.214000+22	0.335680+22	0.294110+22	0.143740+22	0.199130+21	0.138450+20	0.405250+18

Figure 3.6-The Matrix E_n of the Remote Pole Case after Scaling the System Coefficients

```
SCALING NECESSARY,  SCALE FACTOR =        0.50

UNSCALED DENOMINATOR COEFFICIENTS
    0.2300E+10    0.7626E+10    0.6659E+10    0.2611E+10    0.5734E+09    0.7089E+08    0.2617E+07    0.4936E+05
    0.4399E+03    0.1000E+01

UNSCALED NUMERATOR COEFFICIENTS
    0.2639E+10    0.6046E+10    0.4015E+10    0.8835E+09    0.3112E+08    0.3001E+06

SCALED DENOMINATOR COEFFICIENTS
    0.1178E+13    0.1952E+13    0.8524E+12    0.1671E+12    0.1835E+11    0.1134E+10    0.2094E+08    0.1974E+06
    0.8798E+03    0.1000E+01

SCALED NUMERATOR COEFFICIENTS
    0.1351E+13    0.1548E+13    0.5139E+12    0.5655E+11    0.9957E+09    0.4801E+07

ACOF =  0.1149E+13  0.1920E+13  0.8421E+12  0.1659E+12  0.1829E+11  0.1130E+10  0.2088E+08  0.1971E+06  0.8794E+03

ACOF =  0.1000E+01

BCOF =  0.1149E+13  0.1317E+13  0.4372E+12  0.4810E+11  0.8470E+09  0.4084E+07
```

Figure 3.7-Inverted Scaling of the System Coefficients of the Remote Pole Case

```
DETERMINANT OF E=     0.50594D+19

E MATRIX

 0.22985D+13  0.0           0.16843D+13  0.0           0.36576D+11  0.0           0.41751D+08  0.0           0.17598D+04
-0.20213D+16 -0.33750D+16 -0.14811D+16 -0.29037D+15 -0.32164D+14 -0.19515D+13 -0.36715D+11 -0.30489D+09 -0.15466D+07
 0.17775D+19  0.29679D+19  0.12991D+19  0.25506D+18  0.27594D+17  0.17161D+16  0.30335D+14  0.26811D+12  0.10552D+10
-0.12127D+22 -0.20244D+22 -0.88563D+21 -0.17374D+21 -0.19042D+20 -0.11647D+19 -0.20311D+17 -0.17763D+15 -0.65977D+12
 0.75825D+24  0.12657D+25  0.55360D+24  0.10856D+24  0.11892D+23  0.72676D+21  0.12608D+20  0.10973D+18  0.40256D+15
-0.46264D+27 -0.77227D+27 -0.33774D+27 -0.66225D+26 -0.72534D+25 -0.44315D+24 -0.76768D+22 -0.66733D+20 -0.24427D+18
 0.28073D+30  0.46861D+30  0.20494D+30  0.40182D+29  0.44010D+28  0.26837D+27  0.46562D+25  0.40468D+23  0.14807D+21
-0.17018D+33 -0.28406D+33 -0.12423D+33 -0.24358D+32 -0.26678D+31 -0.16298D+30 -0.28223D+28 -0.24528D+26 -0.89745D+23
 0.10314D+36  0.17217D+36  0.75294D+35  0.14763D+35  0.16169D+34  0.98779D+32  0.17105D+31  0.14866D+29  0.54391D+26
```

Figure 3.8-The Matrix E_n of the Remote Pole Case after Inverted Scaling

CHAPTER 4

ALTERNATIVE METHODS FOR APPLICATION IN A PARAMETER OPTIMIZATION PROGRAM

4.1 Introduction

The search algorithm of a parameter optimization program must utilize other routines that execute some specific parts of the computation, like the system mathematical representation and the solution of equations like the matrix equation (1.6). The methods presented in this chapter are an attempt to take more accurate and cost effective approaches to the problems of the matrix equation solution and the system representation.

4.2 A Minimal Method for the Solution of the Matrix Equation: $AX + XA^T = -X_0$

The method for the solution of equation (1.6) that was suggested by Palsson [10] and was described and analyzed in sections 1.3 and 2.1 through 2.5 is extensive and for the most part inefficient. The number of arithmetic operations in this method is very large, which makes the solution of the equation for many practical design problemns both expensive and inaccurate.

The method proposed in this section minimizes the number of arithmetic operations needed to solve the equation.

It takes advantage of the fact that the matrices X and X_0 are symmetric and the matrix A is in phase variable form.

Since the matrix X is a symmetric n'th order matrix(n being the order of the system), there are $\frac{n(n+1)}{2}$ unknowns. Consider the case of a 4'th order system. The matrix equation in this case is:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{12} & x_{22} & x_{23} & x_{24} \\ x_{13} & x_{23} & x_{33} & x_{34} \\ x_{14} & x_{24} & x_{34} & x_{44} \end{bmatrix}$$

$$+ \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{12} & x_{22} & x_{23} & x_{24} \\ x_{13} & x_{23} & x_{33} & x_{34} \\ x_{14} & x_{24} & x_{34} & x_{44} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & -a_0 \\ 1 & 0 & 0 & -a_1 \\ 0 & 1 & 0 & -a_2 \\ 0 & 0 & 1 & -a_3 \end{bmatrix}$$

$$= \begin{bmatrix} x_{0_{11}} & x_{0_{12}} & x_{0_{13}} & x_{0_{14}} \\ x_{0_{12}} & x_{0_{22}} & x_{0_{23}} & x_{0_{24}} \\ x_{0_{13}} & x_{0_{23}} & x_{0_{33}} & x_{0_{34}} \\ x_{0_{14}} & x_{0_{24}} & x_{0_{34}} & x_{0_{44}} \end{bmatrix}$$

where the system coefficients a_i and the elements $x_{0_{ij}}$ of the matrix x_0 are known, and the elements x_{ij} of the matrix X are unknown.

The following 10 independent equations for the solution of the 10 unknowns are readily obtained from the matrix equation:

$$
\begin{aligned}
2x_{12} &= -x_{0_{11}} \\
x_{22} + x_{13} &= -x_{0_{12}} \\
x_{23} + x_{14} &= -x_{0_{13}} \\
x_{24} - a_0x_{11} - a_1x_{12} - a_2x_{13} - a_3x_{14} &= -x_{0_{14}} \\
2x_{23} &= -x_{0_{22}} \\
x_{33} + x_{24} &= -x_{0_{23}} \\
x_{34} - a_0x_{12} - a_1x_{22} - a_2x_{23} - a_3x_{24} &= -x_{0_{24}} \\
2x_{34} &= -x_{0_{33}} \\
x_{44} - a_0x_{13} - a_1x_{23} - a_2x_{33} - a_4x_{34} &= -x_{0_{34}} \\
-2a_0x_{14} - 2a_1x_{24} - 2a_2x_{34} - 2a_3x_{44} &= -x_{0_{44}}
\end{aligned}
$$

The equations are slightly modified and reordered in groups:

group (a)
$$
\begin{aligned}
x_{12} &= -\tfrac{1}{2}x_{0_{11}} \\
x_{23} &= -\tfrac{1}{2}x_{0_{22}} \\
x_{34} &= -\tfrac{1}{2}x_{0_{33}}
\end{aligned}
$$

group (b)
$$x_{14} = -x_{23} - x_{0_{13}}$$

group (c)
$$
\begin{aligned}
x_{13} &= -x_{22} - x_{0_{12}} \\
x_{24} &= -x_{33} - x_{0_{23}}
\end{aligned}
$$

group (d)
$$
\begin{aligned}
x_{24} - a_0x_{11} - a_1x_{12} - a_2x_{13} - a_3x_{14} &= -x_{0_{14}} \\
x_{23} - a_0x_{12} - a_1x_{22} - a_2x_{23} - a_3x_{24} &= -x_{0_{24}} \\
x_{44} - a_0x_{13} - a_1x_{23} - a_2x_{33} - a_3x_{34} &= -x_{0_{34}} \\
-2a_0x_{14} - 2a_1x_{24} - 2a_2x_{34} - 2a_3x_{44} &= -x_{0_{44}}
\end{aligned}
$$

The original set of 10 independent equations is easily reduced to a set of 4 independent equations in the following way: The unknowns in group(a) are now known. The unknowns in group

