measurement systems laboratory

massachusetts institute of technology, cambridge massachusetts 02139

TE-49

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 Prasun K. Sinha
Department of Aeronautics and Astronautics, January 1972

Certified by James E. Potter
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.

SYMBOL	DEFINITION
A,B	known matrices in matrix-algebraic equation (Chapter 3)
$\underline{b}$	bus variable vector (Chapter 2)
C	known RHS matrix of matrix algebraic equation (Chapter 3)
$\underline{c}$	command variable vector
$\underline{e}$	error vector
E	three dimensional array in equations for control system representation (Chapter 2)
F_s	system differential transition matrix
F_m	model differential transition matrix
F	augmented system differential transition matrix
G	augmented system control matrix
H	three dimensional array in equations for control system representation (Chapter 2)
I	identity matrix
J	performance index
K_i	the i^{th} parameter
L_i	companion matrix (Chapter 3)
$M,\tilde{M},N,\tilde{N},P,Q'$	system matrices in describing equations (Chapter 2)

SYMBOL	DEFINITION
Q_s	system control matrix
Q_m	model control matrix
R,S,T	matrices in system describing equations (Chapter 2)
S_s	system output matrix
S_m	model output matrix
t	time
U,V	matrices in form of assumed solution for X (Chapter 3)
$W_e, W_{\dot{e}}, W_u$	error and control signal weighting matrices
$\underline{w}$	vector of magnitudes of delta function commands
X	unknown matrix in matrix algebraic equation (Chapter 3)
$\underline{x}_s$	system state vector
$\underline{x}_m$	model state vector
$\underline{x}$	augmented system state vector
$\underline{y}_s$	system output vector
$\underline{y}_m$	model output vector

SPECIAL NOTATIONS

F^T ≡Transpose of F

$||\underline{x}|| \equiv \sqrt{\underline{x}^T\underline{x}}$

$||\underline{x}||_Q^2 \equiv \underline{x}^T Q \underline{x}$

CONTENTS

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.

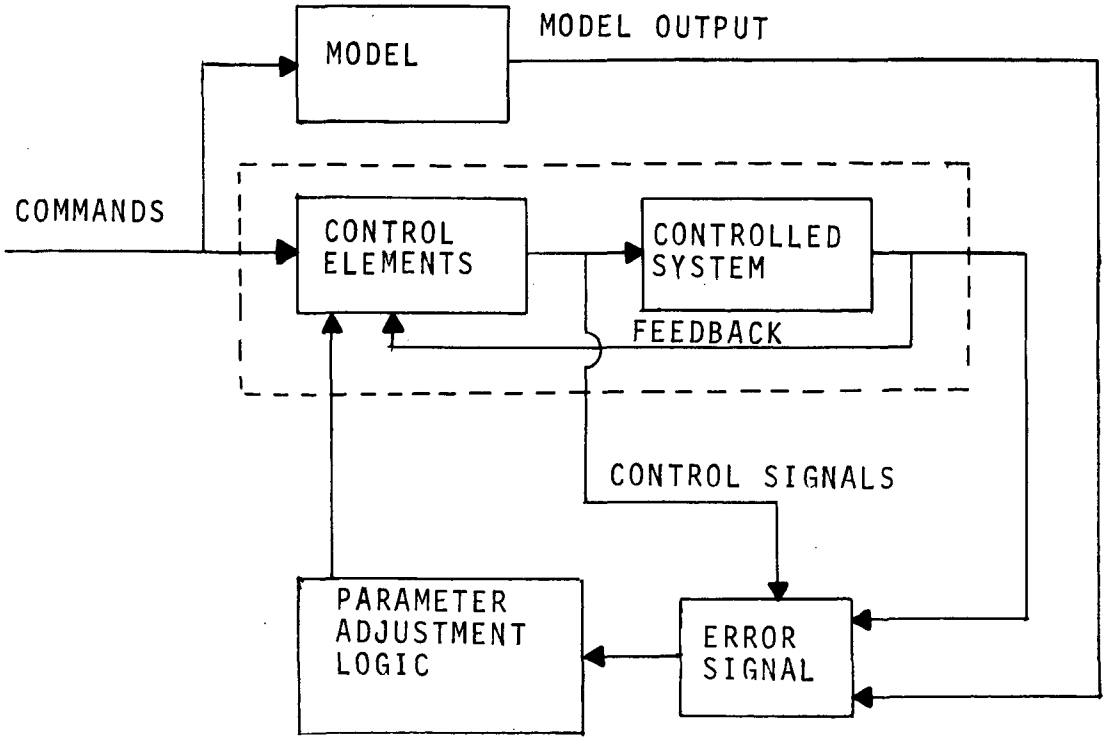


Fig. 1.1 Functional Diagram Illustrating the Model Reference Adaptive Control Concept

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

$$J = \int_0^\infty (e^2 + \beta \dot{e}^2)\, dt \qquad (1.1)$$

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

$$\underline{e} = \underline{y}_s - \underline{y}_m \qquad (1.2)$$

and the performance index could be formulated as

$$J = \int_0^\infty [\underline{e}^T W_e \underline{e} + \beta \dot{\underline{e}}^T W_{\dot{e}} \dot{\underline{e}}] \, dt \qquad (1.3)$$

where W_e and $W_{\dot{e}}$ 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 $\underline{u}$ represents these control signals, then the performance index to be minimized could be formulated as

$$J = \int_0^\infty [\underline{e}^T W_e \underline{e} + \beta \dot{\underline{e}}^T W_{\dot{e}} \dot{\underline{e}} + \underline{u} W_u \underline{u}] dt \qquad (1.4)$$

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 algoritm 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 quantitites 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

$$M\dot{\underline{x}}_s = N\underline{x}_s + P\underline{b} + Q'\underline{c} \tag{2.1}$$

$$\underline{b} = R\underline{x}_s + S\underline{b} + T\underline{c} + \sum_{i=1}^{n}(E_i\underline{x}_s + H_i\underline{b})K_i \tag{2.2}$$

$$\underline{y}_s = S_s\underline{x}_s \tag{2.3}$$

$\underline{x}_s$ is the system state vector, $\underline{b}$ the vector of bus variables, $\underline{c}$ the commands and $\underline{y}_s$ 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,..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 straight forward manner.

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

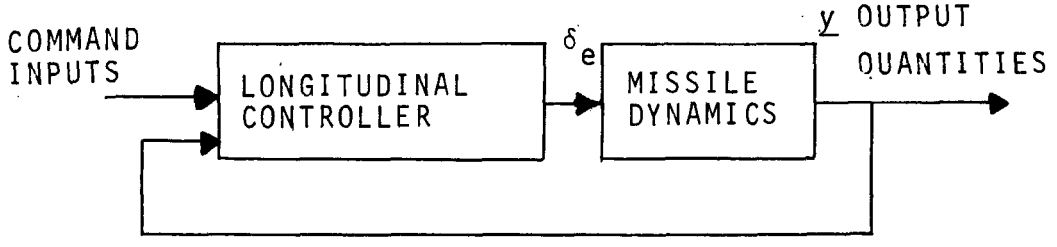


Fig. 2.1 Simplified Functional Diagram of Longitudinal Flight Control System

described by equations of the form

$$M\dot{\underline{x}} = N\underline{x} + Q\delta_e \qquad (2.4\ a)$$

where δ_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

$\dot{\underline{x}} = F\underline{x} + G\underline{\delta}$ 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].

$$13.78'\dot{u} + 0.88'u - 0.392'\alpha + 0.74\theta = 0$$

$$1.48'u + 13.78'\dot{\alpha} + 4.46'\alpha - 13.78\dot{\theta} = -0.246\,\delta_e$$

$$0.552'\dot{\alpha} + 0.619'\alpha + 0.514\ddot{\theta} + 0.192\dot{\theta} = -0.710\ \delta_e$$

$$\dot{h} = \theta - '\alpha \qquad (2.4b)$$

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

$$13.78'\dot{u} + 0.88'u - 0.392'\alpha + 0.74\theta = 0$$

$$1.48'u + 13.78'\dot{\alpha} + 4.46'\alpha - 13.78\theta = -0.246\delta_e$$

$$0.0552'\dot{\alpha} + 0.619'\alpha + 0.514\,\dot{q} + 0.192\dot{\theta} = -0.710\delta_e$$

$$\dot{\theta} = q$$

$$\dot{h} = \theta - '\alpha \qquad (2.4c)$$

The state variables for the vehicle dynamics are

'u - non dimensional perturbation in forward speed

'α - angle of attack perturbation (rads)

θ - pitch angle perturbation (rads)

q - pitch rate (rads/sec)

h - non dimensional altitude perturbation

The elevator deflection is denoted by δ_e.

If the longitudinal controller is of the form shown in Fig. 2.2a with time constant τ and α as free design parameters, two additional state variables δ_e and x_7 indicated in Fig. 2.2b, are introduced.

The actuator and lead-lag are described by the equations

$$\left.\begin{aligned} b_1 &= c_1 - K_3\theta \\ b_2 &= -b_1 + K_1 b_1 \\ b_3 &= K_2 b_2 - K_2 x_7 \\ b_4 &= K_1 b_1 - x_7 \end{aligned}\right\} \qquad (2.5a)$$

$$\left.\begin{aligned} \dot{x}_7 &= b_3 \\ \dot{x}_6 &= -\frac{x_6}{\tau_a} + \frac{b_4}{\tau_a} \end{aligned}\right\} \qquad (2.5b)$$

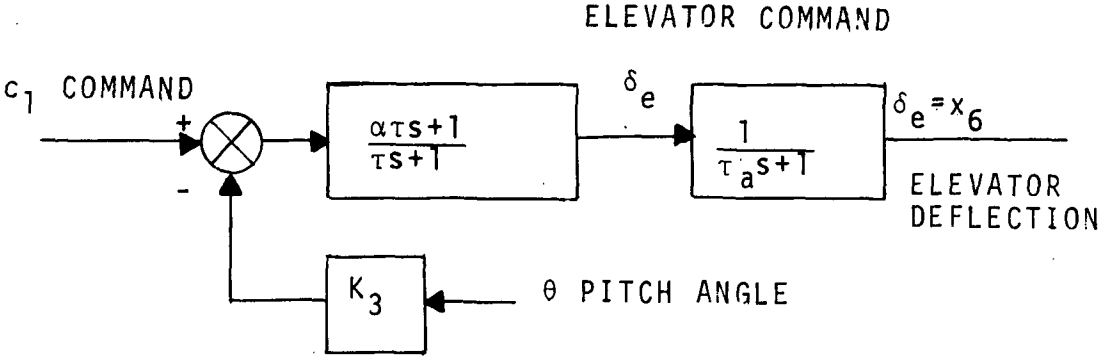
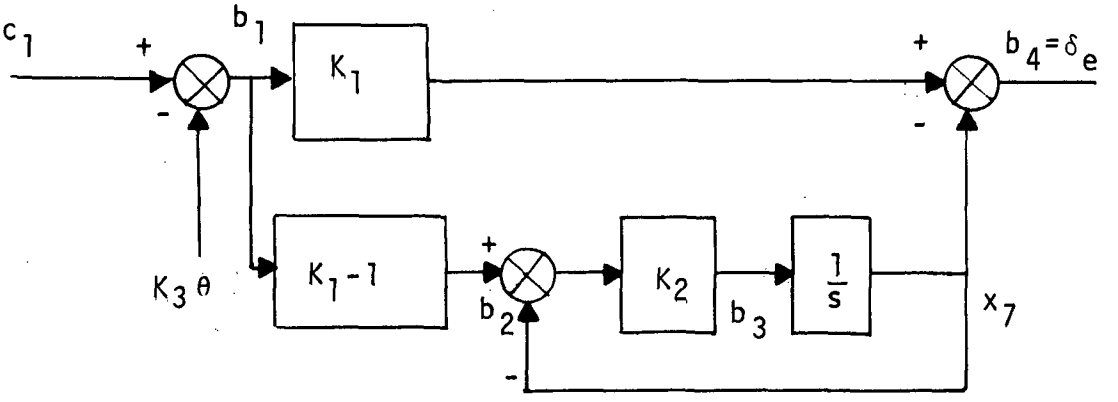


Fig. 2.2a Block Diagram of Simplified Controller



$K_1 = \alpha$; $\alpha > 1$ $K_2 = 1/\tau$

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

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

$$\underline{x}_s^T = ['u'\alpha\ \theta\ q\ h\ x_6\ x_7]$$

$$\underline{b}^T = [b_1\ b_2\ b_3\ b_4]$$

The non zero elements can be written down by inspection. For equation 2.1, these are;

1. M matrix This is a 7×7 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 (7×7)

$N_{11} = -0.088$, $N_{12} = 0.392$, $N_{13} = -0.74$

$N_{21} = -1.48$, $N_{22} = 4.46$, $N_{26} = -0.246$

$N_{31} = -0.619$, $N_{36} = -0.710$

$N_{44} = 1$

$N_{52} = -1$, $N_{53} = 1$

$N_{66} = -\frac{1}{\tau_a}$

3. P Matrix (7×4)

$P_{64} = \frac{1}{\tau_a}$, $P_{73} = 1$

4. Q matrix (7×1) 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

$$\dot{\underline{x}}_m = F_m \underline{x}_m + Q_m \underline{c} \tag{2.6}$$

$$\underline{y}_m = S_m \underline{x}_m \tag{2.7}$$

The error between the model and system response is defined as

$$\underline{e} = \underline{y}_s = \underline{y}_m$$

or

$$\underline{e} = S_s \underline{x}_s + S_m \underline{x}_m \qquad (2.8b)$$

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

$$\underline{u} = S_u \underline{x}_s \qquad (2.9)$$

The command inputs $\underline{c}$ are taken to be impulse inputs, i.e.

$$\underline{c} = \delta(t)\underline{w} \qquad (2.10)$$

where $\underline{w}$ 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 $\underline{b}$ from (2.2), into (2.1), and pre-multiplying by M^{-1}, the system of equations can be written as

$$\dot{\underline{x}}_s = F_s \underline{x}_s + Q_s \underline{c} \qquad (2.11)$$

An augmented state vector is defined as

$$\underline{x} = \begin{bmatrix} \underline{x}_s \\ -- \\ \underline{x}_m \end{bmatrix} \tag{2.12}$$

The equations (2.11), and (2.6), for the system and model can be combined and written as

$$\dot{\underline{x}} = F\underline{x} + G\underline{c} \tag{2.13}$$

where the augmented system differential transition matrix

$$F = \left[\begin{array}{c|c} F_s & 0 \\ \hline 0 & F_m \end{array}\right] \tag{2.14}$$

and the augmented control matrix is

$$G = \begin{bmatrix} Q_s \\ -- \\ Q_m \end{bmatrix} \tag{2.15}$$

Since the impulse response is being considered here, the solution to equation (2.13), with initial conditions $\underline{x}(0)=\underline{0}$ can be obtained by solution of

$$\dot{\underline{x}} = F\underline{x} \tag{2.16}$$

with initial conditions

$$\underline{x}(0) = G\underline{w} \tag{2.17}$$

2.3 Formulation of J in terms of $\underline{x}$

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 $\underline{x}$ as follows:

From equation 2.8b, $\underline{e}$ and $\dot{\underline{e}}$ can be written as

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

$$\dot{\underline{e}} = [S_sF_s \mid S_mF_m]\ \underline{x} \tag{2.19}$$

and from equation 2.9 $\underline{u}$ can be written as

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

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

null matrix.

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

$$J = \int_0^\infty \underline{x}^T \left\{ \left[\begin{array}{c} S_s^T \\ -- \\ S_m^T \end{array}\right] W_e \left[S_s \mid S_m\right] + \left[\begin{array}{c} F_s^T S_s^T \\ ---- \\ F_m^T S_m^T \end{array}\right] \beta W_{\dot{e}} \left[S_s F_s \mid S_m F_m\right] \right.$$

$$\left. + \left[\begin{array}{c} S_u^T \\ -- \\ 0 \end{array}\right] W_u \left[S_u \mid 0\right] \right\} \underline{x} \, dt \qquad (2.21)$$

For the performance index.

