COLLEGE OF ENGINEERING

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

BLACKSBURG, VIRGINIA

VPI-Aero-120

FINAL REPORT

June 15, 1980 - November 15, 1980

NASA Langley Research Center
Research Grant NAG-1-80

Preliminary Demonstration of a Robust Controller Design Method

L. R. Anderson, Assistant Professor
Department of Aerospace and Ocean Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

The NASA Technical Officer for this grant is Jerry R. Newsom, Structures and Dynamics Division, NASA Langley Research Center.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my Graduate Research Assistants, Mr. Rao Vadali and Mr. Alok Das, for their careful and accurate work writing and executing computer programs, and reporting on research results. This work could not have been completed without their capable assistance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

Many mechanical systems can be modeled in state space format as the constant coefficient linear matrix differential equation:

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

where x = n-vector of state variables

u = m-vector of control variables

y = p-vector of output variables.

In a typical mechanical system, one would be able to measure many, but perhaps not all, of the state variables x. An estimate $\hat{x}$ of the complete state might be obtained through a Luenberger observer or Kalman filter. The control system design task then consists of finding an algebraic or dynamic control law $u(x, y, t)$ which yields the control signal based on measurable quantities.

This study is concerned with evaluation of alternative computational procedures for obtaining the feedback control law. It is desired to find computational methods which

1) involve only a small number of free parameters (i.e. two or three) to be specified by the designer so that minimal user interaction or "cut and try" iteration is required, and
2) yield <u>robust</u> control, i.e. the controller is insensitive to small changes in the A and B matrices and performs satisfactorily when the mechanical system is operating away from the nominal design point.

The methods evaluated in this study assume that the full state is measurable, and find a constant m x n feedback matrix K with

$$u = Kx. \tag{3}$$

The three methods evaluated are:

1) the standard linear quadratic regulator (LQR) design method[1], where one minimizes the performance index

$$J = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u)dt$$

with Q positive semi-definite and R positive definite matrices of appropriate dimension.

2) minimization of the norm of the feedback matrix, $||K||$ via nonlinear programming subject to the constraint that the closed-loop eigenvalues be in a specified domain in the complex plane.

3) maximize the angles between the closed-loop eigenvectors (or, equivalently, make corresponding left and right eigenvectors as nearly colinear as possible) in combination with minimizing $||K||$ also via nonlinear programming subject to the closed-loop eigenvalue constraint in 2.

These three methods are called the LQR, min K and robust controller design methods, respectively.

The specific function minimization technique used for design methods 2) and 3) is a modification of Powell's conjugate gradient method requiring no gradient information.[2] Admittedly, this is not the current state-of-the-art in nonlinear programming, but the method is simple and reliable, and adequate for this preliminary study.

The domain of the complex plane chosen for closed-loop eigenvalue placement in methods 2 and 3 is illustrated in Figure 1. The domain is bounded on the right by the maximum real part of eigenvalues, REMAX, and on the left by the minimum real part of eigenvalues, REMIN. The top and bottom boundaries are specified by the maximum ratio of imaginary and real parts of complex

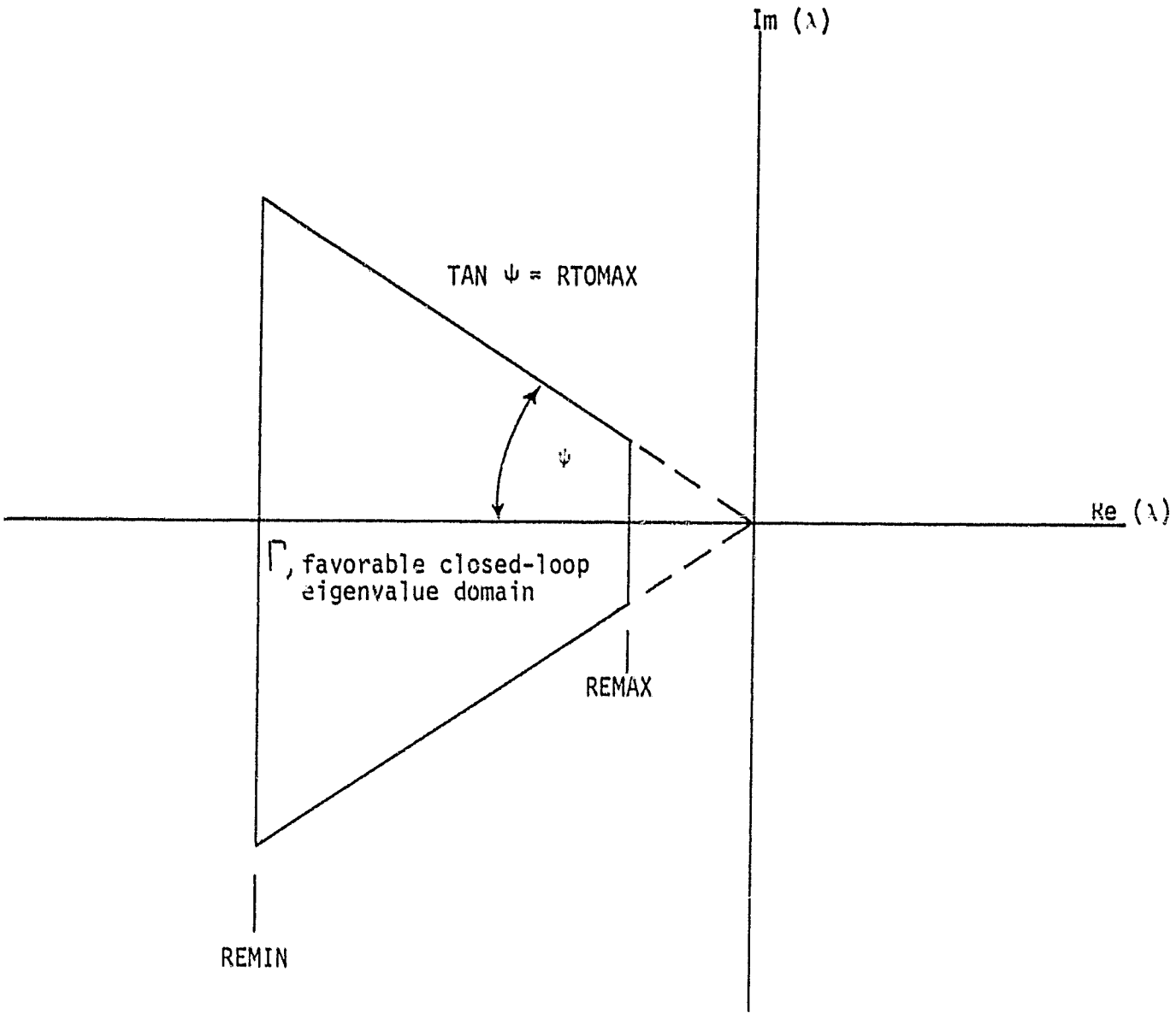


Figure 1 The domain in complex plane specified by parameters REMIN, REMAX and RTOMAX.

eigenvalues, RTOMAX. The closed-loop eigenvalues are those of the matrix

$$\tilde{A} = A + BK \tag{4}$$

specified as

$$\lambda(\tilde{A}) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

Thus the eigenvalue domain can be specified mathematically as

$$\text{REMIN} \leq \text{Re}\ (\lambda_i) \leq \text{REMAX} \tag{5}$$

$$\left|\frac{\text{Im}\ (\lambda_i)}{\text{Re}\ (\lambda_i)}\right| \leq \text{RTOMAX} \tag{6}$$

for all $i = 1, 2, \ldots, n$.

This choice of eigenvalue domain in the complex plane is based on the spectral structure of linear systems[3,4]. That is, given any arbitrary control function u(t), the time response of system (1) is

$$x(t) = e^{\tilde{A}t}\, x(o) + \int_o^t e^{\tilde{A}(t-\tau)}\, Bu(\tau)d\tau. \tag{7}$$

The matrix exponential can be decomposed as

$$e^{\tilde{A}t} = \sum_{i=1}^{n} e^{\lambda_i t}\, v_i w_i^H$$

where v_i and w_i are the right- and left-eigenvectors of the $\tilde{A}$ matrix, i.e.

$$\tilde{A}\, v_i = \lambda_i v_i, \quad w_i^H \tilde{A} = \lambda_i w_i^H. \tag{8}$$

The eigenvalue problem (8) can be written in matrix form as

$$\tilde{A} = MJQ \tag{9}$$

where

$$J = \text{diag}\ (\lambda_1, \lambda_2, \ldots, \lambda_n)$$

$$M = [v_1 \quad v_2 \quad \ldots \quad v_n]$$

$$Q = M^{-1} = \begin{bmatrix} w_1^H \\ w_2^H \\ \vdots \\ w_n^H \end{bmatrix}.$$

In general the eigenvalues will be distinct and stable for the closed-loop system.

Assume that r of the eigenvalues are real and stable, i.e.

$$\lambda_i = -a_i < 0, \; i = 1, 2, \ldots, r$$

and that 2c eigenvalues are complex and stable, so that there are c complex conjugate pairs

$$\lambda_{r+j} = -\alpha_j + i\beta_j, \; -\alpha_j < 0$$

$$\lambda_{r+c+j} = \bar{\lambda}_{r+j} = -\alpha_j - i\beta_j, \; j = 1, 2, \ldots, c.$$

Then the matrix exponential can be written as

$$e^{\tilde{A}t} = \sum_{i=1}^{r} e^{-a_i t} E_i + \sum_{j=1}^{c} e^{-\alpha_j t}(\cos(\beta_j t)F_j + \sin(\beta_j t)G_j) \qquad (10)$$

where E_i, F_i and G_i are the appropriate real projection matrices formed from the dyad product of right- and left-eigenvectors.

As seen from (7) and (10), the rate of decay of the response x(t) of the closed-loop system is characterized by the time constant

$$\tau = \max\left\{\frac{1}{a_1}, \ldots, \frac{1}{a_r}, \frac{1}{\alpha_1}, \ldots \frac{1}{\alpha_c}\right\} . \qquad (11)$$

A larger time constant τ means slower system response, and τ is kept from becoming large by the right-hand eigenvalue boundary REMAX, i.e.

$$\text{Re}\,(\lambda_i) \leq \text{REMAX implies } \tau \leq \frac{-1}{\text{REMAX}} . \qquad (12)$$

Note that REMAX will always be negative for a stable design. The choice of REMAX is governed by (12) to achieve a desired or specified closed-loop time constant τ.

The effect of the eigenvalue ratio

$$\left|\frac{\text{Im}\,(\lambda_i)}{\text{Re}\,(\lambda_i)}\right| \leq \text{RTOMAX}$$

on transient response is well known for the standard damped harmonic oscillator equation

$$\ddot{x} + (2\zeta\omega_n)\dot{x} + \omega_n^2 x = 0.$$

That is, as shown in Figure 1, RTOMAX and ζ are related by the angle

$$\psi = \text{TAN}^{-1}(\text{RTOMAX}) \quad \cos^{-1}(\zeta).$$

In order to have well damped non-oscillatory motion in each mode corresponding to a complex closed-loop eigenvalue, one can choose $\zeta \geq .71$ or $\psi \leq 45°$ which implies RTOMAX ≤ 1.

The left-hand boundary REMIN of the eigenvalue domain is added only to form a closed domain. In general, sending closed-loop eigenvalues far to the left in the complex plane requires large entries in the feedback matrix K, which is prevented by minimizing $||K||$. However, there is no penalty in terms of system stability or transient response if closed-loop eigenvalues have large negative real part.

As stated above, the third or "robust" design method was chosen to yield a closed-loop system whose eigenvalues are insensitive to small changes in the A and B matrices. The relationship between orthogonality of closed-loop eigenvectors and the sensitivity of closed-loop eigenvalues is described in the next chapter.

II. Eigenvalue Robustness Through Eigenvector Orthogonality

As previously described (9), the closed-loop system is assumed to have distinct eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ where

$$A + BK = \tilde{A} = MJM^{-1}$$

$$J = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$$

and corresponding (right) eigenvectors

$$M = [v_1 v_2 \ldots v_n].$$

We assume that the closed-loop system matrix $\tilde{A}$ is perturbed as

$$\tilde{A} + E \tag{12}$$

due to small changes in A and B, and assess the effects of the perturbations E on the eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ using first-order perturbation theory.[5]

Let λ and v represent a particular eigenvalue/eigenvector pair (possibly complex) with the eigenvector normalized so that

$$||v|| = (v^H v)^{1/2} = 1$$

We will proceed to find approximations to an eigenvalue λ' and eigenvector v' of the perturbed system

$$(\tilde{A} + E)\, v' = \lambda' v' \tag{13}$$

that are near λ and v. Since E is small, i.e.

$$||E|| = O(\varepsilon),\ 0 < \varepsilon << 1$$

the differences $\lambda' - \lambda$ and $v' - v$ will also be small, i.e.

$$\lambda' - \lambda = \mu,\ |\mu| = O(\varepsilon) << 1 \tag{14}$$

$$v' - v = q,\ ||q|| = O(\varepsilon) << 1. \tag{15}$$

If v' is also normalized as $v'^H v' = 1$, then q will be orthogonal to v, i.e.

$$q^H v = 0.$$

In order to obtain expressions for μ and q, let U be any n x (n-1) dimensional matrix such that [vU] is n x n and unitary, i.e.

$$[vU]^H\, [vU] = I,$$

so v is orthogonal to each column of U. The perturbation vector q can then be written as a linear combination of the columns of U, i.e.

$$q = Up.$$

The final results

$$|\lambda' - \lambda| \leq ||E||\ ||w^H|| + ||E||^2\ ||U(\lambda I - U^H\tilde{A}U)^{-1}U^H||$$

$$|\lambda' - \lambda| \leq \varepsilon||w^H|| + \varepsilon^2\ ||(\lambda I - U^H\tilde{A}U)^{-1}|| \qquad (16)$$

and

$$v' - v \cong (\lambda I - U^H\tilde{A}U)^{-1}\ U^H E\ v$$

$$|v' - v| \leq \varepsilon||(\lambda I - U^H\tilde{A}U)^{-1}|| \qquad (17)$$

are derived in Appendix E . As equation (16) indicates, for small perturbations E the <u>length</u> of the left eigenvector w_i corresponding to eigenvalue λ_i provides a measure of the sensitivity of λ_i to variations in A + BK. Since the left eigenvectors provide a reciprocal basis[3] for the right eigenvectors, it follows that

$$||w_i^H v_i|| = 1.$$

The angle θ_i between vectors w_i and v_i measured by

$$\theta_i = \cos^{-1}\ \frac{w_i^H v_i}{(w_i^H w_i)^{1/2}\ (v_i^H v_i)^{1/2}} \qquad (18)$$

also provides a measure of the length of w_i and the sensitivity of λ_i to perturbations E. Small angles θ_i will indicate that w_i is small, and therefore the eigenvalue λ_i is "robust".

The second result, equation (17), indicates that the sensitivity of eigenvector v_i is proportional to the distance between eigenvalue λ_i and the rest of the eigenvalue spectrum of $\tilde{A}$. Note that $U^H\tilde{A}U$ has the same eigenvalue as $\tilde{A}$ less λ_i, i.e.

$$\lambda\{U^H\tilde{A}U\} = \lambda\{\tilde{A}\} - \lambda_i.$$

In order for eigenvector v_i to be insensitive to variations in $\tilde{A}$, we want

the eigenvalue spectrum $\lambda\{\tilde{A}\}$ to be well separated in the complex plane, i.e. no two eigenvalues λ_i, λ_j should be closely spaced, or $|\lambda_i - \lambda_j|$ should be maximized for all $i \neq j$.

III. Design Problems

The system matrices used for this design study are fourth order with two and three control variables, and represent linearized lateral plane aircraft dynamics. The state variables are

$$x = \begin{bmatrix} \text{roll rate (rad/sec)} \\ \text{roll angle (rad)} \\ \text{yaw rate (rad/sec)} \\ \text{sideslip angle (rad)} \end{bmatrix}$$

$$u = \begin{bmatrix} \text{aileron (rad)} \\ \text{rudder (rad)} \\ \text{Yaw control* (rad)} \end{bmatrix}$$

*for Hall a/c only.

The first example is taken from Montgomery and Hatch[8] (1969) and represents the lateral dynamics of an early version of the Space Shuttle. The system and control matrices are:

$$A = \begin{bmatrix} -0.367984 & 0.0 & -0.032279 & 26.18750 \\ 1.0 & 0.0 & 0.267949 & 0.0 \\ -0.024209 & 0.0 & -0.110395 & 4.46294 \\ -0.258819 & 0.017835 & -0.965926 & -0.091072 \end{bmatrix}$$

$$B = \begin{bmatrix} -7.67183 & 2.06549 \\ 0.0 & 0.0 \\ 1.96959 & -2.33843 \\ 0.0 & 0.0 \end{bmatrix}$$

The second example models the lateral dynamics of a T-33 trainer and is described in Hall[9] (1971). The yaw control is achieved through asymmetric deflection of drag petals mounted on wing tip tanks.

$$A = \begin{bmatrix} -3.18 & 0.0 & 0.63 & -10.6 \\ 1.0 & 0.0 & 0.0 & 0.0 \\ -0.06 & 0.0 & -0.27 & 4.18 \\ 0.022 & 0.0644 & -0.988 & -0.151 \end{bmatrix}$$

$$B = \begin{bmatrix} -14.4 & 1.5 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & -2.59 & -0.96 \\ 0.0 & 0.037 & 0.0 \end{bmatrix}$$

IV. Programs NEWSOM and LINEAR

As described in Chapter I, a matrix K is computed to feedback the full state to the control variables as

$$u = Kx. \tag{19}$$

This chapter describes the program NEWSOM and the algorithm used for the minimum K and robust controller design methods.

The basic purpose of the program NEWSOM (listed in Appendix A) is to minimize an unconstrained function of several variables with inequality constraints imposed as penalty terms added to the cost function. The multivariable function is minimized using the Powell conjugate gradient nonlinear programming method (Zangwill, 1967) implemented in subroutine POWELL. A key part of the nonlinear program is the line search algorithm, performed by subroutine MINPT. The line search is performed with a combination of outward stepping with doubling of successive step sizes, inward stepping using the golden section search routine, and parabolic curve fit.

The independent variables over which POWELL searches are the nm elements of the K matrix. The cost function is defined as the sum of five scalar terms f_i each with a weighting factor WT.

$$f(x) = \sum_{i=1}^{5} WT_i f_i(x) \tag{20}$$

The scalar functions are defined as

$$f_1(x) = \left[\sum_{i=1}^{nm} x_i^2\right]^{1/2} = ||K|| \tag{21}$$

$$f_2(x) = \sum_{i=1}^{n} \max(0,\ \mathrm{REMIN} - \mathrm{Re}(\lambda_i))^2$$

$$f_3(x) = \sum_{i=1}^{n} \max(0,\ \mathrm{Re}(\lambda_i) - \mathrm{REMAX})^2$$

$$f_4(x) = \sum_{i=1}^{n} \max(0,\ \mathrm{abs}(\mathrm{Im}(\lambda_i)/\mathrm{Re}(\lambda_i)) - \mathrm{RTOMAX})^2$$

where $\{\lambda_i\}$ are the eigenvalues of the closed loop system $\bar{A} = A + BK$. The three terms $f_2(x)$, $f_3(x)$, $f_4(x)$ constrain the closed-loop eigenvalues to remain in the domain Γ of the complex plane illustrated in Figure 1. The fifth scalar function provides a measure of eigenvector orthogonality. Note that the eigenvector angles ϕ_{ij} defined here differ from the angle θ_i introduced in Chapter II. As ϕ_{ij} goes to 90° for all $i \neq j$, θ_i goes to zero.

$$\phi_{ij} = \cos^{-1} \frac{v_i^H v_j}{(v_i^H v_i)^{1/2} (v_j^H v_j)^{1/2}}$$

$$f_5(x) = \left[\sum_{i=2}^{n} \sum_{j=1}^{i-1} (\phi_{ij} - 90°)^{10} \right]^{1/10}$$

The power of ten on each term and the tenth root on the summation are employed to achieve equal penalization of small angles ϕ_{ij}.

The flowchart of program NEWSOM is presented in Figure 2. As shown, the program contains integer flags to control the execution of multiple cases (with a separate namelist block for each case) and the amount of printed output. The program input variables are defined in comment cards at the beginning of the program.

The key program inputs are the initial values of the independent variables (the elements of K) and the weighting terms wT1, ..., wT5. To generate a minimum K control law, the program is run with wT5 = 0 and all other weights chosen to place closed-loop eigenvalues in or near the region Γ. To generate a robust control law, the program is run with primary weight on the term $f_5(x)$ and very little or no weight on the $f_1(x)$ term.

The second program, LINEAR, was used to compute the LQR controller gains for both aircraft. The complete program listing is presented in Appendix B. This program was not written specifically for this research project, but was

Figure 2 Flowchart of program NEWSOM

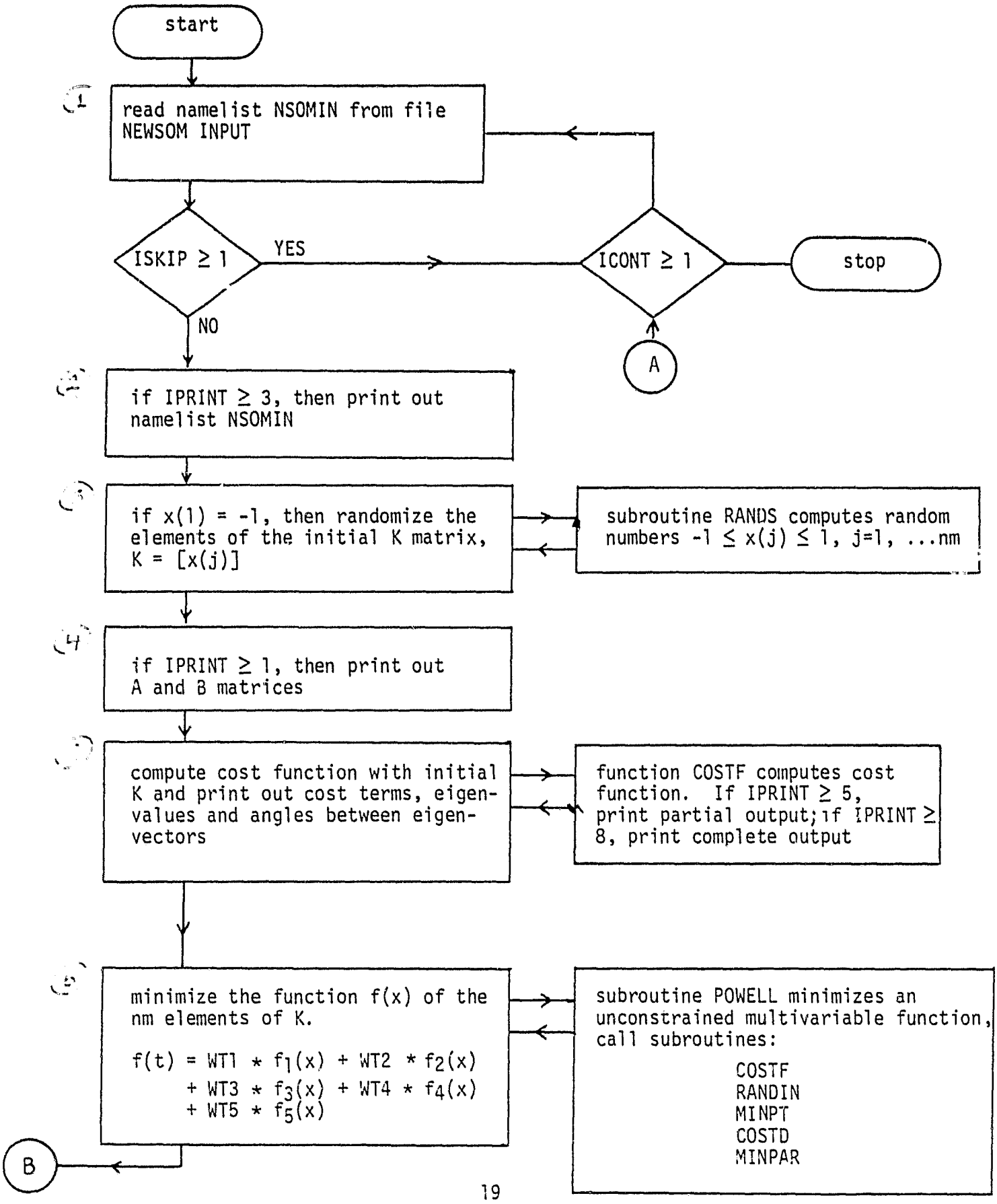
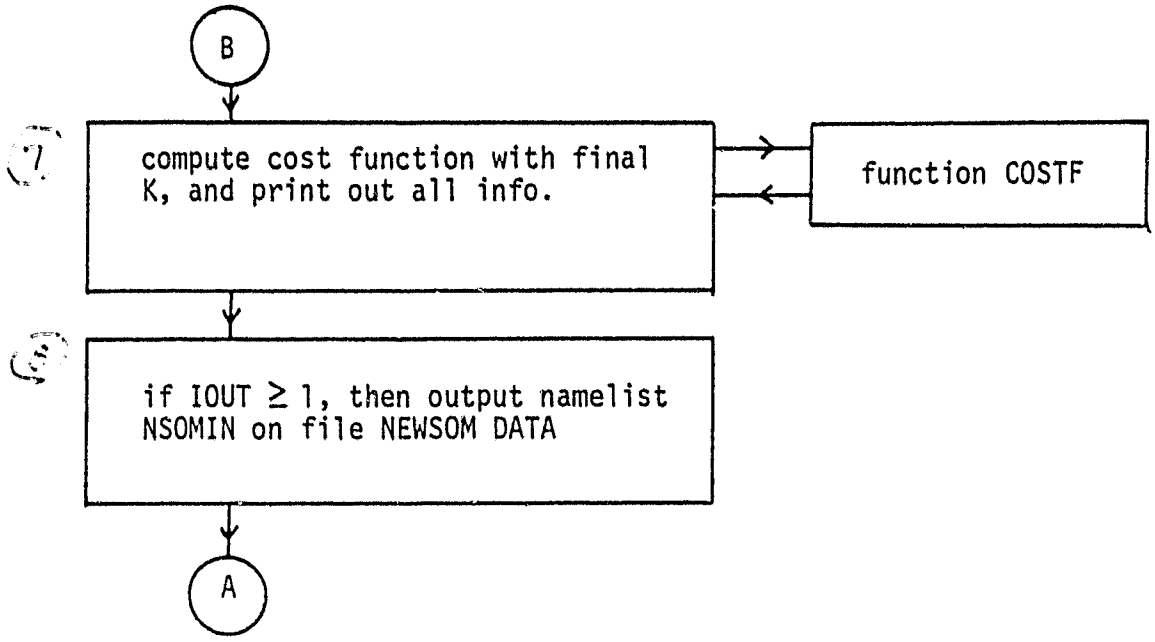


Figure 2 (cont.) Flowchart of program NEWSOM



previously developed by an AOE department graduate student, Mr. Mark Hreha. Flowcharts, input variable definitions and descriptions of the algorithms were not available for LINEAR, but the code is based on a report by Sandell and Athans (1974).

V. Results of programs NEWSOM and LINEAR

For each of the two design problems, i.e. for the Hall and Montgomery aircraft models listed in Chapter III, three minimum K controllers, three robust controllers and three LQR controllers were computed. The three different controllers for the minimum K and robust cases we obtained by specifying different values of the time constant τ (or equivalenlty REMAX $= \frac{-1}{\tau}$), namely $\tau = 1$, .5 and .25 sec. Table 1 presents the input data for these cases. The values of the weights wT1, ..., wT5 were chosen to provide a good tradeoff between the placement of closed-loop eigenvalues in the Γ domain (controlled by wT2, wT3 and wT4), and minimization of $||K||$ (controlled by wT1) or maximization of robustness (controlled wT5). The LQR controllers were generated with the performance index weighting matrices Q and R, also listed in Table 1. The set of three LQR controllers for each aircraft were obtained by varying the control weighting matrix in the performance index as

$$R = \rho I_m$$

with ρ = 100., 1., and .04.

The resulting output K matrices from programs NEWSOM and LINEAR for the Montgomery and Hall aircraft are listed in Tables 2 and 3, respectively. The corresponding closed-loop eigenvalues and angles ϕ_{ij} between eigenvectors are presented in Tables 4 and 5 following Chapter VII.

Once the eighteen feedback cases were generated, they were evaluated by comparing trajectory time histories, and by comparing the sensitivity of closed-loop eigenvalues to perturbations in the A and B matrices. These evaluations are presented in the next four chapters.

Table 1 Input data for programs NEWSOM and LINEAR

Program NEWSOM

aircraft	controller	ϵ	REMIN	REMAX	RTOMAX	WT1	WT2	WT3	WT4	WT5
Montgomery	minimum K	1.0	-10	-1	0.5	1	100	100	100	0
"	"	0.5	"	-2	"	"	"	"	"	"
"	"	0.25	"	-4	"	"	"	"	"	"
"	robust	1.0	-20	-1	"	.01	"	500	"	20
"	"	0.5	"	-2	"	"	"	"	"	"
"	"	0.25	"	-4	"	"	"	"	"	"
Hall	minimum K	1.0	-10	-1	"	1	"	"	500	0
"	"	0.5	"	-2	"	"	"	"	"	"
"	"	0.25	"	-4	"	"	"	"	"	"
"	robust	1.0	-20	-1	"	.01	"	1000	200	10
"	"	0.5	"	-2	"	"	"	500	100	"
"	"	0.3	"	-4	"	"	"	"	"	20

Program LINEAR

aircraft	q_{11}	q_{22}	q_{33}	q_{44}	r_{11}	r_{22}	r_{33}
Montgomery	0.1	1.0	0.1	1.0	100	100	
"	"	"	"	"	1	1	
"	"	"	"	"	.04	.04	
Hall	"	"	"	"	100	100	100
"	"	"	"	"	1	1	1
"	"	"	"	"	.04	.04	.04

Table 2 Gain Matrices for Montgomery Aircraft

MIN K	K MATRIX			
τ = 1.0	1.4941162E-1	1.0535228E-1	1.6827383E0	-2.3151433E-1
	-4.1928029E-1	2.0296771E-2	3.6583920E0	-4.7954880E-2
τ = 0.5	1.7022794E-1	2.5196051E-1	2.3499689E0	-1.1856604E0
	-6.0164034E-1	-2.7018290E-2	4.0587797E0	4.8925018E-1
τ = 0.25	1.2836323E0	8.5040188E-1	1.3259439E0	3.2358761E0
	-3.1967741E-1	-1.5637993E1	4.4778833E0	-1.5113544E0

ROBUST	K MATRIX			
τ = 1.0	6.5132E-1	8.2589E-2	1.60285E0	-7.393E-1
	-3.85575E-1	-1.92548E-1	3.46768E0	-6.0737E-1
τ = 0.5	6.4359E-1	6.764E-1	2.07949E0	-7.41E-1
	-3.8107E-1	-2.44E-1	3.5064E0	-4.32244E-1
τ = 0.25	1.0368E0	1.27027E0	2.05683E0	-7.43469E-1
	-3.599E-1	-3.22E-1	3.81589E0	-5.2096E-1

LQR	K MATRIX			
ρ = 100	3.306E-1	1.0918E-1	-6.9998E-1	2.6222E0
	-1.784E-1	-3.913E-2	4.8287E-1	-1.4068E0
ρ = 1	1.2588E0	1.0046E0	-3.068E-1	5.9393E0
	-1.7087E-1	9.53E-2	1.0189E0	-1.734E0
ρ = .04	5.17E0	4.8719E0	1.06E-1	7.2623E0
	-5.9375E-2	1.1309E0	5.0654E0	-2.5309E0

Table 3 Gain Matrices for Hall Aircraft

MIN K	K MATRIX			
	-3.97326E-2	1.20207E-2	2.16118E-1	-9.48512E-1
τ= 1.0	-4.35985E-3	-9.44925E-1	-3.66113E-3	-5.47779E-1
	1.99877E-2	2.55934E0	4.75163E0	-5.14107E-1
	2.46900E-2	2.40550E-1	-3.01650E-1	-1.46550E-1
τ= 0.5	-1.39400E-3	3.84000E-2	2.29900E-2	1.14765E-2
	-1.16600E-2	-4.38000E-2	4.14500E0	-3.11300E-1
	3.56954E-1	1.45003E0	-1.29224E0	-4.19082E0
τ= 0.25	-3.94971E-2	2.40516E-1	2.69001E0	-1.24536E1
	-1.27874E-2	-8.76188E-1	3.61001E0	1.41500E1

ROBUST	K MATRIX			
	-3.48900E-2	1.37833E-1	2.17500E-1	-9.67330E-1
τ = 1.0	1.51900E-1	-9.33000E-1	2.64400E-3	-6.96500E-1
	-2.63300E0	2.53500E0	5.53060E0	-5.25889E-1
	3.81900E-2	4.13300E-1	-2.75920E-1	-1.22950E-1
τ = 0.5	-1.62450E-3	3.86788E-2	2.53255E-2	7.23800E-3
	-1.25480E-2	-4.43500E-2	4.15544E0	-3.05500E-1
	4.99740E-1	1.46229E0	1.12387E-1	-1.04167E0
τ = 0.25	3.09131E-1	-9.32129E-1	-1.13726E-3	-2.05667E0
	-3.34400E0	-5.79063E0	5.97918E0	-1.36633E0

LQR	K MATRIX			
	4.51300E-2	9.16000E-2	5.69790E-2	-7.54000E-2
ρ = 100	5.74000E-3	2.22000E-2	9.24200E-2	-2.08000E-2
	3.79800E-3	1.16790E-2	3.63300E-2	-8.92500E-3
	8.65300E-1	9.95860E-1	1.17500E-1	-5.56190E-1
ρ = 1	-6.75770E-2	-5.85950E-2	9.39890E-1	-2.79920E-1
	7.83300E-3	1.62460E-2	3.51480E-1	-1.00000E-1
	4.83410E0	4.98000E0	4.18090E-1	-7.36140E-1
ρ = 0.04	-4.26400E-1	-5.01500E-1	4.74800E0	-3.15300E0
	2.78700E-2	1.23700E-2	1.76390E0	-8.60260E-1

VI Program INTODE

This FORTRAN program integrates ordinary differential equations and is used to give the time trajectory of multivariable linear systems. It also provides information on the eigenvalue stability of the linear system. The equations describing the system dynamics and the feedback law should be in the form

$$\dot{x} = Ax + Bu$$

$$u = Kx$$

where x is the n x 1 state vector

u is the m x 1 control vector

A is the n x n system matrix

B is the n x m control matrix

K is the m x n feedback matrix.

The program integrates the differential equations using the Runge-Kutta 4th order method. In this method for a given differential equation

$$\frac{dy}{dx} = f(x,y) ,$$

we have $y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

where $k_1 = hf(x_i, y_i)$

$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = hf(x_i + h, y_i + k_z)$$

and h is the step size.

The results of the program INTODE are available as a trajectory table and two sets of plots, one using the printer and the other using the versatec plotter. The program also computes the eigenvalues, eigenvectors and the angles between the eigenvectors for the closed loop system.

Figure 3 is a basic flowchart of the program INTODE. Input data to the program is given in the form of a namelist. A number of integer variables have been included in the program to provide flexibility in program execution. Multiple cases can be run by appropriately setting the index ICONT and including multiple cases in the namelist. The amount of printout is controlled by the index IPRINT while the index IPLOT controls the generation of plots. A complete listing of the program is given in Appendix C. A list of variables forming the namelist and their notations is included in the program listing.

The program was used to generate time trajectories for the lateral dynamics of the Hall and Montgomery aircrafts. The state and control variables in the model used are

- p - roll rate
- ϕ - roll angle
- r - yaw rate
- β - sideslip angle

} state variables

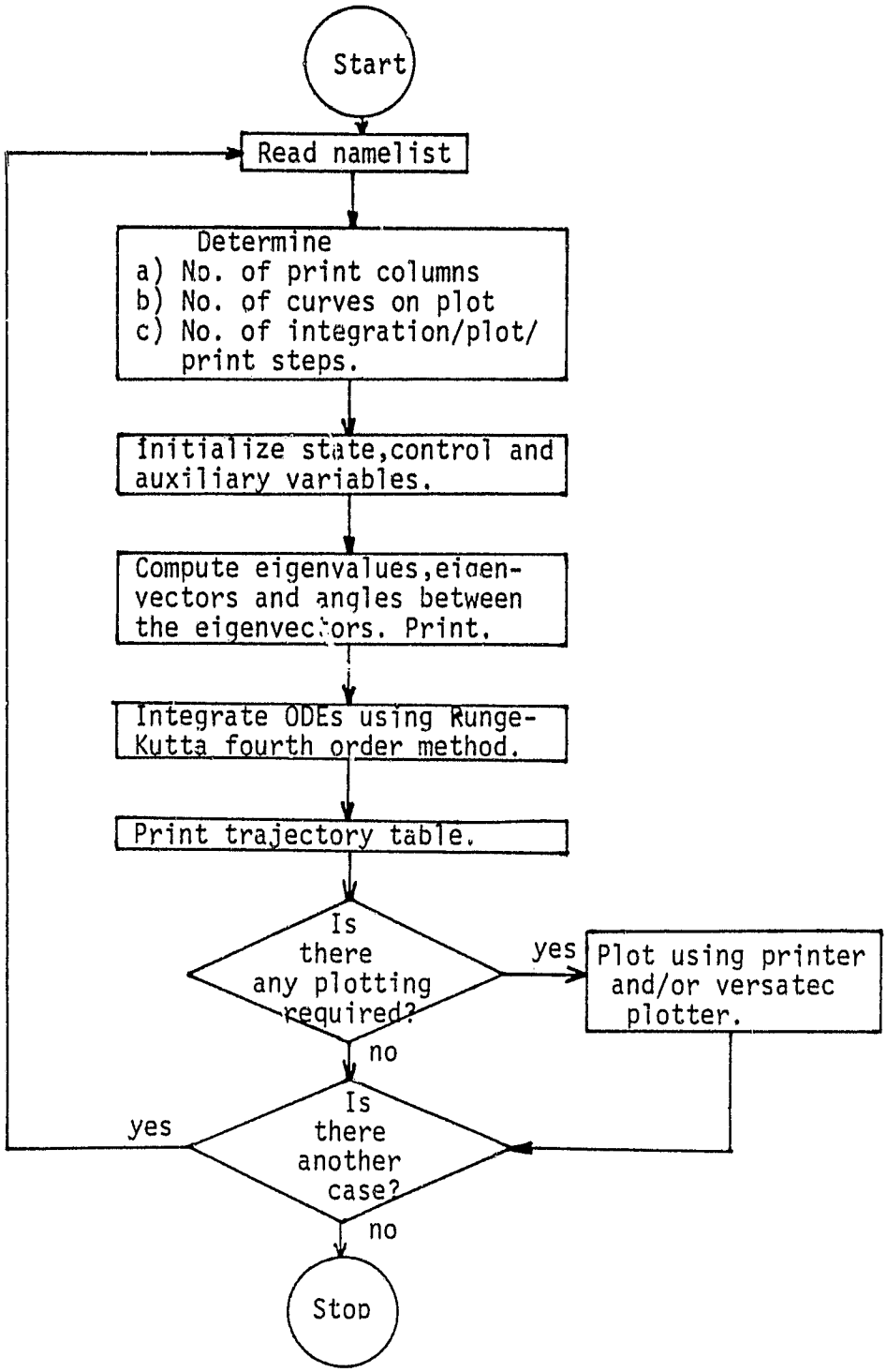
- δa - aileron deflection
- δr - rudder deflection
- δp - yaw control (using asymmetric deflection of drag petals)

} control variables

The yaw control is present only on the Hall aircraft. Chapter III gives the A and B matrices for these models.

In this study we are comparing the performance of three types of controllers. These are the Min K controller, Robust controller and LQR controller. For each type of controller we have considered three cases and so there are a total of nine cases per aircraft. As mentioned earlier, the Min K and robust feedback matrices were generated using the program NEWSON. The LQR

Figure 3 Flowchart for program INTODE



matrices were generated using the program LINEAR. Tables 3 and 2 give these feedback matrices for the Hall and Montgomery aircrafts respectively.

VII Results from Program INTODE

First we discuss the results obtained for the Hall aircraft. The time trajectories obtained with the program INTODE are presented in figures 4, 5 and 6. In these plots the time trajectories of bank angle, yaw angle, aileron deflection, rudder deflection and yaw control deflection are given for the nine control matrices described above. The plots have the angular variables in radians vs time in seconds. Initial conditions for these plots have yaw and bank angle equal to 0.1 radians and all the other variables equal to zero. Some of the main features of the results are given in table 4. The desired time response should have the following features:

(a) the response should settle to the steady state value in minimum time

(b) the response should have minimum overshoot

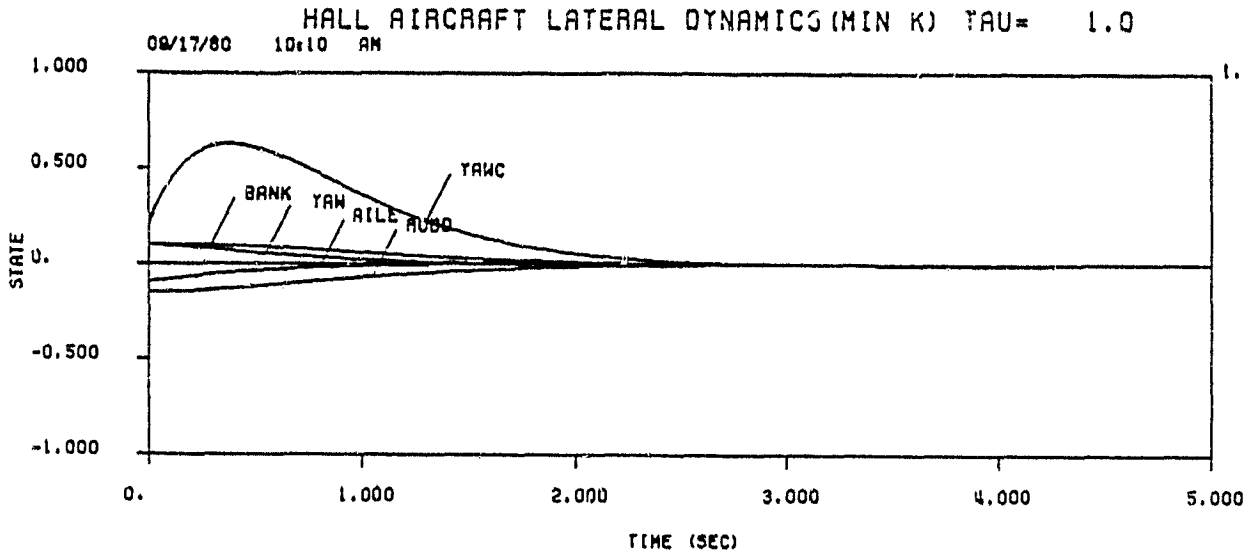
(c) the response should require minimum control effort.

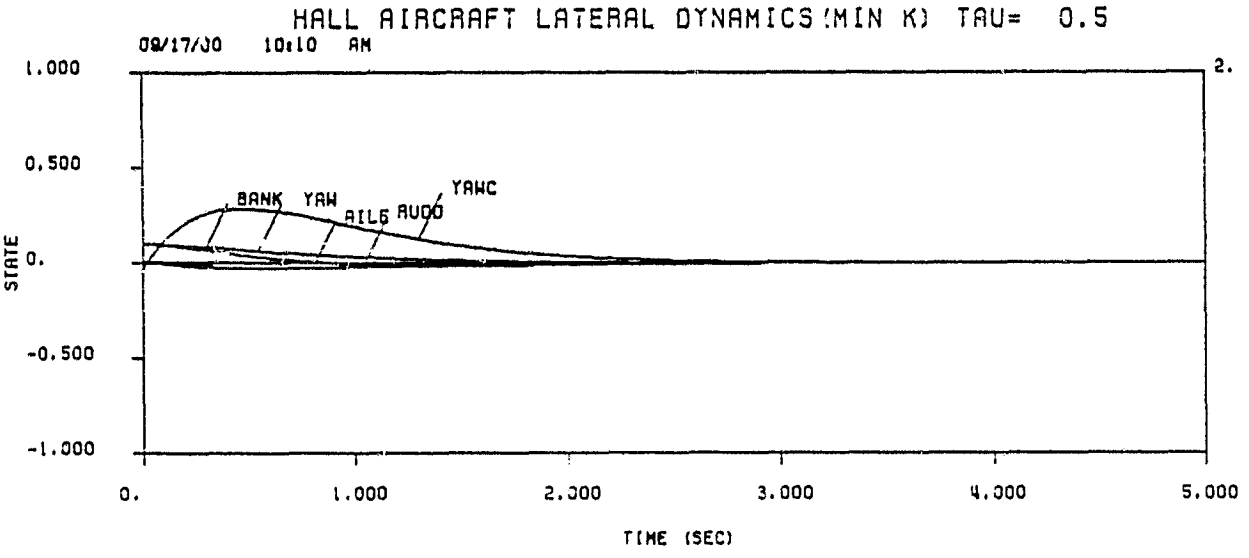
Cases 1, 2 and 3 correspond to the min K controller. Comparing these three cases, we see that the second case (for $\tau = 0.5$, where $\tau = -1/\text{REMAX}$) gives the best response. Time required to settle to the steady state value is about same for all the three cases but the overshoots are minimum for case 2. Also the control effort, given by $\int u^T u dt$, is the least for the $\tau = 0.5$ case. Clearly the third case gives the worst time response.

The three cases which give the time trajectories for Robust controller are cases 4, 5 and 6. The time response for cases 4 and 5 are virtually identical to those for cases 1 and 2 respectively. Here too, for $\tau = 0.5$ (case 5) we get the best response, requiring minimum control effort and having least overshoots.

The remaining three cases are for the LQR controller. We notice that case 7 ($\rho = 100$, where ρ is defined on page 22) requires the minimum control effort but has a settling time much larger than for the other two cases. The best compromise is offered by case 8 ($\rho = 1$).