(b) (here only x_{14}) are directly obtained by substitution of solved unknowns of group (a). The unknowns on the left hand side of group (c) and all the unknowns that have been solved are substituted into the equations of group (d) to give the reduced set of 4 equations for the 4 diagonal unknowns:

$$
\begin{aligned}
-a_0x_{11}+a_2x_{22}- \; x_{33} &= -x_{0_{14}} - a_3x_{0_{13}} + \tfrac{1}{2}a_3x_{0_{22}} - \tfrac{1}{2}a_1x_{0_{11}} + x_{0_{23}} \\
-a_1x_{22}+a_3x_{33} &= -x_{0_{24}} - a_3x_{0_{23}} - \tfrac{1}{2}a_2x_{0_{22}} + \tfrac{1}{2}x_{0_{22}} - \tfrac{1}{2}a_0x_{0_{11}} \\
a_0x_{22}-a_2x_{33}+ \; x_{44} &= -x_{0_{34}} - \tfrac{1}{2}a_3x_{0_{33}} - \tfrac{1}{2}a_1x_{0_{22}} - a_0x_{0_{12}} \\
2a_1x_{33}-2a_3x_{44} &= -x_{0_{44}} + a_2x_{0_{33}} - 2a_1x_{0_{23}} - 2a_0x_{0_{13}}
\end{aligned}
$$

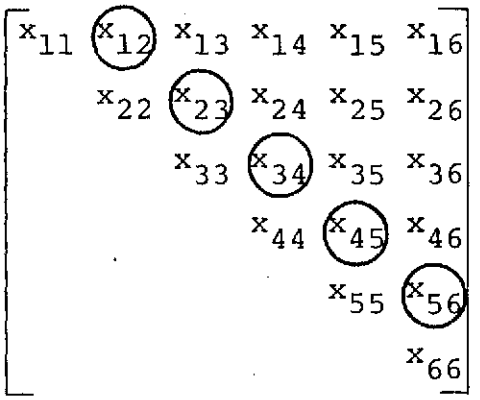
After these equations are solved for the diagonal elements, the unknowns of group (c) are obtained easily.

This method becomes an effective tool for the solution of the matrix equation for systems of any order when its following general properties are realized:

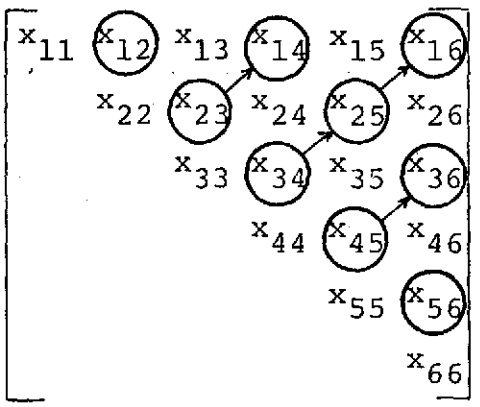
1. The simplicity of the operations in the derivation of the final set of n equations is the same for systems of any order. These operations are simple substitutions of one element by another, which is done by manipulation of the unknowns indices. Relatively few arithmetic operations are done in arranging the final set of equations.
2. Because of the special arrangement of the unity elements in the matrices A and A^T, there are very simple relations between the indices of the unknowns in each

equation, which enable one to describe the reduction operation that was discussed above for the 4'th order case in the following four steps general scheme:

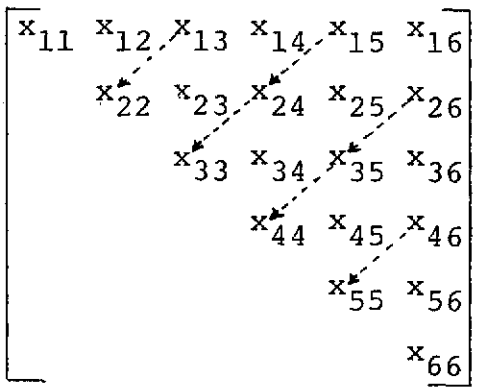
a) The encircled elements are the elements of group (a) which are equal to corresponding elements of the matrix X_0 divided by 2.



b) Consequently, the group (b) elements are obtained.



c) The group (c) elements are expressed in terms of the diagonal elements.



d) All elements are substituted into the group (d) equations that become the reduced set of n equations for the solution of the n diagonal elements.

In the machine procedure steps 3 and 4 will be built by index manipulation in the instructions that form the reduced equations set.

After the reduction it is left to solve n linear equation in n unknowns, or, in other words, it is left to solve an equation of the type:

$$B\,\underline{x} = \underline{c} \qquad (4.1)$$

where $\underline{x}$ and $\underline{c}$ are n dimensional vectors and B is an n x n dimensional matrix. This reduction can be regarded as a reduction of the matrix equation (1.6) to the matrix equation (4.1), or as a reduction of a set of n x n linear equations to a set of n linear equations. For comparison, in the method suggested by Palsson the equation:

$$E_n \underline{x}_1 = \sum_{i=1}^{n} E_{i-1} \underline{x}_{0_{n-i}} \qquad (2.3)$$

(which can be regarded as another reduced form of the original matrix equation) must be solved for the first column of the state matrix X. Numerous arithmetic computations must be done to obtain the E_i matrices first, and then the sum vector on the right hand side of the equation (see sections 2.2 and 2.3). The number of operations that must be done to obtain the matrix B and the vector $\underline{c}$ in equation (4.1) is relatively small. There is no loss of accuracy at all in obtaining the matrix B, and there is a minimal amount of arithmetic computation in obtaining the vector $\underline{c}$.

If the operations that must be done prior to the solution of equation (2.3) are compared to the reduction operations that must be done prior to the solution of equation (4.1) in the new method, the advantage of this new method can be realized. The computation of the remaining elements of the X matrix which is done by simple substitutions in the new method, is also much more complicated in Palsson's method, which again involves matrix operations. (As was shown in section 2.5, if the Model Performance Index is used and the order of the model is small enough, the computation of the remaining elements of the X matrix can be done by simple shifting operations, in which case this part of the computation in the two methods will be of comparable simplicity.)

4.3 An Alternative Approach to the Derivation of a Linear System Transfer Function

One method for the derivation of the transfer function of a multi-loop system with a multi-input multi-output controlled member was mentioned in section 1.4. For high order systems the matrix operations of equation (1.21) require a great deal of computation adjoined by loss of accuracy. As was discussed before, significantly inaccurate system representation may not be tolerated by the optimization procedure.

The approach suggested in this section is to obtain the transfer function of the controlled member first (if it is not given in this form) and then obtain the transfer function

of the whole system by multiplication of the system members transfer functions in the proper order. This approach actually replaces the matrix operations by polynomial operations. The accuracy of arithmetic operations between polynomials has not been examined in this work, and while the amount of computation would probably be smaller than in the n dimensional matrix operations, improved accuracy is not guaranteed. When the system members are represented by their transfer functions, numerator and denominator polynomials can be multiplied and summed in any correct order, to give the over-all transfer function (e.g. obtaining the transfer function for each loop, proceeding from inner to outer loops).

If the controlled member is given by a set of dynamic equations, its transfer function must be derived first. In the case of a single input this can be done by application of equations (1.20) and (1.21). (Notice that in many practical cases the order of the controlled member may be appreciably smaller than the order of the whole system, and the small dimensional matrix operations to obtain its transfer function would not cause inaccuracies.) In the case of a multi-input controlled member whose inputs may be coupled, the transfer function can be derived by polynomial manipulation. Consider a controlled member with two inputs and three outputs, whose Laplace transformed dynamic equations are:

$$\underline{\ell}_1 y_1 + \underline{\ell}_2 y_2 + \underline{\ell}_3 y_3 = \underline{m}_1 u_1 + \underline{m}_2 u_2 \qquad (4.2)$$

where y_i is the i'th output and u_j is the j'th input and $\underline{\ell}_i$ and $\underline{m}_j$ are 3 dimensional vectors whose elements are polynomials of the Laplace variable.

In the control system the controlled member outputs may be fed back to its inputs through control components. All the feedback paths between controlled member inputs and outputs can be arranged so that there is not more than one transfer function in each path, as shown in the figure.

