AN ALGORITHM FOR CONTROL SYSTEM DESIGN VIA PARAMETER OPTIMIZATION

by Prasun K. Sinha

January 1972
AN ALGORITHM FOR CONTROL SYSTEM DESIGN VIA PARAMETER OPTIMIZATION

by

PRASUN K. SINHA

B. Tech Indian Institute of Technology, Kharagpur 1969

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

January 1972

Signature of the Author

Department of Aeronautics and Astronautics, January 1972

Certified by

Thesis Supervisor

Accepted by Chairman, Departmental Graduate Committee
AN ALGORITHM FOR CONTROL SYSTEM DESIGN VIA PARAMETER OPTIMIZATION

by

PRASUN K. SINHA

Submitted to the Department of Aeronautics and Astronautics in January 1972 in partial fulfillment of the requirements for the degree of Master of Science

ABSTRACT

An algorithm for design via parameter optimization has been developed for linear-time-invariant control systems based on the model-reference adaptive control concept. A cost functional is defined to evaluate the system response relative to nominal, which involves in general the error between the system and nominal response, its derivatives and the control signals. A program for the practical implementation of this algorithm has been developed, with the computational scheme for the evaluation of the performance index based on Lyapunov's theorem for stability of Linear-Invariant-Systems.

Thesis Supervisor: James E. Potter, Ph.D.
Title: Associate Professor of Aeronautics and Astronautics
ACKNOWLEDGEMENTS

The author wishes to thank Dr. James E. Potter for supervising this study; for many hours spent in discussions, for his comments and suggestions which made this thesis possible. The author also thanks Professor H. Philip Whitaker who originally suggested the project and on whose grant this entire study was conducted.

This report was prepared under DSR Project 72709 sponsored by the NASA-Edwards Flight Research Center, through NASA grant number NGL 22-009-548.

The publication of this thesis does not constitute approval by the National Aeronautics and Space Administration. It is published only for the exchange and stimulation of ideas.
LIST OF PRINCIPAL NOMENCLATURE

General: Upper case letters represent matrices, lower case letters with underbars represent vectors.

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B</td>
<td>known matrices in matrix-algebraic equation (Chapter 3)</td>
</tr>
<tr>
<td>b</td>
<td>bus variable vector (Chapter 2)</td>
</tr>
<tr>
<td>C</td>
<td>known RHS matrix of matrix algebraic equation (Chapter 3)</td>
</tr>
<tr>
<td>c_</td>
<td>command variable vector</td>
</tr>
<tr>
<td>e</td>
<td>error vector</td>
</tr>
<tr>
<td>E</td>
<td>three dimensional array in equations for control system representation (Chapter 2)</td>
</tr>
<tr>
<td>F_s</td>
<td>system differential transition matrix</td>
</tr>
<tr>
<td>F_m</td>
<td>model differential transition matrix</td>
</tr>
<tr>
<td>F</td>
<td>augmented system differential transition matrix</td>
</tr>
<tr>
<td>G</td>
<td>augmented system control matrix</td>
</tr>
<tr>
<td>H</td>
<td>three dimensional array in equations for control system representation (Chapter 2)</td>
</tr>
<tr>
<td>I</td>
<td>identity matrix</td>
</tr>
<tr>
<td>J</td>
<td>performance index</td>
</tr>
<tr>
<td>K_i</td>
<td>the i \textsuperscript{th} parameter</td>
</tr>
<tr>
<td>L_i</td>
<td>companion matrix (Chapter 3)</td>
</tr>
<tr>
<td>M, \tilde{M}, \tilde{N}, \tilde{N}, P, Q'</td>
<td>system matrices in describing equations (Chapter 2)</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>(Q_s)</td>
<td>system control matrix</td>
</tr>
<tr>
<td>(Q_m)</td>
<td>model control matrix</td>
</tr>
<tr>
<td>(R,S,T)</td>
<td>matrices in system describing equations (Chapter 2)</td>
</tr>
<tr>
<td>(S_s)</td>
<td>system output matrix</td>
</tr>
<tr>
<td>(S_m)</td>
<td>model output matrix</td>
</tr>
<tr>
<td>(t)</td>
<td>time</td>
</tr>
<tr>
<td>(U,V)</td>
<td>matrices in form of assumed solution for (X) (Chapter 3)</td>
</tr>
<tr>
<td>(W_e,W_r,W_u)</td>
<td>error and control signal weighting matrices</td>
</tr>
<tr>
<td>(w)</td>
<td>vector of magnitudes of delta function commands</td>
</tr>
<tr>
<td>(X)</td>
<td>unknown matrix in matrix algebraic equation (Chapter 3)</td>
</tr>
<tr>
<td>(x_s)</td>
<td>system state vector</td>
</tr>
<tr>
<td>(x_m)</td>
<td>model state vector</td>
</tr>
<tr>
<td>(x)</td>
<td>augmented system state vector</td>
</tr>
<tr>
<td>(y_s)</td>
<td>system output vector</td>
</tr>
<tr>
<td>(y_m)</td>
<td>model output vector</td>
</tr>
</tbody>
</table>

**SPECIAL NOTATIONS**

\[
F^T = \text{Transpose of } F \\
||x|| = \sqrt{x^Tx} \\
||x||_Q = x^TQx
\]
CHAPTER 1

Introduction

The algorithm for "design via parameter optimization" to be described in subsequent chapters is based on the model reference adaptive control concept [7]. The concept is briefly reviewed here and the problem to be examined in this study is formulated.

1.1 The Model Reference Adaptive Control Concept

These systems employ a model which represents the nominal transfer function between input and output. The model is placed in parallel configuration with the system. This arrangement is shown schematically in Fig. 1.1.

Fig. 1.1 Functional Diagram Illustrating the Model Reference Adaptive Control Concept
The same command signals are applied to both the model and system. If the system output differs from that of the model, then the control elements must be modified. The objective is to make the system response track closely that of the model by adjustment of the controller parameters (gains and time constants).

1.2 Problem Formulation and Objective

The actuating signals for controller parameter adjustment are quadratic functions of the error between the nominal and plant response and perhaps some of its derivatives. For single outputs, if $y_s$ is the system response and $y_m$ that of the model, then the basis for the parameter optimization design of the controller, is the minimization of the performance index

$$J = \int_0^\infty (e^2 + \beta e^2) dt$$

(1.1)

where $e$ is the error between model and system responses and $\beta$ is a positive constant. If more than one output is involved, let $\mathbf{y}_s$ be the system output vector and $\mathbf{y}_m$ that of the model. The error is then defined by

$$e = \mathbf{y}_s - \mathbf{y}_m$$

(1.2)

and the performance index could be formulated as
where \( W_e \) and \( W_r \) are error weighting matrices.

In some cases it may be required that restrictions be placed on the magnitude of the control signals, limiting the amount of elevator deflection for instance in affecting aircraft longitudinal control. If \( u \) represents these control signals, then the performance index to be minimized could be formulated as

\[
J = \int_0^\infty [e^T W_e e + \beta e^T W_r \dot{e}] dt
\]

In the following chapters an algorithm is developed which adjusts the free parameters (gains and characteristic frequencies) of fixed-configuration, linear-invariant systems to optimum values so as to make the system response track as closely as possible a specified nominal response as measured by the minimum value of a performance index given in Eqns. (1.1), (1.3), or (1.4). Further a program for the practical implementation of this algorithm which involves minimal preprocessing of the model and system configuration and other requisite data is described.
CHAPTER 2

Control System Representation and Optimization Algorithm

In this chapter the form of the equations to represent the system configuration is presented along with the motivation for such a representation. The equations for the parameter optimization algorithm are next derived and the various quantities involved therein are defined. Finally the sequence of the algorithm logic is documented.

2.1 Control System Representation:

The equations of the parameter optimization algorithm are derived for a system configuration input in the form

\[
\dot{x}_s = Nx_s + Pb + Q'c
\]  
\[b = Rx_s + Sb + Tc + \sum_{i=1}^{n}(E_i x_s + H_i b)K_i \]  
\[y_s = S_s x_s \]

\(x_s\) is the system state vector, \(b\) the vector of bus variables, \(c\) the commands and \(y_s\) the system output vector. \(M, N, P, Q', R, S\) and \(T\) are system matrices. \(S_s\) is the output matrix, while \(E\) and \(H\) are three dimensional arrays relating the bus variables to the state and bus variables respectively through controller adjustable parameters \(K_i\) for \(i = 1, 2, \ldots n\).
The elements of the matrices in (2.1), and (2.2), for the system representation are usually considerably easier to obtain from an actual system than a single state equation and involve less manipulation of the system configuration, particularly if the plant dynamics are specified. An example is the format for the equations of flight vehicle longitudinal dynamics [2]. As will be indicated, the representation (2.1), and (2.2) allows the system configuration to be input in a straightforward manner.

Consider the basic longitudinal flight control system of Fig. 2.1. In this instance, [2], the missile dynamics would be described by equations of the form

\[ \dot{Mx} = Nx + Q\delta_e \]  \hspace{1cm} (2.4 a)

where \( \delta_e \) is the control surface (elevator) deflection. \( M \) is not an identity matrix but is a simple function of the missile aerodynamic stability derivatives. If the form
\[ \dot{x} = Fx + G\dot{\theta} \] were used a large ammount of preprocessing of input data would be required.

Consider the following set of equations for the longitudinal dynamics of a jet transport in straight and level flight [2].

\begin{align*}
13.78'\dot{u} + 0.88'u - 0.392'\alpha + 0.74 \theta &= 0 \\
1.48'u + 13.78'\dot{\alpha} + 4.46'\alpha - 13.78\dot{\theta} &= -0.246 \delta_e \\
0.552'\dot{\alpha} + 0.619'\alpha + 0.514 \dot{\theta} + 0.192 \dot{\delta} &= -0.710 \delta_e \\
\dot{\theta} &= \dot{q} \\
h &= \theta - '\alpha \\
\end{align*}

(2.4b)

The definition of the various variables is given below. Noting the form of equation (2.4a), and the appearance of \( \ddot{\theta} \) in the third equation of (2.4b), above; this requires that a new variable \( q \) be defined as \( \dot{\theta} = q \). Other higher derivatives may be similarly replaced. The equations (2.4b), can now be written as

\begin{align*}
13.78'\ddot{u} + 0.88'\dot{u} - 0.392'\ddot{\alpha} + 0.74 \ddot{\theta} &= 0 \\
1.48'\ddot{u} + 13.78'\ddot{\alpha} + 4.46'\dot{\alpha} - 13.78\ddot{\theta} &= -0.246 \delta_e \\
0.0552'\ddot{\alpha} + 0.619'\dot{\alpha} + 0.514 \ddot{\theta} + 0.192 \ddot{\delta} &= -0.710 \delta_e \\
\dot{\theta} &= q \\
h &= \theta - '\alpha \\
\end{align*}

(2.4c)
The state variables for the vehicle dynamics are

\[
\begin{align*}
'\text{u} & \quad \text{non dimensional perturbation in forward speed} \\
'\alpha & \quad \text{angle of attack perturbation (rads)} \\
\theta & \quad \text{pitch angle perturbation (rads)} \\
q & \quad \text{pitch rate (rads/sec)} \\
h & \quad \text{non dimensional altitude perturbation}
\end{align*}
\]

The elevator deflection is denoted by \( \delta_e \).

If the longitudinal controller is of the form shown in Fig. 2.2a with time constant \( \tau \) and \( \alpha \) as free design parameters, two additional state variables \( \delta_e \) and \( x_7 \) indicated in Fig. 2.2b, are introduced.

The actuator and lead-lag are described by the equations

\[
\begin{align*}
\dot{b}_1 &= c_1 - K_3 \theta \\
\dot{b}_2 &= -b_1 + K_1 b_1 \\
\dot{b}_3 &= K_2 b_2 - K_2 x_7 \\
\dot{b}_4 &= K_1 b_1 - x_7 \\
\dot{x}_7 &= b_3 \\
\dot{x}_6 &= -\frac{x_6}{\tau_a} + \frac{b_4}{\tau_a}
\end{align*}
\]

(2.5a) (2.5b)
Fig. 2.2a Block Diagram of Simplified Controller

\[ \text{ELEVATOR COMMAND} \]

\[ c_1 \text{ COMMAND} \]

\[ + \]

\[ - \]

\[ K_3 \]

\[ \theta \text{ PITCH ANGLE} \]

\[ \frac{at_s + 1}{ts + 1} \]

\[ \frac{1}{ta^s + 1} \]

\[ \delta_e \]

\[ \delta_e = x_6 \]

\[ \text{ELEVATOR DEFLECTION} \]

\[ b_1 = K_1 \]

\[ b_2 = K_1 - 1 \]

\[ b_3 = K_2 \]

\[ b_4 = \delta_e \]

\[ x_7 = \frac{1}{s} \]

\[ K_1 = \alpha; \ \alpha > 1 \]

\[ K_2 = \frac{1}{\tau} \]

Fig. 2.2b Block Diagram of Lead-lag Illustrating Use of Bus Variables

14
The equations (2.5a) are of the form (2.2), while (2.5b), together with (2.4b and c), are of the form (2.1). Therefore given the plant dynamics the form of the equations (2.1), and (2.2), makes it possible to augment the state vector to include state variables added by the controller without any redefinition of the plant variables.

