NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.
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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Introduction</td>
<td>6</td>
</tr>
<tr>
<td>II. Eigenvalue Robustness through Eigenvector Orthogonality</td>
<td>12</td>
</tr>
<tr>
<td>III. Design Problems</td>
<td>15</td>
</tr>
<tr>
<td>IV. Programs NEWSOM and LINEAR</td>
<td>17</td>
</tr>
<tr>
<td>V. Results of programs NEWSOM and LINEAR</td>
<td>22</td>
</tr>
<tr>
<td>VI. Program INTODE</td>
<td>26</td>
</tr>
<tr>
<td>VII. Results of program INTODE</td>
<td>30</td>
</tr>
<tr>
<td>VIII. Program PERTB</td>
<td>42</td>
</tr>
<tr>
<td>IX. Results of program PERTB</td>
<td>46</td>
</tr>
<tr>
<td>X. Conclusions and recommendations</td>
<td>60</td>
</tr>
<tr>
<td>XI. Appendix A - NEWSOM computer program</td>
<td>62</td>
</tr>
<tr>
<td>XII. Appendix B - LINEAR computer program</td>
<td>76</td>
</tr>
<tr>
<td>XIII. Appendix C - INTODE computer program</td>
<td>104</td>
</tr>
<tr>
<td>XIV. Appendix D - PERTB computer program</td>
<td>126</td>
</tr>
<tr>
<td>XV. Appendix E - The sensitivity of eigenvalues and eigenvectors</td>
<td>143</td>
</tr>
<tr>
<td>References</td>
<td>146</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1. The domain in the complex plane specified by parameters REMIN, REMAX and RTOMAX .................................................. 8
2. Flowchart of program NEWSOM ........................................... 19
3. Flowchart of program INTODE .............................................. 28
4. Hall a/c lateral dynamics, min K controllers .................................. 31
5. Hall a/c lateral dynamics, robust controllers .................................. 32
6. Hall a/c lateral dynamics, LQR controllers .................................. 33
7. Montgomery a/c lateral dynamics, min K controllers ........................ 36
8. Montgomery a/c lateral dynamics, robust controllers ......................... 37
9. Montgomery a/c lateral dynamics, LQR controllers ........................ 38
10. Integral control effort vs state deviation ........................................ 41
11. Flowchart of program PERTB ............................................... 44
12. Hall a/c eigenvalue perturbations, min K controllers ........................ 47
13. Hall a/c eigenvalue perturbations, min K and robust controllers ............ 48
14. Hall a/c eigenvalue perturbations, robust controllers ........................ 49
15. Hall a/c eigenvalue perturbations, LQR controllers .......................... 50
16. Hall a/c eigenvalue perturbations, LQR controller ............................ 51
17. Montgomery a/c eigenvalue perturbations, min K controllers ............... 54
18. Montgomery a/c eigenvalue perturbations, min K and robust controllers ...... 55
19. Montgomery a/c eigenvalue perturbations, robust controllers ............... 56
20. Montgomery a/c eigenvalue perturbations, LQR controllers ................ 57
21. Montgomery a/c eigenvalue perturbations, LQR controller ................ 58
<table>
<thead>
<tr>
<th>List of Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input data for programs NEWSOM and LINEAR</td>
<td>23</td>
</tr>
<tr>
<td>2. Gain matrices for Montgomery aircraft</td>
<td>24</td>
</tr>
<tr>
<td>3. Gain matrices for Hall aircraft</td>
<td>25</td>
</tr>
<tr>
<td>4. Results from the integration of Hall aircraft dynamics</td>
<td>34</td>
</tr>
<tr>
<td>5. Results from the integration of Montgomery aircraft dynamics</td>
<td>39</td>
</tr>
<tr>
<td>6. Statistics on REMAX, REMIN and RTOMAX for Hall aircraft</td>
<td>52</td>
</tr>
<tr>
<td>7. Statistics on REMAX, REMAN and RTOMAX for Montgomery aircraft</td>
<td>55</td>
</tr>
</tbody>
</table>
I. Introduction

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

\[ \dot{x} = Ax + Bu \quad (1) \]
\[ y = Cx + Du \quad (2) \]

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 robust 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 \( m \times n \) feedback matrix \( K \) with

\[ u = Kx. \quad (3) \]
The three methods evaluated are:

1) the standard linear quadratic regulator (LQR) design method, 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. 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, \text{RTOMAX}. The closed-loop eigenvalues are those of the matrix
\[ \tilde{A} = A + BK \] (4)
specified as
\[ \lambda(\tilde{A}) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\} \]
Thus the eigenvalue domain can be specified mathematically as
\[ \text{REMIN} \leq \text{Re} (\lambda_i) \leq \text{REMAX} \]
\[ \left| \frac{\text{Im} (\lambda_i)}{\text{Re} (\lambda_i)} \right| \leq \text{RTOMAX} \] (5) (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\textsuperscript{3, 4}. That is, given any arbitrary control function \( u(t) \), the time response of system (1) is
\[ x(t) = e^{\tilde{A}t} x(0) + \int_0^t e^{\tilde{A}(t-\tau)} Bu(\tau) d\tau. \] (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.
\[ \tilde{A} v_i = \lambda_i v_i, \quad w_i^H \tilde{A} = \lambda_i w_i^H. \] (8)
The eigenvalue problem (8) can be written in matrix form as
\[ \tilde{A} = MJQ \] (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.
\[
\lambda_i = -a_i < 0, \quad i = 1, 2, \ldots, r
\]
and that \( 2c \) eigenvalues are complex and stable, so that there are \( c \) complex conjugate pairs
\[
\lambda_{r+j} = -\alpha_j + i\beta_j, \quad \alpha_j < 0
\]
\[
\lambda_{r+c+j} = \lambda_{r+j} = -\alpha_j - i\beta_j, \quad j = 1, 2, \ldots, c.
\]
Then the matrix exponential can be written as
\[
e^{At} = \sum_{i=1}^{r} e^{-a_i t} E_i + \sum_{j=1}^{c} e^{-\alpha_j t} (\cos(\beta_j t) F_j + \sin(\beta_j t) G_j)
\]
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 (11), the rate of decay of the response \( x(t) \) of the closed-loop system is characterized by the time constant
\[
\tau = \max \left\{ \frac{1}{a_1}, \ldots, \frac{1}{a_r}, \frac{1}{\alpha_1}, \ldots, \frac{1}{\alpha_c} \right\}. 
\]
A larger time constant \( \tau \) means slower system response, and \( \tau \) is kept from becoming large by the right-hand eigenvalue boundary \( \text{REMAX} \), i.e.
\[
\text{Re} (\lambda_i) \leq \text{REMAX} \quad \text{implies} \quad \tau \leq \frac{-1}{\text{REMAX}}. 
\]
Note that \( \text{REMAX} \) will always be negative for a stable design. The choice of \( \text{REMAX} \) is governed by (12) to achieve a desired or specified closed-loop time constant \( \tau \).

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 \( \zeta \) are related by the angle
\[
\psi = \tan^{-1}(RTOMAX) \quad \cos^{-1}(\zeta).
\]
In order to have well damped non-oscillatory motion in each mode corresponding to a complex closed-loop eigenvalue, one can choose \( \zeta \geq 0.71 \) or \( \psi \leq 45^\circ \) which implies RTOMAX \( \leq 1 \).

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
\[
M = [v_1 \ v_2 \ldots \ v_n]^T.
\]
We assume that the closed-loop system matrix \( \tilde{A} \) is perturbed as
\[
\tilde{A} + E
\]
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 \( \lambda \) 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 \( \lambda' \) and eigenvector \( v' \) of the perturbed system
\[
(\tilde{A} + E) v' = \lambda' v'
\]
that are near \( \lambda \) and \( v \). Since \( E \) is small, i.e.
\[
||E|| = O(\varepsilon), \quad 0 < \varepsilon << 1
\]
the differences \( \lambda' - \lambda \) and \( v' - v \) will also be small, i.e.
\[
\lambda' - \lambda = \mu, \quad ||\mu|| = O(\varepsilon) << 1
\]
\[
v' - v = q, \quad ||q|| = O(\varepsilon) << 1.
\]
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 \( \mu \) and \( q \), let \( U \) be any \( n \times (n-1) \) dimensional matrix such that \([vU]\) is \( n \times n \) 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

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

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

(16)

and

\[
v' - v \approx (\lambda I - U^H\tilde{A}U)^{-1}U^H E v
\]

\[
|v' - v| \leq \varepsilon\|U(\lambda I - U^H\tilde{A}U)^{-1}\|
\]  

(17)

are derived in Appendix E. As equation (16) indicates, for small perturbations \( E \) the length of the left eigenvector \( w_i \) corresponding to eigenvalue \( \lambda_i \) provides a measure of the sensitivity of \( \lambda_i \) to variations in \( \tilde{A} + BK \). Since the left eigenvectors provide a reciprocal basis for the right eigenvectors, it follows that

\[
\|w_i^H v_i\| = 1.
\]

The angle \( \theta_i \) between vectors \( w_i \) and \( v_i \) measured by

\[
\theta_i = \cos^{-1} \frac{w_i^Hv_i}{(w_i^H w_i)^{1/2} (v_i^H v_i)^{1/2}}
\]  

(18)

also provides a measure of the length of \( w_i \) and the sensitivity of \( \lambda_i \) to perturbations \( E \). Small angles \( \theta_i \) will indicate that \( w_i \) is small, and therefore the eigenvalue \( \lambda_i \) is "robust".

The second result, equation (17), indicates that the sensitivity of eigenvector \( v_i \) is proportional to the distance between eigenvalue \( \lambda_i \) 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 \( \lambda_i \), 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 $\lambda_i$, $\lambda_j$ 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

\[
\begin{bmatrix}
\text{roll rate (rad/sec)} \\
\text{roll angle (rad)} \\
\text{yaw rate (rad/sec)} \\
\text{sideslip angle (rad)}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{aileron (rad)} \\
\text{rudder (rad)} \\
\text{Yaw control* (rad)}
\end{bmatrix}
\]

*for Hall a/c only.

The first example is taken from Montgomery and Hatch (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 (1971). The yaw control is achieved through asymmetric deflection of drag petals mounted on wing tip tanks.
\[
A = \begin{bmatrix}
-3.18 & 0.0 & 0.63 & -10.6 \\
1.0 & 0.0 & 0.0 & 0.0 \\
-0.06 & 0.0 & -0.27 & 4.18 \\
0.022 & 0.0644 & -0.988 & -0.151
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
-14.4 & 1.5 & 1.0 \\
0.0 & 0.0 & 0.0 \\
0.0 & -2.59 & -0.96 \\
0.0 & 0.037 & 0.0
\end{bmatrix}
\]
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

$$u = Kx.$$  \hspace{1cm} (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 $w_i$.

$$f(x) = \sum_{i=1}^{5} w_i f_i(x)$$  \hspace{1cm} (20)

The scalar functions are defined as

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

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

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

$$f_4(x) = \sum_{i=1}^{n} \max(0, \text{abs}(\text{Im}(\lambda_i)/\text{Re}(\lambda_i)) - \text{RTOMAX})^2$$
where \( \{\lambda_i\} \) are the eigenvalues of the closed loop system \( \tilde{A} = A + BK \). The three terms \( f_2(x), f_3(x), f_4(x) \) constrain the closed-loop eigenvalues to remain in the domain \( \Gamma \) of the complex plane illustrated in Figure 1. The fifth scalar function provides a measure of eigenvector orthogonality. Note that the eigenvector angles \( \phi_{ij} \) defined here differ from the angle \( \psi_i \) introduced in Chapter II. As \( \phi_{ij} \) goes to 90° for all \( i \neq j \), \( \psi_i \) goes to zero.