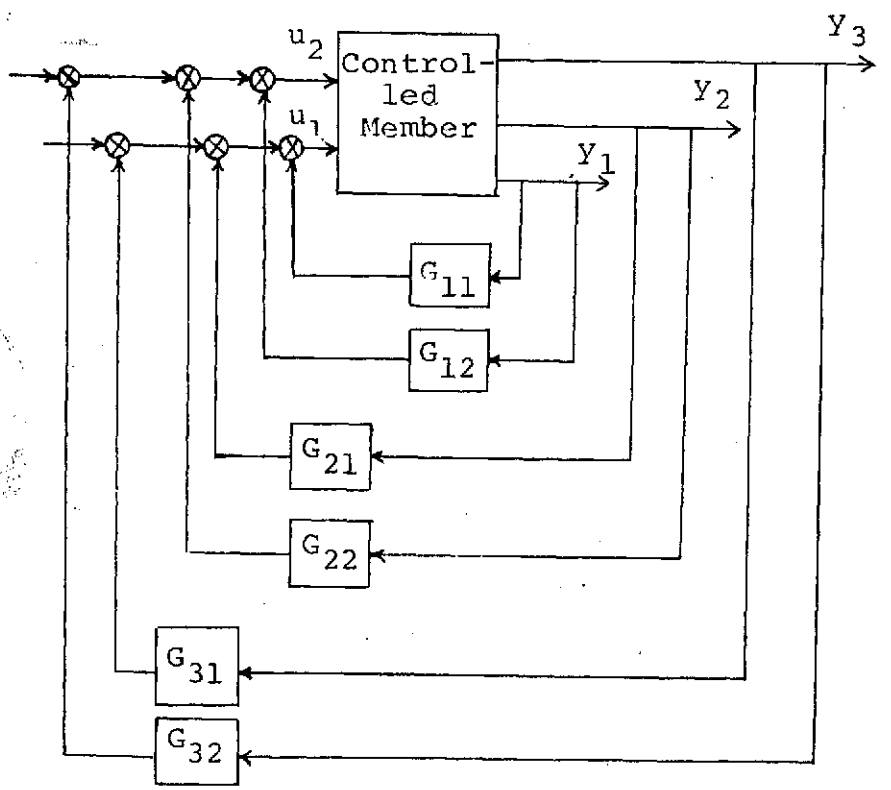


Figure 4.1 A Controlled Member with Coupled Inputs

If the transfer function between the input u_1 and any output is desired, then u_2 is expressed as:

$$u_2 = G_{12}y_1 + G_{22}y_2 + G_{23}y_3 \tag{4.3}$$

and substituted into equation (4.2):

$$(\underline{\ell}_1 - G_{12})y_1 + (\underline{\ell}_2 - G_{22})y_2 + (\underline{\ell}_3 - G_{23})y_3 = \underline{m}_1 u_1$$

or

$$\underline{\ell}_1^* y_1 + \underline{\ell}_2^* y_2 + \underline{\ell}_3^* y_3 = \underline{m}_1 u_1 \tag{4.4}$$

where the elements of the vectors $\underline{\ell}_1^*$, $\underline{\ell}_2^*$ and $\underline{\ell}_3^*$ are ratios of polynomials of the Laplace variable, since:

$$\begin{aligned} \underline{\ell}_1^* &= \underline{\ell}_1 - G_{12} \\ \underline{\ell}_2^* &= \underline{\ell}_2 - G_{22} \\ \underline{\ell}_3^* &= \underline{\ell}_3 - G_{23} \end{aligned} \tag{4.5}$$

The transfer function between the input u_1 and any of the outputs can be obtained by application of Cramer's rule, e.g.:

$$\frac{y_1}{u_1} = \frac{\begin{vmatrix} \underline{m}_1 & \underline{\ell}_2^* & \underline{\ell}_3^* \end{vmatrix}}{\begin{vmatrix} \underline{\ell}_1^* & \underline{\ell}_2^* & \underline{\ell}_3^* \end{vmatrix}} \tag{4.6}$$

To obtain the transfer function as a ratio of two expanded polynomials, the numerator and denominator determinants must be expanded, which requires operations between the polynomial elements of the vectors $\underline{m}_1$, $\underline{\ell}_2^*$, $\underline{\ell}_3^*$ etc. In many practical cases some of the transfer functions G_{ij} would be zeros or simple gains and the others would usually be ratios of low order polynomials, so that these operations would usually be

simple and accurate.

It is interesting to realize that the operations between the elements of the vectors actually replace the operations that would have to be done between the transfer functions G_{ij} and the controlled member transfer function in the regular case without coupled inputs, to obtain the over-all system transfer function. Thus, the coupling between the inputs does not require additional operations but rather a different arrangement of the controlled member equations and the corresponding control system members, according to the procedure that was described above.

4.4 A Method for Finding the Relationships between the Transfer Function Coefficients and the Design Parameters

In a parameter optimization procedure a representation of the control system must be done at each step of the optimum search in order to evaluate its performance and to compute the new step. The derivation of the system transfer function may be a time consumming process, especially for high order systems. In the algorithm that was discussed previously the transfer function must be computed twice for each design parameter for the computation of the gradients at each step. Inaccurate computation of the transfer function coefficients may cause problems like the "performance index increase in half a step" and incorrect gradient values, arising usually when the system transfer function is computed twice

for close values of design parameters.

If the transfer function coefficients can be expressed as simple functions of the design parameters, then after these functions have been determined, the values of the coefficients at each step of the optimum search can be found with relatively small additional effort, and the derivatives of the system coefficients with respect to the design parameters can be found by directly deriving these functions.

The method proposed in this section applies to high order complicated systems, where the computation of the transfer function must be done numerically and the distribution of the design parameters cannot be followed easily, if at all, through the computation from the initial system, given by its components, to the final mathematical representation.

The design parameters may be system gains, time constants, damping ratios and natural frequencies. The over-all system transfer function will be considered in the following form:

$$G(S) = \frac{b_m S^m + b_{m-1} S^{m-1} + \ldots + b_0}{a_n S^n + a_{n-1} S^{n-1} + \ldots + a_0}$$

The coefficients : $b_i \quad 0 \leq i \leq m$

$a_j \quad 0 \leq j \leq n$

can be replaced by the coefficients: $c_k \quad 0 \leq k \leq m+n$

where:

$$c_k = b_i \quad \begin{cases} k = i \\ 0 \leq i \leq m \end{cases}$$

$$c_k = a_j \quad \begin{cases} k = m+j+1 \\ 0 \leq j \leq n \end{cases}$$

When the system transfer function is derived by properly ordered multiplication of its members transfer functions, the over-all transfer function coefficients are sums of products of design parameters and constant numbers. Each one of the transfer function coefficients can be expressed by the following general expression:

$$c_k = \sum_j k_j \Pi P_i \tag{4.7}$$

where k_j are constants and P_i are design parameters. Expression (4.7) should be read as follows: "Each polynomial coefficient is equal to the sum of all the combinatorial possibilities of products of the design parameters and constant numbers". If there are two design parameters, for example, each polynomial coefficient can be expressed as:

$$c_k = k_1 + k_2P_1 + k_3P_2 + k_4P_1P_2 \tag{4.8}$$

In the case of three design parameters the general expression of the coefficients would be:

$$c_k = k_1+k_2P_1+k_3P_2+k_4P_3+k_5P_1P_2+k_6P_1P_3+k_7P_2P_3+k_8P_1P_2P_3 \tag{4.9}$$

In the general case of ℓ design parameters the number of products or the number of constants k_j can be computed by the combinatorial expressions:

$$C_\ell^0 + C_\ell^1 + C_\ell^2 + \ldots\ldots + C = 2^\ell$$

where

$$C_\ell^k \equiv \frac{\ell!}{(\ell-k)!\ k!}$$

Example 4.1. The transfer function of the system in Figure 4.3 is:

$$G(S) = \frac{10S + 20}{P_1S^3 + S^2 + (P_1 + 10P_2)S + (1+20P_2)}$$

The constants k_j for each polynomial coefficient are listed in the following table:

coefficient	k_1	k_2	k_3	k_4
$c_0=b_0=20$	20	0	0	0
$c_1=b_1=10$	10	0	0	0
$c_2=a_0=1+20P_2$	0	0	20	1
$c_3=a_1=P_1+10P_2$	0	1	10	0
$c_4=a_2=1$	1	0	0	0
$c_5=a_3=P_1$	0	1	0	0

Figure 4.2 The Constant k_j in the Polynomial Coefficients of Example 4.1

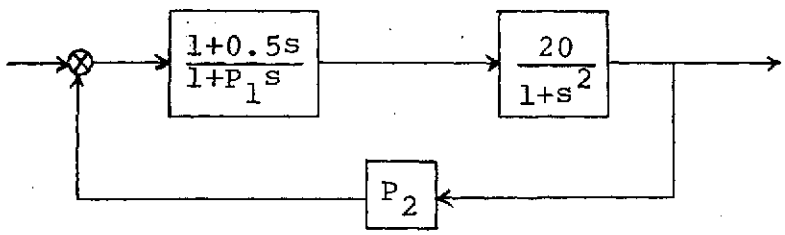


Figure 4.3 A Control System for Example 4.1

To find the relationship between the design parameters and the transfer function coefficients it is necessary to find the values of the constants k_j of the general equation (4.7) for each polynomial coefficient. Notice that the equation (4.7) is linear in the constants k_j (also see equations (4.8) and (4.9)). Thus, the solution for the constants of each design parameter can be obtained by forming sets of enough equations for each transfer function coefficient, to solve for its constants.

To solve the 2^n constants k_j in a polynomial coefficient 2^n equations are needed. These equations can be produced by substituting 2^n sets of design parameter values into the given control system, and computing the corresponding sets of polynomial coefficients. All the combinatorial products of each set of design parameters are also needed to form the equations.