Defining Q as the symmetric positive-semidefinite matrix.

$$Q = \left\{ \left[\begin{array}{c} S_s^T \\ -- \\ S_m^T \end{array}\right] W_e \left[S_s \mid S_m\right] + \left[\begin{array}{c} F_s^T S_s^T \\ ---- \\ F_m^T S_m^T \end{array}\right] \beta W_{\dot{e}} \left[S_s F_s \mid S_m F_m\right] \right.$$

$$\left. + \left[\begin{array}{c} S_u^T \\ -- \\ 0 \end{array}\right] W_u \left[S_u \mid 0\right] \right\} \qquad (2.22)$$

The expression for the performance index can be written more

compactly as

$$J = \int_0^\infty ||\underline{x}(t)||_Q^2 \, dt \qquad (2.23)$$

where $\underline{x}(t)$ 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

$$J = \underline{x}^T(0)P\underline{x}(0) \; ; \; \underline{x}(0) = G\underline{w} \qquad (2.24)$$

where P is given by the solution to the matrix algebraic equation

$$F^TP+PF = -Q \qquad (2.25)$$

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

$$\dot{\underline{x}}_s = M^{-1}N\underline{x}_s + M^{-1}P\underline{b} + M^{-1}Q'\underline{c} \tag{2.26}$$

$$[I-S-\sum_{i=1}^{n} K_iH_i]\underline{b} = R\underline{x}_s + T\underline{c} + \left(\sum_{i=1}^{n} K_iE_i\right)\underline{x}_s \tag{2.27}$$

define

$$\tilde{N} = \left[I-S-\sum_{i=1}^{n} K_iH_i\right] \tag{2.28}$$

and

$$\tilde{M} = \tilde{N}^{-1} \tag{2.29}$$

then

$$\underline{b} = \tilde{M}[R\underline{x}_s + T\underline{c} + \left(\sum_{i=1}^{n} K_iE_i\right)\underline{x}_s] \tag{2.30}$$

substitution into (2.26), gives

$$\dot{\underline{x}}_s = [M^{-1}N+M^{-1}P\tilde{M}R+M^{-1}P\tilde{M}\sum_{i=1}^{n} K_iE_i]\underline{x}_s+[M^{-1}P\tilde{M}T+M^{-1}Q']\underline{c} \tag{2.31}$$

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

$$F_s = [M^{-1}N + M^{-1}P\tilde{M}R + M^{-1}P\tilde{M}\sum_{i=1}^{n} K_iE_i] \tag{2.32}$$

$$Q_s = [M^{-1}\tilde{P}\tilde{M}T + M^{-1}Q'] \tag{2.33}$$

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

$$\frac{\partial \tilde{M}}{\partial K_i} = -\tilde{N}^{-1}\frac{\partial N}{\partial K_i}\tilde{N}^{-1} = -\tilde{M}\frac{\partial \tilde{N}}{\partial K_i}\tilde{M} = \tilde{M}H_i\tilde{M} \tag{2.34}$$

then

$$\frac{\partial F_s}{\partial K_i} = M^{-1}\tilde{P}\tilde{M}H_i[\tilde{M}R+\tilde{M}\sum_{i=1}^{n}K_iE_i] + M^{-1}\tilde{P}\tilde{M}E_i \tag{2.35}$$

Similarly

$$\frac{\partial Q_s}{\partial K_i} = M^{-1}\tilde{P}\tilde{M}H_iMT \tag{2.36}$$

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

$$\frac{\partial F}{\partial K_i} = \left[\begin{array}{c|c} \dfrac{\partial F_s}{\partial K_i} & 0 \\ \hline 0 & 0 \end{array}\right] \tag{2.37a}$$

$$\frac{\partial G}{\partial K_i} = \left[\begin{array}{c} \dfrac{\partial Q_s}{\partial K_i} \\ \hline 0 \end{array}\right] \tag{2.37b}$$

The gradient of the Q matrix defined in equation (2.22)

with respect to K_i is

$$\frac{\partial Q}{\partial K_i} = \begin{bmatrix} \frac{\partial F_s^T}{\partial K_i} S_s^T \\ \hline \\ 0 \end{bmatrix} \beta W_{e_{\cdot}} [S_s F_s \,|\, S_m F_m] + \begin{bmatrix} F_s^T S_s^T \\ \hline \\ F_m^T S_m^T \end{bmatrix} \beta W_{e_{\cdot}} [S_s \frac{\partial F_s}{\partial K_i} \,|\, 0] \qquad (2.38a)$$

If the weighting matrix $W_{e_{\cdot}}$, is a diagonal matrix (or symmetric) then,

$$\frac{\partial Q}{\partial K_i} = \begin{bmatrix} \left(\frac{\partial F_s}{\partial K_i}\right)^T S_s^T \\ \hline \\ 0 \end{bmatrix} \beta W_{e_{\cdot}} [S_s F_s \,|\, S_m F_m] +$$

$$+ \left\{ \begin{bmatrix} \left(\frac{\partial F_s}{\partial K_i}\right)^T S_s^T \\ \hline \\ 0 \end{bmatrix} \beta W_{e_{\cdot}} [S_s F_s \,|\, S_m F_m] \right\}^T \qquad (2.38b)$$

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

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

where λ 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,

$$\frac{\partial J}{\partial K_i} = \frac{\partial \underline{x}^T(0)}{\partial K_i} P\underline{x}(0) + \underline{x}(0) \frac{\partial P}{\partial K_i} \underline{x}(0) + x^T(0) P \frac{\partial \underline{x}(0)}{\partial K_i} \tag{2.40a}$$

or since P is symmetric

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

P and $\partial P/\partial K_i$ are evaluated successively from

$$F^T P + P F^T = -Q \tag{2.41}$$

and

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

The gradient of the initial condition vector is given by

$$\frac{\partial \underline{x}(0)}{\partial K_i} = \frac{\partial G}{\partial K_i} \underline{w} \qquad (2.43)$$

Once $\partial P/\partial K_i$, P and $\partial \underline{x}(0)/\partial K_i$ have been evaluated $\partial J/\partial K_i$ and ΔK_i for i = 1,2,...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,...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 $\underline{x}(0)$ with respect to K_1 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 $\partial P/\partial K_1$.
6. $\partial J/\partial K_1$ is evaluated from (2.40b), and ΔK_1 from (2.39).
7. Steps 4 through 6 are repeated for i=2,3,...n
8. The predicted change in the performance index J is then evaluated according to

$$\Delta J = -\lambda\left[\left(\frac{\partial J}{\partial K_1}\right)^2+\left(\frac{\partial J}{\partial K_2}\right)^2+\ldots\left(\frac{\partial J}{\partial K_n}\right)^2\right]$$

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.

CHAPTER 3

Program Description

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 inputing 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

$$\Lambda = \begin{bmatrix} L_1 & M_{12} & M_{13} & & M_{1l} \\ & L_2 & & & \\ & & L_3 & & \\ & & & \ddots & \\ & & & & L_r \end{bmatrix} \qquad (3.1)$$

where the L_i's are the companion matrices

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

and the off diagonal blocks M_{1i} are of the form

$$M_{1i} = \begin{bmatrix} 0 & 0 & \cdots\cdots\cdots & 0 & p_{1i} \\ 0 & 0 & \cdots\cdots\cdots & 0 & p_{2i} \\ 0 & 0 & \cdots\cdots\cdots & 0 & p_{3i} \\ \cdots & \cdots & \cdots\cdots\cdots & \cdots & \cdots \\ 0 & 0 & \cdots\cdots\cdots & 0 & p_{ki} \end{bmatrix} \qquad (3.3)$$

Due to the quasi-triangular character of Λ, 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. <u>The solution of the matrix equation</u>

$$F^T P + PF = -Q \tag{3.4}$$

for P is a special case of the solution to

$$AX + XB = -C \tag{3.5}$$

The program flow for this stage is described in greater

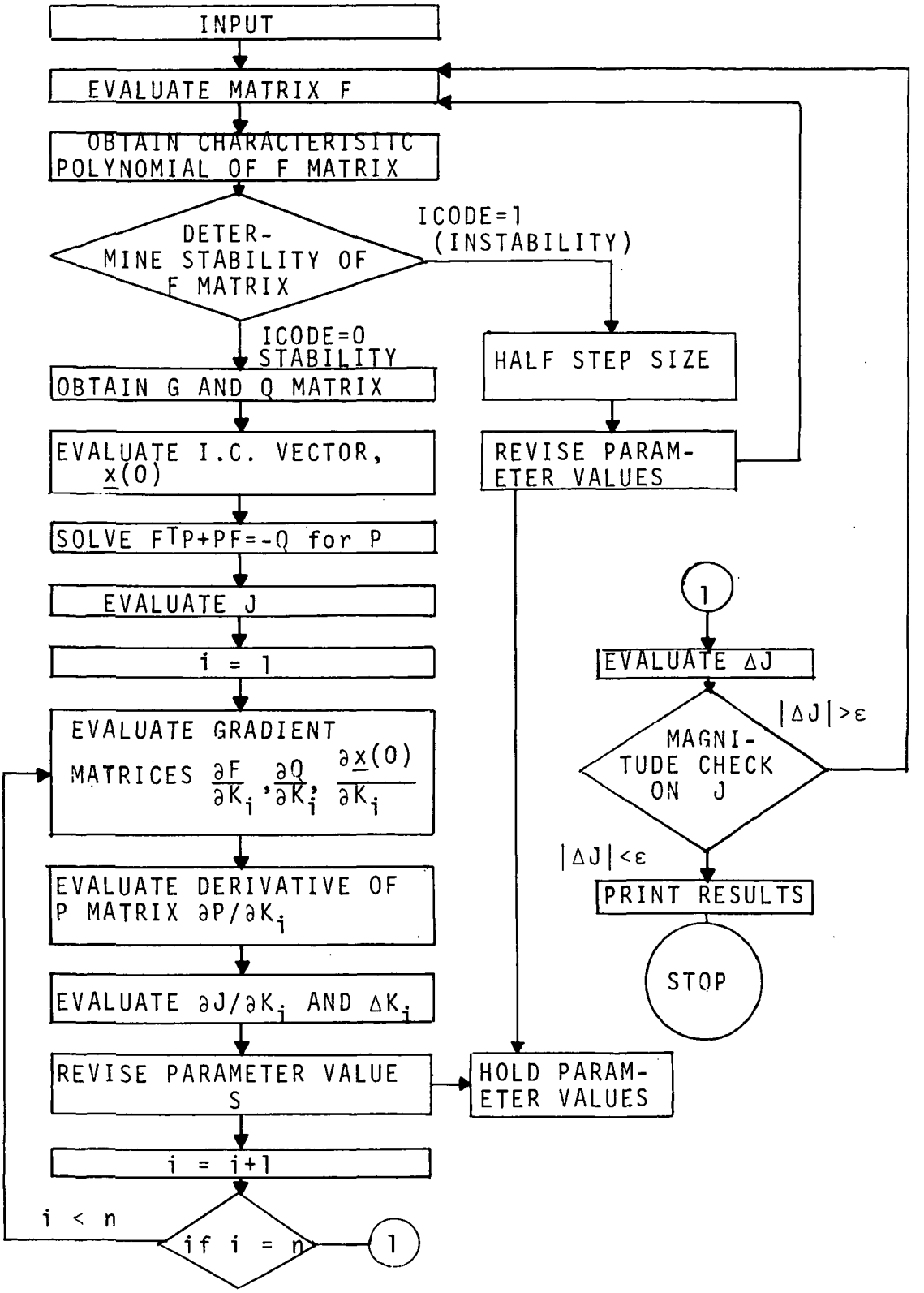


Fig. 3.1 Overall Program Flow Logic

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.

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

$$= \begin{bmatrix} -Q_{11} & -Q_{12} \\ -Q_{21} & -Q_{22} \end{bmatrix} \qquad (3.5)$$

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

$$F_s^T P_{11} + P_{11} F_s = -Q_{11}$$

$$F_s^T P_{12} + P_{12} F_m = -Q_{12}$$

$$F_m^T P_{21} + P_{21} F_s = -Q_{21}$$

$$F_m^T P_{22} + P_{22} F_m = -Q_{22} \qquad (3.6)$$

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

$$P_{22} = -Q_{22}/2F_m \tag{3.7}$$

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

$$\underline{P}_{12} = [F_s^T + IF_m]^{-1}\underline{Q}_{12} \tag{3.8}$$

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,...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 RSCHEK 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

$$AX + XB = -C \tag{3.9}$$

where A and B are arbitrary square matrices, of dimensions n×n and m×m respectively. C is a known n×n matrix while X is the unknown n×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

$$F^T P + PF = -Q \tag{3.10}$$

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

$$X = UV^{-1} \tag{3.11}$$

where V^{-1} is a square matrix of dimension m×m and U is rectangular of dimension n×m.

Substitution into (3.9) leads to

$$AU + CV = -U\Lambda \tag{3.12}$$

where

$$\Lambda = V^{-1}BV \tag{3.13}$$

Referring to Fig. 3.2, the first step is then to obtain

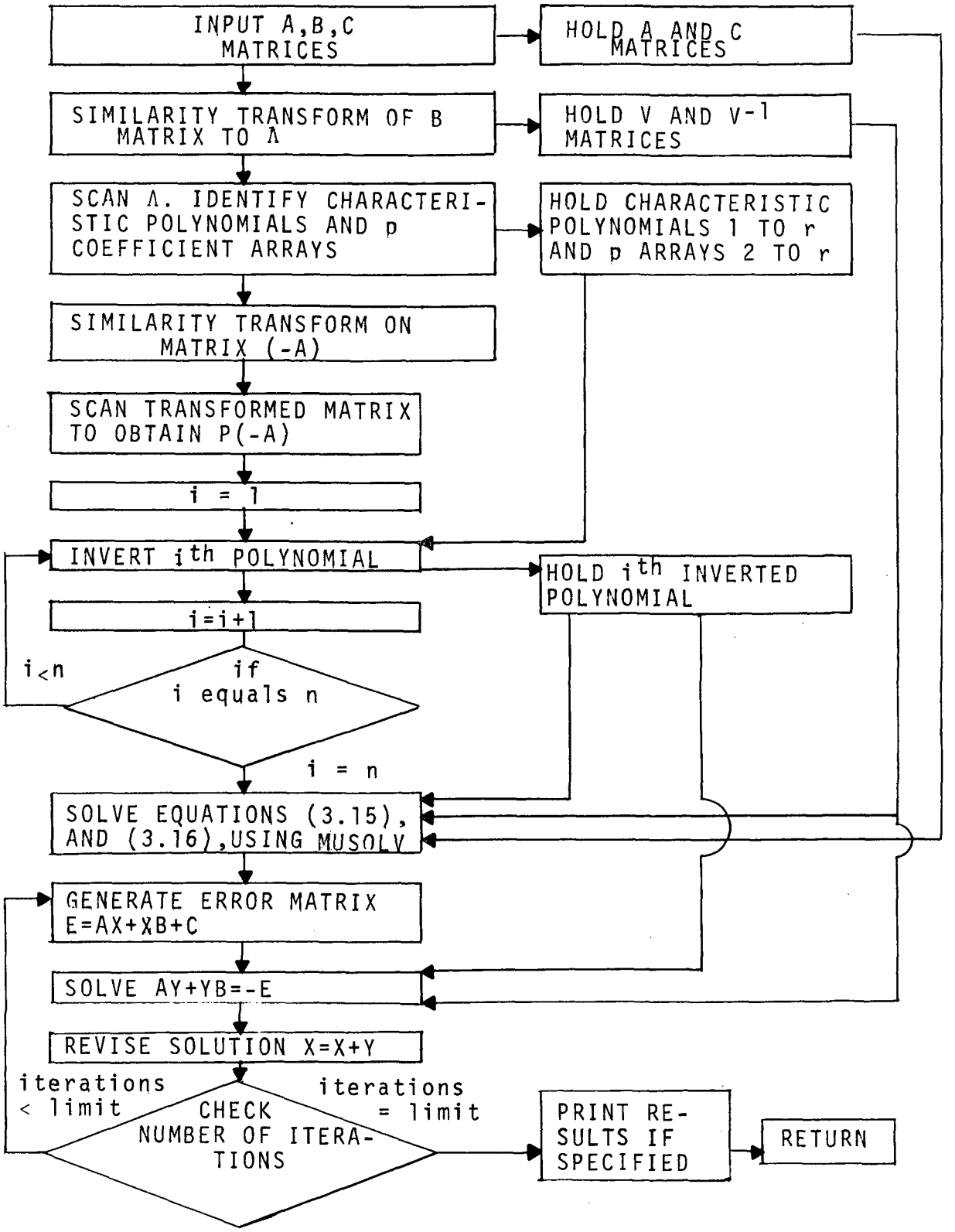


Fig. 3.2 Program Flow Logic for Solution of Matrix Algebraic Equation AX+XB=-C

Λ by a sequence of similarity transformations. The general form of the Λ 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

$$U = [\underline{u}_1 \mid \underline{u}_2 \ \ldots\ldots \ \underline{u}_m]$$

$$CV = [\underline{v}_1 \mid \underline{v}_2 \ \ldots\ldots \ \underline{v}_m] \qquad (3.14)$$