From equation (2.2), it is seen that the bus variables are linear functions of the gains. This requires that E and H be at most three dimensional arrays. If a single state equation is used to describe the total system, the state variables would in general be products of the gains. This, besides involving the additional preprocessing implied in reducing (2.1), and (2.2), to a single equation would also require the definition of four and possibly five dimensional arrays to input the combinations of gains involved. This would provide considerable scope for erroneous inputing of data. Also in the representation of (2.1) and (2.2) most of the elements of each matrix would be zero. The namelist option described in Appendix A provides a convenient scheme for inputing only the non zero elements.

For the example under consideration with

\[ x_6^T = [u \, \alpha \, q \, h \, x_6 \, x_7] \]

\[ b^T = [b_1 \, b_2 \, b_3 \, b_4] \]
The non zero elements can be written down by inspection.

For equation 2.1, these are:

1. **M matrix** This is a 7x7 matrix

   \[ M_{11} = 13.78, M_{22} = 13.78, M_{23} = -13.78, M_{32} = 0.0552 \]
   \[ M_{33} = 0.192, M_{34} = 0.514, M_{43} = 1, M_{55} = 1, M_{66} = 1, \]
   \[ M_{77} = 1 \]

2. **N matrix** (7x7)

   \[ N_{11} = -0.088, N_{12} = 0.392, N_{13} = -0.74 \]
   \[ N_{21} = -1.48, N_{22} = 4.46, N_{26} = -0.246 \]
   \[ N_{31} = -0.619, N_{36} = -0.710 \]
   \[ N_{44} = 1 \]
   \[ N_{52} = -1, N_{53} = 1 \]
   \[ N_{66} = -\frac{1}{\tau a} \]

3. **P Matrix** (7x4)

   \[ P_{64} = \frac{1}{\tau a}, P_{73} = 1 \]

4. **Q matrix** (7x1) is a null matrix

For equation 2.2 the non zero elements are
1. R matrix (4×7)

\[ R_{47} = -1 \]

2. S matrix (4×4)

\[ S_{21} = -1 \]

3. T matrix (4×1)

\[ S_{11} = 1 \]

4. E (4×7×3)

\[ E_{133} = -1, \ E_{372} = -1 \]

5. H(4×4×3)

\[ H_{211} = 1, \ H_{322} = 1, \ H_{411} = 1 \]

The model is represented by the equations

\[ \dot{x}_m = F_m x_m + Q_m C \quad (2.6) \]

\[ y_m = S_m x_m \quad (2.7) \]

The error between the model and system response is defined as

\[ e = y_s - y_m \]
or

\[ e = S_s x_s + S_m x_m \quad (2.8b) \]

The control signals whose magnitudes are to be limited are related to the system state vector by

\[ u = S_u x_s \quad (2.9) \]

The command inputs \( \mathbf{c} \) are taken to be impulse inputs, i.e.

\[ \mathbf{c} = \delta(t) \mathbf{w} \quad (2.10) \]

where \( \mathbf{w} \) is a constant vector with components equal to the magnitude of the input impulses at time \( t = 0 \), and \( \delta(t) \) is the delta function.

2.2 Initial Condition Response

Substituting for \( \mathbf{b} \) from (2.2), into (2.1), and pre-multiplying by \( M^{-1} \), the system of equations can be written as

\[ \dot{x}_s = F_s x_s + Q_s \mathbf{c} \quad (2.11) \]

An augmented state vector is defined as
The equations (2.11), and (2.6), for the system and model can be combined and written as

\[ \dot{x} = Fx + Gc \]  
(2.13)

where the augmented system differential transition matrix

\[ F = \begin{bmatrix} F_s & 0 \\ \hline 0 & F_m \end{bmatrix} \]  
(2.14)

and the augmented control matrix is

\[ G = \begin{bmatrix} G_s \\ \hline G_m \end{bmatrix} \]  
(2.15)

Since the impulse response is being considered here, the solution to equation (2.13), with initial conditions \( x(0) = 0 \) can be obtained by solution of
\[
\dot{x} = Fx \tag{2.16}
\]

with initial conditions

\[
x(0) = Gw \tag{2.17}
\]

2.3 Formulation of J in terms of x

The performance index of equation (1.4), introduced in Chapter 1 is a measure of the error between system and nominal response, the first derivative of this error, and the magnitudes of control signals required to affect the system response. Using the definition of each of these quantities, J can be written in terms of x as follows:

From equation 2.8b, e and \( \dot{e} \) can be written as

\[
e = [S_s \mid S_m] x \tag{2.18}
\]

\[
\dot{e} = [S_s F_s \mid S_m F_m] x \tag{2.19}
\]

and from equation 2.9 u can be written as

\[
u = [S_u \mid 0] x \tag{2.20}
\]

In (2.20), above and in the discussion to follow [0] will be implicitly assumed to be an appropriately dimensioned
null matrix.

Substituting for \( e, \dot{e} \) and \( u \) from equation (2.18), (2.19), and (2.20), into equation (1.4), leads to:

\[
J = \int_0^\infty \begin{bmatrix} \dot{S} \\ S_m \end{bmatrix}^T \begin{bmatrix} S_e^T & F_s^T S_s^T \\ F_s^T S_e & S_m^T \end{bmatrix} \begin{bmatrix} W_e [S_s] S_m + F_s^T S_s^T e \beta W_e [S_s F_s] S_m F_m \end{bmatrix} \begin{bmatrix} S_u^T \\ S_m^T \end{bmatrix} + \begin{bmatrix} W_u [S_u] 0 \\ 0 \end{bmatrix} \ dt \tag{2.21}
\]

For the performance index.

Defining \( Q \) as the symmetric positive-semidefinite matrix.

\[
Q = \begin{bmatrix} S_e^T & F_s^T S_s^T \\ S_e^T S_m & S_m^T \end{bmatrix} W_e [S_s] S_m + \begin{bmatrix} W_e [S_s F_s] S_m F_m \end{bmatrix} \begin{bmatrix} S_u^T \\ S_m^T \end{bmatrix} + \begin{bmatrix} W_u [S_u] 0 \\ 0 \end{bmatrix} \tag{2.22}
\]

The expression for the performance index can be written more
compactly as

\[ J = \int_0^\infty |x(t)|^2_Q \, dt \] (2.23)

where \( x(t) \) is the solution to equation (2.16) with initial conditions (2.17). If the \( F \) matrix in eqn. (2.16), is a "stability" matrix, that is all its eigenvalues have negative real parts, then \( J \) can be evaluated as

\[ J = x^T(0)Px(0) ; \ x(0) = Gw \] (2.24)

where \( P \) is given by the solution to the matrix algebraic equation

\[ F^T P + PF = -Q \] (2.25)

as follows from a corollary to Lyapunov's theorem on the stability of linear invariant systems. [1,5,6]. It is noted in passing that the matrix \( P \) in equation (2.25), is symmetric. This fact leads to some computational simplification as indicated in the next chapter.

2.4 Derivation of Equations for the Parameter Adjustment Algorithm.

The control system is described by equations (2.1),
and (2.2). These are rewritten here as

\[ \dot{x}_s = M^{-1}N_x + M^{-1}P_b + M^{-1}Q'_c \]  \hspace{1cm} (2.26)

\[ [I-S- \sum_{i=1}^{n} K_iH_i]b = R_{x_s} + T_c + \left( \sum_{i=1}^{n} K_iE_i \right)x_s \] \hspace{1cm} (2.27)

define

\[ \tilde{N} = [I-S- \sum_{i=1}^{n} K_iH_i] \] \hspace{1cm} (2.28)

and

\[ \tilde{M} = \tilde{N}^{-1} \] \hspace{1cm} (2.29)

then

\[ b = \tilde{M}[R_{x_s} + T_c + \left( \sum_{i=1}^{n} K_iE_i \right)x_s] \] \hspace{1cm} (2.30)

substitution into (2.26), gives

\[ \dot{x}_s = [M^{-1}N + M^{-1}PR + M^{-1}PMT + \sum_{i=1}^{n} K_iE_i]x_s + [M^{-1}Q + M^{-1}Q'_c] \] \hspace{1cm} (2.31)

This is of the form of equation (2.11), with \( F_s \) and \( Q_s \) defined as

\[ F_s = [M^{-1}N + M^{-1}PR + M^{-1}PMT + \sum_{i=1}^{n} K_iE_i] \] \hspace{1cm} (2.32)
Also required are the gradients of $F_s$ and $Q_s$ with respect to the $i^{th}$ parameter $K_i$. Differentiating (2.32) with respect to $K_i$, using (2.28) and noting that

$$\frac{\partial M}{\partial K_i} = -N^{-1} \frac{\partial N}{\partial K_i} N^{-1} = -M \frac{\partial N}{\partial K_i} M = MH_1M$$ \hspace{1cm} (2.34)

then

$$\frac{\partial F_s}{\partial K_i} = M^{-1} P M H_i [M R + M \sum_{i=1}^{n} K_i E_i] + M^{-1} P M E_i$$ \hspace{1cm} (2.35)

Similarly

$$\frac{\partial Q_s}{\partial K_i} = M^{-1} P M H_i M T$$ \hspace{1cm} (2.36)

The gradients of the augmented system matrices can be written down (observing that the model parameters are invariant) as

$$\frac{\partial F}{\partial K_i} = \begin{bmatrix} \frac{\partial F_s}{\partial K_i} & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}$$ \hspace{1cm} (2.37a)

$$\frac{\partial G}{\partial K_i} = \begin{bmatrix} \frac{\partial Q_s}{\partial K_i} \\ \vdots \\ 0 \end{bmatrix}$$ \hspace{1cm} (2.37b)

The gradient of the $Q$ matrix defined in equation (2.22)
with respect to \( K_i \) is

\[
\frac{\partial Q}{\partial K_i} = \begin{bmatrix}
\frac{\partial F_s^T}{\partial K_i} S_s^T & \frac{F_s^T S_s}{e}
\end{bmatrix} \beta W \begin{bmatrix}
[S_s F_s | S_{m^T} m_m]
\end{bmatrix} + \begin{bmatrix}
F_s^T S_s^T
\end{bmatrix} \beta W \begin{bmatrix}
[S_s \frac{\partial F_s}{\partial K_i} | 0]
\end{bmatrix} \tag{2.38a}
\]

If the weighting matrix \( W_e \) is a diagonal matrix (or symmetric) then,

\[
\frac{\partial Q}{\partial K_i} = \begin{bmatrix}
\left( \frac{\partial F_s}{\partial K_i} \right)^T S_s^T
\end{bmatrix} \beta W_e \begin{bmatrix}
[S_s F_s | S_{m^T} m_m]
\end{bmatrix} + \begin{bmatrix}
\left( \frac{\partial F_s}{\partial K_i} \right)^T S_s^T
\end{bmatrix} \beta W_e \begin{bmatrix}
[S_s \frac{\partial F_s}{\partial K_i} | 0]
\end{bmatrix} \tag{2.38b}
\]

2.5 Parameter Adjustment Algorithm—Sequence Logic

The minimization of the performance index is carried out by first-order gradient techniques. The incremental changes in the parameter values being made so as to produce the maximum decrease in the magnitude of the performance index at any instant. This is accomplished by selecting the incremental changes proportional to the negative gradient of \( J \) at the operating point.
The incremental change of the $i^{th}$ parameter $K_i$ is

$$\Delta K_i = -\lambda \frac{\partial J}{\partial K_i} \quad \text{for } i = 1, 2, \ldots n \quad (2.39)$$

where $\lambda$ is a positive constant. The gradient of $J$ with respect to $K_i$ is obtained by differentiating $(2.24)$, with respect to $K_i$, that is,

$$\frac{\partial J}{\partial K_i} = \frac{\partial x^T(0)}{\partial K_i} P x(0) + x(0) \frac{\partial P}{\partial K_i} x(0) + x^T(0) P \frac{\partial x(0)}{\partial K_i}$$

(2.40a)

or since $P$ is symmetric

$$\frac{\partial J}{\partial K_i} = x^T(0) \frac{\partial P}{\partial K_i} x(0) + 2 \frac{\partial x^T(0)}{\partial K_i} P x(0) \quad (2.40b)$$

$P$ and $\frac{\partial P}{\partial K_i}$ are evaluated successively from

$$F^T P + P F^T = -Q \quad (2.41)$$

and

$$F^T \frac{\partial P}{\partial K_i} + \frac{\partial P}{\partial K_i} F = -\frac{\partial Q}{\partial K_i} - \left[ \left( \frac{\partial F}{\partial K_i} \right)^T P + P \left( \frac{\partial F}{\partial K_i} \right) \right] \quad (2.42)$$

The gradient of the initial condition vector is given by
Once $\frac{\partial x(0)}{\partial K_i} \frac{\partial P}{\partial K_i} \frac{\partial x(0)}{\partial K_i}$ have been evaluated $\frac{\partial J}{\partial K_i}$ and $\Delta K_i$ for $i = 1, 2, \ldots, n$ can be evaluated from equations (2.40b), and (2.39), respectively.