In the case of two design parameters, for example, four equations are needed to solve for the four constants in each polynomial coefficient. Four sets of design parameter values will produce the following set of equations for the constants in the k'th polynomial coefficient c_k:

$$
\begin{aligned}
c_{k_{(1)}} &= k_1 + P_{1_{(1)}} k_2 + P_{2_{(1)}} k_3 + P_{1_{(1)}} P_{2_{(1)}} k_4 \\
c_{k_{(2)}} &= k_1 + P_{1_{(2)}} k_2 + P_{2_{(2)}} k_3 + P_{1_{(2)}} P_{2_{(2)}} k_4 \\
c_{k_{(3)}} &= k_1 + P_{1_{(3)}} k_2 + P_{2_{(3)}} k_3 + P_{1_{(3)}} P_{2_{(3)}} k_4 \\
c_{k_{(4)}} &= k_1 + P_{1_{(4)}} k_2 + P_{2_{(4)}} k_3 + P_{1_{(4)}} P_{2_{(4)}} k_4
\end{aligned}
\qquad (4.10)
$$

The solution of the constants k_j can be obtained by Cramer's rule. e.g.:

$$k_1 = \frac{\begin{vmatrix} c_{k_{(1)}} & P_{1_{(1)}} & P_{2_{(1)}} & P_{1_{(1)}} P_{2_{(1)}} \\ c_{k_{(2)}} & P_{1_{(2)}} & P_{2_{(2)}} & P_{1_{(2)}} P_{2_{(2)}} \\ c_{k_{(3)}} & P_{1_{(3)}} & P_{2_{(3)}} & P_{1_{(3)}} P_{2_{(3)}} \\ c_{k_{(4)}} & P_{1_{(4)}} & P_{2_{(4)}} & P_{1_{(4)}} P_{2_{(4)}} \end{vmatrix}}{D} \qquad (4.11)$$

where:

$$D = \begin{vmatrix} 1 & P_{1_{(1)}} & P_{2_{(1)}} & P_{1_{(1)}} P_{2_{(1)}} \\ 1 & P_{1_{(2)}} & P_{2_{(2)}} & P_{1_{(2)}} P_{2_{(2)}} \\ 1 & P_{1_{(3)}} & P_{2_{(3)}} & P_{1_{(4)}} P_{2_{(3)}} \\ 1 & P_{1_{(4)}} & P_{2_{(4)}} & P_{1_{(4)}} P_{2_{(4)}} \end{vmatrix} \qquad (4.12)$$

Notice that the divider D does not depend on the system at all, but only on the values of the design parameters that were chosen to produce the equations. Also notice that in the numerator determinant only one column depends on the system, and the other elements are functions of the chosen design parameters. It follows that four sets of values for two design parameters can be chosen once and then be used for all systems with two design parameters. The divider can be computed for these constant values independently of the control system. Since the numerator determinants in the expressions of the constants k_j differ only by the values in the column of system coefficients $c_{k_{(i)}}$ and by the location of this column, the co-

factors of all the elements of the determinant D can be computed as system independent constants and be used for the computation of each k_j. This computation will then be done by multiplying a set of system coefficients by the corresponding constant cofactors and dividing by the constant divider. This, of course, applies to any number of design parameters as well.

Example 4.2. Consider all systems with two design parameters. Four sets of design parameter values are necessary to produce the equations for the solution of the constants k_j, as was explained previously. The following sets may be selected:

set 1: $P_1=1$ $P_2=1$

set 2: $P_1=2$ $P_2=1$

set 3: $P_1=1$ $P_2=2$

set 4: $P_1=2$ $P_2=2$

The corresponding divider D (see equation 4.12) is 1 and the cofactors are listed in the following table:

column	cofactors			
1	4	2	-2	-1
2	2	2	-1	-1
3	-2	-1	2	1
4	-1	-1	1	1

Figure 4.4 Cofactors for Systems with Two Design Parameters

Suppose the constants k_j in the polynomial coefficients of the system in example (4.1) are to be determined by this method. The computation of the constants in the coefficients c_0 and c_3 is illustrated below:

$$c_0 = 20$$

For the four sets of design parameter values we get:

$$c_{0_{(1)}} = c_{0_{(2)}} = c_{0_{(3)}} = c_{0_{(4)}} = 20$$

Multiplying the coefficients column by the cofactors we get:

$$\begin{aligned} k_1 &= 20\cdot 4 \quad -20\cdot 2 \quad +20(-2) \quad -20(-1) = 20 \\ -k_2 &= 20\cdot 2 \quad -20\cdot 2 \quad +20(-1) \quad -20(-1) = 0 \\ k_3 &= 20(-2) \quad -20(-1) \quad +20\cdot 2 \quad -20\cdot 1 = 0 \\ -k_4 &= 20(-1) \quad -20(-1) \quad +20\cdot 1 \quad -20\cdot 1 = 0 \end{aligned}$$

and for:

$$c_3 = P_1 + 10P_2$$

the coefficients column as obtained by substituing the design parameters values into the control system and computing the transfer function coefficients is:

$$\begin{aligned} c_{3_{(1)}} &= 1 + 10 = 11 \\ c_{3_{(2)}} &= 2 + 10 = 12 \\ c_{3_{(3)}} &= 1 + 20 = 21 \\ c_{3_{(4)}} &= 2 + 20 = 22 \end{aligned}$$

and the constants for c_3 are:

$$k_1 = 11 \cdot 4 - 12 \cdot 2 + 12(-2) - 22(-1) = 0$$

$$-k_2 = 11 \cdot 2 - 12 \cdot 2 + 21(-1) - 22(-1) = -1$$

$$k_3 = 11(-2) - 12(-1) + 21 \cdot 1 - 22 \cdot 1 = 10$$

$$-k_4 = 11(-1) - 12(-1) + 21 \cdot 1 - 22 \cdot 1 = 0$$

A program for the computation of c factors and dividers using specified sets of design parameters values is given in Appendix C. Values of cofactors and dividers are also listed. The program is written for systems of up to six design parameters and the listed results are for up to four parameters. The computation was found to be time consuming as the number of design parameters gets larger. It should be noted, however, that the values of the cofactors and the divider must be obtained but once, and then be used for all systems of a given number of design parameters. The program also punches out the values of cofactors, as listed in the table in Appendix C. For the specific choice of design parameters sets the divider value is 1 for any number of design parameters. To use these results for finding the constants in the transfer function coefficients according to the procedure described above, the same design parameters sets must be used to obtain the corresponding transfer function coefficients. The procedure listed in the main program may be used to obtain the same sets of design parameters.

Since the values of cofactors and dividers have been ob-

tained independently of the control system itself (only the number of design parameters needs to be specified), the system's mathematical representation in terms of the design parameters can be done by finding the corresponding numerical values of the transfer function coefficients, for the computations of the constants k_j in each coefficient. This means that the system transfer function must be derived numerically a number of times, even when this proposed method is used. This, however, is done before the optimization process begins. For ℓ design parameters the transfer function derivation must be done 2^{ℓ} times. This number is small in the case of few design parameters (for three design parameters the transfer function must be computed eight times). In the case of many design parameters this number would get considerably large (32 for five design parameters). It should be noted, however, that in the currently used method, in the case of five design parameters the transfer function must be computed at least 30 times in three optimization steps. The number of steps grows rapidly with the number of design parameters. Notice that the number of steps also grows with the order of the system, and in the currently used method this means more computations of the system transfer function, while in the newly proposed method the number of times that the transfer function must be computed depends only on the number of design parameters.

An even more significant advantage of the new method would be the accuracy of the computation as outlined by the following three points:

1. As can be seen in Appendix C for the specific choice of design parameter values (only values of 1 and 2 are used) the divider is always 1 and all the cofactors are small integers so that the computation of the transfer function coefficients involves minimal round-off errors imposed by the design parameters. This computation can always be checked for accuracy by use of another set of design parameters values. The resulting k_j constants must be the same for any choice of design parameters. If they vary for different sets of parameters values, average values can be taken.
2. Once the constants k_j have been determined for each tranfer function coefficient, the computation of the coefficients at each step involves very few arithmetic operations. Problems like the "performance index increase in half a step" are not likely to happen.
3. The computation of the derivatives of the transfer function coefficients with respect to the design parameters is very simple. For example, in the case of three design parameters, using equation (4.9) we have:

$$\frac{\partial c_k}{\partial P_1} = k_2 + k_5 P_2 + k_6 P_3 + k_8 P_2 P_3$$

$$\frac{\partial c_k}{\partial P_2} = k_3 + k_5 P_1 + k_7 P_3 + k_8 P_1 P_3$$

$$\frac{\partial c_k}{\partial P_3} = k_4 + k_6 P_1 + k_7 P_2 + k_8 P_1 P_2$$

Finally, it should be noted that if the system transfer function is computed by the state-space method (see equation 1.21), the transfer function coefficients discussed in this section appear divided by the coefficient a_n. In this case the dependence of a_n on the design parameters must be determined separately. If the polynomial approach is used for obtaining the transfer function (as proposed in the previous section) no special consideration must be given to a_n.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The numerical accuracy problems arising in some stages of a parameter optimization technique have been presented and analysed. Special attention was given to the solution of the matrix equation(1.6) and to the system mathematical representation. The accuracy obtained at each stage of the computation depends on the operations and the numerical values involved.