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

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

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

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 \( wT1, \ldots, wT5 \). 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 \( \Gamma \). 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
start

read namelist NSOMIN from file NEWSOM INPUT

if IPRINT ≥ 3, then print out namelist NSOMIN

if x(1) = -1, then randomize the elements of the initial K matrix, K = [x(j)]

if IPRINT ≥ 1, then print out A and B matrices

compute cost function with initial K and print out cost terms, eigenvalues and angles between eigenvectors

minimize the function f(x) of the nm elements of K.

\[ f(t) = W T_1 \cdot f_1(x) + W T_2 \cdot f_2(x) + W T_3 \cdot f_3(x) + W T_4 \cdot f_4(x) + W T_5 \cdot f_5(x) \]

subroutine RANDS computes random numbers -1 ≤ x(j) ≤ 1, j=1, ...nm

function COSTF computes cost function. If IPRINT ≥ 5, print partial output; if IPRINT ≥ 8, print complete output

subroutine POWELL minimizes an unconstrained multivariable function, call subroutines:

COSTF
RANDIN
MINPT
COSTD
MINPAR

stop
compute cost function with final K, and print out all info.

if IOUT \geq 1, then output namelist NSOMIN on file NEWSOM DATA
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 $\tau$ (or equivalently $\text{REM} = \frac{-1}{\tau}$), 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 $\Gamma$ 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 $\rho = 100., 1.,$ 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 $\phi_{ij}$ 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

<table>
<thead>
<tr>
<th>Program</th>
<th>aircraft</th>
<th>controller</th>
<th>( t )</th>
<th>REMIN</th>
<th>REMAX</th>
<th>RTOMAX</th>
<th>WT1</th>
<th>WT2</th>
<th>WT3</th>
<th>WT4</th>
<th>WT5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWSOM</td>
<td>Montgomery</td>
<td>minimum K</td>
<td>1.0</td>
<td>-10</td>
<td>-1</td>
<td>0.5</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.25</td>
<td></td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>robust</td>
<td>1.0</td>
<td>-20</td>
<td>-1</td>
<td>.01</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.25</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hall</td>
<td>minimum K</td>
<td>1.0</td>
<td>-10</td>
<td>-1</td>
<td></td>
<td>1</td>
<td>500</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.25</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>robust</td>
<td>1.0</td>
<td>-20</td>
<td>-1</td>
<td>.01</td>
<td>1000</td>
<td>200</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>-2</td>
<td></td>
<td></td>
<td>500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program</th>
<th>aircraft</th>
<th>( q_{11} )</th>
<th>( q_{22} )</th>
<th>( q_{33} )</th>
<th>( q_{24} )</th>
<th>( r_{11} )</th>
<th>( r_{22} )</th>
<th>( r_{33} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINEAR</td>
<td>Montgomery</td>
<td>0.1</td>
<td>1.0</td>
<td>0.1</td>
<td>1.0</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.04</td>
<td>.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.04</td>
<td>.04</td>
<td>.04</td>
</tr>
</tbody>
</table>
**Table 2  Gain Matrices for Montgomery Aircraft**

<table>
<thead>
<tr>
<th>MIN K</th>
<th>K MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ = 1.0</td>
<td>1.4941162E-1 1.0535228E-1 1.6827383E0 -2.3151433E-1</td>
</tr>
<tr>
<td></td>
<td>-4.1928029E-1 2.0296771E-2 3.6583920E0 -4.7954880E-2</td>
</tr>
<tr>
<td>τ = 0.5</td>
<td>1.7022794E-1 2.6196051E-1 2.3499689E0 -1.1856604E0</td>
</tr>
<tr>
<td>τ = 0.25</td>
<td>1.2836323E0 8.5040188E-1 1.3259439E0 3.2358761E0</td>
</tr>
<tr>
<td></td>
<td>-3.1967741E-1 -1.5637993E1 4.4778833E0 -1.5135440E0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROBUST</th>
<th>K MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ = 1.0</td>
<td>6.5132E-1 8.2589E-2 1.60285EO -7.393E-1</td>
</tr>
<tr>
<td></td>
<td>-3.85575E-1 -1.92548E-1 3.46768EO -6.0737E-1</td>
</tr>
<tr>
<td>τ = 0.5</td>
<td>6.4359E-1 6.764E-1 2.07949EO -7.41E-1</td>
</tr>
<tr>
<td></td>
<td>-3.8107E-1 -2.44E-1 3.5064EO -4.32244E-1</td>
</tr>
<tr>
<td>τ = 0.25</td>
<td>1.0368E0 1.27027E0 2.05683EO -7.43469E-1</td>
</tr>
<tr>
<td></td>
<td>-3.599E-1 -3.22E-1 3.81589E0 -5.2096E-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LQR</th>
<th>K MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ω = 100</td>
<td>3.306E-1 1.0918E-1 -6.9998E-1 2.6222E0</td>
</tr>
<tr>
<td></td>
<td>-1.784E-1 -3.9913E-2 4.8287E-1 -1.4068E0</td>
</tr>
<tr>
<td>ω = 1</td>
<td>1.2588E0 1.0046E0 -3.06E-1 5.9393E0</td>
</tr>
<tr>
<td></td>
<td>-1.7087E-1 9.53E-2 1.0189E0 -1.734E0</td>
</tr>
<tr>
<td>ω = .04</td>
<td>5.17E0 4.8719E0 1.06E-1 7.2623E0</td>
</tr>
<tr>
<td></td>
<td>-5.9375E-2 1.1309E0 5.0654E0 -2.5309E0</td>
</tr>
</tbody>
</table>

24
### Table 3 Gain Matrices for Hall Aircraft

<table>
<thead>
<tr>
<th>MIN $K$</th>
<th>$K$ MATRIX</th>
<th>$K$ MATRIX</th>
<th>$K$ MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 1.0$</td>
<td>-3.97326E-2</td>
<td>1.20207E-2</td>
<td>2.16118E-1</td>
</tr>
<tr>
<td></td>
<td>-4.35985E-3</td>
<td>-9.44925E-1</td>
<td>-3.66113E-3</td>
</tr>
<tr>
<td></td>
<td>1.99877E-2</td>
<td>2.55934E0</td>
<td>4.75163E0</td>
</tr>
<tr>
<td>$\tau = 0.5$</td>
<td>2.46900E-2</td>
<td>2.40550E-1</td>
<td>-3.01650E-1</td>
</tr>
<tr>
<td></td>
<td>-1.39400E-3</td>
<td>3.84000E-2</td>
<td>2.29900E-2</td>
</tr>
<tr>
<td></td>
<td>-1.16600E-2</td>
<td>-4.38000E-2</td>
<td>4.14500E0</td>
</tr>
<tr>
<td>$\tau = 0.25$</td>
<td>3.56954E-1</td>
<td>1.45003E0</td>
<td>-1.29224E0</td>
</tr>
<tr>
<td></td>
<td>-3.94971E-2</td>
<td>2.40516E-1</td>
<td>2.69001E0</td>
</tr>
<tr>
<td></td>
<td>-1.27874E-2</td>
<td>-8.76188E-1</td>
<td>3.61001E0</td>
</tr>
</tbody>
</table>

### ROBUST

<table>
<thead>
<tr>
<th>K MATRIX</th>
<th>K MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 1.0$</td>
<td>-3.48900E-2</td>
</tr>
<tr>
<td></td>
<td>1.51900E-1</td>
</tr>
<tr>
<td></td>
<td>-2.63300E0</td>
</tr>
<tr>
<td>$\tau = 0.5$</td>
<td>3.81900E-2</td>
</tr>
<tr>
<td></td>
<td>-1.62450E-3</td>
</tr>
<tr>
<td></td>
<td>-1.25480E-2</td>
</tr>
<tr>
<td>$\tau = 0.25$</td>
<td>4.99740E-1</td>
</tr>
<tr>
<td></td>
<td>3.09131E-1</td>
</tr>
<tr>
<td></td>
<td>-3.34400E0</td>
</tr>
</tbody>
</table>

### LQR

<table>
<thead>
<tr>
<th>$K$ MATRIX</th>
<th>$K$ MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 100$</td>
<td>4.51300E-2</td>
</tr>
<tr>
<td></td>
<td>5.74600E-3</td>
</tr>
<tr>
<td></td>
<td>3.79800E-3</td>
</tr>
<tr>
<td>$\tau = 1$</td>
<td>8.65300E-1</td>
</tr>
<tr>
<td></td>
<td>-6.75770E-2</td>
</tr>
<tr>
<td></td>
<td>7.83300E-3</td>
</tr>
<tr>
<td>$\tau = 0.04$</td>
<td>4.83410E0</td>
</tr>
<tr>
<td></td>
<td>-4.26400E-1</td>
</tr>
<tr>
<td></td>
<td>2.78700E-2</td>
</tr>
</tbody>
</table>
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

\[ x = Ax + Bu \]

\[ u = Kx \]

where \( x \) is the \( n \times 1 \) state vector

\( u \) is the \( m \times 1 \) control vector

\( A \) is the \( n \times n \) system matrix

\( B \) is the \( n \times m \) control matrix

\( K \) is the \( m \times n \) 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

\[ k_1 = hf(x_i, y_i) \]

\[ k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \]

\[ k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \]

\[ k_4 = hf(x_i + h, y_i + k_2) \]

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

\[
\begin{align*}
\theta & - \text{ roll rate} \\
\phi & - \text{ roll angle} \\
r & - \text{ yaw rate} \\
\beta & - \text{ sideslip angle} \\
\end{align*}
\]

\[
\begin{align*}
\delta_a & - \text{ aileron deflection} \\
\delta_r & - \text{ rudder deflection} \\
\delta p & - \text{ yaw control(using asymmetric deflection of drag petals)}
\end{align*}
\]

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

Start

Read namelist

Determine
a) No. of print columns
b) No. of curves on plot
c) No. of integration/plot/print steps.

Initialize state, control and auxiliary variables.

Compute eigenvalues, eigenvectors and angles between the eigenvectors. Print.

Integrate ODEs using Runge-Kutta fourth order method.

Print trajectory table.

Is there any plotting required? yes Plot using printer and/or versatec plotter.

Is there another case? yes

No Stop

no
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 ($\omega = 100$, where $\omega$ 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 ($\omega = 1$).
Figure 5

HALL AIRCRAFT LATERAL DYNAMICS (ROBUST) TAU = 1.0

09/17/80 10:10 AM

HALL AIRCRAFT LATERAL DYNAMICS (ROBUST) TAU = 0.5

09/17/80 10:10 AM

HALL AIRCRAFT LATERAL DYNAMICS (ROBUST) TAU = 0.25

09/17/80 10:10 AM

TIME (SEC)

STATE

1.000

0.500

0.