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.

$$\begin{aligned}
A\underline{u}_1 \quad & +\underline{v}_1 \quad = -u_2 \\
A\underline{u}_2 \quad & +\underline{v}_2 \quad = -u_3 \\
& \vdots \\
A\underline{u}_{k-1} \quad & +\underline{v}_{k-1} \quad = -\underline{u}_k \\
A\underline{u}_k \quad & +\underline{v}_k \quad = = a_{11}u_1 + a_{21}\underline{u}_2 + \ldots + a_{k1}\underline{u}_k \\
A\underline{u}_{k+1} \quad & +\underline{v}_k \quad = -\underline{u}_{k+2} \\
& \vdots \\
A\underline{u}_{k+\ell} \quad & +\underline{v}_{k+\ell} \quad = -p_{12}\underline{u}_1 - p_{22}\underline{u}_2 - \ldots - p_{k2}\underline{u}_k + a_{12}u_{k+1} + a_{22}\underline{u}_{k+2} \\
& \qquad\qquad\qquad\quad + \ldots a_{\ell 2}\underline{u}_{k+\ell} \\
\vdots \qquad & \qquad\qquad\qquad\quad \vdots
\end{aligned}$$

$$
\begin{aligned}
&\vdots \qquad\qquad\qquad\qquad \vdots \\
&A\underline{u}_{m-q+1} + \underline{v}_{m-q-1} = -\underline{u}_{m-q+2} \\
&\vdots \\
&A\underline{u}_{m} \quad + \underline{v}_{m} \quad = -p_{1r}\underline{u}_1 - p_{2r}\underline{u}_2 - \cdots p_{(m-q)r}\underline{u}_{m-q} + a_{1r}\underline{u}_{m-q+1} \\
&\qquad\qquad\qquad\qquad + a_{2r}\underline{u}_{m-q+2} + \ldots + a_{qr}\underline{u}_{m} \qquad\qquad (3.15)
\end{aligned}
$$

In the above, the dimension of the first diagonal block is k that of the 2nd ℓ and that of the rth is q. Taking $\underline{u}_1$, $\underline{u}_{k+1} \cdots \underline{u}_{k+\ell+1} \cdots \underline{u}_{m-q+1}$ as unknowns and substituting for the other vectors of $\underline{u}$ in terms of these into kth $(k+\ell)^{th}$ and m^{th} equation of (3.15), leads to

$$
\begin{aligned}
&[(-A)^k + a_{k1}(-A)^{k-1} + \ldots + a_{21}(-A) + a_{11}I]\underline{u}_1 \\
&\qquad\qquad = \sum_{j=1}^{k} \sum_{h=j+1}^{k+1} a_{h1}(-A)^{h-(j+1)}\underline{v}_j \\
&[(-A) + a_{\ell 2}(-A)^{\ell-1} + \ldots + a_{22}(-A) + a_{12}I]\underline{u}_{k+1} \\
&\qquad\qquad = \sum_{j=1}^{\ell} \sum_{h=j+1}^{\ell+1} a_{h2}(-A)^{h-(j+1)}\underline{v}_{j+k} + \sum_{i=1}^{k} p_{12}\underline{u}_i \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
&[(-A)^q + a_{qr}(-A)^{q-1} + \ldots + a_{2r}(-A) + a_{1i}I]\underline{u}_{m-q+1} \\
&\qquad\qquad = \sum_{j=1}^{q} \sum_{h=j+1}^{q+1} a_{hr}(-A)^{h-(j+1)}\underline{v}_{j+m-q} + \sum_{i=1}^{m-q} p_{1i}\underline{u}_i \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3.16)
\end{aligned}
$$

In particular if Λ is composed of a single companion block, then

$$[p_B(-A)]u_1 = \sum_{j=1}^{m} \sum_{h=j+1}^{m+1} a_h (-A)^{h-(j+1)} \underline{v}_j \qquad (3.17)$$

The polynomials on the left hand side of the equation in (3.16), are simply the characteristic polynomials of the individual companion blocks with λ 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 Λ 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 $\underline{u}_1$. $\underline{u}_2$, $\underline{u}_3$ ... $\underline{u}_k$ are then determined from (3.15). These in turn are used in the second equation of (3.16), to obtain $\underline{u}_{k+1}$ and so on for the r blocks.

The solution for $\underline{u}_1$ $\underline{u}_{k+1} \ldots u_{m-q+1}$ 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.

$$P(x) = x^m + a_m x^{m-1} + a_{m-1} x^{m-2} + \ldots + a_2 x + a_1$$

$$Q(x) = x^p + b_p x^{p-1} + b_{p-1} x^{p-2} + \ldots + b_2^2 x + b_1 \qquad (3.18)$$

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

$$\tilde{Q}(x)P(x) + \tilde{P}(x)Q(x) = k \qquad (3.19)$$

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

$$\tilde{P}(-A)Q(-A) = kI \qquad (3.20)$$

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

$$Q(-A)^{-1} = \tilde{P}(-A) \qquad (3.21)$$

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

$$E = AX + XB + C \tag{3.22}$$

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

$$AY + YB = -E \tag{3.23}$$

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 preceeding 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

$$\left.\begin{aligned} \dot{x}_1 &= -2x_1 + b_2 \\ \dot{x}_2 &= x_1 \\ \dot{x}_3 &= -x_3 + x_2 \end{aligned}\right\} \qquad (4.1a)$$

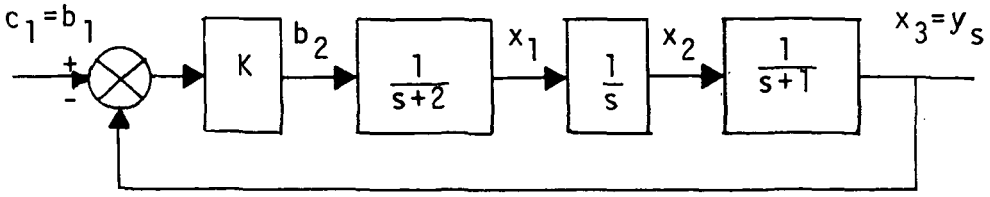


Fig. 4.1a Block Diagram of Closed Loop System with Position Feedback. K is a Free Design Parameter.

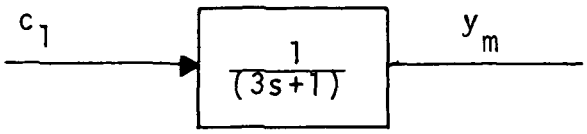


Fig. 4.1b The Nominal Transfer Function

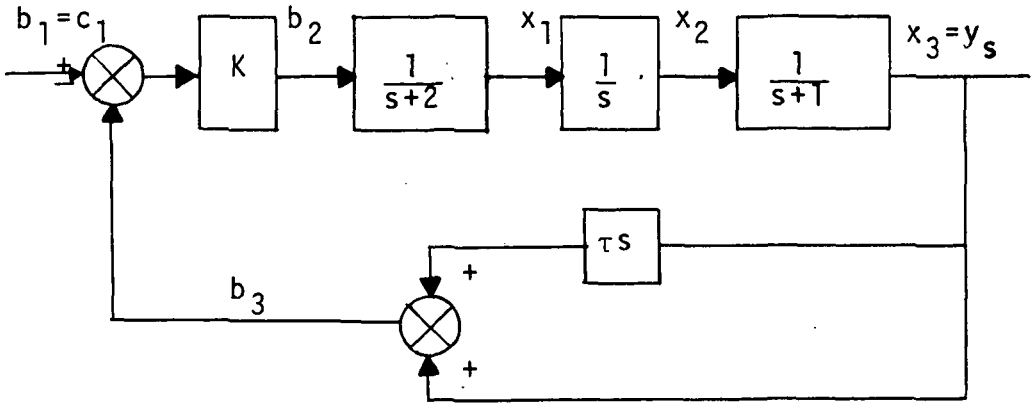


Fig. 4.2 Closed Loop System with Rate Feedback Added K and τ are free parameters.

$$b_1 = c_1$$

$$b_2 = Kb_1 - Kx_3 \qquad (4.1b)$$

The model is described by

$$\dot{x}_m = -0.33\ x_m + 0.33\ c_1 \qquad (4.2)$$

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

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{\underline{x}}_s = \begin{bmatrix} -2 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix} \underline{x}_s + \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \underline{b} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \underline{c} \qquad (4.3a)$$

$$\underline{b} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \underline{x}_s + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \underline{b} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \underline{c} + K_1 \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \underline{b} + K_1 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \underline{x}_s \qquad (4.3b)$$

where

$$\underline{x}_s^T = [x_1\ x_2\ x_3] \quad \text{and} \quad \underline{b}^T = [b_1\ b_2]$$

The system and model outputs are defined by

$$\underline{y}_s = [0 \quad 0 \quad 1]\underline{x}_s \qquad (4.4a)$$

$$y_m = -x_m \qquad (4.4b)$$

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 paramter(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 inital 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 τ 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 τ. The values as adjusted by the program were 0.795 for K and 1.26 for τ.

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.

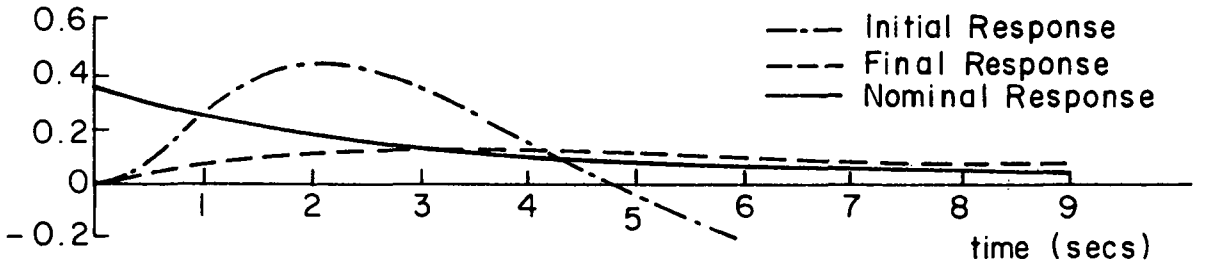


Fig. 4.3a CLOSED LOOP SYSTEM RESPONSE WITHOUT RATE FEEDBACK

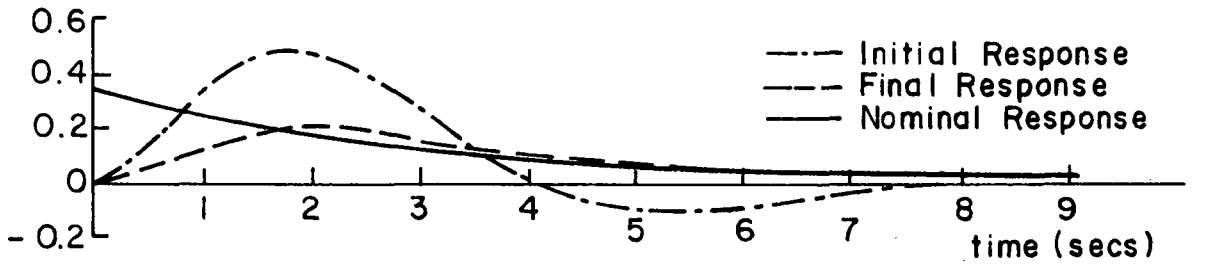


Fig. 4.3b CLOSED LOOP SYSTEM RESPONSE WITH RATE FEEDBACK

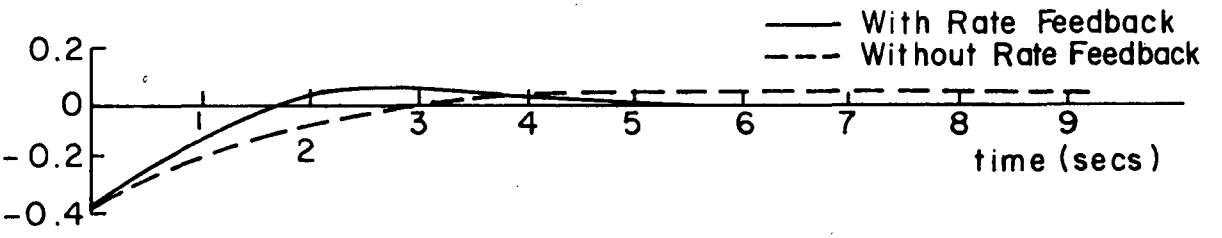


Fig. 4.3c SYSTEM RESPONSE ERROR WITH AND WITHOUT RATE FEEDBACK

CHAPTER 5

Conclusions and Recommendations

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, $W_{\dot{e}}$, and the control signal weighting matrix W_u.

```
&WGTERR  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 ≤ 2

KX is the dimension of the system state vector ≤ 10

KM is the dimension of the model state vector ≤ 5

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

KC is the dimension of the command vector ≤ 5

KB is the dimension of the bus vector ≤ 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^{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.