In summary then: the parameter adjustment algorithm proceeds as follows:

1. A set of initial parameter values $K_i$, $i=1, 2, \ldots, n$ is selected. Assuming the resulting system is stable [the asymptotic stability is checked using the Routh Criterion], the next step is
2. Determination of the matrices $F$ and $Q$ from equation (2.14), and (2.22).
3. The matrix algebraic equation (2.25), is solved for $P$
4. The gradients of $F$, $Q$ and $x(0)$ with respect to $K_i$ are obtained from equations (2.37a), (2.38a), and (2.43), respectively.
5. Using $F$ and $Q$ determined in step 2 and $P$ determined in step 3, equation (2.42), is solved for $\frac{\partial P}{\partial K_i}$.
6. $\frac{\partial J}{\partial K_i}$ is evaluated from (2.40b), and $\Delta K_i$ from (2.39).
7. Steps 4 through 6 are repeated for $i=2, 3, \ldots, n$
8. The predicted change in the performance index $J$ is then evaluated according to

\[
\frac{\partial x(0)}{\partial K_i} = \frac{\partial G}{\partial K_i} W
\]
\[ \Delta J = -\lambda \left[ \left( \frac{\partial J}{\partial K_1} \right)^2 + \left( \frac{\partial J}{\partial K_2} \right)^2 + \ldots + \left( \frac{\partial J}{\partial K_n} \right)^2 \right] \]

9. The procedure from steps 2 through 8 is repeated using the revised values of the parameters until

\[ \left[ \left( \frac{\partial J}{\partial K_1} \right)^2 + \left( \frac{\partial J}{\partial K_2} \right)^2 + \ldots + \left( \frac{\partial J}{\partial K_n} \right)^2 \right] \]

is very small.
In this chapter a computer program scheme for implementing the algorithm developed in the previous chapter is outlined. Fig. 3.1 presents the overall flow of the program logic while Fig. 3.2 is the flow diagram for the solution of the matrix algebraic equation \( AX + XB = -C \).

3.1 The overall program

1. Input. The system configuration represented by equations 2.1, 2.2 and 2.3 and the model are input by inputting only the non-zero elements of each of the system and model matrices, the rest being automatically initialized to zeros. All data is input using the namelist option. The data input format is described in Appendix A.

2. The augmented F matrix is evaluated according to the equation (2.37), derived in Chapter 2.

3. The characteristic polynomial of the F matrix is obtained in two stages. The matrix is transformed using subroutine SMLTRN (A subroutine in the program for the solution of the matrix equation) to a block diagonal form. The transformation method used is that due to A.M. Danilevskii [3]. In the general case the process will degenerate so that the transformed matrix will consist of more than one
diagonal block.

The general form of the transformed matrix [4] is

\[
A = \begin{bmatrix}
L_1 & M_{12} & \cdots & M_{1r} \\
\vdots & L_2 & \cdots & M_{2r} \\
& \cdots & \cdots & \cdots \\
\vdots & & & \vdots \\
& & & \vdots \\
M_{r1} & & & \vdots \\
\end{bmatrix} \quad (3.1)
\]

where the \(L_i\)'s are the companion matrices

\[
L_i = \begin{bmatrix}
0 & 0 & \cdots & \cdots & \cdots & 0 & a_{1i} \\
1 & 0 & \cdots & \cdots & \cdots & 0 & a_{2i} \\
0 & 1 & \cdots & \cdots & \cdots & 0 & a_{3i} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & 0 & 1 & a_{ki} \\
\end{bmatrix} \quad (3.2)
\]

and the off diagonal blocks \(M_{li}\) are of the form

\[
M_{li} = \begin{bmatrix}
0 & 0 & \cdots & \cdots & \cdots & 0 & p_{1i} \\
0 & 0 & \cdots & \cdots & \cdots & 0 & p_{2i} \\
0 & 0 & \cdots & \cdots & \cdots & 0 & p_{3i} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & \cdots & \cdots & 0 & p_{ki} \\
\end{bmatrix} \quad (3.3)
\]
Due to the quasi-triangular character of $A$, its characteristic polynomial is simply the product of the characteristic polynomials of the diagonal blocks. Furthermore the characteristic polynomial is invariant under the similarity transformation. Subroutine SCAN1 identifies the characteristic polynomials of the individual blocks and forms the product to give the characteristic polynomial of the $F$ matrix.

4. The stability of the $F$ matrix is next checked by means of subroutine RSCHEK which is simply an implementation of the Routh criterion and works on the characteristic polynomial of the $F$ matrix. Only asymptotic stability is checked here. The condition code $ICODE = 0$ indicates a stability matrix while $ICODE = 1$ indicates a non-stability matrix.

5. The evaluation of the $G$ and $Q$ matrices as well as the augmented initial condition vector proceeds according to the equation (2.15), and (2.22), respectively of the previous chapter.

6. The solution of the matrix equation

$$F^T P + PF = -Q$$

(3.4)

for $P$ is a special case of the solution to

$$AX + XB = -C$$

(3.5)

The program flow for this stage is described in greater
Fig. 3.1 Overall Program Flow Logic
detail in the next section. At this point the following is noted.

In order to avoid the evaluation and inversion of matrix polynomials of relatively higher degrees involving powers of matrices of relatively larger dimensions, it may be more efficient to reduce the orders of the matrix algebraic equation to be solved. This may be done by partitioning the system of equation (3.4), as indicated below.

\[
\begin{bmatrix}
F_s^T & 0 \\
0 & F_m^T
\end{bmatrix} + 
\begin{bmatrix}
P_{11} & P_{12} \\
P_{21} & P_{22}
\end{bmatrix} + 
\begin{bmatrix}
P_{11} & P_{12} \\
P_{21} & P_{22}
\end{bmatrix} \begin{bmatrix}
F_s & 0 \\
0 & F_m
\end{bmatrix}
\]

\[
\begin{bmatrix}
-Q_{11} & -Q_{12} \\
-Q_{21} & -Q_{22}
\end{bmatrix}
\]  \hspace{1cm} (3.5)

Therefore the solution to (3.4), can be obtained by solving the set of equations

\[
\begin{align*}
F_s^T P_{11} + P_{11} F_s &= -Q_{11} \\
F_s^T P_{12} + P_{12} F_m &= -Q_{12} \\
F_m^T P_{21} + P_{21} F_s &= -Q_{21} \\
F_m^T P_{22} + P_{22} F_m &= -Q_{22}
\end{align*}
\]  \hspace{1cm} (3.6)
Each of these equations could be solved by the same procedure. It was shown earlier that $Q$ is a symmetric matrix. From (3.4), $P$ is therefore symmetric. The second and third equations of (3.6), are therefore identical. If the nominal transfer function is taken to be a simple first order lag, additional simplification is possible by observing that the last equation of (3.6), becomes a scalar equation with $P_{22}$ given by

\[ P_{22} = -Q_{22}/2F_m \]  

(3.7)

and the 2\textsuperscript{nd} and 3\textsuperscript{rd} equations reduce to linear algebraic equations, denoting $P_{12}$ and $Q_{12}$ by $P'_{12}$ and $Q'_{12}$, the solution is given by

\[ P'_{12} = [F_s^T + IF_m]^{-1}Q'_{12} \]  

(3.8)

The solution to the first equation is obtained by the method of the next section.

7. The next stage is the computation of the various gradient matrices with respect to the parameters $K_i$, $i=1, \ldots, n$ and then the gradient of the performance index. The parameter values are revised typically by about 10% of their original values, the original values being held, should the revised parameter lead to an unstable augmented
F matrix. If indeed the revised set of parameters lead to an unstable system as determined by RSCHENK in the subsequent iteration, the parameter values are revised by 50% of the old revision. The process of successive revision of parameter values with 50% reduction in step size could be terminated for maximum number of iterations exceeding say, four.

8. The change in the performance index is easily evaluated from the gradients. The magnitude of the change is used as one stopping condition. Limiting the number of iterations if this proves excessive is also used.

3.2 The solution of the equation $AX + XB = -C$.

It was indicated earlier that solution of equations of the type (3.4), is involved at each iteration of the parameter optimization algorithm. Equation (2.4), is a special case of the equation (3.5). A program for the solution of the equation (3.5) is described in the following paragraphs. The details of the algorithm on which MSOLVE is based is presented in [4]. The solution proceeds essentially as follows.

Consider the equation

$$AX + XB = -C$$

(3.9)

where A and B are arbitrary square matrices, of dimensions $n \times n$ and $m \times m$ respectively. C is a known $n \times n$ matrix while X is the unknown $n \times m$ matrix. The necessary and
sufficient condition that the system (3.9) possess a unique solution is that the matrices \(-A\) and \(B\) not have common eigenvalues [9]. In terms of the computation procedure, violation of this condition leads to a divide check, a sentinel and indication of common eigenvalues is therefore included. For the specialized case

\[
F^T P + PF = -Q \quad (3.10)
\]

a unique solution exists if and only if all eigenvalues of \(F\) have negative real parts, or \(F\) is a stability matrix [1].

According to [4], a solution is assumed of the form

\[
X = UV^{-1} \quad (3.11)
\]

where \(V^{-1}\) is a square matrix of dimension \(m \times m\) and \(U\) is rectangular of dimension \(n \times m\).

Substitution into (3.9) leads to

\[
AU + CV = -UA \quad (3.12)
\]

where

\[
\Lambda = V^{-1}BV \quad (3.13)
\]

Referring to Fig. 3.2, the first step is then to obtain
Fig. 3.2 Program Flow Logic for Solution of Matrix Algebraic Equation $AX+XB=-C$
A by a sequence of similarity transformations. The general form of the $\Lambda$ matrix is given by equations (3.1), (3.2), and (3.3). The similarity transformation matrices $V$ and $V^{-1}$ are also generated by SMLTRN and held.

Writing $U$ and $CV$ in 3.12 as

$$U = [u_1 | u_2 | \ldots | u_m]$$

$$CV = [v_1 | v_2 | \ldots | v_m] \quad (3.14)$$

Expanding (3.12), by columns and using (3.2) and (3.3) leads to the following set of equations in terms of the columns of $U$.

$$Au_1 + v_1 = -u_2$$

$$Au_2 + v_2 = -u_3$$

$$\vdots$$

$$Au_{k-1} + v_{k-1} = -u_k$$

$$Au_k + v_k = a_{11}u_1 + a_{21}u_2 + \ldots + a_{k1}u_k$$

$$Au_{k+1} + v_{k+1} = -u_{k+2}$$

$$\vdots$$

$$Au_{k+l} + v_{k+l} = -p_{12}u_1 - p_{22}u_2 - \ldots - p_{k2}u_k + a_{12}u_{k+1} + a_{22}u_{k+2} + \ldots + a_{k2}u_{k+l}$$

$$\vdots$$

$$\vdots$$
\[ \begin{align*}
A_{m-q+1}u_{m-q+1} & + v_{m-q-1} = -u_{m-q+2} \\
& \vdots \\
A_u & + v_m = -P_1u_1 - P_2u_2 - \cdots - P_{m-q}u_{m-q} + a_{1r}u_{m-q+1} \\
& \quad + a_{2r}u_{m-q+2} + \cdots + a_{qr}u_m \\
\text{(3.15)}
\end{align*} \]

In the above, the dimension of the first diagonal block is \( k \) that of the 2nd \( \ell \) and that of the \( r \)th is \( q \). Taking \( u_1, u_{k+1}, \ldots, u_{k+\ell+1}, \ldots, u_{m-q+1} \) as unknowns and substituting for the other vectors of \( u \) in terms of these into \( k \)th \((k+\ell)\)th and \( m \)th equation of (3.15), leads to

\[
\frac{\begin{align*}
[-A]^{k+a_{k1}} & \frac{(-A)^{k-1} + \cdots + a_{21}(-A) + a_{11}I]}{u_1} \\
& = \sum_{j=1}^{k} \sum_{h=j+1}^{k+1} a_{h1}(-A)^{h-(j+1)}v_j \\
[(-A) + a_{k2}(-A)^{k-1} + \cdots + a_{22}(-A) + a_{12}I] & u_{k+1} \\
& = \sum_{j=1}^{\ell} \sum_{h=j+1}^{\ell+1} a_{h2}(-A)^{h-(j+1)}v_j + \sum_{i=1}^{k} p_{12}u_i \\
& \vdots \\
[(-A)^q+a_{qr}(-A)^{q-1} + \cdots + a_{2r}(-A) + a_{1r}I] & u_{m-q+1} \\
& = \sum_{j=1}^{q} \sum_{h=j+1}^{q+1} a_{hr}(-A)^{h-(j+1)}v_{j+m-q} + \sum_{i=1}^{m-q} p_{1i}u_i \\
\text{(3.16)}
\end{align*}}
\]
In particular if $\Lambda$ is composed of a single companion block, then

$$[p_B(-A)]u_1 = \sum_{j=1}^{m} \sum_{h=j+1}^{m+1} a_h(-A)^{h-(j+1)} v_j$$

(3.17)

The polynomials on the left hand side of the equation in (3.16), are simply the characteristic polynomials of the individual companion blocks with $\lambda$ replaced by $(-A)$. The right hand side contains the elements of the last columns of each of the off diagonal $M$ blocks. Referring to Fig. (3.2), the $\Lambda$ matrix is next scanned to identify and hold the relevant arrays to be used in (3.16).