-0.500

-1.000

1.000

2.000

3.000

4.000

5.000

3.000

4.000

5.000

6.000

TIME (SEC)
### Table 4: Results from the Integration of Hall Aircraft Dynamics

<table>
<thead>
<tr>
<th></th>
<th>Min K Controller</th>
<th>Robust Controller</th>
<th>LQR Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \tau = 1.0 )</td>
<td>( \tau = 0.5 )</td>
<td>( \tau = 0.25 )</td>
</tr>
<tr>
<td><strong>Eigenvalues</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.99519</td>
<td>-1.99818</td>
<td>-4.09501</td>
<td>-3.00771</td>
</tr>
<tr>
<td>-2.39960</td>
<td>-10.81421</td>
<td>-0.16414</td>
<td>-12.4053</td>
</tr>
<tr>
<td>-2.10588</td>
<td>-1.99975</td>
<td>-3.85276</td>
<td>-1.09811</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Min. eigenvector angles (in degrees)</strong></td>
<td>6.31</td>
<td>5.06</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>23.10</td>
<td>67.67</td>
<td>6.82</td>
</tr>
<tr>
<td></td>
<td>27.57</td>
<td>72.51</td>
<td>7.56</td>
</tr>
<tr>
<td><strong>\int x^T x dt</strong></td>
<td>0.02386</td>
<td>0.02622</td>
<td>0.1013</td>
</tr>
<tr>
<td><strong>\int u^T u dt</strong></td>
<td>0.3311</td>
<td>0.06697</td>
<td>1.024</td>
</tr>
<tr>
<td><strong>Max. Aileron (rad.)</strong></td>
<td>-0.096</td>
<td>-0.027</td>
<td>-0.313</td>
</tr>
<tr>
<td><strong>Max. Rudder (rad.)</strong></td>
<td>-0.149</td>
<td>0.006</td>
<td>-1.221</td>
</tr>
<tr>
<td><strong>Max Yaw Control</strong></td>
<td>0.630</td>
<td>0.286</td>
<td>1.627</td>
</tr>
</tbody>
</table>
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 $\tau = 1$) 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 $\tau$ (or $\rho$) decreases, the eigenvalues in general move further to the left. When we go from $\tau = 1$ (or $\rho = 100$) to $\tau = 0.5$ (or $\rho = 1$) the time response improves but when $\tau$ (or $\rho$) 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 ($\tau = 0.25$) 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 ($\tau = 0.5$).

Comparison of the results for Robust controller shows that cases 5 ($\tau = 0.5$) and 6 ($\tau = 0.25$) give nearly identical response. Case 6 requires slightly lower maximum control deflections and overall control effort, whereas 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

09/18/00 12:59 PM

Montgomery A/C Lateral Dynamics (Min K) TAU = 0.5

09/18/00 12:59 PM

Montgomery A/C Lateral Dynamics (Min K) TAU = 0.25

09/18/00 12:59 PM

36
Table 5 Results from the Integration of Montgomery Aircraft Dynamics

<table>
<thead>
<tr>
<th>Eigenvalues</th>
<th>Min K Controller</th>
<th>Robust Controller</th>
<th>LQR Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \tau = 1.0 )</td>
<td>( \tau = 1.0 )</td>
<td>( \nu = 100 )</td>
</tr>
<tr>
<td>-4.82860</td>
<td>-4.21222</td>
<td>-2.84629</td>
<td>-9.54026</td>
</tr>
<tr>
<td>-1.99521</td>
<td>-2.25198</td>
<td>-10.49551</td>
<td>-1.03527</td>
</tr>
<tr>
<td>-3.99847</td>
<td>-4.02489</td>
<td>-10.3037918</td>
<td></td>
</tr>
<tr>
<td>-0.99789</td>
<td>+10.97874</td>
<td>+12.86409</td>
<td></td>
</tr>
<tr>
<td>+11.78356</td>
<td>+10.37918</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+10.47245</td>
<td>-4.42521</td>
<td>-1.04682</td>
<td>-13.08237</td>
</tr>
<tr>
<td>-1.99513</td>
<td>-3.04453</td>
<td>-6.53534</td>
<td></td>
</tr>
<tr>
<td>-4.0032</td>
<td>-1.04682</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+11.83253</td>
<td>+10.36014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6.93968</td>
<td>-1.03512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.99785</td>
<td>-10.68740</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+11.66794</td>
<td>-2.38666</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Min. eigenvector angles (in degrees)</th>
<th>Min. eigenvector angles (in degrees)</th>
<th>Min. eigenvector angles (in degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. eigenvector angles (in degrees)</td>
<td>Min. eigenvector angles (in degrees)</td>
<td>Min. eigenvector angles (in degrees)</td>
</tr>
<tr>
<td>5.63</td>
<td>1.11</td>
<td>4.96</td>
</tr>
<tr>
<td>51.22</td>
<td>26.74</td>
<td>8.78</td>
</tr>
<tr>
<td>47.00</td>
<td>2.12</td>
<td>13.98</td>
</tr>
<tr>
<td>53.05</td>
<td>34.66</td>
<td>25.77</td>
</tr>
<tr>
<td>52.58</td>
<td>89.20</td>
<td>17.44</td>
</tr>
<tr>
<td>67.44</td>
<td>66.48</td>
<td>58.07</td>
</tr>
<tr>
<td>30.60</td>
<td>53.07</td>
<td>70.42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( |K| )</th>
<th>( |K| )</th>
<th>( |K| )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0597</td>
<td>4.0158</td>
<td>3.1197</td>
</tr>
<tr>
<td>4.9093</td>
<td>4.2932</td>
<td>6.4843</td>
</tr>
<tr>
<td>16.780</td>
<td>4.7474</td>
<td>11.666</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \int K^T x , dt )</th>
<th>( \int K^T x , dt )</th>
<th>( \int K^T x , dt )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.838</td>
<td>0.258</td>
<td>0.7189</td>
</tr>
<tr>
<td>0.6615</td>
<td>0.07921</td>
<td>0.04791</td>
</tr>
<tr>
<td>0.06562</td>
<td>0.04739</td>
<td>0.02122</td>
</tr>
<tr>
<td>0.258</td>
<td>0.07921</td>
<td>0.04791</td>
</tr>
<tr>
<td>0.06562</td>
<td>0.04739</td>
<td>0.02122</td>
</tr>
<tr>
<td>1.838</td>
<td>0.258</td>
<td>0.7189</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \int u^T u , dt )</th>
<th>( \int u^T u , dt )</th>
<th>( \int u^T u , dt )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.560</td>
<td>0.264</td>
<td>0.01608</td>
</tr>
<tr>
<td>0.5949</td>
<td>0.02565</td>
<td>0.66171</td>
</tr>
<tr>
<td>0.2217</td>
<td>0.02219</td>
<td>0.1874</td>
</tr>
<tr>
<td>0.560</td>
<td>0.264</td>
<td>0.01608</td>
</tr>
<tr>
<td>0.5949</td>
<td>0.02565</td>
<td>0.66171</td>
</tr>
<tr>
<td>0.2217</td>
<td>0.02219</td>
<td>0.1874</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max. Aileron (rad.)</th>
<th>Max. Aileron (rad.)</th>
<th>Max. Aileron (rad.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.455</td>
<td>0.434</td>
<td>0.273</td>
</tr>
<tr>
<td>0.578</td>
<td>0.468</td>
<td>0.694</td>
</tr>
<tr>
<td>0.447</td>
<td>0.474</td>
<td>1.213</td>
</tr>
<tr>
<td>0.455</td>
<td>0.434</td>
<td>0.273</td>
</tr>
<tr>
<td>0.578</td>
<td>0.468</td>
<td>0.694</td>
</tr>
<tr>
<td>0.447</td>
<td>0.474</td>
<td>1.213</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Max. Rudder (rad.)</th>
<th>Max. Rudder (rad.)</th>
<th>Max. Rudder (rad.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.540</td>
<td>0.487</td>
<td>-0.145</td>
</tr>
<tr>
<td>0.671</td>
<td>0.533</td>
<td>0.204</td>
</tr>
<tr>
<td>-1.715</td>
<td>0.515</td>
<td>0.457</td>
</tr>
<tr>
<td>0.540</td>
<td>0.487</td>
<td>-0.145</td>
</tr>
<tr>
<td>0.671</td>
<td>0.533</td>
<td>0.204</td>
</tr>
<tr>
<td>-1.715</td>
<td>0.515</td>
<td>0.457</td>
</tr>
</tbody>
</table>

39
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 \( \omega \) decreases the time required to reach steady state also decreases but the total control effort and maximum control deflections increase. For case 9 (\( \omega = 0.04 \)) an unacceptably large aileron deflection is required (1.213 radians). Case 7 (\( \omega = 100 \)) requires a large settling time and has a large overshoot in the bank angle response. So case 8 (\( \omega = 1 \)) 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 \( \omega = 1.0 \). 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

Hall aircraft

Montgomery aircraft
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. \[ PA_{(I,J)} = (1 + \text{Rand} \times P) \ A_{(I,J)} \]
where \text{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 \[ PB_{(I,J)} = (1 + \text{Rand} \times P) \ B_{(I,J)} \]

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 IPILOT 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

A

Start

Read namelist

Calculate the eigenvalues and eigenvectors of the closed loop system. Print.

Store the eigenvalues

B

Perturb matrices A and B by P using random numbers

Calculate the eigenvalues and eigenvectors of the perturbed closed loop system.

IS there any printing required

yes Print eigenvalues and eigenvectors

no

Store the eigenvalues**

C

Calculate and store REMAX, REMIN and RTOMAX.

*Eigenvectors are not required for Monte Carlo runs

**Eigenvectors are not required for Monte Carlo runs

For statistics and plotting
REMEX = Max(real part of eigenvalues)
REMIN = Min(real part of eigenvalues)
RTOMAX = Max(ratio of imaginary to real part of eigenvalues)
Figure 11 (cont.)

C

Is there any printing required?

no

yes

Any more samples to be taken?

no

Compute the required statistics on REMAX, REMIN and RTOMAX.

is there any plotting required?

yes

Plot the eigenvalues on the versatec plotter.

no

Is there another case?

no

Stop

yes

A

REMEX = max(real part of eigenvalues)

REMIN = min(real part of eigenvalues)

RTOMEX = 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 \( \gamma \) 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 \( \gamma = 1.0 \) and 0.5. For any particular value of \( \gamma \), 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 \( \gamma = 1.0 \) and two unperturbed eigenvalues for \( \gamma = 0.04 \) are not shown on their respective
Figure 12

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = 0.1 (MIN K) TAU = 1.0

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = 0.1 (MIN K) TAU = 0.5

47
Figure 13

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = 1.1 (MIN K) TAU = 0.25

10/21/80 10:31 PM

0 2 4 6 8 10 12

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = 1.1 (ROBUST) TAU = 1.0

10/21/80 10:32 PM

0 2 4 6 8 10

48
Figure 14

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = .1 (ROBUST) TAU = 0.5

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = .1 (ROBUST) TAU = 0.25
Figure 15

HALF AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = 0.1 (LQR), RHQ = 100

HALL AIRCRAFT CLOSED LOOP EIGENVALUE PERTURBATIONS P = 0.1 (LQR), RHQ = 1.0
Figure 16

Hall Aircraft Closed Loop Eigenvalue Perturbations $P=0.1$ (LQR), $\rho=0.04$

```
9.900
2.000
1.000
-1.000
-2.000
-3.000
3.000
-10.000 -8.000 -6.000 -4.000 -2.000 0.
```

Real (LQR)
<table>
<thead>
<tr>
<th>Controller</th>
<th>Parameter</th>
<th>Statistics on REMAX</th>
<th></th>
<th>Statistics on REMIN</th>
<th></th>
<th>Statistics on RTOMAX</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Max</td>
<td>Min</td>
<td>Mean</td>
<td>$S_D$</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>Min K</td>
<td>$\tau = 1.0$</td>
<td>-0.59</td>
<td>-1.74</td>
<td>-1.02</td>
<td>0.22</td>
<td>-1.79</td>
<td>-3.95</td>
</tr>
<tr>
<td></td>
<td>$\tau = 0.5$</td>
<td>-1.25</td>
<td>-2.05</td>
<td>-1.70</td>
<td>0.16</td>
<td>-1.85</td>
<td>-3.53</td>
</tr>
<tr>
<td></td>
<td>$\tau = 0.25$</td>
<td>-2.32</td>
<td>-4.30</td>
<td>-3.20</td>
<td>0.36</td>
<td>-5.38</td>
<td>-9.59</td>
</tr>
<tr>
<td>Robust</td>
<td>$\tau = 1.0$</td>
<td>-0.78</td>
<td>-1.49</td>
<td>-1.10</td>
<td>0.12</td>
<td>-2.41</td>
<td>-3.61</td>
</tr>
<tr>
<td></td>
<td>$\tau = 0.5$</td>
<td>-1.26</td>
<td>-2.12</td>
<td>-1.80</td>
<td>0.17</td>
<td>-1.97</td>
<td>-3.59</td>
</tr>
<tr>
<td></td>
<td>$\tau = 0.25$</td>
<td>-2.08</td>
<td>-4.13</td>
<td>-3.23</td>
<td>0.28</td>
<td>-3.73</td>
<td>-7.34</td>
</tr>
<tr>
<td>LQR</td>
<td>$\nu = 100$</td>
<td>-0.28</td>
<td>-0.34</td>
<td>-0.31</td>
<td>0.01</td>
<td>-3.11</td>
<td>-3.85</td>
</tr>
<tr>
<td></td>
<td>$\nu = 1$</td>
<td>-0.94</td>
<td>-1.02</td>
<td>-0.98</td>
<td>0.02</td>
<td>-13.25</td>
<td>-16.26</td>
</tr>
<tr>
<td></td>
<td>$\nu = 0.04$</td>
<td>-0.96</td>
<td>-1.04</td>
<td>-1.01</td>
<td>0.01</td>
<td>-65.39</td>
<td>-79.69</td>
</tr>
</tbody>
</table>
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 $\tau$ decreases from 1.0 to 0.5 to 0.25. For any one of the three values of $\tau$, both Min k and Robust controllers give nearly the same amount of scatter.