```
&CONTRL VCOMM(J) =   ,&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),XO(20),DELFS(10,10),DELQ(20,20),DELF(20,20),
1DELFT(20,20),P(20,20),TEMP3(10,5),DELGS(10,5),DELG(20,5),DELX(20),
3U(20,20),V(20,20),VINV(20,20),A(20,20),B(20,20),XSOLN(20,20),
3RHS(20,20),QINV(5,21),QB(5,21),PB(5,21),SUM,SUM1,SIGMA,PI,DELJ,
4DELTAJ,RH11(10,10),RH12(10,5),RH21(5,10),RH22(5,5),SYSG(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),PI1
 INTEGER*4 IDENT(5),IDIMQB(5),IDIMPB(5),IDIMQI(5)
 DOUBLE PRECISION SYSM(10,10),SYSN(10,10),SYSP(10,10),SYSQ(10,5),
1SYSP1(10,10),SYSN1(10,10),SYSQ1(10,5),TEMP(10,10),S1(10,10),X(11),
2SINV(10,10),R(10,10),S(10,10),T(10,5),R1(10,10),T1(10,5),XDOT(11),
3HK(10,10),EK(10,10),H(10,10,10),E(10,10,10),SYSF(10,10),
4 FMOD(5,5),FT(20,20),F(20,20),F1(20,20),F2(20,20),ARRAYF(21),
5 ROUTH1(21),ROUTH2(21),SYSS(5,10),SMOD(5,5),SFS(5,10),SFM(5,5),
6SF(5,20),SFT(20,5),SS(5,20),SST(20,5),W1(5,5),W2(5,5),W3(5,5),
7TEMP1(20,5),TEMP2(20,20),SU(5,10),SU1(5,20),SU1T(20,5),Q(20,20),
8F3(20,20),EK1(10,10),VCOMM(5),PARAM(10),QMOD(5,5),BETA,STEP
 COMMON /HOLD/U,V/HOLD1/A,C/HOLD2/QINV,IDIMQI
1/HOLD3/QB,PB,IDIMQB,IDIMPB,IDENT/HOLD4/VINV,B/SECTR1/XSOLN
 EQUIVALENCE (HK(1),EK(1)),(FT(1),F2(1)),(Q(1),F3(1)),
1(TEMP2(1),F1(1))
 DATA SYSM/100*0.0/,SYSN/100*0.0/,SYSP/100*0.0/,SYSQ/50*0.0/,
1R/100*0.0/,S/100*0.0/,T/50*0.0/,E/1000*0.0/,H/1000*0.0/,
2X/1.0,10*0.0/,XDOT/1.0,10*0.0/,W1/25*0.0/,W2/25*0.0/,W3/25*0.0/,
3FMOD/25*0.0/,SMOD/25*0.0/,SU/50*0.0/,SYSS/50*0.0/,PARAM/10*0.0/,
4VCOMM/5*0.0/,QMOD/25*0.0/
 NAMELIST /INPUTM/SYSM/INPUTN/SYSN/INPUTP/SYSP/INPUTQ/SYSQ/
1INPUTR/R/INPUTS/S/INPUTT/T/INPUTH/H/INPUTE/E/MODEF/FMOD/MODEQ/
2QMOD/SYSOUT/SYSS/MODOUT/SMOD/WGTERR/W1/WGTEDT/W2,BETA/WGTU/W3/
3DIMEN/KE,KX,KM,KX1,KC,KB/GAIN/IGAIN,PARAM,STEP/CONTRL/VCOMM/
4INPUTU/SU
 READ(5,INPUTM)
 READ(5,INPUTN)
 READ(5,INPUTP)
 READ(5,INPUTQ)
 READ(5,INPUTR)
```

```
      READ(5,INPUTS)                                                    MAIN0037
      READ(5,INPUTT)                                                    MAIN0038
      READ(5,INPUTH)                                                    MAIN0039
      READ(5,INPUTE)                                                    MAIN0040
      READ(5,MODEF)                                                     MAIN0041
      READ(5,MODEQ)                                                     MAIN0042
      READ(5,SYSOUT)                                                    MAIN0043
      READ(5,MODOUT)                                                    MAIN0044
      READ(5,WGTERR)                                                    MAIN0045
      READ(5,WGTEDT)                                                    MAIN0046
      READ(5,WGTU)                                                      MAIN0047
      READ(5,DIMEN)                                                     MAIN0048
      READ(5,GAIN)                                                      MAIN0049
      READ(5,INPUTU)                                                    MAIN0050
      READ(5,CONTRL)                                                    MAIN0051
      PI=0.0                                                            MAIN0052
      L4=20                                                             MAIN0053
      L2=5                                                              MAIN0054
      L1=10                                                             MAIN0055
      ITTER=0                                                           MAIN0056
      ITTER1=0                                                          MAIN0057
      ISTOP=0                                                           MAIN0058
      CALL SIMEQ(SYSM,XDOT,KX,SINV,X,IERR)                              MAIN0059
      CALL MXMULT(SINV,SYSN,SYSN1,KX,KX,KX,KX,M,N,L1,L1,L1,L1)          MAIN0060
      CALL MXMULT(SINV,SYSP,SYSP1,KX,KX,KX,KB,M,N,L1,L1,L1,L1)          MAIN0061
      CALL MXMULT(SINV,SYSQ,SYSQ1,KX,KX,KX,KC,M,N,L1,L1,L1,L2)          MAIN0062
    5 DO 20 I=1,KB                                                      MAIN0063
      DO 20 J=1,KB                                                      MAIN0064
      IF(I.EQ.J)GO TO 10                                                MAIN0065
      S1(I,J)=-S(I,J)                                                   MAIN0066
      GO TO 20                                                          MAIN0067
   10 S1(I,J)=1.0-S(I,J)                                                MAIN0068
   20 CONTINUE                                                          MAIN0069
   30 DO 40 K=1,IGAIN                                                   MAIN0070
      DO 40 I=1,KB                                                      MAIN0071
      DO 40 J=1,KB                                                      MAIN0072
```

```
      HK(I,J)=H(I,J,K)*PARAM(K)                                          MAIN0073
   40 S1(I,J)=S1(I,J)-HK(I,J)                                            MAIN0074
      CALL SIMEQ(S1,XDOT,KB,SINV,X,IERR)                                 MAIN0075
      CALL MXMULT(SINV,R,R1,KB,KB,KB,KX,M,N,L1,L1,L1,L1)                 MAIN0076
      DO 50 I=1,KB                                                       MAIN0077
      DO 50 J=1,KX                                                       MAIN0078
   50 EK(I,J)=E(I,J,1)*PARAM(1)                                          MAIN0079
      IF(IGAIN.EQ.1)GO TO 61                                             MAIN0080
      DO 60 K=2,IGAIN                                                    MAIN0081
      DO 60 I=1,KB                                                       MAIN0082
      DO 60 J=1,KX                                                       MAIN0083
   60 EK(I,J)=EK(I,J)+E(I,J,K)*PARAM(K)                                  MAIN0084
   61 CALL MXMULT(SINV,EK,EK1,KB,KB,KB,KX,M,N,L1,L1,L1,L1)               MAIN0085
      CALL MXMULT(SYSP1,EK1,SYSF,KX,KB,KB,KX,M,N,L1,L1,L1,L1)            MAIN0086
      CALL MXMULT(SYSP1,R1,TEMP,KX,KB,KB,KX,M,N,L1,L1,L1,L1)             MAIN0087
      DO 70 I=1,KX                                                       MAIN0088
      DO 70 J=1,KX                                                       MAIN0089
   70 SYSF(I,J)=SYSF(I,J)+TEMP(I,J)+SYSN1(I,J)                           MAIN0090
      DO 100 I=1,KX                                                      MAIN0091
      DO 100 J=1,KX1                                                     MAIN0092
      IF(J.GT.KX)GO TO 90                                                MAIN0093
      F(I,J)=SYSF(I,J)                                                   MAIN0094
      F1(I,J)=F(I,J)                                                     MAIN0095
      GO TO 100                                                          MAIN0096
   90 F(I,J)=0.0                                                         MAIN0097
      F1(I,J)=0.0                                                        MAIN0098
  100 CONTINUE                                                           MAIN0099
      IR=KX+1                                                            MAIN0100
      DO 120 I=IR,KX1                                                    MAIN0101
      DO 120 J=1,KX1                                                     MAIN0102
      IF(J.LE.KX)GO TO 110                                               MAIN0103
      JJ=J-KX                                                            MAIN0104
      II=I-KX                                                            MAIN0105
      F(I,J)=FMOD(II,JJ)                                                 MAIN0106
      F1(I,J)=F(I,J)                                                     MAIN0107
      GO TO 120                                                          MAIN0108
```

```
  110 F(I,J)=0.0                                                          MAIN0109
      F1(I,J)=0.0                                                         MAIN0110
  120 CONTINUE                                                            MAIN0111
C                                                                         MAIN0112
C           INVESTIGATE MODSYS ASYMPTOTIC STABILITY                       MAIN0113
C                                                                         MAIN0114
      CALL SMLTRN(F1,F2,F3,KX1,INDXF)                                     MAIN0115
      CALL SCAN1(F1,ARRAYF,IDIMF,KX1)                                     MAIN0116
      CALL RSCHEK(ARRAYF,IDIMF,ROUTH1,IDIMR1,ROUTH2,IDIMR2,ICODE)         MAIN0117
      DO 130 I=1,KX1                                                      MAIN0118
      DO 130 J=1,KX1                                                      MAIN0119
  130 FT(J,I)=F(I,J)                                                      MAIN0120
      CALL MXMULT(SYSS,SYSF,SFS,KE,KX,KX,KX,M,N,L2,L1,L1,L1)              MAIN0121
      CALL MXMULT(SMOD,FMOD,SFM,KE,KM,KM,KM,M,N,L2,L2,L2,L2)              MAIN0122
C                                                                         MAIN0123
C        F MATRIX PARTITION                                               MAIN0124
C                                                                         MAIN0125
      DO 150 I=1,KE                                                       MAIN0126
      DO 150 J=1,KX1                                                      MAIN0127
      IF(J.GT.KX)GO TO 140                                                MAIN0128
      SF(I,J)=SFS(I,J)                                                    MAIN0129
      SFT(J,I)=SF(I,J)                                                    MAIN0130
      SS(I,J)=SYSS(I,J)                                                   MAIN0131
      SST(J,I)=SS(I,J)                                                    MAIN0132
      GO TO 150                                                           MAIN0133
  140 JJ=J-KX                                                             MAIN0134
      SF(I,J)=SFM(I,JJ)                                                   MAIN0135
      SFT(J,I)=SF(I,J)                                                    MAIN0136
      SS(I,J)=SMOD(I,JJ)                                                  MAIN0137
      SST(J,I)=SS(I,J)                                                    MAIN0138
  150 CONTINUE                                                            MAIN0139
      DO 170 I=1,KC                                                       MAIN0140
      DO 170 J=1,KX1                                                      MAIN0141
      IF(J.GT.KX)GO TO 160                                                MAIN0142
      SU1(I,J)=SU(I,J)                                                    MAIN0143
      SU1T(J,I)=SU1(I,J)                                                  MAIN0144
```

```
      GO TO 170                                                          MAIN0145
  160 SU1(I,J)=0.0                                                       MAIN0146
      SU1T(J,I)=SU1(I,J)                                                 MAIN0147
  170 CONTINUE                                                           MAIN0148
      CALL MXMULT(SST,W1,TEMP1,KX1,KE,KE,KE,M,N,L4,L2,L2,L2)             MAIN0149
  171 FORMAT(5X,D10.3)                                                   MAIN0150
      CALL MXMULT(TEMP1,SS,Q,KX1,KE,KE,KX1,M,N,L4,L2,L2,L4)              MAIN0151
      CALL MXMULT(SFT,W2,TEMP1,KX1,KE,KE,KE,M,N,L4,L2,L2,L2)             MAIN0152
      CALL MXMULT(TEMP1,SF,TEMP2,KX1,KE,KE,KX1,M,N,L4,L2,L2,L4)          MAIN0153
      DO 180 I=1,KX1                                                     MAIN0154
      DO 180 J=1,KX1                                                     MAIN0155
  180 Q(I,J)=Q(I,J)+BETA*TEMP2(I,J)                                      MAIN0156
      CALL MXMULT(SU1T,W3,TEMP1,KX1,KC,KC,KC,M,N,L4,L2,L2,L2)            MAIN0157
      CALL MXMULT(TEMP1,SU1,TEMP2,KX1,KC,KC,KX1,M,N,L4,L2,L2,L4)         MAIN0158
      DO 190 I=1,KX1                                                     MAIN0159
      DO 190 J=1,KX1                                                     MAIN0160
  190 Q(I,J)=Q(I,J)+TEMP2(I,J)                                           MAIN0161
      DO 221 I=1,KX                                                      MAIN0162
      DO 221 J=1,KX                                                      MAIN0163
  221 SYSFT(I,J)=SYSF(J,I)                                               MAIN0164
C                                                                        MAIN0165
C        Q MATRIX PARTITION                                              MAIN0166
C                                                                        MAIN0167
  230 DO 231 I=1,KX                                                      MAIN0168
      DO 231 J=1,KX                                                      MAIN0169
  231 RH11(I,J)=Q(I,J)                                                   MAIN0170
      IF(ICODE.EQ.1.AND.ITTER.EQ.0)GO TO 1000                            MAIN0171
      IF(ICODE.EQ.1)GO TO 800                                            MAIN0172
      ITTER1=0                                                           MAIN0173
      DO 238 I=1,KX                                                      MAIN0174
      DO 238 J=1,KX                                                      MAIN0175
      A1(I,J)=SYSFT(I,J)                                                 MAIN0176
      B1(I,J)=SYSF(I,J)                                                  MAIN0177
  238 C1(I,J)=RH11(I,J)                                                  MAIN0178
      CALL MSOLVE(KX,KX,KX,KX,A1,B1,C1,JB1,JB2)                          MAIN0179
      DO 232 I=1,KX                                                      MAIN0180
```

```
      DO 232 J=1,KX                                                    MAIN0181
  232 P(I,J)=XSOLN(I,J)                                                MAIN0182
      KX2=KX+1                                                         MAIN0183
      DO 233 I=1,KX                                                    MAIN0184
      DO 233 J=KX2,KX1                                                 MAIN0185
      JJ=J-KX                                                          MAIN0186
  233 RH12(I,JJ)=Q(I,J)                                                MAIN0187
      DO 234 I=1,KX                                                    MAIN0188
  234 XDOT(I)=-RH12(I,1)                                               MAIN0189
      DO 236 I=1,KX                                                    MAIN0190
      DO 236 J=1,KX                                                    MAIN0191
      IF(I.EQ.J)GO TO 235                                              MAIN0192
      SYSF2(I,J)=SYSFT(I,J)                                            MAIN0193
      GO TO 236                                                        MAIN0194
  235 SYSF2(I,J)=SYSFT(I,J)+FMOD(1,1)                                  MAIN0195
  236 CONTINUE                                                         MAIN0196
      CALL SIMEQ(SYSF2,XDOT,KX,FINV,X,IERR)                            MAIN0197
      DO 237 I=1,KX                                                    MAIN0198
      P(I,KX1)=X(I)                                                    MAIN0199
  237 P(KX1,I)=X(I)                                                    MAIN0200
      P(KX1,KX1)=-Q(KX1,KX1)/(2.0*FMOD(1,1))                           MAIN0201
      CALL MXMULT(SINV,T,T1,KB,KB,KB,KC,M,N,L1,L1,L1,L2)               MAIN0202
      CALL MXMULT(SYSP1,T1,SYSG,KX,KB,KB,KC,M,N,L1,L1,L1,L2)           MAIN0203
      DO 250 I=1,KX                                                    MAIN0204
      DO 250 J=1,KC                                                    MAIN0205
  250 SYSG(I,J)=SYSG(I,J)+SYSQ1(I,J)                                   MAIN0206
      IF(ISTOP.EQ.1)GO TO 520                                          MAIN0207
C                                                                      MAIN0208
C     EVALUATE AUGMENTED G MATRIX                                      MAIN0209
C                                                                      MAIN0210
      DO 270 I=1,KX1                                                   MAIN0211
      DO 270 J=1,KC                                                    MAIN0212
      IF(I.GT.KX)GO TO 260                                             MAIN0213
      SMG(I,J)=SYSG(I,J)                                               MAIN0214
      GO TO 270                                                        MAIN0215
  260 II=I-KX                                                          MAIN0216
```