A slight difference between the computed static sensitivities of the control system and the model, when the Model Performance Index is used, was found to cause some of the severe inaccuracy problems encountered in the design by parameter optimization. In the example case the difference was one part of 10^7. Another major source of inaccuracy is the inversion of the matrix E_n. Large values of the matrix elements, which result in a large determinant value, may cause severe inaccuracies even when digital overflow or underflow do not occur in the matrix inversion. This problem can be handled by reduction of the matrix elements before its inversion. The required reduction factor would depend on the transfer function coefficients and on the order of the system. No specific criterion for the value of the reduction factor

has been established. However, keeping the matrix determinant value below 10^{40} has given satisfactory results in the cases that were examined. A situation where the diagonal elements are significantly smaller than elements on their right in the rows is also unfavorable. Scaling the system coefficients is a useful means for conditioning matrices and vectors in a desirable way. The importance of accurate system representation to the optimization procedure has been emphasized.

Alternative approaches to the problems of the system mathematical representation and the solution of the matrix equation (1.6) have been suggested for better accuracy and cost effectiveness.

5.2 Recommendations

Some of the changes recommended here have been applied to the parameter optimization program used at the Measurement Systems Laboratory at M.I.T. and have given the desired improvement in the accuracy of the computation. The recommendations are listed below:

1. Fixing the system static sensitivity to be exactly equal to that of the model (fixing the value of x_{0_1} at -1 when the model static sensitivity is 1) has resulted in immediate improvement of the computed results.
2. Reducing the elements of the matrix E_n prior to its inversion when these elements are large has improved

the accuracy appreciably. The decision to reduce the matrix elements and the choice of the reduction factor can be built in the program by examination of the characteristic coefficients and the system order. Consideration of the general expression of the E_n matrix (see Appendix A) may be useful in estimating the magnitude order of the matrix determinant without computing the determinant value itself.

3. Scaling the system coefficients can be used to control the conditioning of matrices and vectors. It is recommended that the scaling operation be separated from the reduction of the matrix elements that was discussed above. Additional research should establish more specific criteria for scaling the system coefficients on a selective basis. Conditioning the matrix E_n prior to its inversion is one use of such scaling. The reduction of the matrix elements, on the other hand, should be done routinely whenever these elements are too large.
4. The option of inverting the matrix E_n in extended precision has been suggested. The matrix inversion program and the control cards are given in Appendix B.
5. The computation of the initial conditions can be done in double precision without substantial additional storage. This change has given significant improvement in the computed results.

6. As was discussed in section 2.5 some of the computation currently done in the determination of the elements of the state matrix X is unnecessary, since many of these elements do not take part in the computation of the performance index. Except for the first column of the matrix X only the elements included in the $(\ell+1)x(\ell+1)$ left upper corner of the X matrix must be computed (ℓ being the order of the model).
7. In the optimization program currently used a routine for checking and correcting the matrix X to be symmetric is include. This is not neccesary since by the shifting operation in the computation of the columns of the matrix, discussed in section 2.5, the computed X matrix is always symmetric.
8. The method for solving the matrix equation (1.6), that was proposed in section 4.1 would provide a more accurate and less expensive solution than the method currently used. This method can replace the current algorithm without additional changes in the optimization program.
9. The method proposed in section 4.2 for the computation of the system transfer function may be more accurate than the one currently used, but its accuracy has not been examined. It suggests a different approach, avoiding high order matrix operations.

10. The method proposed in section 4.4 is recommended as an accurate and cost effective way for computing the system transfer function and the gradients at each step of the optimization procedure.

APPENDIX A

THE MATRICES E_i

The first three E_i matrices are given below:

$$\begin{bmatrix} a_{n-1} & -1 & 0 & 0 \cdots\cdots 0 & 0 \\ 0 & a_{n-1} & -1 & 0 \cdots\cdots 0 & 0 \\ \vdots & & & & \\ 0 & 0 & \cdots\cdots\cdots\cdots & a_{n-1} & -1 \\ a_0 & a_1 & \cdots\cdots\cdots\cdots & a_{n-2} & 2a_{n-1} \end{bmatrix}$$

$$E_2 = \begin{bmatrix} a_{n-2} & -a_{n-1} & 1 & 0 & 0 \cdots\cdots\cdots\cdots 0 & 0 \\ 0 & a_{n-2} & a_{n-1} & 1 & 0 \cdots\cdots\cdots\cdots 0 & 0 \\ 0 & 0 & a_{n-2} & -a_{n-1} & 1 \cdots\cdots\cdots\cdots 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 \cdots\cdots\cdots & a_{n-2} & -a_{n-1} & 1 \\ -a_0 & -a_1 & -a_2 \cdots\cdots\cdots & a_{n-3} & 0 & -2a_{n-1} \\ 2a_0a_{n-1} & -a_0+2a_1a_{n-1} & -a_1+2a_2a_{n-1} \cdots\cdots & -a_{n-3}+2a_{n-1}a_{n-2} & & 2a_{n-1}^2 \end{bmatrix}$$

$$E_3=\begin{bmatrix} a_{n-3} & -a_{n-2} & a_{n-1} & -1 & 0 & 0 & \cdots & 0 \\ 0 & a_{n-3} & -a_{n-2} & a_{n-1} & -1 & 0 & & 0 \\ 0 & 0 & a_{n-3} & -a_{n-2} & a_{n-1} & -1 & & 0 \\ & & & & & & & \\ 0 & 0 & & & a_{n-1} & & -1 & \\ a_0 & a_1 & & & 0 & & 2a_{n-1} & \\ -2a_0a_{n-1} & a_0-2a_1a_{n-1} & & & 2a_{n-3}-a_{n-1}a_{n-2} & & -2a_{n-1}^2 & \\ 2a_0a_{n-1}^2 & -2a_0a_{n-1}+2a_1a_{n-1}^2 & \cdots\cdots\cdots\cdots & & -2a_{n-3}a_{n-1} & & 2a_{n-3} & \\ & & & & +2a_{n-2}a_{n-1}^2 & & -2a_{n-2}a_{n-1} & \\ & & & & & & +2^3a_{n-1} & \end{bmatrix}$$

The first and last columns of E_4 and E_5 are given below:

$$E_4=\begin{bmatrix} a_{n-4} & & 0 \\ 0 & & \vdots \\ \vdots & & \\ 0 & & 1 \\ -a_0 & & -2a_{n-1} \\ 2a_0a_{n-1} & & 2a_{n-1}^2 \\ -2a_0a_{n-1}^2 & & -2a_{n-3}+2a_{n-2}a_{n-1}-2a_{n-1}^3 \\ 2a_0a_{n-3}-2a_0a_{n-2}a_{n-1}+2a_0a_{n-1}^3 & \cdots\cdots & 4a_{n-1}a_{n-3}-4a_{n-2}a_{n-1}^2+2a_{n-1}^4 \end{bmatrix}$$

$$E_5=\begin{bmatrix} a_{n-5} & \cdots & 0 \\ 0 & & 0 \\ \vdots & & \vdots \\ 0 & & -1 \\ a_0 & & 2a_{n-1} \\ 2a_0a_{n-1} & & -2a_{n-1}^2 \\ 2a_0a_{n-1}^2 & & 2a_{n-3}-2a_{n-2}a_{n-1}+2a_{n-1}^3 \\ 2a_0a_{n-3}+2a_0a_{n-2}a_{n-1}-2a_0a_{n-1}^3 & & -4a_{n-3}a_{n-1}+4a_{n-2}a_{n-1}^2-2a_{n-1}^4 \\ 4a_0a_{n-3}a_{n-1}-4a_0a_{n-2}a_{n-1}^2+2a_0a_{n-1}^4 & \cdots & 2a_{n-5}-2a_{n-4}a_{n-1}+2a_{n-3}a_{n-2} \\ & & +2a_{n-2}^2a_{n-1}-6a_{n-2}a_{n-1}^3 \\ & & +6a_{n-1}^2a_{n-3}+2a_{n-1}^5 \end{bmatrix}$$

As the computation of the E_i matrices proceeds, the elements shift upwards in the columns, inverting signs, while the last element of, say, the kth column is computed by multiplying the kth column of the previous E matrix by the last row of the system's matrix A.