The first equation of (3.16), is next solved for $u_1, u_2, u_3 \ldots u_k$ are then determined from (3.15). These in turn are used in the second equation of (3.16), to obtain $u_{k+1}$ and so on for the $r$ blocks.

The solution for $u_1, u_{k+1} \ldots u_{m-q+1}$ involves the inversion of the matrix polynomials on the left hand side of (3.16). The inversion scheme uses the characteristic polynomial of the $(-A)$ matrix, accordingly this is next evaluated as indicated in Figure (3.2). The inversion scheme is based on the following.

Given two polynomials.
\[ P(x) = x^m + a_m x^{m-1} + a_{m-1} x^{m-2} + \ldots + a_2 x + a_1 \]

\[ Q(x) = x^p + b_p x^{p-1} + b_{p-1} x^{p-2} + \ldots + b_2 x + b_1 \]  \hspace{1cm} (3.18)

which are relatively co-prime, then [4] there exist polynomials \( \tilde{P}(x) \) and \( \tilde{Q}(x) \), such that

\[ \tilde{Q}(x) P(x) + \tilde{P}(x) Q(x) = k \]  \hspace{1cm} (3.19)

If \( P(\lambda) \) is the characteristic polynomial of matrix \((-A)\)

\[ \tilde{P}(-A) Q(-A) = kI \]  \hspace{1cm} (3.20)

Therefore if \( Q(-A) \) is the polynomial to be inverted,

\[ Q(-A)^{-1} = \tilde{P}(-A) \]  \hspace{1cm} (3.21)

In Fig. 3.2, MPINV is a subroutine which evaluates \( \tilde{P}(-A) \) iteratively using the Euclidean Algorithm.

The required polynomials in the equation (3.16), having been evaluated, these together with the set (3.15), are solved to obtain the matrix \( U \), from which \( X \) is obtained from (3.11).

The preliminary solution so obtained can be improved by iteration in the following manner. An error matrix \( E \) is generated as
\[ E = AX + XB + C \quad (3.22) \]

The last part of the solution routine, MUSOLV, is used to solve

\[ AY + YB = -E \quad (3.23) \]

The revised solution is then

\[ X = X + Y \]

which can be revised again. The number of iterations required will depend on the dimensions of the A and B matrices. If for instance both are three by three square with elements of the order unity, underflows could result if the number of iterations specified exceeds say 2 or 3.
CHAPTER 4
Design Via Parameter Optimization

The preceding chapters presented the analytical development of the parameter optimization algorithm, and outlined a program for its practical implementation.

In this chapter the basic design procedure is illustrated by means of a single input-single output system. The model used is a simple first order lag. However as indicated in Chapter 3, the scheme could be extended to include higher order models. In fact a higher order model would certainly be required if a specification envelope is to be set up around the nominal response.

4.1 System and Model Configuration Describing Equations

The closed loop systems considered in the following discussion are shown in Figs. 4.1a, and 4.2.

Referring to Fig. 4.1a, the system state variables are $x_1$, $x_2$ and $x_3$ while $b_1$ and $b_2$ are bus variables. These are indicated on Fig. 4.1a. The model and system both have d.c. gains of unity for step inputs.

The equations describing the system and model are:

$$
\begin{align*}
\dot{x}_1 &= -2x_1 + b_2 \\
\dot{x}_2 &= x_1 \\
\dot{x}_3 &= -x_3 + x_2
\end{align*}
$$

(4.1a)
Fig. 4.1a Block Diagram of Closed Loop System with Position Feedback. \( K \) is a Free Design Parameter.

\[
c_1 = b_1 \quad \begin{array}{c}
  \hspace{0.5cm} \quad K \quad \frac{1}{s+2} \quad x_1 \quad \frac{1}{s} \quad x_2 \quad \frac{1}{s+1} \quad x_3 = y_s
\end{array}
\]

Fig. 4.1b The Nominal Transfer Function

\[
c_1 \quad \frac{1}{(3s+1)} \quad y_m
\]

Fig. 4.2 Closed Loop System with Rate Feedback Added \( K \) and \( \tau \) are free parameters.
\[ b_1 = c_1 \]
\[ b_2 = Kb_1 - Kx_3 \]  \hspace{1cm} (4.1b)

The model is described by

\[ \dot{x}_m = -0.33 x_m + 0.33 c_1 \]  \hspace{1cm} (4.2)

The system equations can be written in the form 2.1 and 2.2 as

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x_s \\
\dot{x}_s \\
0
\end{bmatrix} = \begin{bmatrix}
-2 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & -1
\end{bmatrix} \begin{bmatrix}
x_s \\
\dot{x}_s \\
0
\end{bmatrix} + \begin{bmatrix}
0 & 1 \\
0 & 0 \\
0 & 0
\end{bmatrix} b + \begin{bmatrix}
ob \\
0 \\
0
\end{bmatrix} c \hspace{1cm} (4.3a)
\]

\[
b = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} x_s + \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} x_s + \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} b + \begin{bmatrix}
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0
\end{bmatrix} \begin{bmatrix}
c + K_1 \\
0 \\
0
\end{bmatrix} b + K_1 \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} x_s \hspace{1cm} (4.3b)
\]

where

\[ x_s^T = [x_1 \ x_2 \ x_3] \; \text{and} \; b^T = [b_1 \ b_2] \]

The system and model outputs are defined by

\[ y_s = \begin{bmatrix}
0 & 0 & 1
\end{bmatrix} x_s \]  \hspace{1cm} (4.4a)
\[ y_m = -x_m \]  

(4.4b)

No control signals are defined to be included in the cost functional. The system of equations 4.2, 4.3 and 4.4 are input using the namelist option, the input data format for which is described in Appendix A. Other input data include

1. The number of free design parameter \( IGAIN \). \( IGAIN = 1 \) for the system of Fig. 4.1a and 2 for Fig. 4.2.
2. The dimensions of the various vectors \( KE = 1, KM = 1, KC = 1, KX = 3, KX1 = 4, KB = 2 \) for the system of Fig. 4.1a where the number of bus variables is 2 and \( KB = 3 \) for the system of Fig. 4.2.
3. The initial values of the parameter(s). \( PARAM(1) = 1.4 \) for the first system while \( PARAM(1) = 1, PARAM(2) = 1 \) for the system with rate feedback added.
4. The step size for parameter revision, \( STEP \).
5. The error weighting matrices \( W1, W2 \) and \( W3 \) and the positive constant \( BETA \), taken in these examples \( = 0.15 \).

For the first system the initial value of \( K \) was taken to be 1.4. The adjusted value was 0.32. The response for these two values of loop gain are shown in Fig. 4.3a. As seen from this figure, compared to the nominal, the response with position feedback alone is somewhat slow.
The next step was the addition of rate feedback as shown in the block diagram of Fig. (4.2). There are now two free parameters $K$ and $\tau$ which allow independent adjustment of the maximum value and speed of the response. Initial parameter values were taken here to be unity for both $K$ and $\tau$. The values as adjusted by the program were 0.795 for $K$ and 1.26 for $\tau$.

Fig. 4.3b, is a plot of the closed loop system response with rate feedback. Comparison with Fig. 4.3a indicates the improvement in system response relative to nominal. A plot of the position errors for the two system is shown in Fig. 4.3c.
Fig. 4.3a CLOSED LOOP SYSTEM RESPONSE WITHOUT RATE FEEDBACK

Fig. 4.3b CLOSED LOOP SYSTEM RESPONSE WITH RATE FEEDBACK

Fig. 4.3c SYSTEM RESPONSE ERROR WITH AND WITHOUT RATE FEEDBACK
5.1 Conclusions

An algorithm for parameter optimization of linear-invariant control systems has been developed. A computer program which implements this algorithm has also been developed and illustrated by means of an example.

The performance index is evaluated using a corollary to Lyapunov's theorem on the stability of linear-invariant systems, rather than by integration of the system and model state equations and subsequent integration of the error at each stage. The aforementioned theorem makes it unnecessary at any stage to integrate the state equations. A result of this scheme was the development of a program for solution of the matrix algebraic equation $AX + XB = -C$ based on the algorithm presented in Ref. [4].

A minimal amount of preprocessing is required to input the system and model configurations as well as the other requisite data. In this context, the defining of bus variables, the form of the equations (2.1), and (2.2), and the namelist option input data format are particularly useful. The state variable selection is not restricted to phase variables. It is possible to select state variables as the outputs of the various integrators as well as physical variables directly from the plant dynamics. Variables of
the first type were a convenient selection for the illustrative example, while the equations for aircraft longitudinal dynamics for instance affords an example of the second kind.

Finally the implementation of the Routh stability criterion enables the program to indicate when the parameter revision sequence leads to unstable systems.

5.2 Recommendations

In its present form the program is restricted to the use of first order models, but as indicated in Chapter 3, the scheme could be modified to include higher order models. This would also make it possible if necessary to use a two dimensional error vector in the cost functional.

A fixed step size was used with a first order gradient method. The step size was kept small to avoid nonlinearities as well as overshooting of the minimum. However to improve the convergence rate, it would be more efficient to vary the step size in the sequence; possibly by including in the parameter revision logic the magnitude of the current parameter values.
APPENDIX A

This section describes the input data format used. All data is input using the namelist option. The necessary data cards for a specific problem are indicated below in the order in which they appear. The namelist option makes it possible to input only the non-zero elements of each matrix or vector, the rest being initialized to zeros.

The system configuration is input first. These are the system matrices of equations 2.1 and 2.2. Each list starts in column 2.

&INPUTM SYSM(I,J) = numeric data,&END
&INPUTN SYSN(I,J) = numeric data,&END
&INPUTP SYSP(I,J) = numeric data,&END
&INPUTQ SYSQ(I,J) = numeric data,&END
&INPUTR R(I,J) = numeric data,&END
&INPUTS S(I,J) = numeric data,&END
&INPUTT T(I,J) = numeric data,&END
&INPUTH H(I,J) = numeric data,&END
&INPUTE E(I,J,K) = numeric data,&END

For the three dimensional arrays E and H in the above format, I corresponds to the bus variable on the left hand side of the equation 2.2, J to the system or bus variable on the right hand side of the equation and K to the parameter relating the two.
The model configuration or nominal transfer function is input by means of the following two lists.

&MODEF FMOD(I,J) = numeric data,&END
&MODEQ QMOD(I,J) = numeric data,&END

where FMOD and QMOD are the matrices $F_m$ and $Q_m$ respectively.

The system and model output matrices are listed as

&SYSOUT SYSS(I,J) = numeric data,&END
&MODOUT SMOD(I,J) = numeric data,&END

Note that in the above namelist, the output defined by SMOD is the negative of the model output.

The next three lists input the error weighting matrices $W_e$, $W'_e$, and the control signal weighting matrix $W_u$.

&WGTERW W1(I,J) = numeric data,&END
&WGTEDT W2(I,J) = numeric data,BETA=numeric value,&END
&WGTU W3(I,J) = numeric data,&END

The dimensions of the various vectors are input according to

&DIMEN KE= , KE= , KM= , KX1= , KC= , KB=,&END

where
KE is the dimension of the error vector \( \leq 2 \)

KX is the dimension of the system state vector \( \leq 10 \)

KM is the dimension of the model state vector \( \leq 5 \)

KX1 is the dimension of the augmented system = KX+KM

KC is the dimension of the command vector \( \leq 5 \)

KB is the dimension of the bus vector \( \leq 10 \)

The next list is

&GAIN IGAIN= ,PARAM(K)= ,STEP= ,&END

Here IGAIN is put equal to the number of free design parameters. PARAM(K) is the initial value of the \( K^\text{th} \) parameter \((K=1,2...IGAIN)\). STEP is the step size used for revision of the parameter values.

The matrix relating the control signals to the system state vector is SU. If any control signal is to be included in the performance index integral, the appropriate elements of the SU matrix are input as

&INPUTU SU(I,J) = numeric data,&END

If there is no penalty to be imposed on any of the control signals, then SU is a null matrix. The input list takes the form (as is the case for any null matrix)

&INPUTU &END.
The last list contains the magnitudes of the delta functions for the command variable vector, i.e.

\[ &\text{CONTRL VCOMM}(J) = , \&\text{END} \]

where \( J=1,2,...,KC \).
APPENDIX B

The program implementing the algorithm is listed on the following pages. The listing presents the following program packages.