The scatter obtained for the LQR controller decreases as $\tau$ 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

Montgomery A/C Closed Loop Eigenvalue Perturbations, P=1, (MIN K) TAU=0.5

10/22/80 2:59 PM
Figure 18

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = .1 (MIN K) TAU = 0.25

10/22/80  3:11 PM

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = .11 (ROBUST), TAU = 1.0

10/22/80  3:12 PM
Figure 19

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = .11 (ROBUST), TAU = 0.5

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = .11 (ROBUST), TAU = 0.25
Figure 20

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = 0.11 (LQR; RHO = 1

10/22/80 3:15 PM

3.000
-3.000
-2.000
-1.000
0.
1.000
2.000
3.000

REAL (LAMDA)

MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = 0.11 (LQR; RHO = 1

10/22/80 3:15 PM

3.000
-3.000
-2.000
-1.000
0.
1.000
2.000
3.000

REAL (LAMDA)

57
MONTGOMERY A/C CLOSED LOOP EIGENVALUE PERTURBATIONS, P = 0.11 (LQR), RHO = 0.04
10/22/60 3:18 PM

Figure 20
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Controller</th>
<th>Robust</th>
<th>LQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 1.0$</td>
<td>$\bar{K}$ = 0.16</td>
<td>-5.98</td>
<td>-3.24</td>
</tr>
<tr>
<td>$\tau = 0.5$</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
</tr>
<tr>
<td>$\tau = 0.25$</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
</tr>
<tr>
<td>$\mu = 100$</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
</tr>
<tr>
<td>$\mu = 1$</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
</tr>
<tr>
<td>$\mu = 0.04$</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
<td>Max Min SD</td>
</tr>
</tbody>
</table>
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 non-linear programming with penalty terms to impose the closed-loop eigenvalue constraints

\[ \lambda(A + BK) = 1 \]

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.
Appendix A NEWSOM Computer Program

FILE NEWSOM FORTRAN

JULY 22, 1990

REAL*8 DSFED
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/OUT/NODEV
COMMON/ISEED/ISEED
NAMFIL/NSOMIN/N,M,FPS,MAXF4,NLOOPS,NSRCH,NODEV,IPRINT,ICONT,IOUT
1 ,REMIN,REMAX,RTOMAX,WT1,WT2,WT3,WT4,WT5,ISKIP,A,B,X
NAMFIL/NSMOUT/N,M,A,B,X

C INPUTS
N=DIMENSION OF THE STATE
M= DIMENSION OF CONTROL
FPS=CONVERGENCE TOLERANCE OF THE POWELL ITERATION
NLOOPS=MAX NO OF POWELL ITERATION LOOPS
NSRCH=MAX NO OF POINTS ON A LINE SEARCH
MODEV=SPECIFIES THE OUTPUT DEVICE (6=TERMINAL, 8=FILE NEWSOM OUTPUT)
(IPRINT=CONTROLS THE AMOUNT OF PRINTOUT. 1=NORMAL PRINTOUT)
ICONT=1,10 CONTINUE AFTER THIS CASE
0, TO STOP AFTER THIS CASE
IOUT=1, TO WRITE NAMELIST BLOCK IN OUTPUT
0, NOT TO DO THIS
WT1...WT5=WIGHTS USED IN COST FUNCTION
REMIN=LEFT BOUNDARY OF E—VALUE DOMAIN
REMAX=RIGHT BOUNDARY OF E—VALUE DOMAIN
RTOMAX=MAXIMUM RATIO IM(LAMDA)/RF(LAMDA)
A=N*N SYSTEM MATRIX
B=N*M CONTROL MATRIX
X=M*N INITIAL FEEDBACK MATRIX

DATA NCASE/0/
CALL IRRSET(208,256,-1,1)
ISEED = 566387
5 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(IT4,IT,DY,HN,AP,'NEWSOM ',NODEV)
ISEED = IH*100 + IMN
IF(IPRINT.GE.3)WRITE(NSOMIN)
NVARS = N*M
(1) IF(X(1).NE.-1.)GO TO 9
CALL RANDS(ISFEN,NVARS,X)
WRITE(NSOMIN,1)ISEED,(X(I),I=1,NVARS)
101 FORMAT(/,** GGUFRS CALLED WITH ISEED =',I20,
1 /,' RAND NOS =',10F7.4,/,11X,10F7.4)
IF 6 I = 1, NVARS
6 X(I) = 2.*X(I) - 1.
8 IF(IPRINT.GE.1)CALL WRMTMAT(A,N,N,4,'A ')
62
Appendix A NEWSOM Computer Program

```fortran
IF(IPRINT.GE.1)CALL WRTMAT(B,N,M,4, 'B ', FPRINT) 
IPRINT = IPRINT + 8 
fmin = COSTF(X) 
IPRINT = IPRINT - 8 
call powell(nvars,X,FMIN,EPS,IPRINT,MAXFN,NLOOPS,NSRCH) 
IPRINT = IPRINT + 8 
FMIN = COSTF(X) 
IPRINT = IPRINT - 8 
IF(ICONT.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 fig(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/print/printf 
common/ncost/ncost 
common/nnode/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,4,m,5) 
call matadd(a,d1,d1,n,n,4,5,5) 
C do 12 i = 1,n 
C 12 do 12 j = 1,n 
12 d1(i,j) = x(i)*a(i,j)/x(j) 
    if(iprint.lf.8)go to 15 
call wrtmat(x,m,n,w,'feedback') 
call wrtmat(d1,n,5,'v,n,5,'') 
15 ijob = 1 
call eigrf(d1,n,5,ijob,ei,evec,5,d2,ier) 
    if(ier.gt.0)write(nodev,130)ier 
130 format(/ 'error in eigrf in costf  ier =',i3) 
do 30 i = 1,n 
30 er(i) = real(eig(i)) 
i1(i) = aimag(eig(i)) 
if(e1(i).lt.remin)if2 = f2 + wt2*(er(i)-remin)**2 
    if(ier(i).gt.remax)if3 = f3 + wt3*(er(i)-remax)**2 
    ratio = abs(e1(i)/er(i)) 
    if(ratio.gt.rtomax)f4 = f4 + wt4*(ratio-rtomax)**2 
    temp = 0. 
do 20 j = 1,n 
20 d1(j,i) = rfal(evec(j,i)) 
30 d2(i,j) = 0. 
    if(e1(i).lt.0.)d1(j,i) = aimag(evec(j,i)) 
```

63
Appendix A NEWSOM Computer Program

IF(I.EQ.J)D2(I,I) = 1.
20 TEMP = TEMP + D1(J,J)*D1(J,J)**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
         TEMP = TEMP + D1(K,I)*D1(K,J)
      ARG = TEMP/(D4(I)*D4(J))
      D3(I,J) = 57.2958*ACOS(ABS(ARG),1.)
   36 CONTINUE
   FI = FI + (90.-D3(I,J))**10
   IF(IPRINT.GE .5)
      WRITE(NODEV,104)(I,J,03(I,J),J=t,Il)
      104 FORMAT(/,5(ANGL,11.0,'*','I1=',',F7.2))
38 CONTINUE
   FI = WT5*(FI**.1)
   IF(IPRINT.LT .5)GO TO 60
   CALL LEQT2F(D1,N,N,D2,0,0,0)
   IF(19R.GT.0)WRITE(N00EV,105)IER
      105 FORMAT(/,'ERROR IN LEQT2F I= ', I6)
   01N = 0.
   D2N = 0.
   FR = 0.
   DO 50 I = 1,N
      TEM = 0.
      DO 45 K = 1,N
         TEM = TEM + D1(K,I)*D1(K,J)
      IF(I.EQ.J)TEM = TEMP - 1.
      FRR = FRR + TEMP**2
      45 CONTINUE
      D1N = D1N + D1(I,J)**2
      D2N = D2N + D2(I,J)**2
      FR = SQRT(FRR)
      D1N = SQRT(D1N)
      D2N = SQRT(D2N)
   50 CONTINUE
   F = F1 + F2 + F3 + F4 + F5
   C0STF = F
   IF(IPRINT.LT.5)RETURN
   WRITE(NODEV,106)FRR,01N,D2N
      106 FORMAT(/,* IN COST F ERR, D1N, D2N =',1P3E12.4)
   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,FI
      101 FORMAT(* FI,F2,F3,F4,F5,FI,FT =',6G12.4)
   WRITE(NODEV,102)ER(I),I=1,N
      102 FORMAT(* ER(I),I=1,N)
   WRITE(NODEV,103)EI(I),I=1,N
      103 FORMAT(* IMAG EIG =',5G12.4)
   C0STF = F
   RETURN
END
Appendix A  NEWSOM Computer Program

**SUBROUTINE DATIME *******

SUBROUTINE DATIME(IMO, IDAY, IYR, IHOURS, IMIN, AMPM, PNAME, NODEV)
REAL*8 PNAME
DATA A4/' 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
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,'/',A2,'/',A2,'/',A2,'/',A4)
RETURN
END

**SUBROUTINE RANDS ******

SUBROUTINE RANDS(IX, N, Z)
DIMENSION Z(1)
DATA 4/1048576/FM/1048576./IA/1027/
DO 10 I = 1,N
IX = MOD(IA*IX, M)
FX = IX
Z(I) = FX/FM
RETURN
END

**SUBROUTINE POWELL ******

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, ITI, ITP, IGO, NONU
COMMON/NNOUT/NODEV
DATA DETMIN/.01/
LIMIT = NLOOPS
LIMS = NSRCH
CALL TIMFON
INFO = IPRINT
NCOST = 0
NONU = 0
*N = N
IT = 1
N1 = N + 1
NP = 1.5*N
F = COSTF(X)
F1 = F