```
      SMG(I,J)=QMCD(II,J)                                              MAIN0217
  270 CONTINUE                                                         MAIN0218
C                                                                      MAIN0219
C          EVALUATE AUGMENTED INITIAL CONDITION VECTOR                 MAIN0220
C                                                                      MAIN0221
      DO 280 I=1,KX1                                                   MAIN0222
      XO(I)=0.0D+00                                                    MAIN0223
      DO 280 J=1,KC                                                    MAIN0224
  280 XO(I)=XC(I)+SMG(I,J)*VCOMM(J)                                    MAIN0225
      WRITE(6,261)(XO(I),I=1,KX1)                                      MAIN0226
C                                                                      MAIN0227
C          EVALUATE PERFORMANCE INDEX                                  MAIN0228
C                                                                      MAIN0229
      PI1=PI                                                           MAIN0230
      PI=0.0D+00                                                       MAIN0231
      DO 300 I=1,KX1                                                   MAIN0232
      SIGMA=0.0D+00                                                    MAIN0233
      DO 290 J=1,KX1                                                   MAIN0234
  290 SIGMA=SIGMA+P(I,J)*XO(J)                                         MAIN0235
  300 PI=PI+SIGMA*XO(I)                                                MAIN0236
      WRITE(6,511)PI                                                   MAIN0237
      DELTAJ=PI-PI1                                                    MAIN0238
      WRITE(6,511)DELTAJ                                               MAIN0239
C                                                                      MAIN0240
C        EVALUATION OF GRADIANT OF F MATRIX                            MAIN0241
C                                                                      MAIN0242
      DO 320 I=1,KB                                                    MAIN0243
      DO 320 J=1,KX                                                    MAIN0244
  320 R1(I,J)=R1(I,J)+EK1(I,J)                                         MAIN0245
      DELTAJ=0.0D+00                                                   MAIN0246
      DO 500 K=1,IGAIN                                                 MAIN0247
      DO 340 I=1,KB                                                    MAIN0248
      DO 340 J=1,KB                                                    MAIN0249
      SUM=0.0D+00                                                      MAIN0250
      DO 330 II1=1,KB                                                  MAIN0251
  330 SUM=SUM+SINV(I,II1)*H(II1,J,K)                                   MAIN0252
```

```
  340 HK(I,J)=SUM                                                          MAIN0253
      CALL MXMULT(HK,R1,TEMP,KB,KB,KB,KX,M,N,L1,L1,L1,L1)                  MAIN0254
      DO 360 I=1,KB                                                        MAIN0255
      DO 360 J=1,KX                                                        MAIN0256
      SUM=0.0D+00                                                          MAIN0257
      DO 350 II1=1,KB                                                      MAIN0258
  350 SUM=SUM+SINV(I,II1)*E(II1,J,K)                                       MAIN0259
  360 TEMP(I,J)=TEMP(I,J)+SUM                                              MAIN0260
      CALL MXMULT(SYSP1,TEMP,DELFS,KX,KB,KB,KX,M,N,L1,L1,L1,L1)           MAIN0261
      CALL MXMULT(SYSS,DELFS,SFS,KE,KX,KX,KX,M,N,L2,L1,L1,L1)             MAIN0262
      DO 380 I=1,KE                                                        MAIN0263
      DO 380 J=1,KX1                                                       MAIN0264
      IF(J.GT.KX)GO TO 370                                                 MAIN0265
      SS(I,J)=SFS(I,J)                                                     MAIN0266
      SST(J,I)=SS(I,J)                                                     MAIN0267
      GO TO 380                                                            MAIN0268
  370 SS(I,J)=0.0                                                          MAIN0269
      SST(J,I)=SS(I,J)                                                     MAIN0270
  380 CONTINUE                                                             MAIN0271
      CALL MXMULT(SST,W2,TEMP1,KX1,KE,KE,KE,M,N,L4,L2,L2,L2)              MAIN0272
      CALL MXMULT(TEMP1,SF,TEMP2,KX1,KE,KE,KX1,M,N,L4,L2,L2,L4)           MAIN0273
      DO 390 I=1,KX1                                                       MAIN0274
      DO 390 J=1,KX1                                                       MAIN0275
  390 DELQ(I,J)=BETA*(TEMP2(I,J)+TEMP2(J,I))                               MAIN0276
C                                                                          MAIN0277
C          EVALUATE DELF                                                   MAIN0278
C                                                                          MAIN0279
      DO 410 I=1,KX1                                                       MAIN0280
      DO 410 J=1,KX1                                                       MAIN0281
      IF(I.GT.KX.OR.J.GT.KX)GO TO 400                                      MAIN0282
      DELF(I,J)=DELFS(I,J)                                                 MAIN0283
      DELFT(J,I)=DELF(I,J)                                                 MAIN0284
      GO TO 410                                                            MAIN0285
  400 DELF(I,J)=0.0                                                        MAIN0286
      DELFT(J,I)=0.0                                                       MAIN0287
  410 CONTINUE                                                             MAIN0288
```

```
      CALL MXMULT(DELFT,P,TEMP2,KX1,KX1,KX1,KX1,M,N,L4,L4,L4,L4)          MAIN0289
      DO 420 I=1,KX1                                                      MAIN0290
      DO 420 J=1,KX1                                                      MAIN0291
  420 RHS(I,J)=DELQ(I,J)+TEMP2(I,J)+TEMP2(J,I)                            MAIN0292
C                                                                         MAIN0293
C          RHS MATRIX PARTITION                                           MAIN0294
C                                                                         MAIN0295
      DO 431 I=1,KX                                                       MAIN0296
      DO 431 J=1,KX                                                       MAIN0297
  431 C(I,J)=RHS(I,J)                                                     MAIN0298
      CALL MUSOLV(KX,KX,KX,KX,JB1,JB2)                                    MAIN0299
      DO 432 I=1,KX                                                       MAIN0300
      DO 432 J=1,KX                                                       MAIN0301
  432 DELP(I,J)=XSOLN(I,J)                                                MAIN0302
      DO 433 I=1,KX                                                       MAIN0303
  433 RH12(I,1)=-RHS(I,KX1)                                               MAIN0304
      DO 445 I=1,KX                                                       MAIN0305
      DELP(I,KX1)=0.0D+00                                                 MAIN0306
      DO 444 J=1,KX                                                       MAIN0307
  444 DELP(I,KX1)=DELP(I,KX1)+FINV(I,J)*RH12(J,1)                         MAIN0308
  445 DELP(KX1,I)=DELP(I,KX1)                                             MAIN0309
      DELP(KX1,KX1)=-RHS(KX1,KX1)/(2.0*FMCD(1,1))                         MAIN0310
C                                                                         MAIN0311
C        EVALUATE DELGS MATRIX                                            MAIN0312
C                                                                         MAIN0313
      CALL MXMULT(HK,T1,TEMP3,KB,KB,KB,KC,M,N,L1,L1,L1,L2)                MAIN0314
      CALL MXMULT(SYSP1,TEMP3,DELGS,KX,KB,KB,KC,M,N,L1,L1,L1,L2)          MAIN0315
C                                                                         MAIN0316
C        EVALUATE AUGMENTED DELG MATRIX                                   MAIN0317
C                                                                         MAIN0318
      DO 450 I=1,KX1                                                      MAIN0319
      DO 450 J=1,KC                                                       MAIN0320
      IF(I.GT.KX)GO TO 440                                                MAIN0321
      DELG(I,J)=DELGS(I,J)                                                MAIN0322
      GO TO 450                                                           MAIN0323
  440 DELG(I,J)=0.0D+00                                                   MAIN0324
```

```
  450 CONTINUE                                                          MAIN0325
C                                                                       MAIN0326
C         EVALUATE DELX                                                 MAIN0327
C                                                                       MAIN0328
      DO 460 I=1,KX1                                                    MAIN0329
      DELX(I)=0.0D+00                                                   MAIN0330
      DO 460 J=1,KC                                                     MAIN0331
  460 DELX(I)=DELG(I,J)*VCOMM(J)                                        MAIN0332
C                                                                       MAIN0333
C         EVALUATE DELJ                                                 MAIN0334
C                                                                       MAIN0335
      DELJ=0.0D+00                                                      MAIN0336
      DO 490 I=1,KX1                                                    MAIN0337
      SUM=0.0D+00                                                       MAIN0338
      SUM1=0.0D+00                                                      MAIN0339
      DO 480 J=1,KX1                                                    MAIN0340
      SUM=SUM+DELP(I,J)*XO(J)                                           MAIN0341
  480 SUM1=SUM1+P(I,J)*XO(J)                                            MAIN0342
  490 DELJ=DELJ+SUM*XO(I)+2.0*SUM1*DELX(I)                              MAIN0343
      WRITE(6,511)DELJ                                                  MAIN0344
      PARM(K)=PARAM(K)                                                  MAIN0345
      DELK(K)=-STEP*DELJ                                                MAIN0346
      PARAM(K)=PARAM(K)+DELK(K)                                         MAIN0347
      PAR(K)=PARAM(K)                                                   MAIN0348
      WRITE(6,511)PARAM(K)                                              MAIN0349
  500 DELTAJ=DELTAJ+DELJ*DELK(K)                                        MAIN0350
      WRITE(6,511)DELTAJ                                                MAIN0351
  511 FORMAT(5X,D10.4)                                                  MAIN0352
      ITTER=ITTER+1                                                     MAIN0353
      IF(DABS(DELTAJ).LE.1.0D-05.OR.ITTER.EQ.15)GO TO 510               MAIN0354
      GO TO 5                                                           MAIN0355
  510 ISTOP=1                                                           MAIN0356
      GO TO 5                                                           MAIN0357
  520 WRITE(6,530)                                                      MAIN0358
  530 FORMAT('0',5X,'THE ADJUSTED PARAMETER VALUES ARE')                MAIN0359
      DO 540 K=1,IGAIN                                                  MAIN0360
```

```
  540 WRITE(6,550)K,PAR(K)                                                  MAIN0361
  550 FORMAT(20X,'K(',I1,')=',D12.4)                                        MAIN0362
      WRITE(6,560)                                                          MAIN0363
  560 FORMAT('0',5X,'THE SYSTEM DIFFERENTIAL TRANSITION MATRIX IS')         MAIN0364
      DO 570 I=1,KX                                                         MAIN0365
  570 WRITE(6,580)(SYSF(I,J),J=1,KX)                                        MAIN0366
  580 FORMAT(2X,10D12.4)                                                    MAIN0367
      WRITE(6,590)                                                          MAIN0368
  590 FORMAT('0',5X,'THE SYSTEM CONTROL MATRIX IS')                         MAIN0369
      DO 600 I=1,KX                                                         MAIN0370
  600 WRITE(6,610)(SYSG(I,J),J=1,KC)                                        MAIN0371
  610 FORMAT(10X,10D12.4)                                                   MAIN0372
      WRITE(6,620)ITTER                                                     MAIN0373
  620 FORMAT('0','THE NUMBER OF ITTERATIONS=',I3)                           MAIN0374
      GO TO 1000                                                            MAIN0375
  800 ITTER1=ITTER1+1                                                       MAIN0376
      IF(ITTER1.GE.2)GO TO 520                                              MAIN0377
      DO 810 K=1,IGAIN                                                      MAIN0378
      DELK(K)=DELK(K)/2.0                                                   MAIN0379
  810 PARAM(K)=PARM(K)+DELK(K)                                              MAIN0380
      GO TO 5                                                               MAIN0381
 1000 STOP                                                                  MAIN0382
      END                                                                   MAIN0383
```

```
      SUBROUTINE MSOLVE(MA1,MB1,MC1,NC1,A,MXB,C,JB1,JB2)                MSLV0001
C                                                                       MSLV0002
C        SUBROUTINE LINK SEQUENCE FOR SOLUTION OF MATRIX EQUATION       MSLV0003
C                              AX+XB=-C                                 MSLV0004
C                                                                       MSLV0005
      REAL*8 MXB(20,20),MXV(20,20),MXA0(20,20),MXA1(20,20),MXA2(20,20),  MSLV0006
     1MXVIN(20,20)                                                      MSLV0007
      DIMENSION PB(5,21),QB(5,21),ARRAYA(21),IDIMQB(5),IDIMPB(5),       MSLV0008
     1QINV(5,21),IDIMQI(5),IDENT(5),QIN(21),ARRAYB(21),A(20,20),C(20,20) MSLV0009
     2,DMA(20,20),DMB(20,20),DMC(20,20),DMU(20,20),DMV(20,20),          MSLV0010
     3DVINV(20,20),X(20,20)                                             MSLV0011
      DOUBLE PRECISION ARRAYA,QB,PB,ARRAYB,QIN,QINV,DMA,DMB,DMC,DMU,    MSLV0012
     1DMV,DVINV,CONST,X,A,C                                             MSLV0013
      COMMON /HOLD/DMU,DMV/HOLD1/DMA,DMC/HOLD2/QINV,IDIMQI/             MSLV0014
     1HOLD3/QB,PB,IDIMQB,IDIMPB,IDENT/HOLD4/DVINV,DMB/SECTR1/X          MSLV0015
      DO 50 I=1,MB1                                                     MSLV0016
      DO 50 J=1,MB1                                                     MSLV0017
   50 DMB(I,J)=MXB(I,J)                                                 MSLV0018
      DO 60 I=1,MA1                                                     MSLV0019
      DO 60 J=1,MA1                                                     MSLV0020
   60 DMA(I,J)=A(I,J)                                                   MSLV0021
      DO 70 I=1,MC1                                                     MSLV0022
      DO 70 J=1,NC1                                                     MSLV0023
   70 DMC(I,J)=C(I,J)                                                   MSLV0024
      DO 80 I=1,MA1                                                     MSLV0025
      DO 80 J=1,MA1                                                     MSLV0026
   80 MXA0(I,J)=-A(I,J)                                                 MSLV0027
      CALL SMLTRN(MXB,MXV,MXVIN,MB1,INDXB)                              MSLV0028
      DO 90 I=1,MB1                                                     MSLV0029
      DO 90 J=1,MB1                                                     MSLV0030
      DMV(I,J)=MXV(I,J)                                                 MSLV0031
   90 DVINV(I,J)=MXVIN(I,J)                                             MSLV0032
      CALL SCAN2(MXB,MB1,JB1,JB2)                                       MSLV0033
      CALL SMLTRN(MXA0,MXA1,MXA2,MA1,INDXA)                             MSLV0034
      CALL SCAN1(MXA0,ARRAYA,IDIMA,MA1)                                 MSLV0035
      DO 300 LL1=1,JB1                                                  MSLV0036
```

```
      IDIMB=IDIMQB(LL1)                                                    MSLV0037
      DO 100 LL2=1,IDIMB                                                   MSLV0038
  100 ARRAYB(LL2)=QB(LL1,LL2)                                              MSLV0039
      CALL MPINV(QIN,IDM,ARRAYA,IDIMA,ARRAYB,IDIMB,CONST,IER)              MSLV0040
      IF(DABS(CONST).LE.1.0E-06)GO TO 350                                  MSLV0041
      DO 200 LL3=1,IDM                                                     MSLV0042
  200 QINV(LL1,LL3)=QIN(LL3)                                               MSLV0043
  300 IDIMQI(LL1)=IDM                                                      MSLV0044
      CALL MUSOLV(MB1,MA1,MC1,NC1,JB1,JB2)                                 MSLV0045
      GO TO 400                                                            MSLV0046
  350 WRITE(6,351)                                                         MSLV0047
  351 FORMAT('MATRICES HAVE COMMON EIGENVALUES')                           MSLV0048
  400 RETURN                                                               MSLV0049
      END                                                                  MSLV0050
```