Figure 4


HALL AIRCRAFT LATERAL DYNAMICS (MIN K) TAU= 1.0
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
YAWC
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)


HALL AIRCRAFT LATERAL DYNAMICS (MIN K) TAU= 0.5
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
YAWC
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

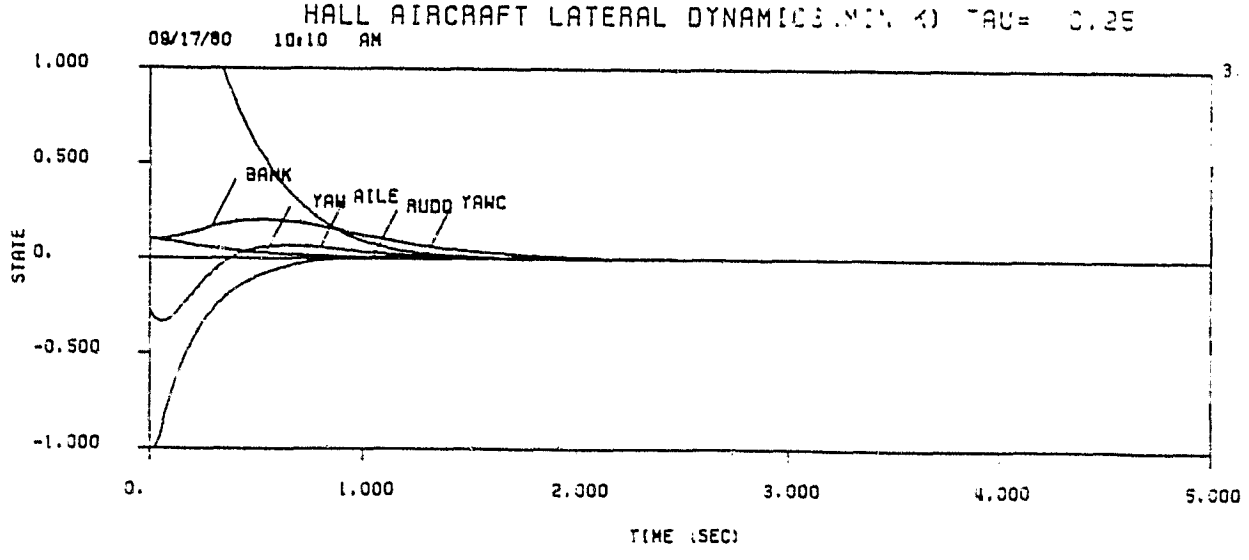
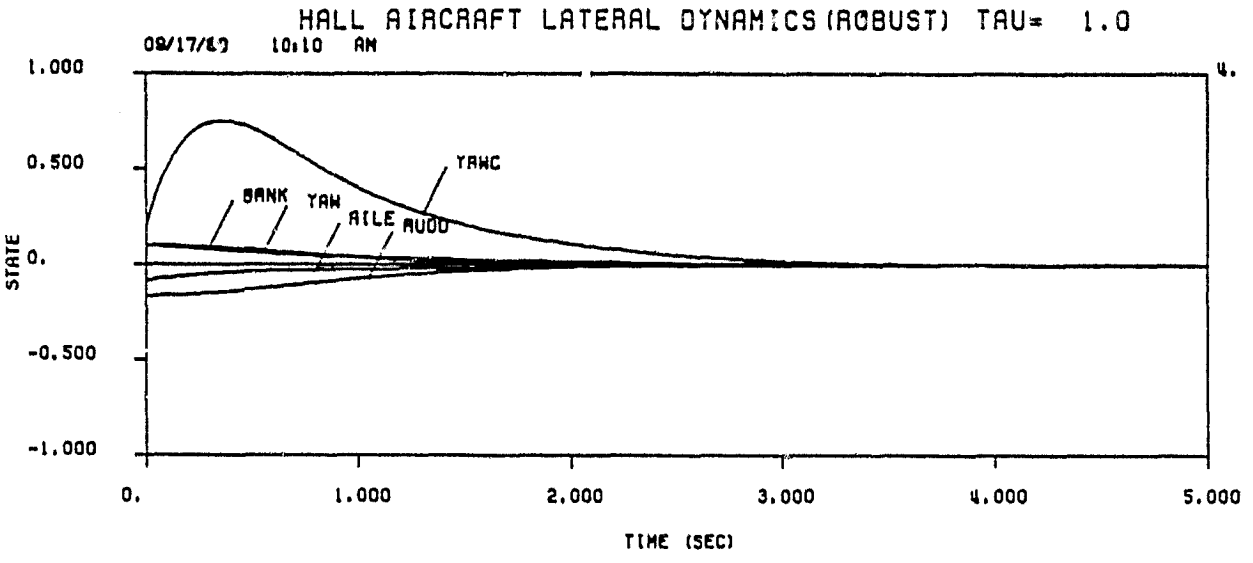
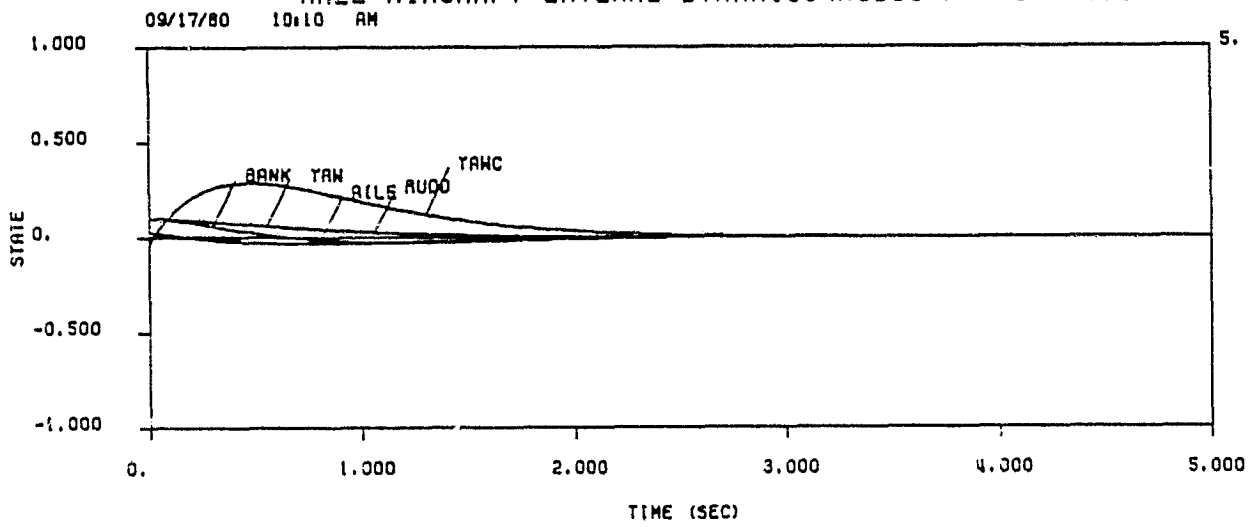

09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
YAWC
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

Figure 5


HALL AIRCRAFT LATERAL DYNAMICS(ROBUST) TAU= 1.0
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
4.
BANK
YAW
AILE
RUDD
YAWC
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)


HALL AIRCRAFT LATERAL DYNAMICS(ROBUST) TAU= 0.5
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
5.
BANK
YAW
AILE
RUDD
YAWC
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

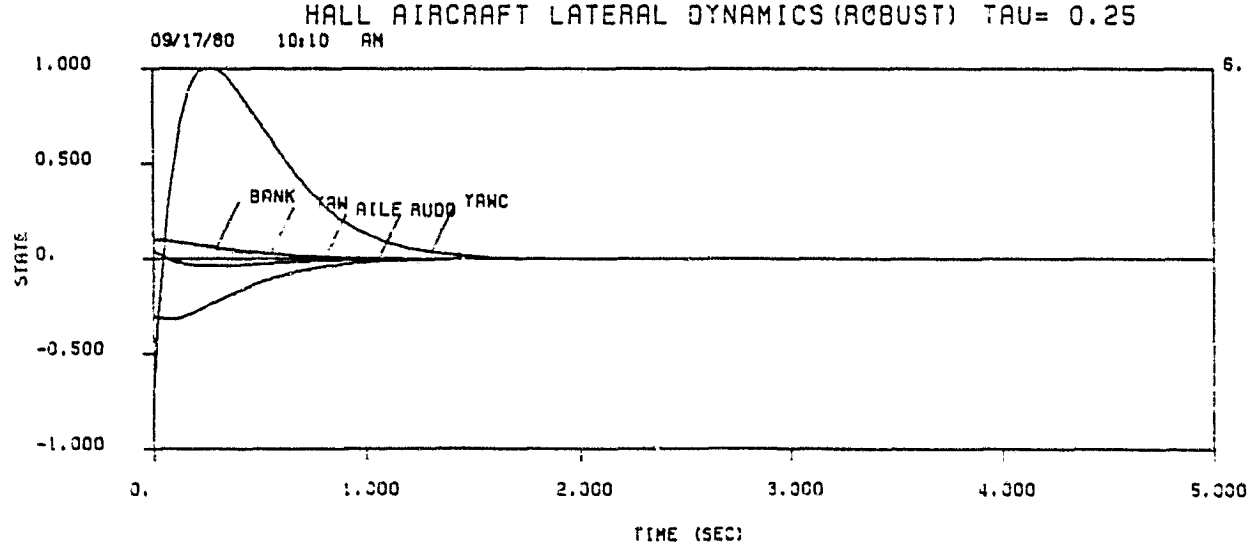
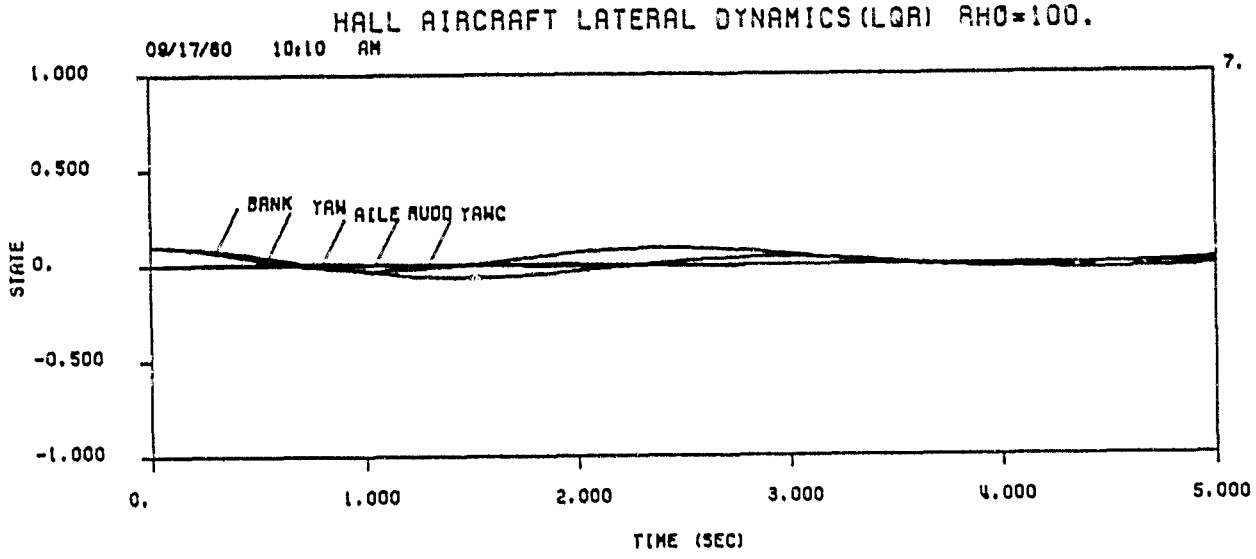
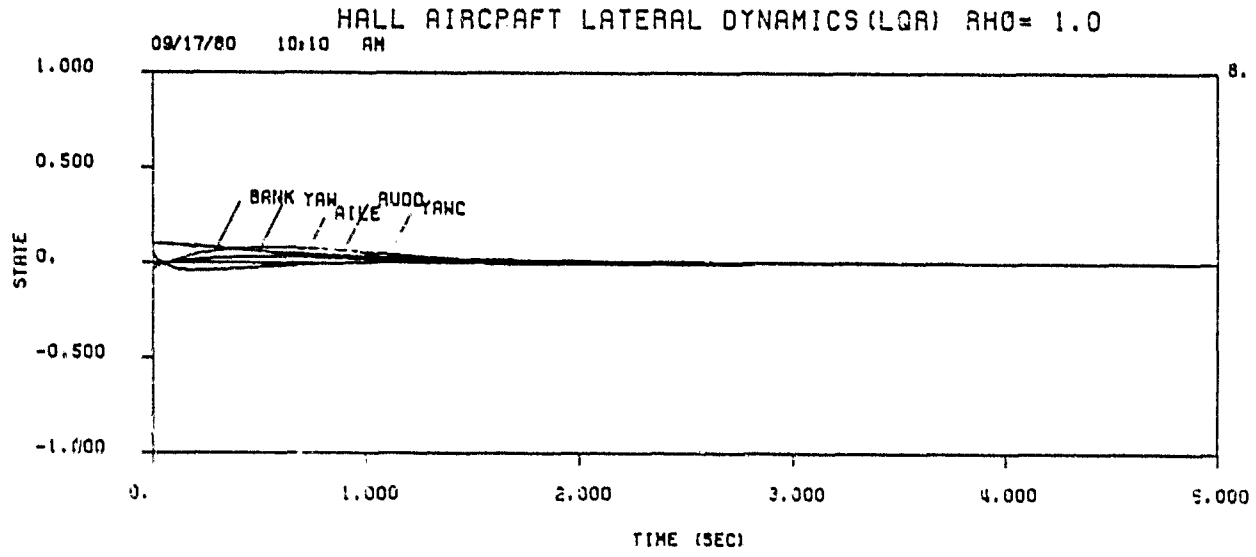

HALL AIRCRAFT LATERAL DYNAMICS(ROBUST) TAU= 0.25
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
6.
BANK
YAW
AILE
RUDD
YAWC
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

Figure 6


HALL AIRCRAFT LATERAL DYNAMICS(LQR) RHO=100.
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
YAWC
7.
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)


HALL AIRCRAFT LATERAL DYNAMICS(LQR) RHO= 1.0
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
YAWC
8.
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

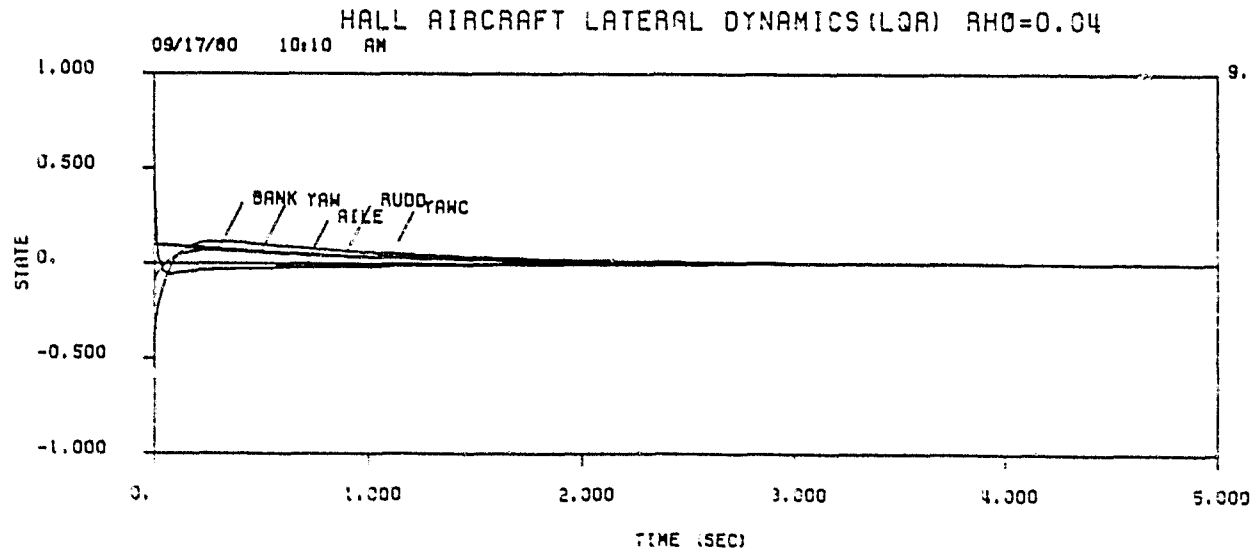

HALL AIRCRAFT LATERAL DYNAMICS(LQR) RHO=0.04
09/17/80 10:10 AM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
YAWC
9.
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

Table 4 Results from the Integration of Hall Aircraft Dynamics

	Min K Controller			Robust Controller			LQR Controller		
	τ = 1.0	τ = 0.5	τ = 0.25	τ = 1.0	τ = 0.5	τ = 0.25	ρ = 100	ρ = 1	ρ = 0.04
Eigenvalues	-0.99519	-1.99818	-4.09501	-3.00771	-1.89012	-5.19458	-3.48596	-14.7623	-72.4585
	-2.39960	$\pm$i0.81421	$\pm$i0.16414	$\pm$i2.48053	$\pm$i1.67409	-3.45839	-0.40705	-0.97982	-1.01422
	-2.10588	-1.99975	-3.85276	-1.09811	-2.21331	-3.74627	-0.31161	-1.60080	-13.4122
	$\pm$i0.45249	$\pm$i0.18111	-7.64974	$\pm$i0.67690	$\pm$i0.26771	$\pm$i1.35823	$\pm$i2.12334	$\pm$i1.65318	-1.07225
Min. eigenvector angles (in degrees)	6.31	5.06	0.86	35.83	9.74	22.26	30.83	41.75	39.98
	23.10	67.67	6.82	43.88	65.88	30.09	51.98	71.50	46.66
	27.57	72.51	7.56	58.57	68.28	32.21	63.00	71.99	50.67
$\|\| K \|\|$	5.6155	4.1777	19.947	6.8268	4.1987	9.5400	0.17438	1.7797	9.2551
$\int x^T x \, dt$	0.02386	0.02622	0.1013	0.02158	0.03185	0.02666	0.08993	0.02240	0.01966
$\int u^T u \, dt$	0.3311	0.06697	1.024	0.4505	0.06724	0.4515	0.000378	0.006098	0.01654
Max. Aileron (rad.)	-0.096	-0.027	-0.313	-0.083	-0.031	0.042	0.01083	0.044	0.424
Max. Rudder (rad.)	-0.149	0.006	-1.221	-0.163	0.00545	-0.314	0.01427	0.080	-0.365
Max Yaw Control (Radians)	0.630	0.286	1.627	0.748	0.288	1.006	0.005401	0.030	-0.085

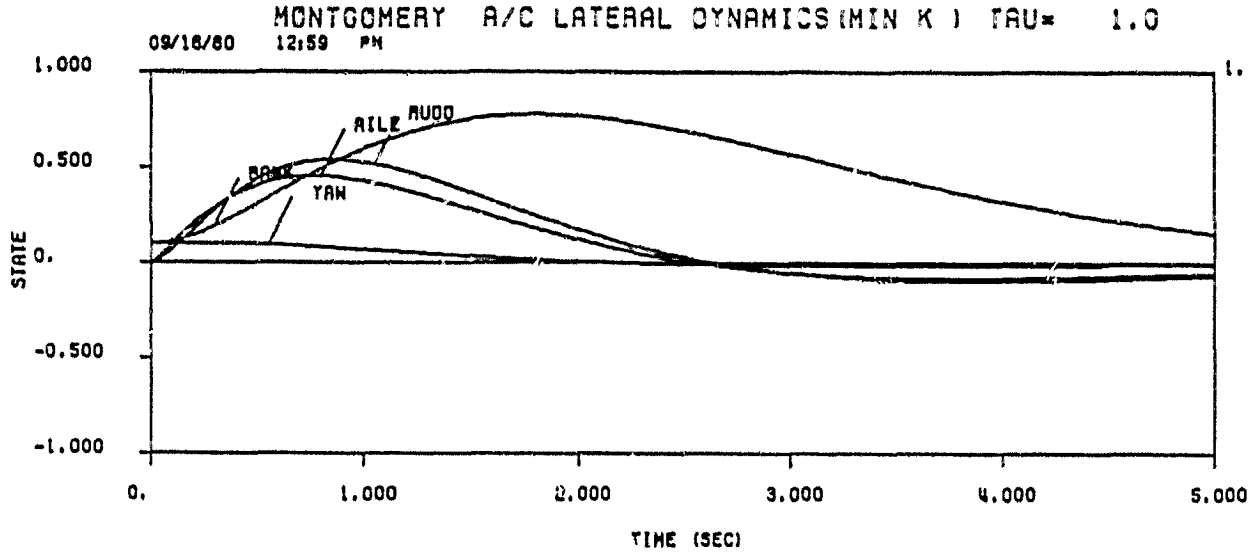
Comparison of cases 2, 5 and 8 provides some information on the relative performance of the three types of controllers considered in this study. The response for case 8 is significantly superior than the response for cases 2 and 5, which are nearly identical. Thus the LQR controller (with ρ = 1) provides the best time response for the Hall aircraft. Another interesting observation is that moving the eigenvalues further to the left does not always improve the time response. As τ (or ρ) decreases, the eigenvalues in general move further to the left. When we go from τ = 1 (or ρ = 100) to τ = 0.5 (or ρ = 1) the time response improves but when τ (or ρ) is further reduced to 0.25 (or 0.04) the response deteriorates.

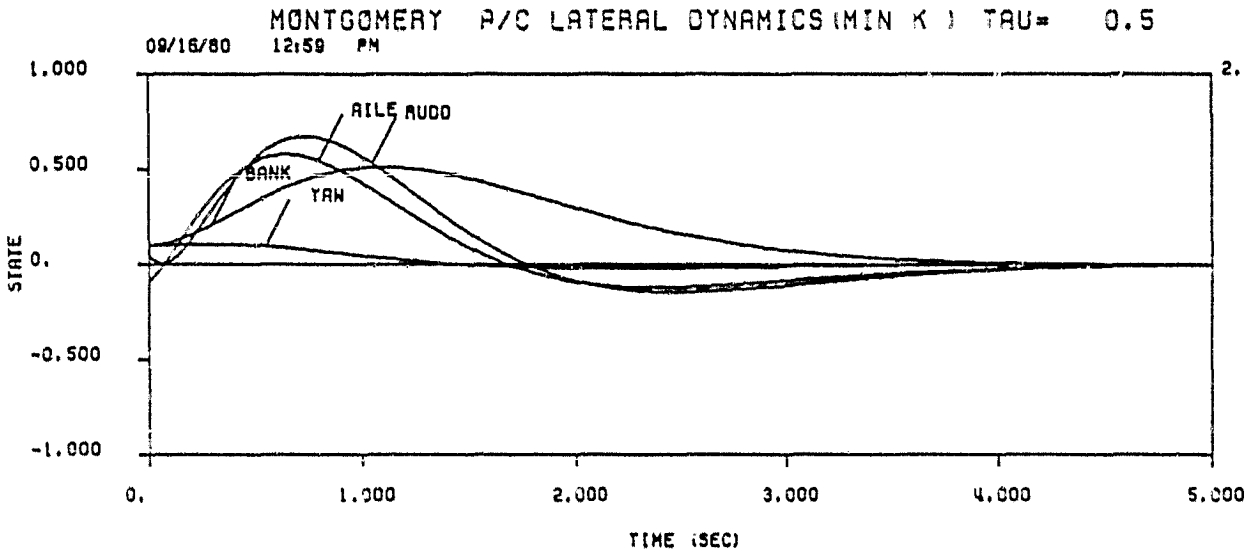
Figures 7, 8 and 9 give the time trajectories obtained for the Montgomery aircraft. Each plot corresponds to a specific case and contains the time response of bank angle, yaw angle, aileron deflection and rudder deflection. Initial conditions were same as for the Hall aircraft, i.e., bank and yaw angles are taken as 0.1 radians while all other variables are equal to zero. These plots have the variables in radians vs time in seconds. Table 5 presents the main features of the results.

Using the criterion mentioned earlier, we may infer that of the three cases of Min K controller case 3 (τ = 0.25) gives the best time response, but this requires a very large maximum rudder deflection (-1.715 radians). As this is not practically possible, this case is discarded. Comparing the other two cases, we notice that the second case requires slightly more control effort but it has a much smaller settling time. Hence of these three cases, the best compromise is offered by case 2 (τ = 0.5).

Comparison of the results for Robust controller shows that cases 5 (τ = 0.5) and 6 (τ = 0.25) give nearly identical response. Case 6 requires slightly lower maximum control deflections and overall control effort, where as case 5 has a slightly smaller settling time. If we compare the overall

Figure 7


MONTGOMERY A/C LATERAL DYNAMICS (MIN K) TAU= 1.0
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
AILE
RUDD
BANK
YAW
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)


MONTGOMERY A/C LATERAL DYNAMICS (MIN K) TAU= 0.5
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
AILE
RUDD
BANK
YAW
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

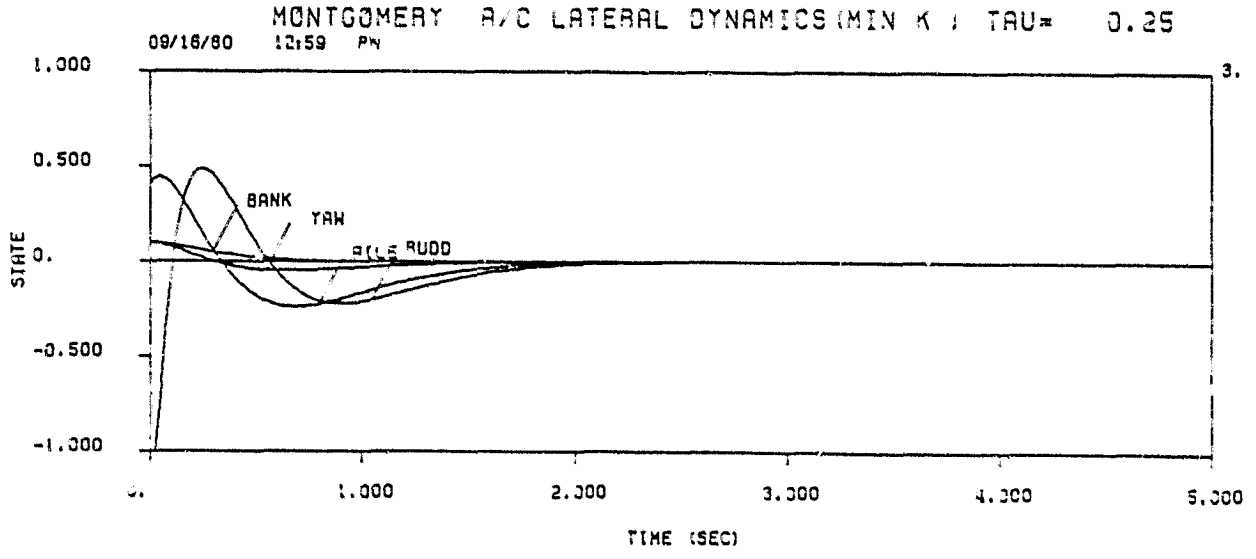
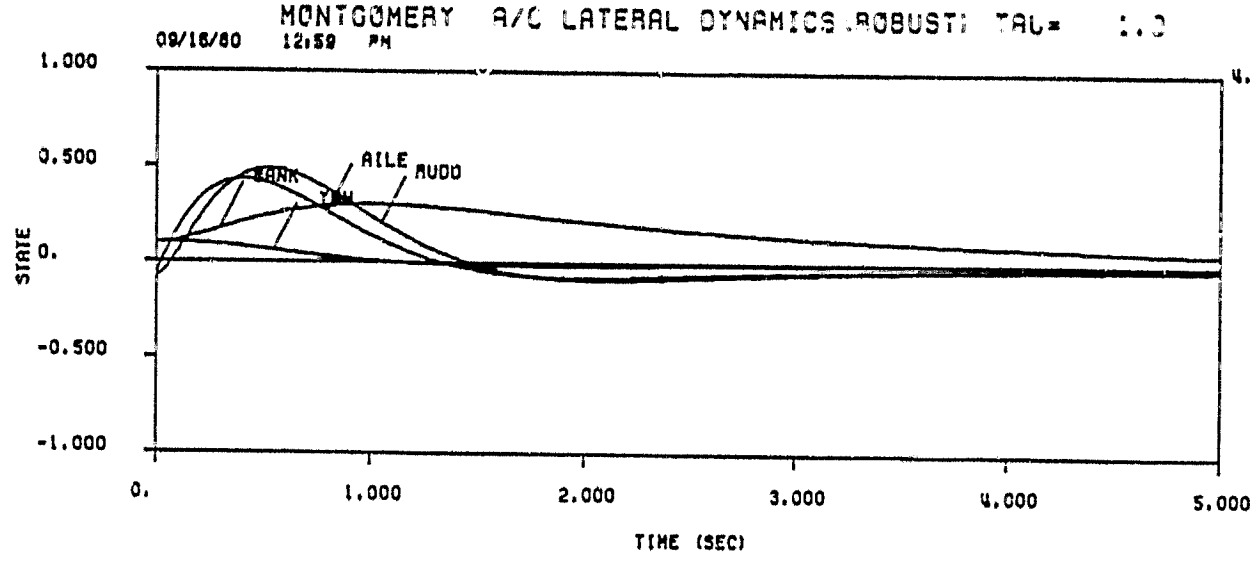
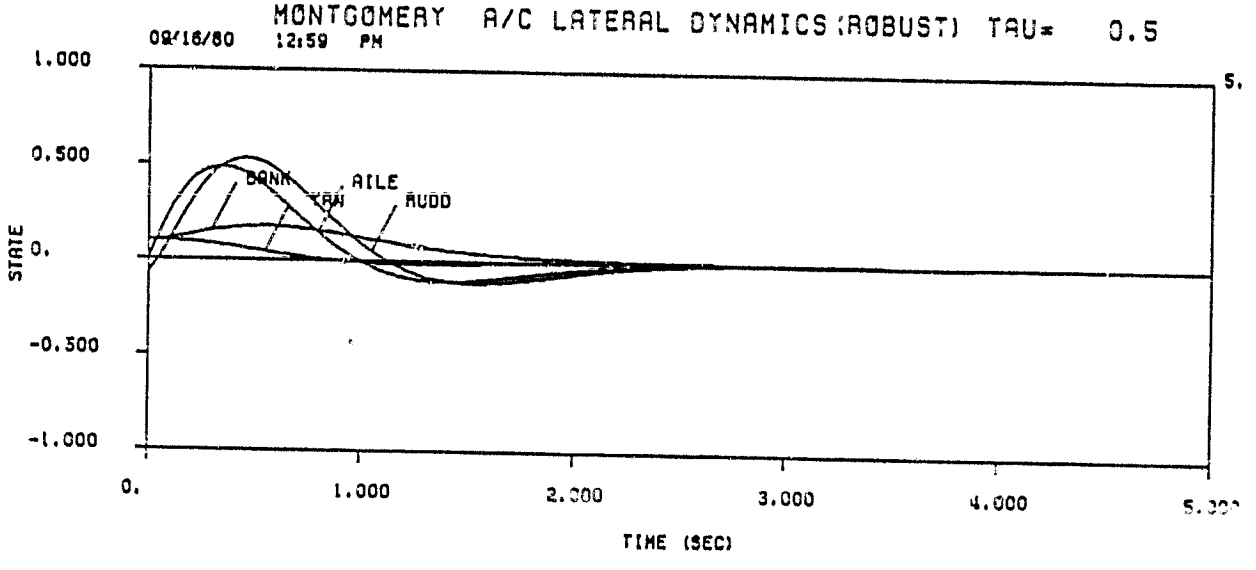

MONTGOMERY A/C LATERAL DYNAMICS (MIN K) TAU= 0.25
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
RUDD
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

Figure 8


MONTGOMERY A/C LATERAL DYNAMICS (ROBUST) TAU= 1.0
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)


MONTGOMERY A/C LATERAL DYNAMICS (ROBUST) TAU= 0.5
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
0.
1.000
2.000
3.000
4.000
TIME (SEC)

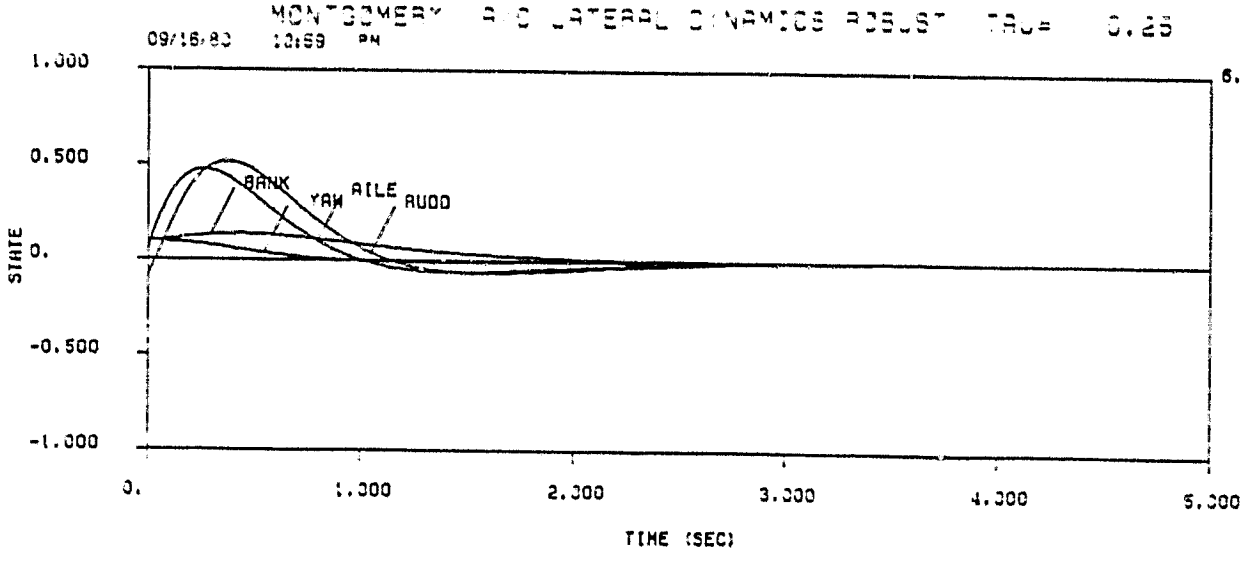
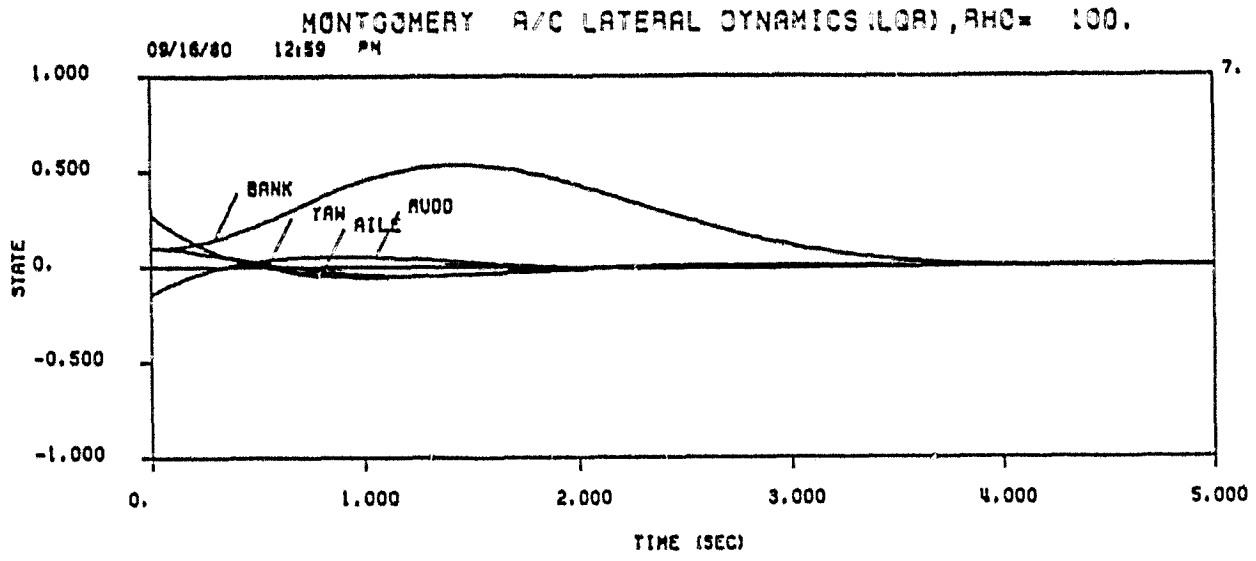
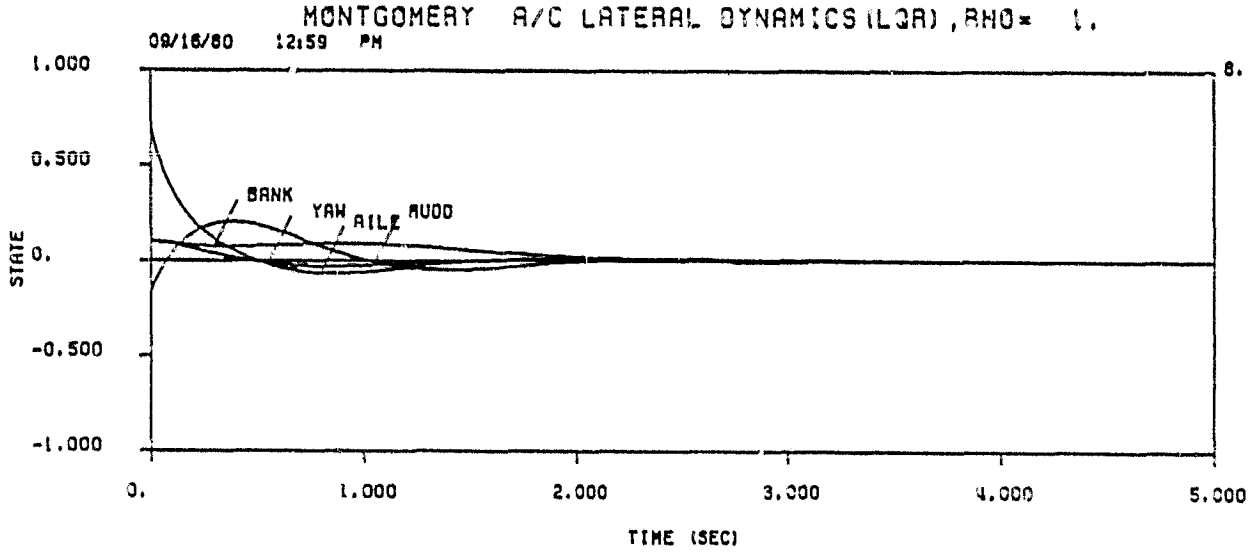

09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

Figure 9


MONTGOMERY A/C LATERAL DYNAMICS(LQR),RHO= 100.
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
7.
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)


MONTGOMERY A/C LATERAL DYNAMICS(LQR),RHO= 1.
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
8.
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

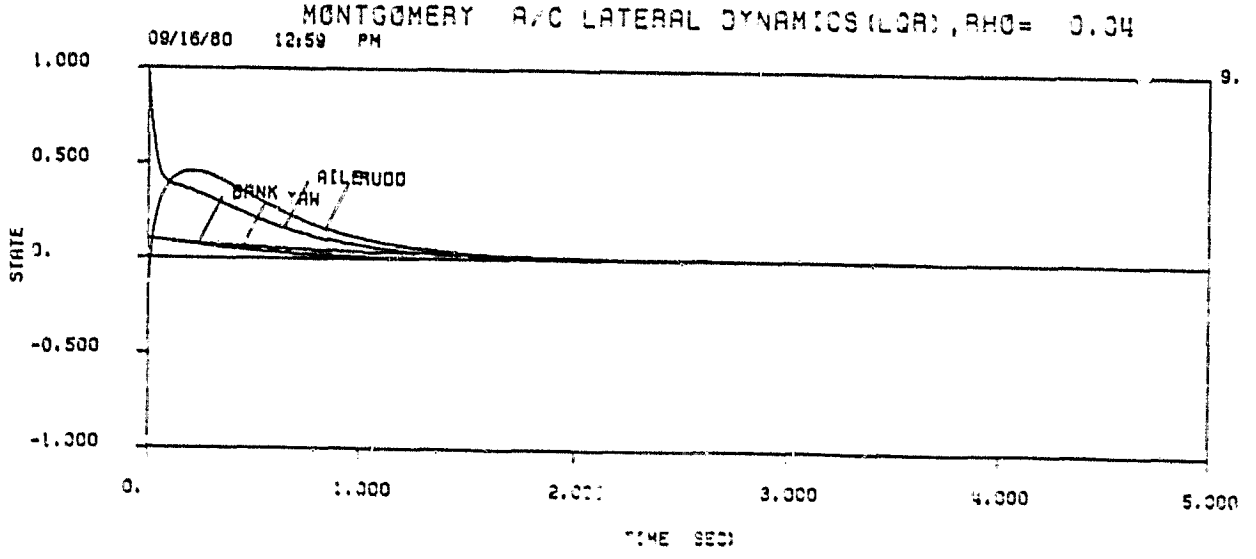

MONTGOMERY A/C LATERAL DYNAMICS(LQR),RHO= 0.04
09/16/80 12:59 PM
1.000
0.500
0.
-0.500
-1.000
STATE
BANK
YAW
AILE
RUDD
9.
0.
1.000
2.000
3.000
4.000
5.000
TIME (SEC)

Table 5 Results from the Integration of Montgomery Aircraft Dynamics

	Min K Controller			Robust Controller			LQR Controller		
	τ = 1.0	τ = 0.5	τ = 0.25	τ = 1.0	τ = 0.5	τ = 0.25	ρ = 100	ρ = 1	ρ = 0.04
Eigenvalues	-4.82860	-1.99521	-3.99847	-6.66671	-2.25198	-4.02489	-2.84629	-9.54026	-42.0340
	-0.99789 $\pm$i0.47245	$\pm$i0.97874	$\pm$i1.78356	-2.11222 $\pm$i1.83253	$\pm$i2.86409	$\pm$i0.37918	-1.04203	-1.49551	-1.03527
		-1.99513	-4.0032		-4.42521	-3.04453	-1.04682	$\pm$i3.08237	-6.53534
	-0.99785	$\pm$i0.68740	-6.93968	-0.42337	-1.46849	$\pm$i0.36014	$\pm$i1.66794	-1.03512	-2.38666
Min. eigenvector angles (in degrees)	5.63	1.11	4.96	51.22	26.74	8.78	15.77	23.70	21.28
	47.00	2.12	13.98	53.05	34.66	25.77	26.50	45.80	51.12
	52.58	89.20	17.44	67.44	66.48	58.07	30.60	53.07	70.42
$\|\| K \|\|$	4.0597	4.9093	16.780	4.0158	4.2932	4.7474	3.1197	6.4843	11.686
$\int x^T x\, dt$	1.838	0.6615	0.06562	0.258	0.07921	0.04739	0.7189	0.04791	0.02122
$\int u^T u\, dt$	0.560	0.5949	0.2217	0.264	0.02565	0.02219	0.01608	0.06171	0.1874
Max. Aileron (rad.)	0.455	0.578	0.447	0.434	0.488	0.474	0.273	0.694	1.213
Max. Rudder (rad.)	0.540	0.671	-1.715	0.487	0.533	0.515	-0.145	0.204	0.457

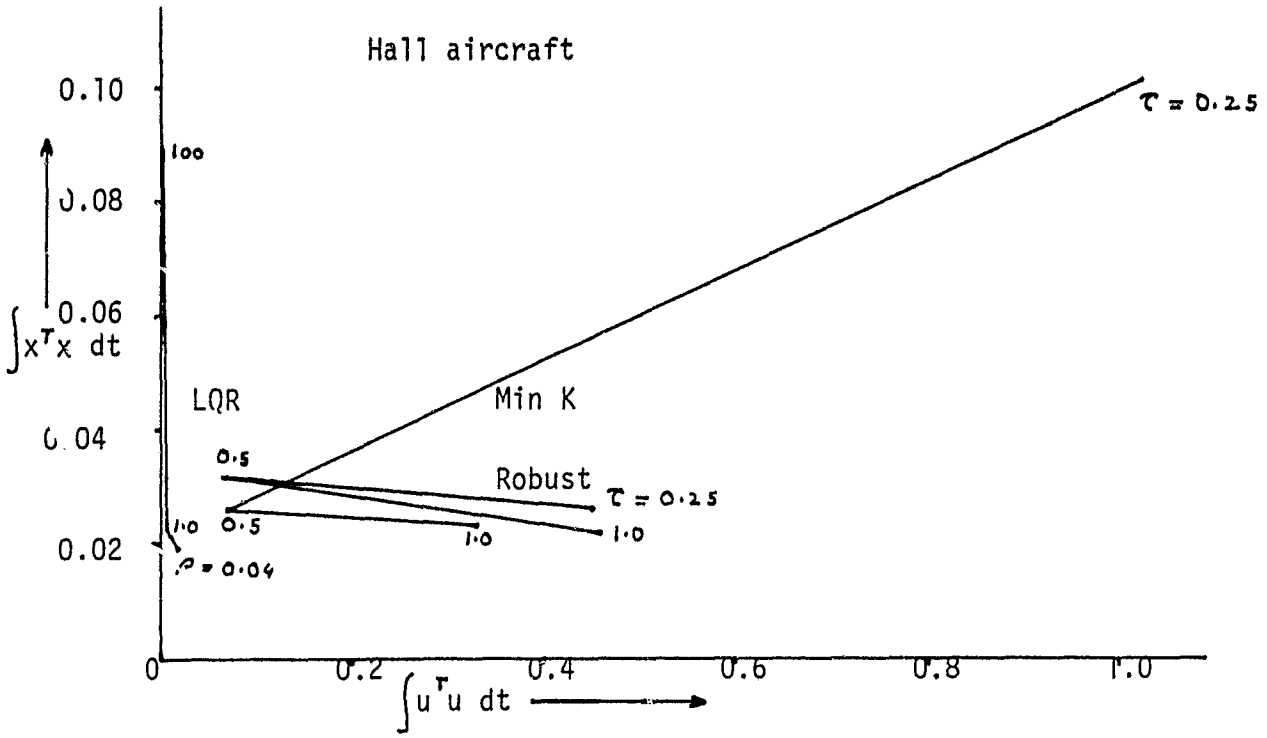
state deviation (given by $\int x^T x\, dt$) we find that case 6 gives a lower value. So we may choose case 6 as the one giving the best response among the three cases of Robust controller.

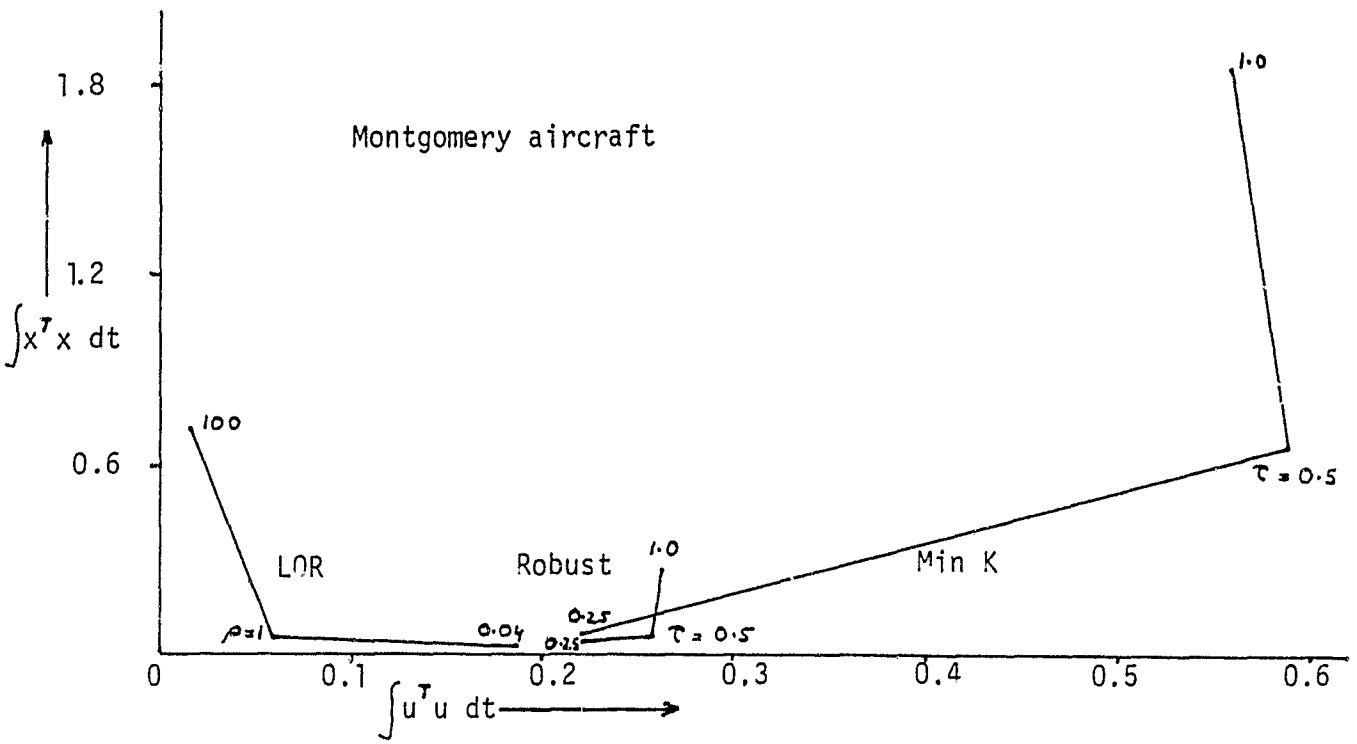
For the LQR controller, we notice that as ρ decreases the time required to reach steady state also decreases but the total control effort and maximum control deflections increase. For case 9 ($\rho = 0.04$) an unacceptably large aileron deflection is required (1.213 radians). Case 7 ($\rho = 100$) requires a large settling time and has a large overshoot in the bank angle response. So case 8 ($\rho = 1$) has a better time response than cases 7 and 9.