1. The main program consisting of the algorithm sequence logic. The output consists of the parameter values at each iteration, the iterate number and changes in the performance index at each iteration. The program also lists the final adjusted parameter values and the corresponding system differential transition and control matrices.

2. MSOLVE: this interfaces the main program with the subroutines of the solution to the matrix algebraic equation. These subroutines are

   2a. SMLTRN: this program implements the Danilevskii Similarity transformation

   2b. SCAN1: identifies and stores the polynomials of the diagonal blocks of the A matrix. The characteristic polynomial is computed from these.

   2c. SCAN2: identifies and stores the various arrays of the B matrix for use in MPINV

   2d. MPINV: this subroutine evaluates the inverses of the various matrix polynomials using the Euclidian algorithm.
2e. **MUSOLV**: evaluates the matrix $U$ and forms the product $UV^{-1}$ to give the solution matrix $X$. $JWRITE$ is set to zero if a print out of the solution and error matrices is required. Otherwise $JWRITE = 1$.

The other subroutines included in the listing are

3. **RSCHEK**: this is an implementation of the Routh stability criterion and determines asymptotic stability. The condition code $ICODE = 0$ indicates a stable system, while $ICODE = 1$ indicates instability.

4. **MPDIV, MPNORM, MPMPY, MPSUB, MPMOVE**: for polynomial operations and **MXMULT** a variable dimension matrix multiplication subroutine.
REAL*8 SMG(20,20), X0(20), DELFS(10,10), DELQ(20,20), DELF(20,20),
1DELF1(20,20), P(20,20), TEMP3(10,5), DELGS(10,5), DELG(20,5), DELX(20),
3U(20,20), V(V20,20), VIN(20,20), A(20,20), B(20,20), XSLN(20,20),
3RHS(20,20), QINV(5,21), QB(5,21), SUM, SUM1, SIGMA, PI, DELJ,
4DELTAJ, RH1(10,10), RH2(10,5), RH2(10,5), RH2(10,5), RH2(10,5), RH2(10,5),
5SYSFT(10,10), SYSF2(10,10), DELP(20,20), FINV(10,10), DELK(10),
6PARM(10), C(20,20), C1(20,20), A1(20,20), B1(20,20), PAR(10), P11,
INTEGER*4 IDENT(5), IDIMQ(5), IDIMB(5), IDIMQ(5),
DOUBLE PRECISION SYSM(10,10), SYSP(10,10), SYSP(10,10),
1SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
2SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
3SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
4SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
5SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
6SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
7SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
8SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
9SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
10SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
11SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
12SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
13SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
14SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
15SYSN(10,10), SYSN(10,10), SYSN(10,10), SYSN(10,10),
READ(5, INPUTS)
READ(5, INPUTT)
READ(5, INPUTH)
READ(5, INPUTE)
READ(5, MODEF)
READ(5, MODEQ)
READ(5, SYSCUT)
READ(5, MODCUT)
READ(5, WGTERR)
READ(5, WGTEDT)
READ(5, WGTU)
READ(5, DIMEN)
READ(5, GAIN)
READ(5, INPUTU)
READ(5, CONTU)
PI=0.0
L4=20
L2=5
L1=10
ITTER=0
ITTER1=0
ISTOP=0
CALL SIMEQ(SYSM, XDOT, KK, SINV, X, IERR)
CALL MXMULT(SINV, SYSN, SYSN1, KK, KK, XX, M, N, L1, L1, L1, L1)
CALL MXMULT(SINV, SYSP, SYSPL, KK, KK, KB, M, N, L1, L1, L1, L1)
CALL MXMULT(SINV, SYSQ, SYSQ1, KK, XX, KC, M, N, L1, L1, L1, L1, L2)
5 DO 20 I=1, KB
   DO 20 J=1, KB
   IF(I.EQ.J) GO TO 10
   S1(I, J)=-S(I, J)
   GO TO 20
10 S1(I, J)=1.0-S(I, J)
20 CONTINUE
30 DO 40 K=1, IGAIN
   DO 40 I=1, KB
   DO 40 J=1, KB
HK(I,J)=H(I,J,K)*PARAM(K)

40 SL(I,J)=SL(I,J)-HK(I,J)
CALL SIMEQ(SL,XDOT,KB,SLIN,XX,IERR)
CALL MXMULT(SINV,K,R1,KB,KB,KX,M,N,L1,L1,L1,L1)
DO 50 I=1,KB
DO 50 J=1,KX
50 EK(I,J)=E(I,J,I)*PARAM(I)
IF(IGAIN.EQ.1)GO TO 61
DO 60 K=2,IGAIN
DO 60 I=1,KB
DO 60 J=1,KX
60 EK(I,J)=EK(I,J)+E(I,J,K)*PARAM(K)
61 CALL MXMULT(SYSP1,EK1,EK1,KB,KB,KX,M,N,L1,L1,L1,L1)
CALL MXMULT(SYSP1,EK1,SYSF,KX,KB,KB,KX,M,N,L1,L1,L1,L1)
DO 70 I=1,KX
DO 70 J=1,KX
70 SYSF(I,J)=SYSF(I,J)+TEMP(I,J)+SYSN1(I,J)
DO 100 I=1,KX
DO 100 J=1,KX
IF(J.GT.KX)GO TO 90
F(I,J)=SYSF(I,J)
F1(I,J)=F(I,J)
GO TO 100
90 F1(I,J)=0.0
F1(I,J)=0.0
100 CONTINUE
IR=KK+1
DO 120 I=IR,KX1
120 IR=120
DO 120 J=1,KX1
IF(J.LE.KX)GO TO 110
JJ=J-KX
II=I-KX
F(I,J)=FMOD(II,JJ)
F1(I,J)=F(I,J)
GO TO 120
110 F(I,J)=C.0
120 CONTINUE
C INVESTIGATE MODSYS ASYMPTOTIC STABILITY
C
CALL SMULTRN(F1,F2,F3,KX1,INDXF)
CALL SCAN1(F1,ARRAYF,IDIMF,KX1)
CALL RCSHEK(ARRAYF,IDIMF,ROUTH1,IDIMR1,ROUTH2,IDIMR2,ICODE)
DC 130 I=1,KX1
DC 130 J=1,KX1
130 FT(J,I)=F(1,J)
CALL MXMULT(SYSS,SYSF,SFS,KE,KK,KK,M,N,L2,L1,L1,L1)
CALL MXMULT(SMOD,FMOD,SMF,KE,KM,KM,KM,N,L2,L2,L2)
C F MATRIX PARTITION
C
DC 150 I=1,KE
DC 150 J=1,KX1
IF(J.GT.KX)GO TO 140
SF(I,J)=SFS(I,J)
SFT(J,I)=SF(I,J)
SS(I,J)=SYSS(I,J)
SST(J,I)=SS(I,J)
GO TO 150
140 JJ=J-KX
SF(I,J)=SFM(I,JJ)
SFT(J,I)=SF(I,J)
SS(I,J)=SMOD(I,JJ)
SST(J,I)=SS(I,J)
150 CONTINUE
DC 170 I=1,KC
DC 170 J=1,KX1
IF(J.GT.KX)GO TO 160
SU1(I,J)=SU1(I,J)
SU1T(J,I)=SU1(I,J)
GC TO 170
160 SU1(I,J)=0.0
   SU1(I,J)=SU1(I,J)
170 CONTINUE
   CALL MXMULT(T,SST,W1,TEMP1,KX1,KE,KE,M,N,L4,L2,L2,L2)
   FORMAT(5X,D10.3)
   CALL MXMULT(T,TEMP1,SS,Q,KX1,KE,KE,KX1,M,N,L4,L2,L2,L2)
   CALL MXMULT(T,TEMP1,SF,TEMP2,KX1,KE,KE,KX1,M,N,L4,L2,L2,L2)
   DC 180 I=1,KX1
   DO 180 J=1,KX1
   Q(I,J)=Q(I,J)+BETA*TEMP2(I,J)
180 CALL MXMULT(SU1,W3,TEMP1,KX1,KE,KE,KC,KC,M,N,L4,L2,L2,L2)
   CALL MXMULT(T,TEMP1,SU1,TEMP2,KX1,KE,KE,KC,KC,KX1,M,N,L4,L2,L2,L2)
   DC 190 I=1,KX1
   DO 190 J=1,KX1
   Q(I,J)=Q(I,J)+TEMP2(I,J)
190 CALL MXMULT(SU1,W3,TEMP1,KX1,KE,KE,KC,KC,M,N,L4,L2,L2,L2)
   CALL MXMULT(T,TEMP1,SU1,TEMP2,KX1,KE,KE,KC,KC,KX1,M,N,L4,L2,L2,L2)
   DC 221 I=1,KX
   DO 221 J=1,KX
   SYSFT(I,J)=SYSF(J,I)
   C
   Q MATRIX PARTITION
   C
230 DO 231 I=1,KX
   DO 231 J=1,KX
   RH11(I,J)=Q(I,J)
   IF(ICODE.EQ.1.AND.ITER.EQ.0)GO TO 1000
   IF(ICODE.EQ.1)GO TO 800
   ITER1=0
   DC 238 I=1,KX
   DC 238 J=1,KX
   AL(I,J)=SYSFT(I,J)
   BI(I,J)=SYSF(I,J)
238 CALL MSOLVE(KK,KX,KK,KK,A1,B1,C1,JB1,JB2)
   DC 232 I=1,KX
232 P(I,J)=X+1
233 Q(K,K)=Q(I,J)
234 X=Q(I,J)-Q(I,J)
235 CALL SXEQ(SYSF2*xDT*KX,FINV,KX)
236 CONTINUE
237 P(I,K)=X*J
238 Q(K,J)=Q(I,J)
239 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
240 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
241 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
242 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
243 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
244 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
245 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
246 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
247 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
248 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
249 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
250 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
251 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
252 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
253 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
254 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
255 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
256 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
257 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
258 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
259 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)
260 CALL MXUL(SXSP1*TSYSG*KN,KN,KB,KB,NEW,NE,Y)

EVALUATE AUGMENTED G MATRIX

DC 270 I=1,KX
271 J=1,KJ
272 G(I,J)=0
273 IF(I,J)=I,KJ
274 GO TO 260
275 GO TO 270
276 KJ=KJ+1
277 IF(KJ=1,1)
278 GO TO 270
SMG(I,J)=OMEG(I,J)

270 CONTINUE

C
C       EVALUATE AUGMENTED INITIAL CCADDITION VECTOR
C
D=280 I=1,KX1
X0(I)=0.0D+00
D=280 J=1,KC
280 XC(I)=XC(I)+SMG(I,J)*VCOMM(J)
WRITE(6,261)(X(I),I=1,KX1)

C
C       EVALUATE PERFORMANCE INDEX
C
PI1=PI
PI=0.0D+00
D=300 I=1,KX1
SIGMA=0.0D+00
D=300 J=1,KX1
290 SIGMA=SIGMA+P(I,J)*XO(J)
300 PI1=PI1+SIGMA*XO(I)
WRITE(6,511)PI
DELTAJ=PI1-PI1
WRITE(6,511)DELTAJ

C
C       EVALUATION OF GRADIENT CF F MATRIX
C
D=320 I=1,KB
D=320 J=1,KX
320 R1(I,J)=R1(I,J)+EK1(I,J)
DELTAJ=0.0D+00
D=500 K=1,IGAIN
D=340 I=1,KB
D=340 J=1,KB
SUM=0.0D+00
D=330 III=1,KB
330 SUM=SUM+SNV(III,J)*H(III,J,K)