```
      SUBROUTINE SMLTRN(MATRXB,MATRXV,MTRXIN,N,INDEX)                    SMLT0001
C                                                                        SMLT0002
C          DANILEVSKII SIMILARITY TRANSFORMATION                         SMLT0003
C                                                                        SMLT0004
      DOUBLE PRECISION ROW(20),COLUMN(20),AUX(20,20),V1(20,20),VI(20,20) SMLT0005
      REAL*8 MTXSIM(20,20),INVSIM(20,20),MUNIT(20,20),MATRXB(20,20),     SMLT0006
     1MATRXV(20,20),MTRXIN(20,20),MTX(20,20)                             SMLT0007
C                                                                        SMLT0008
C         CONSTRUCT UNIT MATRICES                                        SMLT0009
C                                                                        SMLT0010
      DO 5 K=1,N                                                         SMLT0011
      DO 5 K1=1,N                                                        SMLT0012
      IF(K.EQ.K1)GO TO 3                                                 SMLT0013
      MUNIT(K,K1)=0.                                                     SMLT0014
      MTXSIM(K,K1)=0.                                                    SMLT0015
      INVSIM(K,K1)=0.                                                    SMLT0016
      MATRXV(K,K1)=0.                                                    SMLT0017
      MTRXIN(K,K1)=0.                                                    SMLT0018
      MTX(K,K1)=0.                                                       SMLT0019
      GO TO 5                                                            SMLT0020
    3 MUNIT(K1,K1)=1.                                                    SMLT0021
      MTXSIM(K1,K1)=1.                                                   SMLT0022
      INVSIM(K1,K1)=1.                                                   SMLT0023
      MATRXV(K1,K1)=1.                                                   SMLT0024
      MTRXIN(K1,K1)=1.                                                   SMLT0025
      MTX(K1,K1)=1.0                                                     SMLT0026
    5 CONTINUE                                                           SMLT0027
C                                                                        SMLT0028
C          INDEX J INDICATES COLUMN OF MATRXB                            SMLT0029
C                                                                        SMLT0030
      J=1                                                                SMLT0031
C                                                                        SMLT0032
         INDEX COUNTS NUMBER OF DIAGONAL BLOCKS                          SMLT0033
C                                                                        SMLT0034
      INDEX=1                                                            SMLT0035
C                                                                        SMLT0036
```

```
C       INDEX J2 INDICATES COLUMN OF MTXSIM                                    SMLT0037
   10 J2=J+1                                                                   SMLT0038
C     SEARCH IN COLUMN J OF MATRXB FOR NON ZERO ELEMENT                        SMLT0039
C     IF PIVOT ELEMENT OF MATRXB IS NON ZERO, DISCONTINUE SEARCH               SMLT0040
C     PROCEED TO CONSTRUCT MTXSIM, INVSIM                                      SMLT0041
C     OTHERWISE SEARCH IN REMAINING ROWS FOR FIRST NON ZERO ELEMENT            SMLT0042
C                                                                              SMLT0043
      IF(DABS(MATRXB(J2,J)).GT.1.0D-06)GO TO 1000                              SMLT0044
      MATRXB(J2,J)=0.0D+00                                                     SMLT0045
      J3=J2+1                                                                  SMLT0046
      IF(J3.EQ.(N+1))GO TO 4500                                                SMLT0047
  100 DO 200 I=J3,N                                                            SMLT0048
      IF(DABS(MATRXB(I,J)).GT.1.0D-06)GO TO 300                                SMLT0049
      MATRXB(I,J)=0.0D+00                                                      SMLT0050
  200 CONTINUE                                                                 SMLT0051
      J=J+1                                                                    SMLT0052
      IF(J.EQ.N)GO TO 4500                                                     SMLT0053
      INDEX=INDEX+1                                                            SMLT0054
      GO TO 10                                                                 SMLT0055
C                                                                              SMLT0056
C     INTERCHANGE ROWS I AND J2 AND COLUMNS I AND J2                           SMLT0057
C                                                                              SMLT0058
  300 DO 400 J4=J,N                                                            SMLT0059
      ROW(J4)=MATRXB(J2,J4)                                                    SMLT0060
      MATRXB(J2,J4)=MATRXB(I,J4)                                               SMLT0061
  400 MATRXB(I,J4)=ROW(J4)                                                     SMLT0062
      DO 500 I1=1,N                                                            SMLT0063
      COLUMN(I1)=MATRXB(I1,J2)                                                 SMLT0064
      MATRXB(I1,J2)=MATRXB(I1,I)                                               SMLT0065
  500 MATRXB(I1,I)=COLUMN(I1)                                                  SMLT0066
C                                                                              SMLT0067
C     CONSTRUCT MATRIX MTX                                                     SMLT0068
C                                                                              SMLT0069
      MTX(J2,J2)=0.                                                            SMLT0070
      MTX(I,I)=0.                                                              SMLT0071
      MTX(J2,I)=1.                                                             SMLT0072
```

```
      MTX(I,J2)=1.                                                      SMLT0073
      CALL DMPROD(MATRXV,MTX,V1,N,N,N,N,N1,N1)                          SMLT0074
      CALL DMCPY(V1,MATRXV,N,N,N1,N1)                                   SMLT0075
      CALL DMPROD(MTX,MTRXIN,VI,N,N,N,N,N1,N1)                          SMLT0076
      CALL DMCPY(VI,MTRXIN,N,N,N1,N1)                                   SMLT0077
      CALL DMCPY(MUNIT,MTX,N,N,N1,N1)                                   SMLT0078
C                                                                       SMLT0079
C     CONSTRUCT MTXSIM AND INVSIM                                       SMLT0080
C                                                                       SMLT0081
 1000 PIVOT=MATRXB(J2,J)                                                SMLT0082
      DO 1100 I2=1,N                                                    SMLT0083
      IF(I2.EQ.J2)GO TO 1050                                            SMLT0084
      INVSIM(I2,J2)=-MATRXB(I2,J)/PIVOT                                 SMLT0085
      GO TO 1100                                                        SMLT0086
 1050 INVSIM(I2,I2)=1./PIVOT                                            SMLT0087
 1100 MTXSIM(I2,J2)=MATRXB(I2,J)                                        SMLT0088
C        COMPUTE AUXILIARY MATRIX VECAUX AND RECURSIVE MATRXB           SMLT0089
      CALL DMPROD(INVSIM,MATRXB,AUX,N,N,N,N,N1,N1)                      SMLT0090
C     COMPUTE RECURSIVE MATRXB                                          SMLT0091
      CALL DMPROD(AUX,MTXSIM,MATRXB,N,N,N,N,N1,N1)                      SMLT0092
C     COMPUTE RECURSIVE MATRICES; MATRXV, MTRXIN                        SMLT0093
      CALL DMPROD(MATRXV,MTXSIM,V1,N,N,N,N,N1,N1)                       SMLT0094
      CALL DMCPY(V1,MATRXV,N,N,N1,N1)                                   SMLT0095
      CALL DMPROD(INVSIM,MTRXIN,VI,N,N,N,N,N1,N1)                       SMLT0096
      CALL DMCPY(VI,MTRXIN,N,N,N1,N1)                                   SMLT0097
      J=J+1                                                             SMLT0098
C          CHECK TO DETERMINE NEW BASIS VECTORS HAVE BEEN ESTABLISHED   SMLT0099
      IF(J.EQ.N)GO TO 4500                                              SMLT0100
C      RECONVERT MTXSIM AND INVSIM TO IDENTITY MATRECES                 SMLT0101
      CALL DMCPY(MUNIT,MTXSIM,N,N,N1,N1)                                SMLT0102
      CALL DMCPY(MUNIT,INVSIM,N,N,N1,N1)                                SMLT0103
C      RETURN TO START OF PROCEDURE                                     SMLT0104
      GO TO 10                                                          SMLT0105
 4500 RETURN                                                            SMLT0106
      END                                                               SMLT0107
```

```
      SUBROUTINE SCAN1(MATRXA,ARRAY1,IDIM1,MN)                          SCN10001
C                                                                       SCN10002
C          THIS SUBROUTINE EVALUATES THE CHARACTERISTIC POLYNOMIAL       SCN10003
C          OF THE A MATRIX                                              SCN10004
C                                                                       SCN10005
      DIMENSION ARRAYQ(5,21),IDENT(20),IDIMNQ(5),ARRAY1(21),ARRAY(21),  SCN10006
     1ARRAY2(21)                                                        SCN10007
      REAL*8 MATRXA(20,20)                                              SCN10008
      DOUBLE PRECISION ARRAY1,ARRAY2,ARRAY,ARRAYQ                       SCN10009
C                                                                       SCN10010
C     INITIALIZE ARRAYQ                                                 SCN10011
C                                                                       SCN10012
      DO 10 I=1,5                                                       SCN10013
      DO 10 J=1,21                                                      SCN10014
   10 ARRAYQ(I,J)=0.                                                    SCN10015
      J1=0                                                              SCN10016
C                                                                       SCN10017
C     IDENTIFY NON UNITY(ZERO) SUBDIAGONAL ELEMENTS                     SCN10018
C                                                                       SCN10019
      DO 1000 K1=2,MN                                                   SCN10020
      L1=K1-1                                                           SCN10021
      IF(K1.EQ.MN)GO TO 500                                             SCN10022
      IF(DABS(MATRXA(K1,L1)-1.0).LE.1.0D-04)GO TO 1000                  SCN10023
      J1=J1+1                                                           SCN10024
      IDENT(J1)=K1-1                                                    SCN10025
      GO TO 1000                                                        SCN10026
  500 J1=J1+1                                                           SCN10027
      IDENT(J1)=K1                                                      SCN10028
      IF(DABS(MATRXA(K1,L1)-1.0).GE.1.0D-04)GO TO 750                   SCN10029
      GO TO 1000                                                        SCN10030
  750 J1=J1+1                                                           SCN10031
      IDENT(J1)=K1                                                      SCN10032
      IDENT(J1-1)=K1-1                                                  SCN10033
 1000 CONTINUE                                                          SCN10034
      IDIMNQ(1)=IDENT(1)+1                                              SCN10035
      ARRAYQ(1,IDIMNQ(1))=1.                                            SCN10036
```

```
      ID=IDENT(1)                                                        SCN10037
      DO 2000 I1=1,ID                                                    SCN10038
 2000 ARRAYQ(1,I1)=-MATRXA(I1,ID)                                        SCN10039
      IDIM1=IDIMNQ(1)                                                    SCN10040
      DO 2500 J3=1,IDIM1                                                 SCN10041
 2500 ARRAY1(J3)=ARRAYQ(1,J3)                                            SCN10042
      IF(J1.EQ.1)GO TO 4000                                              SCN10043
      DO 3000 K2=2,J1                                                    SCN10044
      IDIMNQ(K2)=IDENT(K2)-IDENT(K2-1)+1                                 SCN10045
      INDEX=IDIMNQ(K2)-1                                                 SCN10046
      ARRAYQ(K2,IDIMNQ(K2))=1.                                           SCN10047
      DO 3000 K3=1,INDEX                                                 SCN10048
      K4=IDENT(K2-1)+K3                                                  SCN10049
      K5=IDENT(K2)                                                       SCN10050
      ARRAYQ(K2,K3)=-MATRXA(K4,K5)                                       SCN10051
 3000 CONTINUE                                                           SCN10052
      DO 3500 K2=2,J1                                                    SCN10053
      IDIM2=IDIMNQ(K2)                                                   SCN10054
      DO 3400 J3=1,IDIM2                                                 SCN10055
 3400 ARRAY2(J3)=ARRAYQ(K2,J3)                                           SCN10056
      CALL MPMPY(ARRAY,IDIM,ARRAY1,IDIM1,ARRAY2,IDIM2)                   SCN10057
      CALL MPMOVE(ARRAY1,IDIM1,ARRAY,IDIM)                               SCN10058
 3500 CONTINUE                                                           SCN10059
 4000 RETURN                                                             SCN10060
      END                                                                SCN10061
```

```
      SUBROUTINE SCAN2(MATRXB,MN,J1,J2)                                  SCN20001
C                                                                        SCN20002
C         THIS SUBROUTINE IDENTIFIES AND HOLDS THE P & Q ARRAYS          SCN20003
C         FOR SUBSEQUENT  MATRIX POLYNOMIAL INVERSION                    SCN20004
C                                                                        SCN20005
      DIMENSION ARRAYP(5,21),ARRAYQ(5,21),IDIMNP(5),IDIMNQ(5),IDENT(5)   SCN20006
      REAL*8 MATRXB(20,20)                                               SCN20007
      DOUBLE PRECISION ARRAYP,ARRAYQ                                     SCN20008
      COMMON /HOLD3/ARRAYQ,ARRAYP,IDIMNQ,IDIMNP,IDENT                    SCN20009
C                                                                        SCN20010
C     INITIALIZE ARRAYP,ARRAYQ                                           SCN20011
C                                                                        SCN20012
      DO 20 I=1,5                                                        SCN20013
      DO 20 J=1,21                                                       SCN20014
      ARRAYP(I,J)=0.                                                     SCN20015
   20 ARRAYQ(I,J)=0.                                                     SCN20016
C                                                                        SCN20017
C     FOLLOWING SEQUENCE IDENTIFIES AND STORES RELEVANT ARRAYS           SCN20018
C                                                                        SCN20019
      J1=0                                                               SCN20020
      DO 1000 K1=2,MN                                                    SCN20021
      L1=K1-1                                                            SCN20022
      IF(K1.EQ.MN) GO TO 500                                             SCN20023
      IF(DABS(MATRXB(K1,L1)-1.0).LE.1.0D-04)GO TO 1000                   SCN20024
      J1=J1+1                                                            SCN20025
      IDENT(J1)=K1-1                                                     SCN20026
      GO TO 1000                                                         SCN20027
  500 J1=J1+1                                                            SCN20028
      IDENT(J1)=K1                                                       SCN20029
      IF(DABS(MATRXB(K1,L1)-1.0).GE.1.0D-04)GO TO 750                    SCN20030
      GO TO 1000                                                         SCN20031
  750 J1=J1+1                                                            SCN20032
      IDENT(J1)=K1                                                       SCN20033
      IDENT(J1-1)=K1-1                                                   SCN20034
 1000 CONTINUE                                                           SCN20035
      IDIMNQ(1)=IDENT(1)+1                                               SCN20036
```

```
      ARRAYQ(1,IDIMNQ(1))=1.                                          SCN20037
      ID=IDENT(1)                                                     SCN20038
      DO 2000 I1=1,ID                                                 SCN20039
 2000 ARRAYQ(1,I1)=-MATRXB(I1,ID)                                     SCN20040
      IF(J1.EQ.1)GO TO 6000                                           SCN20041
      IDIMNP(1)=IDIMNQ(1)                                             SCN20042
      ID1=IDENT(2)                                                    SCN20043
      DO 3000 K2=2,J1                                                 SCN20044
      IDIMNQ(K2)=IDENT(K2)-IDENT(K2-1)+1                              SCN20045
      INDEX=IDIMNQ(K2)-1                                              SCN20046
      ARRAYQ(K2,IDIMNQ(K2))=1.                                        SCN20047
      DO 3000 K3=1,INDEX                                              SCN20048
      K4=IDENT(K2-1)+K3                                               SCN20049
      K5=IDENT(K2)                                                    SCN20050
 3000 ARRAYQ(K2,K3)=-MATRXB(K4,K5)                                    SCN20051
      DO 4000 I2=1,ID                                                 SCN20052
 4000 ARRAYP(1,I2)=MATRXB(I2,ID1)                                     SCN20053
      J2=J1-1                                                         SCN20054
      IF(J2.EQ.1)GO TO 7000                                           SCN20055
      DO 5000 K6=2,J2                                                 SCN20056
      IDIMNP(K6)=IDIMNP(K6-1)+IDIMNQ(K6)-1                            SCN20057
      INDEX1=IDIMNP(K6)-1                                             SCN20058
      K8=IDENT(K6+1)                                                  SCN20059
      DO 5000 K7=1,INDEX1                                             SCN20060
 5000 ARRAYP(K6,K7)=MATRXB(K7,K8)                                     SCN20061
      GO TO 7000                                                      SCN20062
 6000 J2=0                                                            SCN20063
      IDQ=IDIMNQ(1)                                                   SCN20064
 7000 RETURN                                                          SCN20065
      END                                                             SCN20066
```