Of the three cases 2, 6 and 8, case 8 requires the minimum control effort and also has the minimum settling time, hence case 8 gives the best time response. So for the Montgomery aircraft, the best response is given by LQR controller with $\rho = 1.0$. Here too moving the eigenvalues further to the left (i.e., making the real part more negative) does not necessarily mean a better time response.

Figure 10 gives the plots of the control effort, $\int u^T u\, dt$. versus state error $\int x^T x\, dt$ for both the aircraft. As shown, the LQR controller yields lowest control effort and state error. This is to be expected since the LQR is by definition the one which minimizes these integrals.

Figure 10 Integral Control Effort VS State Deviation





VIII Program PERTB

This program was written to compare the robustness of the three types of controllers considered here. The robustness of a controller is measured in terms of the insensitivity of the closed loop eigenvalues to variations or uncertainties in the A and B matrices, the more insensitive the closed loop eigenvalues, the more robust the controller.

The elements of the A and B matrices are perturbed around their specified value by the help of random numbers.

i.e. $$PA(I,J) = (1 + Rand \times P)\, A(I,J)$$

where Rand is a random number between -1 and 1

P is a fraction giving the maximum perturbation and PA is the perturbed A matrix.

Similarly $$PB(I,J) = (1 + Rand \times P)\, B(I,J)$$

The closed loop eigenvalues of this perturbed system are calculated. This constitutes a single sample. The program also calculates and stores REMAX, REMIN and RTOMAX,

where

REMAX is Max (Real part of eigenvalues for a particular sample)

REMIN is Min (Real part of eigenvalues for a particular sample)

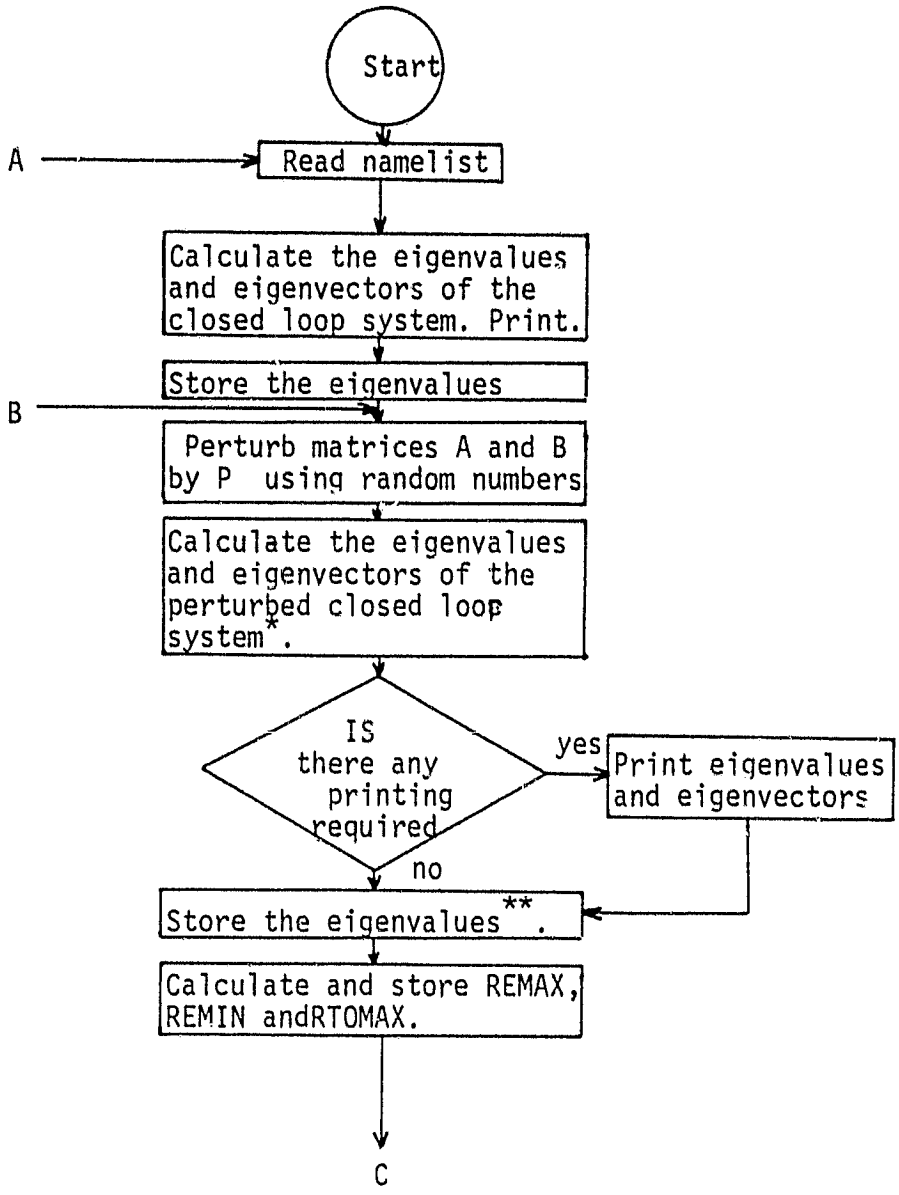
RTOMAX is Max (Ratio of imaginary to real parts of eigenvalues for a particular sample)

The percentage change in REMAX, REMIN and RTOMAX from the unperturbed values are then calculated. The program repeats this for a large number of samples, typically 1000. Statistics on REMAX, REMIN and RTOMAX and their percentage changes are calculated and printed. The program calculates the maximum, minimum, mean and standard deviation and presents the variation in the form of tabular histogram.

The variation in the eigenvalues due to perturbations in the matrices A and B can be presented very elegantly as a scatter diagram on the complex plane. The unperturbed eigenvalues are circled so as to indicate the amount of scatter in the eigenvalues due to perturbations. Such a pictorial representation provides qualitative information on the robustness of the system. It is a very helpful tool in comparing the robustness of different controllers.

Figure 11 gives the flow chart for the program PERTB and complete listing is given in Appendix D . Data is given as the namelist PERT and details of the variables forming the namelist are included in the program listing. As in program INTODE indices ICONT, IPRINT and IPLOT provide flexibility in program execution. Sample program input files are included with the program listings in the appendixes C and D.

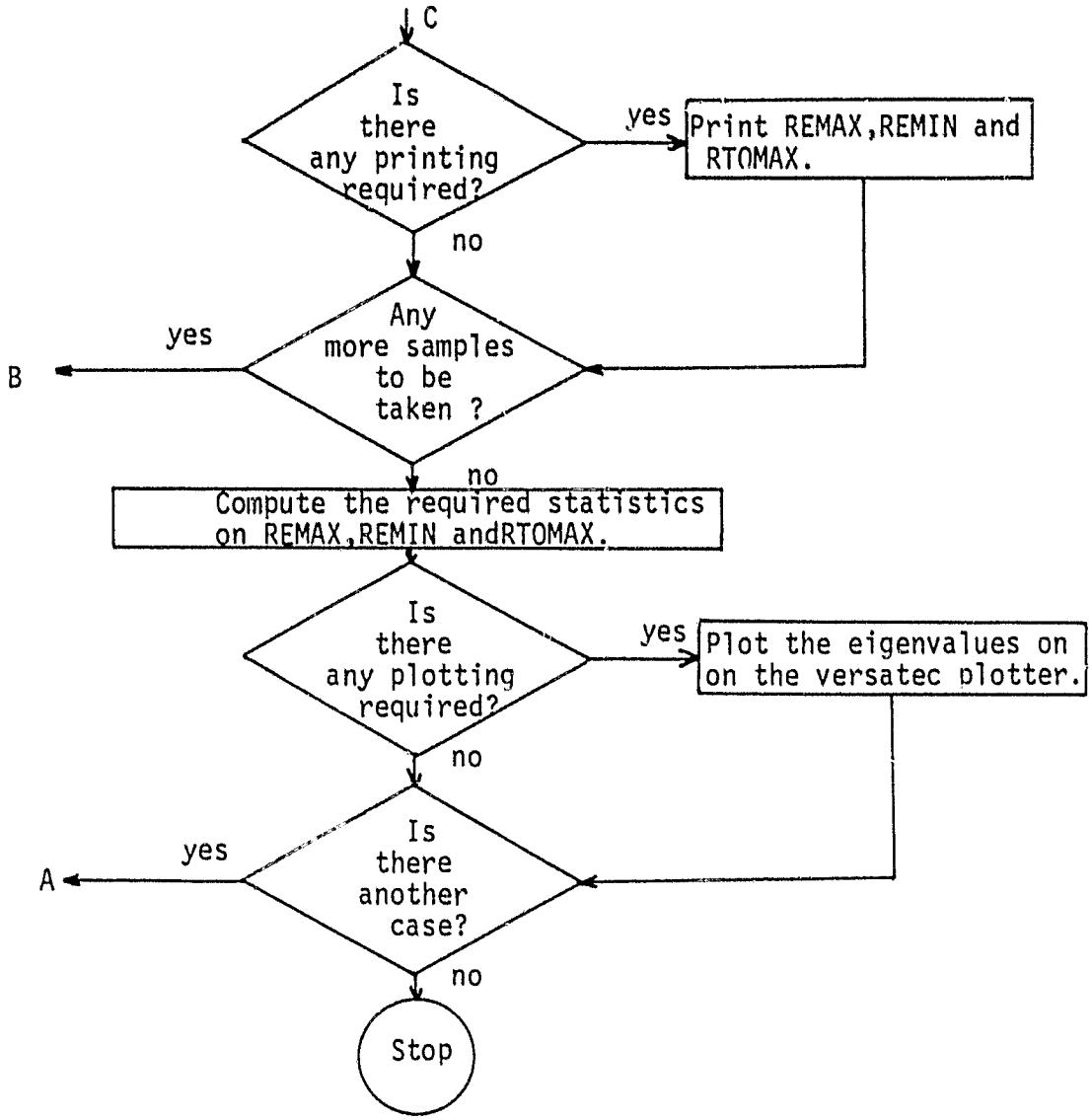
Figure 11 Flowchart for program PERTB



*

**Eigenvectors are not required for Monte Carlo runs
For statistics and plotting
REMAX = Max(real part of eigenvalues)
REMIN = Min(real part of eigenvalues)
RTOMAX = Max(ratio of imaginary to real part of eigenvalues)

Figure 11(cont.)



REMAX = max(real part of eigenvalues)

REMIN = min(real part of eigenvalues)

RTOMAX= max(ratio of imaginary to real part of eigenvalues)

IX Results from Program PERTB

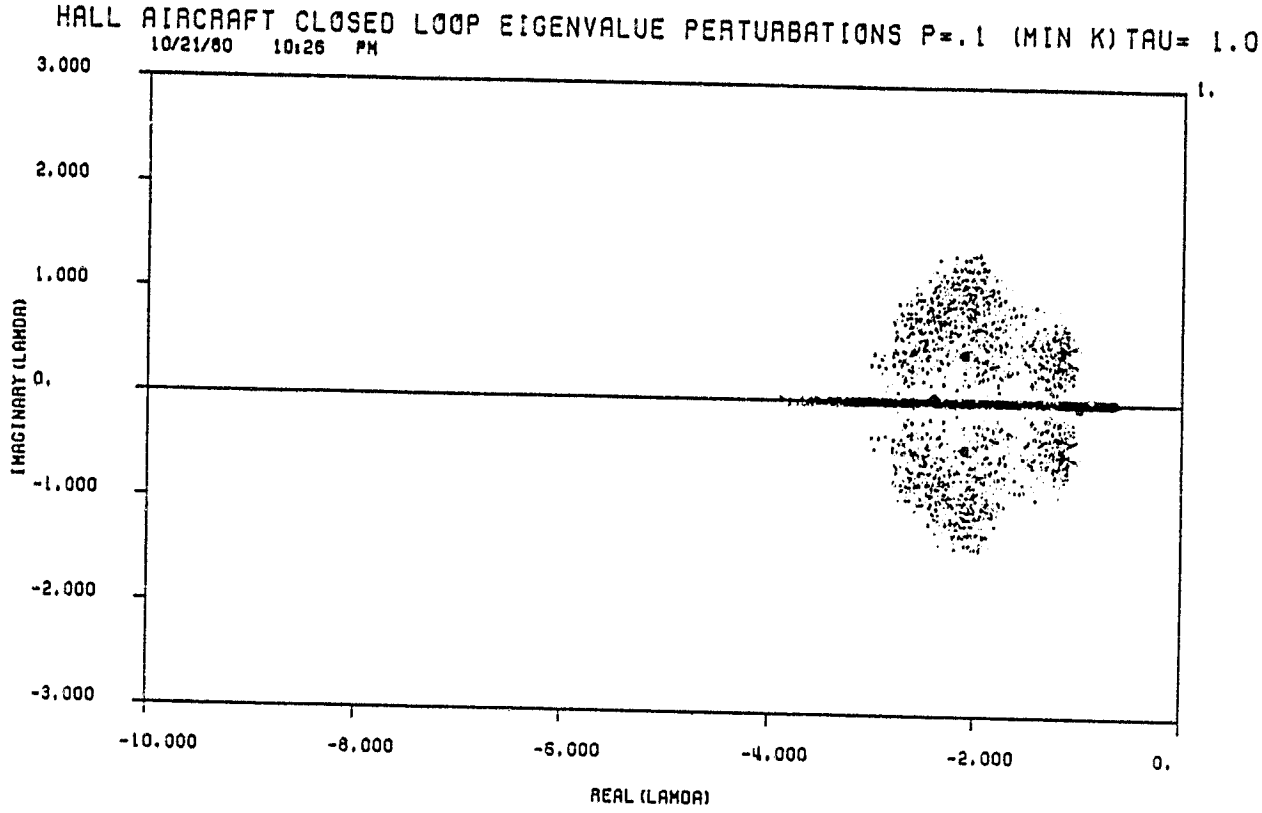
In this section we compare the robustness of the controllers. This is done by observing the scatter of the eigenvalues due to perturbations in matrices A and B. For each case a thousand samples, each with 10% perturbations of the elements of the matrices A and B, were taken.

Figures 12 to 16 give the scatter diagrams of the eigenvalues. Tables and 6 present the statistics obtained on REMAX, REMIN and RTOMAX. In the scatter diagrams, the unperturbed eigenvalues have been circled. This helps in estimating the amount of scatter due to the perturbations. Secondly, a number of perturbed eigenvalues are real and so are plotted on the real axis. As such, it is difficult to get an idea about their distribution along the real axis. So for the figures, we attach a small randomly generated imaginary part to real eigenvalues obtained after perturbation. These eigenvalues now form a narrow band around the real axis. The density of the band at any location gives an idea of the number of perturbed eigenvalues on the real axis at that location.

We first discuss the results obtained for Hall aircraft. On studying the plots we notice that for both Min k and Robust controllers, as τ decreases, the scatter of the eigenvalues obtained after perturbations first decreases slightly and then increases. This decrease is more apparent for Min k controller. The Robust controller gives near identical scatter for both $\tau = 1.0$ and $\tau = 0.5$. For any particular value of τ, the Robust controller gives a little less scatter than the Min k controller.

As shown by the scatter diagrams, perturbations on A and B matrices produce very little change in the eigenvalues of the LQR controller. So the LQR controller is much more robust than the Min k and Robust controllers. It should be noted that one unperturbed eigenvalue for $\tau = 1.0$ and two unperturbed eigenvalues for $\tau = 0.04$ are not shown on their respective

Figure 12


HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (MIN K) TAU= 1.0
10/21/80 10:26 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
1.
REAL (LAMDA)

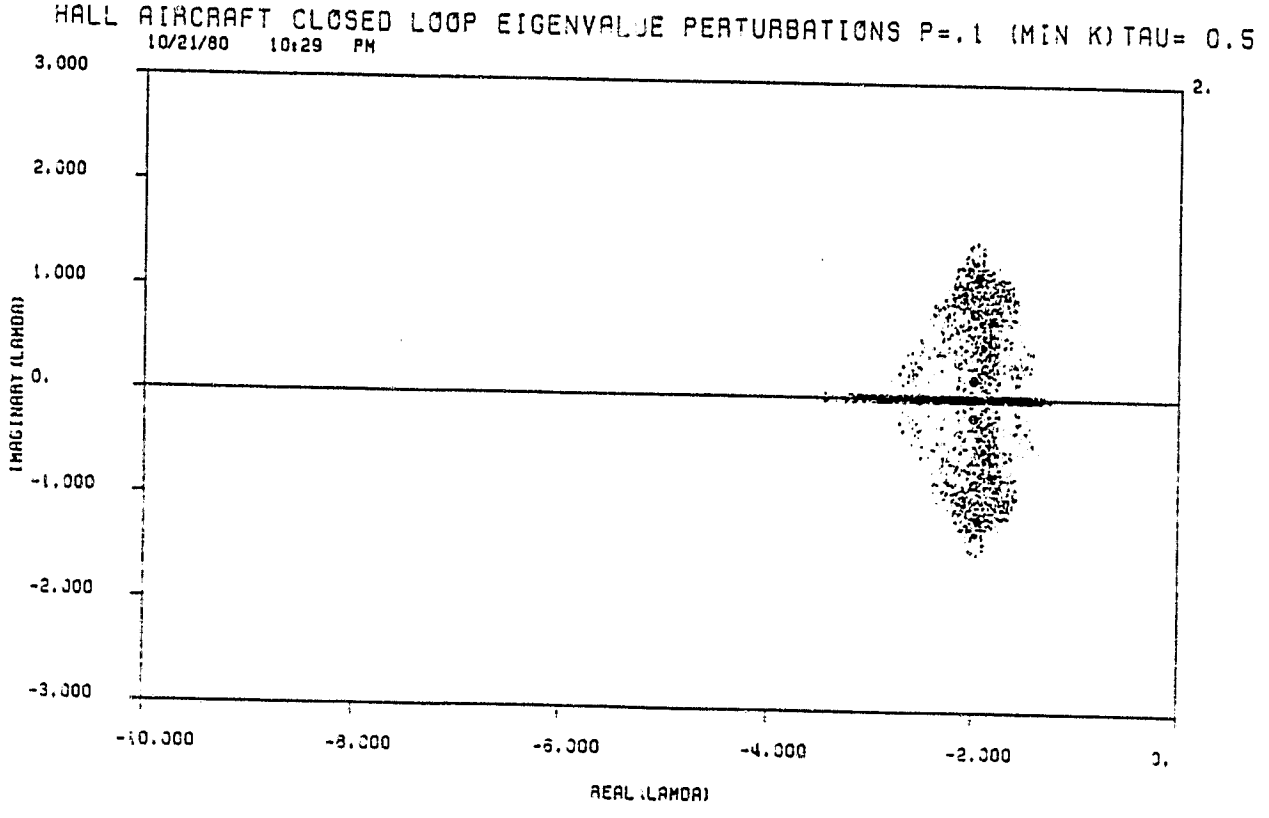
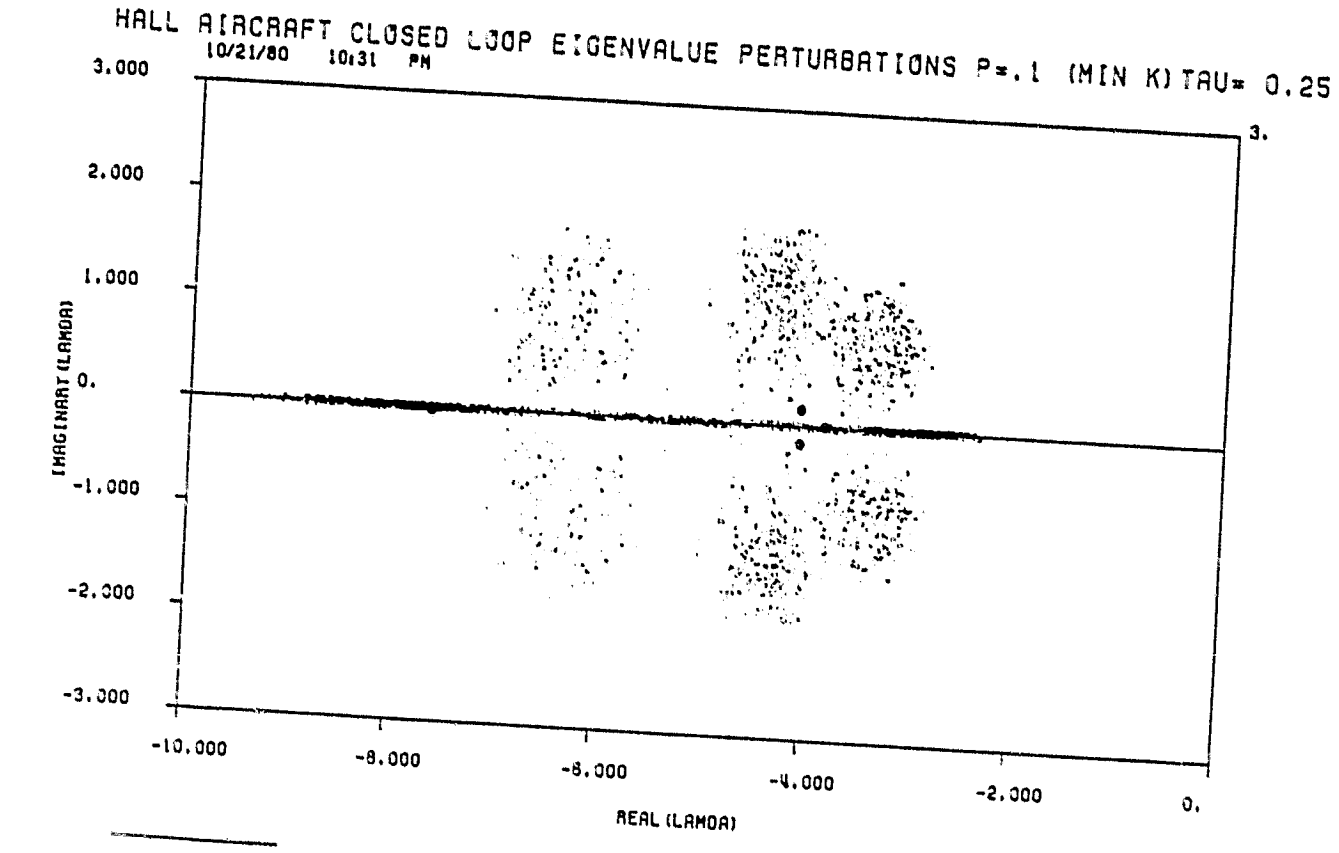

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (MIN K) TAU= 0.5
10/21/80 10:29 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
2.
REAL (LAMDA)

Figure 13


HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (MIN K) TAU= 0.25
10/21/80 10:31 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
3.
REAL (LAMDA)

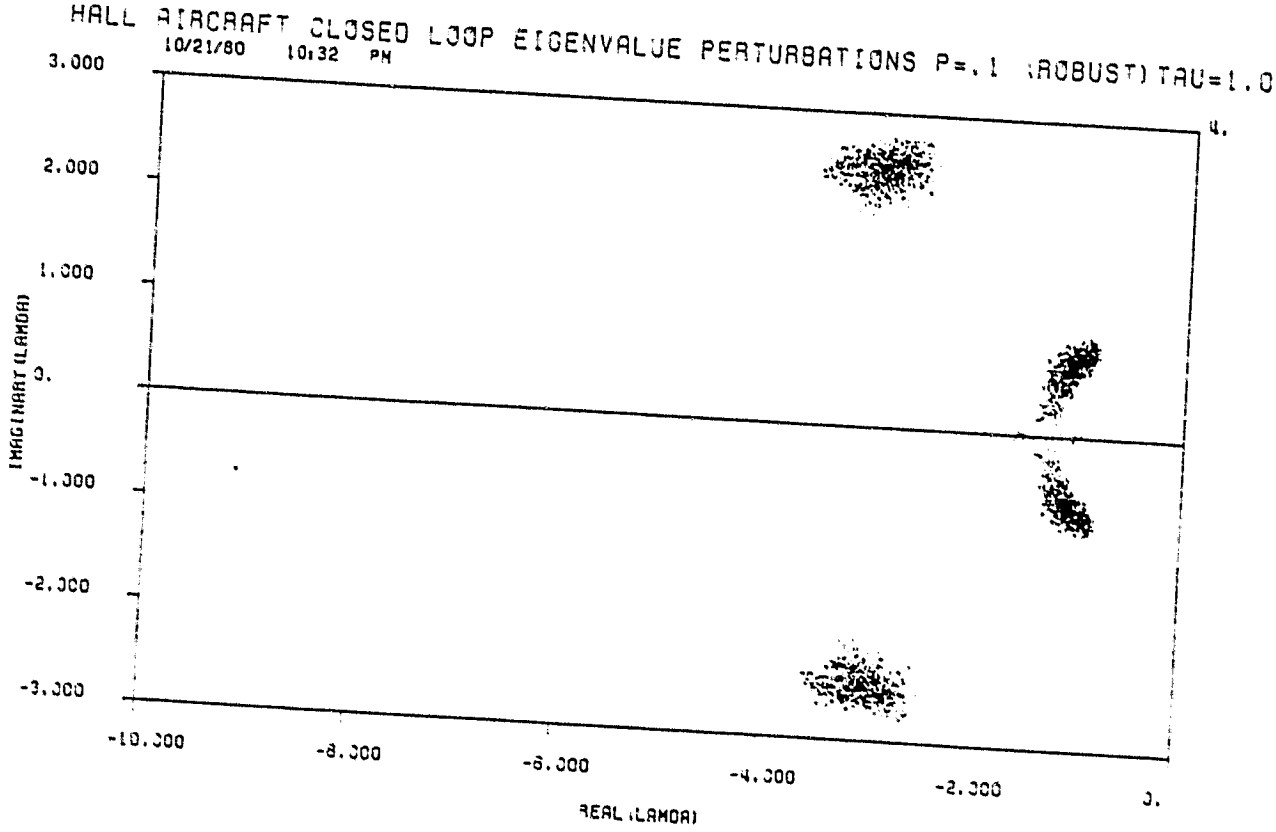
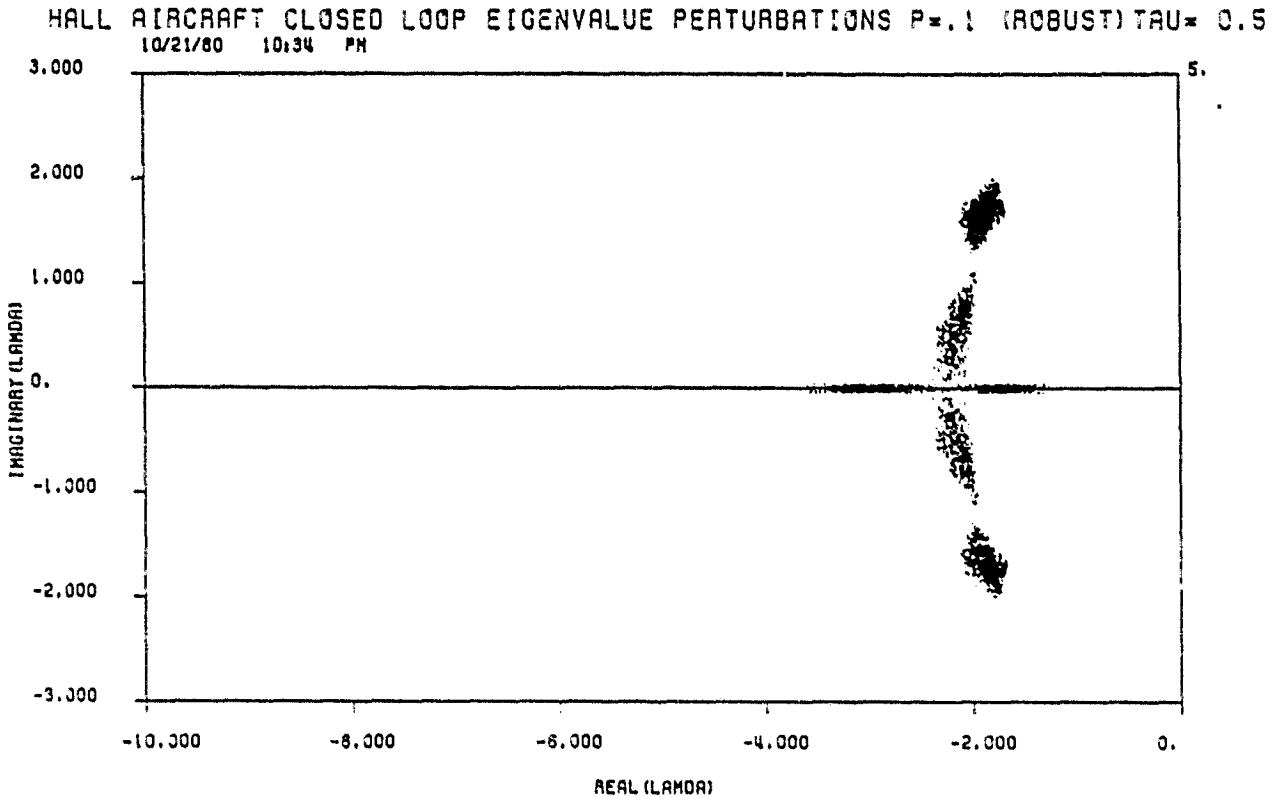

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (ROBUST) TAU=1.0
10/21/80 10:32 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
4.
REAL (LAMDA)

Figure 14


HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (ROBUST) TAU= 0.5
10/21/80 10:34 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
5.
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
REAL (LAMDA)

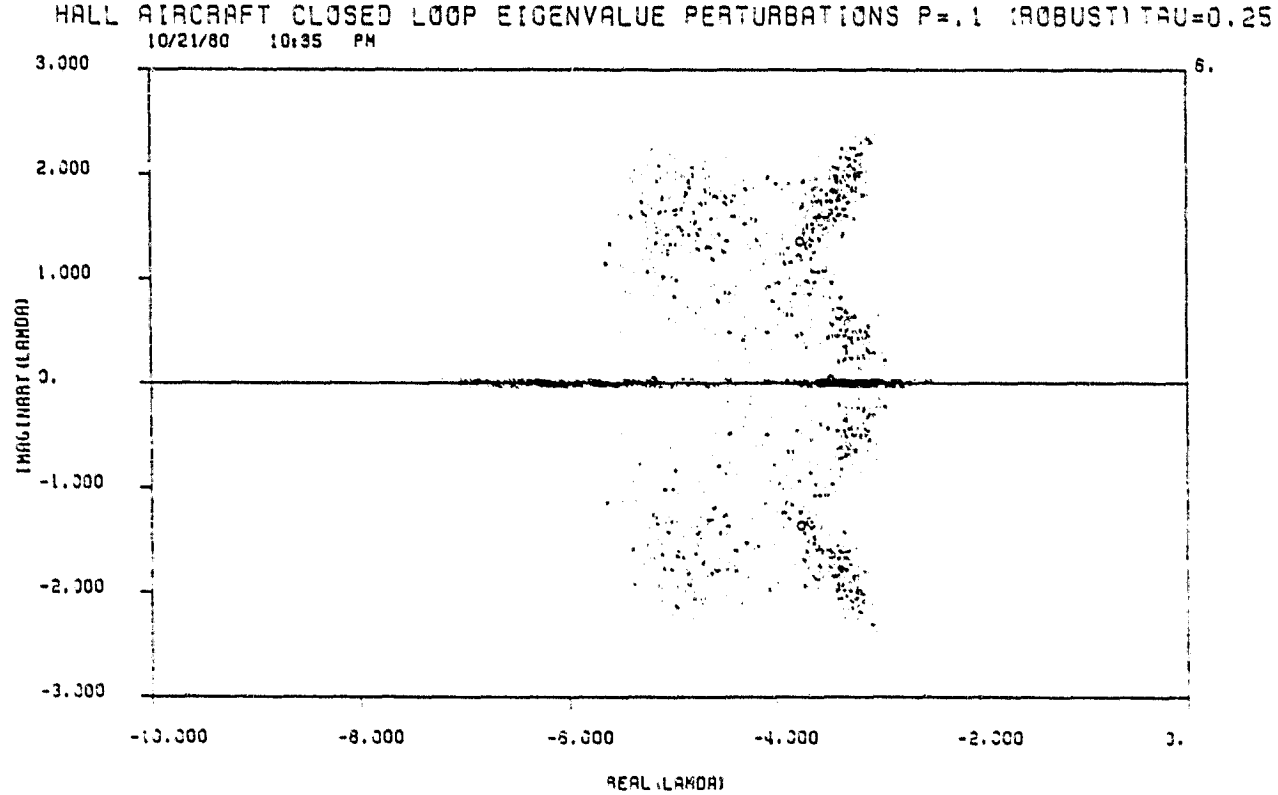
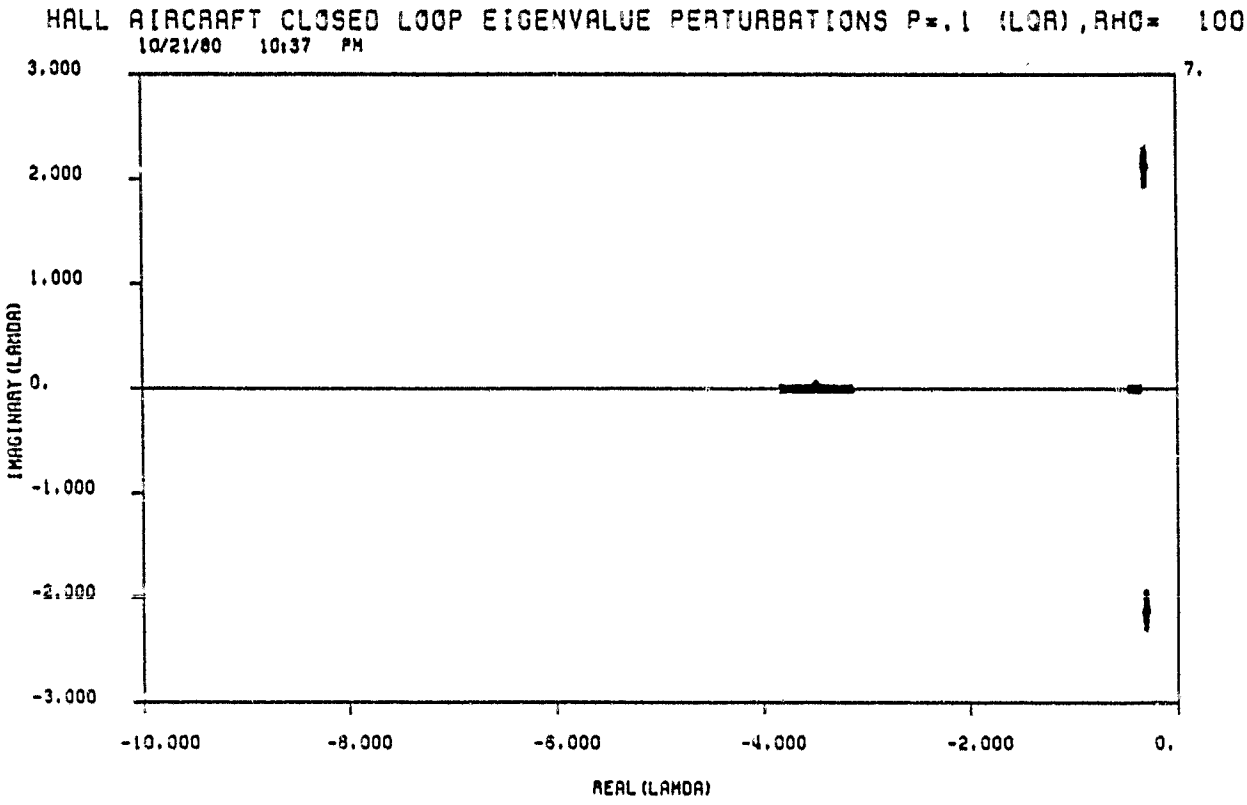

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (ROBUST) TAU=0.25
10/21/80 10:35 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
6.
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
REAL (LAMDA)

Figure 15


HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (LQR),RHO= 100
10/21/80 10:37 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
7.
REAL (LAMDA)

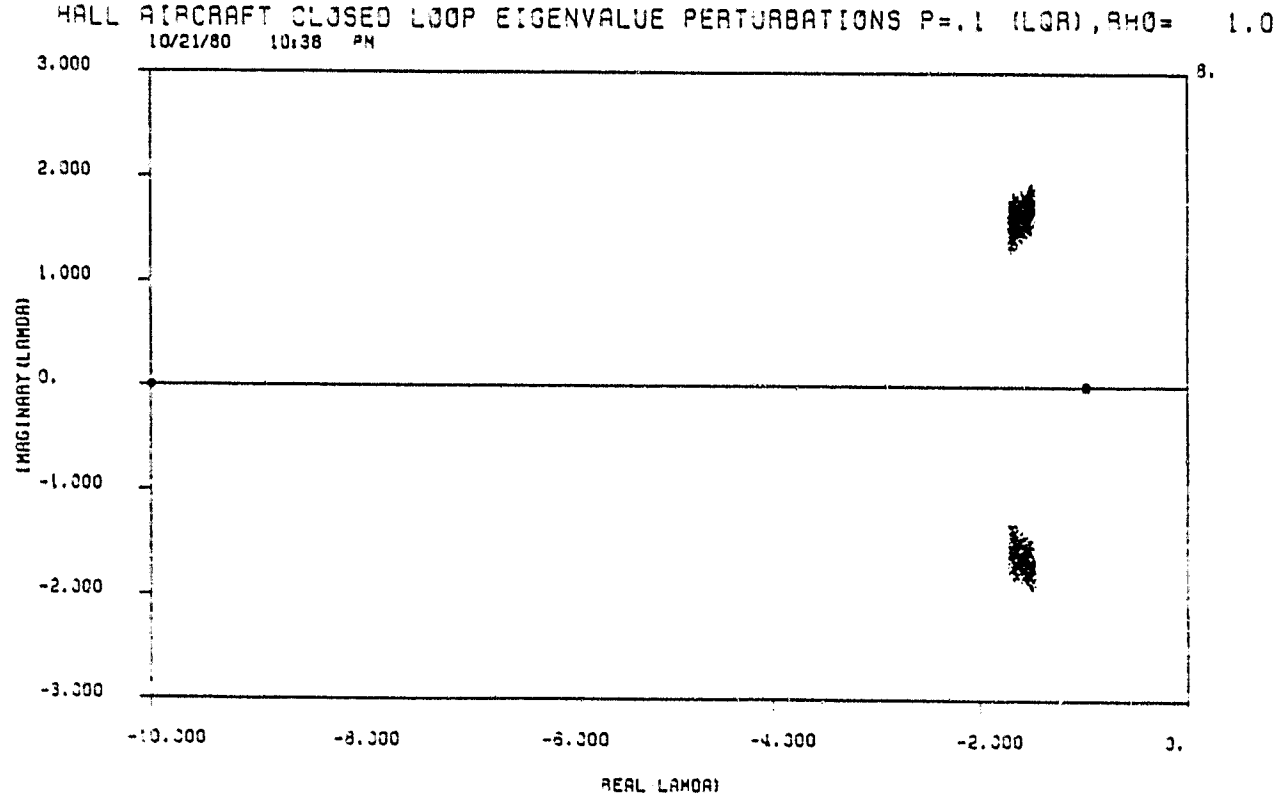

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (LQR),RHO= 1.0
10/21/80 10:38 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
8.
REAL LAMDA)

Figure 16

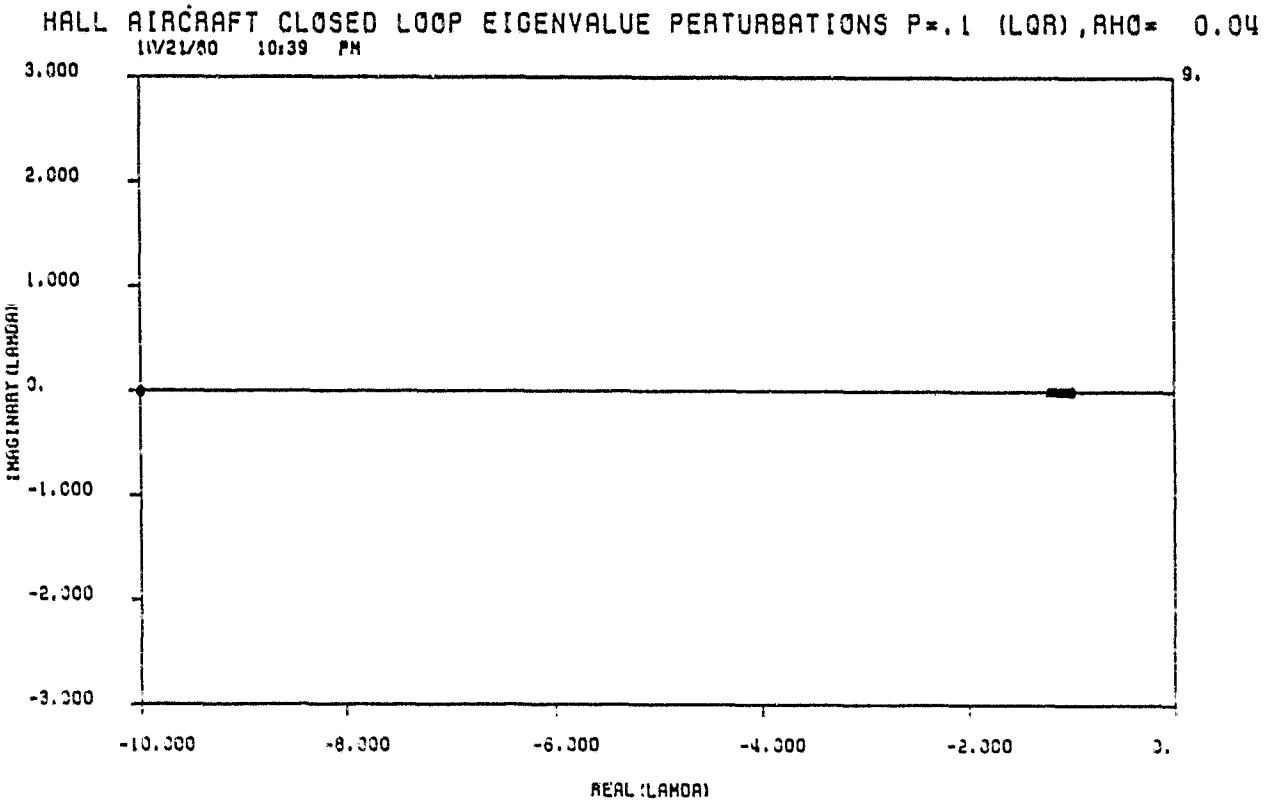

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P=.1 (LQR),RHO= 0.04
11/21/80 10:39 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
9.
REAL (LAMDA)

Table 6 Statistics on REMAX, REMIN and RTOMAX for Hall Aircraft

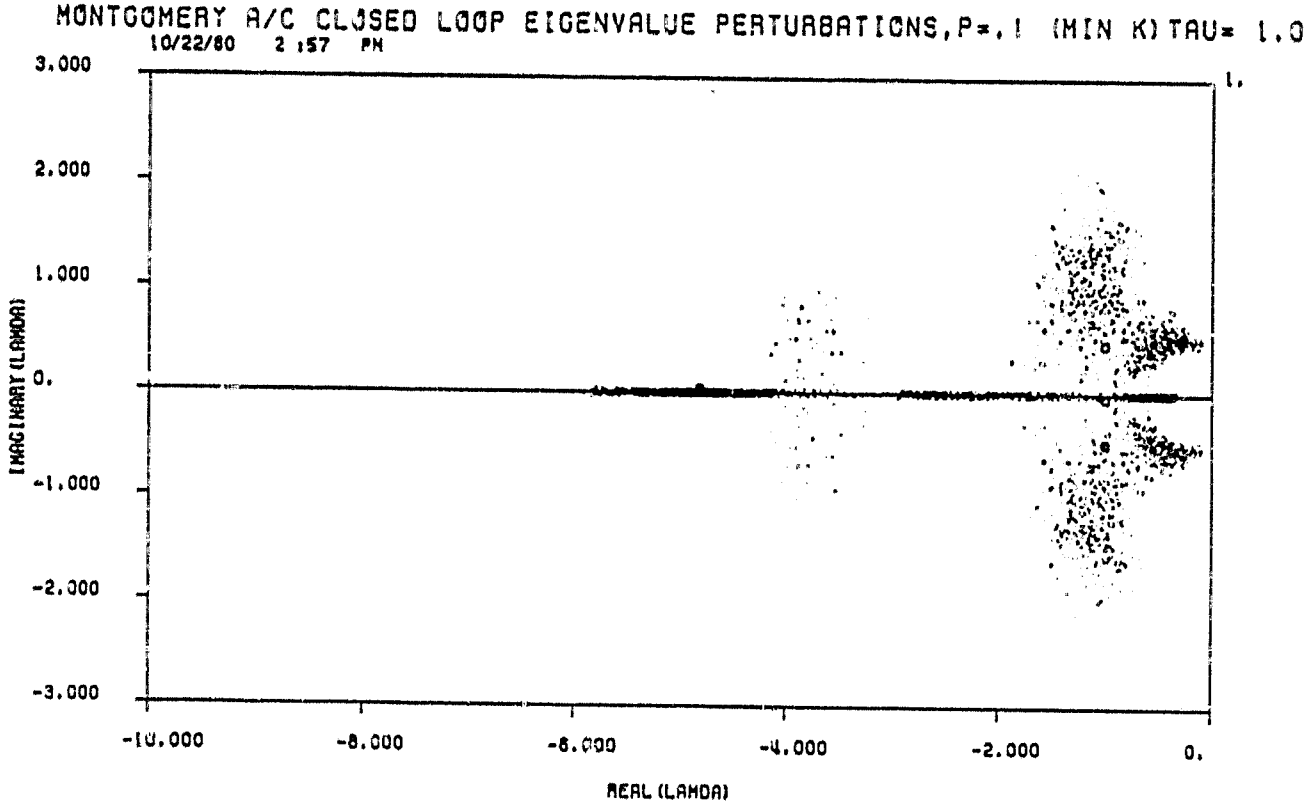
Controller	Parameter	Statistics on REMAX				Statistics on REMIN				Statistics on RTOMAX			
		Max	Min	Mean	S_D	Max	Min	Mean	S_D	Max	Min	Mean	S_D
Min K	τ = 1.0	-0.59	-1.74	-1.02	0.22	-1.79	-3.95	-2.76	0.43	0.75	0.0	0.36	0.18
	τ = 0.5	-1.25	-2.05	-1.70	0.16	-1.85	-3.53	-2.38	0.39	0.76	0.0	0.46	0.14
	τ = 0.25	-2.32	-4.30	-3.20	0.36	-5.38	-9.59	-7.55	0.88	0.47	0.0	0.24	0.11
Robust	τ = 1.0	-0.78	-1.49	-1.10	0.12	-2.41	-3.61	-3.00	0.23	1.14	0.60	0.85	0.09
	τ = 0.5	-1.26	-2.12	-1.80	0.17	-1.97	-3.59	-2.43	0.43	1.12	0.64	0.90	0.10
	τ = 0.25	-2.08	-4.13	-3.23	0.28	-3.73	-7.34	-5.37	0.80	0.80	0.0	0.41	0.14
LQR	ρ = 100	-0.28	-0.34	-0.31	0.01	-3.11	-3.85	-3.48	0.20	7.83	5.87	6.83	0.56
	ρ = 1	-0.94	-1.02	-0.98	0.02	-13.25	-16.26	-14.77	0.74	1.30	0.73	1.03	0.11
	ρ = 0.04	-0.96	-1.04	-1.01	0.01	-65.39	-79.69	-72.56	4.05	1.30	0.73	1.03	0.11

plots as their real parts are less than -10.