65
Appendix A NEWSOM Computer Program

```
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=','3(I3,',' EPS=',G12.4,
1     '/' F=',G14.7,
1     '/' X=',5F14.4,/(3X,5F14.4))
IF(INFO.GF.1)WRITE(NODEV,104)

104 FORMAT(* IT IS LIT ITO ITP IGO NONU NCOST*,2X,'COS','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
5  D(J,I) = 0.
10  D(I,I) = 1.
   GO TO 15

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(1),I=1,N

107 FORMAT(* ISEED,INDX =',111,25I3)
DO 20 I=1,N
 I = INDX(I)
 NPCOL = IN
 NDCOL = I
   CALL MINPT(DIST(I),F,EPS,LIMS)
   IF(INFO.GF.2)
    WRITE(NODEV,100)I,LIT,IT0,IT1,IT2,IGO,NONU,NCOST,F,DIST(I)

100 FORMAT(1X,I5,5I4,15,I6,2G13.6)
IF(INFO.GF.2)NONU = 0
   IF(ABS(DIST(I)) .GE. DMAX)GO TO 18
   DMAX = ABS(DIST(I))
   IS = I
   GO TO 18

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
5  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
```

66
Appendix A \textit{NEWSOM} Computer Program

\begin{verbatim}
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)
     WRITE(NODEV, 101) IT, IS, LIT, IT0, IT, ITP, IGO, NONU, NCOST, F, DIST(N1), I
   ERR
101  FORMAT(1X, I29, 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
   CHECK TO SEE IF THE NEW SEARCH DIRECTION SHOULD BE USED
   DETN = DET*OMAX/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,
     ' FUNCTION CALLS. ', ' INITIAL COST = ', 1P5E14.7, ', ', ' FINAL COST = ', 1E14.7,
     ', TIME = ', 1PF10.5, ', X = ', 1P5E14.7, 1E14.7))
RETURN
\end{verbatim}
SUBROUTINE MINPT(DMIN,CMIN,EPS,LIMIT)
COMMON/IMINPT/IT,ITO,ITP,IGO,NONU
COMMON/NMOUT/NODEV
COMMON/NPRINT/NPRINT
DATA BIG,DLTOL/1.E10,.1180339886/
DATA DISTMN/1.E-6/,VBIG/1.E20/
D = DMIN
ZERO = 0.
DMIN = VBIG
IGO = 3
CALL COSTD(D,FR,C,DMIN,CMIN)
C = C
IF(ABS(D).LT.DISTMN)D = DISTMN
DELTA = ABS(D)/2.
D2 = 0.
C2 = C
SIDE = SIGN(1.,D)
IT = NO. OF POINTS IN LINE SEARCH
ITP = NO. OF PARABOLIC FITS
ITO = NO. OF OUTWARD STEPS
ITI = NO. OF INWARD STEPS
IGO = 0 FOR EXIT ON OUTWARD SEARCH
1 FOR EXIT ON INWARD SEARCH
2 FOR EXIT ON PARABOLIC SEARCH
IT = 0
ITP = 0
ITO = 0
ITI = 0
BEGIN THE OUTWARD SEARCH
IGO = 0
D1 = -BIG
D3 = BIG
C1 = VBIG
C3 = VBIG
GO TO 20
CHANGE THE DIRECTION OF THE OUTWARD SEARCH
10 IF(D1.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
68
Appendix A NEWSOM Computer Program

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 DO PARABOLIC FIT
C
45 IGO = 2
DTEST = .7*ABS(D1-D3)
CALL MINPAR(D1,D2,D3,C1,C2,C3,C0,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 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,D2,D3,C1,C2,C3
101 FORMAT(/,3X,'D1',24X,'D2',13X,'D3',13X,'C1',24X,'C2',13X,'C3',
       1 /,1X,6G13.6,13X,2G13.6,13X,G13.6)
CALL COS40(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(1,' CAUTION...FUNCTION IS NOT UNIMODAL, D1-3, C1-3 =',
       1 /,1X,6G10.3)
C
C TAKE AN INWARD STEP

69
Appendix A  NEWSCM Computer Program

C

50  D = (D3*D1/2) + SIDE*DTAU*(D3-D1)
    IF(PRINT.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)
      WRITE(NODEV,103)D1,D2,D3,D,C1,C2,C3,C
     FORMAT(//,"CAUTION...FUNCTION IS NOT UNIMODAL. D1-3,D,\'\'\',C1-3,C =\',/1X,8G10.3)
    ITI = ITI + 1
    ERR = AMIN1(ABS(C),ABS((C-C2)/C2))
    IF(ERR.LE.EPS.AND.C.LT.CO )RETURN
    IF( I.T.GE.LIMIT)RETURN
    IT = IT + 1
    SIDE = SIGN(1.,(C2-C)*SIDE)
    DT = D
    CT = C
    IF(C.GT.C2)GO TO 60
    DT = 02
    CT = C2
    02 = 0
    C2 = C
    60 IF(SIDE.GT.0.)GO TO 62
    D3 = DT
    C3 = CT
    GO TO 40
    62 C1 = DT
    GO TO 40
    END

C

SUBROUTINE MINPAR(X1,X2,X3,F1,F2,F3,FO,FPrICONV,DMIN,CMIN)
COMMON/NODEV/COMMON/NPRINT/NODEV
COMMON/IPRINT/IPRINT
ICONV = 0
A1 = X2 - X3
A2 = X3 - X1
A3 = X1 - X2
DEN = F1*A1 + F2*A2 + F3*A3
IF(DEN.FB.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(IPRINT.GE.4)WRITE(NODEV,100)X1,X2,X3,F1,F2,F3
      /1X,8G10.3)
    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)
      WRITE(NODEV,101)X1,X2,X3,X4,F1,F2,F3,F4
     FORMAT(//,"CAUTION...FUNCTION IS NOT UNIMODAL. D1-4,C1-4 =\',\',1
      /1X,8G10.3)
    ER = AMIN1(ABS(F4),ABS((F4-F2)/F2))
APPENDIX A NEWSOM COMPUTER PROGRAM

IF (FR-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), INDX(25), RAND(25)
REAL*8 DSEED
COMMON /ISFED/ ISFED
COMMON /NOUT/NODEV
DO 10 I = 1, N
10  IND(I) = I
C WRITE (N)DEV, 100) ISFED, N, DSEED)
100  FORMAT (/,' IN RANDIN, ISFED, N, DSEED = ', I12, I4, 1PD14.6)
    CALL RANDS (ISFED, N, RAND)
C WRITE (N)DEV, 101) N, DSEED, (RAND(I), I = 1, N)
101  FORMAT (/,' AFTER GUNS IS CALLED, N, DSEED = ', I4, 1PD14.6,
        /,' RAND = ', 10F8.5)
   DO 20 K = 2, N
      I = N + 1 - K
      RI = I
      II = IFIX (RI*RAND(I)) + 1
      II = MAX(1, MINO(I, II))
      INDX(I) = IND(II)
      IF (II.EQ.I) GO TO 20
      II = II + 1
   DO 15 J = II, 1
   15  CONTINUE
      INDX(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
Appendix A  NEWSOM Computer Program

C
SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
DIMENSION A(IA,1)
COMMON/YNOUT/NPRINT
REAL*8 ANAME
WRITE(NPRINT,100)ANAME,N,M
100 FORMAT(/,' MATRIZ ',A6,3X,'(',I3,','R,W,3x,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(1P10F13.5)
RETURN
END
C
C  TRANSP - TRANSPOSES A MATRIX
C
SUBROUTINE TRANSP(A,N1,N2,NA)
DIMENSION A(NA,1)
COMMON/YNOUT/NOUT
M = MAX0(N1,N2)
IF(M.GT.NAOR 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 *** N102rNA =',3(I3))
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/YNOUT/NOUT
IF(N.GT.MIN0(NA,NC)OR L.GT.NB)GO TO 90
DO 20 I=1,N
DO 20 J=1,NC
TEMP = 0.
DO 10 K=1,L
TEMP = TEMP + DBLE(A(I,K))*DBLE(B(K,J))
10 C(I,J) = TEMP
RETURN
90 WRITE(NOUT,100)N,NA,NC,L,NB
100 FORMAT(' *** ERROR IN MULT *** N,NA,NC,L,NB =',5(I3))
RETURN
72
C  MSHIFT — TRANSFERS VECTORS AND MATRICES

SUBROUTINE MSHIFT(A,B,N,NA,NB)
DIMENSION A(NA,1),B(NB,1)
COMMON/NNOT/NOUT
IF(N.GT.MINO(NA,NB))GO TO 90
DO 10 I=1,N
10 DO 10 J=1,4
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  MATADD — MATRIX ADDITION

SUBROUTINE MATADD(A,B,C,N,4,NA,NB,NC)
DIMENSION A(NA,1),B(NB,1),C(NC,1)
COMMON/NNOT/NOUT
IF(N.GT.MINO(NA,NB,NC))GO TO 90
DO 10 I=1,N
10 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  MATSUB — MATRIX SUBTRACTION

SUBROUTINE MATSUB(A,B,C,N,M,NA,NB,NC)
DIMENSION A(NA,1),B(NB,1),C(NC,1)
COMMON/NNOT/NOUT
IF(N.GT.MINO(NA,NB,NC))GO TO 90
DO 10 I=1,N
10 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  SWAP — INTERCHANGES TWO VARIABLES

SUBROUTINE SWAP(A,B)
C = A
A = B
B = C
RETURN
END
Appendix A NEWSOM Computer Program

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 IMAT — LOADS AN ARRAY WITH THE IDENTITY MATRIX
C
SUBROUTINE IMAT(A,N,NA)
DIMENSION A(NA,1)
DO 5 I=1,N
DO 5 J=1,N
5 A(I,J) = 0.
10 A(I,I) = 1.
RETURN
END
C
C ANORM — CALCULATES THE RSS NORM OF A MATRIX
C
FUNCTION ANORM(A,N1,N2,NA)
DIMENSION A(NA,1)
COMMON/NOUT/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)N1,NA
100 FORMAT(* *** ERROR IN ANORM *** N,NA =',2I5)
RETURN
C FILE NEWSOM INPUT
C
C JULY 22, 1980
C
C NEWSOM
Appendix A  NEWSOM Computer Program

\[ N = 4, \quad M = 2, \]
\[ \text{EPS} = 1.E-7, \]
\[ \text{MAXFN} = 5000, \]
\[ \text{NLOOPS} = 40, \]
\[ \text{NSRCH} = 12, \]
\[ \text{NDFV} = 6, \]
\[ \text{PRINT} = 1, \]
\[ \text{ICONT} = 1, \]
\[ \text{ICUT} = 0, \]
\[ \text{WT1} = .01, \quad \text{WT2} = 100., \quad \text{WT3} = 500., \quad \text{WT4} = 100., \]
\[ \text{WT5} = 20., \]
\[ \text{REMIN} = -20., \quad \text{REMAX} = -1., \quad \text{RTOMAx} = .5, \]
\[ \text{A} = -0.367984, 1., -0.024209, 0.258819, 3*0., 0.017285, -0.032279, \]
\[ 0.267949, -0.110395, -0.965926, 26.1875, 0., 4.46294, -0.091072, \]
\[ \text{B} = -7.67183, 0., 1.96959, 0., 2.06549, 0., -2.33843, 5*0., \]
\[ \text{X} = 1.8794, -6.33, 3.59, 0.1327, 2.366, 4.141, -1.313, 0.683, \]
\[ \text{X} = 1.494, -4.193, 1.054, .0203, 1.6827, \]
\[ 3.658, -2.315, -0.4795, \]
\[ \text{X} = 6.481, -4.298, -1.054, -2.476, 1.6827, \]
\[ 3.678, -7.3046, -0.4789, \]
\[ \text{REM} = -2., \]
\[ \text{END} \]
\[ \text{NSIMIN} \]
\[ \text{REMAX} = -2., \]
\[ \text{END} \]
\[ \text{NSIMIN} \]
\[ \text{REMAX} = -4., \]
\[ \text{END} \]
\[ \text{NSIMIN} \]
\[ M = 3, \]
\[ A = -3.18, 1., -0.06, 3.022, 3*0., 0.0644, 0.63, 0., -0.27, -0.998, \]
\[ -10.6, 0., 4.18, -0.151, \]
\[ \text{B} = -14.4, 3*0., 1.5, 0., -2.59, 0.037, 2*0., -0.96, 0., \]
\[ X = -2.23E-8, -3.5E-8, 3.43E-9, 1.73E-10, .45, -9.92, -1.8, -1.76, 10.7, 78, -5.9 \]
\[ X = -3.97E-2, -4.56E-3, 1.99E-2, 1.232E-2, -9.49, \]
\[ X = -5.593, 2.1612, -3.6611E-3, 4.7516, -9.4851, \]
\[ X = -5.4778, -5.141, \]
\[ X = -3.7498, -1.5196, -2.329, 1.373, -0.3295, \]
\[ 2.5753, -2.175, 2.6445E-3, 5.5306, -0.4673, \]
\[ -0.695, -5.259, \]
\[ \text{REMAX} = -1., \]
\[ \text{END} \]
\[ \text{NSIMIN} \]
\[\text{REMAX} = -2., \]
\[ \text{END} \]
\[ \text{NSIMIN} \]
\[\text{REMAX} = -4., \]
\[ \text{ICONT} = 0, \]
\[ \text{END} \]
FILE LINEAR FORTRAN