```
      SUBROUTINE MPINV(QINV,IDIMQI,ARRAYA,IDIMA,ARRAYB,IDIMB,CONST,IER)          MPNV0001
C                                                                               MPNV0002
C          MATRIX POLYNOMIAL INVERSION SCHEME BASED ON THE                      MPNV0003
C              EUCLIDIAN ALGORITHM                                              MPNV0004
C                                                                               MPNV0005
      DIMENSION ARRAYA(20),ARRAYB(20),ARAYA1(20),ARAYB1(20),AUXIL(20),          MPNV0006
     1ARRAYR(20),ARAYP1(20),ARAYP2(20),ARAYP3(20),QINV(20),Q1(20),              MPNV0007
     2Q3(20),ARAYR1(20),ARAYP4(20),Q2(20),ARAYQ1(20),ARAYQ2(20),                MPNV0008
     3ARAYQ3(20),ARAYQ4(20)                                                     MPNV0009
      DOUBLE PRECISION ARRAYA,ARRAYB,ARAYA1,ARAYB1,ARAYP4,ARRAYR,AUXIL,         MPNV0010
     1ARAYP1,ARAYP2,ARAYP3,ARAYQ1,ARAYQ2,ARAYQ3,QINV,CONST,Q1,Q2,Q3,            MPNV0011
     2ARAYR1,ARAYQ4                                                             MPNV0012
C                                                                               MPNV0013
C          INITIALIZE ALL ARRAY HOLDS                                           MPNV0014
C                                                                               MPNV0015
      DO 10 I=1,20                                                              MPNV0016
      ARAYA1(I)=0.                                                              MPNV0017
      ARAYB1(I)=0.                                                              MPNV0018
      ARRAYR(I)=0.                                                              MPNV0019
      ARAYP1(I)=0.                                                              MPNV0020
      ARAYP2(I)=0.                                                              MPNV0021
   10 ARAYP3(I)=0.                                                              MPNV0022
      IDIMP1=1                                                                  MPNV0023
      ARAYP1(1)=1.                                                              MPNV0024
C                                                                               MPNV0025
C          CHECK DEGREES OF POLYNOMIALS                                         MPNV0026
C                                                                               MPNV0027
      IF(IDIMB.LT.IDIMA)GO TO 110                                               MPNV0028
      CALL MPMOVE(ARAYA1,IDIMA1,ARRAYA,IDIMA)                                   MPNV0029
      CALL MPMOVE(ARAYB1,IDIMB1,ARRAYB,IDIMB)                                   MPNV0030
      CALL MPDIV(Q1,IDIMQ1,ARAYB1,IDIMB1,ARAYA1,IDIMA1,IER)                     MPNV0031
      IF(IER.EQ.1)GO TO 200                                                     MPNV0032
      ICOUNT=1                                                                  MPNV0033
      CALL MPMOVE(ARRAYR,IDIMR,ARAYB1,IDIMB1)                                   MPNV0034
      IF(IDIMB1.EQ.1)GO TO 70                                                   MPNV0035
      ICOUNT=ICOUNT+1                                                           MPNV0036
```

```
      CALL MPDIV(Q2,IDIMQ2,ARAYA1,IDIMA1,ARRAYR,IDIMR,IER)                MPNV0037
      IF(IER.EQ.1)GO TO 200                                               MPNV0038
      IDIMP2=IDIMQ2                                                       MPNV0039
      DO 55 J=1,IDIMP2                                                    MPNV0040
   55 ARAYP2(J)=-Q2(J)                                                    MPNV0041
      IF(IDIMA1.EQ.1)GO TO 80                                             MPNV0042
      CALL MPMOVE(ARAYR1,IDIMR1,ARAYA1,IDIMA1)                            MPNV0043
   60 ICOUNT=ICOUNT+1                                                     MPNV0044
      CALL MPDIV(Q3,IDIMQ3,ARRAYR,IDIMR,ARAYR1,IDIMR1,IER)                MPNV0045
      IF(IER.EQ.1)GO TO 200                                               MPNV0046
      CALL MPMPY(ARAYP4,IDIMP4,Q3,IDIMQ3,ARAYP2,IDIMP2)                   MPNV0047
      CALL MPSUB(ARAYP3,IDIMP3,ARAYP1,IDIMP1,ARAYP4,IDIMP4).              MPNV0048
      IF(IDIMR.EQ.1)GO TO 100                                             MPNV0049
      CALL MPMOVE(ARAYP1,IDIMP1,ARAYP2,IDIMP2)                            MPNV0050
      CALL MPMOVE(ARAYP2,IDIMP2,ARAYP3,IDIMP3)                            MPNV0051
      CALL MPMOVE(AUXIL,IDIMU,ARRAYR,IDIMR)                               MPNV0052
      CALL MPMOVE(ARRAYR,IDIMR,ARAYR1,IDIMR1)                             MPNV0053
      CALL MPMOVE(ARAYR1,IDIMR1,AUXIL,IDIMU)                              MPNV0054
      GO TO 60                                                            MPNV0055
C     REMAINDER POLYNOMIAL IS CONSTANT                                    MPNV0056
   70 CONST=ARAYB1(1)                                                     MPNV0057
      IF(DABS(CONST).LE.1.0E-06)GO TO 200                                 MPNV0058
      QINV(1)=1./CONST                                                    MPNV0059
      IDIMQI=1                                                            MPNV0060
      GO TO 200                                                           MPNV0061
   80 CONST=ARAYA1(1)                                                     MPNV0062
      IF(DABS(CONST).LE.1.0E-06)GO TO 200                                 MPNV0063
      DO 90 I1=1,IDIMP2                                                   MPNV0064
   90 QINV(I1)=ARAYP2(I1)/CONST                                           MPNV0065
      IDIMQI=IDIMP2                                                       MPNV0066
      GO TO 200                                                           MPNV0067
  100 CONST=ARRAYR(1)                                                     MPNV0068
      IF(DABS(CONST).LE.1.0E-06)GO TO 200                                 MPNV0069
      DO 95 I2=1,IDIMP3                                                   MPNV0070
   95 QINV(I2)=ARAYP3(I2)/CONST                                           MPNV0071
      IDIMQI=IDIMP3                                                       MPNV0072
```

```
      GO TO 200                                                          MPNV0073
C     INITIALIZE ARRAYS                                                  MPNV0074
  110 DO 115 I=1,20                                                      MPNV0075
      ARAYA1(I)=0.                                                       MPNV0076
      ARAYB1(I)=0.                                                       MPNV0077
      ARRAYR(I)=0.                                                       MPNV0078
      ARAYQ1(I)=0.                                                       MPNV0079
      ARAYQ2(I)=0.                                                       MPNV0080
  115 ARAYQ3(I)=0.                                                       MPNV0081
      CALL MPMOVE(ARAYA1,IDIMA1,ARRAYB,IDIMB)                            MPNV0082
      CALL MPMOVE(ARAYB1,IDIMB1,ARRAYA,IDIMA)                            MPNV0083
      CALL MPDIV(Q1,IDIMQ1,ARAYB1,IDIMB1,ARAYA1,IDIMA1,IER)              MPNV0084
      IF(IDIMB1.EQ.1)GO TO 120                                           MPNV0085
      DO 116 J=1,IDIMQ1                                                  MPNV0086
  116 ARAYQ1(J)=-Q1(J)                                                   MPNV0087
      IDQ1=IDIMQ1                                                        MPNV0088
      CALL MPDIV(Q2,IDIMQ2,ARAYA1,IDIMA1,ARAYB1,IDIMB1,IER)              MPNV0089
      CALL MPMPY(ARAYQ2,IDQ2,Q2,IDIMQ2,Q1,IDIMQ1)                        MPNV0090
      ARAYQ2(1)=ARAYQ2(1)+1.                                             MPNV0091
      IF(IDIMA1.EQ.1)GO TO 130                                           MPNV0092
      CALL MPMOVE(ARRAYR,IDIMR,ARAYB1,IDIMB1)                            MPNV0093
      CALL MPMOVE(ARAYR1,IDIMR1,ARAYA1,IDIMA1)                           MPNV0094
  117 CALL MPDIV(Q3,IDIMQ3,ARRAYR,IDIMR,ARAYR1,IDIMR1,IER)               MPNV0095
      CALL MPMPY(ARAYQ4,IDQ4,Q3,IDIMQ3,ARAYQ2,IDQ2)                      MPNV0096
      CALL MPSUB(ARAYQ3,IDQ3,ARAYQ1,IDQ1,ARAYQ4,IDQ4)                    MPNV0097
      CALL MPMPY(AUXIL,IDIMU,ARAYQ3,IDQ3,ARRAYB,IDIMB)                   MPNV0098
      IF(IDIMR.EQ.1)GO TO 140                                            MPNV0099
      CALL MPMOVE(ARAYQ1,IDQ1,ARAYQ2,IDQ2)                               MPNV0100
      CALL MPMOVE(ARAYQ2,IDQ2,ARAYQ3,IDQ3)                               MPNV0101
      CALL MPMOVE(AUXIL,IDIMU,ARRAYR,IDIMR)                              MPNV0102
      CALL MPMOVE(ARRAYR,IDIMR,ARAYR1,IDIMR1)                            MPNV0103
      CALL MPMOVE(ARAYR1,IDIMR1,AUXIL,IDIMU)                             MPNV0104
      GO TO 117                                                          MPNV0105
  120 CONST=ARAYB1(1)                                                    MPNV0106
      IF(DABS(CONST).LE.1.0E-06)GO TO 200                                MPNV0107
      IDIMQI=IDIMQ1                                                      MPNV0108
```

```
      DO 125 J=1,IDIMQI                                                 MPNV0109
  125 QINV(J)=-Q1(J)/CONST                                              MPNV0110
      GO TO 200                                                         MPNV0111
  130 CONST=ARAYA1(1)                                                   MPNV0112
      IF(DABS(CONST).LE.1.0E-06)GO TO 200                               MPNV0113
      IDIMQ1=IDQ2                                                       MPNV0114
      DO 135 J=1,IDIMQI                                                 MPNV0115
  135 QINV(J)=ARAYQ2(J)/CONST                                           MPNV0116
      GO TO 200                                                         MPNV0117
  140 CONST=ARRAYR(1)                                                   MPNV0118
      IF(DABS(CONST).LE.1.0E-06)GO TO 200                               MPNV0119
      IDIMQI=IDQ3                                                       MPNV0120
      DO 145 J=1,IDIMQI                                                 MPNV0121
  145 QINV(J)=ARAYQ3(J)/CONST                                           MPNV0122
  200 RETURN                                                            MPNV0123
      END                                                               MPNV0124
```

```
      SUBROUTINE MUSOLV(MB1,MA1,MC1,NC1,JB1,JB2)                        USLV0001
      DIMENSION V1(20,20),V2(20),VSUM(20),V3(20),U(20,20),C(20,20),     USLV0002
     1A(20,20),U1(20),IDIMQI(5),QINV(5,21),V(20,20),QB(5,21),PB(5,21),  USLV0003
     2IDENT(5),IDIMQB(5),IDIMPB(5),E(20,20),VINV(20,20),Y(20,20),       USLV0004
     3B(20,20),AX(20,20),XB(20,20),AXXB(20,20),X(20,20)                 USLV0005
      DOUBLE PRECISION U,V1,V2,V3,VSUM,A,U1,QINV,PB,QB,V,VINV,AX,XB,B,E, USLV0006
     1Y,AXXB,X,C                                                        USLV0007
      COMMON /HOLD/U,V/HOLD1/A,C/HOLD2/QINV,IDIMQI                      USLV0008
     1/HOLD3/QB,PB,IDIMQB,IDIMPB,IDENT/HOLD4/VINV,B/SECTR1/X            USLV0009
      ICOUNT=0                                                          USLV0010
      JWRITE=0                                                          USLV0011
C     V1=C*V                                                            USLV0012
      CALL DMULT(C,V,V1,MC1,NC1,MB1,MB1,MM,NN)                          USLV0013
C     INITIALIZE VSUM                                                   USLV0014
    5 DO 10 II=1,MA1                                                    USLV0015
   10 VSUM(II)=0.                                                       USLV0016
C     FIRST PASS SOLVE FOR  U1,U2,.........UIDENT(1)                    USLV0017
      II1=IDENT(1)                                                      USLV0018
      II5=II1+1                                                         USLV0019
      DO 40 II2=1,II1                                                   USLV0020
      DO 20 II3=1,MA1                                                   USLV0021
   20 V2(II3)=V1(II3,II2)                                               USLV0022
      II4=II2+1                                                         USLV0023
   30 DO 35 II3=1,MA1                                                   USLV0024
   35 VSUM(II3)=VSUM(II3)+QB(1,II4)*V2(II3)                             USLV0025
      II4=II4+1                                                         USLV0026
      IF(II4.GT.II5)GO TO 40                                            USLV0027
C     V2(II3)=(-A)*V2(II3)                                              USLV0028
      CALL AVMULT(A,MA1,V2)                                             USLV0029
      GO TO 30                                                          USLV0030
   40 CONTINUE                                                          USLV0031
C     U1(L1)=(-A)*U(L1,L2-1)                                            USLV0032
      DO 41 L1=1,MA1                                                    USLV0033
   41 U(L1,1)=QINV(1,1)*VSUM(L1)                                        USLV0034
      INDXQI=IDIMQI(1)                                                  USLV0035
      DO 43 L2=2,INDXQI                                                 USLV0036
```

```
      CALL AVMULT(A,MA1,VSUM)                                          USLV0037
      DO 42 L1=1,MA1                                                   USLV0038
   42 U(L1,1)=U(L1,1)+QINV(1,L2)*VSUM(L1)                              USLV0039
   43 CONTINUE                                                         USLV0040
      DO 70 L2=2,II1                                                   USLV0041
      DO 60 L1=1,MA1                                                   USLV0042
   60 U1(L1)=U(L1,L2-1)                                                USLV0043
      CALL AVMULT(A,MA1,U1)                                            USLV0044
      DO 65 L1=1,MA1                                                   USLV0045
   65 U(L1,L2)=U1(L1)-V1(L1,L2-1)                                      USLV0046
   70 CONTINUE                                                         USLV0047
      IF(JB1.EQ.1.AND.ICOUNT.EQ.0)GO TO 300                            USLV0048
      IF(JB1.EQ.1.AND.ICOUNT.GT.0)GO TO 400                            USLV0049
C     SOLVE FOR REMAINING VECTORS OF U                                 USLV0050
      DO 200 JJ=2,JB1                                                  USLV0051
      JJ0=JJ-1                                                         USLV0052
      JJ2=IDENT(JJ)                                                    USLV0053
      JJ1=IDENT(JJ-1)+1                                                USLV0054
C     RE INITIALIZE VSUM                                               USLV0055
      DO 80 JJ3=1,MA1                                                  USLV0056
   80 VSUM(JJ3)=0.                                                     USLV0057
      JJ4=IDENT(JJ-1)                                                  USLV0058
      DO 90 JJ5=1,JJ4                                                  USLV0059
      DO 90 JJ3=1,MA1                                                  USLV0060
      V3(JJ3)=PB(JJ0,JJ5)*U(JJ3,JJ5)                                   USLV0061
   90 VSUM(JJ3)=VSUM(JJ3)+V3(JJ3)                                      USLV0062
      JJ6=0                                                            USLV0063
      DO 130 JJ7=JJ1,JJ2                                               USLV0064
      JJ6=JJ6+1                                                        USLV0065
      DO 100 JJ3=1,MA1                                                 USLV0066
  100 V2(JJ3)=V1(JJ3,JJ7)                                              USLV0067
      JJ8=JJ6+1                                                        USLV0068
  110 DO 120 JJ3=1,MA1                                                 USLV0069
  120 VSUM(JJ3)=VSUM(JJ3)+QB(JJ,JJ8)*V2(JJ3)                           USLV0070
      JJ8=JJ8+1                                                        USLV0071
      IF(JJ8.GT.(JJ2-JJ1+2))GO TO 130                                  USLV0072
```