MAIN0217
MAIN0218
MAIN0219
MAIN0220
MAIN0221
MAIN0222
MAIN0223
MAIN0224
MAIN0225
MAIN0226
MAIN0227
MAIN0228
MAIN0229
MAIN0230
MAIN0231
MAIN0232
MAIN0233
MAIN0234
MAIN0235
MAIN0236
MAIN0237
MAIN0238
MAIN0239
MAIN0240
MAIN0241
MAIN0242
MAIN0243
MAIN0244
MAIN0245
MAIN0246
MAIN0247
MAIN0248
MAIN0249
MAIN0250
MAIN0251
MAIN0252
340 HK(I,J)=SUM
   CALL MXMUL(T(HK,RI,TEMP,KB,KB,KB,KX,M,N,L1,L1,L1,L1)
   DC 360 I=1,KB
   DC 360 J=1,KX
   SUM=0.DO+0D
   DC 350 III=1,KB
   350 SUM=SUM+SIGN(I,III)*E(I,III,J,K)
   360 TEMP(I,J)=TEMP(I,J)+SUM
   CALL MXMUL(SYSP1,TEMP,DELSF1,KX,KB,KB,KX,M,N,L1,L1,L1,L1)
   CALL MXMUL(SYSS,DELSF2,KE,KX,KX,KX1,M,N,L2,L1,L1,L1)
   DO 380 I=1,KE
   DO 380 J=1,KX1
   IF (J.GT.KX) GO TO 370
   SS(I,J)=SFS(I,J)
   GTO 380
   370 SS(I,J)=0.0
   SST(I,J)=SS(I,J)
   CONTINUE
   CALL MXMUL(SST,W2,TEMP1,KX1,KE,KX1,KE,KX1,M,N,L4,L2,L2,L2)
   CALL MXMUL(TEMP1,TEMP2,KX1,KE,KX1,KE,KX11,M,N,L4,L2,L2,L2)
   DO 390 I=1,KX1
   DO 390 J=1,KX1
   390 DELQ(I,J)=ETA*(TEMP2(I,J)*TEMP2(J,I))
C
   EVALUATE DELF
   C
   DO 410 I=1,KX1
   DO 410 J=1,KX1
   IF (I.GT.KX .OR. J.GT.KX) GO TO 400
   DELF(I,J)=DELSF(I,J)
   DELF(J,I)=DELF(I,J)
   GTO 410
   400 DELF(I,J)=0.0
   DELF(J,I)=0.0
   CONTINUE
CALL MXMULT(DELFT,P,TEMP2,KX1,KX1,KX1,KX1,M,N,L4,L4,L4,L4)
DO 420 I=1,KX1
DO 420 J=1,KX1
420 RHS(I,J)=DELQ(I,J)+TEMP2(I,J)+TEMP2(J,I)
C
RHS MATRIX PARTITION
C
DO 431 I=1,KX
DO 431 J=1,KX
431 C(I,J)=RHS(I,J)
CALL MUSOLV(KX,KX,KX,KX,JB1,JB2)
DO 432 I=1,KX
DO 432 J=1,KX
432 DELP(I,J)=XSOLN(I,J)
DO 433 I=1,KX
433 RH12(I,1)=-RHS(I,KX1)
DO 445 I=1,KX
DELP(I,KX1)=0.0D+00
DO 444 J=1,KX
444 DELP(I,KX1)=DELP(I,KX1)+FINV(I,J)*RH12(J,1)
445 DELP(KX1,I)=DELP(I,KX1)
DELP(KX1,KX1)=-RHS(KX1,KX1)/(2.0*FMCD(1,1))
C
EVALUATE DELGS MATRIX
C
CALL MXMULT(HK,T1,TEMP3,KB,KB,KB,KB,KC,M,N,L1,L1,L1,L1,L2)
CALL MXMULT(SYSPL,TEMP3,DELGS,KB,KB,KB,KC,M,N,L1,L1,L1,L1,L2)
C
EVALUATE AUGMENTED DELG MATRIX
C
DO 450 I=1,KX1
DO 450 J=1,KC
IF(I.GT.KX)GO TO 440
DELG(I,J)=DELGS(I,J)
GO TO 450
440 DELG(I,J)=0.0D+00
450 CONTINUE
C
C     EVALUATE DELX
C
DC 450 I=1,KX1
DELEX(I)=0.0D+00
DO 460 J=1,KC
460 DELEX(I)=DELG(I,J)*VCOMM(J)
C
C     EVALUATE DELJ
C
DELJ=0.0D+00
DC 490 I=1,KX1
SUM=0.0D+00
SUM1=0.0D+00
DO 490 J=1,KX1
SUM=SUM+DELP(I,J)*XG(J)
480 SUM=SUM+P(I,J)*XO(J)
490 DELJ=DELJ+SUM*XO(I)+2.0*SUM1*DELEX(I)
WRITE(6,511)DELJ
PARAM(K)=PARAM(K)
DELK(K)=-STEP*DELJ
PARAM(K)=PARAM(K)+DELK(K)
PARAM(K)=PARAM(K)
WRITE(6,511)PARAM(K)
500 DELTAJ=DELTAJ+DELJ*DELK(K)
WRITE(6,511)DELTAJ
511 FORMAT(5X,D10.4)
ITTER=ITTER+1
IF(DABS(DELTAJ).LE.1.0D-05.CR.ITER.EQ.15)GO TO 510
GC TO 5
510 ISTCP=1
GC TO 5
520 WRITE(6,530)
530 FORMAT(5X,'THE ADJUSTED PARAMETER VALUES ARE')
DO 540 K=1,IGAIN
540 WRITE(5,550)K,PAR(K)
550 FORMAT(20X,'K(','II,'')= ',D12.4)
      WRITE(6,560)
560 FORMAT('O',5X,'THE SYSTEM DIFFERENTIAL TRANSITION MATRIX IS')
      DO 570 I=1,KX
570 WRITE(6,590)(SYSF(I,J),J=1,KX)
580 FORMAT(2X,10D12.4)
      WRITE(6,59C)
590 FORMAT('O',5X,'THE SYSTEM CONTROL MATRIX IS')
      DO 600 I=1,KX
600 WRITE(6,610)(SYSG(I,J),J=1,KC)
610 FORMAT(10X,10D12.4)
      WRITE(6,62C)ITER
620 FORMAT('O',5X,'THE NUMBER OF ITERATIONS=',I3)
      GE TO 1000
800 ITER1=ITER1+1
      IF(ITER1.GE.2)GO TO 520
      DO 910 K=1,IGAIN
         DELK(K)=DELK(K)/2.0
510 PARAM(K)=PARAM(K)+DELK(K)
      GO TO 5
1000 STOP
END
SUBROUTINE MSOLVE(MA1,MB1,MC1,NC1,A,MBX,C,JB1,JB2)

SUBROUTINE LINK SEQUENCE FOR SOLUTION OF MATRIX EQUATION

\[ \mathbf{A} \mathbf{x} + \mathbf{b} = \mathbf{c} \]

REAL*8 MXB(20,20), MXV(20,20), MXA0(20,20), MXA1(20,20), MXA2(20,20), IMXVIN(20,20)
DIMENSION PB(5,21), QB(5,21), ARRAYA(21), IDIMOB(5), IDIMPB(5),
1 INV(5,21), IDIMQ(5), IN(21), ARRAYB(21), A(20,20), C(20,20),
2, DMA(20,20), DMB(20,20), DMC(20,20), DMU(20,20), DMV(20,20),
3 DVINV(2C,20), X(20,20)
DOUBLE PRECISION ARRAYA, QB, PB, ARRAYB, QIN, QINV, DMA, DMB, DMC, DMU,
1 DMV, DVINV, CONST, X, A, C
COMM/N /HOLD/DMU, DMV/HOLD1/DMA, DMB/HCI/DMU/QINV, IDIMQ/
1 HGLD3/QB, PB, IDIMOB, IDIMPB, IDENT/HGLD4/DVINV, DMB/SECTR1/X

DC 50 I=1, MB1
DO 50 J=1, MB1
50 DMB(I,J) = MXB(I,J)
DO 60 I=1, MA1
DO 60 J=1, MA1
60 DMA(I,J) = A(I,J)
DO 70 I=1, MC1
DO 70 J=1, NC1
70 DMC(I,J) = C(I,J)
DO 80 I=1, MA1
DO 80 J=1, MA1
80 MXA0(I,J) = A(I,J)
CALL SMLTRN(MXB, MXV, MXVIN, MB1, INDXE)
DC 90 I=1, MB1
DC 90 J=1, MB1
DMV(I,J) = MXV(I,J)
90 DVINV(I,J) = MXVIN(I,J)
CALL SCAN2(MXB, MB1, JB1, JB2)
CALL SMLTRN(MXA0, MXA1, MXA2, MA1, INDXA)
CALL SCN1(MXAC, ARRAYA, IDIMA, MA1)
DC 300 LL1=1, JB1
IDIMB=ICIMQR(LL1)
DC 100 LL2=1,IDIMB
100 ARRAYB(LL2)=QM(LL1,LL2)
   CALL MPINV(QIN,IDM,ARRAYA,ICIMA,ARRAYB,IDIMB,CONST,IER)
   IF(ABS(CONST).LE.1.0E-06)GO TO 35C
   DO 200 LL3=1,IDM
200 QINV(LL1,LL3)=QIN(LL3)
300 IDIMO1(LL1)=IDM
   CALL MUSOLV(MB1,MA1,MC1,NC1,JB1,JB2)
   GO TO 400
350 WRITE(6,351)
351 FORMAT('MATRICES HAVE COMMON EIGENVALUES')
400 RETURN
END
SUBROUTINE SMLTRN(MATRIXB, MATRIXV, MTRXIN, N, INDEX)

C
C DANILEVSKII SIMILARITY TRANSFORMATION
C
C DOUBLE PRECISION ROW(20), COLUMN(20), AUX(20, 20), VI(20, 20), VJ(20, 20)
REAL AB MTXSIM(20, 20), INVSM(20, 20), MUNIT(20, 20), MATRXB(20, 20),
1 MTRXV(20, 20), MTRXIN(20, 20), MTX(20, 20)
C
C CONSTRUCT UNIT MATRICES
C
DO 5 K=1, N
DO 5 K1=1, N
IF (K.EQ. K1) GOTO 3
MUNIT(K, K1) = 0.
MTXSIM(K, K1) = 0.
INSM(K, K1) = 0.
MATRXV(K, K1) = 0.
MTRXIN(K, K1) = 0.
MTX(K, K1) = 0.
GOTO 5
3 MUNIT(K1, K1) = 1.
MTXSIM(K1, K1) = 1.
INVM(K1, K1) = 1.
MATRXV(K1, K1) = 1.
MTRXIN(K1, K1) = 1.
MTX(K1, K1) = 1.0
5 CONTINUE

C INDEX J INDICATES COLUMN OF MATRXB
C
J = 1
C
INDEX COUNTS NUMBER OF DIAGONAL BLOCKS
C
INDEX = 1
C INDEX J2 INDICATES COLUMN OF MTXSIM
10 J2=J+1
C SEARCH IN COLUMN J OF MATRXB FOR NON ZERO ELEMENT
C IF PIVOT ELEMENT OF MATRXB IS NON ZERO, DISCONTINUE SEARCH
C PROCEED TO CONSTRUCT MTXSIM, INVSIM
C OTHERWISE SEARCH IN REMAINING ROWS FOR FIRST NON ZERO ELEMENT
C
IF(CABS(MATRXB(J2,J)).GT.1.0D-06)GO TO 1000
MATRXB(J2,J)=0.0D+00
J3=J2+1
IF(J3.EQ.(N+1))GO TO 4500
100 DO 200 I=J3,N
IF(CABS(MATRXB(I,J)).GT.1.0D-06)GO TO 300
MATRXB(I,J)=0.0D+00
200 CONTINUE
J=J+1
IF(J.EQ.N)GO TO 4500
INDEX=INDEX+1
GO TO 10
C C INTERCHANGE ROWS I AND J2 AND COLUMNS I AND J2
C
300 DO 400 J4=J,N
ROW(J4)=MATRXB(J2,J4)
MATRXB(J2,J4)=MATRXB(I,J4)
400 MATRXB(I,J4)=ROW(J4)
DO 500 I1=1,N
COLUMN(I1)=MATRXB(I1,J2)
MATRXB(I1,J2)=MATRXB(I1,I)
500 MATRXB(I1,I)=COLUMN(I1)
C C CONSTRUCT MATRIX MTX
C
MTX(J2,J2)=0.
MTX(I,I)=0.
MTX(J2,I)=1.
MTX(I,J2)=I.
   CALL DMPROD(MATRXV,MTX,V1,N,N,N,N1,N1)
   CALL DMCOPY(V1,MATRXV,N,N,N1,N1)
   CALL DMPROD(MTX,MTRXIN,V1,N,N,N,N1,N1)
   CALL DMCOPY(V1,MTRXIN,N,N,N1,N1)
   CALL DMCOPY(MUNIT,MTX,N,N,N1,N1)

C
   CONSTRUCT MTXSIM AND INVSIM

C
   1000 PIVOT=MATRIX8(J2,J)
   DO 1100 I2=1,N
   IF(I2.EQ.J2)GO TO 1050
   INVSIM(I2,J2)=MATRIX8(I2,J)/PIVOT
   GO TO 1100

1050 INVSIM(I2,J2)=1./PIVOT
   1100 MTXSIM(I2,J2)=MATRIX8(I2,J)

C
   COMPUTE AUXILIARY MATRIX VECAX AND RECURSIVE MATRIX8
   CALL DMPROD(INVSIM,MTXV,AX,AX,AX,AX,AX,AX,AX,AX,AX)
   CALL DMPROD(AUX,MTXSIM,MATRIX8,AX,AX,AX,AX,AX,AX,AX,AX)
   CALL DMPROD(MATRIX8,MTXSIM,V1,N,N,N,N1,N1)
   CALL DMCOPY(V1,MATRIX8,N,N,N1,N1)
   CALL DMCOPY(INVSIM,MTRXIN,V1,N,N,N,N1,N1)
   CALL DMCOPY(V1,MTRXIN,N,N,N1,N1)
   J=J+1