Figures 17 to 21 contain the scatter diagrams for the nine cases of the Montgomery aircraft. For both Min k and Robust controllers, the scatter of the perturbed eigenvalues continuously increases as τ decreases from 1.0 to 0.5 to 0.25. For any one of the three values of τ, both Min k and Robust controllers give nearly the same amount of scatter.

The scatter obtained for the LQR controller decreases as ρ decreases from 100 to 1 and then to 0.04. The scatter obtained for the LQR controller is significantly less than that obtained for Min k and Robust controllers.

Figure 17


MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.1 (MIN K) TAU= 1.0
10/22/80 2:57 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
1.
REAL (LAMDA)

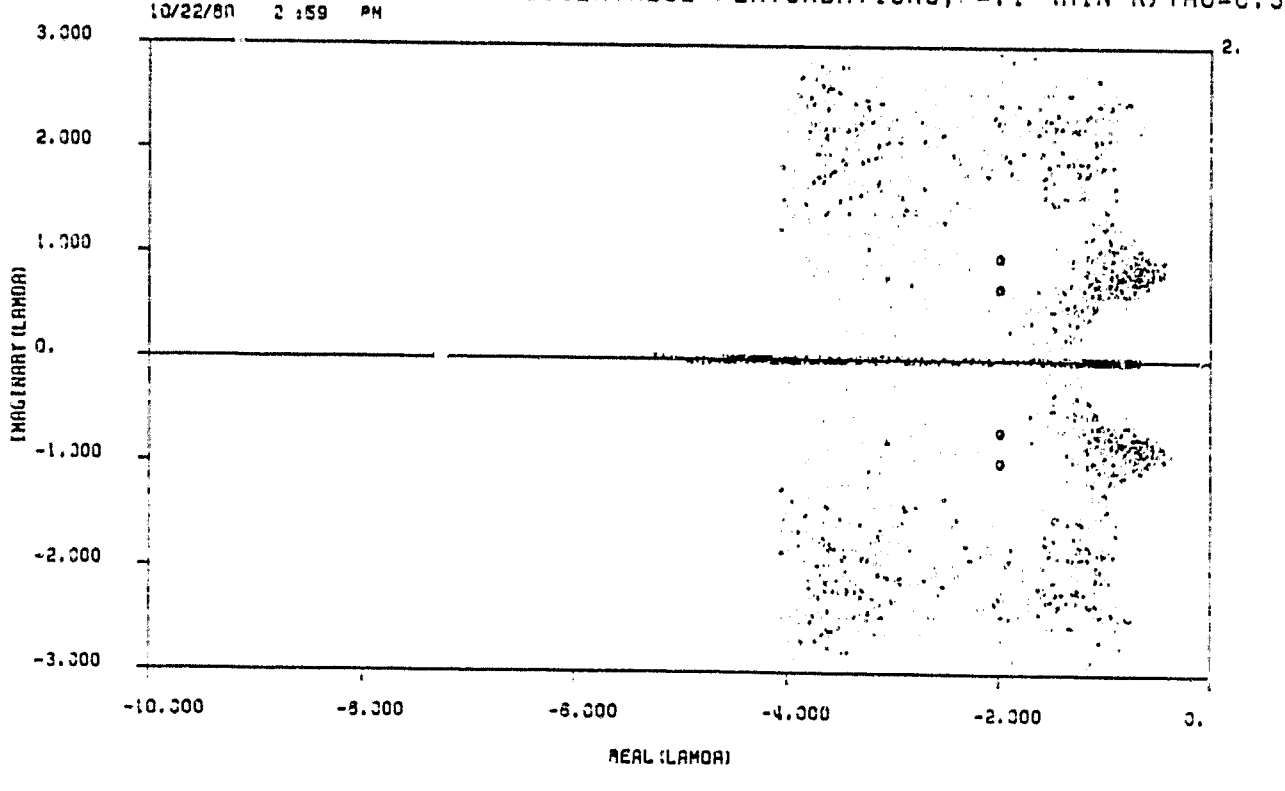
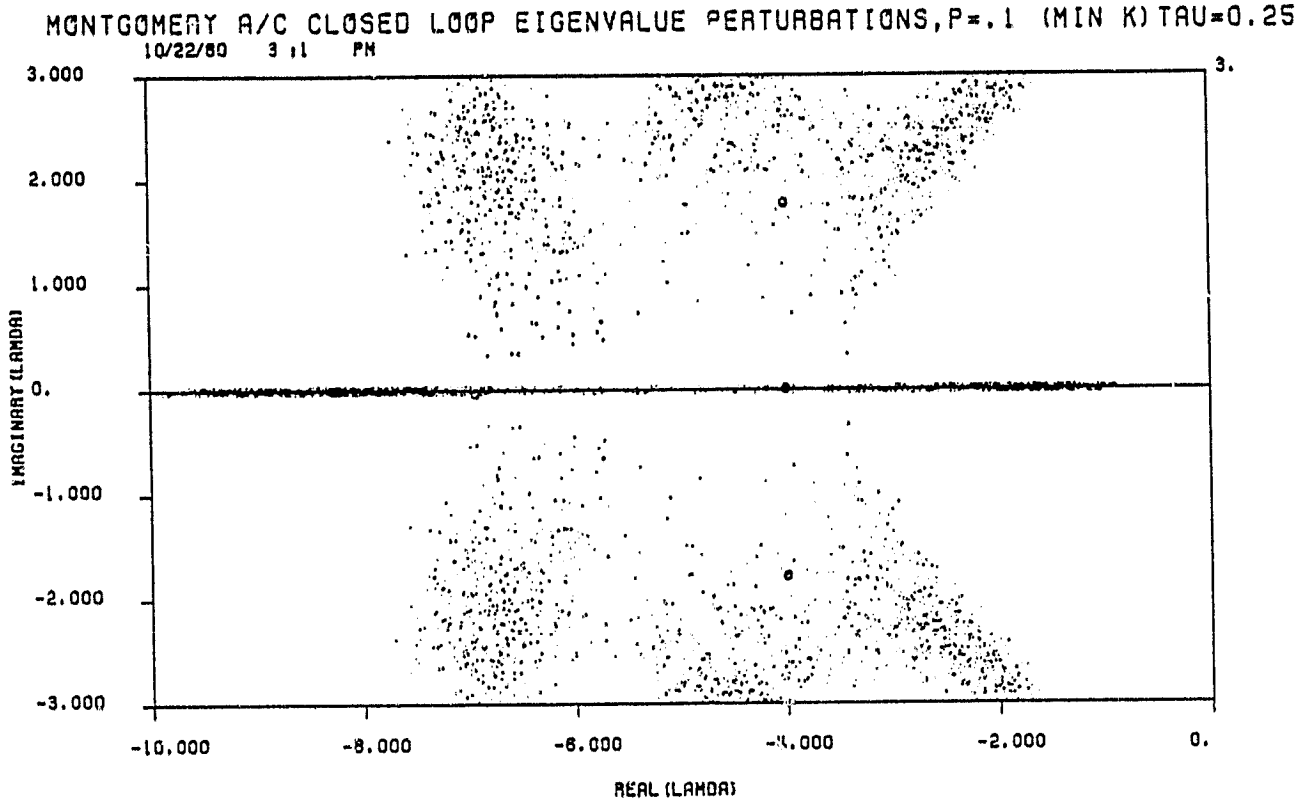

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.1 (MIN K) TAU=0.5
10/22/80 2:59 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
2.
REAL (LAMDA)

Figure 18


MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.1 (MIN K) TAU=0.25
10/22/80 3 :1 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
3.
REAL (LAMDA)

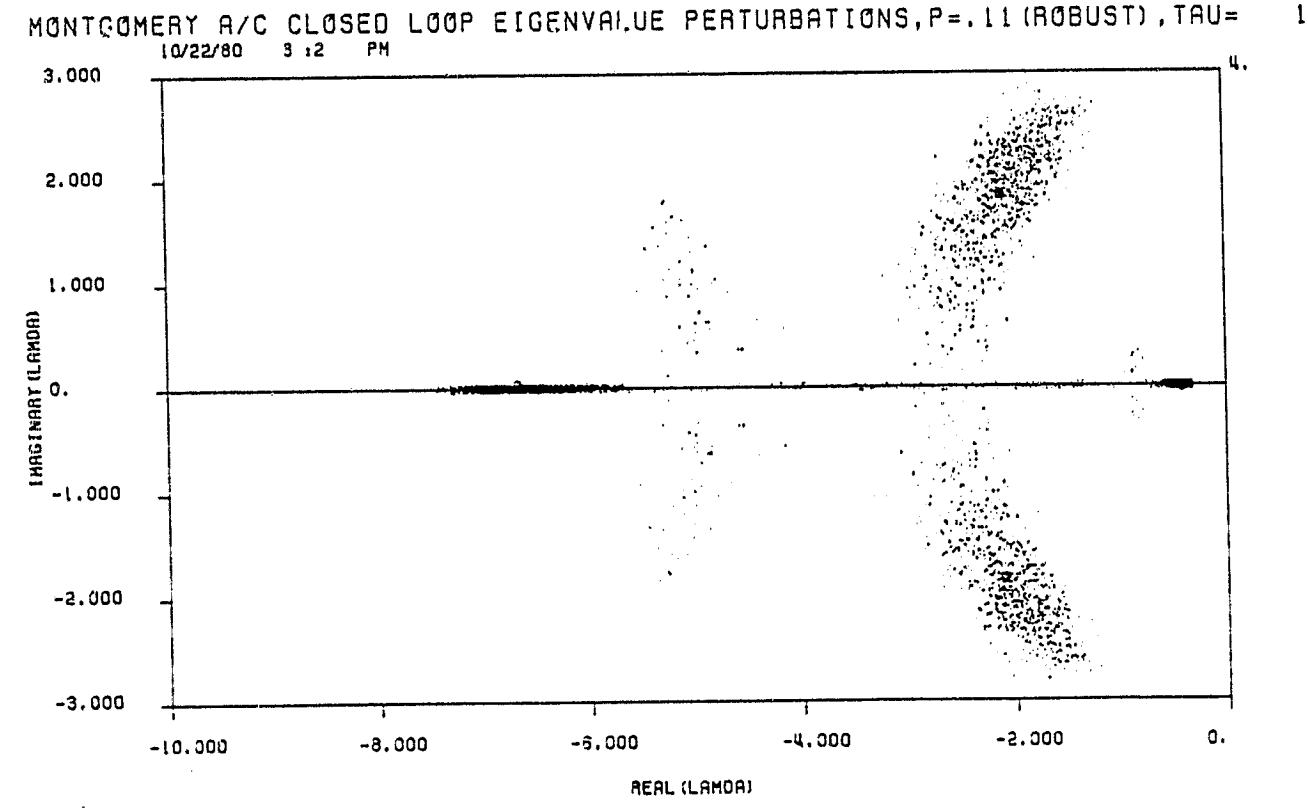
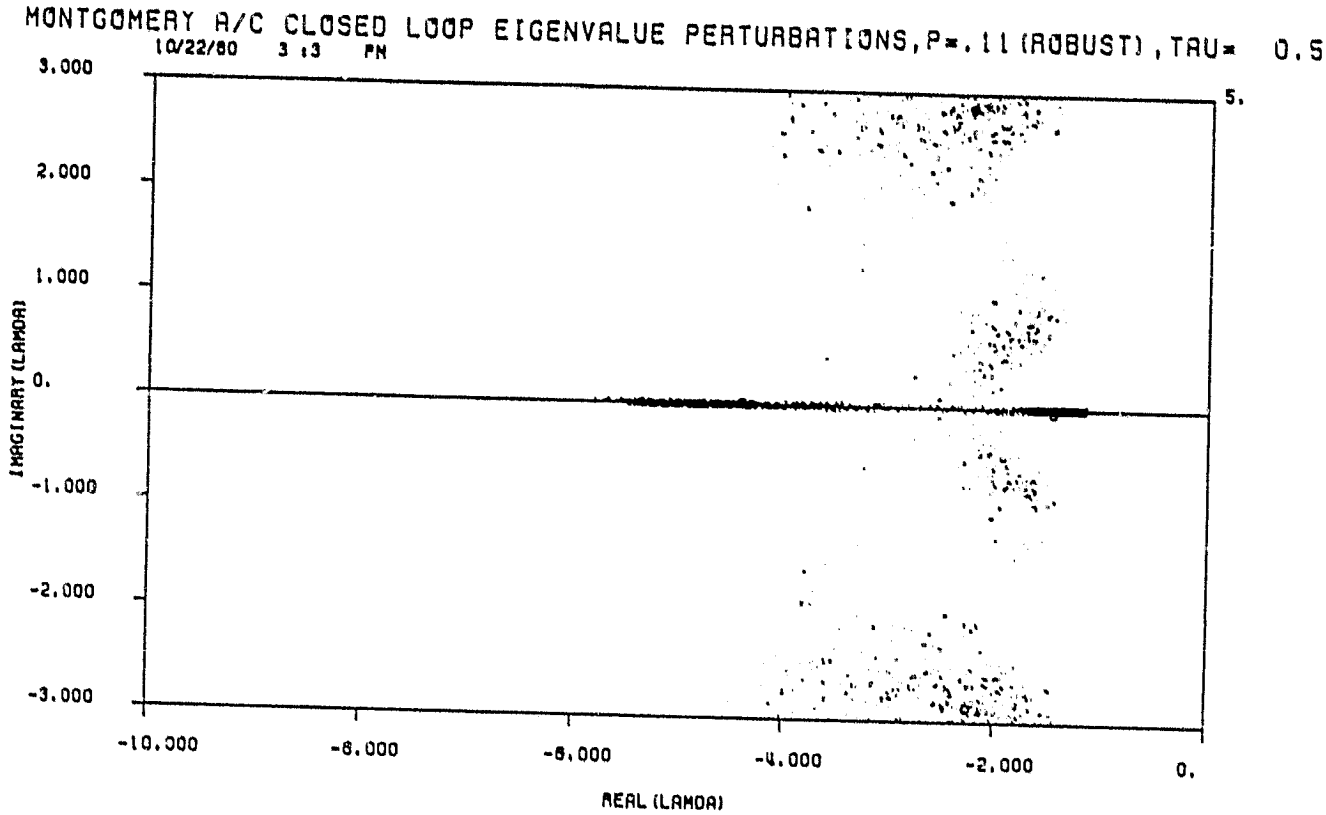

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.11 (ROBUST), TAU= 1.0
10/22/80 3 :2 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
4.
REAL (LAMDA)

Figure 19


MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.11 (ROBUST), TAU= 0.5
10/22/80 3:3 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
5.
REAL (LAMDA)

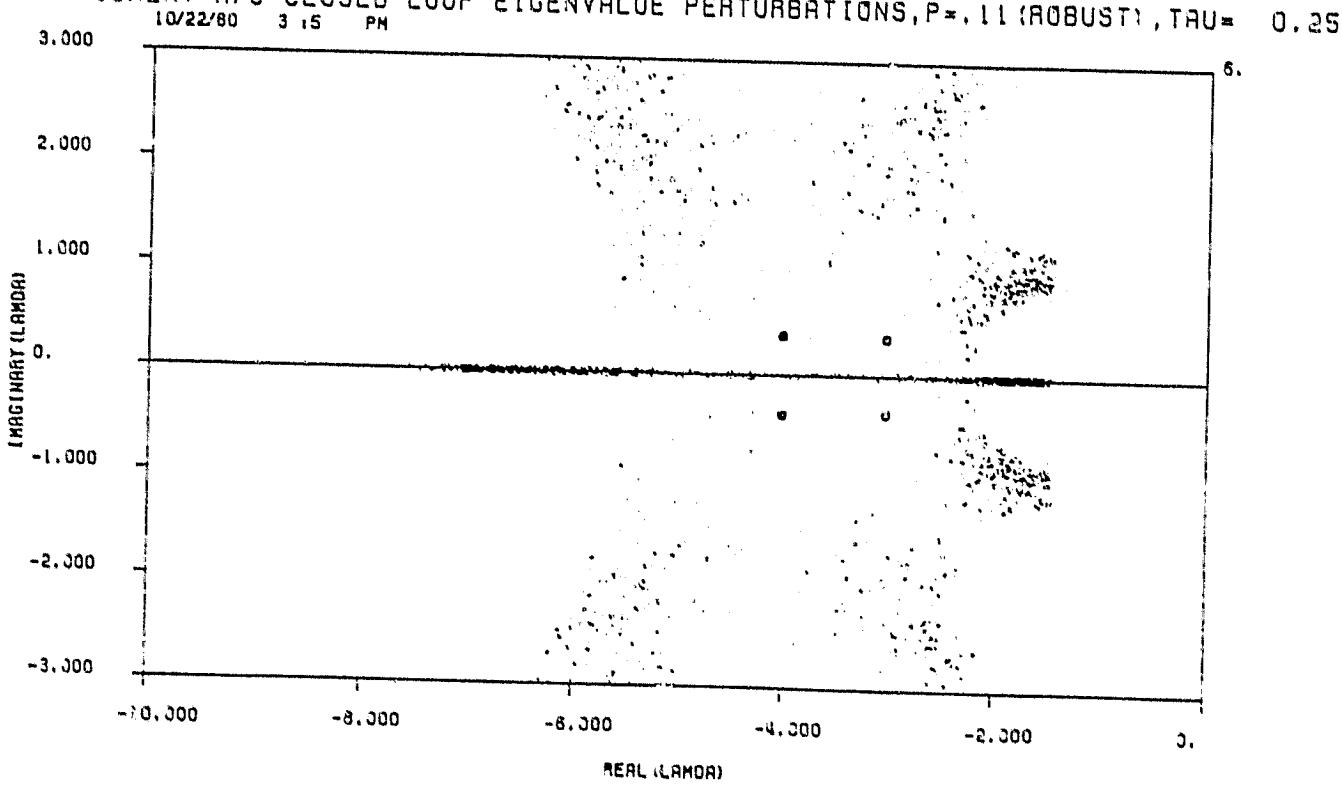
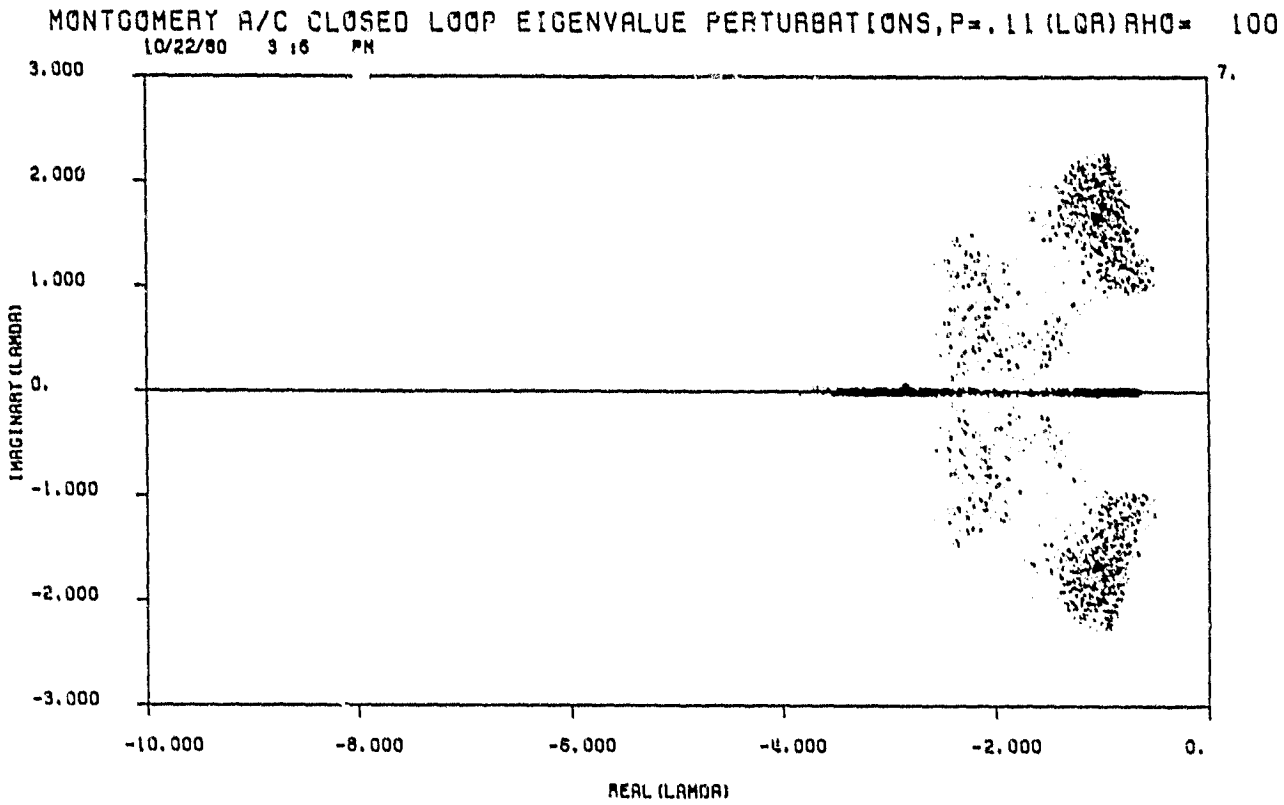

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.11 (ROBUST), TAU= 0.25
10/22/80 3:5 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-8.000
-6.000
-4.000
-2.000
6.
REAL (LAMDA)

Figure 20


MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.11 (LQR) RHO= 100
10/22/80 3:16 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
REAL (LAMDA)

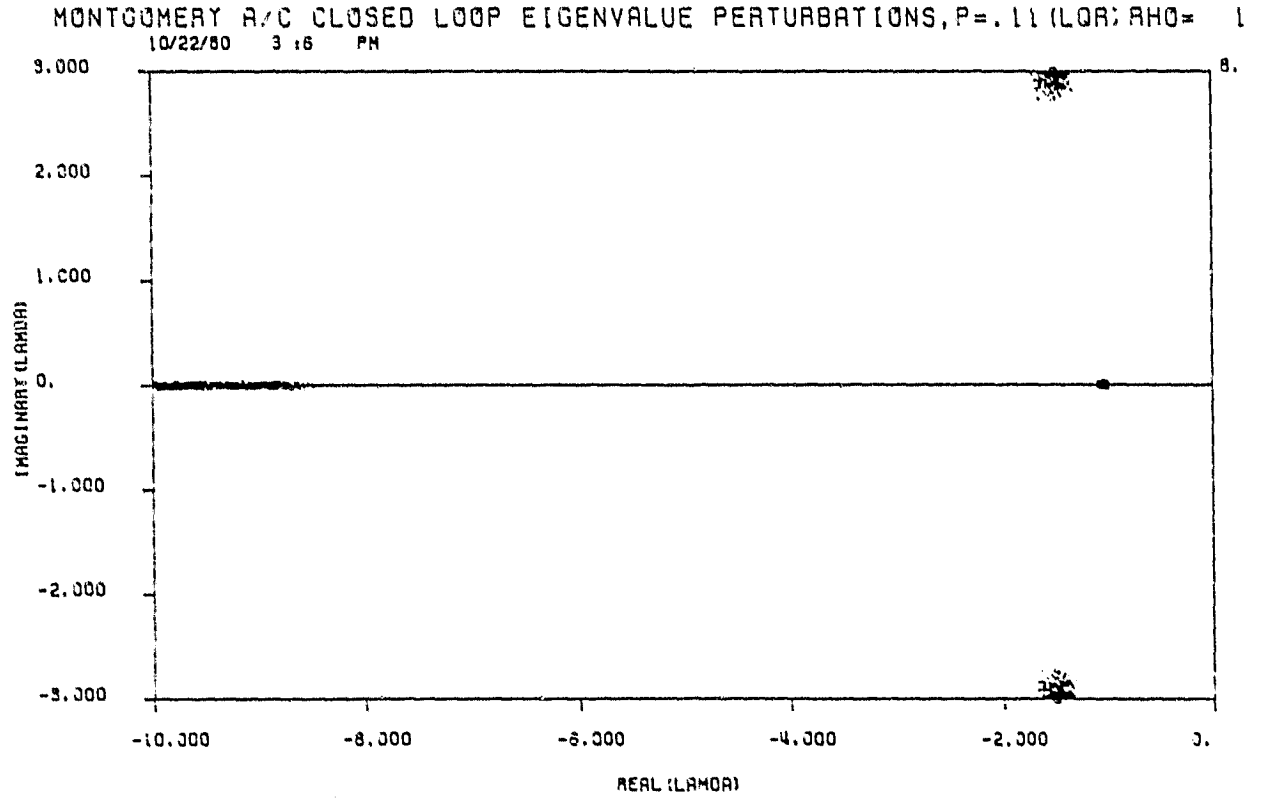

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.11 (LQR) RHO= 1
10/22/80 3:16 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
-10.000
-8.000
-6.000
-4.000
-2.000
0.
REAL (LAMDA)

Figure 20

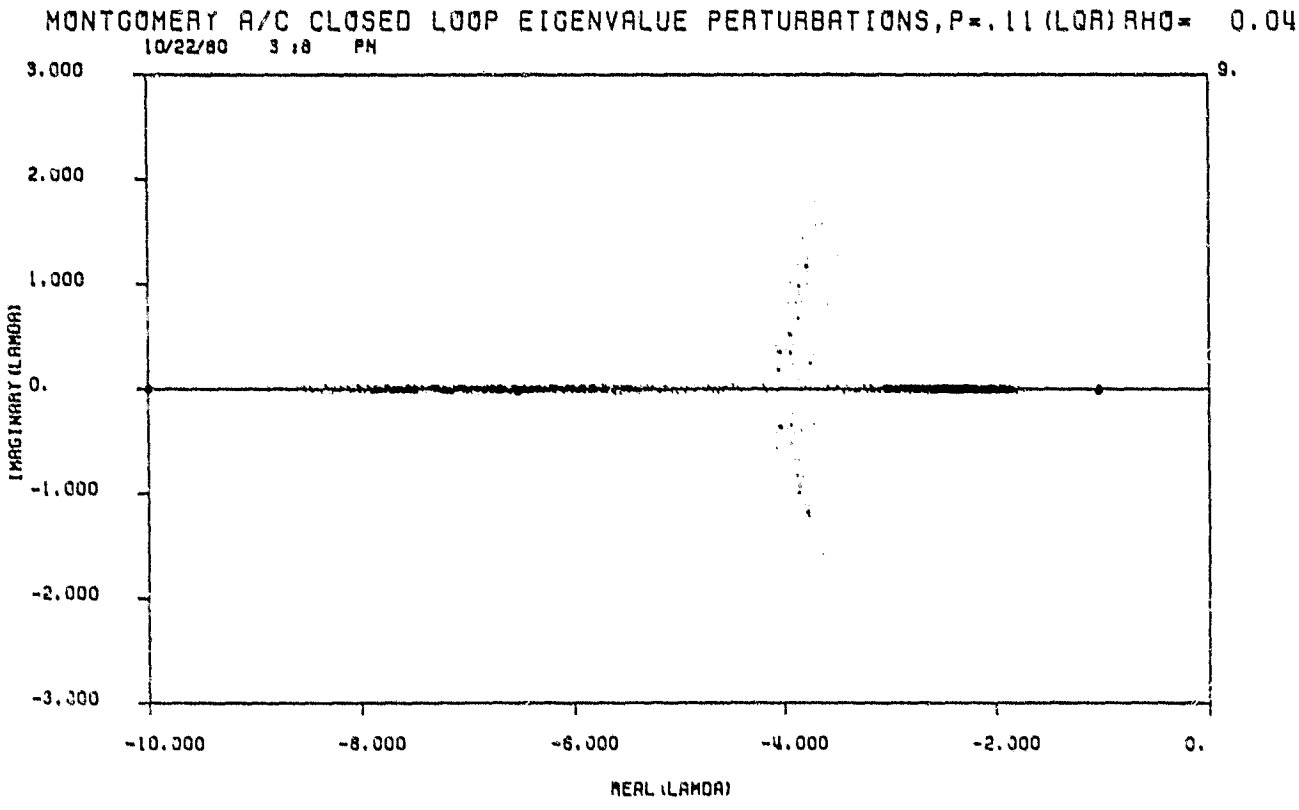

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P=.11 (LQR) RHO= 0.04
10/22/80 3 :8 PM
3.000
2.000
1.000
0.
-1.000
-2.000
-3.000
IMAGINARY (LAMDA)
9.
-10.000
-8.000
-6.000
-4.000
-2.000
0.
REAL (LAMDA)

Table 7 Statistics on REMAX, REMIN and RTOMAX for Montgomery Aircraft

Controller	Parameter	Statistics on REMAX				Statistics on REMIN				Statistics on RTOMAX			
		Max	Min	Mean	S_D	Max	Min	Mean	S_D	Max	Min	Mean	S_D
Min K	τ = 1.0	-0.09	-1.07	-0.51	0.16	-3.24	-5.98	-4.77	0.48	5.53	0.0	1.15	0.66
	τ = 0.5	-0.37	-1.85	-0.95	0.25	-1.87	-5.26	-3.52	0.69	4.35	0.35	1.32	0.55
	τ = 0.25	-0.74	-3.95	-2.30	0.68	-5.10	-10.16	-7.52	1.01	1.94	0.17	0.79	0.29
Robust	τ = 1.0	-0.31	-0.96	-0.44	0.08	-4.16	-7.45	-6.54	0.52	2.22	0.0	0.84	0.44
	τ = 0.5	-1.03	-2.78	-1.53	0.28	-2.61	-5.81	-4.36	0.70	3.40	0.46	1.34	0.49
	τ = 0.25	-1.34	-2.91	-1.90	0.28	-3.77	-7.56	-5.80	0.76	1.89	0.24	0.71	0.28
LQR	ρ = 100	-0.50	-1.55	-0.90	0.18	-1.46	-3.84	-2.71	0.53	2.53	0.71	1.64	0.35
	ρ = 1	-0.99	-1.10	-1.04	0.02	-8.47	-10.62	-9.54	0.52	2.62	1.67	2.06	0.16
	ρ = 0.04	-1.02	-1.06	-1.04	0.01	-38.09	-46.18	-42.08	2.09	2.62	1.67	2.06	0.16

X. Conclusions and Recommendations

Based on the results of this preliminary study, it is concluded that:

1) Searching directly on the nm elements of the K matrix using nonlinear programming with penalty terms to impose the closed-loop eigenvalue constraints

$$\lambda(A + BK) = \Gamma$$

is a feasible, but not highly attractive method of control system design. Feedback gain matrices obtained by this method do not appear to have any advantages over LQR controllers, at least for these design examples. The LQR controllers seem to be naturally more robust than those obtained from direct pole placement.

2) The hoped for increase in system robustness through orthogonalizing closed-loop eigenvalues was not verified in this study. As shown in Chapter II and Appendix E, there is a proven mathematical relationship between eigenvector orthogonality and eigenvalue sensitivity, but this relationship was not clearly demonstrated for these design examples.

In this continuing research effort to find new and better methods for robust control system design, it is recommended to proceed in the following directions:

1) Consider improving robustness only for the class of LQR controllers. Perhaps this can be best achieved by searching directly on the diagonal elements of the Q and R matrices via nonlinear programming to maximize orthogonality of closed-loop eigenvectors.

2) Test the controller design methods on a wider variety of mechanical systems to evaluate controller characteristics. Certain classes of problems (such as linearized aircraft equations of motion)

may have unique system and control matrix eigenstructures which do not provide a sufficiently general test of the design methods. Other prototype design examples which could be considered include:

a) wing flutter suppression in high-speed aircraft

b) throttle control of multivariable turbofan engines

c) mathematical examples containing random system and control matrices to establish general mathematical properties.

```
C    FILE NEWSOM FORTRAN
C
C    JULY 22, 1980
C
      REAL*8 DSEED
      COMMON/ADATA/N,M,A(4,4),B(4,3),X(12)
      COMMON/BDATA/NVARS,REMIN,REMAX,RTOMAX,WT1,WT2,WT3,WT4,WT5
      COMMON/IPRINT/IPRINT
      COMMON/NNOUT/NODEV
      COMMON/ISEED/ISEED
      NAMELIST/NSOMIN/N,M,EPS,MAXFN,NLOOPS,NSRCH,NODEV,IPRINT,ICONT,IOUT
     1  ,REMIN,REMAX,RTOMAX,WT1,WT2,WT3,WT4,WT5,ISKIP,A,B,X
      NAMELIST/NSMOUT/N,M,A,B,X
C
C     INPUTS
C     N=DIMENSION OF THE STATE
C     M=DIMENSION OF CONTROL
C     EPS=CONVERGENCE TOLERANCE OF THE POWELL ITERATION
C     NLOOPS=MAX NO OF POWELL ITERATION LOOPS
C     NSRCH=MAX NO OF POINTS ON A LINE SEARCH
C     NODEV=SPECIFIES THE OUTPUT DEVICE(6=TERMINAL,8=FILE NEWSOM OUTPUT)
C     IPRINT=CONTROLS THE AMOUNT OF PRINTOUT. 1=NORMAL PRINTOUT
C     ICONT=1,TO CONTINUE AFTER THIS CASE
C           =0,TO STOP AFTER THIS CASE
C     IOUT=1,TO WRITE NAMELIST BLOCK IN OUTPUT
C        0,NOT TO DO THIS
C     WT1...WT5 =WEIGHTS USED IN COST FUNCTION
C     REMIN=LEFT BOUNDARY OF E-VALUE DOMAIN
C     REMAX=RIGHT BOUNDARY OF E-VALUE DOMAIN
C     RTOMAX=MAXIMUM RATIO IM(LAMDA)/RE(LAMDA)
C     A=N*N SYSTEM MATRIX
C     B=N*M CONTROL MATRIX
C     X=M*N INITIAL FEED BACK MATRIX
C
      DATA NCASE/0/
      CALL ERRSET(208,256,-1,1)
      ISEED = 566387
5  (1) READ(3,NSOMIN)
      NCASE = NCASE + 1
      IF(ISKIP.GE.1.AND.ICONT.LE.0)STOP
      IF(ISKIP.GE.1)GO TO 5
      WRITE(NODEV,100)NCASE
100   FORMAT(//,' **********   PROGRAM NEWSOM   **********   CASE',I3)
      CALL DATIME(IM,ID,IY,IH,IMN,AP,'NEWSOM  ',NODEV)
C     ISEED = IH*100 + IMN
  (2) IF(IPRINT.GE.3)WRITE(NODEV,NSOMIN)
      NVARS = N*M
  (3) IF(X(1).NE.-1.)GO TO 8
      CALL RANDS(ISEED,NVARS,X)
      WRITE(NODEV,101)ISEED,(X(I),I=1,NVARS)
101   FORMAT(/,' GGURS CALLED WITH ISEED =',I20,
     1      /,' RAND NOS =',10F7.4,/,(11X,10F7.4))
      DO 6 I = 1,NVARS
6     X(I) = 2.*X(I) - 1.
8  (4) IF(IPRINT.GE.1)CALL WRTMAT(A,N,N,4,'A       ')
```

```
      IF(IPRINT.GE.1)CALL WRTMAT(B,N,M,4,'B       ')
      IPRINT = IPRINT + 8
(5)   FMIN = COSTF(X)
      IPRINT = IPRINT - 8
(6)   CALL POWELL(NVARS,X,FMIN,EPS,IPRINT,MAXFN,NLOOPS,NSRCH)
      IPRINT = IPRINT + 8
(7)   FMIN = COSTF(X)
      IPRINT = IPRINT - 8
(8)   IF(IOUT.GE.1)WRITE(7,NSMOUT)
      IF(ICONT.GE.1)GO TO 5
      STOP
      END
C
      FUNCTION COSTF(X)
      DIMENSION X(1),D1(5,5),D2(5,5),D3(5,6),D4(5)
      COMPLEX EIG(5),EVEC(5,5)
      COMMON/ADATA/N,M,A(4,4),B(4,3),XX(12)
      COMMON/BDATA/NVARS,REMIN,REMAX,RTOMAX,WT1,WT2,WT3,WT4,WT5
      COMMON/IPRINT/IPRINT
      COMMON/NCOST/NCOST
      COMMON/NNOUT/NODEV
      COMMON/EIGDAT/ER(5),EI(5)
      NCOST = NCOST + 1
      F1 = 0.
      F2 = 0.
      F3 = 0.
      F4 = 0.
      F5 = 0.
      DO 10 I=1,NVARS
10    F1 = F1 + X(I)**2
      F1 = WT1*SQRT(F1)
      CALL MULT(B,X,D1,N,M,N,4,M,5)
      CALL MATADD(A,D1,D1,N,N,4,5,5)
C      DO 12 I = 1,N
C      DO 12 J = 1,N
C12    D1(I,J) = X(I)*A(I,J)/X(J)
      IF(IPRINT.LE.8)GO TO 15
      CALL WRTMAT(X,M,N,M,'FEEDBACK')
      CALL WRTMAT(D1,N,N,5,'A+B*K   ')
15    IJOB = 1
      CALL EIGRF(D1,N,5,IJOB,EIG,EVEC,5,D2,IER)
      IF(IER.GT.0)WRITE(NODEV,100)IER
100   FORMAT(/' ERROR IN EIGRF IN COSTF   IER =',I3)
      DO 30 I=1,N
      ER(I) = REAL(EIG(I))
      EI(I) = AIMAG(EIG(I))
      IF(ER(I).LT.REMIN)F2 = F2 + WT2*(ER(I)-REMIN)**2
      IF(ER(I).GT.REMAX)F3 = F3 + WT3*(ER(I)-REMAX)**2
      RATIO = ABS(EI(I)/ER(I))
      IF(RATIO.GT.RTOMAX)F4 = F4 + WT4*(RATIO-RTOMAX)**2
      TEMP = 0.
      DO 20 J = 1,N
      D1(J,I) = REAL(EVEC(J,I))
      D2(I,J) = 0.
      IF(EI(I).LT.0.)D1(J,I) = AIMAG(EVEC(J,I))
```

```
      IF(I.EQ.J)D2(I,I) = 1.
20    TEMP = TEMP + D1(J,I)**2
30    D4(I) = SQRT(TEMP)
      DO 38 I = 2,N
      I1 = I - 1
      DO 36 J = 1,I1
      TEMP = 0.
      DO 34 K = 1,N
34    TEMP = TEMP + D1(K,I)*D1(K,J)
      ARG = TEMP/(D4(I)*D4(J))
      D3(I,J) = 57.2958*ACOS(AMIN1(ABS(ARG),1.))
36    F5 = F5 + (90.-D3(I,J))**10
      IF(IPRINT.GE.5)
     1WRITE(NODEV,104)(I,J,D3(I,J),J=1,I1)
104   FORMAT(/,5(' ANG',I1,',',I1,' =',F7.2))
38    CONTINUE
      F5 = WT5*(F5**.1)
      IF(IPRINT.LT.5)GO TO 60
      IDGT = 0
      CALL LEQT2F(D1,N,N,5,D2,IDGT,D3,IER)
      IF(IER.GT.0)WRITE(NODEV,105)IER
105   FORMAT(/,' ERROR IN LEQT2F IN COSTF   IER =',I6)
      D1N = 0.
      D2N = 0.
      ERR = 0.
      DO 50 I = 1,N
      DO 50 J = 1,N
      TEMP = 0.
      DO 45 K = 1,N
45    TEMP = TEMP + D1(I,K)*D2(K,J)
      IF(I.EQ.J)TEMP = TEMP - 1.
      ERR = ERR + TEMP**2
      D1N = D1N + D1(I,J)**2
50    D2N = D2N + D2(I,J)**2
      ERR = SQRT(ERR)
      D1N = SQRT(D1N)
      D2N = SQRT(D2N)
60    F = F1 + F2 + F3 + F4 + F5
      COSTF = F
      IF(IPRINT.LT.5)RETURN
      WRITE(NODEV,106)ERR,D1N,D2N
106   FORMAT(/,' IN COSTF   ERR, D1N, D2N =',1P3E12.4)
      WRITE(NODEV,108)REMIN,REMAX,RTOMAX
108   FORMAT(/,' REMIN =',G10.3,'  REMAX =',G10.3,'  RTOMAX =',G10.3)
      WRITE(NODEV,107)WT1,WT2,WT3,WT4,WT5
107   FORMAT(/,' WT1,WT2,WT3,WT4,WT5 =',5G12.5)
      WRITE(NODEV,101)F1,F2,F3,F4,F5,F
101   FORMAT(' F1,F2,F3,F4,F5,FTOT =',6G12.4)
      WRITE(NODEV,102)(ER(I),I=1,N)
102   FORMAT(/,' REAL EIG =',5G12.4)
      WRITE(NODEV,103)(EI(I),I=1,N)
103   FORMAT(' IMAG EIG =',5G12.4)
      COSTF = F
      RETURN
      END
```

```
C
C**********   SUBROUTINE DATIME   **********
C
      SUBROUTINE DATIME(IMO,IDAY,IYR,IHOURS,IMIN,AMPM,PNAME,NODEV)
      REAL*8 PNAME
      DATA AM/'  AM'/,PM/'  PM'/
      CALL DATE(IMO,IDAY,IYR)
      CALL STIME(ITIME)
      XHOURS = FLOAT(ITIME)/10000.
      AMPM = AM
      IF(XHOURS.GE.12.)AMPM = PM
      IF(XHOURS.GE.13.)XHOURS = XHOURS - 12.
      IHOURS = XHOURS
5     XMIN = (XHOURS - IHOURS)*60.
      IMIN = XMIN
      IF(NODEV.GT.0)WRITE(NODEV,100)PNAME,IMO,IDAY,IYR,IHOURS,IMIN,AMPM
100   FORMAT(/' TIME IN ',A8,' IS ',A2,'/',A2,'/',A2,5X,I2,':',
     1   I2,3X,A4)
      RETURN
      END
C
C   **********   SUBROUTINE RANDS   **********
C
      SUBROUTINE RANDS(IX,N,Z)
      DIMENSION Z(1)
      DATA M/1048576/,FM/1048576./,IA/1027/
      DO 10 I = 1,N
      IX = MOD(IA*IX,M)
      FX = IX
10    Z(I) = FX/FM
      RETURN
      END
C       FILE POWELL FORTRAN
C
        SUBROUTINE POWELL(N,X,F,EPS,IPRINT,MAXFN,NLOOPS,NSRCH)
        DIMENSION X(1),DIST(26),INDX(25)
        COMMON/POWEL/NN,D(26,26),P(26,26),NPCOL,NDCOL
      COMMON/ISEED/ISEED
        COMMON/OUTPUT/INFO
        COMMON/NCOST/NCOST
        COMMON/IMINPT/LIT,ITO,ITI,ITP,IGO,NONU
        COMMON/NNOUT/NODEV
        DATA DETMIN/.01/
      LIMIT = NLOOPS
      LINS = NSRCH
      CALL TIMEON
        INFO = IPRINT
        NCOST = 0
      NONU = 0
        NN = N
        IT = 1
        N1 = N + 1
        N2 = 1.5*N
        F = COSTF(X)
        F1 = F
```

```
      FOLD = F
      IF(INFO.GE.1)WRITE(NODEV,103)N,LIMIT,LIMS,EPS,F,(X(I),I=1,N)
103   FORMAT(/' POWELL ITERATION',
     1   /' N,LIMIT,LIMS=',3I3,', EPS=',G12.4,
     1   /' F =',G14.7,
     1   /' X=',5F14.4,/(3X,5F14.4))
      IF(INFO.GE.1)WRITE(NODEV,104)
104   FORMAT(' IT IS LIT ITO ITI ITP IGO NONU NCOST',2X,'COST' 9X,
     1    'DIST',10X,'ERR')
C     INITIALIZE SEARCH DIRECTIONS
      DO 1 I=1,N
      P(I,1) = X(I)
1     DIST(I) = 1.
     DIST(N1) = 1.
2     NIT = 1
      DET = 1.
      DO 10 I=1,N
      DO 5 J=1,N
5     D(J,I) = 0.
10    D(I,I) = 1.
      GO TO 15
12    WRITE(NODEV,105)I,DET,DMAX,ALF
105   FORMAT(' DN1 REJ  I,DET,DMAX,ALF=',I3,1P3E14.7)
C     SEARCH IN N DIRECTIONS
15    DMAX = 0.
C   RANDOMIZE THE ORDER OF CHOOSING THE N SEARCH DIRECTIONS
     CALL RANIND(N,INDX)
     IF(IPRINT.GE.3)WRITE(NODEV,107)ISEED,(INDX(I),I=1,N)
107  FORMAT(' ISEED,INDX =',I11,25I3)
      DO 20 II=1,N
     I = INDX(II)
      NPCOL = II
      NDCOL = I
      CALL MINPT(DIST(I),F,EPS,LIMS)
      IF(INFO.GE.2)
    1WRITE(NODEV,100)I,LIT,ITO,ITI,ITP,IGO,NONU,NCOST,F,DIST(I)
100   FORMAT(1X,I5,5I4,I5,I6,2G13.6)
     IF(INFO.GE.2)NONU = 0
      IF(ABS(DIST(I)).LE.DMAX)GO TO 18
      DMAX = ABS(DIST(I))
      IS = I
18    DO 20 J=1,N
20    P(J,II+1) = P(J,II) + DIST(I)*D(J,I)
C     FIND NEW SEARCH DIRECTION
      ALF = 0.
      DO 30 J=1,N
      D(J,N1) = P(J,N1) - P(J,1)
30    ALF = ALF + D(J,N1)**2
      ALF = SQRT(ALF)
      IF(ALF.EQ.0.)GO TO 90
C     NORMALIZE NEW SEARCH DIRECTION
      DO 40 J=1,N
40    D(J,N1) = D(J,N1)/ALF
      F3 = F2
      F2 = F1
```

```
        F1 = F
        NPCOL = N1
        NDCOL = N1
      LIMS = 2*NSRCH
        CALL MINPT(DIST(N1),F,EPS,LIMS)
      LIMS = NSRCH
        ERR = AMIN1(ABS((F-F3)/F),ABS(F))
        DO 50 J=1,N
        P(J,1) = P(J,N1) + DIST(N1)*D(J,N1)
50      X(J) = P(J,1)
        IF(INFO.GE.1)
     1WRITE(NODEV,101)IT,IS,LIT,ITO,ITI,ITP,IGO,NONU,NCOST,F,DIST(N1),
     1      ERR
101     FORMAT(1X,I2,I3,5I4,I5,I6,3G13.6)
      IF(INFO.GE.1)NONU = 0
        IF(INFO.GE.4)WRITE(NODEV,102)(X(K),K=1,N)
102     FORMAT(' X =',1P5E13.6,(3X,5E13.6))
        IF(ERR.LE.EPS.AND.IT.GE.2)GO TO 90
        IF(IT.GE.LIMIT)GO TO 90
      IF(NCOST.GE.MAXFN)GO TO 90
        IT = IT + 1
        IF(NIT.GE.N2)GO TO 2
        NIT = NIT + 1
C       CHECK TO SEE IF THE NEW SEARCH DIRECTION SHOULD BE USED
        DETN = DET*DMAX/ALF
        IF(DETN.LT.DETMIN)GO TO 12
        DO 60 J=1,N
60      D(J,IS) = D(J,N1)
        DIST(IS) = DIST(N1)
        DET = DETN
        GO TO 15
90    CONTINUE
      CALL TIMECK(NTIME)
      TIME = NTIME/100.
        WRITE(NODEV,106)IT,NCOST,FOLD,F,TIME,(X(I),I=1,N)
106     FORMAT(/' SEARCH COMPLETE AFTER',I3,' ITERATIONS AND',I5,
     1' FUNCTION CALLS.',/' INITIAL COST =',1PE14.7,', FINAL COST =',
     1 E14.7,'    TIME = ',0PF10.5,//' X =',1P5E14.7,/(4X,5E14.7))
      RETURN
        END
C
      SUBROUTINE COSTD(DIST,COST,DMIN,CMIN)
        COMMON/POWEL/NN,D(26,26),P(26,26),NPCOL,NDCOL
        COMMON/IMINPT/LIT,ITO,ITI,ITP,IGO,NONU
        COMMON/OUTPUT/INFO
      COMMON/NNOUT/NODEV
        DIMENSION X(25)
        DO 10 I=1,NN
10      X(I) = P(I,NPCOL) + DIST*D(I,NDCOL)
        COST = COSTF(X)
      IF(COST.GE.CMIN)GO TO 20
      CMIN = COST
      DMIN = DIST
20      IF(INFO.GE.4)WRITE(NODEV,100)IGO,DIST,COST,DMIN,CMIN
100   FORMAT(' IGO=',I1,' DIST=',G10.3,' COST=',G14.7,
```

```
     1     ' DMIN=',G10.3,' CMIN=',G14.7)
           RETURN
           END
C
           SUBROUTINE MINPT(DMIN,CMIN,EPS,LIMIT)
           COMMON/IMINPT/IT,ITO,ITI,ITP,IGO,NONU
          COMMON/NNOUT/NODEV
          COMMON/IPRINT/IPRINT
           DATA BIG,DTAU/1.E10,.1180339886/
           DATA DISTMN/1.E-6/,VBIG/1.E20/
          D = DMIN
          DZERO = 0.
          CMIN = VBIG
          IGO = 3
          CALL COSTD(DZERO,C,DMIN,CMIN)
           C0 = C
           IF(ABS(D).LT.DISTMN)D = DISTMN
           DELTA = ABS(D)/2.
           D2 = 0.
           C2 = C
           SIDE = SIGN(1.,D)
C
C      IT = NO. OF POINTS IN LINE SEARCH
C      ITP = NO. OF PARABOLIC FITS
C      ITO = NO. OF OUTWARD STEPS
C      ITI = NO. OF INWARD STEPS
C      IGO = 0 FOR EXIT ON OUTWARD SEARCH
C            1 FOR EXIT ON INWARD SEARCH
C            2 FOR EXIT ON PARABOLIC SEARCH
C
           IT = 0
           ITP = 0
           ITO = 0
           ITI = 0
C
C    BEGIN THE OUTWARD SEARCH
C
           IGO = 0
           D1 = -BIG
           D3 = BIG
          C1 = VBIG
          C3 = VBIG
           GO TO 20
C
C    CHANGE THE DIRECTION OF THE OUTWARD SEARCH
C
10         IF(D.GT.D2)GO TO 12
           D1 = D
           C1 = C
           GO TO 14
12         D3 = D
           C3 = C
14         IF(D1.GT.-BIG.AND.D3.LT.BIG)GO TO 35
           SIDE = -SIDE
C
```

```
C    TAKE AN OUTWARD STEP
C
20        ITO = ITO + 1
          IF(IT.GT.1)DELTA = DELTA*2.
          D = D2 + SIDE*DELTA
        IF(IPRINT.GE.4)WRITE(NODEV,100)D1,D2,D3,C1,C2,C3
100     FORMAT(/,3X,'D1',11X,'D2',11X,'D3',11X,'C1',11X,'C2',11X,'C3',
       1       /,1X,6G13.6)
        CALL COSTD(D,C,DMIN,CMIN)
          IF(IT.GE.LIMIT)RETURN
          IT = IT + 1
          IF(C.GT.C2)GO TO 10
          IF(D2.GT.D)GO TO 16
          D1 = D2
          C1 = C2
          GO TO 18
16        D3 = D2
          C3 = C2
18        D2 = D
          C2 = C
          GO TO 20
C
C    THIS COMPLETES THE OUTWARD SEARCH,  NOW DO INWARD SEARCH
C
35        IGO = 1
          GO TO 45
40        IF(MOD(IT-IPLAST,3))45,45,50
C
C    DO PARABOLIC FIT
C
45      IGO = 2
        DTEST = .7*ABS(D1-D3)
          CALL MINPAR(D1,D2,D3,C1,C2,C3,CO,EPS,ICONV,DMIN,CMIN)
        ITP = ITP + 1
        IT = IT + 1
        IF(IT.GE.LIMIT)RETURN
          IF(ICONV.EQ.2)RETURN
        IF(ABS(D1-D3).LE.DTEST)GO TO 45
C
C    GIVE UP ON PARABOLIC SEARCH AND GO TO INWARD GOLDEN SECTION SEARCH
C
          IPLAST = IT
          D2 = (D3+D1)/2. - SIDE*DTAU*(D3-D1)
        IGO = 1
        IF(IPRINT.GE.4)WRITE(NODEV,101)D1,D3,C1,C3
101     FORMAT(/,3X,'D1',24X,'D3',11X,'C1',24X,'C3',
       1       /,1X,G13.6,13X,2G13.6,13X,G13.6)
        CALL COSTD(D2,C2,DMIN,CMIN)
        IF(C2.GT.AMIN1(C1,C3))NONU = NONU + 1
        IF(C2.GT.AMIN1(C1,C3).AND.IPRINT.GE.4)
       1      WRITE(NODEV,102)D1,D2,D3,C1,C2,C3
102     FORMAT(/,' CAUTION...FUNCTION IS NOT UNIMODAL.  D1-3, C1-3 =',
       1       /,1X,6G10.3)
C
C    TAKE AN INWARD STEP
```

```
C
50      D = (D3+D1)/2. + SIDE*DTAU*(D3-D1)
      IF(IPRINT.GE.4)WRITE(NODEV,100)D1,D2,D3,C1,C2,C3
      CALL COSTD(D,C,DMIN,CMIN)
      IF(C.GT.AMIN1(C1,C3))NONU = NONU + 1
      IF(C.GT.AMIN1(C1,C3).AND.IPRINT.GE.4)
     1     WRITE(NODEV,103)D1,D2,D3,D,C1,C2,C3,C
103   FORMAT(/,' CAUTION...FUNCTION IS NOT UNIMODAL.  D1-3,D, '
     1     ,'C1-3,C =',/,1X,8G10.3)
        ITI = ITI + 1
        ERR = AMIN1(ABS(C),ABS((C-C2)/C2))
        IF(ERR.LE.EPS.AND.C.LT.C0)RETURN
        IF(IT.GE.LIMIT)RETURN
        IT = IT + 1
        SIDE = SIGN(1.,(C2-C)*SIDE)
        DT = D
        CT = C
        IF(C.GT.C2)GO TO 60
        DT = D2
        CT = C2
        D2 = D
        C2 = C
60      IF(SIDE.GT.0.)GO TO 62
        D3 = DT
        C3 = CT
        GO TO 40
62      D1 = DT
        C1 = CT
        GO TO 40
        END
C
        SUBROUTINE MINPAR(X1,X2,X3,F1,F2,F3,FU,EP,ICONV,DMIN,CMIN)
      COMMON/NNOUT/NODEV
      COMMON/IPRINT/IPRINT
      ICONV = 0
        A1 = X2 - X3
        A2 = X3 - X1
        A3 = X1 - X2
        DEN = F1*A1 + F2*A2 + F3*A3
        IF(DEN.EQ.0.)RETURN
        B1 = X2**2 - X3**2
        B2 = X3**2 - X1**2
        B3 = X1**2 - X2**2
        X4 = .5*(F1*B1+F2*B2+F3*B3)/DEN
        IF(X4.LE.X1.OR.X4.GE.X3)RETURN
      IF(IPRINT.GE.4)WRITE(NODEV,100)X1,X2,X3,F1,F2,F3
100   FORMAT(/,3X,'D1',11X,'D2',11X,'D3',11X,'C1',11X,'C2',11X,'C3',
     1       /,1X,6G13.6)
      CALL COSTD(X4,F4,DMIN,CMIN)
      IF(F4.GT.AMIN1(F1,F3))NONU = NONU + 1
      IF(F4.GT.AMIN1(F1,F3).AND.IPRINT.GE.4)
     1     WRITE(NODEV,101)X1,X2,X3,X4,F1,F2,F3,F4
101   FORMAT(/,' CAUTION...FUNCTION IS NOT UNIMODAL.  D1-4,C1-4 =',
     1     /,1X,8G10.3)
        ER = AMIN1(ABS(F4),ABS((F4-F2)/F2))
```

```
      IF(ER.LE.EP.AND.F4.LT.F0)ICONV = 2
      IF(F4-F2)10,10,12
10    XT = X2
      FT = F2
      X2 = X4
      F2 = F4
      IF(X4-XT)14,14,16
14    X3 = XT
      F3 = FT
      RETURN
16    X1 = XT
      F1 = FT
      RETURN
12    IF(X4-X2)18,18,20
18    X1 = X4
      F1 = F4
      RETURN
20    X3 = X4
      F3 = F4
      RETURN
      END
C
      SUBROUTINE RANIND(N,INDX)
      DIMENSION INDX(1),IND(25),RAND(25)
      REAL*8 DSEED
      COMMON/ISEED/ISEED
      COMMON/NNOUT/NODEV
      DO 10 I=1,N
10    IND(I) = I
C     WRITE(NODEV,100)ISEED,N,DSEED
100   FORMAT(/,' IN RANDIN,  ISEED,N,DSEED =',I12,I4,1PD14.6)
      CALL RANDS(ISEED,N,RAND)
C     WRITE(NODEV,101)N,DSEED,(RAND(I),I=1,N)
101   FORMAT(/,' AFTER GGUBS IS CALLED,  N,DSEED =',I4,1PD14.6,
     1     /,'   RAND =',10F8.5)
      DO 20 K=2,N
      I = N + 2 - K
      RI = I
      II = IFIX(RI*RAND(I)) + 1
      II = MAX0(1,MIN0(I,II))
      INDX(I) = IND(II)
      IF(II.EQ.I)GO TO 20
      III = II + 1
      DO 15 J=III,I
15    IND(J-1) = IND(J)
20    CONTINUE
      INDX(1) = IND(1)
      RETURN
      END
C     FILE WRTMAT FORTRAN A1
C
C     5/8/79
C     FILE OF UTILITY SUBROUTINES TO SUPPORT VIBE FORTRAN A1
C
C     WRTMAT - GENERAL MATRIX OUTPUT SUBROUTINE
```

```
C
      SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
      DIMENSION A(IA,1)
      COMMON/NNOUT/NPRINT
      REAL*8 ANAME
      WRITE(NPRINT,100)ANAME,N,M
100   FORMAT(/,' MATRIX ',A8,3X,'(',I3,' ROWS X',I3,' COLS)')
      IF(M.LE.10)GO TO 15
      DO 10 I=1,N
10    WRITE(NPRINT,101)(A(I,J),J=1,M)
101   FORMAT(/,(1P10E13.5))
      RETURN
15    DO 20 I=1,N
20    WRITE(NPRINT,102)(A(I,J),J=1,M)
102   FORMAT(1P10E13.5)
      RETURN
      END
C
C       TRANSP - TRANSPOSES A MATRIX
C
        SUBROUTINE TRANSP(A,N1,N2,NA)
        DIMENSION A(NA,1)
        COMMON/NNOUT/NOUT
        M = MAXO(N1,N2)
        IF(M.GT.NA.OR.M.LE.0)GO TO 90
        M1 = M-1
        DO 10 I=1,M1
        I1 = I + 1
        DO 10 J=I1,M
        TEMP = A(I,J)
        A(I,J) = A(J,I)
10      A(J,I) = TEMP
        RETURN
90      WRITE(NOUT,100)N1,N2,NA
100     FORMAT(' *** ERROR IN TRANSP *** N1,N2,NA =',3I4)
        RETURN
        END
C
C        MULT - MATRIX MULTIPLICATION
C
        SUBROUTINE MULT(A,B,C,N,L,M,NA,NB,NC)
        DIMENSION A(NA,1),B(NB,1),C(NC,1)
        DOUBLE PRECISION TEMP
        COMMON/NNOUT/NOUT
        IF(N.GT.MINO(NA,NC).OR.L.GT.NB)GO TO 90
        DO 20 I=1,N
        DO 20 J=1,M
        TEMP = 0.
        DO 10 K=1,L
10      TEMP = TEMP + DBLE(A(I,K))*DBLE(B(K,J))
20      C(I,J) = TEMP
        RETURN
90      WRITE(NOUT,100)N,NA,NC,L,NB
100     FORMAT(' *** ERROR IN MULT *** N,NA,NC,L,NB=',5I5)
        RETURN
```

```
      END
C
C       MSHIFT - TRANSFERES VECTORS AND MATRICES
C
      SUBROUTINE MSHIFT(A,B,N,M,NA,NB)
      DIMENSION A(NA,1),B(NB,1)
      COMMON/NNOUT/NOUT
      IF(N.GT.MINO(NA,NB))GO TO 90
      DO 10 I=1,N
      DO 10 J=1,M
10    B(I,J) = A(I,J)
      RETURN
90    WRITE(NOUT,100)N,NA,NB
100   FORMAT(' *** ERROR IN MSHIFT *** N,NA,NB=',3I5)
      RETURN
      END
C
C       MATADD - MATRIX ADDITION
C
      SUBROUTINE MATADD(A,B,C,N,M,NA,NB,NC)
      DIMENSION A(NA,1),B(NB,1),C(NC,1)
      COMMON/NNOUT/NOUT
      IF(N.GT.MINO(NA,NB,NC))GO TO 90
      DO 10 I=1,N
      DO 10 J=1,M
10    C(I,J) = A(I,J) + B(I,J)
      RETURN
90    WRITE(NOUT,100)N,NA,NB,NC
100   FORMAT(' *** ERROR IN MATADD *** N,NA,NB,NC=',4I5)
      RETURN
      END
C
C       MATSUB - MATRIX SUBTRACTION
C
      SUBROUTINE MATSUB(A,B,C,N,M,NA,NB,NC)
      DIMENSION A(NA,1),B(NB,1),C(NC,1)
      COMMON/NNOUT/NOUT
      IF(N.GT.MINO(NA,NB,NC))GO TO 90
      DO 10 I=1,N
      DO 10 J=1,M
10    C(I,J) = A(I,J) - B(I,J)
      RETURN
90    WRITE(NOUT,100)N,NA,NB,NC
100   FORMAT(' *** ERROR IN MATSUB *** N,NA,NB,NC=',4I5)
      RETURN
      END
C
C       SWAP - INTERCHANGES TWO VARABLES
C
      SUBROUTINE SWAP(A,B)
      C = A
      A = B
      B = C
      RETURN
      END
```

```
C
C       MSMULT - MATRIX*SCALAR MULTIPLICATION
C
        SUBROUTINE MSMULT(S,A,N,M,NA)
        DIMENSION A(NA,1)
        DO 10 I=1,N
        DO 10 J=1,M
10      A(I,J) = S*A(I,J)
        RETURN
        END
C
C       ZERO - FILLS A MATRIX WITH ZEROS
C
        SUBROUTINE ZERO(A,N,M,NA)
        DIMENSION A(NA,1)
        DO 10 I=1,N
        DO 10 J=1,M
10      A(I,J) = 0.
        RETURN
        END
C
C
C
C       IMAT - LOADS AN ARRAY WITH THE IDENTITY MATRIX
C
        SUBROUTINE IMAT(A,N,NA)
        DIMENSION A(NA,1)
        DO 10 I=1,N
        DO 5 J=1,N
5       A(I,J) = 0.
10      A(I,I) = 1.
        RETURN
        END
C
C
C        ANORM - CALCULATES THE RSS NORM OF A MATRIX
C
        FUNCTION ANORM(A,N1,N2,NA)
        DIMENSION A(NA,1)
        COMMON/NNOUT/NOUT
        IF(N1.GT.NA)GO TO 90
        ANORM = 0.
        DO 10 I=1,N1
        DO 10 J=1,N2
10      ANORM = ANORM + A(I,J)**2
        ANORM = SQRT(ANORM)
        RETURN
90      WRITE(NOUT,100)N,NA
100     FORMAT(' *** ERROR IN ANORM *** N,NA =',2I5)
        RETURN
C   FILE NEWSOM INPUT
C
C   JULY 22, 1980
C
 &NSOMIN
```

```
 N = 4, M = 2,
 EPS = 1.E-7,
 MAXFN = 5000,
 NLOOPS = 40,
 NSRCH = 12,
 NODEV = 6,
 IPRINT = 1,
 ICONT = 1,
 ICOUT = 0,
 WT1 = .01, WT2 = 100., WT3 = 500., WT4 = 100.,
 WT5 = 20.,
 REMIN = -20., REMAX = -1., RTOMAX = .5,
 A = -0.367984, 1., -0.024209, 0.258819, 3*0., 0.017835, -0.032279,
     0.267949, -0.110395, -0.965926, 26.1875, 0., 4.46294, -0.091072,
 B = -7.67183, 0., 1.96959, 0., 2.06549, 0., -2.33843, 5*0.,
  X =.1894,-.633,.359,.01327,2.366,4.1414,-1.313,.6683,
 X = .1494,-.4193,.1054,.0203,1.6827,
     3.658,-.2315,-.04795,
 X = .6481, -.4298, .1054, -.2476, 1.6827,
     3.608, -.73056, -.04789,
&END
&NSOMIN
 REMAX = -2.,
&END
&NSOMIN
 REMAX = -4.,
&END
&NSOMIN
 M = 3,
 A = -3.18, 1., -0.06, 0.022,3*0., 0.0644, 0.63, 0., -0.27, -0.998,
     -10.6, 0., 4.18, -0.151,
 B = -14.4, 3*0., 1.5, 0., -2.59, 0.037, 2*0., -0.96, 0.,
    X=-2.23E-8,-3.5E-8,3.43E-8,1.73E-10,.45,-.82,-1.8,-1.76,10.7,78,-5.9
 X = -3.97E-2,-4.56E-3,1.99E-2,1.202E-2,-.9449,
     3.5593,.21612,-3.6611E-3,4.7516,-.94851,
     -.54778,-.5141,
 X = -.03498, .15196, -2.6329, .1378, -.93295,
     2.5353, .2175, 2.6445E-3, 5.5306, -.9673,
     -.6965, -.5259,
 REMAX = -1.,
&END
&NSOMIN
 REMAX = -2.,
&END
&NSOMIN
 REMAX = -4.,
 ICONT = 0,
&END
```

```
C
C       FILE LINEAR FORTRAN
C
        REAL A(10,10),B(10,10),C(10,10),Q(10,10),R(10,10)
        REAL K(10,10),G(10,10),ACL(10,10),Q1(10,10),XC(10),UE(10)
        REAL A1(10,10),B1(10,10),UREF(10)
        COMMON/INOU/KIN,KOUT
        COMMON/MAIN1/NDIM,DUM1(10,10)
        COMMON/MAIN2/DUM2(10,10)
        COMMON/PLOT/SCALE,ARRAY(101,11)
        KIN=3
        KOUT=4
        NDIM=10
        N=4
        M=3
        IR=4
        CALL MATIO(N,N,A,4)
C       CALL EQUATE(N,N,A1,A)
        CALL MATIO(N,M,B,4)
C       CALL EQUATE(N,M,B1,B)
        CALL MATIO(IR,N,C,4)
        CALL MATIO(N,N,Q,4)
        CALL MATIO(M,M,R,4)
        CALL CON(N,M,A,B,NCS)
        CALL OBS(N,N,A,C,NCS)
        CALL REG(N,M,A,B,R,Q,K,G,ACL)
C       CALL VECTIO(N,XC,4)
C       CALL VECTIO(M,UE,4)
CC      CALL VECTIO(M,UREF,4)
        CALL MATIO(M,N,G,4)
C       CALL LTRACK(A1,N,B1,M,G,XC,UE)
        STOP
        END
C
C       SUBROUTINE  REG
C
        SUBROUTINE REG(N,M,A,B,R,Q,X,G,ACL)
        DIMENSION A(1),B(1),R(1),Q(1),X(1),G(1),ACL(1)
        DIMENSION RR(30),RI(30)
        COMMON/MAIN1/NDIM,DUM1(1)
        COMMON/MAIN2/DUM2(1)
        COMMON/INOU/KIN,KOUT
        IF(NDIM.LT.N) WRITE(KOUT,1000)
        IF(NDIM.LT.M) WRITE(KOUT,1000)
        IF(NDIM.LT.N) CALL EXIT
        IF(NDIM.LT.M) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
        CALL MEIGV(N,A,RR,RI)
        WRITE(KOUT,200)
        WRITE(KOUT,300)(RR(I),RI(I),I=1,N)
        CALL EQUATE(M,M,DUM1,R)
        CALL GMINV(M,M,DUM1,DUM2,MR,0)
        CALL MAT4(N,M,DUM2,B,G)
        CALL MRIC(N,A,G,Q,X,ACL)
        CALL EQUATE(M,M,DUM1,R)
```

```
      CALL GMINV(M,M,DUM1,DUM2,MR,0)
      CALL MAT6(M,M,N,DUM2,B,DUM1)
      CALL MMUL(DUM1,X,M,N,N,G)
      WRITE(KOUT,60)
      CALL MATIO(N,N,X,3)
      WRITE(KOUT,70)
      CALL MATIO(N,N,ACL,3)
      CALL MEIGV(N,ACL,RR,RI)
      WRITE(KOUT,400)
      WRITE(KOUT,300)(RR(I),RI(I),I=1,N)
      WRITE(KOUT,80)
      CALL MATIO(M,N,G,3)
   60 FORMAT(//,19H RICATTI SOLUTION K,/)
   70 FORMAT(//,31H OPTIMAL CLOSED LOOP MATRIX ACL,/)
   80 FORMAT(//,22H OPTIMAL GAIN MATRIX G,/)
  200 FORMAT(/,22H OPEN LOOP EIGENVALUES,/)
  300 FORMAT(13H REAL PART = ,E10.3,13H IMAG PART = ,E10.3)
  400 FORMAT(/,27H CLOSED LOOP EIGENVALUES = ,/)
      RETURN
      END
C
C     SUBROUTINE MRIC
C
      SUBROUTINE MRIC(N,A,S,Q,X,Z)
      DIMENSION A(1),S(1),Q(1),X(1),Z(1),TR(30),TIMES(3)
      COMMON/MAIN1/NDIM,F(1)
      COMMON/INOU/KIN,KOUT
      NDIM1=NDIM+1
      TIMES(1)=.5
      TIMES(2)=2.0
      TIMES(3)=4.0
      NN=N*NDIM
      T1=-.5*ALOG(XNORM(N,Q)+.0001)
      IF(T1.LT.-1.0)T1=-1.0/T1
      IF(ABS(T1).LT.1.0)T1=1.0
      T2=1.*N/(1.+XNORM(N,A))*T1
      T1=T2
      KEY=0
    5 KEY=KEY+1
   10 DO 15 I=1,N
      DO 15 J=I,NN,NDIM
   15 X(J)=-S(J)
      CALL INTEG(N,A,X,Z,-T1)
      CALL FACTOR(N,Z,F,MR)
C     POSSIBLE UNCONTROLLABILITY IF MR.NE.N
      IF(MR.EQ.-1) CALL MATIO(N,N,Z,3)
      IF(MR.EQ.-1) CALL EXIT
      CALL GMINV(N,N,F,Z,MR,0)
      CALL MAT2(N,N,Z,Z,X)
C     A+SX IS STABLE
      TOL=1.E-5
      ADV=TOL*1.E-7
      NN=N*NDIM
      NM1=N-1
      DO 19 I=1,N
```

```
      TR(I)=-1.0
   19 CONTINUE
      TOL1=TOL/10.
      MAXIT=30+N
      DO 40 IT=1,MAXIT
      CALL MMUL(S,X,N,N,N,F)
      CALL MMUL(X,F,N,N,N,Z)
      DO 20 I=1,NN,NDIM
      II=I+NM1
      DO 20 J=I,II
      X(J)=A(J)-F(J)
   20 Z(J)=Z(J)+Q(J)
      CALL MLINEQ(N,X,Z,X,TOL1)
      L=0
      C1=0.0
      II=1
      DO 25 I=1,N
      IF(ABS(X(II)-TR(I)).LT.(ADV+TOL*X(II))) L=L+1
      TR(I)=X(II)
      II=II+NDIM1
   25 C1=C1+TR(I)
      IF(ABS(C1).GT.1.E20) GO TO 50
      IF(L.NE.N) GO TO 40
      CALL GMINV(N,N,Z,F,MR,0)
      CALL MMUL(S,X,N,N,N,Z)
      WRITE(KOUT,27)IT
   27 FORMAT(17HORICCATI SOLN IN ,I2,11H ITERATIONS)
      DO 30 I=1,NN,NDIM
      II=I+NM1
      DO 30 J=I,II
   30 Z(J)=A(J)-Z(J)
      IF(MR.NE.N)WRITE(KOUT,35)MR
   35 FORMAT(26HORICCATI SOLN IS PSD--RANK13)
      RETURN
   40 CONTINUE
      WRITE(KOUT,45)MAXIT,T1
45    FORMAT(26HORICCATI NON-CONVERGENT IN12,11H ITERATIONS,12H  INITIAL
     1 T=F10.5)
      GO TO 60
   50 WRITE(KOUT,55)IT,T1
   55 FORMAT(29HORICCATI BLOW UP AT ITERATION12,12H  INITIAL T=F10.5)
   60 IF(KEY.EQ.4)CALL EXIT
      T1=T2*TIMES(KEY)
      WRITE(KOUT,65)T1
   65 FORMAT(14HORESET WITH T=F10.5)
      GO TO 5
      END
C
C     SUBROUTINE MLINEQ
C
      SUBROUTINE MLINEQ(N,A,C,X,TOL)
C     SOLVES A'X+XA+C=0
C     A AND X CAN BE IN SAME LOCATION IF DESIRED
C     ANSWER RETURNED IN C AND X
      DIMENSION A(1),C(1),X(1),RR(30),RI(30)
```

```
      COMMON/MAIN1/NDIM,F(1)
      COMMON/MAIN2/Y(1)
      COMMON/INOU/KIN,KOUT
      NDIM1=NDIM+1
      DT=.5
      DT1=0.
      NN=N*NDIM
      DO 5 II=1,NN,NDIM1
    5 DT1=DT1+ABS(A(II))
      DT1=DT1/N
      IF(DT1.GT.4.0) DT=DT*4.0/DT1
      II=1
      DO 20 I=1,N
      DO 15 JJ=I,NN,NDIM
   15 Y(JJ)=DT*A(JJ)
      Y(II)=Y(II)-.5
   20 II=II+NDIM1
      CALL GMINV(N,N,Y,F,MR,1)
      CALL EQUATE(N,N,Y,A)
      IF(MR.EQ.N) GO TO 21
      IT=0
      DO 18 I=1,NN,NDIM1
   18 C(I)=1.E25
      GO TO 95
   21 CALL MMUL(C,F,N,N,N,X)
C     INITIALIZATION OF X,F
      I=1
      DO 40 II=1,NN,NDIM
      J=II
      IF(I.EQ.1) GO TO 30
      DO 25 JJ=I,II,NDIM
      C(J)=C(JJ)
   25 J=J+1
   30 ID=J
      DO 35 JJ=II,NN,NDIM
      C(J)=DT*DOT(N,F(II),X(JJ))
   35 J=J+1
      F(ID)=F(ID)+1.0
   40 I=I+1
   50 ADV=TOL*1.E-7
      DO 90 IT=1,30
      NEZ=0
      SIZE=0.0
      CALL MMUL(C,F,N,N,N,X)
      I=1
      II=1
      J=1
      GO TO 70
   60 J=II
      DO 65 JJ=I,II,NDIM
      C(J)=C(JJ)
   65 J=J+1
   70 ID=J
      DT1=C(J)
      DO 75 JJ=II,NN,NDIM
```

```
      C(J)=C(J)+DOT(N,F(II),X(JJ))
   75 J=J+1
      J=J-1
      DO 80 JJ=II,J
   80 X(JJ)=F(JJ)
      IF(ABS(C(ID)-DT1).LT.(ADV+TOL*ABS(C(ID)))) NEZ=NEZ+1
      I=I+1
      II=II+NDIM
      SIZE=SIZE+DT1
      IF(I.LE.N) GO TO 60
      IF(NEZ.EQ.N) GO TO 150
      IF(ABS(SIZE).GT.1.E25) GO TO 95
      CALL MMUL(X,X,N,N,N,F)
   90 CONTINUE
   95 WRITE(KOUT,100)IT
  100 FORMAT(33HOLIN EQN ALGORITHM NON-CONVERGENT,I3,10HITERATIONS)
      WRITE(KOUT,110)
  110 FORMAT(35HOA MATRIX EIGENVALUES..REAL     IMAJ)
      CALL MEIGV(N,Y,RR,RI)
      WRITE(KOUT,120)(RR(I),RI(I),I=1,N)
  120 FORMAT(18X,1P2E12.3)
  150 CONTINUE
      CALL EQUATE(N,N,X,C)
      RETURN
      END
C
C     SUBROUTINE INTEG
C
      SUBROUTINE INTEG(N,A,C,S,T)
C     S=INTEGRAL EA*C*EA' FROM 0 TO T
C     C IS DESTROYED
      DIMENSION A(1),C(1),S(1),COEF(15)
      COMMON/MAIN1/NDIM,X(1)
      NDIM1=NDIM+1
      NN=N*NDIM
      NM1=N-1
      NT=13
      NTM1=NT-1
      IND=0
      ANORM=XNORM(N,A)
      DT=T
    5 IF (ANORM*ABS(DT).LE.0.5) GO TO 10
      DT=DT/2.
      IND=IND+1
      GO TO 5
   10 DO 15 I=1,NN,NDIM
      J=I+NM1
      DO 15 JJ=I,J
   15 S(JJ)=DT*C(JJ)
      T1=DT**2/2.
      DO 25 IT=3,NT
      CALL MMUL(A,C,N,N,N,X)
      DO 20 I=1,N
      II=(I-1)*NDIM
      DO 20 JJ=I,NN,NDIM
```

```
      II=II+1
      C(JJ)=(X(JJ)+X(II))*T1
   20 S(JJ)=S(JJ)+C(JJ)
   25 T1=DT/FLOAT(IT)
      IF(IND.EQ.0) GO TO 100
      COEF(NT)=1.0
      DO 30 I=1,NTM1
      II=NT-I
   30 COEF(II)=DT*COEF(II+1)/FLOAT(I)
      DO 40 I=1,NN,NDIM
      II=1
      J=I+NM1
      DO 35 JJ=I,J
   35 X(JJ)=A(JJ)*COEF(1)
      X(II)=X(II)+COEF(2)
   40 II=II+NDIM1
      DO 55 L=3,NT
      CALL MMUL(A,X,N,N,N,C)
      II=1
      T1=COEF(L)
      DO 55 I=1,NN,NDIM
      J=I+NM1
      DO 50 JJ=I,J
   50 X(JJ)=C(JJ)
      X(II)=X(II)+T1
   55 II=II+NDIM1
C     X=EXP(A*DT)
      L=0
   60 L=L+1
      CALL MMUL(X,S,N,N,N,C)
      II=1
      DO 90 I=1,N
      J=II
      IF (I.EQ.1) GO TO 75
      DO 70 JJ=I,II,NDIM
      S(JJ)=S(J)
   70 J=J+1
   75 DO 85 JJ=I,N
      KK=JJ
      DO 80 K=I,NN,NDIM
      S(J)=S(J)+C(K)*X(KK)
   80 KK=KK+NDIM
   85 J=J+NDIM
      DO 87 JJ=I,NN,NDIM
   87 C(JJ)=X(JJ)
   90 II=II+NDIM
      IF(L.EQ.IND) GO TO 100
      CALL MMUL(C,C,N,N,N,X)
      GO TO 60
  100 CONTINUE
      RETURN
      END
C
C     SUBROUTINE GMINV
C
```

```
      SUBROUTINE GMINV(NR,NC,A,U,MR,MT)
      DIMENSION A(1),U(1),S(30)
      COMMON/MAIN1/NDIM
      COMMON/INOU/KIN,KOUT
      NDIM1=NDIM+1
      TOL=1.E-14
      ADV=1.E-24
      MP=NC
      NRM1=NR-1
      TOL1=0.
      JJ=1
      DO 10 J=1,NC
      S(J)=DOT(NR,A(JJ),A(JJ))
      IF(S(J).GT.TOL1)TOL1=S(J)
10    JJ=JJ+NDIM
      TOL1=ADV*TOL1
      ADV=TOL1
      JJ=1
      DO 100 J=1,NC
      FAC=S(J)
      JM1=J-1
      JRM=JJ+NRM1
      JCM=JJ+JM1
      DO 20 I=JJ,JCM
20    U(I)=0.
      U(JCM)=1.0
      IF(J.EQ.1) GO TO 54
      KK=1
      DO 30 K=1,JM1
      IF(S(K).EQ.1.0) GO TO 30
      TEMP=-DOT(NR,A(JJ),A(KK))
      CALL VADD(K,TEMP,U(JJ),U(KK))
30    KK=KK+NDIM
      DO 50 L=1,2
      KK=1
      DO 50 K=1,JM1
      IF(S(K).EQ.0.) GO TO 50
      TEMP=-DOT(NR,A(JJ),A(KK))
      CALL VADD(NR,TEMP,A(JJ),A(KK))
      CALL VADD(K,TEMP,U(JJ),U(KK))
50    KK=KK+NDIM
      TOL1=TOL*FAC+ADV
      FAC=DOT(NR,A(JJ),A(JJ))
54    IF(FAC.GT.TOL1) GO TO 70
      DO 55 I=JJ,JRM
55    A(I)=0.
      S(J)=0.
      KK=1
      DO 65 K=1,JM1
      IF(S(K).EQ.0.) GO TO 65
      TEMP=-DOT(K,U(KK),U(JJ))
      CALL VADD(NR,TEMP,A(JJ),A(KK))
65    KK=KK+NDIM
      FAC=DOT(J,U(JJ),U(JJ))
      MR=MR-1
```

```
      GO TO 75
   70 S(J)=1.0
      KK=1
      DO 72 K=1,JM1
      IF(S(K).EQ.1.) GO TO 72
      TEMP=-DOT(NR,A(JJ),A(KK))
      CALL VADD(K,TEMP,U(JJ),U(KK))
   72 KK=KK+NDIM
   75 FAC=1./SQRT(FAC)
      DO 80 I=JJ,JRM
   80 A(I)=A(I)*FAC
      DO 85 I=JJ,JCM
   85 U(I)=U(I)*FAC
  100 JJ=JJ+NDIM
      IF(MR.EQ.NR.OR.MR.EQ.NC) GO TO 120
      IF(MT.NE.0)WRITE(KOUT,110)NR,NC,MR
  110 FORMAT(I3,1HX,I2,8H M: RANK,I2)
  120 NEND=NC*NDIM
      JJ=1
      DO 135 J=1,NC
      DO 125 I=1,NR
      II=I-J
      S(I)=0.
      DO 125 KK=JJ,NEND,NDIM
  125 S(I)=S(I)+A(II+KK)*U(KK)
      II=J
      DO 130 I=1,NR
      U(II)=S(I)
  130 II=II+NDIM
  135 JJ=JJ+NDIM1
      RETURN
      END
C
C     SUBROUTINE FACTOR
C
      SUBROUTINE FACTOR(N,A,S,MR)
C     A=S*S
      DIMENSION A(1),S(1)
      COMMON/MAIN1/NDIM
      COMMON/INOU/KIN,KOUT
      NDIM1=NDIM+1
      MR=0
      NN=N*NDIM
      TOL=1.E-7
      TOL1=0.
      DO 1 I=1,NN,NDIM1
      R=ABS(A(I))
    1 IF(R.GT.TOL1)TOL1=R
      TOL1=TOL1*1.E-12
      II=1
      DO 50 I=1,N
      IM1=I-1
      DO 5 JJ=I,NN,NDIM
    5 S(JJ)=0.
      ID=II+IM1
```

```
      R=A(ID)-DOT(IM1,S(II),S(II))
      IF(ABS(R).LT.(TOL*A(ID)+TOL1)) GO TO 50
      IF (R) 15,50,20
   15 MR=-1
      WRITE(KOUT,1000)
 1000 FORMAT(37HOTRIED TO FACTOR AN INDEFINITE MATRIX)
      RETURN
   20 S(ID)=SQRT(R)
      MR=MR+1
      IF (I.EQ.N) RETURN
      L=II+NDIM
      DO 25 JJ=L,NN,NDIM
      IJ=JJ+IM1
   25 S(IJ)=(A(IJ)-DOT(IM1,S(II),S(JJ)))/S(ID)
   50 II=II+NDIM
      RETURN
      END
C
C     SUBROUTINE MEIGV
C
      SUBROUTINE MEIGV(N,A,RR,RI)
      DIMENSION A(1),RR(1),RI(1),C(31),TEMP(30)
      COMMON/MAIN1/NDIM,X(1)
      NDIM1=NDIM+1
      NN=N*NDIM
      DO 1 I=1,N
      DO 1 J=I,NN,NDIM
    1 X(J)=A(J)
      C(N+1)=1.0
      L=1
    5 C1=0.0
      DO 10 I=1,NN,NDIM1
   10 C1=C1-X(I)
      C1=C1/FLOAT(L)
      I=N+1-L
      C(I)=C1
      DO 15 I=1,NN,NDIM1
   15 X(I)=X(I)+C1
      IF(L.EQ.N) GO TO 50
      DO 40 I=1,N
      JJ=1
      DO 35 J=I,NN,NDIM
      C1=0.
      KK=J-I
      DO 25 K=I,NN,NDIM
      KK=KK+1
   25 C1=C1+X(K)*A(KK)
      TEMP(JJ)=C1
   35 JJ=JJ+1
      JJ=1
      DO 40 J=I,NN,NDIM
      X(J)=TEMP(JJ)
   40 JJ=JJ+1
      L=L+1
      GO TO 5
```

```
   50 CALL POLRT(C,N,RR,RI)
      RETURN
      END
C
C     SUBROUTINE POLRT
C
       SUBROUTINE POLRT(A,N,U,V)
C     DETERMINES THE ROOTS OF AN N-TH ORDER POYNOMIAL
C       X**N + A(N)*X**(N-1) + ... + A(1) = 0
C     WHERE:     U(N) WILL CONTAIN THE ROOT REAL PARTS
C                V(N) WILL CONTAIN ROOT IMAGINARY PARTS
C                A(N) WILL BE DESTROYED DURING COMPUTATION
      DIMENSION A(1),U(1),V(1)
    1  NR = N
   10  IF (NR-2) 61,71,11
   11  IF (A(1)) 20,12,20
   12  U(NR) = 0.0
       V(NR) = 0.0
       NR = NR-1
      CALL VECTEQ(NR,A(2),A(1))
       GO TO 10
   20  EMIN = 1.
       TOL = .1
       V4 = 1.0
       P = 0.
       Q = 0.
       R = 0.0
   30  U(NR) = A(NR) - P
       U(NR-1) = A(NR-1) - P*U(NR) - Q
       V(NR) = U(NR) - P
       V(NR-1) = U(NR-1) - P*V(NR) - Q
       I = NR-2
   35  U(I) = A(I) - (P*U(I+1) + Q*U(I+2))
       V(I) = U(I) - (P*V(I+1) + Q*V(I+2))
       I = I-1
       IF (I.GT.0) GO TO 35
   40  IF (A(2)) 42,41,42
   41  E = U(2)/A(1)
       GO TO 43
   42  E = U(2)/A(2)
   43  F = AMAX1(ABS(E),1.0E-6)*AMAX1(ABS(U(1)/A(1)),1.0E-6)
       IF (E.LE.1.E-12) GO TO 70
       IF (E.GE.EMIN) GO TO 44
C         THIS FORCES EMIN TO HOLD STEADY FOR 5 ITERATIONS
       EMIN = E
       TOL = EMIN*0.7
       GO TO 45
C         THIS WILL ALLOW AN ERROR X*EMIN ONLY AFTER N ITERATIONS
C           WHERE  X = (1.1)**N
   44  IF (E.LT.TOL) GO TO 70
   45  CBAR = V(2) - U(2)
       IF (NR.GT.3) V4 = V(4)
       D = V(3)**2 - CBAR*V4
       IF (D) 47,46,47
   46  P = P - 2.0
```

```
      Q = Q*(Q+1.0)
      GO TO 50
   47 P = P + (U(2)*V(3) - U(1)*V4)/D
      Q = Q + (-U(2)*CBAR + U(1)*V(3))/D
   50 U(NR) = A(NR) + R
      V(NR) = U(NR) + R
      I = NR - 1
   55 U(I) = A(I) + R*U(I+1)
      V(I) = U(I) + R*V(I+1)
      I = I-1
      IF (I.GT.0) GO TO 55
      E = ABS(U(1)/A(1))
      IF (E.LE.1.E-12) GO TO 60
      IF (E.GE.EMIN) GO TO 56
      EMIN = E
      TOL = EMIN*0.7
      GO TO 57
   56 IF (E.LT.TOL) GO TO 60
   57 IF (V(2).NE.0) GO TO 58
      R = R+1.
      GO TO 59
   58 R = R - U(1)/V(2)
   59 TOL = TOL*1.1
      GO TO 30
C        STORE A SINGLE REAL ROOT
   60 CALL VECTEQ(NR-1,U(2),A)
      GO TO 62
   61 R = -A(1)
   62 U(NR) = R
      V(NR) = 0.0
      NR = NR-1
      GO TO 80
C        STORE A PAIR OF ROOTS
   70 CALL VECTEQ(NR-2,U(3),A)
      GO TO 72
   71 P = A(2)
      Q = A(1)
   72 P = (-0.5)*P
      D = P*P - Q
      IF (D) 75,78,78
   75 U(NR) = P
      U(NR-1) = P
      V(NR) = -SQRT(-D)
      V(NR-1) = -V(NR)
      GO TO 79
   78 V(NR) = 0.0
      V(NR-1) = 0.0
      D = ABS(P) + SQRT(D)
      IF (P.LT.0.0) D = -D
      U(NR) = D
      U(NR-1) = Q/D
   79 NR = NR-2
   80 IF (NR.GT.0) GO TO 10
      RETURN
      END
```

```
C
C       SUBROUTINE EQUATE
C
        SUBROUTINE EQUATE(NR,NC,A,B)
        DIMENSION A(1),B(1)
        COMMON/MAIN1/NDIM
        NN=NC*NDIM
        NR1=NR-1
        DO 1 J=1,NN,NDIM
        II=J+NR1
        DO 1 IJ=J,II
        A(IJ)=B(IJ)
      1 CONTINUE
        RETURN
        END
C
C       SUBROUTINE MAT4
C
        SUBROUTINE MAT4(N1,N2,X,Y,Z)
C       Z=YXY' X=X' IS N2XN2, Y IS N1XN2, Z IS N1XN1
        DIMENSION X(1),Y(1),Z(1)
        COMMON/MAIN1/NDIM
        CALL MMUL(Y,X,N1,N2,N2,Z)
        NN2=N2*NDIM
        DO 5 I=1,N1
        IM1=I-1
        II=IM1*NDIM
        JJ=I+II
        DO 3 J=I,N1
        TEMP=0.
        KK=J
        DO 1 K=I,NN2,NDIM
        TEMP=TEMP+Y(K)*Z(KK)
      1 KK=KK+NDIM
        Z(JJ)=TEMP
      3 JJ=JJ+NDIM
        JJ=I
        K=II+1
        KK=II+IM1
        DO 5 J=K,KK
        Z(JJ)=Z(J)
        JJ=JJ+NDIM
      5 CONTINUE
        RETURN
        END
C
C       SUBROUTINE MAT6
C
        SUBROUTINE MAT6(N1,N2,N3,X,Y,Z)
        DIMENSION X(1),Y(1),Z(1)
C            COMPUTE Z=XY' WHEN X IS N1XN2, Y IS N3XN2, Z IS N1XN3
        COMMON/MAIN1/NDIM
        DO 2 I=1,N1
        DO 2 J=1,N3
        TM=0.
```

```
      DO 1 K=1,N2
    1 TM=TM+X(I+(K-1)*NDIM)*Y(J+(K-1)*NDIM)
    2 Z(I+(J-1)*NDIM)=TM
      RETURN
      END
C
C     SUBROUTINE MMUL
C
      SUBROUTINE MMUL(X,Y,N1,N2,N3,Z)
      DIMENSION X(1),Y(1),Z(1)
      COMMON/MAIN1/NDIM
      NEND3=NDIM*N3
      NEND2=NDIM*N2
      DO 1 I=1,N1
      DO 1 J=I,NEND3,NDIM
      TM=0.
      K=I
      KK=J-I
    5 KK=KK+1
      TM=TM+X(K)*Y(KK)
      K=K+NDIM
      IF(K.LE.NEND2) GO TO 5
    1 Z(J)=TM
      RETURN
      END
C
C     SUBROUTINE MATIO
C
      SUBROUTINE MATIO(NR,NC,X,IO)
      DIMENSION X(1)
      COMMON/MAIN1/NDIM
      COMMON/INOU/KIN,KOUT
      JEND=NC*NDIM
      IF(IO.EQ.4) GO TO 40
      IF(IO.EQ.3)GO TO 20
    5 CONTINUE
C
C     INPUT
C
      DO 10 I=1,NR
      READ(KIN,1000)(X(IJ),IJ=I,JEND,NDIM)
   10 CONTINUE
      IF(IO.EQ.1)RETURN
C
C     OUTPUT
C
   20 DO 30 I=1,NR
      WRITE(KOUT,1001)(X(IJ),IJ=I,JEND,NDIM)
   30 CONTINUE
      GO TO 50
C
C     TITLE
C
   40 READ(KIN,1002)
      WRITE(KOUT,1003)
```

```
      WRITE(KOUT,1002)
      WRITE(KOUT,1004)
      GO TO 5
   50 CONTINUE
 1000 FORMAT(8E10.0)
 1001 FORMAT(1X,1P10E13.4)
 1002 FORMAT(1X,79H
     C                                )
 1003 FORMAT(//)
 1004 FORMAT(/)
      RETURN
      END
      FUNCTION DOT(NR,A,B)
      DIMENSION A(1),B(1)
      DOT=0.
      DO 1 I=1,NR
    1 DOT=DOT+A(I)*B(I)
      RETURN
      END
C
C     SUBROUTINE VADD
C
      SUBROUTINE VADD(N,C1,A,B)
      DIMENSION A(1),B(1)
      DO 1 I=1,N
    1 A(I)=A(I)+C1*B(I)
      RETURN
      END
      FUNCTION XNORM(N,A)
C          COMPUTES AN APPROXIMATION TO NORM OF A-- NOT A BOUND
      DIMENSION A(1)
      COMMON/MAIN1/NDIM
      NDIM1=NDIM+1
      NN=N*NDIM
      C1=0.
      TR=A(1)
      IF(N.EQ.1) GO TO 20
      I=2
      DO 10 II=NDIM1,NN,NDIM
      J=II
      DO 5 JJ=I,II,NDIM
      C1=C1+ABS(A(J)*A(JJ))
    5 J=J+1
      TR=TR+A(J)
   10 I=I+1
      TR=TR/FLOAT(N)
      DO 15 II=1,NN,NDIM1
   15 C1=C1+(A(II)-TR)**2
   20 XNORM=ABS(TR)+SQRT(C1)
      RETURN
      END
C
C     SUBROUTINE MAT2
C
      SUBROUTINE MAT2(N1,N2,X,Y,Z)
```

```
C       Z=XY' X,Y=N1*N2,Z=Z'
C       Z AND Y CAN BE EQUIVALENT
        DIMENSION X(1),Y(1),Z(1)
        COMMON/MAIN1/NDIM
        NDIM1=NDIM+1
        NN2=N2*NDIM
        II=1
        DO 10 I=1,N1
        IJ=II
        DO 5 J=I,N1
        Z(IJ)=DOT2(NN2,X(I),Y(J))
     5  IJ=IJ+NDIM
        J=II
        IJ=J
     3  IJ=IJ-NDIM
        IF(IJ.LT.I) GO TO 10
        J=J-1
        Z(IJ)=Z(J)