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/PLT/SCALE, ARRAY(101, 11)
KIN=3
KOUT=4
NDIM=10
N=4
M=3
NR=4
CALL MATIO(N,N,A,4)
CALL EQUATE(N,N,A1,A)
CALL MATIO(N,M,B,4)
CALL EQUATE(N,M,B1,B)
CALL MATIO(IR,N,C,4)
CALL MATIO(N,N,Q,4)
CALL MATIO(4,4,R,4)
CALL CON(N,M,A,B,NCS)
CALL ORS(N,M,A,C,Q,ACL)
CALL REG(N,M,A,B,R,Q,K,G,ACL)
CALL VECTIO(N,XC,4)
CALL VECTIO(M,UE,4)
CALL VECTIO(M,UREF,4)
CALL MATIO(M,N,G,4)
CALL LTRACK(A1,N,B1,M,G,XC,UE)
STOP
END

SUBROUTINE REG

SUBROUTINE REG(NM,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(NM,A,RR,RI)
WRITE(KOUT,200)
WRITE(KOUT,300)(RR(I),RI(I), I=1,N)
CALL EQUATE(M,M,DUM1,R)
CALL MINV(M,M,DUM1,DUM2, MR,0)
CALL MAT4(N,M,DUM2,B,G)
CALL MRIC(N,M,A,G,Q,X,ACL)
CALL EQUATE(M,M,DUM1,R)

76
CALL GMINV(M,4,DUM1,DUM2,MR,G)
CALL MAT6(4,N,DUM2,B,DUM1)
CALL GMUL(DUM1,X,M,N,N,S)
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,90)
CALL MATIO(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 SUBROUTINE WRIC

SUBROUTINE WRIC(N,A,S,Q,X,Z)
DIMENSION A(1),S(1),Q(1),X(1),Z(1),TR(3),TIMES(3)
COMMON/Main1/NDIM,F(1)
COMMON/INOU/KIN,NOUT
NDIM=NDIM+1
TIMES(1)=.5
TIMES(2)=2.0
TIMES(3)=4.0
NN=N*NDIM
T1=-.5*ALOG(XNORM(Q)*.5001)
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
20 DO 15 J=I,NN,NDIM
15 X(J)=-S(J)
CALL INTEG(N,A,X,Z,-T1)
CALL FACTOR(N,Z,F,M)
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,G)
CALL MAT2(N,N,Z,Z,X)
C A+SX IS STABLE
TOL=1.E-5
ADV=TOL*1.E-7
NN=N*NDIM
NN1=N-1
DO 19 I=1,N
19 WRITE(KOUT,60)
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
I1=I+NM1
DO 20 J=I,YI
X(J)=A(I,J)-F(J)
20 Z(J)=Z(J)+O(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)
IT=II+NDIM1
25 C1=C1+TR(I)
IF(ABS(C1).GT.1.E20) GO TO 50
IF(L.NE.N)GO TO 40
CALL SMINV(N,N,Z,F,MR,O)
CALL MMUL(S,X,N,N,N,Z)
WRITE(KOUT,27)IT
27 FORMAT(17HORICC ATI SOLN IN I2,I2,11H ITERATIONS)
DO 30 I=1,NN,NDIM
II=I+NM1
DO 30 J=I,YI
30 Z(J)=A(J)-Z(J)
IF(MR.NE.N)WRITE(KOUT,35)MR
35 FORMAT(26HORICC ATI SOLN IS PSD--RANK13)
RETURN
40 CONTINUE
WRITE(KOUT,45)MAXIT,T1
45 FORMAT(26HORICC ATI NON-CONVERGENT INI2,I2,11H ITERATIONS,12H INITIAL
1 T=F10.5)
GO TO 60
50 WRITE(KOUT,55)IT,T1
55 FORMAT(29HORICC ATI BLOW JP AT ITERATIONI2,I2,12H 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
SUBROUTINE MLINEQ
C
SUBROUTINE MLINEQ(N,A,C,X,TOL)
C SOLVES A*X+X*A+C=0
C A AND X CAN BE IN SAME LOCATION IF DESIRED
C ANSWER RETURNED IN C AND X
DIMENSION A(I),C(I),X(I),RI(30),RI(30)
Appendix B LINEAR Computer Program

COMMON/MAIN1/NDIM,F(1)
COMMON/MAIN2/Y(I)
COMMON/INOUT/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
  20 JJ=I,NN,NDIM
  Y(JJ)=DT*A(JJ)
  Y(II)=Y(II)-.5
  II=II+NDIM1
  CALL GMINV(N,NrY,F,MR,I)
  CALL EQUATE(NrN,Y,A)
  TF(MR.EQ.N) ;0 TO 21
  IT=0
  DO 18 I=1,NN,NDIM1
  18 C(I)=1.E25
  GO TO 95
  21 CALL *MUL(C,F,N,N,N,X)

C INITIALIZATION OF X,F
I=1
DO 40 I=1,NN,NDIM
J=I1
IF(I.EQ.1) GO TO 30
  DO 25 JJ=I,II,NDIM1
  C(J)=C(JJ)
  J=J+1
  25 J=J+1
  30 J=J+1
  DO 35 JJ=II,NN,NDIM1
  C(J)=DT*DOT(N,F(II),X(JJ))
  J=J+1
  F(ID)=F(ID)+1.0
  35 J=J+1
  40 I=I+1
  ADV=TOL*1.E-7
  DO 90 IT=1,30
  NEZ=0
  SIZE=0.0
  CALL *MUL(C,F,N,N,N,X)
  I=1
  90 IT=IT+1
  70 J=1
  GO TO 70
  60 J=II
  DO 65 JJ=I,II,NDIM1
  C(J)=C(JJ)
  J=J+1
  65 J=J+1
  70 J=J
  DT1=C(JJ)
  DO 75 JJ=II,NN,NDIM1
Appendix B  LINEAR Computer Program

C(J)=C(J)+DOT(N,F(I1!#X(JJ))
75 J=J+1
J=J-1
DO 80 JJ=I1,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).LT.1,F25) 50 TO 95
CALL MMUL(X,X,N,N,N,F)
CONTINUE
90 WRITE(KOUT,100)IT
100 FORMAT(33HOLIN EQN ALGORITHM NON-CONVERGENT,13,10HITERATIONS)
WRITE(KOUT,110)
110 FORMAT(35H0A 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)
CONTINUE
150 CALL FQUATE(N,N,X,C)
RETURN
END

SUBROUTINE INTEG

SUBROUTINE INTEG(N,A,C,S,T)
S=INTEGRAL EA*C*EA' FROM O TO T
C IS DESTROYED
DIMENSION A(1),C(1),S(1),COEF(15)
COMMON/MAIN1/NDIM,X(1)
NDIM1=NDIM+1
NY=N*NDIM1
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=ItNV,NDIM
J=I+NM1
DO 15 JJ=I,J
15 S(JJ)=DT*C(JJ)
T1=DT**2/2.
DO 25 IT=3,17
CALL MMUL(A,C,N,N,N,X)
DO 20 I=1,N
II=(I-1)*NDIM1
DO 20 JJ=I,NN,NDIM1
CONTINUE
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,NT+1
II=NT-I
30 COEF(I)=DT*COEF(I+1)/FLOAT(I)
DO 40 I=1,NN,NDIM
I=1
J=I+1
DO 35 JJ=I,J
35 X(JJ)=A(JJ)*COEF(I)
X(II)=X(II)+COEF(I)
40 II=II+NDIM
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+1
DO 50 JJ=I,J
50 X(JJ)=C(JJ)
X(II)=X(II)+T1
55 II=II+NDIM
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=II+1
75 DO 95 JJ=I,J
KK=JJ
DO 90 K=1,NN,NDIM
S(J)=S(J)+C(J)*X(KK)
90 KK=KK+NDIM
J=II+NDIM
DO 87 J=I,NN,NDIM
37 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 'MMUL
C
SUBROUTINE GMINV(NR,NC,A,U,MR,MT)
DIMENSION A(1),U(1),S(30)
COMMON/MAIN1/NDIM
COMMON/INOU/KIN,KCUT
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
JR=JJ+NRM1
JC=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=TOL1+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(JJ),U(JJ))
CALL VADD(NR,TEMP,A(JJ),A(KK))
65 KK=KK+NDIM
FAC=DOT(J,U(JJ),U(JJ))
NR=NR-1
Appenlix B  LINEAR Computer Program

GO TO 75
70 S(JJ)=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./SORT(FAC)
    DO 80 I=JJ,JCM
        A(I)=A(I)*FAC
80    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 FORMA(I3,1HX,I2,8H M: RANKP I2)
120 NEND=NC*NDIM
    JJ=1
    DO 135 J=1,NC
        N   =I+J
        S(J)=0.
        DO 125 K=J,JNEND,NDIM
            U(K)=S(K)+A(K)*U(K)
125 II=JJ+NDIM
    JJ JJ+NDIM
    RETURN
END

C SUBROUTINE FACTOR
C
SUBROUTINE FACTOR(N,A,S,MR)
C A=S*S
DIMENSION A(1),S(1)
COMMON/MAINZ/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 5 JJ=1,NN,NDIM
        S(JJ)=0.
        ID=II+I41
Appendix B LINEAR Computer Program

\[ R = A(I) - \text{DOT}(I, S(I), S(I)) \]

IF \((-\text{ABS}(R)) > (TOL * A(I) + TOL1)) \) GO TO 50
IF \((R) > (TOL * A(I) + TOL1)) \) GO TO 50

15 MR = -1
WRITF(KOUT, 1000)

1000 FORMAT(37HOTRIED TO FACTOR AN INDEFINITE MATRIX)
RETURN

20 S(I) = \sqrt{R}
MR = MR + 1
IF \((I.EQ.N)) \) RETURN
L = I + NDIM
DO 25 JJ = L, NN, NDIM
IJ = JJ + I
25 S(IJ) = (A(IJ) - \text{DOT}(I, S(I), S(JJ))) / S(I)
RETURN
END

C
SUBROUTINE MEIGV
C
SUBROUTINE MEIGV(N, A, R, RR, RI)
DIMENSION A(1), R(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(I) = A(J)
C(N+1) = 1.0
L = 1
5 CL = 0.0
DO 10 I = 1, NN, NDIM
10 CL = CL - X(I)
C(I) = CL / FLOAT(L)
I = N + 1 - L
C(I) = C1
DO 15 I = 1, NN, NDIM
15 X(I) = X(I) + CL
15 IF \((L.EQ.1) \) GO TO 50
DO 40 I = 1, N
JJ = 1
DO 25 J = I, NN, NDIM
5 C1 = 0.0
KK = J - 1
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)
35 JJ = JJ + 1
JJ = 1
40 JJ = JJ + 1
L = L + 1
GO TO 5
SUBROUTINE POLRT