C
   CHECK TO DETERMINE NEW BASIS VECTORS HAVE BEEN ESTABLISHED
   IF(J.EQ.N)GO TO 4500
   CALL DMCOPY(MUNIT,MTXSIM,N,N,N1,N1)
   CALL DMCOPY(MUNIT,INVSIM,N,N,N1,N1)
   GO TO 10

4500 RETURN

END
SUBROUTINE SCAN1(MATRXA,ARRAY1,IDIM1,MN)
C
C THIS SUBROUTINE EVALUATES THE CHARACTERISTIC POLYNOMIAL
C OF THE A MATRIX
C
DIMENSION ARRAYQ(5,21),IDENT(20),IDIMNG(5),ARRAY1(21),ARRAY(21),
1 ARRAY2(21)
REAL& MATRXA(20,20)
DOUBLE PRECISION ARRAY1,ARRAY2,ARRAY,ARRAYQ
C
INITIALIZE ARRAYQ
C
DC 10 I=1,5
DC 10 J=1,21
10 ARRAYQ(I,J)=0.
J1=0
C
IDENTIFY NON UNITY(ZERO) SUBDIAGONAL ELEMENTS
C
DC 1000 K1=2,MN
L1=K1-1
IF(K1.EQ.MN)GO TO 500
IF(DABS(MATRXA(K1,L1)-1.0).LE.1.0D-04)GO TO 1000
J1=J1+1
IDENT(J1)=K1-1
GO TO 1000
500 J1=J1+1
IDENT(J1)=K1
IF(DABS(MATRXA(K1,L1)-1.0).GE.1.0D-04)GO TO 750
GO TO 1000
750 J1=J1+1
IDENT(J1)=K1
IDENT(J1-1)=K1-1
1000 CONTINUE
IDIMNG(1)=IDENT(1)+1
ARRAYQ(1,IDIMNG(1))=1.
SUBROUTINE SCAN2(MATRXB,MN,J1,J2)
C THIS SUBROUTINE IDENTIFIES AND HOLDS THE P & Q ARRAYS
C FOR SUBSEQUENT MATRIX POLYNOMIAL INVERSION
C
DIMENSION ARRAYP(5,21),ARRAYQ(5,21),IDIMNP(5),IDIMNQ(5),IDENT(5)
REAL*8 MATRXB(20,20)
DOUBLE PRECISION ARRAYP,ARRAYQ
COMMON /HOLD3/ARRAYQ,ARRAYP,IDIMNQ,IDIMNP,IDENT
C
INITIALIZE ARPAYP,ARRAYQ
C
DC 20 I=1,5
DC 20 J=1,21
ARRAYP(I,J)=0.
20 ARRAYQ(I,J)=0.
C
FOLLOWING SEQUENCE IDENTIFIES AND STORES RELEVANT ARRAYS
C
J1=0
DC 1000 K1=2,MN
L1=K1-1
IF(K1.EQ.MN) GO TO 500
IF(CABS(MATRXB(K1,L1)-1.0).LE.1.0D-04) GO TO 1000
J1=J1+1
IDENT(J1)=K1-1
GO TO 1000
500 J1=J1+1
IDENT(J1)=K1
IF(CABS(MATRXB(K1,L1)-1.0).GE.1.0D-04) GO TO 750
GO TO 1000
750 J1=J1+1
IDENT(J1)=K1
IDENT(J1-1)=K1-1
1000 CONTINUE
IDIMNQ(1)=IDENT(1)+1
ARRAYQ(1, IDIMNQ(1)) = 1.
IC = IDENT(1)
DC 2000 I1 = 1, ID
2000 ARRAYQ(1, I2) = MATRXB(I1, ID)
IF(J2.EQ.2) GO TO 6000
IDIMNP(1) = IDIMNQ(1)
IC1 = IDENT(2)
DC 3000 K2 = 2, J1
IDIMNQ(K2) = IDENT(K2) - IDENT(K2-1) + 1
INDEX = IDIMNQ(K2) - 1
ARRAYQ(K2, IDIMNQ(K2)) = 1.
DC 3000 K3 = 1, INDEX
K4 = IDENT(K2-1) + K3
K5 = IDENT(K2)
3000 ARRAYQ(K2, K3) = MATRXB(K4, K5)
DC 4000 I2 = 1, ID
4000 ARRAYP(I1, I2) = MATRXB(I2, ID)
J2 = J1 - 1
IF(J2.EQ.1) GO TO 7000
DC 5000 K6 = 2, J2
IDIMNP(K6) = IDIMNP(K6-1) + IDIMNQ(K6) - 1
INDEX1 = IDIMNP(K6) - 1
K8 = IDENT(K5 + 1)
DC 5000 K7 = 1, INDEX1
5000 ARRAYP(K6, K7) = MATRXB(K7, K8)
GO TO 7000
6000 J2 = 0
1EQ = IDIMNQ(1)
7000 RETURN
END
SUBROUTINE MPINV(QINV,IDIMQ1,ARRAYA,IDIMA,ARRAYB,IDIMB,CONST,IER)

MATRIX POLYNOMIAL INVERSION SCHEME BASED ON THE
EUCLIDIAN ALGORITHM

DIMENSION ARRAYA(20), ARRAYB(20), ARAYA1(20), ARAYB1(20), AUXIL(20),
ARRAYR(20), ARRAYP1(20), ARAYP2(20), ARAYP3(20), QINV(20), Q(20),
Q3(20), ARAYR1(20), ARAYP4(20), Q2(20), ARAYQ1(20), ARAYQ2(20),
3ARAYQ3(20), ARAYQ4(20)

CRUCIAL PRECISION ARRAYA, ARRAYB, ARAYA1, ARAYB1, ARAYP4, ARRAYR, AUXIL,
1ARAYP1, ARAYP2, ARAYP3, ARAYQ1, ARAYQ2, ARAYQ3, QINV, CONST, Q1, Q2, Q3,
2ARAYR1, ARAYQ4

INITIALIZE ALL ARRAY HOLDS

DC 10 I=1,20
ARAYA1(I)=0.
ARAYB1(I)=0.
ARRAYR(I)=0.
ARRAYP1(I)=0.
ARRAYP2(I)=0.
10 ARAYP3(I)=0.
IC1MP1=1
ARAYP1(I)=1.

CHECK DEGREES OF POLYNOMIALS

IF(IDIMA.LT.IDIMA)GO TO 110
CALL MPMOVE(ARAYA1,IDIMA1,ARRAYA,IDIMA1)
CALL MPMOVE(ARAYB1,IDIMB1,ARRAYB,IDIMB1)
CALL MPDIV(Q1,IDIMQ1,ARAYB1,IDIMB1,ARAYA1,IDIMA1,IER)
IF(IER.EQ.1)GO TO 200
ICCUNT=1
GO TO 10
CALL MPMOVE(ARRAYR,IDIMR,ARAYB1,IDIMB1)
IF(IDC1MP1.EQ.1)GO TO 70
ICCUNT=ICCUNT+1
CALL MPDIV(02, Q3, DIMQ2, ARAY1, IDIMR1, IDIMR1)
55 ARAYP2 = IDIMQ2
DC 55 J = 0, IDIMP2
IF (IDIMP2 = IDIMQ2) G0 100
CALL MPDIV (Q2, EQ1, IDIMP2, J, IDIMP2)
60 J = J + 1
GO TO 55
70 REMAINDER POLYNOMIAL IS CONSTANT

C 70 CONSt = ARAY1(11)
GO TO 200
80 IF (CABS(CONSt) > LE.1E-06) G0 100
DC 90 1 = 1, IDIMR1
90 QINVI1 = ARAYP2(11) / CONSt
GC TO 200
100 CONSt = ARAY1(11)
GO TO 95
95 QINVI2 = ARAYP3(12) / CONSt
GC TO 200
GC TO 200
C
INITIALIZE ARRAYS
110 DO 115 I=1,20
  ARAYA1(I)=0.
  ARAYB1(I)=0.
  ARAYR(I)=0.
  ARAYQ1(I)=0.
  ARAYQ2(I)=0.
115 ARAYQ3(I)=0.
  CALL MPMOVE(ARAYA1, IDIMA1, ARAYB1, IDIMB)
  CALL MPMOVE(ARAYB1, IDIMB1, ARAYA, IDIMA)
  CALL MPDIV(Q1, IDIMQ1, ARAYB1, IDIMB1, ARAYA1, IDIMA1, IER)
  IF(IDIMB1.EQ.1) GO TO 12C
  DO 116 J=1, IDIMQ1
116 ARAYQ1(J)=-Q1(J)
  IDQ1=IDIMQ1
  CALL MPDIV(Q2, IDIMOQ2, ARAYA1, IDIMA1, ARAYB1, IDIMB1, IER)
  CALL MPMPY(ARAYQ2, IDQ2, Q2, IDIMOQ2, Q1, IDIMQ1)
  ARAYQ2(I)=ARAYQ2(I)+1.
  IF(IDIMA1.EQ.1) GO TO 130
  CALL MPMOVE(ARAYR, IDIMR, ARAYB1, IDIMB1)
  CALL MPMOVE(ARAYR1, IDIMR, ARAYA1, IDIMA1)
117 CALL MPDIV(Q3, IDIMOQ3, ARAYR, IDIMR, ARAYR1, IDIMR, IER)
  CALL MPMPY(ARAYQ4, IDQ4, Q3, IDIMOQ3, ARAYQ2, IDQ2)
  CALL MPSUB(ARAYQ3, IDQ3, ARAYC1, IDQ1, ARAYQ4, IDQ4)
  CALL MPMPY(AUXIL, IDIMU, ARAYC3, IDQ3, ARAYQ2, IDIMB)
  IF(IDIMR.EQ.1) GO TO 140
  CALL MPMOVE(ARAYQ1, IDQ1, ARAYQ2, IDQ2)
  CALL MPMOVE(ARAYQ2, IDQ2, ARAYQ3, IDQ3)
  CALL MPMOVE(AUXIL, IDIMU, ARAYR, IDIMR)
  CALL MPMOVE(ARAYK, IDIMR, ARAYR1, IDIMR1)
  CALL MPMOVE(ARAYR1, IDIMR1, AUXIL, IDIMU)
  GO TO 117
120 CONST=ARAYB1(1)
  IF(CARS(CONST).LE.1.0E-06) GO TO 20C
  IDIMQ1=IDIMQ1
DC 125 J=1,IDIMQI
125 QINV(J)=-Q1(J)/CONST
GO TO 200
130 CCONST=ARRAY1(1)
   IF(CABS(CONST).LE.1.0E-06)GO TO 200
   IDIMQI=IDQ2
   DC 135 J=1,IDIMQI
135 QINV(J)=ARRAY2(J)/CONST
GO TO 200
140 CCONST=ARRAYR(1)
   IF(CABS(CONST).LE.1.0E-06)GO TO 200
   IDIMQI=IDQ3
   DO 145 J=1,IDIMQI
145 QINV(J)=ARRAY3(J)/CCONST
200 RETURN
END
SUBROUTINE MUSOLV(MB1,MA1,MC1,NC1,JB1,JB2)
DIMENSION V1(20,20),V2(20),VSUM(20),V3(20),U(20,20),C(20,20),
1A(20,20),U1(20),IDIMQI,51,QINV(5,21),V(20,20),Q8(5,21),PB(5,21),
ZIDENT(5),IDIMQB(5),IDIMPB(5),E(20,20),VINV(20,20),Y(20,20),
3A(20,20),AX(20,20),X8(20,20),AXX8(20,20),X(20,20)
DOUBLE PRECISION U,V1,V2,V3,VSUM,A,UL,QINV,PB,Q8,V,VINV,AX,X8,B,E,
1Y,AXX8,X,C
COMMON /HOLD/U,V/HOLD1/A,C/HOLD2/CINV,DIIMQI
1/HOLD3/Q8,PB,IDIMQB,IDIMPB,IDENT/HOLD4/VINV,B/SECTR1/X
ICOUNT=0
JWRITE=0
C
V1=C*V
CALL DMULT(C,V1,MC1,NC1,MR1,MB1,FM,NN)
C INITIALIZE VSUM
5 DO 10 II=1,MA1
10 VSUM(II)=0.
C FIRST PASS SOLVE FOR U1, U2, .............. UIDENT(1)
11=IDENT(1)
1I5=II1+1
DO 40 II2=1,111
40 II3=1,MA1
20 V2(II3)=V1(II3,II2)
II4=II2+1
30 DO 35 II3=1,MA1
35 VSUM(II3)=VSUM(II3)+Q8(1,II4)*V2(II3)
II4=II4+1
IF(II4.GT.II5)GO TO 40
C V2(II3)=(-A)*V2(II3)
CALL AVMULT(A,MA1,V2)
GC TO 30
40 CONTINUE
C
U1(L1)=(-A)*U1(L1,L2-1)
DO 41 L1=1,MA1
41 U1(L1,L1)=QINV(L1,1)*VSUM(L1)
INOXQI=IDIMQI(J1)
DO 43 L2=1,INOXQI
USLV00G1
USLV00C2
USLV0003
USLV0004
USLV0005
USLV0006
USLV0007
USLV0008
USLV0009
USLV0010
USLV0011
USLV0012
USLV0013
USLV0014
USLV0015
USLV0016
USLV0017
USLV0018
USLV0019
USLV0020
USLV0021
USLV0022
USLV0023
USLV0024
USLV0025
USLV0026
USLV0027
USLV0028
USLV0029
USLVCC30
USLVCC31
USLV0032
USLV0033
USLV0034
USLV0035
USLV0036
CALL AVMULT(A, MA1, VSUM)
DO 42 L1 = 1, MA1
42 U(L1, 1) = U(L1, 1) + QINV(1, L2) * VSUM(L1)
43 CONTINUE
DO 70 L2 = 2, II1
DO 60 L1 = 1, MA1
60 U1(L1) = U(L1, L2 - 1)
CALL AVMULT(A, MA1, U1)
DO 65 L1 = 1, MA1
65 U(L1, L2) = U1(L1) - V1(L1, L2 - 1)
70 CONTINUE
IF(JB1 .EQ. 1.0 .AND. ICOUNT .EQ. 0.0) GO TO 300
IF(JB1 .EQ. 1.0 .AND. ICOUNT .GT. 0.0) GO TO 400
C SOLVE FOR REMAINING VECTORS OF U
DO 200 JJ = 2, JB1
JJ0 = JJ - 1
JJ1 = IDENT(JJ)
JJ2 = IDENT(JJ - 1) + 1
C RE INITIALIZE VSUM
DO 90 JJ3 = 1, MA1
80 VSUM(JJ3) = 0.
JJ4 = IDENT(JJ - 1)
DO 90 JJ5 = 1, JJ4
DO 90 JJ3 = 1, MA1
V3(JJ3) = PB(JJ0, JJ5) * U(JJ3, JJ5)
90 VSUM(JJ3) = VSUM(JJ3) + V3(JJ3)
JJ6 = 0
DO 130 JJ7 = JJ1, JJ2
JJ6 = JJ6 + 1
DO 100 JJ3 = 1, MA1
100 V2(JJ3) = V1(JJ3, JJ7)
JJ8 = JJ6 + 1
110 DO 120 JJ3 = 1, MA1
120 VSUM(JJ3) = VSUM(JJ3) + Q8(JJ, JJ8) * V2(JJ3)
JJ8 = JJ8 + 1
IF(JJ8 .GT. (JJ2 - JJ1 + 2)) GO TO 130