        GO TO 3
    10  II=II+NDIM1
        RETURN
        END
        FUNCTION DOT2(NN,A,B)
        DIMENSION A(1),B(1)
        COMMON/MAIN1/NDIM
        DOT2=0
        DO 1 I=1,NN,NDIM
        DOT2=DOT2+A(I)*B(I)
     1  CONTINUE
        RETURN
        END
C
C       SUBROUTINE REGSIM
C
        SUBROUTINE REGSIM(N,M,IR,ACL,G,C,XO,DT,NS,NP)
        DIMENSION ACL(1),G(1),C(1),XO(1)
        COMMON/MAIN1/NDIM,DUM1(1)
        COMMON/INOU/KIN,KOUT
        IF(NDIM.LT.N) WRITE(KOUT,1000)
        IF(NDIM.LT.M) WRITE(KOUT,1000)
        IF(NDIM.LT.IR) WRITE(KOUT,1000)
        IF(NDIM.LT.N) CALL EXIT
        IF(NDIM.LT.M) CALL EXIT
        IF(NDIM.LT.IR) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
        LOGICAL CONT
        CONT=.TRUE.
        CALL LNSIM2(N,IR,ACL,C,XO,DT,NS,NP,M,G,CONT)
        RETURN
        END
C
C       SUBROUTINE LNSIM2
C
        SUBROUTINE LNSIM2(N,R,A,C,XO,DT,NSTEPS,NPRPL,M,G,CONT)
        DIMENSION A(1),C(1),XO(1),G(1)
```

```
      DIMENSION X(30),Y(30),INDEX(30),NSYM(1),U(30)
      INTEGER R,RP1
      LOGICAL CONT
      COMMON/INOU/KIN,KOUT
      COMMON/PLOT/SCALE,ARRAY(1)
      COMMON/MAIN1/NDIM,DUM1(1)
      COMMON/MAIN2/FA(1)
      LOGICAL PRINT,PLOT
      IF(CONT) CALL MSCALE(M,N,G,-1.0,G)
      RP1=R+1
      NR=NSTEPS+1
      NC=R+1
      IF(CONT) NC=NC+M
      PLOT=.TRUE.
      PRINT=.TRUE.
      IF(NPRPL.LT.0) PRINT=.FALSE.
      IF(NPRPL.EQ.0) PLOT=.FALSE.
      IF(NPRPL.EQ.0) GO TO 1
      IF(NSTEPS.GT.( 50*SCALE)) WRITE(KOUT,2000)
      IF(NSTEPS.GT.( 50*SCALE)) CALL EXIT
 2000 FORMAT(/,12H SCALE ERROR,/)
    1 CALL MEXP(N,A,DT,EA)
      T=0.
      CALL VMMUL(R,N,C,XO,Y)
      IF(CONT) CALL VMMUL(M,N,G,XO,U)
      IF(.NOT.PRINT) GO TO 5
      IF(.NOT.CONT) WRITE(KOUT,80)
      IF(CONT) WRITE(KOUT,85)
      WRITE(KOUT,90)
      IF(CONT) WRITE(KOUT,95)
      WRITE(KOUT,100) T
      WRITE(KOUT,105)(Y(I),I=1,R)
      IF(CONT) WRITE(KOUT,105)(U(I),I=1,M)
    5 IF(.NOT.PLOT) GO TO 9
      ARRAY(1)=T
      DO 6 J=2,RP1
    6 ARRAY(1+(J-1)*NR)=Y(J-1)
      IF(.NOT.CONT) GO TO 9
      DO 7 J=1,M
    7 ARRAY(1+(R+J)*NR)=U(J)
    9 DO 60 K=1,NSTEPS
      CALL VMMUL(N,N,EA,XO,X)
      CALL VECTEQ(N,X,XO)
      CALL VMMUL(R,N,C,XO,Y)
      T=K*DT
      IF(CONT) CALL VMMUL(M,N,G,X,U)
      IF(.NOT.PRINT) GO TO 54
   52 WRITE(KOUT,100) T
      WRITE(KOUT,105)(Y(I),I=1,R)
      IF(CONT) WRITE(KOUT,105)(U(I),I=1,M)
   54 IF(.NOT.PLOT) GO TO 60
      DO 58 J=2,RP1
   58 ARRAY(1+K+(J-1)*NR)=Y(J-1)
      ARRAY(1+K)=K*DT
      IF(.NOT.CONT) GO TO 60
```

```
      DO 59 J=1,M
   59 ARRAY(1+K+(R+J)*NR)=U(J)
   60 CONTINUE
      IF(CONT) CALL MSCALE(M,N,G,-1.0,G)
      IF(.NOT.PLOT) GO TO 120
      NSYM(1)=13
      NVAR=1
      NHORZ=50*SCALE+1
      NGRIDH=5*SCALE
      NSORT=0
      DO 65 I=1,R
      INDEX(1)=I
      WRITE(KOUT,110) I
   65 CALL PRNPLT(NR,NC,NVAR,ARRAY,INDEX,NHORZ,NGRIDH,NSORT,NSYM)
      IF(.NOT.CONT) GO TO 120
      NSYM(1)=12
      DO 70 I=1,M
      INDEX(1)=R+I
      WRITE(KOUT,115) I
   70 CALL PRNPLT(NR,NC,NVAR,ARRAY,INDEX,NHORZ,NGRIDH,NSORT,NSYM)
   80 FORMAT(/,28H SIMULATION OF LINEAR SYSTEM,/)
   85 FORMAT(/,31H SIMULATION OF LINEAR REGULATOR,/)
   90 FORMAT(11H   OUTPUT Y)
   95 FORMAT(12H   CONTROL U)
  100 FORMAT(/,5H T = ,F5.2,/)
  105 FORMAT(1X,0P10E13.4)
  110 FORMAT(1H1,/,31X,1HY,I2,12H VERSUS TIME)
  115 FORMAT(1H1,/,31X,1HU,I2,12H VERSUS TIME)
  120 CONTINUE
      RETURN
      END
C
C     SUBROUTINE MEXP
C
      SUBROUTINE MEXP(N,A,T,EA)
      DIMENSION A(1),EA(1),C(30),D(31),E(30)
      COMMON/MAIN1/NDIM,X(1)
      NDIM1=NDIM+1
      NN=NDIM*N
      NM1=N-1
      IF(N.GT.1) GO TO 5
      EA(1)=EXP(T*A(1))
      RETURN
    5 W=1.0
      DO 10 I=1,NN,NDIM
      IL=I+NM1
      DO 10 J=I,IL
   10 EA(J)=A(J)
      C1=XNORM(N,A)
      IND=0
      L=1
      T1=T
   15 IF(ABS(T1*C1).LE.3.0) GO TO 20
      T1=T1/2.
      IND=IND+1
```

```
      GO TO 15
   20 C2=0.
      DO 25 I=1,NN,NDIM1
   25 C2=C2-EA(I)
      C2=C2/FLOAT(L)
      C(L)=C2
      D(L+1)=0.
      II=N+1-L
      E(II)=W
      II=1
      DO 35 I=1,NN,NDIM
      IL=I+NM1
      DO 30 J=I,IL
   30 X(J)=EA(J)
      X(II)=X(II)+C2
   35 II=II+NDIM1
      IF(L.EQ.N) GO TO 40
      CALL MMUL(X,A,N,N,N,EA)
      W=W*T1/FLOAT(L)
      L=L+1
      GO TO 20
   40 CONTINUE
C     CAN CHECK X:0 FOR ACCURACY
      J=N+25
      DO 50 L=N,J
      DO 45 K=1,N
      D(K)=(D(K+1)-W*C(K))*T1/FLOAT(L)
   45 E(K)=E(K)+D(K)
   50 W=D(1)
      II=1
      DO 60 I=1,NN,NDIM
      IL=I+NM1
      DO 55 J=I,IL
   55 EA(J)=E(1)*A(J)
      EA(II)=EA(II)+E(2)
   60 II=II+NDIM1
      IF(N.EQ.2) GO TO 85
      DO 80 L=3,N
      CALL MMUL(EA,A,N,N,N,X)
      II=1
      C2=E(L)
      DO 75 I=1,NN,NDIM
      IL=I+NM1
      DO 70 J=I,IL
   70 EA(J)=X(J)
      EA(II)=EA(II)+C2
   75 II=II+NDIM1
   80 CONTINUE
   85 IF(IND.EQ.0) RETURN
      DO 100 L=1,IND
      DO 90 I=1,NN,NDIM
      IL=I+NM1
      DO 90 J=I,IL
   90 X(J)=EA(J)
  100 CALL MMUL(X,X,N,N,N,EA)
```

```
      RETURN
      END
C
C     SUBROUTINE PRNPLT
C
      SUBROUTINE PRNPLT(NR,NC,NVAR,ARRAY,INDEX,NHORZ,NGRIDH,NSORT,NSYM)
      DIMENSION ARRAY(1),INDEX(1),KAXIS(101),NSYM(1),YLABEL(11)
      INTEGER CHAR(15)
      COMMON/INOU/KIN,KOUT
      DATA CHAR(1),CHAR(2),CHAR(3),CHAR(4),CHAR(5),
     *CHAR(6),CHAR(7),CHAR(8),CHAR(9),CHAR(10),CHAR(11),
     *CHAR(12),CHAR(13),CHAR(14),CHAR(15)/1H1,1H2,1H3,1H4,
     *1H5,1H6,1H7,1H8,1H9,1HX,1HE,1HU,1HY,1HR,1H /
      DATA ISTAR,IPER,IDASH,IBLANK/1H*,1H.,1H-,1H /
      XMIN=ARRAY(1)
      XMAX=ARRAY(NR)
      IF(NSORT.EQ.0) GO TO 10
      DO 5 I=1,NR
      IF(XMIN.GT.ARRAY(I)) XMIN=ARRAY(I)
    5 IF(XMAX.LT.ARRAY(I)) XMAX=ARRAY(I)
   10 YMAX=ARRAY(1+INDEX(1)*NR)
      YMIN=YMAX
      DO 50 J=1,NVAR
      DO 50 I=1,NR
      IJ=I+INDEX(J)*NR
      IF(YMIN.GT.ARRAY(IJ)) YMIN=ARRAY(IJ)
   50 IF(YMAX.LT.ARRAY(IJ)) YMAX=ARRAY(IJ)
      IF(ABS(YMAX-YMIN).GE.1.0E-20) GO TO 55
      YMAX=YMAX+1.
      YMIN=YMIN-1.
   55 CALL RNDOFF(YMAX,YMIN)
      YLABEL(1)=YMIN
      YLABEL(6)=YMAX
      DO 60 I=2,5
   60 YLABEL(I)=(I-1)*(YMAX-YMIN)/5.0+YMIN
      WRITE(KOUT,500)(YLABEL(I),I=1,6)
      IF(NGRIDH.GE.1) GO TO 65
      LINEPR=NHORZ/5
      GO TO 70
   65 LINEPR=NGRIDH
   70 KLINE=1
      LINE=1
      STEPX=(XMAX-XMIN)/(NHORZ-1)
      STEPY=(YMAX-YMIN)/50.
      DO 220 IND=1,NR
      KSTEP=(ARRAY(IND)-XMIN)/STEPX+1.5
   75 IF(LINE-KLINE) 140,80,80
   80 XLABEL=STEPX*(KLINE-1)+XMIN
      IF((NGRIDH.EQ.0).AND.(LINE.NE.1)) GO TO 130
      DO 90 I=2,51
   90 KAXIS(I)=IDASH
      DO 100 I=1,51,10
  100 KAXIS(I)=ISTAR
  105 IF(KSTEP-LINE) 110,110,125
  110 DO 120 J=1,NVAR
```

```
      K=(ARRAY(IND+INDEX(J)*NR)-YMIN)/STEPY+1.5
  120 KAXIS(K)=CHAR(NSYM(J))
  125 WRITE(KOUT,600) XLABEL,(KAXIS(I),I=1,51)
      KLINE=KLINE+LINEPR
      GO TO 200
  130 DO 135 I=2,51
  135 KAXIS(I)=IBLANK
      KAXIS(1)=IPER
      GO TO 105
  140 DO 150 I=2,51
  150 KAXIS(I)=IBLANK
      KAXIS(1)=IPER
      IF(NGRIDH.EQ.0) GO TO 165
      DO 160 I=11,51,10
  160 KAXIS(I)=IPER
  165 IF(KSTEP-LINE) 170,170,190
  170 DO 180 J=1,NVAR
      K=(ARRAY(IND+INDEX(J)*NR)-YMIN)/STEPY+1.5
  180 KAXIS(K)=CHAR(NSYM(J))
  190 WRITE(KOUT,700)(KAXIS(I),I=1,51)
  200 LINE=LINE+1
      IF(LINE-NHORZ-1) 210,210,230
  210 IF(KSTEP-LINE) 220,75,75
  220 CONTINUE
  230 RETURN
  500 FORMAT(//,9X,6(1PE10.2))
  600 FORMAT(1PE13.2,2X,51A1)
  700 FORMAT(15X,51A1)
      RETURN
      END
C
C     SUBROUTINE MSCALE
C
      SUBROUTINE MSCALE(N1,N2,A,X,B)
      DIMENSION A(1),B(1)
      COMMON/MAIN1/NDIM
      JEND=N2*NDIM
      DO 10 I=1,N1
      DO 10 IJ=I,JEND,NDIM
   10 B(IJ)=X*A(IJ)
      RETURN
      END
C
C     SUBROUTINE VMMUL
C
      SUBROUTINE VMMUL(N1,N2,X,Y,Z)
      DIMENSION X(1),Y(1),Z(1)
      COMMON/MAIN1/NDIM
      DO 10 I=1,N1
      Z(I)=0.
      DO 10 J=1,N2
   10 Z(I)=Z(I)+X(I+(J-1)*NDIM)*Y(J)
      RETURN
      END
C
```

```
C       SUBROUTINE VECTEQ
C
        SUBROUTINE VECTEQ(N,X,Y)
        DIMENSION X(1),Y(1)
        DO 10 I=1,N
   10   Y(I)=X(I)
        RETURN
        END
C
C       SUBROUTINE RNDOFF
C
        SUBROUTINE RNDOFF(DMAX,DMIN)
        COMMON/INOU/KIN,KOUT
        RANGE=DMAX-DMIN
        J=-1
        K=0
        IF(RANGE.LE.2.) J=1
        DO 10 I=1,25
        IF((RANGE.GT.2.).AND.(RANGE.LE.20.)) GO TO 20
        K=K+J
   10   RANGE=RANGE*10.**J
        WRITE(KOUT,100)
        CALL EXIT
   20   T=DMIN*10.**K
        IF(DMIN.LT.0) GO TO 30
        N=T
        DMIN=N*10.**(-K)
        GO TO 40
   30   N=T
        T1=N
        IF(T1.GT.T) N=N-1
        DMIN=N*10.**(-K)
   40   T=DMAX*10.**K
        IF(DMAX.GT.0) GO TO 50
        N=T
        DMAX=N*10.**(-K)
        GO TO 60
   50   N=T
        T1=N
        IF(T1.LT.T) N=N+1
        DMAX=N*10.**(-K)
   60   CONTINUE
  100   FORMAT(/,1X,32HNUMBERS OUT OF RANGE FOR PLOTTER,/)
        RETURN
        END
C
C       SUBROUTINE OBS
C
        SUBROUTINE OBS(N,IR,A,C,NOS)
        DIMENSION A(1),C(1)
        COMMON/INOU/KIN,KOUT
        COMMON/MAIN1/NDIM
        IF(NDIM.LT.N) WRITE(KOUT,1000)
        IF(NDIM.LT.IR) WRITE(KOUT,1000)
        IF(NDIM.LT.N) CALL EXIT
```

```
      IF(NDIM.LT.IR) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
      CALL TRANS1(N,A,A)
      CALL TRANS3(IR,N,C,C)
      CALL CONT(N,IR,A,C,NOS)
      IF(NOS.LT.N) GO TO 10
      WRITE(KOUT,20)
      GO TO 50
   10 WRITE(KOUT,30)
      WRITE(KOUT,40) NOS
   20 FORMAT(/,10X,20HSYSTEM IS OBSERVABLE,/)
   30 FORMAT(/,10X,22HSYSTEM IS UNOBSERVABLE,/)
   40 FORMAT(15X,30HNUMBER OF OBSERVABLE STATES = ,I2,/)
   50 CALL TRANS1(N,A,A)
      CALL TRANS3(N,IR,C,C)
      RETURN
      END
C
C     SUBROUTINE TRANS1
C
      SUBROUTINE TRANS1(N,A,AT)
C     SETS AT= ATRANSPOSE A AND AT CAN BE EQUIVALENT, BOTH ARE SQUARE
      DIMENSION A(1),AT(1)
      COMMON/MAIN1/NDIM
      NDIM1=NDIM+1
      NN=N*NDIM
      DO 1 I=1,NN,NDIM1
      IJ=I
      DO 1 JI=I,NN,NDIM
      TEMP=A(IJ)
      AT(IJ)=A(JI)
      AT(JI)=TEMP
      IJ=IJ+1
    1 CONTINUE
      RETURN
      END
C
C     SUBROUTINE TRANS3
C
      SUBROUTINE TRANS3(NR,NC,A,AT)
C  SETS AT= TRANSPOSE OF A , A AND AT CAN BE EQUIVALENT , A IS NR X NC
      DIMENSION A(1),AT(1)
      COMMON/MAIN1/NDIM
      NDIM1=NDIM+1
      NR1=NR-1
      NC1=NC-1
      IF(NR.GE.NC) N=NC
      IF(NR.LT.NC) N=NR
      NN=N*NDIM
      DO 1 I=1,NN,NDIM1
      IJ=I
      DO 1 JI=I,NN,NDIM
      TEMP=A(IJ)
      AT(IJ)=A(JI)
      AT(JI)=TEMP
```

```
      IJ=IJ+1
    1 CONTINUE
      IF(NR-NC) 5,2,3
    2 RETURN
    3 DO 4 I=NC,NR1
      DO 4 J=1,NC
    4 A(J+I*NDIM)=A(I+1+(J-1)*NDIM)
      RETURN
    5 DO 6 I=1,NR
      DO 6 J=NR,NC1
    6 A(J+1+(I-1)*NDIM)=A(I+J*NDIM)
      RETURN
      END
C
C     SUBROUTINE CONT
C
      SUBROUTINE CONT(N,M,A,B,NCS)
      DIMENSION A(1),B(1)
      COMMON/MAIN1/NDIM,DUM1(1)
      CALL EQUATE(N,M,DUM1,B)
   20 INDEX=1
      IR=M
      CALL ORTHNM(DUM1,N,IR,0)
      NCS=IR
       IF (IR.EQ.N) RETURN
       IRLAST = 0
   30  INDEX = INDEX+1
       M1 = IRLAST+1
       IRLAST = IR
C         MULTIPLY COLUMNS 'M1' THROUGH 'IRLAST' BY MATRIX A
      K1=(M1-1)*NDIM+1
      K2=IR*NDIM+1
   31  ML = MINO(IRLAST+1-M1,N-IR)
   33  CALL MMUL(A,DUM1(K1),N,N,ML,DUM1(K2))
   35  M1 = M1+ML
       IRHOLD = IR
       IR = IR+ML
      CALL ORTHNM(DUM1,N,IR,IRHOLD)
      NCS=IR
       IF (IR.EQ.N) RETURN
       IF (M1-IRLAST) 31,31,50
   50 IF (IR-IRLAST) 60,60,30
   60 INDEX = 0
      RETURN
      END
C
C     SUBROUTINE ORTHNM
C
      SUBROUTINE ORTHNM(A,NR,NC,NNCO)
      DIMENSION A(1)
      COMMON/MAIN1/NDIM
C     COLUMN ORTHO-NORMALIZATION ROUTINE (GRAM-SCHMIDT ALGORITHM)
C               NR IS THE COLUMN LENGTH
C               NC IS THE NUMBER OF COLUMNS
C               NNCO IS THE NUMBER OF ORTHONORMAL COLUMNS AT CALL TIME
```

```
C       AND:        NC WILL BE REDUCED IF LINEARLY DEPENDANT COLUMNS OCCUR.
        NNC=NNC0
     1   II = 1
     4   IF (NNC) 5,21,5
     5   IF (NC-NNC) 50,50,10
    10  ILIM=NDIM*NNC
        II = ILIM + 1
    15  DO 20 K=1,2
        DO 20 I=1,ILIM,NDIM
        W=-DOT(NR,A(I),A(II))
        CALL WADD(A(II),A(I),A(II),W,NR)
    20  CONTINUE
    21  W=DOT(NR,A(II),A(II))
        IF (W - 1.0E-12) 30,40,40
    30  IF (NC-NNC-1) 33,33,35
    33 NC = NNC
        RETURN
    35  I=(NC-1)*NDIM+1
        CALL VECTEQ(NR,A(I),A(II))
        NC=NC-1
        GO TO 4
     40  W = 1.0/SQRT(W)
        CALL MSCALE(NR,1,A(II),W,A(II))
        NNC=NNC+1
        GO TO 5
    50  RETURN
        END
C
C       SUBROUTINE WADD
C
        SUBROUTINE WADD(A,B,C,W,N)
        DIMENSION A(1),B(1),C(1)
        DO 10 I=1,N
    10  C(I)=A(I)+W*B(I)
        RETURN
        END
C
C       SUBROUTINE VECTIO
C
        SUBROUTINE VECTIO(N,X,IO)
        DIMENSION X(1)
        COMMON/INOU/KIN,KOUT
        IF(IO.EQ.4) GO TO 40
        IF(IO.EQ.3) GO TO 20
     5 CONTINUE
C
C       INPUT
C
        READ(KIN,1000)(X(I),I=1,N)
        IF(IO.EQ.1)RETURN
C
C       OUTPUT
C
    20 WRITE(KOUT,1001)(X(I),I=1,N)
        GO TO 50
```

```
C
C       TITLE
C
   40   READ(KIN,1002)
        WRITE(KOUT,1003)
        WRITE(KOUT,1002)
        WRITE(KOUT,1004)
        GO TO 5
   50   CONTINUE
 1000 FORMAT(8E10.0)
 1001 FORMAT(1X,1P10E13.4)
 1002 FORMAT(1X,79H
     C                                   )
 1003 FORMAT(//)
 1004 FORMAT(/)
        RETURN
        END
C
C       SUBROUTINE CON
C
        SUBROUTINE CON(N,M,A,B,NCS)
        DIMENSION A(1),B(1)
        COMMON/INOU/KIN,KOUT
        COMMON/MAIN1/NDIM
        IF(NDIM.LT.N) WRITE(KOUT,1000)
        IF(NDIM.LT.M) WRITE(KOUT,1000)
        IF(NDIM.LT.N) CALL EXIT
        IF(NDIM.LT.M) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
        CALL CONT(N,M,A,B,NCS)
        IF(NCS.LT.N) GO TO 10
        WRITE(KOUT,20)
        RETURN
   10   WRITE(KOUT,30)
        WRITE(KOUT,40) NCS
   20   FORMAT(/,10X,22HSYSTEM IS CONTROLLABLE,/)
   30   FORMAT(/,10X,24HSYSTEM IS UNCONTROLLABLE,/)
   40   FORMAT(15X,32HNUMBER OF CONTROLLABLE STATES = ,I2,/)
        RETURN
        END
C
C       SUBROUTINE TRANS2
C
        SUBROUTINE TRANS2(NR,NC,A,AT)
C       AT=ATRANSPOSE,  A IS NR X NC
        DIMENSION A(1),AT(1)
        COMMON/MAIN1/NDIM
        NN=NC*NDIM
        II=1
        DO 6 I=1,NR
        JJ=II
        DO 5 J=I,NN,NDIM
        AT(JJ)=A(J)
     5  JJ=JJ+1
        II=II+NDIM
```

```
    6 CONTINUE
      RETURN
      END
C
C     SUBROUTINE EIGVAL
C
      SUBROUTINE EIGVAL(N,A)
      DIMENSION A(1),RR(30),RI(30)
      COMMON/MAIN1/NDIM
      COMMON/INOU/KIN,KOUT
      IF(NDIM.LT.N) WRITE(KOUT,1000)
      IF(NDIM.LT.N) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
      CALL MEIGV(N,A,RR,RI)
      WRITE(KOUT,100)
      WRITE(KOUT,200)(RR(I),RI(I),I=1,N)
  100 FORMAT(/,19H MATRIX EIGENVALUES,/)
  200 FORMAT(13H REAL PART = ,E10.3,13H IMAG PART = ,E10.3)
      RETURN
      END
C
C     SUBROUTINE VMAT2
C
      SUBROUTINE VMAT2(N1,N2,N3,A,B,C,D,E)
      DIMENSION A(1),B(1),C(1),D(1),E(1)
      COMMON/MAIN1/NDIM
      DO 30 I=1,N1
      TEMP=0.
      DO 10 J=1,N2
   10 TEMP=TEMP+A(I+(J-1)*NDIM)*B(J)
      DO 20 J=1,N3
   20 TEMP=TEMP+C(I+(J-1)*NDIM)*D(J)
   30 E(I)=TEMP
      RETURN
      END
C
C     SUBROUTINE MAT5
C
      SUBROUTINE MAT5(N1,N2,N3,X,Y,Z)
      DIMENSION X(1),Y(1),Z(1)
C     COMPUTE Z=X*Y WHERE X IS N1XN2,Y IS N1XN3,Z IS N2XN3
      COMMON/MAIN1/NDIM
      DO 2 I=1,N2
      DO 2 J=1,N3
      TM = 0.
      II=(I-1)*NDIM
      JJ=(J-1)*NDIM
      DO 1 K=1,N1
    1 TM=TM+X(II+K)*Y(K+JJ)
    2 Z(I+JJ)=TM
      RETURN
      END
C
C     SUBROUTINE MAT3A
C
```

```
      SUBROUTINE MAT3A(N1,N2,X,Y,Z)
C     Z=X'YX  Y=Y' IS N2XN2  X' IS N1XN2
      DIMENSION X(1),Y(1),Z(1)
      COMMON/MAIN1/NDIM
      NDIM1=NDIM+1
      NN1=N1*NDIM
      CALL MMUL(Y,X,N2,N2,N1,Z)
      I=0
      DO 10 II=1,NN1,NDIM
      J=II+I
      IJ=J-1
      DO 5 JJ=II,NN1,NDIM
      Z(J)=DOT(N2,X(II),Z(JJ))
    5 J=J+1
      I=I+1
      JJ=I
      DO 10 J=II,IJ
      Z(J)=Z(JJ)
   10 JJ=JJ+NDIM
      RETURN
      END
C
C     SUBROUTINE LINSIM
C
      SUBROUTINE LINSIM(N,IR,A,C,XO,DT,NS,NP)
      DIMENSION A(1),C(1),XO(1),G(1)
      COMMON/MAIN1/NDIM,DUM1(1)
      COMMON/INOU/KIN,KOUT
      IF(NDIM.LT.N) WRITE(KOUT,1000)
      IF(NDIM.LT.IR) WRITE(KOUT,1000)
      IF(NDIM.LT.N) CALL EXIT
      IF(NDIM.LT.IR) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
      LOGICAL CONT
      CONT=.FALSE.
      M=1
      CALL LNSIM2(N,IR,A,C,XO,DT,NS,NP,M,G,CONT)
      RETURN
      END
C
C     SUBROUTINE EQCOST
C
      SUBROUTINE EQCOST(N,M,A,B,R,S,Q,AEQ,QEQ)
      DIMENSION A(1),B(1),R(1),S(1),Q(1)
      DIMENSION AEQ(1),QEQ(1)
      COMMON/INOU/KIN,KOUT
      COMMON/MAIN1/NDIM,DUM1(1)
      COMMON/MAIN2/DUM2(1)
      IF(NDIM.LT.N) WRITE(KOUT,1000)
      IF(NDIM.LT.M) WRITE(KOUT,1000)
      IF(NDIM.LT.N) CALL EXIT
      IF(NDIM.LT.M) CALL EXIT
 1000 FORMAT(/,16H DIMENSION ERROR,/)
      CALL EQUATE(M,M,DUM2,R)
      CALL GMINV(M,M,DUM2,DUM1,MR,0)
```

```
      IF(MR.EQ.M) GO TO 10
      WRITE(KOUT,100) MR
      CALL EXIT
   10 CALL MAT4(N,M,DUM1,S,DUM2)
      DO 20 I=1,N
      DO 20 J=1,N
   20 QEQ(I+(J-1)*NDIM)=Q(I+(J-1)*NDIM)-DUM2(I+(J-1)*NDIM)
      CALL MAT6(M,M,N,DUM1,S,DUM2)
      CALL MMUL(B,DUM2,N,M,N,DUM1)
      DO 30 I=1,N
      DO 30 J=1,N
   30 AEQ(I+(J-1)*NDIM)=A(I+(J-1)*NDIM)-DUM1(I+(J-1)*NDIM)
  100 FORMAT(/,25H ERROR MATRIX R HAS RANK ,I2,12H LESS THAN M)
      RETURN
      END
C
C
C
      SUBROUTINE TITLE
      COMMON/INOU/KIN,KOUT
      READ(KIN,1002)
      WRITE(KOUT,1003)
      WRITE(KOUT,1002)
      WRITE(KOUT,1004)
 1002 FORMAT(1X,79H
     C                                )
 1003 FORMAT(//)
 1004 FORMAT(/)
      RETURN
      END
```

```
C      FILE INTODE1 FORTRAN A1
C      PROGRAM TO INTEGRATE GENERAL ODE'S
C
C    THIS VERSION WAS USED ESPECIALLY TO GENERATE TRAJECTORIES FOR
C    THE LANGLEY/NEWSOM CONTRACT  SUMMER 1980
C
       DIMENSION XX(2010),YY(2010),NPTS(10),XYL(4)
       DIMENSION NXPL(10),NXPR(10),XO(12),XI(12)
       DIMENSION X(12),F(10),XNAMES(10),XLBLS(5,10)
       DIMENSION CNAMES(10),XPLBL(10),YPLBL(10),TITL(20),NCINDX(10)
       DATA ISTART/1/
       DATA XYL/0.,5.0,-1.,1./
       DATA WIDTH,HEIGHT,TICKL/7.,5.,.08/
       DATA NXPR/1,2,3,4,5,5*0/,NXPL/1,2,3,7*0/
       DATA NSTEPS,NPLOT,NPRINT/3*1/
       COMMON/CASE/NCASE
       COMMON/NNOUT/NODEV
       COMMON/MAT/A(4,4),B(4,3),XK(3,4),NX,NU,NINT,NTOT
       NAMELIST/INTODE/XI,TMAX,IPRINT,TPRINT,DT,A,B,XK,NX,NU,NINT,NTOT,
      1 NXPR,IPLOT,TPLOT,NODEV,ICONT,NXPL,WIDTH,HEIGHT,TICKL,
      1 NCHARS,NLINES,XNAMES,XLBLS,XPLBL,NXC,YPLBL,NYC,TITL,NTTL
C
C   INPUT VARIABLES IN NAMELIST:
C
C   XI( ) = INITIAL VALUES FOR STATE VARIABLES
C   TMAX = MAXIMUM (FINAL) INTEGRATION TIME
C   IPRINT = INDEX TO DETERMINE THE AMOUNT OF PRINTOUT
C            1, NORMAL OUTPUT
C            0, MINIMUM OUTPUT
C            2, EXTRA OUTPUT
C   TPRINT = TIME INCREMENT FOR TRAJECTORY PRINTOUT IN TABLE
C   DT = INTEGRATION STEP SIZE, SHOULD BE A WHOLE DIVISOR
C        OF TPRINT AND TPLOT
C   A( , ) = NX BY NX SYSTEM MATRIX
C   B( , ) = NX BY NU CONTROL MATRIX
C   XK( , ) = NU BY NX FEEDBACK MATRIX
C   NX = NO. OF STATE VARIABLES
C   NU = NO. OF CONTROL VARIABLES
C   NINT = NUMBER OF VARIABLES TO BE INTEGRATED IN RKINT
C   NTOT = TOTAL NUMBER OF STATE, CONTROL AND AUXILLARY VARIABLES
C   NXPR( ) = ARRAY OF INDICES OF VARIABLES IN TRAJECTORY TABLE
C   IPLOT = INDEX THAT CONTROLS THE GENERATION OF PLOTS
C            0, NO PLOTTER OUTPUT
C            1, GENERATE PRINTER PLOT ONLY
C            2, GENERATE BOTH PRINTER PLOT AND VERSATEC PLOT
C   TPLOT = TIME INCREMENT FOR STORING POINTS FOR PLOTTER
C   NODEV = UNIT NUMBER OF OUTPUT DEVICE, NOMINALLY 6
C   ICONT = INDEX TO CONTINUE OR STOP MULTIPLE CASE RUNS
C            0, STOP AFTER THIS CASE
C            1, CONTINUE TO NEXT CASE
C   NXPL( ) = ARRAY OF INDICES OF VARIABLES TO BE PLOTTED
C   WIDTH = WIDTH OF VERSATEC PLOT IN INCHES
C   HEIGHT = HEIGHT OF VERSATEC PLOT IN INCHES
C   TICKL = LENGTH OF TICK MARKS AND HEIGHT OF CHARACTERS (INCHES)
C            IF NEGATIVE, PLACE TICK MARKS ON INWARD SIDE OF AXIS
```

```
C            IF POSITIVE, PLACE TICK MARKS ON OUTSIDE OF AXIS
C   NCHARS = NO. OF CHARACTERS PER LINE IN PRINTOUT (WIDTH)
C   NLINES = NO. OF LINES FOR PRINTER PLOT (HEIGHT)
C   XNAMES( ) = ARRAY OF ONE-WORD (FOUR-LETTER) NAMES OF VARIABLES
C   XLBLS(5, ) = ARRAY OF 5-WORD (20-CHARACTER) DESCRIPTIONS OF VARIABLES
C   XPLBL( ) = ARRAY CONTAINING THE X-AXIS LABEL FOR PLOTS
C   NXC = NO. OF CHARACTERS (LETTERS ) IN X-AXIS LABEL
C   YPLBL( ) = ARRAY CONTAINING THE Y-AXIS LABEL FOR PLOTS
C   NYC = NO. OF CHARACTERS IN Y-AXIS LABEL
C   TITL( ) = ARRAY CONTAINING A TITLE TO APPEAR IN PRINTOUT AND ON PLOTS
C   NTTL = NO. OF CHARACTERS IN TITLE
C
      NPMAX = 2010
      NODEV = 6
      NCASE = 0
1     READ(7,INTODE)
      IF(IPRINT.GE.2)WRITE(NODEV,INTODE)
      NCASE = NCASE + 1
C
C    DETERMINE NO. OF PRINT COLUMNS
C
      DO 2 I=1,10
      IF(NXPR(I).LE.0)GO TO 3
2     CONTINUE
3     NPRTOT = I-1
C
C   DETERMINE NO. OF CURVES ON PLOT
C
      DO 4 I = 1,10
      IF(NXPL(I).LE.0)GO TO 5
4     CONTINUE
5     NCURV = I-1
C
C    DETERMINE NO. OF INTEGRATION/PLOT/PRINT STEPS
C
      IF(DT.NE.0.)GO TO 6
      WRITE(NODEV,104)
104   FORMAT(/,' *** FATAL ERROR ***   DT IS ZERO.   GO TO NEXT CASE.')
      GO TO 30
6     IF(IPLOT.LE.0)TPLOT = TPRINT
      TPLOT = AMAX1(TPLOT,DT)
      TPRINT = AMAX1(TPRINT,TPLOT)
      NSTEPS = (TPLOT + .00001)/DT
      NPLOT = (TPRINT + .00001)/TPLOT
      NPRINT = (TMAX + .00001)/TPRINT
      NPNTS = NPLOT*NPRINT + 1
C
C   SET UP DATA FOR PLOTS
C
      IF(NCURV.NE.0)NPNTS = MINO(NPNTS,(NPMAX-NCURV)/NCURV)
      DO 7 I=1,NCURV
C     NCINDX(I) = NPNTS*(FLOAT(I)/FLOAT(NCURV+1))
      NCINDX(I)=NPNTS/15.+10*(I-1)
C    THIS LINE PUTS THE LABLES ON  APLOT CURVES CLOSE TO THE LEFT MARGIN.
      K=NXPL(I)
```

```
      CNAMES(I)=XNAMES(K)
7     NPTS(I) = NPNTS
C
C    PRINT BANNER
C
      WRITE(NODEV,105)NCASE
105   FORMAT(1H1,' **********   PROGRAM INTODE     CASE',I3,/)
      CALL DATIME(IM,ID,IY,IH,IMN,AP,'INTODE  ',NODEV)
      IF(IPRINT.GT.0)WRITE(NODEV,100)TITL,TMAX,TPRINT,TPLOT,DT,
     1 IPRINT,IPLOT,NCURV,NPNTS
100   FORMAT(/,1X,20A4,
     1       //' TMAX,TPRINT,TPLOT,DT =',
     1     4F12.5,//' IPRINT,IPLOT,NCURV,NPNTS =',4I5)
C
C   INITIALIZE STATE, CONTROL AND AUXILLARY VARIABLES
C
      CALL INITAL(X,XI,IPRINT)
      TIME = 0.
      CALL AUXVAR(X,TIME)
      IP = 1
      IF(IPLOT.GT.0)CALL XYSTOR(XX,YY,IP,TIME,X,NPNTS,NPMAX,NCURV,NXPL)
      MODE=0
      CALL PTABLE(TIME,X,NXPR,NPRTOT,XNAMES,XLBLS,MODE,NODEV,NCHARS)
C
C   BEGIN INTEGRATION LOOP
C
      DO 10 I=1,NPRINT
      DO 20 J=1,NPLOT
      DO 25 K=1,NSTEPS
25    CALL RKINT(X,TIME,DT,NINT)
      CALL AUXVAR(X,TIME)
      IF(IPLOT.GT.0)CALL XYSTOR(XX,YY,IP,TIME,X,NPNTS,NPMAX,NCURV,NXPL)
20    CONTINUE
      MODE=1
      CALL PTABLE(TIME,X,NXPR,NPRTOT,XNAMES,XLBLS,MODE,NODEV,NCHARS)
10    CONTINUE
C
C   DO PLOTTING
C
      IF(IPLOT.GE.1)CALL PPLOT(XX,YY,NCURV*IP,XYL,NCHARS,NLINES)
      IF(IPLOT.LE.1)GO TO 30
      IF(ISTART.GE.1)CALL PLOTS(0,0,50)
      IF(ISTART.LE.0)CALL PLOT(WIDTH+2.,0.,-3)
      ISTART = 0
      CALL APLOT(XX,YY,NPTS,NCURV,XYL,WIDTH,HEIGHT,TICKL,NCASE,
     1     XPLBL,NXC,YPLBL,NYC,TITL,NTTL,CNAMES,NCINDX)
C
C   GO TO NEXT CASE IF ICONT IS GREATER THAN ZERO
C
30    IF(ICONT.GE.1)GO TO 1
      IF(IPLOT.GT.0)CALL PLOT(0.,0.,999)
      STOP
      END
C
C
```

```
      SUBROUTINE RKINT(X,TIME,DT,NX)
      DIMENSION X(1),DX(10,4),XTEMP(10)
      CALL XDOT(X,TIME,DX(1,1))
      TIMEO = TIME
      DO 20 I=1,3
      T = DT/FLOAT(2-I/3)
      TIME = TIMEO + T
      DO 10 J=1,NX
10    XTEMP(J) = X(J) + T*DX(J,I)
20    CALL XDOT(XTEMP,TIME,DX(1,I+1))
      DO 30 I=1,NX
30    X(I) = X(I)+DT*(DX(I,1)+2.*DX(I,2)+2.*DX(I,3)+DX(I,4))/6.
      TIME = TIMEO + DT
      RETURN
      END
C
      SUBROUTINE XYSTOR(XX,YY,IP,TIME,X,NPTS,NPMAX,NCURV,NXPL)
      DIMENSION XX(1),YY(1),X(1),NXPL(1)
      DO 10 I=1,NCURV
      II = IP + (I-1)*NPTS
      IF(II.GT.NPMAX)RETURN
      XX(II) = TIME
      JJ = NXPL(I)
10    YY(II) = X(JJ)
      IP = IP + 1
      RETURN
      END
C
C
      SUBROUTINE PTABLE(TIME,X,NXPR,NPR,XNAME,XLBLS,MODE,NODEV,NCHARS)
      DIMENSION X(1),NXPR(1),XNAME(1),XO(20),XNAMO(20),XLBLS(5,1)
C
C     INPUTS:
C     MODE=0,PRINT DEFINITIONS, COLUMN HEADINGS AND 1 ROW OF DATA
C         =1,  PRINT DATA ONLY
C     TIME
C     X( )=ARRAY OF STATE AND AUXILLARY VARIABLES
C     NXPR( )=ARRAY OF INDICES OF VARIABLES TO BE PRINTED
C     NPR=NO. OF VARIABLES PRINTED
C     XNAME( )=ARRAY OF Y-CHARECTER LABLES OF VARIABLES
C     XLBLS(5, ) = ARRAYS OF 20-CHARACTER DEFINITIONS OF VARIABLES
C     NODEV=OUTPUT UNIT NO
C     NCHARS=WIDTH OF PAPER(NO OF CHARECTERS)
C
      COMMON/MAT/A(4,4),B(4,3),XK(3,4),NX,NU,NINT,NTOT
C
C  PRINT DEFINITIONS AND HEADINGS
C
      IF(MODE.GE.1)GO TO 20
      WRITE(NODEV,106)(I,XNAME(I),(XLBLS(J,I),J=1,5),I=1,NTOT)
106   FORMAT(//,' STATE, CONTROL AND AUXILLARY VARIABLES:',
     1       //,' VAR',4X,'LABEL',4X,'DEFINITION',
     1       /,(' X(',I2,') =',A4,', ',5A4))
      DO 10 I = 1,NPR
      J = NXPR(I)
```

```
10      XNAMO(I) = XNAME(J)
        IF(NCHARS .LT. 130) WRITE(NODEV,100)(XNAMO(I),I=1,NPR)
100     FORMAT(//,7X,'TIME',7(6X,A4),//,11X,7(6X,A4))
        IF(NCHARS .GE. 130) WRITE(NODEV,105)(XNAMO(I),I=1,NPR)
105        FORMAT(//,7X,'Time',12(6X,A4),//,11X,12(6X,A4))
        WRITE(NODEV,107)
107     FORMAT(10X)
C
C   PRINT ONE (OR MORE) ROWS OF DATA
C
20      NCOL=12
        IF(NCHARS .LT. 130) NCOL=7
        NCOL1=NCOL+1
        XMIN = 10.E10
        XMAX = 10.E-10
        DO 30 I = 1,NPR
        J = NXPR(I)
        XO(I) = X(J)
        IF(XO(I).NE.0.)XMIN = AMIN1(XMIN,ABS(XO(I)))
30      XMAX = AMAX1(XMAX,ABS(XO(I)))
        NPM = MINO(NPR,NCOL)
C
C   DECIDE IF DATA SHOULD BE PRINTED IN FLOATING POINT OR E-FORMAT
C
        IF(XMIN.LT..001.OR.XMAX.GT.99999.)GO TO 40
C
C   PRINT IN F10.3 FORMAT
C
        WRITE(NODEV,101)TIME,(XO(I),I=1,NPM)
101     FORMAT(1X,13F10.3)
        IF(NPR.LE.NCOL)RETURN
C
C   A SECOND ROW MUST BE USED TO PRINT ALL THE DATA
C
        WRITE(NODEV,102)(XO(I),I=NCOL1,NPR)
102     FORMAT(11X,12F10.3)
        RETURN
C
C   PRINT IN E10.3 FORMAT
C
40      WRITE(NODEV,103)TIME,(XO(I),I=1,NPM)
103     FORMAT(1X,F10.3,1P12E10.3)
        IF(NPR.LE.NCOL)RETURN
C
C   A SECOND ROW MUST BE USED TO PRINT ALL THE DATA
C
        WRITE(NODEV,104)(XO(I),I=NCOL1,NPR)
104     FORMAT(11X,1P12E10.3)
        RETURN
        END
C
        SUBROUTINE INITAL(X,XI,IPRINT)
        DIMENSION X(1),XI(1),DUM(4,4)
        DIMENSION D1(5,5),D2(5,5),D3(5,6),D4(5),ER(5),EI(5)
        COMPLEX EIG(5),EVEC(5,5),CNORM
```

```
      COMMON/NNCUT/NODEV
      COMMON/MAT/A(4,4),B(4,3),XK(3,4),NX,NU,NINT,NTOT
C
C  INITIALIZE TIME AND STATE
C
      DO 2 I = 1,NINT
2     X(I) = XI(I)
      TIME=0.
      CALL AUXVAR(X,TIME)
C
C  PRINT OUT A, B AND K MATRICES
C
      CALL WRTMAT(A,NX,NX,NX,'A         ')
      CALL WRTMAT(B,NX,NU,NX,'B         ')
      CALL WRTMAT(XK,NU,NX,NU,'K         ')
C
C  COMPUTE CLOSED-LOOP MATRIX
C
      DO 10 I=1,NX
      DO 10  J=1,NX
      TEMP=0
      DO 5 K=1,NU
5     TEMP=TEMP+B(I,K)*XK(K,J)
10    DUM(I,J)=TEMP+A(I,J)
      CALL WRTMAT(DUM,NX,NX,4,'A+B*K    ')
      TEMP = 0.
      DO 15 I = 1,NU
      DO 15 J = 1,NX
15    TEMP = TEMP + XK(I,J)**2
      TEMP = SQRT(TEMP)
      WRITE(NODEV,102)TEMP
102   FORMAT(/,10X,'||K|| =',G15.5)
C
C  COMPUTE CLOSED-LOOP EIGENSOLUTIONS
C
      IJOB=1
      CALL EIGRF(DUM,NX,4,IJOB,EIG,EVEC,5,D2,IER)
      DO 24 I=1,NX
      ER(I)=REAL(EIG(I))
      EI(I)=AIMAG(EIG(I))
      IF(EI(I).LT.0.)GO TO 24
      EMAX = 0.
      DO 20 J = 1,NX
      EMAG = CABS(EVEC(J,I))
      IF(EMAG.LE.EMAX)GO TO 20
      EMAX = EMAG
      CNORM = CONJG(EVEC(J,I))/EMAG**2
20    CONTINUE
      DO 22 J = 1,NX
      DI(J,I) = REAL(EVEC(J,I)*CNORM)
      IF(EI(I).GT.0.)DI(J,I+1) = AIMAG(EVEC(J,I)*CNORM)
22    CONTINUE
24    CONTINUE
      DO 30 I = 1,NX
      TEMP = 0.
```

```
      DO 29 J = 1,NX
28    TEMP = TEMP + D1(J,I)**2
30    D4(I) = SQRT(TEMP)
C
C  PRINT OUT E-VALUES AND E-VECTORS
C
      WRITE(NODEV,100)NX
100   FORMAT(/,' THE',I2,'TH ORDER A+BK MATRIX HAS EIGENSOLUTIONS',
     1       /,3X,'EVAL(REAL)',4X,'EVAL(IMAG)',4X,'E-VECTOR',/)
      DO 32 I = 1,NX
32    WRITE(NODEV,101)ER(I),EI(I),(D1(J,I),J=1,NX)
101   FORMAT(1X,2G14.6,10F10.6)
C
C  COMPUTE AND PRINT OUT THE ANGLES BETWEEN E-VECTORS
C
      DO 38 I = 2,NX
      I1 = I - 1
      DO 36 J = 1,I1
      TEMP = 0.
      DO 34 K = 1,NX
34    TEMP = TEMP + D1(K,I)*D1(K,J)
      ARG = TEMP/(D4(I)*D4(J))
36    D3(I,J) = 57.2958*ACOS(AMIN1(ABS(ARG),1.))
38    WRITE(6,104)(I,J,D3(I,J),J=1,I1)
104   FORMAT(/,5(' ANG',I1,',',I1,' =',F7.2))
      RETURN
      END
C
      SUBROUTINE XDOT(X,TIME,F)
      DIMENSION X(1),F(1)
      COMMON/MAT/A(4,4),B(4,3),XK(3,4),NX,NU,NINT,NTOT
C
C   COMPUTE CONTROLS U(1) , U(2)AND
C  X(6) = INTEGRAL(U(1)**2+U(2)**2)DT
C  F(6) = U(1)**2+U(2)**2
C
      TEMP = 0.
      DO 3 I=1,NU
      SUM=0.
      DO 4 J=1,NX
4     SUM=SUM+XK(I,J)*X(J)
      X(NINT+I)=SUM
3     TEMP = TEMP + SUM**2
      F(NX+2) = TEMP
C
C  COMPUTE STATES X(1) TO X(4)
C  AND INTEGRAL OF STATES SQUARED
C
      TEMP = 0.
      DO 1 I=1,NX
      SUM=0.
      DO 2 J=1,NX
2     SUM=SUM+A(I,J)*X(J)
      DO 5 K=1,3
5     SUM=SUM+B(I,K)*X(NINT+K)
```

```
      F(I)=SUM
1     TEMP = TEMP + X(I)**2
      F(NX+1) = TEMP
      RETURN
      END
C
      SUBROUTINE AUXVAR(X,TIME)
      DIMENSION X(1)
      COMMON/MAT/A(4,4),B(4,3),XK(3,4),NX,NU,NINT,NTOT
      DO 1 I=1,NU
      SUM=0.
      DO 2 J=1,NX
2     SUM=SUM+XK(I,J)*X(J)
1     X(NINT+1)=SUM
C
      RETURN
      END
C     FILE WRTMAT FORTRAN A1
C
C     5/8/79
C     FILE OF UTILITY SUBROUTINES TO SUPPORT VIBE FORTRAN A1
C
C     WRTMAT - GENERAL MATRIX OUTPUT SUBROUTINE
C
      SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
      DIMENSION A(IA,1)
      COMMON/NNOUT/NPRINT
      REAL*8 ANAME
      WRITE(NPRINT,100)ANAME,N,M
100   FORMAT(/,' MATRIX ',A8,3X,'(',I3,' ROWS X',I3,' COLS)')
      IF(M.LE.10)GO TO 15
      DO 10 I=1,N
10    WRITE(NPRINT,101)(A(I,J),J=1,M)
101   FORMAT(/,(1P10E13.5))
      RETURN
15    DO 20 I=1,N
20    WRITE(NPRINT,102)(A(I,J),J=1,M)
102   FORMAT(1P10E13.5)
      RETURN
      END
C
C       TRANSP - TRANSPOSES A MATRIX
C
        SUBROUTINE TRANSP(A,N1,N2,NA)
        DIMENSION A(NA,1)
        COMMON/NNOUT/NOUT
        M = MAXO(N1,N2)
        IF(M.GT.NA.OR.M.LE.0)GO TO 90
        M1 = M-1
        DO 10 I=1,M1
        I1 = I + 1
        DO 10 J=I1,M
        TEMP = A(I,J)
        A(I,J) = A(J,I)
10      A(J,I) = TEMP
```

```
      RETURN
90    WRITE(NOUT,100)N1,N2,NA
100   FORMAT(' *** ERROR IN TRANSP *** N1,N2,NA =',3I4)
      RETURN
      END
C
C      MULT - MATRIX MULTIPLICATION
C
      SUBROUTINE MULT(A,B,C,N,L,M,NA,NB,NC)
      DIMENSION A(NA,1),B(NB,1),C(NC,1)
      DOUBLE PRECISION TEMP
      COMMON/NNOUT/NOUT
      IF(N.GT.MINO(NA,NC).OR.L.GT.NB)GO TO 90
      DO 20 I=1,N
      DO 20 J=1,M
      TEMP = 0.
      DO 10 K=1,L
10    TEMP = TEMP + DBLE(A(I,K))*DBLE(B(K,J))
20    C(I,J) = TEMP
      RETURN
90    WRITE(NOUT,100)N,NA,NC,L,NB
100   FORMAT(' *** ERROR IN MULT *** N,NA,NC,L,NB=',5I5)
      RETURN
      END
C
C      MSHIFT - TRANSFERS VECTORS AND MATRICES
C
      SUBROUTINE MSHIFT(A,B,N,M,NA,NB)
      DIMENSION A(NA,1),B(NB,1)
      COMMON/NNOUT/NOUT
      IF(N.GT.MINO(NA,NB))GO TO 90
      DO 10 I=1,N
      DO 10 J=1,M
10    B(I,J) = A(I,J)
      RETURN
90    WRITE(NOUT,100)N,NA,NB
100   FORMAT(' *** ERROR IN MSHIFT *** N,NA,NB=',3I5)
      RETURN
      END
C
C      MATADD - MATRIX ADDITION
C
      SUBROUTINE MATADD(A,B,C,N,M,NA,NB,NC)
      DIMENSION A(NA,1),B(NB,1),C(NC,1)
      COMMON/NNOUT/NOUT
      IF(N.GT.MINO(NA,NB,NC))GO TO 90
      DO 10 I=1,N
      DO 10 J=1,M
10    C(I,J) = A(I,J) + B(I,J)
      RETURN
90    WRITE(NOUT,100)N,NA,NB,NC
100   FORMAT(' *** ERROR IN MATADD *** N,NA,NB,NC=',4I5)
      RETURN
      END
C
```

```
C       MATSUB - MATRIX SUBTRACTION
C
        SUBROUTINE MATSUB(A,B,C,N,M,NA,NB,NC)
        DIMENSION A(NA,1),B(NB,1),C(NC,1)
        COMMON/NNOUT/NOUT
        IF(N.GT.MINO(NA,NB,NC))GO TO 90
        DO 10 I=1,N
        DO 10 J=1,M
10      C(I,J) = A(I,J)   B(I,J)
        RETURN
90      WRITE(NOUT,100)N,NA,NB,NC
100     FORMAT(' *** ERROR IN MATSUB *** N,NA,NB,NC=',4I5)
        RETURN
        END
C
C        SWAP - INTERCHANGES TWO VARABLES
C
        SUBROUTINE SWAP(A,B)
        C = A
        A = B
        B = C
        RETURN
        END
C
C       MSMULT - MATRIX*SCALAR MULTIPLICATION
C
        SUBROUTINE MSMULT(S,A,N,M,NA)
        DIMENSION A(NA,1)
        DO 10 I=1,N
        DO 10 J=1,M
10      A(I,J) = S*A(I,J)
        RETURN
        END
C
C       ZERO - FILLS A MATRIX WITH ZEROS
C
        SUBROUTINE ZERO(A,N,M,NA)
        DIMENSION A(NA,1)
        DO 10 I=1,N
        DO 10 J=1,M
10      A(I,J) = 0.
        RETURN
        END
C
C
C
C       IMAT - LOADS AN ARRAY WITH THE IDENTITY MATRIX
C
        SUBROUTINE IMAT(A,N,NA)
        DIMENSION A(NA,1)
        DO 10 I=1,N
        DO 5 J=1,N
5       A(I,J) = 0.
10      A(I,I) = 1.
        RETURN
```

```
      END
C
C
C       ANORM - CALCULATES THE RSS NORM OF A MATRIX
C
      FUNCTION ANORM(A,N1,N2,NA)
      DIMENSION A(NA,1)
      COMMON/NNOUT/NOUT
      IF(N1.GT.NA)GO TO 90
      ANORM = 0.
      DO 10 I=1,N1
      DO 10 J=1,N2
10    ANORM = ANORM + A(I,J)**2
      ANORM = SQRT(ANORM)
      RETURN
90    WRITE(NOUT,100)N,NA
100   FORMAT(' *** ERROR IN ANORM *** N,NA =',2I5)
      RETURN
      END
C
C**********   SUBROUTINE PPLOT   **********
C
      SUBROUTINE PPLOT(X,Y,NPOINT,XYLIMS,NCHARS,NLINES)
C
C  PPLOT GENERATES A PRINTER PLOT UP TO 130 CHARACTERS WIDE
C  BY UP TO 80 LINES HIGH
C
C  PPLOT INPUTS:
C
C  X( )=ARRAY OF X-COORDINATES OF POINTS
C  Y( )=ARRAY OF Y-COORDINATES OF POINTS
C  NPOINT = TOTAL NUMBER OF POINTS IN CURVES
C  XYLIMS(4)=LIMITS OF X AND Y AXES IN THE ORDER XMIN,XMAX,YMIN,YMAX
C  IF XYLIMS( )=0,SCALING IS AUTOMATIC
C  NCHARS = WIDTH OF PRINTER PLOT (NUMBER OF CHARACTERS)
C  NLINES = LENGTH OF PRINTER PLOT (NUMBER OF LINES)
C
      DIMENSION X(1),Y(1),XYLIMS(1),XGRID(12),YGRID(12)
      INTEGER IFLD(130,80),ISTAR,IHOR,IVERT,IBLANK,ICROSS
      DATA ISTAR,IHOR,IVERT/'*   ','-   ','I   '/
      DATA IBLANK,ICROSS/'    ','+   '/
      COMMON/NNOUT/NODEV
      WRITE(NODEV,100)NPOINT,NCHARS,NLINES
100   FORMAT(/' PPLOT CALLED ... TOTAL NUMBER OF POINTS =',I8,
     1   //' NCHARS,NLINES =',2I4)
      IF(NPOINT.LE.0)RETURN
      IF(NCHARS.LE.130.AND.NLINES.LE.80)GO TO 1
      WRITE(NODEV,106)NCHARS,NLINES
106   FORMAT(/,' LIMITS EXCEEDED IN PPLOT.  NCHARS,NLINES =',2I10)
1     XMIN = XYLIMS(1)
      XMAX = XYLIMS(2)
      YMIN = XYLIMS(3)
      YMAX = XYLIMS(4)
      IF(XMIN.GE.XMAX)CALL MINMAX(X,NPOINT,XMIN,XMAX)
      IF(YMIN.GE.YMAX)CALL MINMAX(Y,NPOINT,YMIN,YMAX)
```

```
      CALL SCALE(XMIN,XMAX,XST,XDEL,NXP)
      XFIN = XST + XDEL*NXP
      DX = XFIN - XST
      IXZ = -XST/DX*NCHARS + 1.00000
      IXAXIS = 0
      IF(IXZ.GE.1.AND.IXZ.LE.NCHARS)IXAXIS = 1
      CALL SCALE(YMIN,YMAX,YST,YDEL,NYP)
      YFIN = YST + YDEL*NYP
      DY = YFIN - YST
      IYZ = YFIN/DY*NLINES + 1.000001
      IYAXIS = 0
      IF(IYZ.GE.1.AND.IYZ.LE.NLINES)IYAXIS = 1
C     WRITE(NODEV,105)XST,XFIN,YST,YFIN,IXZ,IYZ
105   FORMAT(/,' XST,XFIN,YST,YFIN =',1P4E12.4,
     1     //,' INDEX FOR X ANY Y ZERO REF AXES =',0P2I6)
      N1 = NCHARS - 1
      N2 = NLINES - 1
      DO 10 I = 2,N1
      DO 10 J = 2,N2
10    IFLD(I,J) = IBLANK
      DO 20 I = 1,NCHARS
      IF(IYAXIS.EQ.1)IFLD(I,IYZ) = IHOR
      IFLD(I,1) = IHOR
20    IFLD(I,NLINES) = IHOR
      DO 30 I = 1,NLINES
      IF(IXAXIS.EQ.1)IFLD(IXZ,I) = IVERT
      IFLD(1,I) = IVERT
30    IFLD(NCHARS,I) = IVERT
      NX = NXP + 1
      NY = NYP + 1
      DO 45 I = 1,NX
      XGRID(I) = XST + (I-1)*XDEL
      IX = ((I-1)*XDEL/DX)*NCHARS + 1.00001
      IX = MINO(IX,NCHARS)
      DO 45 J = 1,NY
      YGRID(J) = YST + (J-1)*YDEL
      IY = ((J-1)*YDEL/DY)*NLINES + 1.00001
      IY = MINO(IY,NLINES)
45    IFLD(IX,IY) = ICROSS
      WRITE(NODEV,101)(XGRID(I),I=1,NX)
101   FORMAT(/,' X-GRID =',7F10.3,/,9X,7F10.3)
      WRITE(NODEV,102)(YGRID(I),I=1,NY)
102   FORMAT(/,' Y-GRID =',7F10.3,/,9X,7F10.3)
      DO 50 I = 1,NPOINT
      IX = ((X(I)-XST)/DX)*NCHARS + 1
      IY = ((YFIN-Y(I))/DY)*NLINES + 1
C     IF(I.LE.50)WRITE(NODEV,107)X(I),Y(I),IX,IY
107   FORMAT(' X(I),Y(I) =',1P2E12.4,'    IX,IY =',0P2I6)
      IX = MAXO(MINO(IX,NCHARS),1)
      IY = MAXO(MINO(IY,NLINES),1)
50    IFLD(IX,IY) = ISTAR
      WRITE(NODEV,103)
103   FORMAT(1H1)
      DO 60 IY = 1,NLINES
60    WRITE(NODEV,104)(IFLD(IX,IY),IX=1,NCHARS)
```

```
104   FORMAT(1X,130A1)
      RETURN
C     DEBUG UNIT(6),SUBCHK,TRACE,INIT
C     AT 20
C     TRACE ON
C     AT 60
C     TRACE OFF
      END
C
C**********   SUBROUTINE MINMAX   **********
C
      SUBROUTINE MINMAX(X,N,XMIN,XMAX)
      DIMENSION X(1)
      XMIN = X(1)
      XMAX = X(1)
      IF(N.LE.1)RETURN
      DO 10 I=2,N
      XMAX = AMAX1(XMAX,X(I))
10    XMIN = AMIN1(XMIN,X(I))
      RETURN
      END
C
C**********   SUBROUTINE SCALE   **********
C
      SUBROUTINE SCALE(XMIN,XMAX,START1,DEL1,NPTS1)
      INTX(X) = IFIX(X-.5+SIGN(.5,X))
      IF(XMAX.LE.XMIN)GO TO 90
      XSIZE = XMAX - XMIN
      XEXP = ALOG10(XSIZE)
      IEXP = INTX(XEXP)
      XORD = 10.**IEXP
      XNORM = XSIZE/XORD
      IF(XNORM.LE.1.6)XMOD = .2
      IF(XNORM.GT.1.6.AND.XNORM.LE.4.)XMOD = .5
      IF(XNORM.GT.4..AND.XNORM.LE.8.)XMOD = 1.
      IF(XNORM.GT.8.)XMOD = 2.
      DEL1 = XORD*XMOD
      DO 10 I=1,30
      XMAG = 10.**(I-15)
      XPOINT = FLOAT(INTX(XMAG*XMAX))/XMAG
      IF(XPOINT.GE.XMIN)GO TO 20
10    CONTINUE
      GO TO 90
20    ISHIFT = (XPOINT-XMIN)/DEL1
      START1 = XPOINT - DEL1*ISHIFT
      IF(START1.GT.XMIN)START1 = START1 - DEL1
      NPTS1 = (XMAX-START1)/DEL1
      IF(START1+DEL1*(FLOAT(NPTS1)+.01).LT.XMAX)NPTS1 = NPTS1 + 1
      RETURN
90    WRITE(6,100)XMIN,XMAX
100   FORMAT(//' ERROR IN SCALE   XMIN,XMAX =',2E12.4)
      RETURN
      END
C
C**********   SUBROUTINE APLOT    VERSION 1   11/26/80   *********
```

```
C
      SUBROUTINE APLOT(X,Y,NPT,NPLOT,XYLIMS,WIDTH,HEIGHT,TICKL,
     1  NCASE,XLBL,NXC,YLBL,NYC,TITL,NTTL,CNAMES,NCINDX)
C
C   NOTE    THE X( ), Y( ) ARRAYS ARE UNCHANGED IN THIS SUBROUTINE
C
C   APLOT INPUTS:
C
C   X( )=ARRAY OF X-COORDINATES OF POINTS
C   Y( )=ARRAY OF Y COORDINATES OF POINTS
C   NPT( )=NO. OF POINTS(X,Y PAIRS)IN EACH CURVE
C   NPLOT=NUMBER OF CURVES
C   XYLIMS(4)=LIMITS OF X AND Y AXES IN THE ORDER XMIN,XMAX,YMIN,YMAX
C   IF XYLIMS( )=0,SCALING IS AUTOMATIC
C   WIDTH=WIDTH OF PLOT IN INCHES
C   HEIGHT=HEIGHT OF PLOT IN INCHES
C   TICKLE=LENGTH OF TICK MARKS ON AXES IN INCHES AND HEIGHT OF SCALE NOS
C       IN INCHES.  IF TICKL IS NEGETIVE,PUT THE TICK MARKS INSIDE AXES
C   NCASE=A NO. PRINTED AT THE UPPER RIGHT HAND CORNER OF PLOT FOR
C       INDEXING PURPOSES.
C   XLBL( ) = ALPHANUMERIC STRING FOR X-AXIS LABEL
C   NXC = NUMBER OF CHARACTERS IN X AXIS LABEL
C   YLBL( ), NYC = LIKEWISE FOR Y AXIS LABEL
C   TITL( ) = ALPHANUMERIC STRING FOR TITLE
C   NTTL = NUMBER OF CHARACTERS IN TITLE
C   CNAMES( ) = ARRAY OF FOUR-CHARACTER IDENTIFIERS FOR THE NPLOT CURVES
C   NCINDX( ) = INTEGER (.GE.1. AND .LE.NPT( )) INDICATING LOCATION