SUBROUTINE POLRT(A,N,U,V)

DETERMINES THE ROOTS OF AN N-TH ORDER POLYNOMIAL

X**N + A(N)*X**(N-1) + ... + A(1) = 0

WHERE: U(N) WILL CONTAIN THE ROOT REAL PARTS

V(N) WILL CONTAIN ROOT IMAGINARY PARTS

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) - F
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 F = U(2)/A(1)
GO TO 43
42 E = U(2)/A(2)
43 F = AMAX1(ARS(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 C=BAR = V(2) - U(2)
IF (NR.GT.3) V4 = V(4)
D = V(3)**2 - C=BAR*V4
IF (D) 47,46,47
46 P = P - 2.0
Appendix B  LINEAR  Computer Program

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
F = ABS(U(1)/A(1))
IF (F.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 (F.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
SUBROUTINE FQUATE

SUBROUTINE EQUATE(NR,NC,A,B)
DIMENSION A(O),B(O)
COMMON/MAIN1/NDIM
NN=NC*NDIM
NR1=NR-1
DO 1 J=1,NN,NDIM
II=J+NR1
DO IJ=J,I
A(IJ)=B(IJ)
1 CONTINUE
RETURN
END

SUBROUTINE MAT4

SUBROUTINE MAT4(N1,N2,X,Y,Z)
DIMENSION X(0),Y(0),Z(0)
COMMON/MAIN1/NDIM
CALL MMUL(Y,X,N1,N2,N2,Z)
NN2=N2*NDIM
DO 5 I=1,N1
II=I-1
JJ=II+1*NDIM
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=J+NDIM
JJ=I
K=II+1
KK=II+1*NDIM
DO 5 J=K,KK
Z(JJ)=Z(J)
JJ=JJ+NDIM
5 CONTINUE
RETURN
END

SUBROUTINE MAT6

SUBROUTINE MAT6(N1,N2,N3,X,Y,Z)
DIMENSION X(0),Y(0),Z(0)
COMMON/MAIN1/NDIM
DO 2 I=1,N1
DO 2 J=1,N3
TM=0.
2 CONTINUE
RETURN
END
APPENDIX B LINEAR COMPUTER PROGRAM

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 SUBROUTINE MMUL
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=1,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 SUBROUTINE MATIO
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 INPUT
DO 10 I=1,NR
READ(KIN,1000)(X(IJ),IJ=I,JEND,NDIM)
10 CONTINUE
IF(IO.EQ.1)RETURN

C OUTPUT
20 DO 30 I=1,NR
WRITE(KOUT,1001)(X(IJ),IJ=I,JEND,NDIM)
30 CONTINUE
GO TO 50

C TITLE
40 READ(KIN,1002)
WRITE(KOUT,1003)

88
Appendix B LINEAR Computer Program

```
WRITE(KOUT,1002)
WRITE(KOUT,1004)
GO TO 5
5 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

FUNCTION XNORM(N,A)
  COMMON/MAIN1/NDIM
  NDIM=NDIM+1
  NN=N*NDIM
  C1=0.
  TR=A(1)
  IF(N.E.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=I,NN,NDIM1
      15 C1=C1+(A(II)-TR)**2
  XNORM=C1/SQR(C1)
RETURN
END

SUBROUTINE VADD

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

SUBROUTINE MAT2

SUBROUTINE MAT2(N1,N2,X,Y,Z)
```
C Z = X * Y, X, Y = N1 * N2, Z = Z'
C Z AND Y CAN BE EQUIVALENT
DIMENSION X(1), Y(1), Z(1)
COMMON/MAIN1/NDIM
NDIM = NDIM + 1
NN2 = N2 * NDIM
II = 1
DO 10 I = 1, N1
   IJ = II
   DO 5 J = 1, N1
      Z(IJ) = DOT2(NN2, X(I), Y(J))
   5 IJ = IJ * NDIM
   J = II
   IJ = J
   DO 3 I, J = IJ - NDIM
   IF(IJ .LT. I) GO TO 10
   J = J - 1
   Z(IJ) = Z(J)
   3 IJ = IJ - NDIM
   IF(IJ .LT. I) GO TO 10
   J = J - 1
   Z(IJ) = Z(J)
   GO TO 10
2 I = II + NDIM
END

FUNCTION DOT2(NN, A, B)
DIMENSION A(N), B(1)
COMMON/MAIN2/NDIM
DOT2 = 0
DO 1 I = 1, NN, NDIM
   DOT2 = DOT2 + A(I) * B(I)
1 CONTINUE
RETURN
END

SUBROUTINE REGSIM

SUBROUTINE REGSIM(N, M, IR, ACL, G, C, XO, DT, NS, NP)
DIMENSION ACL(N), G(N), C(N), XO(N)
COMMON/MAIN2/NDIM, DUM1(N)
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. M) CALL EXIT
IF(NDIM .LT. IR) CALL EXIT
1000 FORMAT(/,16H DIMENSION ERROR,/) LOGICAL CONT
CONT = .TRUE.
CALL LNSIM2(N, IR, ACL, G, C, XO, DT, NS, NP, M, CONT)
RETURN
END

SUBROUTINE LNSIM2

SUBROUTINE LNSIM2(N, R, A, C, XO, DT, NSTEPS, NPRPL, M, G, CONT)
DIMENSION A(N), C(N), XO(N), G(N)
Appendix B  LINEAR  Computer Program

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, DUMI(1)
COMMON/MAIN2/FA(1)
LOGICAL PRINT, PLOT
IF (CONT) CALL MSCALE(4, 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) 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(4, 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 50 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(4, 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 * T
IF (.NOT. CONT) GO TO 60
Appendix B LINEAR Computer Program

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(I)=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(I)=R+I
WRITE(KOUT,115) I
70 CALL PRNPLT(NR,NC,NVAR,ARRAY,INDEX,NHORZ,NGRIDH,NSORT,NSYM)
IF(.NOT.CONT) GO TO 120
NSYM(1)=11
FORMAT(/31H SIMULATION OF LINEAR REGULATOR,/) 
90 FORMAT(/,31H SIMULATION OF LINEAR SYSTEM/) 
95 FORMAT(/11H OUTPUT Y) 
99 FORMAT(12H CONTROL U) 
100 FORMAT(/5H T = ,F5.2,/) 
105 FORMAT(1X,0P10,E13.4) 
110 FORMAT(1H1/,31X,1HY,12H VERSUS TIME) 
115 FORMAT(1H1/,31X,1HY,12H VERSUS TIME) 
120 CONTINUE
RETURN
END

SUBROUTINE MEXP

SUBROUTINE MEXP(N,A,T,EA)
DIMENSION A(1),EA(1),C(30),D(31),E(30)
COMMON/MAIN1/NDIM,X(1)
NDIM=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=1,IL
10 EA(J)=A(I,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
Appendix B  LINEAR Computer Program

GO TO 15

20 C2 = 0.
25 DO 25 I = 1, NN, NDIM
25 C2 = C2 - EA(I)
 C2 = C2 / FLOAT(L)
 C(L) = C2
 DO (L+1) = 0.
 II = N + 1 - L
 E(I) = W
 II = 1
 DO 35 I = 1, NN, NDIM
 IL = I + NM1
 DO 30 J = I, IL
 30 X(J) = FA(J)
 X(I) = X(I) + C2
 35 II = II + NDIM1
 IF (L .EQ. N) GO TO 40
 CALL MMUL(X, A, N, N, N, EA)
 W = W*1/FLOAT(L)
 L = L + 1
 GO TO 20

40 CONTINUE
 CAN CHECK X:0 FOR ACCURACY
 J = N + 25
 DO 50 L = N, J
 DO 45 K = 1, N
 O(K) = (D(K + 1) - W*C(K)) * T1/FLOAT(L)
 45 E(K) = E(K) + O(K)
 50 W = O(1)
 II = 1
 DO 55 I = 1, NN, NDIM
 IL = I + NM1
 DO 50 J = I, IL
 55 EA(J) = E(1) * A(J)
 EA(I) = EA(I) + 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(I) = EA(I) + C2
 75 II = II + NDIM1
90 CONTINUE
85 IF (N .EQ. 0) RETURN
 DO 100 L = 1, N
 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)
SUBROUTINE PRNPLT

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/INOUT/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), CHAR(16), CHAR(17), CHAR(18),
* 1H5, 1H6, 1H7, 1H8, 1H9, 1HX, 1HE, 1HU, 1HY, 1HR, 1H/
DATA ISTAR, IPER, IDASH, IBLANK/*H*, 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(I)*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)
I5(A8S(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)=IGTA
105 IF(KSTEP—LINE) 110, 110, 125
110 DO 120 J=1, NVAR
RETURN
END
Appendix B LINEAR Computer Program

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)
KLXNE = KLXNE + LINEPR
GO TO 200
130 DO 135 I = 2, 51
135 KAXIS(I) = IBLANK
KAXIS(I) = IPER
GO TO 105
140 DO 150 I = 2, 51
150 KAXIS(I) = IRLANK
KAXIS(I) = IPER
IF (GRIDH * FQ.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 LINX = LINX + 1
IF (LINE = NHORZ - 1) 210, 210, 230
210 IF (KSTEP - LINE) 220, 75, 75
220 CONTINUE
230 RETURN
500 FORMAT(1PE10.2)
600 FORMAT(1PE13.2, 2X, 51A1)
700 FORMAT(15X, 51A1)
RETURN
END

C SUBROUTINE MSCALE

SUBROUTINE MSCALE(N1, N2, A, X, B)
DIMENSION A(I), B(I)
COMMON/MAIN1/NDIM, JEND = N2 * NDIM
DO 10 I = 1, N1
DO 10 J = 1, JEND, NDIM
10 B(IJ) = X * A(IJ)
RETURN
END

C SUBROUTINE VMUL

SUBROUTINE VMUL(N1, N2, X, Y, Z)
DIMENSION X(I), Y(I), Z(I)
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
SUBROUTINE VECTEO

SUBROUTINE VECTEO(N, X, Y)
DIMENSION X(1), Y(1)
DO 10 I = 1, N
10 Y(I) = X(I)
RETURN
END

SUBROUTINE RNDOFF

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 (DMIN .LT. 0) GO TO 50
N = T
DMIN = 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(/, A1X, J2HNUMBERS OUT OF RANGE FOR PLOTTING, /)
RETURN
END

SUBROUTINE ORS

SUBROUTINE ORS(N, IR, A, C, NOS)
DIMENSION A(1), C(1)
COMMON/INOU/KIN, KOUT
COMMON/YAIN1/NDIM
IF (NDIM .LT. N) WRITE(KOUT, 1000)
IF (NDIM .LT. IR) WRITE(KOUT, 1000)
IF (NDIM .LT. N) CALL EXIT
Appendix B LINEAR Computer Program

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 SUBROUTINE TRANS1

SUBROUTINE TRANS1(N,A,AT)
SETS AT = A TRANSPOSE 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(JI)
AT(IJ)=A(JI)
AT(JI)=TEMP
IJ=IJ+1
1 CONTINUE
RETURN
END

C SUBROUTINE TRANS3

SUBROUTINE TRANS3(NR,NC,A,AT)
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(JI)
AT(IJ)=A(JI)
AT(JI)=TEMP
1 CONTINUE
RETURN
END
Appendix B LINEAR Computer Program