APPENDIX B

A PROGRAM FOR

MATRIX INVERSION IN QUADRAUPLE PRECISION

The program is a modification of the IBM/I Scientific Subroutine Package MINV program for extended precision and for utilization by a FORTRAN main program. Also listed is the control cards set up for the IBM/370 system at M.I.T. Information Processing Center. This set up may change due to improvements of the system.

```
MINV..                                                                  MINV  10
/*************************************************************************/MINV  20
/*                                                                      */MINV  30
/*      TO INVERT A MATRIX                                              */MINV  40
/*                                                                      */MINV  50
/*************************************************************************/MINV  60
   PROCEDURE (AQ,N,DQ,CONQ)OPTIONS(FORTRAN),.                            ***   7
     PUT SKIP LIST('JUST ENTERED MINV');
    PUT SKIP DATA(AQ,N,DQ,CONQ);
   DECLARE                                                              MINV  80
        ERROR EXTERNAL CHARACTER(1),                                    MINV  90
        (I,J,K,N,L(N),M(N))                                             MINV 100
    /*
        FIXED BINARY,                                                   MINV 110
     */
        FIXED BINARY(31,0),
     (BIGA,HOLD,D,CON,S) BINARY FLOAT(109),
     A(1:N,1:N) BINARY FLOAT(109),
     DC(1:N,1:N) BINARY FLOAT(109),
     (DQ,AQ(20,20),CONQ) BINARY FLOAT(53),./*DOUBLE PRECISION INPUT      ***131
/*      BINARY FLOAT (53),.           /*DOUBLE PRECISION VERSION  /*D*/MINV 140
/*                                                                      */MINV 150
     D=DQ,.
     DO I=1 TO N,.
     DO J=1 TO N,.
     A(I,J)=AQ(I,J),.
     END,.
     END,.
     CON=CONQ,.
   ERROR='0',.                                                          MINV 160
   IF N LE 0                                                            MINV 170
   THEN DO,.                                                            MINV 180
        ERROR='1',.                  /* ORDER OF MATRIX = 0.          */MINV 190
        GO TO FIN,.                                                     MINV 200
        END,.                                                           MINV 210
   IF CON= 0                                                            MINV 220
```

```
  THEN S     =1.0E-5,.                  /* SINGLE PRECISION VERSION /*S*/MINV 230
/*THEN S     =1.0E-15,.                 /* DOUBLE PRECISION VERSION /*D*/MINV 240
  ELSE S     =CON,.                                                    MINV 250
  IF N = 1                              /* INVERT A SCALAR           */MINV 260
  THEN DO,.                                                           MINV 270
       D     =A(1,1),.                                                MINV 280
       IF ABS(D) LE S                                                 MINV 290
       THEN DO,.                                                      MINV 300
            ERROR='2',.                                               MINV 310
            END,.                                                     MINV 320
       ELSE A(1,1) = 1/D,.                                            MINV 330
       GO TO FIN,.                                                    MINV 340
       END,.                                                          MINV 350
  D    =1.0,.                           /* SEARCH FOR LARGEST ELEMENT */MINV 360
       DO K = 1 TO N,.                                                MINV 370
       L(K) =K,.                                                      MINV 380
       M(K) =K,.                                                      MINV 390
       BIGA =A(K,K),.                                                 MINV 400
            DO I=K TO N,.                                             MINV 410
                 DO J=K TO N,.                                        MINV 420
                 IF ABS(BIGA) LT ABS(A(I,J))                          MINV 430
                 THEN DO,.                                            MINV 440
                      BIGA =A(I,J),.                                  MINV 450
                      L(K) =I,.                                       MINV 460
                      M(K) =J,.                                       MINV 470
                      END,.                                           MINV 480
                 END,.                                                MINV 490
            END,.                                                     MINV 500
       J    =L(K),.                     /* INTERCHANGE ROWS          */MINV 510
       IF L(K) GT K                                                   MINV 520
       THEN DO,.                                                      MINV 530
                 DO I = 1 TO N,.                                      MINV 540
                 HOLD =-A(K,I),.                                      MINV 550
                 A(K,I)=A(J,I),.                                      MINV 560
                 A(J,I)=HOLD,.                                        MINV 570
                 END,.                                                MINV 580
            END,.                                                     MINV 590
       I    =M(K),.                     /* INTERCHANGE COLUMNS       */MINV 600
```

```
          IF M(K) GT K                                                    MINV 610
          THEN DO;.                                                       MINV 620
                    DO J = 1 TO N;.                                       MINV 630
                    HOLD =-A(J,K);.                                       MINV 640
                    A(J,K)=A(J,I);.                                       MINV 650
                    A(J,I)=HOLD;.                                         MINV 660
                    END;.                                                 MINV 670
               END;.                                                      MINV 680
          IF ABS(BIGA) LE S                                               MINV 690
          THEN DO;.                                                       MINV 700
               D    =0.0;.                                               MINV 710
               GO TO COMP;.                                               MINV 720
               END;.                                                      MINV 730
     /*                                                                  */MINV 740
     /*   DIVIDE COLUMNS BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS       */MINV 750
     /*   CONTAINED IN BIGA)                                             */MINV 760
               DO I = 1 TO N;.                                            MINV 770
               IF I NE K                                                  MINV 780
               THEN A(I,K)=A(I,K)/(-A(K,K));.                             MINV 790
               END;.                                                      MINV 800
               DO I = 1 TO N;.          /* REDUCE MATRIX                 */MINV 810
               IF I NE K                                                  MINV 820
               THEN DO;.                                                  MINV 830
                         DO J = 1 TO N;.                                  MINV 840
                         IF J NE K                                        MINV 850
                         THEN A(I,J)=A(I,K)*A(K,J)+A(I,J);.               MINV 860
                         END;.                                            MINV 870
                    END;.                                                 MINV 880
               END;.                                                      MINV 890
               DO J = 1 TO N;.                                            MINV 900
               IF J NE K                /* DIVIDE BY ROW PIVOT          */MINV 910
               THEN A(K,J)=A(K,J)/A(K,K);.                                MINV 920
               END;.                                                      MINV 930
          D    =D*A(K,K);.              /* COMPUTE DETERMINANT          */MINV 940
COMP;.                                                                    MINV 950
          IF ABS(D) LE S                                                  MINV 960
```

```
         THEN DO;.                                                    MINV 970
              ERROR='2';.              /* DETERMINANT IS ZERO        */MINV 980
              GO TO FIN;.                                             MINV 990
              END;.                                                   MINV1000
         A(K,K)=1.0/A(K,K);.           /* REPLACE PIVOT BY RECIPROCAL */MINV1010
         END;.                                                        MINV1020
/*                                                                    */MINV1030
/*       FINAL ROW AND COLUMN INTERCHANGE                             */MINV1040
/*                                                                    */MINV1050
   K     =N;.                                                        MINV1060
LOOP.;                                                              MINV1070
   K     =K-1;.                                                      MINV1080
   IF K GT 0                                                         MINV1090
   THEN DO;.                                                         MINV1100
        I     =L(K);.                                                MINV1110
        IF I GT K                                                    MINV1120
        THEN DO;.                                                    MINV1130
                  DO J = 1 TO N;.                                    MINV1140
                  HOLD =A(J,K);.                                     MINV1150
                  A(J,K)=-A(J,I);.                                   MINV1160
                  A(J,I)=HOLD;.                                      MINV1170
                  END;.                                              MINV1180
             END;.                                                   MINV1190
        J     =M(K);.                                                MINV1200
        IF J GT K                                                    MINV1210
        THEN DO;.                                                    MINV1220
                  DO I = 1 TO N;.                                    MINV1230
                  HOLD =A(K,I);.                                     MINV1240
                  A(K,I)=-A(J,I);.                                   MINV1250
                  A(J,I)=HOLD;.                                      MINV1260
                  END;.                                              MINV1270
             END;.                                                   MINV1280
        GO TO LOOP;.                                                 MINV1290
        END;.                                                        MINV1300
FIN.;                                                               MINV1310
     DO=0;.
```

```
 /*
  AQ=A;.
 */
 DO I = 1 TO N;
 DO J = 1 TO N;
 AQ(I,J) = A(I,J);
 END;
 END;
 PUT SKIP LIST('ABOUT TO RETURN FROM MINV');
  CLOSE FILE(SYSPRINT);
RETURN;.                                                    MINV1320
END;.                          /*END OF PROCEDURE MINV      */MINV1330
```

B.2 Control Cards Set Up for IBM/370 System at M.I.T.

```
// 'NAME ',CLASS=A,REGION=350K
/*MITID   USER=
/*MAIN TIME=2,LINES=6,CARDS=0
//STEP1  EXEC  FORC
//C.SYSIN DD *
```