C               OF CNAMES IDENTIFIERS FOR EACH CURVE
C
C
      DIMENSION X(1),Y(1),XAR(10),NPT(1),XYLIMS(1),LSCALE(2)
      DIMENSION XLBL(1),YLBL(1),TITL(1),CNAMES(1),NCINDX(1)
      COMMON/NNOUT/NODEV
      DATA EPS/1.E-8/
      WRITE(NODEV,100)NPLOT,NCASE,(NPT(I),I=1,NPLOT)
100   FORMAT(/' APLOT CALLED   NPLOT =',I5,3X,'NCASE =',I3,3X,'NPT( ) ='
     1    ,10I4,/,(10X,10I4))
C       DO 110 I=1,10
C       NNPT = NPT(I)
C110    WRITE(NODEV,109) I,X(I),Y(I+NNPT),Y(I+202),Y(I+303),Y(I+404)
C109    FORMAT(2X,I5,'COORDINATES=', 6F10.4)
      IF(NPLOT.LE.0)RETURN
      TICK = ABS(TICKL)
      H1 = 12*TICK
      H2 = H1 + WIDTH
      V1 = 9*TICK
      V2 = V1 + HEIGHT
C
C   FIND TOTAL NUMBER OF POINTS ON PLOT
C
      NPOINT = 0
      DO 2 I=1,NPLOT
2     NPOINT = NPOINT + NPT(I)
C
C   SCALE X-AXES
```

```
C
      XMIN = XYLIMS(1)
      XMAX = XYLIMS(2)
      IF(XYLIMS(1).GE.XYLIMS(2))CALL MINMAX(X,NPOINT,XMIN,XMAX)
      CALL SCALE(XMIN,XMAX,XST,XDEL,NXP)
      XFIN = XST + XDEL*NXP
      DX = XFIN - XST
      XF = (H2-H1)/DX
      WRITE(NODEV,101)XMIN,XMAX,XST,XDEL,XFIN,NXP
101   FORMAT(/' XMIN,XMAX =',1P2E14.7,5X,'XSTART,XDEL,XFINISH =',
     1  3E10.3,5X,'NXP =',I4)
C
C    SCALE Y-AXES
C
      YMIN = XYLIMS(3)
      YMAX = XYLIMS(4)
      IF(XYLIMS(3).GE.XYLIMS(4))CALL MINMAX(Y,NPOINT,YMIN,YMAX)
      CALL SCALE(YMIN,YMAX,YST,YDEL,NYP)
      YFIN = YST + YDEL*NYP
      DY = YFIN - YST
      YF = (V2-V1)/DY
      WRITE(NODEV,102)YMIN,YMAX,YST,YDEL,YFIN,NYP
102   FORMAT(/' YMIN,YMAX =',1P2E14.7,5X,'YSTART,YDEL,YFINISH =',
     1  3E10.3,5X,'NYP =',I4)
C
C    DRAW OUTER BORDER FOR PLOTS
C
      CALL PLOT(H1,V1,3)
      CALL PLOT(H2,V1,2)
      CALL PLOT(H2,V2,2)
      CALL PLOT(H1,V2,2)
      CALL PLOT(H1,V1,2)
C
C    IF APPROPRIATE ADD X = 0, Y = 0 AXES
C
      IF(XST.GT.0..OR.XFIN.LT.0.)GO TO 12
      XX = -XST*XF + H1
      CALL PLOT(XX,V1,3)
      CALL PLOT(XX,V2,2)
12    IF(YST.GT.0..OR.YFIN.LT.0.)GO TO 14
      YY = -YST*YF + V1
      CALL PLOT(H1,YY,3)
      CALL PLOT(H2,YY,2)
14    CONTINUE
C
C    ADD TITLE, X-AXIS LABEL AND Y-AXIS LABEL
C
      HT = TICK
      ANG = 0.
      ANG90 = 90.
      WD = .9*HT
      XP = (H1+H2)/2. - .5*NTTL*1.5*WD
      IF(NTTL.GT.0)CALL SYMBOL(XP,V2+4.*HT,1.5*HT,TITL,ANG,NTTL)
      XP = (H1+H2)/2. - .5*NXC*WD
      IF(NXC.GT.0)CALL SYMBOL(XP,V1-3.*HT,HT,XLBL,ANG,NXC)
```

```
      YP = (V1+V2)/2. - .5*NYC*WD
      IF(NYC.GT.0)CALL SYMBOL(H1-11.*HT,YP,HT,YLBL,ANG90,NYC)
C
C     ADD CASE NUMBER (IF NONZERO) AND DATE
C
      NDG = 0
      IF(NCASE.GT.0)CALL NUMBER(H2+HT,V2,HT,FLOAT(NCASE),ANG,NDG)
      CALL DATIMP(H1,V2+2*HT,HT,'APLOT   ')
C
C     ADD X-AXIS TICK MARKS AND SCALE
C
      N = NXP + 1
      YY = V1 - 4.*TICK
      DO 22 I=1,N
      XA = H1 + (I-1)*XDEL*XF
      CALL PLOT(XA,V1,3)
      CALL PLOT(XA,V1-TICKL,2)
      XR = XST + (I-1)*XDEL
      XR = XR + SIGN(EPS,XR)
      NDGT = MIN0(MAX0(0,-IFIX(ALOG10(ABS(XR)))+3),8)
      IF(ABS(XR) .LE. 2.*EPS) NDGT=0
22    CALL NUMBER(XA-2.*TICK,YY,HT,XR,ANG,NDGT)
C
C     ADD Y-AXIS TICK MARKS AND SCALE
C
      N = NYP + 1
      XX = V1 - 7.*TICK
      DO 26 I=1,N
      YA = V1 + (I-1)*YDEL*YF
      CALL PLOT(H1,YA,3)
      CALL PLOT(H1-TICKL,YA,2)
      YY = YA
      YR = YST + (I-1)*YDEL
      YR = YR + SIGN(EPS,YR)
      NDGT = MIN0(MAX0(0,-IFIX(ALOG10(ABS(YR)))+3),8)
      IF(ABS(YR) .LE. 2.*EPS) NDGT=0
26    CALL NUMBER(XX,YY,HT,YR,ANG,NDGT)
C
C     ADD LABEL FOR EACH CURVE
C
      K = 0
      DO 30 I=1,NPLOT
      IF(NCINDX(I).LE.0)GO TO 32
      JPT = MAX0(MIN0(NCINDX(I),NPT(I)),1)
      XX = AMIN1(AMAX1(H1,(X(K+JPT)-XST)*XF+H1),H2)
      YY = AMIN1(AMAX1(V1,(Y(K+JPT)-YST)*YF+V1),V2)
      CALL PLOT(XX,YY,3)
      CALL PLOT(XX+2.*HT,YY+4.*HT,2)
      CALL SYMBOL(XX+3.*HT,YY+4.*HT,HT,CNAMES(I),ANG,4)
32    N = NPT(I) - 1
C
C     PLOT EACH CURVE
C
      K = K + 1
      XX = AMIN1(AMAX1(H1,(X(K)-XST)*XF+H1),H2)
```

```
      YY = AMIN1(AMAX1(V1,(Y(K)-YST)*YF+V1),V2)
      CALL PLOT(XX,YY,3)
      DO 30 J=1,N
      K = K + 1
      XX = AMIN1(AMAX1(H1,(X(K)-XST)*XF+H1),H2)
      YY = AMIN1(AMAX1(V1,(Y(K)-YST)*YF+V1),V2)
      CALL PLOT(XX,YY,2)
30    CONTINUE
      RETURN
      END
C
      SUBROUTINE SCALE(XMIN,XMAX,START1,DEL1,NPTS1)
      INTX(X) = IFIX(X-.5+SIGN(.5,X))
      IF(XMAX.LE.XMIN)GO TO 90
      XSIZE = XMAX - XMIN
      XEXP = ALOG10(XSIZE)
      IEXP = INTX(XEXP)
      XORD = 10.**IEXP
      XNORM = XSIZE/XORD
      IF(XNORM.LE.1.6)XMOD = .2
      IF(XNORM.GT.1.6.AND.XNORM.LE.4.)XMOD = .5
      IF(XNORM.GT.4..AND.XNORM.LE.8.)XMOD = 1.
      IF(XNORM.GT.8.)XMOD = 2.
      DEL1 = XORD*XMOD
      DO 10 I=1,30
      XMAG = 10.**(I-15)
      XPOINT = FLOAT(INTX(XMAG*XMAX))/XMAG
      IF(XPOINT.GE.XMIN)GO TO 20
10    CONTINUE
      GO TO 90
20    ISHIFT = (XPOINT-XMIN)/DEL1
      START1 = XPOINT - DEL1*ISHIFT
      IF(START1.GT.XMIN)START1 = START1 - DEL1
      NPTS1 = (XMAX-START1)/DEL1
      IF(START1+DEL1*(FLOAT(NPTS1)+.01).LT.XMAX)NPTS1 = NPTS1 + 1
      RETURN
90    WRITE(6,100)XMIN,XMAX
100   FORMAT(//' ERROR IN SCALE   XMIN,XMAX =',2E12.4)
      RETURN
      END
C
      SUBROUTINE MINMAX(X,N,XMIN,XMAX)
      DIMENSION X(1)
      XMIN = X(1)
      XMAX = X(1)
      IF(N.LE.1)RETURN
      DO 10 I=2,N
      XMAX = AMAX1(XMAX,X(I))
10    XMIN = AMIN1(XMIN,X(I))
      RETURN
      END
C
C**********   SUBROUTINE DATIME   **********
C
      SUBROUTINE DATIME(IMO,IDAY,IYR,IHOURS,IMIN,AMPM,PNAME,NODEV)
```

```
      REAL*8 PNAME
      DATA AM/'  AM'/,PM/'  PM'/
      CALL DATE(IMO,IDAY,IYR)
      CALL STIME(ITIME)
      XHOURS = FLOAT(ITIME)/10000.
      AMPM = AM
      IF(XHOURS.GE.12.)AMPM = PM
      IF(XHOURS.GE.13.)XHOURS = XHOURS - 12.
      IHOURS = XHOURS
5     XMIN = (XHOURS - IHOURS)*60.
      IMIN = XMIN
      IF(NODEV.GT.0)WRITE(NODEV,100)PNAME,IMO,IDAY,IYR,IHOURS,IMIN,AMPM
100   FORMAT(/' TIME IN ',A8,' IS ',A2,'/',A2,'/',A2,5X,I2,':',
     1   I2,3X,A4)
      RETURN
      END
C
C**********   SUBROUTINE DATIMP   **********
C
      SUBROUTINE DATIMP(XP,YP,HT,PNAME)
      REAL*8 PNAME
      COMMON/NNOUT/NODEV
      ANG = 0.
      WD = .9*HT
      CALL DATIME(IMO,IDAY,IYR,IHOURS,IMIN,AMPM,PNAME,NODEV)
      XHOURS = IHOURS
      XMIN = IMIN
      CALL SYMBOL(XP,YP,HT,IMO,ANG,2)
      CALL SYMBOL(XP+2.*WD,YP,HT,1H/,ANG,1)
      CALL SYMBOL(XP+3.*WD,YP,HT,IDAY,ANG,2)
      CALL SYMBOL(XP+5.*WD,YP,HT,1H/,ANG,1)
      CALL SYMBOL(XP+6.*WD,YP,HT,IYR,ANG,2)
      CALL NUMBER(XP+12.*WD,YP,HT,XHOURS,ANG,-1)
      CALL SYMBOL(XP+14.*WD,YP,HT,1H:,ANG,1)
      CALL NUMBER(XP+15.*WD,YP,HT,XMIN,ANG,-1)
      CALL SYMBOL(XP+18.*WD,YP,HT,AMPM,ANG,4)
      RETURN
      END
C
C    FILE INTODE1 INPUT
C
C     INPUT FOR MONTGOMERY'S AIRCRAFT LATERAL SIMULATION
C   JUL 29, 1980
C
 &INTODE
  XI = 0.,.1,0.,.1,6*0.,
  TMAX = 5.,
  IPRINT = 2,
  NODEV = 6,
  DT = .025,
  TPRINT = .05,
  NXPR = 1,2,3,4,5,6,7,8,2*0,
     A=-3.67984E-1,1.E0,-2.4209E-2,2.58819E-1,3*0.,1.7335E-2,
    -3.2279E-2,2.67949E-1,-1.10395E-1,-9.65926E-1,2.61875E1,0.,4.461072E-2
     B=-7.67183,0.,1.96959,0.,2.06549,0.,-2.33943E0,0.,
```

```
 XK = 1.4941162E-01,-4.1928029E-01,1.0535228E-01,2.0296771E-02,1.3E+00,
    3.6583920E+00,-2.3151433E-01,-4.7954880E-02,
    NX=4,
    NU=2,
 IPLOT = 2,
 NXPL = 2,4,7,9,6*0,
 NCHARS = 80, NLINES = 48,
 NCHARS = 130,NLINES = 62,
 XNAMES=4HRORA,4HBANK,4HYAWR,4H YAW,4HINX2,4HINU2,4HAILE,4HRUDD,42*0.,
 XLBLS(1,1) = 4HROLL, 4H RAT, 4HE (R, 4HAD/S, 4HEC) ,
 XLBLS(1,2) = 4HBANK, 4H ANG, 4HLE (, 4HRAD),
 XLBLS(1,3) = 4HYAW , 4HRATE, 4H (RA, 4HD/SE, 4HC)  ,
 XLBLS(1,4) = 4HYAW , 4HANGL, 4HE (R, 4HAD) ,
 XLBLS(1,5) = 4HINTE, 4HGRAL, 4HXT*X,
 XLBLS(1,6) = 4HINTE, 4HGRAL, 4HUT*U,
 XLBLS(1,7) = 4HAILE, 4HRON , 4H(RAD, 4H)   ,
 XLBLS(1,8) = 4HRUDD, 4HER (, 4HRAD),
 NINT = 6,
 NTOT = 9,
 XPLBL = 4HTIME, 4H (SE, 4HC)  ,7*0.,
 NXC = 10,
 YPLBL = 4HSTAT, 4HE   , 8*0.,
 NYC = 5,
    TITL=4HMONT,4HGOME,4HRY  ,4HA/C ,4HLATE,4HRAL ,4HDYNA,4HMICS,,4H K )
    4H TAU,4H=   ,
    TITL(13)=4H1.0 ,
    NTTL=52,
 WIDTH = 5.5, HEIGHT = 2., TICKL = .06,
 ICONT = 1,
&END
&INTODE
    TITL(13)=4H0.5 ,
XK = 1.7022794E-01,-6.0164034E-01,2.6196051E-01,-2.7018290E-02,2.9E+00,
    4.0587797E+00,-1.1856604E+00,4.8925018E-01,
&END
&INTODE
    TITL(13)=4H0.25,
XK = 1.2836323E+00,-3.1967741E-01,8.5040188E-01,-1.5637993E+01,1.9E+00,
    4.4778833E+00,3.2358761E+00,-1.5113544E+00,
&END
&INTODE
    TITL(9)=4H(ROB,
    TITL(10)=4HUST),
         TITL(13)=4H1.0 ,
    XK=.65132,-.385575,.082589,-.192548,1.60285,3.46768,-.7396,-.
&END
&INTODE
    TITL(13)=4H0.5 ,
    XK=.64359,-.38107,.6764,-.244,2.07949,3.5064,-.741,-.432244,
&END
&INTODE
    TITL(13)=4H0.25,
    XK=1.0368,-.3599,1.27027,-.322,2.05683,3.81589,-.743469,-.520
&END
&INTODE
```

```
      TITL(9)=4H(LQR,
      TITL(10)=4H),RH,
      TITL(11)=4HO=   ,
      TITL(12)=4H100.,
      NTTL=48,
      XK=.3306,-.1784,.10918,-.03913,-.69998,.48287,2.6222,-1.4068,
 &END
 &INTODE
      TITL(12)=4H1.   ,
      XK=1.2588,-.17087,1.0046,.0953,-.30686,1.0189,5.9393,-1.734,
 &END
 &INTODE
      ICONT=0,
      DT=.01,
      TITL(12)=4H0.04,
      XK=5.17,-.059375,4.8719,1.1309,.106,5.06 4,7.2623,-2.5309,
 &END
C
C    FILE INTODE INPUT
C
C     INPUT FOR HALL'S AIRCRAFT LATERAL SIMULATION
C   JUL 31, 1980
C
 &INTODE
  XI = 0.,.1,0.,.1,6*0.,
  TMAX = 5.,
  IPRINT = 2,
  NODEV = 6,
  DT = .025,
  TPRINT = .1,
  NPR = 1,2,3,4,5,6,7,8,9,0,
     A=-3.18,1.,-.06,.022,3*0.,.0644,.63,0.,-.27,-.998,-10.6,0.,4.5,
     B=-14.4,3*0.,1.5,0.,-2.59,.037,2*0.,-.96,0.,
  XK =-3.9732631E-02,-4.3598451E-03,1.9987654E-02,1.2020696E-02,-959E-01,
     2.5593357E+00,2.1611822E-01,-3.6611261E-03,4.7516279E+00,-9.4E-01,
    -5.4777920E-01,-5.1410699E-01,
     NX=4,
     NU=3,
  IPLOT = 2,
  NXPL = 2,4,7,8,9,5*0
  NCHARS = 80, NLINES = 48,
  NCHARS = 130,NLINES = 62,
  XNAMES = 4HRORA, 4HBANK, 4HYAWR,4H YAW,4HINX2,4HINU2, 4HAILE,4HRYAWC,0,
  XLBLS(1,1) = 4HROLL, 4H RAT, 4HE (R, 4HAD/S, 4HEC) ,
  XLBLS(1,2) = 4HBANK, 4H ANG, 4HLE (, 4HRAD),
  XLBLS(1,3) = 4HYAW , 4HRATE, 4H (RA, 4HD/SE, 4HC)  ,
  XLBLS(1,4) = 4HYAW , 4HANGL, 4HE (R, 4HAD) ,
  XLBLS(1,5) = 4HINTE, 4HGRAL, 4HXT*X,
  XLBLS(1,6) = 4HINTE, 4HGRAL, 4HUT*U,
  XLBLS(1,7) = 4HAILE, 4HRON , 4H(RAD, 4H)   ,
  XLBLS(1,8) = 4HRUDD, 4HER (, 4HRAD),
  XLBLS(1,9) = 4HYAW , 4HCONT, 4HROL , 4H(RAD, 4H)   ,
  NTOT = 9,
  NINT = 6,
  XPLBL = 4HTIME, 4H (SE, 4HC)  ,7*0.,
```

```
 NXC = 10,
 YPLBL = 4HSTAT, 4HE    , 8*0.,
 NYC = 5,
    TITL=4HHALL,4H AIR,4HCRAF,4HT LA,4HTERA,4HL DY,4HNAMI,4HCS(M,4HIN K,
    4H) TA,4HU=  ,
    TITL(12)=4H 1.0,
 NTTL = 48,
 WIDTH = 5.5, HEIGHT = 2., TICKL = .06,
 ICONT = 1,
&END
&INTODE
    TITL(12)=4H0.5 ,
  XK=.02469,-.001394,-.01166,.24055,.0384,-.0438,-.30165,.02299,4.14655,
    .0114765,-.3113,
&END
&INTODE
    TITL(12)=4H0.25,
XK = 3.5695368E-01,-3.9497081E-02,-1.2787379E-02,1.4500313E+00,2.9E-01,
   -8.7618792E-01,-1.2922382E+00,2.6900148E+00,3.6100111E+00,-4.1E+00,
   -1.2453620E+01,1.4150048E+01,
&END
&INTODE
    TITL(8)=4HCS(R,
    TITL(9)=4HOBUS,
    TITL(10)=4HT) T,
    TITL(11)=4HAU=  ,
    TITL(12)=4H 1.0,
  XK=-.03489,.1519,-2.633,.137833,-.933,2.535,.2175,.002644,5.530
    -.96733,-.6965,-.525889,
&END
&INTODE
    TITL(12)=4H 0.5,
    XK=.03819,-.0016245,-.012548,.4133,.0386788,-.04435,-.27592,
    .0253255,4.15544,-.12295,.007238,-.3055,
&END
&INTODE
    TITL(12)=4H0.25,
    XK=.49974,.309131,-3.344,1.46229,-.932129,-5.79063,.112387,-.26,
    5.97918,-1.04167,-2.05667,-1.36633,
&END
&INTODE
    TITL(8)=4HCS(L,
    TITL(9)=4HQR) ,
    TITL(10)=4HRHO=,
    TITL(11)=4H100.,
    NTTL=44,
    XK=.04513,.00574,.003798,.0916,.0222,.011679,.056979,.09242,
    .03633,-.0754,-.0208,-.008925,
&END
&INTODE
    TITL(11)=4H 1.0,
    DT=0.01,
    TPLOT=0.02,
    XK=.8653,-.067577,.007833,.99586,-.058595,.016246,.1175,.93989,
     .35148,-.55619,-.27992,-.1,
```

```
&END
&INTODE
     ICONT=0,
     TITL(11)=4HO.04,
     XK=4.8341,-.4264,.02787,4.98,-.5015,.01237,.41809,4.748,1.7639,
     -.73614,-3.153,-.86026,
&END
```

```
C
C      FILE PERTB FORTRAN A1
C      PROGRAM TO TEST ROBUSTNESS OF LINEAR CONTROL SYSTEMS
C
       DOUBLE PRECISION DSEED
       DIMENSION EVR1(4),EVI1(4),EVEC1(4,4),EVR2(4),EVI2(4),EVEC2(4,4)
       DIMENSION XX(4004),YY(4004),XYL(4),NPTS(1)
       DIMENSION CNAMES(1),XLBL(10),YLBL(10),TITL(20),NCINDX(1)
       COMMON/MAT/A(4,4),B(4,3),XK(3,4),CA(4,4),PA(4,4),PB(4,3),N,M
       COMMON/ISEED/DSEED,RAND(28),P
       COMMON/NNOUT/NOUT
       NAMELIST/PERT/N,M,ICONT,A,B,XK,P,NSAMP,IPRINT,IPLOT,
      1XLBL,NXC,YLBL,NYC,TITL,NTTL,CNAMES,NCINDX
C
C      INPUT VARIABLES IN THE NAMELIST
C
C      N  =  NO. OF STATE VARIABLES
C      M  =  NO. OF CONTROL VARIABLES
C      ICONT  =  INDEX TO CONTINUE OR STOP MULTIPLE CASE RUNS
C           0 , STOP AFTER THIS CASE
C           1 , CONTINUE TO NEXT CASE
C      A  =  N BY N SYSTEM MATRIX
C      B  =  N BY M CONTROL MATRIX
C      XK  =  M BY N FEEDBACK MATRIX
C      P  =  FRACTION GIVING THE MAXIMUM PERTURBATION
C      NSAMP  =  NO. OF SAMPLES
C      IPRINT  =  INDEX TO DETERMINE THE AMOUNT OF PRINTOUT
C      IPLOT  =  INDEX THAT CONTROLS THE GENERATION OF PLOTS
C      XLBL  =  ARRAY CONTAINING THE X-AXIS LABEL FOR PLOTS
C      NXC  =  NO. OF CHARACTERS (LETTERS) IN X-AXIS LABEL
C      YLBL  =  ARRAY CONTAINING THE Y-AXIS LABEL FOR PLOTS
C      NYC  =  NO. OF CHARACTERS IN Y-AXIS LABEL
C      TITL  =  ARRAY CONTAINING A LABEL TO APPEAR IN PLOTS
C      NTTL  =  NO. OF CHARACTERS IN TITLE
C
       DATA NCASE/0/,ISTART/1/
       DATA XYL/-10.,0.,-3.,3./
       DATA WIDTH,HEIGHT,TICKL/5.5,3.3,.06/
999    READ(7,PERT)
       DSEED=13D0
       NCASE=NCASE+1
       NOUT=6
       NPTS(1)=NSAMP*N+N
       NCURV=1
C
C      PRINT BANNER
C
       WRITE(6,100) NCASE
100    FORMAT(1H1,'****PROGRAM PERTURB,  CASE',I5,'****')
C      WRITE(6,PERT)
        CALL DATIME(IM,ID,IY,IH,IMN,AP,'PERTB   ',6)
       WRITE(6,101) TITL
101    FORMAT(/1X,20A4//)
C
C      PRINT MATRICES A,B,K AND A+B*K
```

```
C
      CALL WRTMAT(A,N,N,N,'A=       ')
      CALL WRTMAT(B,N,M,N,'B=       ')
      CALL WRTMAT(XK,M,N,M,'K=       ')
      CALL MATM(A,B,XK,CA,N,M)
      I=1
      CALL WRTMAT(CA,N,N,N,'A+B*K    ')
C
C     CALCULATE EIGENVALUES AND EIGENVECTORS
C
      CALL GENE(CA,N,EVR1,EVI1,EVEC1,I,IPRINT)
      IP=1
      IF(IPLOT .GT. 0) CALL XYSTO(XX,YY,EVR1,EVI1,N,IP)
C
C     START MONTE CARLO RUNS
C
      DO 10 I=1,NSAMP
      CALL PERTB
      CALL MATM(PA,PB,XK,CA,N,M)
      IF((I-1)/IPRINT*IPRINT .NE. I-1) GO TO 20
      CALL WRTMAT(PA,N,N,N,'PERT A= ')
      CALL WRTMAT(PB,N,M,N,'PERT B= ')
      CALL WRTMAT(CA,N,N,N,'PRT A+BK')
 20   CALL GENE(CA,N,EVR2,EVI2,EVEC2,I,IPRINT)
      IF(IPLOT .GT. 0) CALL XYSTO(XX,YY,EVR2,EVI2,N,IP)
C
C     CALCULATE AND PRINT STATISTICS ON REMAX , REMIN  AND RTOMAX
C
      CALL STATS(EVR1,EVI1,EVR2,EVI2,N,I,NSAMP,IPRINT)
10    CONTINUE
      IF(IPLOT .LE. 1) GO TO 99
      IF(ISTART .GE. 1) CALL PLOTS(0,0,50)
      IF(ISTART .LE. 0) CALL PLOT(WIDTH+2.,0.,-3)
      ISTART=0
C
C     DO PLOTTING
C
      CALL APLOT(XX,YY,NPTS,NCURV,XYL,WIDTH,HEIGHT,TICKL,NCASE,
     1   XLBL,NXC,YLBL,NYC,TITL,NTTL,CNAMES,NCINDX)
C
C     GO TO NEXT CASE IF ICONT IS GRATER THAN ZERO
C
99    IF(ICONT .GE. 1)  GO TO 999
      IF(IPLOT .GT. 0) CALL PLOT(0.,0.,999)
      STOP
      END
C
C     MATM - COMPUTES CA = A + B * XK
C            WHERE A ,  B AND XK ARE GIVEN MATRICES
C
      SUBROUTINE MATM(A,B,XK,CA,N,M)
      DIMENSION A(N,N),B(N,M),XK(M,N),CA(N,N)
      DO 10 I=1,N
      DO 10 J=1,N
      TEMP=0.
```

```
      DO 5 K=1,M
5     TEMP=TEMP+B(I,K)*XK(K,J)
10    CA(I,J)=TEMP+A(I,J)
      RETURN
      END
C
C     PERTB - RANDOMLY PERTURBS THE ELEMENTS OF MATRICES A AND B
C
      SUBROUTINE PERTB
      DOUBLE PRECISION DSEED
      COMMON/MAT/A(4,4),B(4,3),XK(3,4),CA(4,4),PA(4,4),PB(4,3),N,M
      COMMON/ISEED/DSEED,RAND(28),P
      NN=N*(N+M)
      CALL GGUBS(DSEED,NN,RAND)
      DO 6 I=1,NN
6     RAND(I)=(-1+2*RAND(I))
      NNN=0
      DO 1 I=1,N
C
C     PERTURB MATRIX A
C
      DO 2 J=1,N
      NNN=NNN+1
      PA(I,J)=A(I,J)
      IF(A(I,J) .EQ. 0. .OR. A(I,J) .EQ. 1.) GO TO 2
      PA(I,J)=A(I,J)*(1.+RAND(NNN)*P)
2     CONTINUE
C
C     PERTURB MATRIX B
C
      DO 3 K=1,M
      NNN=NNN+1
      PB(I,K)=B(I,K)
      IF(B(I,K) .EQ. 0. .OR. B(I,K) .EQ. 1.) GO TO 3
      PB(I,K)=B(I,K)*(1.+RAND(NNN)*P)
3     CONTINUE
1     CONTINUE
      RETURN
      END
C
C     STATS - COMPUTES AND PRINTS STATISTICS ON REMAX , REMIN
C             AND RTOMAX
C
      SUBROUTINE STATS(EVR1,EVI1,EVR2,EVI2,N,IS,NSAMP,IPRINT)
      REAL*8 ANAME1,ANAME2,ANAME3
      DIMENSION EVR1(N),EVI1(N),EVR2(N),EVI2(N),RT(4)
      DIMENSION CREX(1000),CREM(1000),CRTX(1000),NY(20)
      DIMENSION CREXA(1000),CREMA(1000),CRTXA(1000)
      DATA RMAXMA,RMINMA,RTMXMA,SRXA,SRMA,SRTXA/6*0./
      DATA RMAXM,RMINM,RTMXM,SRX,SRM,SRTX/6*0./
      DATA ANAME1,ANAME2,ANAME3/' REMAX  ',' REMIN  ',' RTOMAX '/
      IF(IS .NE. 1) GO TO 3
      CALL MINMAX(EVR1,N,REMIN1,REMAX1)
      DO 1 I=1,N
1     RT(I)=EVI1(I)/EVR1(I)
```

```
      RTMAX1=AMAX1(RT(1),RT(2),RT(3),RT(4))
3     CALL MINMAX(EVR2,N,REMIN2,REMAX2)
      DO 2 I=1,N
2     RT(I)=EVI2(I)/EVR2(I)
      RTMAX2=AMAX1(RT(1),RT(2),RT(3),RT(4))
      CREX(IS)=(REMAX2-REMAX1)/REMAX1*100.
      CREM(IS)=(REMIN2-REMIN1)/REMIN1*100.
 100  FORMAT(2X,'PERCENTAGE CHANGES IN'//2X,'REMAX=',F10.4,//2X,'
     1REMIN=',F10.4,//2X,'RTOMAX=',F10.4)
123   FORMAT(/10X,'PERCENTAGE CHANGE IN ',A8,/10X,'MAX  ',F10.4,'   MIN',
     1F10.4,'    MEAN ',F10.4,'     STD DVN ',F10.4)
      CREXA(IS)=REMAX2
      CREMA(IS)=REMIN2
124   FORMAT(2X,'UNPERTURBED VALUES ARE'//2X,'REMAX=',F10.4,//2X,'
     1REMIN=',F10.4,//2X,'RTOMAX=',F10.4)
125   FORMAT(/10X,'STATISTICS ON ',A8,'WITH PERTURBATIONS'//10X,'MAX
     1',F10.4,'  MIN',F10.4,'   MEAN ',F10.4,'     STD DVN ',F10.4)
       IF(RTMAX1 .EQ. 0. .AND. RTMAX2 .EQ. 0.) GO TO 8
      IF(RTMAX1 .EQ. 0) CRTX(IS)=100.
      IF(RTMAX1 .NE. 0.) GO TO 7
      GO TO 9
7       CRTX(IS)=(RTMAX2-RTMAX1)/RTMAX1*100.
      CRTXA(IS)=RTMAX2
      GO TO 9
8     CRTX(IS)=0.
9     IF((IS-1)/IPRINT*IPRINT .NE. IS-1) GO TO 6
      WRITE(6,99) REMAX1,REMIN1,RTMAX1,REMAX2,REMIN2,RTMAX2
99    FORMAT(2X,'REMAX1,REMIN1,RTMAX1,REMAX2,REMIN2,RTMAX2 =',6E14.6)
      WRITE(6,100) CREX(IS),CREM(IS),CRTX(IS)
6     IF(IS .LT. NSAMP) RETURN
      DO 4 I=1,NSAMP
      RMAXM=RMAXM+CREX(I)/NSAMP
      RMINM=RMINM+CREM(I)/NSAMP
      RMAXMA=RMAXMA+CREXA(I)/NSAMP
      RMINMA=RMINMA+CREMA(I)/NSAMP
      RTMXMA=RTMXMA+CRTXA(I)/NSAMP
4     RTMXM=RTMXM+CRTX(I)/NSAMP
      DO 5 I=1,NSAMP
      SRX=SRX+(CREX(I)-RMAXM)**2/(NSAMP-1)
      SRM=SRM+(CREM(I)-RMINM)**2/(NSAMP-1)
      SRXA=SRXA+(CREXA(I)-RMAXMA)**2/(NSAMP-1)
      SRMA=SRMA+(CREMA(I)-RMINMA)**2/(NSAMP-1)
      SRTXA=SRTXA+(CRTXA(I)-RTMXMA)**2/(NSAMP-1)
5     SRTX=SRTX+(CRTX(I)-RTMXM)**2/(NSAMP-1)
      SRX=SQRT(SRX)
      SRM=SQRT(SRM)
      SRTX=SQRT(SRTX)
      SRXA=SQRT(SRXA)
      SRMA=SQRT(SRMA)
      SRTXA=SQRT(SRTXA)
      CALL MINMAX(CREX,NSAMP,CREXN,CREXX)
      CALL MINMAX(CREM,NSAMP,CREMN,CREMX)
      CALL MINMAX(CRTX,NSAMP,CRTXN,CRTXX)
      CALL MINMAX(CREXA,NSAMP,CREXNA,CREXXA)
       CALL MINMAX(CREMA,NSAMP,CREMNA,CREMXA)
```

```
      CALL MINMAX(CRTXA,NSAMP,CRTXNA,CRTXXA)
      WRITE(6,120)
120   FORMAT(//25X,'STATISTICS FROM THE MONTE CARLO SIMULATIONS')
      WRITE(6,123) ANAME1,CREXX,CREXN,RMAXM,SRX
      CALL HISTO(CREX,NSAMP,CREXN,CREXX,D,20,NY,NT)
      WRITE(6,121) ANAME1,D
121   FORMAT(10X,A8,' HISTOGRAM ,EACH INTERVAL IS ',F10.4,' UNITS'/)
      WRITE(6,122) (I,I=1,20),(NY(J),J=1,20),NT
122   FORMAT(5X,20I6,//4X,20I6,/40X,'TOTAL NUMBER OF SAMPLES=',I6)
      WRITE(6,123) ANAME2,CREMX,CREMN,RMINM,SRM
      CALL HISTO(CREM,NSAMP,CREMN,CREMX,D,20,NY,NT)
      WRITE(6,121) ANAME2,D
      WRITE(6,122) (I,I=1,20),(NY(J),J=1,20),NT
      WRITE(6,123) ANAME3,CRTXX,CRTXN,RTMXM,SRTX
      CALL HISTO(CRTX,NSAMP,CRTXN,CRTXX,D,20,NY,NT)
      WRITE(6,121) ANAME3,D
      WRITE(6,122) (I,I=1,20) ,(NY(J),J=1,20),NT
      RMAXM=0.
      WRITE(6,124) CREXA(IS),CREMA(IS),CRTXA(IS)
      WRITE(6,125) ANAME1,CREXXA,CREXNA,RMAXMA,SRXA
      CALL HISTO(CREXA,NSAMP,CREXNA,CREXXA,D,20,NY,NT)
      WRITE(6,121) ANAME1, D
         WRITE(6,122) (I,I=1,20),(NY(J),J=1,20),NT
      WRITE(6,125) ANAME2,CREMXA,CREMNA,RMINMA,SRMA
      CALL HISTO(CREMA,NSAMP,CREMNA,CREMXA,D,20,NY,NT)
      WRITE(6,121) ANAME2,D
      WRITE(6,122) (I,I=1,20),(NY(J),J=1,20),NT
      WRITE(6,125) ANAME3,CRTXXA,CRTXNA,RTMXMA,SRTXA
      CALL HISTO(CRTXA,NSAMP,CRTXNA,CRTXXA,D,20,NY,NT)
      WRITE(6,121) ANAME3,D
      WRITE(6,122) (I,I=1,20),(NY(J),J=1,20),NT
      RMAXMA=0.
      RMINMA=0.
       RTMXMA=0.
      SRXA=0.
      SRMA=0.
       SRTXA=0.
      RMAXM=0.
      RMINM=0.
      RTMXM=0.
      SRX=0.
      SRM=0.
      SRTX=0.
      RETURN
      END
C
C     XYSTO - STORES ARRAYS X AND Y IN ARRAYS XX AND YY
C
      SUBROUTINE XYSTO(XX,YY,X,Y,N,IP)
      DOUBLE PRECISION DSEED
      DIMENSION XX(1),YY(1),X(N),Y(N),RAND(4)
      DATA SEED /13.00/
      CALL GGUBS(SEED,N,RAND)
      DO 1 I=1,N
      RAND(I)=-1+2*RAND(I)
```

```
      XX(IP)=X(I)
      YY(IP)=Y(I)
      IF(Y(I) .EQ. 0.) YY(IP)=YY(IP)+RAND(I)*.04
1     IP=IP+1
      RETURN
      END
C
C     HISTO - COMPUTES HISTOGRAM
C
      SUBROUTINE HISTO(X,N,XMIN,XMAX,D,NH,NY,NT)
      DIMENSION X(1),NY(NH),XH(20)
      D=(XMAX-XMIN)/NH
      XDUM=XMIN
      DO 1 I=1,NH
      NY(I)=0
      XH(I)=D+XDUM
1     XDUM=XH(I)
      XH(NH)=XMAX
      DO 2 I=1,N
      DO 3 J=1,NH
      IF(X(I) .LE. XH(J)) GO TO 2
3     CONTINUE
C5     IF(J .EQ. 1 .OR. J .EQ. 20) WRITE(6,101) X(I)
2     NY(J)=NY(J)+1
101   FORMAT(15X,5F10.4)
      NT=0
      DO 4 I=1,NH
4     NT=NT+NY(I)
      RETURN
      END
C   FILE GENE FORTRAN           4/23/80
C
C   PROGRAM TO FIND EIGENSOLUTIONS OF A GENERAL MATRIX
C
      SUBROUTINE GENE(A,N,EVR,EVI,EVEC,IS,IPRT)
      DIMENSION A(4,4),EVR(4),EVI(4),EVEC(4,4),WORK(25)
      COMMON/NNOUT/NOUT
      NOUT = 6
C     CALL WRTMAT(A,4,4,4,'A       ')
      N = 4
      IA = 4
      MODE = 1
      IE = 4
      IPRINT = 2
      CALL EIGER(A,N,IA,MODE,EVR,EVI,EVEC,IE,WORK,IPRINT,'A+B*K    ',
     1IS,IPRT)
      RETURN
      END
C
      SUBROUTINE ESHIFT(CEVAL,CEVEC,N,NC,EVALR,EVALI,EVEC,WORK,NE)
      COMPLEX CEVAL(1),CEVEC(NC,1),CNORM
      DIMENSION EVALR(1),EVALI(1),EVEC(NE,1),WORK(NE,1)
      DIMENSION INDX(16),EVALA(16)
      DO 10 I=1,N
      EVALR(I) = REAL(CEVAL(I))
```

```
      EVALI(I) = AIMAG(CEVAL(I))
      EVALA(I) = CABS(CEVAL(I))
      IF(EVALI(I))26,15,15
15    EMAX = 0.
      DO 20 J=1,N
      EMAG = CABS(CEVEC(J,I))
      IF(EMAG.LE.EMAX)GO TO 20
      EMAX = EMAG
      CNORM = CONJG(CEVEC(J,I))/EMAG**2
20    CONTINUE
      DO 22 J=1,N
22    WORK(J,I) = REAL(CEVEC(J,I)*CNORM)
      GO TO 10
26    DO 28 J=1,N
28    WORK(J,I) = AIMAG(CEVEC(J,I-1)*CNORM)
10    CONTINUE
      CALL ORDER(EVALA,INDX,N,2,EVALR,EVALI)
      DO 30 J=1,N
      JJ = INDX(J)
      DO 30 I=1,N
30    EVEC(I,J) = WORK(I,JJ)
      I = 0
40    I = I + 1
      IF(I.GT.N)RETURN
      IF(EVALI(I))45,40,45
45    EVALI(I) = ABS(EVALI(I))
      I = I + 1
      EVALI(I) = -EVALI(I-1)
      GO TO 40
      END
C
C      ORDER - ORDERS THE ELEMENTS OF A VECTOR BY ALGEBRAIC SIZE.  THE
C      RESULTING PERMUTATION OF THE INDICES IS RETURNED IN VECTOR IV.
C      IF MODE = 1, THE ELEMENTS OF VECTOR V1 ARE ALSO REORDERED
C      IF MODE = 2, BOTH VECTORS V1 AND V2 ARE REORDERED
C
      SUBROUTINE ORDER(RV,IV,N,MODE,V1,V2)
      DIMENSION RV(1),IV(1),V1(1),V2(1)
      IF(N.LE.1)RETURN
      DO 1 I=1,N
1     IV(I) = I
      DO 4 II = 2,N
      I = II
2     J = I - 1
      IF(RV(I).GE.RV(J))GO TO 4
      CALL SWAP(RV(I),RV(J))
      IF(MODE.GE.1)CALL SWAP(V1(I),V1(J))
      IF(MODE.GE.2)CALL SWAP(V2(I),V2(J))
      KK = IV(I)
      IV(I) = IV(J)
      IV(J) = KK
      IF(J.LE.1)GO TO 4
      I = I - 1
      GO TO 2
4     CONTINUE
```

```
      RETURN
      END
C
C       EIGER - CALCULATES THE EIGENVALUES OF A MATRIX (MODE = 0)
C               OR BOTH EIGENVALUES AND EIGENVECTORS (MODE = 1)
C              THE EIGENVALUES ARE ORDERED BY ABS VALUE
C       IPRINT CONTROLS THE PRINTOUT
C           IPRINT = 1, LIST EIGENVALUES ONLY
C           IPRINT = 2, ALSO LIST EIGENVECTORS
C
      SUBROUTINE EIGER(A,N,IA,MODE,EVR,EVI,EVEC,IE,WORK,IPRINT,ANAME,
     1IS,IPRT)
      DIMENSION A(IA,1),EVEC(IE,1),EVR(1),EVI(1),WORK(1)
      REAL*8 ANAME
      COMPLEX CEVAL(16),CEVEC(16,16)
      COMMON/NNOUT/NPRINT
      IF(N.GT.16)STOP1
      IJOB = MODE
      IC = 16
      CALL EIGRF(A,N,IA,IJOB,CEVAL,CEVEC,IC,WORK,IER)
      IF(IER.GT.0)WRITE(NPRINT,102)IER
102   FORMAT(/,' ERROR IN EIGRF    IER =',I5)
      IF(IPRINT.GE.3)CALL CEVPRT(CEVAL,CEVEC,N,IC,WORK,MODE)
      CALL ESHIFT(CEVAL,CEVEC,N,IC,EVR,EVI,EVEC,WORK,IE)
      IF(IPRINT.LE.0)RETURN
      IF(IPRINT.EQ.1)GO TO 40
      IF((IS-1)/IPRT*IPRT .NE. IS-1) RETURN
      IF(MODE.LE.0)GO TO 40
      WRITE(NPRINT,104)N,ANAME
104   FORMAT(/' THE',I3,' ORDER ',A8,' MATRIX HAS',
     1   /,3X,'EVAL(REAL)',4X,'EVAL(IMAG)',4X,'E-VECTOR')
      DO 30 I=1,N
30    WRITE(NPRINT,103)EVR(I),EVI(I),(EVEC(J,I),J=1,N)
103   FORMAT(1X,1P2E14.6,0P10F10.6,/,(29X,10F10.6))
      RETURN
40    WRITE(NPRINT,105)N,ANAME,(EVR(I),I=1,N)
105   FORMAT(/' THE',I3,' ORDER ',A8,' MATRIX HAS EIGENVALUES'
     1      ,' (REAL)/(IMAG)',/(1P10E13.5))
      WRITE(NPRINT,106)(EVI(I),I=1,N)
106   FORMAT(/(1P10E13.5))
      RETURN
      END
C
      SUBROUTINE CEVPRT(CEVAL,CEVEC,N,IC,WORK,MODE)
      COMPLEX CEVAL(1),CEVEC(IC,1)
      DIMENSION WORK(N,1)
      DO 10 I=1,N
      WORK(1,I) = REAL(CEVAL(I))
10    WORK(2,I) = AIMAG(CEVAL(I))
      CALL WRTMAT(WORK,2,N,N,'E-VALS  ')
      DO 20 I=1,N
      DO 20 J=1,N
20    WORK(I,J) = REAL(CEVEC(I,J))
      CALL WRTMAT(WORK,N,N,N,'RL E-VEC')
      DO 30 I=1,N
```

```
      DO 30 J=1,N
30    WORK(I,J) = AIMAG(CEVEC(I,J))
      CALL WRTMAT(WORK,N,N,N,'IM E-VEC')
      RETURN
      END
C
C     WRTMAT - WRITES MATRIX A
C
      SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
      DIMENSION A(IA,1)
      COMMON/NNOUT/NPRINT
      REAL*8 ANAME
      WRITE(NPRINT,100)ANAME,N,M
100   FORMAT(/,' MATRIX ',A8,3X,'(',I3,' ROWS X',I3,' COLS)')
      IF(M.LE.10)GO TO 15
      DO 10 I=1,N
10    WRITE(NPRINT,101)(A(I,J),J=1,M)
101   FORMAT(/,(1P10E13.5))
      RETURN
15    DO 20 I=1,N
20    WRITE(NPRINT,102)(A(I,J),J=1,M)
102   FORMAT(1P10E13.5)
      RETURN
      END
C
C        MSHIFT - TRANSFERES VECTORS AND MATRICES
C
        SUBROUTINE MSHIFT(A,B,N,M,NA,NB)
        DIMENSION A(NA,1),B(NB,1)
        COMMON/NNOUT/NOUT
        IF(N.GT.MIN0(NA,NB))GO TO 90
        DO 10 I=1,N
        DO 10 J=1,M
10      B(I,J) = A(I,J)
        RETURN
90      WRITE(NOUT,100)N,NA,NB
100     FORMAT(' *** ERROR IN MSHIFT *** N,NA,NB=',3I5)
        RETURN
        END
C
C        SWAP - INTERCHANGES TWO VARABLES
C
        SUBROUTINE SWAP(A,B)
        C = A
        A = B
        B = C
        RETURN
        END
C
C**********   SUBROUTINE APLOT   **********
C
      SUBROUTINE APLOT(X,Y,NPT,NPLOT,XYLIMS,WIDTH,HEIGHT,TICKL,
     1     NCASE,XLBL,NXC,YLBL,NYC,TITL,NTTL,CNAMES,NCINDX)
C
C  NOTE   THE X( ), Y( ) ARRAYS ARE UNCHANGED IN THIS SUBROUTINE
```

```
C
C   APLOT INPUTS:
C
C   X( )=ARRAY OF X-COORDINATES OF POINTS
C   Y( )=ARRAY OF Y COORDINATES OF POINTS
C   NPT( )=NO. OF POINTS(X,Y PAIRS)IN EACH CURVE
C   NPLOT=NUMBER OF CURVES
C   XYLIMS(4)=LIMITS OF X AND Y AXES IN THE ORDER XMIN,XMAX,YMIN,YMAX
C   IF XYLIMS( )=0,SCALING IS AUTOMATIC
C   WIDTH=WIDTH OF PLOT IN INCHES
C   HEIGHT=HEIGHT OF PLOT IN INCHES
C   TICKLE=LENGTH OF TICK MARKS ON AXES IN INCHES AND HEIGHT OF SCALE NOS
C       IN INCHES.  IF TICKL IS NEGETIVE,PUT THE TICK MARKS INSIDE AXES
C   NCASE=A NO. PRINTED AT THE UPPER RIGHT HAND CORNER OF PLOT FOR
C   INDEXING PURPOSES.
C   XLBL( ) = ALPHANUMERIC STRING FOR X-AXIS LABEL
C   NXC = NUMBER OF CHARACTERS IN X AXIS LABEL
C   YLBL( ), NYC = LIKEWISE FOR Y AXIS LABEL
C   TITL( ) = ALPHANUMERIC STRING FOR TITLE
C   NTTL = NUMBER OF CHARACTERS IN TITLE
C   CNAMES( ) = ARRAY OF FOUR-CHARACTER IDENTIFIERS FOR THE NPLOT CURVES
C   NCINDX( ) = INTEGER (.GE.1. AND .LE.NPT( )) INDICATING LOCATION
C               OF CNAMES IDENTIFIERS FOR EACH CURVE
C
C
      DIMENSION X(1),Y(1),XAR(10),NPT(1),XYLIMS(1)
      DIMENSION XLBL(1),YLBL(1),TITL(1),CNAMES(1),NCINDX(1)
      COMMON/NNOUT/NODEV
      DATA EPS/1.E-8/
      WRITE(NODEV,100)NPLOT,NCASE,(NPT(I),I=1,NPLOT)
100   FORMAT(/' APLOT CALLED   NPLOT =',I5,3X,'NCASE =',I3,3X,'NPT( ) ='
     1    ,10I4,/,(10X,10I4))
C       DO 110 I=1,10
C       NNPT = NPT(I)
C110    WRITE(NODEV,109) I,X(I),Y(I+NNPT),Y(I+202),Y(I+303),Y(I+404)
C109    FORMAT(2X,I5,'COORDINATES=', 6F10.4)
      IF(NPLOT.LE.0)RETURN
      XMN = XYLIMS(1)
      XMX = XYLIMS(2)
      YMN = XYLIMS(3)
      YMX = XYLIMS(4)
      TICK = ABS(TICKL)
      H1 = 12*TICK
      H2 = H1 + WIDTH
      V1 = 9*TICK
      V2 = V1 + HEIGHT
C
C   FIND TOTAL NUMBER OF POINTS ON PLOT
C
      NPOINT = 0
      DO 2 I=1,NPLOT
2     NPOINT = NPOINT + NPT(I)
      IF(XMN.GE.XMX)GO TO 4
      XMIN = XMN
      XMAX = XMX
```

```
      GO TO 6
4     CALL MINMAX(X,NPOINT,XMIN,XMAX)
6     IF(YMN.GE.YMX)GO TO 8
      YMIN = YMN
      YMAX = YMX
      GO TO 10
8     CALL MINMAX(Y,NPOINT,YMIN,YMAX)
C
C    SCALE X-AXES
C
10    CALL SCALE(XMIN,XMAX,XST,XDEL,NXP)
      XFIN = XST + XDEL*NXP
      DX = XFIN - XST
      XF = (H2-H1)/DX
C
C    SCALE Y-AXES
C
      CALL SCALE(YMIN,YMAX,YST,YDEL,NYP)
      YFIN = YST + YDEL*NYP
      DY = YFIN - YST
      YF = (V2-V1)/DY
      WRITE(NODEV,101)XMIN,XMAX,XST,XDEL,XFIN,NXP
      WRITE(NODEV,102)YMIN,YMAX,YST,YDEL,YFIN,NYP
101   FORMAT(/' XMIN,XMAX =',1P2E14.7,5X,'XSTART,XDEL,XFINISH =',
     1  3E10.3,5X,'NXP =',I4)
102   FORMAT(/' YMIN,YMAX =',1P2E14.7,5X,'YSTART,YDEL,YFINISH =',
     1  3E10.3,5X,'NYP =',I4)
C
C    DRAW OUTER BORDER FOR PLOTS
C
      CALL PLOT(H1,V1,3)
      CALL PLOT(H2,V1,2)
      CALL PLOT(H2,V2,2)
      CALL PLOT(H1,V2,2)
      CALL PLOT(H1,V1,2)
C
C    IF APPROPRIATE ADD X = 0, Y = 0 AXES
C
77    IF(XST.GT.0..OR.XFIN.LT.0.)GO TO 12
      XX = -XST*XF + H1
      CALL PLOT(XX,V1,3)
      CALL PLOT(XX,V2,2)
12    IF(YST.GT.0..OR.YFIN.LT.0.)GO TO 14
      YY = -YST*YF + V1
      CALL PLOT(H1,YY,3)
      CALL PLOT(H2,YY,2)
14    CONTINUE
C
C    ADD TITLE, X-AXIS LABEL AND Y-AXIS LABEL
C
      HT = TICK
      ANG = 0.
      ANG90 = 90.
      WD = HT
      XP = (H1+H2)/2. - .5*NTTL*1.5*WD
```

```
      IF(NTTL.GT.0)CALL SYMBOL(XP,V2+4.*HT,1.5*HT,TITL,ANG,NTTL)
      XP = (H1+H2)/2. - .5*NXC*WD
      IF(NXC.GT.0)CALL SYMBOL(XP,V1-8.*HT,HT,XLBL,ANG,NXC)
      YP = (V1+V2)/2. - .5*NYC*WD
      IF(NYC.GT.0)CALL SYMBOL(H1-11.*HT,YP,HT,YLBL,ANG90,NYC)
C
C     ADD CASE NUMBER (IF NONZERO) AND DATE
C
      NDG = 0
      IF(NCASE.GT.0)CALL NUMBER(H2+HT,V2,HT,FLOAT(NCASE),ANG,NDG)
      CALL DATIMP(H1,V2+2*HT,HT,'APLOT   ')
C
C     ADD X-AXIS TICK MARKS AND SCALE
C
      N = NXP + 1
      YY = V1 - 4.*TICK
      DO 22 I=1,N
      XA = (I-1)*XDEL*XF
      CALL PLOT(H1+XA,V1,3)
      CALL PLOT(H1+XA,V1-TICKL,2)
      XX = H1 + XA - 2.*TICK
      XR = XST + (I-1)*XDEL
      XR = XR + SIGN(EPS,XR)
C     IF(ABS(XR).LE.2.*EPS)XR = 0.
      NDGT = MINO(MAXO(0,-IFIX(ALOG10(ABS(XR)))+3),8)
      IF(ABS(XR) .LE. 2.*EPS) NDGT=0
      CALL NUMBER(XX,YY,HT,XR,ANG,NDGT)
22    CONTINUE
C
C     ADD Y-AXIS TICK MARKS AND SCALE
C
      N = NYP + 1
      XX = V1 - 7.*TICK
      DO 24 I=1,N
      YA = (I-1)*YDEL*YF
      CALL PLOT(H1,V1+YA,3)
      CALL PLOT(H1-TICKL,V1+YA,2)
      YY = V1 + YA
      YR = YST + (I-1)*YDEL
      YR = YR + SIGN(EPS,YR)
C     IF(ABS(YR).LE.2.*EPS)YR = 0.
      NDGT = MINO(MAXO(0,-IFIX(ALOG10(ABS(YR)))+3),8)
      IF(ABS(YR) .LE. 2.*EPS) NDGT=0
      CALL NUMBER(XX,YY,HT,YR,ANG,NDGT)
24    CONTINUE
C
C     ADD LABEL FOR EACH CURVE
C
      K = 0
      DO 30 I=1,NPLOT
      IF(NCINDX(I).LE.0)GO TO 28
      JPT = MAXO(MINO(NCINDX(I),NPT(I)),1)
      XX = AMIN1(AMAX1(H1,(X(K+JPT)-XST)*XF+H1),H2)
      YY = AMIN1(AMAX1(V1,(Y(K+JPT)-YST)*YF+V1),V2)
      CALL PLOT(XX,YY,3)
```

```
      CALL PLOT(XX+2.*HT,YY+4.*HT,2)
      CALL SYMBOL(XX+3.*HT,YY+4.*HT,HT,CNAMES(I),ANG,4)
28    N = NPT(I) - 1
C
C   PLOT EACH CURVE
C
      K = K + 1
      XX = AMIN1(AMAX1(H1,(X(K)-XST)*XF+H1),H2)
      YY = AMIN1(AMAX1(V1,(Y(K)-YST)*YF+V1),V2)
      CALL PLOT(XX,YY,3)
C     CALL SYMBOL(XX,YY,.04,1H.,0.,1)
      CALL PLOT(XX,YY,2)
      CALL CIRCLE(XX,YY,.02)
      DO 30 J=1,N
      K = K + 1
      XX = AMIN1(AMAX1(H1,(X(K)-XST)*XF+H1),H2)
      YY = AMIN1(AMAX1(V1,(Y(K)-YST)*YF+V1),V2)
      CALL PLOT(XX,YY,3)
      CALL PLOT(XX,YY,2)
C     CALL SYMBOL(XX,YY,.04,1H.,0.,1)
      IF(J .LE. 3) CALL CIRCLE(XX,YY,.02)
30    CONTINUE
      RETURN
      END
C
      SUBROUTINE SCALE(XMIN,XMAX,START1,DEL1,NPTS1)
      INTX(X) = IFIX(X-.5+SIGN(.5,X))
      IF(XMAX.LE.XMIN)GO TO 90
      XSIZE = XMAX - XMIN
      XEXP = ALOG10(XSIZE)
      IEXP = INTX(XEXP)
      XORD = 10.**IEXP
      XNORM = XSIZE/XORD
      IF(XNORM.LE.1.6)XMOD = .2
      IF(XNORM.GT.1.6.AND.XNORM.LE.4.)XMOD = .5
      IF(XNORM.GT.4..AND.XNORM.LE.8.)XMOD = 1.
      IF(XNORM.GT.8.)XMOD = 2.
      DEL1 = XORD*XMOD
      DO 10 I=1,30
      XMAG = 10.**(I-15)
      XPOINT = FLOAT(INTX(XMAG*XMAX))/XMAG
      IF(XPOINT.GE.XMIN)GO TO 20
10    CONTINUE
      GO TO 90
20    ISHIFT = (XPOINT-XMIN)/DEL1
      START1 = XPOINT - DEL1*ISHIFT
      IF(START1.GT.XMIN)START1 = START1 - DEL1
      NPTS1 = (XMAX-START1)/DEL1
      IF(START1+DEL1*(FLOAT(NPTS1)+.01).LT.XMAX)NPTS1 = NPTS1 + 1
      RETURN
90    WRITE(6,100)XMIN,XMAX
100   FORMAT(//' ERROR IN SCALE    XMIN,XMAX =',2E12.4)
      RETURN
      END
C
```

```
      SUBROUTINE MINMAX(X,N,XMIN,XMAX)
      DIMENSION X(1)
      XMIN = X(1)
      XMAX = X(1)
      IF(N.LE.1)RETURN
      DO 10 I=2,N
      XMAX = AMAX1(XMAX,X(I))
10    XMIN = AMIN1(XMIN,X(I))
      RETURN
      END
C
C**********   SUBROUTINE DATIME   **********
C
      SUBROUTINE DATIME(IMO,IDAY,IYR,IHOURS,IMIN,AMPM,PNAME,NODEV)
      REAL*8 PNAME
      DATA AM/'  AM'/,PM/'  PM'/
      CALL DATE(IMO,IDAY,IYR)
      CALL STIME(ITIME)
      XHOURS = FLOAT(ITIME)/10000.
      AMPM = AM
      IF(XHOURS.GE.12.)AMPM = PM
      IF(XHOURS.GE.13.)XHOURS = XHOURS - 12.
      IHOURS = XHOURS
5     XMIN = (XHOURS - IHOURS)*60.
      IMIN = XMIN
      IF(NODEV.GT.0)WRITE(NODEV,100)PNAME,IMO,IDAY,IYR,IHOURS,IMIN,AMPM
100   FORMAT(/' TIME IN ',A8,' IS ',A2,'/',A2,'/',A2,5X,I2,':',
     1   I2,3X,A4)
      RETURN
      END
C
C**********   SUBROUTINE DATIMP   **********
C
      SUBROUTINE DATIMP(XP,YP,HT,PNAME)
      REAL*8 PNAME
      COMMON/NNOUT/NODEV
      ANG = 0.
      WD = .9*HT
      CALL DATIME(IMO,IDAY,IYR,IHOURS,IMIN,AMPM,PNAME,NODEV)
      XHOURS = IHOURS
      XMIN = IMIN
      CALL SYMBOL(XP,YP,HT,IMO,ANG,2)
      CALL SYMBOL(XP+2.*WD,YP,HT,1H/,ANG,1)
      CALL SYMBOL(XP+3.*WD,YP,HT,IDAY,ANG,2)
      CALL SYMBOL(XP+5.*WD,YP,HT,1H/,ANG,1)
      CALL SYMBOL(XP+6.*WD,YP,HT,IYR,ANG,2)
      CALL NUMBER(XP+12.*WD,YP,HT,XHOURS,ANG,-1)
      CALL SYMBOL(XP+14.*WD,YP,HT,1H:,ANG,1)
      CALL NUMBER(XP+15.*WD,YP,HT,XMIN,ANG,-1)
      CALL SYMBOL(XP+18.*WD,YP,HT,AMPM,ANG,4)
      RETURN
      END
C
C     FILE PERT INPUT
C     IT CONTAINS THE INPUT FOR THE MONTGOMERY AIRCRAFT
```

```
C
 &PERT
 N=4,
      M=2,
      ICONT=1,
      IPRINT=1000,
      IPLOT=2,
      XLBL=4HREAL,4H(LAM,4HDA) ,
      YLBL=4HIMAG,4HINAR,4HY(LA,4HMDA),
           NXC=12, NYC=16,
     TITL=4HMONT,4HGOME,4HRY A,4H/C C,4HLOSE,4HD LO,4HOP E,4HIGEN,44HE PE,
      4HRTUR,4HBATI,4HONS,,4HP=.1,4H (MI,4HN K),4HTAU=,4H 1.0,
      NTTL=72,CNAMES=4H     ,NCINDX=0,
      NSAMP=5,
      P=.10,
      A=-3.67984E-1,1.E0,-2.4209E-2,2.58819E-1,3*0.,1.7835E-2,
    -3.2279E-2,2.67949E-1,-1.10395E-1,-9.65926E-1,2.61875E1,0.,4.461072E-2
      B=-7.67183,0.,1.96959,0.,2.06549,0.,-2.33843E0,0.,
 XK = 1.4941162E-01,-4.1928029E-01,1.0535228E-01,2.0296771E-02,1.6E+00,
      3.6583920E+00,-2.3151433E-01,-4.7954880E-02,
 &END
 &PERT
      TITL(18)=4H0.5 , ICONT=1,
 XK = 1.7022794E-01,-6.0164034E-01,2.6196051E-01,-2.7018290E-02,2.9E+00,
      4.0587797E+00,-1.1856604E+00,4.8925018E-01,
 &END
 &PERT
      TITL(18)=4H0.25,
 XK = 1.2836323E+00,-3.1967741E-01,8.5040188E-01,-1.5637993E+01,1.9E+00,
      4.4778833E+00,3.2358761E+00,-1.5113544E+00,
 &END
 &PERT
      TITL(15)=4H1(RO,TITL(16)=4HBUST,TITL(17)=4H),TA,TITL(18)=4HU=     ,
      TITL(19)=4H 1.0,
      NTTL=76,
      XK=.65132,-.385575,.082589,-.192548,1.60285,3.46768,-.7396,-. 60737,
 &END
 &PERT
      TITL(19)=4H0.5 ,
      XK=.64359,-.38107,.6764,-.244,2.07949,3.5064,-.741,-.432244,
 &END
 &PERT
      TITL(19)=4H0.25,
      XK=1.0368,-.3599,1.27027,-.322,2.05683,3.81589,-.743469,-.52096,
 &END
 &PERT
      TITL(15)=4H1(LQ,
      TITL(16)=4HR)RH,
      TITL(17)=4HO=   ,
      TITL(18)=4H100 ,
      NTTL=72,
      XK=.3306,-.1784,.10918,-.03913,-.69998,.48287,2.6222,-1.4068,
 &END
 &PERT
      TITL(18)=4H1    ,
```

```
     XK=1.2589,-.17087,1.0046,.0953,-.30686,1.0189,5.9393,-1.734,
 &END
 &PERT
     ICONT=0,
     TITL(18)=4H0.04,
     XK=5.17,-.059375,4.8719,1.1309,.106,5.0654,7.2623,-2.5309,
 &END
C
C      FILE PERT2 INPUT
C      IT CONTAINS THE INPUT FOR THE HALL AIRCRAFT
C
 &PERT
     N=4,M=3,ICONT=1,IPRINT=1000,IPLOT=2,
     XLBL=4HREAL,4H(LAM,4HDA) ,
     YLBL=4HIMAG,4HINAR,4HY(LA,4HMDA),
     NXC=12,NYC=16,
     TITL=4HHALL,4H AIR,4HCRAF,4HT CL,4HOSED,4H LOO,4HP EI,4HGENV,4HALUE,
     4H PER,4HTURB,4HATIO,4HNS P,4H=.1 ,4H(MIN,4H K)T,4HAU= ,4HL.0 ,
     NTTL=72,CNAMES=4H     ,NCINDX=0,
     NSAMP=5,
     P=.1,
     A=-3.18,1.,-.06,.022,3*0.,.0644,.63,0.,-.27,-.998,-10.6,0.,4.5,
     B=-14.4,3*0.,1.5,0.,-2.59,.037,2*0.,-.96,0.,
 XK =-3.9732631E-02,-4.3598451E-03,1.9987654E-02,1.2020696E-02,-9.9E-01,
     2.5593357E+00,2.1611822E-01,-3.6611261E-03,4.7516279E+00,-9.4E-01,
    -5.4777920E-01,-5.1410699E-01,
 &END
 &PERT
     TITL(18)=4H0.5 ,
   XK=.02469,-.001394,-.01166,.24055,.0384,-.0438,-.30165,.02299,4.14655,
     .0114765,-.3113,
 &END
 &PERT
     TITL(18)=4H0.25,
 XK = 3.5695368E-01,-3.9497081E-02,-1.2787379E-02,1.4500313E+00,2.9E-01,
    -8.7618792E-01,-1.2922382E+00,2.6900148E+00,3.6100111E+00,-4.1E+00,
    -1.2453620E+01,1.4150048E+01,
 &END
 &PERT
     TITL(15)=4H(ROB,TITL(16)=4HUST),TITL(17)=4HTAU=,TITL(18)=4HL.0 ,
   XK=-.03489,.1519,-2.633,.137833,-.933,2.535,.2175,.002644,5.5306,
     -.96733,-.6965,-.525889,
 &END
 &PERT
     TITL(18)=4H 0.5,
     XK=.03819,-.0016245,-.012548,.4133,.0386788,-.04435,-.27592,
     .0253255,4.15544,-.12295,.007238,-.3055,
 &END
 &PERT
     TITL(18)=4H0.25,
     XK=.49974,.309131,-3.344,1.46229,-.932129,-5.79063,.112387,-.0013726,
     5.97918,-1.04167,-2.05667,-1.36633,
 &END
 &PERT
     TITL(15)=4H(LQR,TITL(16)=4H),RH,TITL(17)=4H0=  ,TITL(18)=4H100 ,
```

```
     XK=.04513,.00574,.003798,.0916,.0222,.011679,.056979,.09242,
     .03633,-.0754,-.0208,-.008925,
&END
&PERT
     TITL(18)=4H 1.0,
     XK=.8653,-.067577,.007833,.99586,-.058595,.016246,.1175,.93989,
      .35148,-.55619,-.27992,-.1,
&END
&PERT
     ICONT=0,
     TITL(18)=4H0.04,
     XK=4.8341,-.4264,.02787,4.98,-.5015,.01237,.41809,4.748,1.7639,
     -.73614,-3.153,-.86026,
&END
```