USLV0037
USLV0038
USLV0039
USLV0040
USLV0041
USLV0042
USLV0043
USLV0044
USLV0045
USLV0046
USLV0047
USLV0048
USLV0049
USLV0050
USLV0051
USLV0052
USLV0053
USLV0054
USLV0055
USLV0056
USLV0057
USLV0058
USLV0059
USLV0060
USLV0061
USLV0062
USLV0063
USLV0064
USLV0065
USLV0066
USLV0067
USLV0068
USLV0069
USLV0070
USLV0071
USLV0072
C  V2(JJ3)=(-A)*V2(JJ3)
    CALL AVMULT(A,MA1,V2)
    GO TO 110
130 CONTINUE
C    SOLVE FOR U(JJ1)
    DO 140 L1=1,MA1
140 U(L1,JJ1)=QINV(JJ,1)*VSUM(L1)
    INDXQ=IDI*QI(JJ)
    DO 160 L2=2,INDXQ
    CALL AVMULT(A,MA1,VSUM)
    DO 150 L1=1,MA1
150 U(L1,JJ1)=U(L1,JJ1)+QINV(JJ,L2)*VSUM(L1)
160 CONTINUE
    IF(JJ1.EQ.JJ2)GO TO 200
    JJ9=JJ1+1
    DO 190 L2=JJ9,JJ2
    DO 170 L1=1,MA1
170 U1(L1)=U(L1,L2-1)
    CALL AVMULT(A,MA1,U1)
    DO 180 L1=1,MA1
180 U(L1,L2)=U1(L1)-V1(L1,L2-1)
190 CONTINUE
200 CONTINUE
    IF(ICOUNT.GT.0)GO TO 400
300 CALL DMULT(U,VINV,X,MA1,MB1,MB1,NN)
    CALL DMULT(A,X,AX,MA1,MA1,MB1,NN)
    CALL DMULT(X,B,MA1,MB1,MB1,NN)
    DO 340 I=1,MA1
340 AXXB(I,J)=AX(I,J)+XB(I,J)
    DO 350 I=1,MA1
350 E(I,J)=AXXB(I,J)+C(I,J)
    DO 330 I=1,MA1
330 WRITE(6,600)E(I,J),J=1,MB1
    ICOUNT=ICOUNT+1
CALL DMULT(E,V,VL,MA1,MB1,MB1,MB1,PM,NN)
GO TO 5
400 CALL DMULT(U,VINV,Y,MA1,MB1,MB1,MM,NN)
CALL DMULT(A,Y,AX,MA1,MA1,MA1,MB1,PM,NN)
CALL DMULT(Y,B,XB,MA1,MB1,MB1,MB1,PM,NN)
DO 450 I=1,MA1
   DO 450 J=1,MB1
      X(I,J)=X(I,J)+Y(I,J)
      AXXB(I,J)=AX(I,J)+XB(I,J)
   450 E(I,J)=AXXB(I,J)+E(I,J)
      IF(ICOUNT.EQ.2)GO TO 500
      ICCOUNT=ICOUNT+1
      CALL DMULT(E,V,VL,MA1,MB1,MB1,MB1,PM,NN)
   GO TO 5
500 JWRITE=JWRITE+1
      IF(JWRITE.EQ.2)GO TO 700
      WRITE(6,550)
550 FORMAT(5X,'THE SOLUTION MATRIX X IS')
   DO 560 I=1,MA1
      WRITE(6,600)(X(I,J),J=1,MB1)
   560 WRITE(6,650)
650 FORMAT(5X,'THE ERROR MATRIX EMATRIX=AX+XB+C IS')
   DO 670 I=1,MA1
      WRITE(6,600)(E(I,J),J=1,MB1)
   600 FORMAT(10D15.5)
700 RETURN
END
SUBROUTINE AVMULT(AA, MAA, V4)
DIMENSION AA(20, 20), V4(20), V5(20)
DOUBLE PRECISION AA, V4, V5, SSUM
DO 50 I = 1, MAA
      SSUM = 0.
  DO 45 J = 1, MAA
   45 SSMUV = SSMUV + (-AA(I, J)) * V4(J)
  DO 55 K = 1, MAA
   55 V4(K) = V5(K)
RETURN
END
SUBROUTINE RSCHEK(ARRAY,IDIM,ROUTH1,IDIMR1,ROUTH2,IDIMR2,ICODE)
C
C THIS SUBROUTINE CHECKS FOR SYSTEM ASYMPTOTIC
C STABILITY USING THE ROUGHT CRITERION
C
C CONDITION CODES
C 1)ICODE=0 INDICATES ASYMPTOTIC STABILITY
C 2)ICODE=1 INDICATES ASYMPTOTIC INSTABILITY
C
DIMENSION ARRAY(21),ROUTH1(21),ROUTH2(21),Q(21),AUXIL(21)
DOUBLE PRECISION ARRAY,ROUTH1,ROUTH2,Q,AUXIL
IR=IDIM
3 IF(ARRAY(IR).LE.0.0)GO TO 40
   IR=IR-1
   IF(IR.EQ.0)GO TO 4
   GO TO 3
C
C FORM INITIAL ARRAYS
C
4 IDIMR1=IDIM
   IDIMR2=IDIM-1
   IR1=IDIMR1
   IR2=IDIMR2
5 IR3=IR1-1
   IR4=IR2-1
   ROUTH1(IR1)=ARRAY(IR1)
   IF(IR1.EQ.1)GO TO 10
   ROUTH1(IR3)=0.
   ROUTH2(IR2)=ARRAY(IR2)
   IF(IR2.EQ.1)GO TO 10
   ROUTH2(IR4)=0.
   IR1=IR1-2
   IR2=IR2-2
   GO TO 5
10 CALL MPDIV(Q,IDIMQ,ROUTH1,IDIMR1,ROUTH2,IDIMR2)
RSCK0001
RSCK0002
RSCK0003
RSCK0004
RSCK0005
RSCK0006
RSCK0007
RSCK0008
RSCK0009
RSCK0010
RSCK0011
RSCK0012
RSCK0013
RSCK0014
RSCK0015
RSCK0016
RSCK0017
RSCK0018
RSCK0019
RSCK0020
RSCK0021
RSCK0022
RSCK0023
RSCK0024
RSCK0025
RSCK0026
RSCK0027
RSCK0028
RSCK0029
RSCK0030
RSCK0031
RSCK0032
RSCK0033
RSCK0034
RSCK0035
RSCK0036
IF(IDIMR1.EQ.1.AND.ROUTH1(1).GT.0.0)GO TO 50
IF(IDIMR1.EQ.1.AND.ROUTH1(1).LE.0.0)GO TO 40
IR5=IDIMR1
20 IF(ROUTH1(IR5).LE.0.0)GO TO 40
IR5=IR5-2
IF(IR5.EQ.0.OR(IR5.EQ.-1))GO TO 30
GO TO 20
30 CALL MPMOVE(AUXIL,ICIMU,ROUTH1,IDIMR1)
    CALL MPMOVE(ROUTH1,IDIMR1,ROUTH2,IDIMR2)
    CALL MPMOVE(ROUTH2,IDIMR2,AUXIL,IDICM)
    GO TO 10
40 ICODE=1
    GO TO 60
50 ICODE=0
60 RETURN
END
SUBROUTINE MPDIV(P, IDIMP, X, IDIMX, Y, IDIMY, IER)
DIMENSION P(1), X(1), Y(1)
DOUBLE PRECISION P, X, Y, TOL
TOL=1.0D-06
CALL MPNORM(Y, IDIMY, TOL)
IF(IDIMY).GE.50, 50, 10
10 IDIMP=IDIMX-IDIMY+1
IF(IDIMP).GE.20, 20, 30
20 IDIMP=0
30 IER=0
40 RETURN
50 IER=1
GO TO 40
60 IDIMX=IDIMY-1
I=IDIMP
70 II=I+IDIMX
P(I)=X(II)/Y(IDIMY)
DO 80 K=1, IDIMX
J=K-1+I
X(J)=X(J)-P(I)*Y(K)
80 CONTINUE
I=I-1
IF(I).LT.0, 90, 70
90 CALL MPNORM(X, IDIMX, TOL)
GO TO 30
END
SUBROUTINE MPNORM(X,IDIMX,EPS)
DIMENSION X(1)
DOUBLE PRECISION X,EPS
1 IF(IDIMX)4,4,2
2 IF(DABS(X(IDIMX))-EPS)3,3,4
3 IDIMX=IDIMX-1
   GO TO 1
4   RETURN
END
SUBROUTINE MPMPY(Z, IDIMZ, X, IDIMX, Y, IDIMY)
DIMENSION Z(1), X(1), Y(1)
DOUBLE PRECISION Z, X, Y
IF(IDIMX*IDIMY)10, 10, 20
10 IDIMZ=0
GO TO 50
20 IDIMZ=IDIMX+IDIMY-1
DO 40 I=1, IDIMZ
30 Z(I)=0.
   DO 40 J=1, IDIMY
   K=I+J-1
   40 Z(K)=X(I)*Y(J)+Z(K)
50 RETURN
END
SUBROUTINE MPMOVE(Y, IDIMY, X, IDIMX)
DIMENSION X(1), Y(1)
DOUBLE PRECISION X, Y
IDIMY=IDIMX
IF(IDIMX)30,30,10
10 DC 20 I=1, IDIMX
20 Y(I)=X(I)
30 RETURN
END
SUBROUTINE MPSUB(Z,IDIMZ,X,IDIMX,Y,IDIMY)
DIMENSION Z(1),X(1),Y(1)
DOUBLE PRECISION Z,X,Y
C
TEST DIMENSIONS OF SUMMANDS
NDIM=IDIMX
IF(IDIMX-IDIMY)10,20,20
10 NDIM=IDIMY
20 IF(NDIM)90,9,30
30 DO 90 I=1,NDIM
   40 IF(I-IDIMX)40,40,60
50 Z(I)=X(I)-Y(I)
   60 Z(I)=-Y(I)
   GO TO 80
70 Z(I)=X(I)
80 CONTINUE
90 IDIMZ=NDIM
RETURN
END
SUBROUTINE MXMULT(A,B,C,MA,NA,MB,NB,MC,NC,II,J1,II,J2)
DOUBLE PRECISION A(II,J1),B(II,J2),C(II,J2)
MC=MA
NC=NB
DC 10 I=1,MA
DC 10 J=1,NB
C(I,J)=0.0D+00
DC 10 K=1,MB
10 C(I,J)=C(I,J)+A(I,K)*B(K,J)
RETURN
END
REFERENCES