```
C       V2(JJ3)=(-A)*V2(JJ3)                                              USLV0073
        CALL AVMULT(A,MA1,V2)                                             USLV0074
        GO TO 110                                                         USLV0075
  130 CONTINUE                                                            USLV0076
C       SOLVE FOR U(JJ1)                                                  USLV0077
        DO 140 L1=1,MA1                                                   USLV0078
  140 U(L1,JJ1)=QINV(JJ,1)*VSUM(L1)                                       USLV0079
        INDXQI=IDIMQI(JJ)                                                 USLV0080
        DO 160 L2=2,INDXQI                                                USLV0081
        CALL AVMULT(A,MA1,VSUM)                                           USLV0082
        DO 150 L1=1,MA1                                                   USLV0083
  150 U(L1,JJ1)=U(L1,JJ1)+QINV(JJ,L2)*VSUM(L1)                            USLV0084
  160 CONTINUE                                                            USLV0085
        IF(JJ1.EQ.JJ2)GO TO 200                                           USLV0086
        JJ9=JJ1+1                                                         USLV0087
        DO 190 L2=JJ9,JJ2                                                 USLV0088
        DO 170 L1=1,MA1                                                   USLV0089
  170 U1(L1)=U(L1,L2-1)                                                   USLV0090
        CALL AVMULT(A,MA1,U1)                                             USLV0091
        DO 180 L1=1,MA1                                                   USLV0092
  180 U(L1,L2)=U1(L1)-V1(L1,L2-1)                                         USLV0093
  190 CONTINUE                                                            USLV0094
  200 CONTINUE                                                            USLV0095
        IF(ICOUNT.GT.0)GO TO 400                                          USLV0096
  300 CALL DMULT(U,VINV,X,MA1,MB1,MB1,MB1,MM,NN)                          USLV0097
        CALL DMULT(A,X,AX,MA1,MA1,MA1,MB1,MM,NN)                          USLV0098
        CALL DMULT(X,B,XB,MA1,MB1,MB1,MB1,MM,NN)                          USLV0099
        DO 340 I=1,MA1                                                    USLV0100
        DO 340 J=1,MB1                                                    USLV0101
  340 AXXB(I,J)=AX(I,J)+XB(I,J)                                           USLV0102
        DO 350 I=1,MA1                                                    USLV0103
        DO 350 J=1,MB1                                                    USLV0104
  350 E(I,J)=AXXB(I,J)+C(I,J)                                             USLV0105
        DO 330 I=1,MA1                                                    USLV0106
  330 WRITE(6,600)(E(I,J),J=1,MB1)                                        USLV0107
        ICOUNT=ICOUNT+1                                                   USLV0108
```

```
      CALL DMULT(E,V,V1,MA1,MB1,MB1,MB1,MM,NN)                          USLV0109
      GO TO 5                                                           USLV0110
  400 CALL DMULT(U,VINV,Y,MA1,MB1,MB1,MB1,MM,NN)                        USLV0111
      CALL DMULT(A,Y,AX,MA1,MA1,MA1,MB1,MM,NN)                          USLV0112
      CALL DMULT(Y,B,XB,MA1,MB1,MB1,MB1,MM,NN)                          USLV0113
      DO 450 I=1,MA1                                                    USLV0114
      DO 450 J=1,MB1                                                    USLV0115
      X(I,J)=X(I,J)+Y(I,J)                                              USLV0116
      AXXB(I,J)=AX(I,J)+XB(I,J)                                         USLV0117
  450 E(I,J)=AXXB(I,J)+E(I,J)                                           USLV0118
      IF(ICOUNT.EQ.2)GO TO 500                                          USLV0119
      ICOUNT=ICOUNT+1                                                   USLV0120
      CALL DMULT(E,V,V1,MA1,MB1,MB1,MB1,MM,NN)                          USLV0121
      GO TO 5                                                           USLV0122
  500 JWRITE=JWRITE+1                                                   USLV0123
      IF(JWRITE.EQ.2)GO TO 700                                          USLV0124
      WRITE(6,550)                                                      USLV0125
  550 FORMAT(5X,'THE SOLUTION MATRIX X IS')                             USLV0126
      DO 560 I=1,MA1                                                    USLV0127
  560 WRITE(6,600)(X(I,J),J=1,MB1)                                      USLV0128
      WRITE(6,650)                                                      USLV0129
  650 FORMAT(5X,'THE ERROR MATRIX EMATRIX=AX+XB+C IS')                  USLV0130
      DO 670 I=1,MA1                                                    USLV0131
  670 WRITE(6,600)(E(I,J),J=1,MB1)                                      USLV0132
  600 FORMAT(10D15.5)                                                   USLV0133
  700 RETURN                                                            USLV0134
      END                                                               USLV0135
```

```
      SUBROUTINE AVMULT(AA,MAA,V4)                                      AVMT0001
      DIMENSION AA(20,20),V4(20),V5(20)                                 AVMT0002
      DOUBLE PRECISION AA,V4,V5,SSUM                                    AVMT0003
      DO 50 I=1,MAA                                                     AVMT0004
      SSUM=0.                                                           AVMT0005
      DO 45 J=1,MAA                                                     AVMT0006
   45 SSUM=SSUM+(-AA(I,J))*V4(J)                                        AVMT0007
   50 V5(I)=SSUM                                                        AVMT0008
      DO 55 K=1,MAA                                                     AVMT0009
   55 V4(K)=V5(K)                                                       AVMT0010
      RETURN                                                            AVMT0011
      END                                                               AVMT0012
```

```
      SUBROUTINE RSCHEK(ARRAY,IDIM,ROUTH1,IDIMR1,ROUTH2,IDIMR2,ICODE)          RSCK0001
C                                                                              RSCK0002
C        THIS SUBROUTINE CHECKS FOR SYSTEM ASYMPTOTIC                          RSCK0003
C        STABILITY USING THE ROUTH CRITERION                                   RSCK0004
C                                                                              RSCK0005
C                                                                              RSCK0006
C        CONDITION CODES                                                       RSCK0007
C        1)ICODE=0 INDICATES ASYMPTOTIC STABILITY                              RSCK0008
C        2)ICODE=1 INDICATES ASYMPTOTIC INSTABILITY                            RSCK0009
C                                                                              RSCK0010
      DIMENSION ARRAY(21),ROUTH1(21),ROUTH2(21),Q(21),AUXIL(21)                RSCK0011
      DOUBLE PRECISION ARRAY,ROUTH1,ROUTH2,Q,AUXIL                             RSCK0012
      IR=IDIM                                                                  RSCK0013
    3 IF(ARRAY(IR).LE.0.0)GO TO 40                                             RSCK0014
      IR=IR-1                                                                  RSCK0015
      IF(IR.EQ.0)GO TO 4                                                       RSCK0016
      GO TO 3                                                                  RSCK0017
C                                                                              RSCK0018
C        FORM INITIAL ARRAYS                                                   RSCK0019
C                                                                              RSCK0020
    4 IDIMR1=IDIM                                                              RSCK0021
      IDIMR2=IDIM-1                                                            RSCK0022
      IR1=IDIMR1                                                               RSCK0023
      IR2=IDIMR2                                                               RSCK0024
    5 IR3=IR1-1                                                                RSCK0025
      IR4=IR2-1                                                                RSCK0026
      ROUTH1(IR1)=ARRAY(IR1)                                                   RSCK0027
      IF(IR1.EQ.1)GO TO 10                                                     RSCK0028
      ROUTH1(IR3)=0.                                                           RSCK0029
      ROUTH2(IR2)=ARRAY(IR2)                                                   RSCK0030
      IF(IR2.EQ.1)GO TO 10                                                     RSCK0031
      ROUTH2(IR4)=0.                                                           RSCK0032
      IR1=IR1-2                                                                RSCK0033
      IR2=IR2-2                                                                RSCK0034
      GO TO 5                                                                  RSCK0035
   10 CALL MPDIV(Q,IDIMQ,ROUTH1,IDIMR1,ROUTH2,IDIMR2)                          RSCK0036
```

```
      IF(IDIMR1.EQ.1.AND.ROUTH1(1).GT.0.0)GO TO 50                    RSCK0037
      IF(IDIMR1.EQ.1.AND.ROUTH1(1).LE.0.0)GO TO 40                    RSCK0038
      IR5=IDIMR1                                                      RSCK0039
   20 IF(ROUTH1(IR5).LE.0.0)GO TO 40                                  RSCK0040
      IR5=IR5-2                                                       RSCK0041
      IF(IR5.EQ.0.OR.IR5.EQ.-1)GO TO 30                               RSCK0042
      GO TO 20                                                        RSCK0043
   30 CALL MPMOVE(AUXIL,IDIMU,ROUTH1,IDIMR1)                          RSCK0044
      CALL MPMOVE(ROUTH1,IDIMR1,ROUTH2,IDIMR2)                        RSCK0045
      CALL MPMOVE(ROUTH2,IDIMR2,AUXIL,IDIMU)                          RSCK0046
      GO TO 10                                                        RSCK0047
   40 ICODE=1                                                         RSCK0048
      GO TO 60                                                        RSCK0049
   50 ICODE=0                                                         RSCK0050
   60 RETURN                                                          RSCK0051
      END                                                             RSCK0052
```

```
      SUBROUTINE MPDIV(P,IDIMP,X,IDIMX,Y,IDIMY,IER)                    MPDV0001
      DIMENSION P(1),X(1),Y(1)                                         MPDV0002
      DOUBLE PRECISION P,X,Y,TOL                                       MPDV0003
      TOL=1.0D-06                                                      MPDV0004
      CALL MPNORM(Y,IDIMY,TOL)                                         MPDV0005
      IF(IDIMY)50,50,10                                                MPDV0006
   10 IDIMP=IDIMX-IDIMY+1                                              MPDV0007
      IF(IDIMP)20,30,60                                                MPDV0008
   20 IDIMP=0                                                          MPDV0009
   30 IER=0                                                            MPDV0010
   40 RETURN                                                           MPDV0011
   50 IER=1                                                            MPDV0012
      GO TO 40                                                         MPDV0013
   60 IDIMX=IDIMY-1                                                    MPDV0014
      I=IDIMP                                                          MPDV0015
   70 II=I+IDIMX                                                       MPDV0016
      P(I)=X(II)/Y(IDIMY)                                              MPDV0017
      DO 80 K=1,IDIMX                                                  MPDV0018
      J=K-1+I                                                          MPDV0019
      X(J)=X(J)-P(I)*Y(K)                                              MPDV0020
   80 CONTINUE                                                         MPDV0021
      I=I-1                                                            MPDV0022
      IF(I)90,90,70                                                    MPDV0023
   90 CALL MPNORM(X,IDIMX,TOL)                                         MPDV0024
      GO TO 30                                                         MPDV0025
      END                                                              MPDV0026
```

```
      SUBROUTINE MPNORM(X,IDIMX,EPS)                                    MPNM0001
      DIMENSION X(1)                                                    MPNM0002
      DOUBLE PRECISION X,EPS                                            MPNM0003
    1 IF(IDIMX)4,4,2                                                    MPNM0004
    2 IF(DABS(X(IDIMX))-EPS)3,3,4                                       MPNM0005
    3 IDIMX=IDIMX-1                                                     MPNM0006
      GO TO 1                                                           MPNM0007
    4 RETURN                                                            MPNM0008
      END                                                               MPNM0009
```

```
      SUBROUTINE MPMPY(Z,IDIMZ,X,IDIMX,Y,IDIMY)                    MPMY0001
      DIMENSION Z(1),X(1),Y(1)                                     MPMY0002
      DOUBLE PRECISION Z,X,Y                                       MPMY0003
      IF(IDIMX*IDIMY)10,10,20                                      MPMY0004
   10 IDIMZ=0                                                      MPMY0005
      GO TO 50                                                     MPMY0006
   20 IDIMZ=IDIMX+IDIMY-1                                          MPMY0007
      DO 30 I=1,IDIMZ                                              MPMY0008
   30 Z(I)=0.                                                      MPMY0009
      DO 40 I=1,IDIMX                                              MPMY0010
      DO 40 J=1,IDIMY                                              MPMY0011
      K=I+J-1                                                      MPMY0012
   40 Z(K)=X(I)*Y(J)+Z(K)                                          MPMY0013
   50 RETURN                                                       MPMY0014
      END                                                          MPMY0015
```

```
      SUBROUTINE MPMOVE(Y,IDIMY,X,IDIMX)                                MPMV0001
      DIMENSION X(1),Y(1)                                               MPMV0002
      DOUBLE PRECISION X,Y                                              MPMV0003
      IDIMY=IDIMX                                                       MPMV0004
      IF(IDIMX)30,30,10                                                 MPMV0005
   10 DO 20 I=1,IDIMX                                                   MPMV0006
   20 Y(I)=X(I)                                                         MPMV0007
   30 RETURN                                                            MPMV0008
      END                                                               MPMV0009
```

```
      SUBROUTINE MPSUB(Z,IDIMZ,X,IDIMX,Y,IDIMY)                    MPSB0001
      DIMENSION Z(1),X(1),Y(1)                                     MPSB0002
      DOUBLE PRECISION Z,X,Y                                       MPSB0003
C     TEST DIMENSIONS OF SUMMANDS                                  MPSB0004
      NDIM=IDIMX                                                   MPSB0005
      IF(IDIMX-IDIMY)10,20,20                                      MPSB0006
   10 NDIM=IDIMY                                                   MPSB0007
   20 IF(NDIM)90,90,30                                             MPSB0008
   30 DO 80 I=1,NDIM                                               MPSB0009
      IF(I-IDIMX)40,40,60                                          MPSB0010
   40 IF(I-IDIMY)50,50,70                                          MPSB0011
   50 Z(I)=X(I)-Y(I)                                               MPSB0012
      GO TO 80                                                     MPSB0013
   60 Z(I)=-Y(I)                                                   MPSB0014
      GO TO 80                                                     MPSB0015
   70 Z(I)=X(I)                                                    MPSB0016
   80 CONTINUE                                                     MPSB0017
   90 IDIMZ=NDIM                                                   MPSB0018
      RETURN                                                       MPSB0019
      END                                                          MPSB0020
```

```
      SUBROUTINE MXMULT(A,B,C,MA,NA,MB,NB,MC,NC,I1,J1,I2,J2)          MXMT0001
      DOUBLE PRECISION A(I1,J1),B(I2,J2),C(I1,J2)                     MXMT0002
      MC=MA                                                           MXMT0003
      NC=NB                                                           MXMT0004
      DO 10 I=1,MA                                                    MXMT0005
      DO 10 J=1,NB                                                    MXMT0006
      C(I,J)=0.0D+00                                                  MXMT0007
      DO 10 K=1,MB                                                    MXMT0008
   10 C(I,J)=C(I,J)+A(I,K)*B(K,J)                                     MXMT0009
      RETURN                                                          MXMT0010
      END                                                             MXMT0011
```

REFERENCES

1. Bellman, R: "Introduction to Matrix Analysis", McGraw Hill Book Co., 1970.

2. Blakelock, J.H., "Automatic Control of Aircraft and Missiles", John Wiley and Sons, Inc., 1965.

3. Faddeev, D.K., Faddevva, V.N., "Computational Methods of Linear Algebra", W.H. Freeman and Co., San Francisco, 1963.

4. Grundy, P.A., "The Matrix Equation AX + XB = -C and Its Application to Control System Design". M.I.T. S.M. Thesis, Jan. 1971.

5. Kalman, R.E., Bertram, J.E., "Control System Analysis and Design via the 'Second Method' of Lyapunov, Part 1 Continuous Time Systems", ASME Journal of Pasic Engineering, Vol. 82D, No. 2, pp. 371-393, June 1960.

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

7. Whitaker, H.P., "Design Capabilities of Model Reference Adaptive System", M.I.T. Instrumentation Laboratory Report R-374, July, 1962.