Appendix E The Sensitivity of Eigenvalues and Eigenvectors

This appendix contains a more complete derivation of equations (16) and (17) presented in Chapter II. The derivation presented here was taken from Stewart (1973)[5], and more complete discussions of matrix perturbation theory are available in Horscholder (1964)[6] and Wilkenson (1965)[7].

As previously described, we assume matrices $\tilde{A}$ and $\tilde{A} + E$ have eigensolutions

$$\tilde{A}\, v_i = \lambda_i\, v_i \tag{E1}$$

and

$$(\tilde{A} + E)\, v_i' = \lambda_i'\, v_i' \tag{E2}$$

with $||E||, \mu_i = \lambda_i' - \lambda_i$ and $q_i = v_i' - v_i$ all small perturbations of order $0 < \varepsilon << 1$. Using the notation and assumptions of Chapter II, we consider a specific eigenvalue λ with corresponding right and left eigenvectors v and w satisfying

$$v^H v = 1, \quad w^H v = 1. \tag{E3}$$

Given a unitary matrix [vU], transform $\tilde{A}$ as

$$[vU]^H\, \tilde{A}\, [vU] = \begin{bmatrix} v^H\tilde{A}v & v^H\tilde{A}U \\ U^H\tilde{A}v & U^H\tilde{A}U \end{bmatrix}$$

and

$$v^H\tilde{A}v = \lambda v^H v = \lambda,$$

$$U^H\tilde{A}v = U^H(\lambda v) = \lambda(U^H v) = 0,$$

so

$$[vU]^H\, \tilde{A}\, [vU] = \begin{bmatrix} \lambda & v^H\tilde{A}U \\ 0 & U^H\tilde{A}U \end{bmatrix}.$$

The eigenvalues of $U^H\tilde{A}U$ will be those of $\tilde{A}$ less λ since we have reduced $\tilde{A}$ to block-triangular form by a similarity transformation. Next expand (E2) as

$$(\tilde{A} + E)\ (v + Up) = (\lambda + \mu)\ (v + Up) \tag{E4}$$

with
$$Ev + \tilde{A}Up \doteq \mu v + \lambda Up \tag{E5}$$

since
$$EUp = O(\varepsilon^2) \text{ and } \mu Up = 0\ (\varepsilon^2).$$

Premultiply (E5) by U^H to obtain an expression for p;

$$U^H Ev + U^H \tilde{A} Up \cong \lambda p$$

or
$$p \cong (\lambda I - U^H \tilde{A} U)^{-1}\ U^H Ev \tag{E6}$$

which yields equation (17) in Chapter II.

To derive equation (16), premultiply (E4) by v^H to obtain

$$(v^H \tilde{A} + v^H E)\ (v + Up) = (\lambda + \mu)$$

$$\lambda + v^H \tilde{A} Up + v^H Ev + v^H EUp = \lambda + \mu.$$

Use the previous result (E6) to obtain

$$\mu = v^H Ev + v^H AU\ (\lambda I - U^H \tilde{A} U)^{-1} U^H Ev$$

$$+ v^H EU\ (\lambda I - U^H \tilde{A} U)^{-1}\ U^H Ev$$

$$\mu = [1\ v^H \tilde{A} U (\lambda I - U^H \tilde{A} U)^{-1}] \begin{bmatrix} v^H \\ U^H \end{bmatrix} Ev \tag{E7}$$

$$+ v^H EU(\lambda I - U^H \tilde{A} U)^{-1}\ U^H Ev$$

$$\mu = w^H Ev + v^H EU\ (\lambda I - U^H \tilde{A} U)^{-1}\ U^H Ev \tag{E8}$$

$$|\mu| \leq ||w^H||\ ||E||\ + ||E||^2\ ||(\lambda I - U^H \tilde{A} U)^{-1}||. \tag{16}$$

To proceed from step (E7) to (E8) we used the representation for the left-eigenvector

$$w^H = [1\ v^H \tilde{A} U\ (\lambda I - U^H \tilde{A} U)^{-1}] \begin{bmatrix} v^H \\ U^H \end{bmatrix} \quad \text{of } \tilde{A}. \tag{E9}$$

This identity can be derived as follows. Let $[1\ \ z^H]$ be a left-eigenvector of the matrix

$$[vU]^H\ \tilde{A}[vU],$$

and solve for z^H. Once we find z^H, y^H is given by

$$y^H = [1 \; z^H] \begin{bmatrix} v^H \\ U^H \end{bmatrix}. \tag{E10}$$

so

$$[1 \; z^H] \begin{bmatrix} \lambda & v^H\tilde{A}U \\ 0 & U^H\tilde{A}U \end{bmatrix} = \lambda[1 \; z^H]$$

$$[\lambda \; (v^H + z^H U^H)(\tilde{A}U)] = [\lambda \; \lambda z^H]$$

or

$$z^H(\lambda I - U^H\tilde{A}U) = v^H\tilde{A}U$$

$$z^H = v^H\tilde{A}U(\lambda I - U^HAU)^{-1}$$

which by (E10) yields (E9) and hence (E8).

References

1. A. E. Bryson, Jr., and Y. C. Ho, Applied Optimal Control, Halstead Press, 1975, p. 148-168.

2. W. L. Zangwill, "Minimizing a function without calculating derivatives, Computer Journal, Vol. 10, 1967, p. 293-296.

3. L. A. Zadeh and C. A. Desoer, Linear System Theory, McGraw-Hill, 1963, p. 293-336.

4. W. L. Brogan, Modern Control Theory, Quantum Publishers, New York, 1974, p. 131-135 and p. 197-199.

5. G. W. Stewart, Introduction to Matrix Computations, Academic Press, 1973, p. 289-295.

6. A. S. Householder, The Theory of Matrices in Numerical Analysis, Ginn (Blaisdell), Boston, 1964.

7. J. H. Wilkinson, The Algebraic Eigenvalue Problem, 1965.

8. R. C. Montgomery and H. G. Hatch, "Application of Differential Synthesis to Design of Multiaxes Stability Augmentation Systems," Journal of Aircraft, Vol. 6, No. 4, July-Aug., 1969, p. 336-343.

9. G. W. Hall, A Flight Test Investigation of Direct Side Force Control, Tech Rept. AFFDL-TR-71-106, Sept. 1971, Wright Patterson AFB, Ohio, p. 64.

10. N. R. Sandell and M. Athans, "Modern Control Theory: Manual for Fortran Computer Subroutines for Linear Quadratic, Goussian Design," report from the Center for Advanced Engineering Study, MIT, Cambridge, Massachusetts, 1974.