FORTRAN cards

```
//STEP2  EXEC  PLIKCLG,PARM.C='CS(48),OBJECT,NORUN,COMPATIBLE',
//  PARM.G='COMPATIBLE'
//C.SYSIN  DD  *,DCB=BLKSIZE=2000
```

PL/I cards

```
//L.SYSLIB DD DSN=U.M8230.7182.SUBR.PALSS,DISP=SHR
//  DD
//  DD
//         DD DSN=SYS2.SSP.SUBR,DISP=SHR
//         DD DSN=SYS1.FORTLIB,DISP=SHR
//         DD DSN=SYS5.FORTLIB.SUBR,DISP=SHR
//L.SYSIN  DD  *
```

OBJECT cards

```
ENTRY MAIN
NAME USERPROG
//G.FT06F001  DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=2036)
//G.FT05F001  DD  *,DCB=BLKSIZE=2000
```

DATA cards

APPENDIX C

THE SYSTEM INVARIANT DETERMINANTS AND COFACTORS

A program for the computation of the system invariant determinant and its cofactors for the determination of the relationship between the system coefficients and the design parameters is listed. The program computs determinants and cofactors for systems of up to six design parameters. Also listed are the results for specified sets of design parameters, that can be used for the computation of the constants k_j, as described in section 4.4.

```
      DIMENSION P(6),A(64,64),B(64,64),DET(64)
 1002 FORMAT(' ','DETERMINANT=',F8.1/' THE COFACTORS OF THE I    COLUMN
     1OF THE NUMERATOR DETERMINANT,'/' CORRESPONDING TO THE CONSTANT K(I
     2),ARE LISTED IN THE I    GROUP'//'  I'/)
 1003 FORMAT('1','NUMBER OF DESIGN PARAMETERS = ',I1/' THE FOLLOWING SET
     1S OF DESIGN PARAMETERS VALUES ARE USED' )
 1004 FORMAT(I3,3X, 10(F8.1,2X)  /(6X,10(F8.1,2X)))
 1005 FORMAT(10F8.1)
 1006 FORMAT(' ','SET ',I2,' =',6(F7.1))
      DO 410 L=1,6
      WRITE(6,1003)L
      L1=L-1
      L2=L-2
      L3=L-3
      L4=L-4
      II=1
      DO 10 I1=1,L
   10 P(I1)=1.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
      DO 20 I1=1,L
      II=II+1
      P(I1)=2.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
   20 CONTINUE
      IF(L.EQ.1) GOTO 200
      DO 30 I1=1,L1
      I11=I1+1
      DO 30 I2=I11,L
      II=II+1
      P(I1)=2.
      P(I2)=2.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
   30 CONTINUE
```

```
      IF(L.EQ.2) GOTO 200
      DO 40 I1=1,L2
      I11=I1+1
      DO 40 I2=I11,L1
      I21=I2+1
      DO 40 I3=I21,L
      II=II+1
      P(I1)=2.
      P(I2)=2.
      P(I3)=2.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
   40 CONTINUE
      IF(L.EQ.3) GOTO 200
      DO 50 I1=1,L3
      I11=I1+1
      DO 50 I2=I11,L2
      I21=I2+1
      DO 50 I3=I21,L1
      I31=I3+1
      DO 50 I4=I31,L
      II=II+1
      P(I1)=2.
      P(I2)=2.
      P(I3)=2.
      P(I4)=2.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
   50 CONTINUE
      IF(L.EQ.4) GOTO 200
      DO 60 I1=1,L4
      I11=I1+1
      DO 60 I2=I11,L3
      I21=I2+1
      DO 60 I3=I21,L2
      I31=I3+1
```

```
      DO 60 I4=I31,L1
      I41=I4+1
      DO 60 I5=I41,L
      II=II+1
      P(I1)=2.
      P(I2)=2.
      P(I3)=2.
      P(I4)=2.
      P(I5)=2.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
   60 CONTINUE
      IF(L.EQ.5) GOTO 200
      II=II+1
      P(I1)=2.
      P(I2)=2.
      P(I3)=2.
      P(I4)=2.
      P(I5)=2.
      P(I6)=2.
      WRITE(6,1006)II,(P(J),J=1,L)
      CALL AMAT(P,II,A,L)
  200 LL=2**L
      I=1
      DO 210 II=1,LL
      DO 210 JJ=1,LL
  210 B(II,JJ)=A(II,JJ)
      IX=0
      KC=LL
      J=0
      GOTO 300
  100 J=J+1
      IF(J.GT.LL) GOTO 410
      I=0
  101 I=I+1
      IK=0
```

```
      DO 221 II=1,LL
      IF(II.EQ.I) GOTO 221
      IK=IK+1
      JK=0
      DO 220 JJ=1,LL
      IF(JJ.EQ.J) GOTO 220
      JK=JK+1
      B(IK,JK)=A(II,JJ)
  220 CONTINUE
  221 CONTINUE
      KC=LL-1
  300 IF(KC.EQ.1)GO TO 31
      IREV=0
      DO 22 IT=1,KC
      K=IT
    9 IF(B(K,IT))21,11,21
   11 K=K+1
      IF(K-KC)9,9,51
   21 IF(IT-K)12,14,51
   12 DO 13 M=1,KC
      TEMP=B(IT,M)
      B(IT,M)=B(K,M)
   13 B(K,M)=TEMP
      IREV=IREV+1
   14 IS=IT+1
      IF(IS.GT.KC) GOTO 22
      DO 17 M=IS,KC
   18 IF(B(M,IT))19,17,19
   19 TEMP=B(M,IT)/B(IT,IT)
      DO 16 N=IT,KC
   16 B(M,N)=B(M,N)-B(IT,N)*TEMP
   17 CONTINUE
   22 CONTINUE
      DET(I)=1.
      DO 2 IT=1,KC
    2 DET(I)=DET(I)*B(IT,IT)
```

```
      DET(I)=(-1.D0)**IREV*DET(I)
      GOTO 310
   51 DET(I)=0.
      GOTO 310
   31 DET(I)=B(1,1)
  310 IF(IX.EQ.0) GOTO 400
      IF(I.EQ.LL) GOTO 320
      GOTO 101
  320 WRITE(6,1004)J,(DET(I),I=1,LL)
      WRITE(7,1005) (DET(I),I=1,LL)
      GOTO 100
  400 WRITE(6,1002)  DET(1)
      IX=1
      GOTO 100
  410 CONTINUE
      STOP
      END
      SUBROUTINE AMAT(P,II,A,L)
      DIMENSION A(64,64),P(6)
      L1=L-1
      L2=L-2
      L3=L-3
      L4=L-4
      JJ=1
      A(II,JJ)=1.
      DO 10 I1=1,L
      JJ=JJ+1
   10 A(II,JJ)=P(I1)
      IF(L.EQ.1) GOTO 100
      DO 20 I1=1,L1
      I11=I1+1
      DO 20 I2=I11,L
      JJ=JJ+1
      A(II,JJ)=P(I1)*P(I2)
   20 CONTINUE
```

```
      IF(L.EQ.2) GOTO 100
      DO 30 I1=1,L2
      I11=I1+1
      DO 30 I2=I11,L1
      I21=I2+1
      DO 30 I3=I21,L
      JJ=JJ+1
      A(II,JJ)=P(I1)*P(I2)*P(I3)
   30 CONTINUE
      IF(L.EQ.3)GOTO 100
      DO 40 I1=1,L3
      I11=I1+1
      DO 40 I2=I11,L2
      I21=I2+1
      DO 40 I3=I21,L1
      I31=I3+1
      DO 40 I4=I31,L
      JJ=JJ+1
      A(II,JJ)=P(I1)*P(I2)*P(I3)*P(I4)
   40 CONTINUE
      IF(L.EQ.4) GOTO 100
      DO 50 I1=1,L4
      I11=I1+1
      DO 50 I2=I11,L3
      I21=I2+1
      DO 50 I3=I21,L2
      I31=I3+1
      DO 50 I4=I31,L1
      I41=I4+1
      DO 50 I5=I41,L
      JJ=JJ+1
      A(II,JJ)=P(I1)*P(I2)*P(I3)*P(I4)*P(I5)
      IF(L.EQ.5) GOTO 100
      A(II,JJ)=P(1)*P(2)*P(3)*P(4)*P(5)*P(6)
   50 CONTINUE
  100 DO 500 I=1,L
```

```
500 P(I)=1.
    RETURN
    END
```

C.2 Determinant and Cofactor Values for Specified Design Parameters

```
NUMBER OF DESIGN PARAMETERS = 1
THE FOLLOWING SETS OF DESIGN PARAMETERS VALUES ARE USED
SET  1 =     1.0
SET  2 =     2.0
DETERMINANT=      1.0
THE COFACTORS OF THE I    COLUMN OF THE NUMERATOR DETERMINANT,
CORRESPONDING TO THE CONSTANT K(I),ARE LISTED IN THE I    GROUP

 I

 1         2.0         1.0
 2         1.0         1.0
```

```
NUMBER OF DESIGN PARAMETERS = 2
THE FOLLOWING SETS OF DESIGN PARAMETERS VALUES ARE USED
SET  1 =     1.0     1.0
SET  2 =     2.0     1.0
SET  3 =     1.0     2.0
SET  4 =     2.0     2.0
DETERMINANT=      1.0
THE COFACTORS OF THE I    COLUMN OF THE NUMERATOR DETERMINANT,
CORRESPONDING TO THE CONSTANT K(I),ARE LISTED IN THE I    GROUP

 I

 1         4.0         2.0        -2.0        -1.0
 2         2.0         2.0        -1.0        -1.0
 3        -2.0        -1.0         2.0         1.0
 4        -1.0        -1.0         1.0         1.0
```

```
NUMBER OF DESIGN PARAMETERS = 3
THE FOLLOWING SETS OF DESIGN PARAMETERS VALUES ARE USED
SET  1 =     1.0     1.0     1.0
SET  2 =     2.0     1.0     1.0
SET  3 =     1.0     2.0     1.0
SET  4 =     1.0     1.0     2.0
SET  5 =     2.0     2.0     1.0
SET  6 =     2.0     1.0     2.0
SET  7 =     1.0     2.0     2.0
SET  8 =     2.0     2.0     2.0
DETERMINANT=      1.0
THE COFACTORS OF THE 1    COLUMN OF THE NUMERATOR DETERMINANT,
CORRESPONDING TO THE CONSTANT K(I),ARE LISTED IN THE I    GROUP

 1

 1          8.0       4.0      -4.0       4.0       2.0      -2.0       2.0       1.0
 2          4.0       4.0      -2.0       2.0       2.0      -2.0       1.0       1.0
 3         -4.0      -2.0       4.0      -2.0      -2.0       1.0      -2.0      -1.0
 4          4.0       2.0      -2.0       4.0       1.0      -2.0       2.0       1.0
 5          2.0       2.0      -2.0       1.0       2.0      -1.0       1.0       1.0
 6         -2.0      -2.0       1.0      -2.0      -1.0       2.0      -1.0      -1.0
 7          2.0       1.0      -2.0       2.0       1.0      -1.0       2.0       1.0
 8          1.0       1.0      -1.0       1.0       1.0      -1.0       1.0       1.0
```

NUMBER OF DESIGN PARAMETERS = 4
THE FOLLOWING SETS OF DESIGN PARAMETERS VALUES ARE USED

SET 1 =	1.0	1.0	1.0	1.0
SET 2 =	2.0	1.0	1.0	1.0
SET 3 =	1.0	2.0	1.0	1.0
SET 4 =	1.0	1.0	2.0	1.0
SET 5 =	1.0	1.0	1.0	2.0
SET 6 =	2.0	2.0	1.0	1.0
SET 7 =	2.0	1.0	2.0	1.0
SET 8 =	2.0	1.0	1.0	2.0
SET 9 =	1.0	2.0	2.0	1.0
SET 10 =	1.0	2.0	1.0	2.0
SET 11 =	1.0	1.0	2.0	2.0
SET 12 =	2.0	2.0	2.0	1.0
SET 13 =	2.0	2.0	1.0	2.0
SET 14 =	2.0	1.0	2.0	2.0
SET 15 =	1.0	2.0	2.0	2.0
SET 16 =	2.0	2.0	2.0	2.0

DETERMINANT= 1.0
THE COFACTORS OF THE I COLUMN OF THE NUMERATOR DETERMINANT,
CORRESPONDING TO THE CONSTANT K(I),ARE LISTED IN THE I GROUP

I										
1	16.0	8.0	-8.0	8.0	-8.0	-4.0	4.0	-4.0	4.0	-4.0
	4.0	2.0	-2.0	2.0	-2.0	-1.0				
2	8.0	8.0	-4.0	4.0	-4.0	-4.0	4.0	-4.0	2.0	-2.0
	2.0	2.0	-2.0	2.0	-1.0	-1.0				
3	-8.0	-4.0	8.0	-4.0	4.0	4.0	-2.0	2.0	-4.0	4.0
	-2.0	-2.0	2.0	-1.0	2.0	1.0				
4	8.0	4.0	-4.0	8.0	-4.0	-2.0	4.0	-2.0	4.0	-2.0
	4.0	2.0	-1.0	2.0	-2.0	-1.0				
5	-8.0	-4.0	4.0	-4.0	8.0	2.0	-2.0	4.0	-2.0	4.0
	-4.0	-1.0	2.0	-2.0	2.0	1.0				
6	-4.0	-4.0	4.0	-2.0	2.0	4.0	-2.0	2.0	-2.0	2.0
	-1.0	-2.0	2.0	-1.0	1.0	1.0				
7	4.0	4.0	-2.0	4.0	-2.0	-2.0	4.0	-2.0	2.0	-1.0
	2.0	2.0	-1.0	2.0	-1.0	-1.0				
8	-4.0	-4.0	2.0	-2.0	4.0	2.0	-2.0	4.0	-1.0	2.0
	-2.0	-1.0	2.0	-2.0	1.0	1.0				
9	4.0	2.0	-4.0	4.0	-2.0	-2.0	2.0	-1.0	4.0	-2.0
	2.0	2.0	-1.0	1.0	-2.0	-1.0				
10	-4.0	-2.0	4.0	-2.0	4.0	2.0	-1.0	2.0	-2.0	4.0
	-2.0	-1.0	2.0	-1.0	2.0	1.0				
11	4.0	2.0	-2.0	4.0	-4.0	-1.0	2.0	-2.0	2.0	-2.0
	4.0	1.0	-1.0	2.0	-2.0	-1.0				
12	2.0	2.0	-2.0	2.0	-1.0	-2.0	2.0	-1.0	2.0	-1.0
	1.0	2.0	-1.0	1.0	-1.0	-1.0				
13	-2.0	-2.0	2.0	-1.0	2.0	2.0	-1.0	2.0	-1.0	2.0
	-1.0	-1.0	2.0	-1.0	1.0	1.0				
14	2.0	2.0	-1.0	2.0	-2.0	-1.0	2.0	-2.0	1.0	-1.0
	2.0	1.0	-1.0	2.0	-1.0	-1.0				
15	-2.0	-1.0	2.0	-2.0	2.0	1.0	-1.0	1.0	-2.0	2.0
	-2.0	-1.0	1.0	-1.0	2.0	1.0				
16	-1.0	-1.0	1.0	-1.0	1.0	1.0	-1.0	1.0	-1.0	1.0
	-1.0	-1.0	1.0	-1.0	1.0	1.0				

REFERENCES

1. Beyers, F.H.: A General Synthesis Program for Multi Input Control Systems using a Model Reference Performance Index, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T., 1971

2. Bryson, A.E. and Ho, Y.C.: Applied Optimal Control, Ginn and Company, 1969.

3. Clark, R.N.: Introduction to Automatic Control Systems, J. Wiley Inc., 1962.

4. Griffin, W.R.: A Method for Control System Synthesis, S.M. Thesis, Department of Aeronautics and Stronautics, M.I.T., 1970.

5. Hellerman, H.: Digital Computer System Principles, McGraw-Hill, 1973

6. Hilldebrand,F.B.: Introduction to Numerical Analysis, McGraw-Hill, 1956.

7. Householder, A.S.: The Theory of Matrices in Numerical Analysis, Blaisdell Publishing Co., 1965.

8. IBM - System/360 Scientific Subroutine Package (PL/I). [360A-CM-07X]

9. Osborn, E.E.: "On Pre-Conditioning of Matrices", J.ACM, Vol. 7, p.338, 1960.

10. Palsson, T.: Parameter Uncertainties in Control System Design, Measurement Systems Laboratory, TE-46, M.I.T., 1971.

11. Rediess, H.A.: A New Model Performance Index for the Engineering Design of Control Systems, Experimental Astronomy Laboratory, TE-26, M.I.T., 1968.

12. Von Newman, J. and Goldstine, H.H.: "Numerical Inverting of Matrices of High Order." Bulletin of American Mathematical Society, Vol.53, p.1021-1099, 1947.

13. Whitaker, H.P.: The System Description Program, Measurement Systems Laboratory, RE-79, Revision 1, M.I.T., 1972.

14. Wilkinson, J.H.: Rounding Errors in Algebraic Processes, Prentice-Hall, 1963.