IJJ=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,NR1
   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
   MULTIPLE COLUMNS 'M1' THROUGH 'IRLAST' BY MATRIX A
   K1=(M1-1)*NDIM+1
   K2=IR*NDIM+1
   ML = MINO(IRLAST+I-M1,N-IR)
33 CALL MMULIA(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
6 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
Appendix B LINEAR Computer Program

AND:
NC WILL BE REDUCED IF LINEARLY DEPENDANT COLUMNS OCCUR.
NNC=NNCO
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(W,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,1.0,A(II),W,A(II))
   NNC=NNC+1
   GO TO 5
50  RETURN

END

SUBROUTINE WADD

SUBROUTINE WADD(A,B,C,W,N)
DIMENSION A(I),B(I),C(I)
DO 10 I=1,N
10  C(I)=A(I)+W*B(I)
RETURN
END

SUBROUTINE VECTIO

SUBROUTINE VECTIO(N,X,IO)
DIMENSION X(I)
COMMON/NOU/KIN,KOUT
IF(IO.EQ.4) GO 1'D 40
IF(IO.EQ.3) GO TO 20
5 CONTINUE

INPUT
READ(KIN,1000)(X(I),I=1,N)
IF(IO.EQ.1)RETURN

OUTPUT
20 WRITE(KOUT,1701)(X(I),I=1,N)
GO TO 50
Appendix B  LINEAR Computer Program

C TITLF
C
40 READ(KIN,1002)
  WRITE(KOUT,1003)
  WRITE(KOUT,1002)
  WRITE(KOUT,1004)
  GO TO 5
50 CONTINUE
1000 FORMAT(9E10.0)
1001 FORMAT(1X,1P10E13.4)
1002 FORMAT(1X,79H )
1003 FORMAT(//)
1004 FORMAT(/)
  RETURN
  END

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 SUBROUTINE TRANS2
C
SUBROUTINE TRANS2(NR,NC,A,AT)
  AT=ATRANSPOSE, A IS NR X NC
  DIMENSION A(1),AT(1)
  NN=NC*NDIM
  II=1
  DO 6 I=1, NR
      JJ=II 
  6 DO 5 J=I,NN,NDIM
      AT(JJ)=A(J)
      JJ=JJ+1
  II=II+NDIM
Appendix B LINEAR Computer Program

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=L,N)
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
TEMP=TEMP+A(I+(J-1)*NDIM)*B(J)
10 TEMP=TEMP+C(I+(J-1)*NDIM)*D(J)
30 F(I)=TEMP
RETURN
END

C C SUBROUTINE MAT5
C
SUBROUTINE MAT5(N1,N2,N3,X,Y,Z)
DIMENSION X(1),Y(1),Z(1)
COMMON/MAIN1/NDIM
DO 2 I=1,N2
DO 2 J=1,N3
TM = 0.
II=(I-1)*NDIM
JJ=(J-1)*NDIM
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=XrYX Y=Y r 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
Q=II,NN1,NDIM
Z(I)=DOT(N2,X(II),Z(IJ))
5 J=J+I
I=I+1
JJ=I
DO 10 J=II,IJ
Z(J)=Z(JJ)
10 JJ=JJ+NDIM
RETURN
END

SUBROUTINE LINSIM

SUBROUTINE LINSIM(N,IR,A,C,XO,Q,T,NS,MP)
DIMENSION A(1),C(1),XO(1),Q(1)
COMMON/Main1/NDIM,DUM1(1)
COMMON/Main2/DUM2(1)
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,D,T,NS,MP,M,CONT)
RETURN
END

SUBROUTINE EQCOST

SUBROUTINE EQCOST(N,M,A,R,S,Q,AEQ,QEQ)
DIMENSION A(1),R(1),S(1),Q(1)
DIMENSION AEQ(1),QEQ(1)
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 EQUIATE(M,M,DUM2,R)
CALL GMINV(M,M,DUM2,DUM1,MR,3)
IF(MR,EQ,4) 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 ETQ(I+(J-1)*NDIM)=Q(I+(J-1)*NDIM)-DUM2(I+(J-1)*NDIM)
CALL MAT6(M,N,DUM1,S,DUM2)
CALL MMUL(B,DUM2,N,M,N,DUM1)
   DO 30 I=1,N
   DO 30 J=1,N
30 AETQ(I+(J-1)*NDIM)=A(I+(J-1)*NDIM)-DUM1(I+(J-1)*NDIM)
100 FORMAT(/,25H ERROR MATRIX R HAS RANK ,12,12H LESS THAN M)
RETURN
END

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
FILE INTODE1 FORTRAN A1
PROGRAM TO INTEGRATE GENERAL ODE'S

THIS VERSION WAS USED ESPECIALLY TO GENERATE TRAJECTORIES FOR
THE LANGLEY/NFWSOM CONTRACT SUMMER 1980

DIMENSION XX(120), YY(120), NPTS(10), XYL(4)
DIMENSION NXPL(10), NXPR(10), XO(12), XI(12)
DIMENSION X(12), F(10), XNAMES(10), XLBLS(5,10)
DIMENSION CNAME(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/
DATA NSTEPS, NPRINT/3*1/
COMMON/CASE/NCASE
COMMON/NNOUT/NOUT
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 1 NXPL, IPLOT, TPL DT, NPRINT, ICONT, NXPL, WIDTH, HEIGHT, TICKL,
1 NCHARS, NLINES, XNAMES, XLBLS, XPLBL, YPLBL, YNC, TITL, NTTL

INPUT VARIABLES IN NAMELIST:

XII(1) = INITIAL VALUES FOR STATE VARIABLES
TMAX = MAXIMUM (FINAL) INTEGRATION TIME
IPRINT = INDEX TO DETERMINE THE AMOUNT OF PRINTOUT
    1, NORMAL OUTPUT
    0, MINIMUM OUTPUT
    2, EXTRA OUTPUT
TPRINT = TIME INCREMENT FOR TRAJECTORY PRINTOUT IN TABLE
DT = INTEGRATION STEP SIZE, SHOULD BE A WHOLE DIVISOR
    OF TPRINT AND TPLDT
A(1,1) = NX BY NX SYSTEM MATRIX
B(1,1) = NX BY NU CONTROL MATRIX
XK(1,1) = NU BY NX FEEDBACK MATRIX
NX = NO. OF STATE VARIABLES
NU = NO. OF CONTROL VARIABLES
NINT = NUMBER OF VARIABLES TO BE INTEGRATED IN RKINT
NTOT = TOTAL NUMBER OF STATE, CONTROL AND AUXILLARY VARIABLES
NXPR(1) = ARRAY OF INDICES OF VARIABLES IN TRAJECTORY TABLE
IPLOT = INDEX THAT CONTROLS THE GENERATION OF PLOTS
    0, NO PLOTTER OUTPUT
    1, GENERATE Printer Plot ONLY
    2, GENERATE BOTH PRINTER PLOT AND VERSATEC PLOT
TPLDT = TIME INCREMENT FOR STORING POINTS FOR PLOTTER
NDEV = UNIT NUMBER OF OUTPUT DEVICE, NOMINALLY 6
ICONT = INDEX TO CONTINUE OR STOP MULTIPLE CASE RUNS
    0, STOP AFTER THIS CASE
    1, CONTINUE TO NEXT CASE
NXPL(1) = ARRAY OF INDICES OF VARIABLES TO BE PLOTTED
WIDTH = WIDTH OF VERSATEC PLOT IN INCHES
HEIGHT = HEIGHT OF VERSATEC PLOT IN INCHES
TICKL = LENGTH OF TICK MARKS AND HEIGHT OF CHARACTERS (INCHES)
    IF NEGATIVE, PLACE TICK MARKS ON INWARD SIDE OF AXIS
Appendix C INTOE Computer Program

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 DETERMINE NO. OF PRINT COLUMNS
C DO 2 I=1,10
  IF(NXPR(I).LE.0)GO TO 3
  CONTINUE
2 NPRTOT = I-1
C DETERMINE NO. OF CURVES ON PLOT
C DO 4 I = 1,10
  IF(NXPL(I).LE.0)GO TO 5
  CONTINUE
4 NCURV = I-1
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
  TPRINT = AMAX1(TPRINT,DT)
  NPRINT = (TMAX + .00001)/TPRINT
  NPNTS = NPRINT+1
C SET UP DATA FOR PLOTS
C IF(NCURV.NE.0)NPNTS = MINO(NPNTS,(NPMAX-NCURV)/NCURV)
  DO 7 I=1,NCURV
    VCINOX(I) = 'VPNTS*(FLOAT(I)/FLOAT(NCURV+I))
    NCINOX(I) = NPNTS/15.+L0*(I-1)
    THIS LINE PUTS THE LABLES ON A PLOT CURVES CLOSE TO THE LEFT MARGIN,
    K=NXPL(I)
C
Appendix C INTODE Computer Program

C NAMES(I)=XNAMES(K)
7 NPTS(I) = NPNTS
C C PRINT BANNER
C WRITE(NODEV,105)NCASE
105 FORMAT(1HI,'********** PROGRAM INTODE CASE',I3,/
CALL DATIME(I4,ID,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,2OA4,
1 // TMAX,TPRINT,TPLOT,DT =',
1 4F12.5,/' IPRINT,IPLOT,NCURV,NPNTS =',415)
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,OT,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.I)CALL P PLOT(XX,YY,NCURV*IP,XYL,NCHARS,NLINES)
IF(IPLOT.LE.1)GO TO 30
IF(I START.GE.1)CALL PLOTS(0,0,50)
IF(I START.LE.0)CALL PLO T(WIDTH+2.,0.,-3)
I START = 0
CALL APL OT(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(C.,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
    XTEMP(J) = X(J) + T*DX(J,I)
  END DO
  CALL XDOT(XTEMP,TIME,DX(1,I+1))
  DO 30 I=1,NX
    X(I) = X(I) + DT*(DX(I,1)+2.*DX(I,2)+2.*DX(I,3)+DX(I,4))/6.
    TIME = TIMEO + DT
  END DO
RETURN
END

SUBROUTINE XYSTOR(XX,YY,IP,TIME,X,NPTS,NPMAX,NCURV,NXPL)
DIMENSION XX(1),YY(1),X(1),WXPL(1)
DO 10 I=1, NCURV
  II = IP + (I-1)*NPTS
  IF(II.GT.NPMAX)RETURN
  XX(II) = TIME
  JJ = NXPL(I)
  YY(II) = XX(J)
  IP = IP + 1.
RETURN
END

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,PRACTICE HEADINGS AND COLUMN HEADINGS ONLY
C	=1, PRACTICE 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 CHARACTER LABELS OF VARIABLES
C	XLBLS(5, ) = ARRAYS OF 20-CHARACTER DEFINITIONS OF VARIABLES
C	NODEV=OUTPUT UNIT NO
C	NCHARS=WIDTH OF PAPER(NO OF CHARACTERS)

C	PRINT DEFINITIONS AND HEADINGS
C
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(/,' VAR',' LABEL',' DEFINITION',
      /(' X(','12X') = ',A4,',',5A4))
DO 10 I = 1,NPR
  J = NXPR(I)
END

ORIGINAL PAGE IS OF POOR QUALITY
Appendix C INTODE Computer Program

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
    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)))
         XMAX = AMAX1(XMAX,ABS(XO(I)))
    30   CONTINUE
    NPM = MINO(NPR,NCOL)
    IF(XMIN.LT.10.E-01.OR.XMAX.GT.99999.999) GOTO 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=1,NPM)
102  FORMAT(1X,12F10.3)
    RETURN
C
C PRINT IN E10.3 FORMAT
C
        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=1,NPM)
104  FORMAT(1X,1P12E10.3)
    RETURN

C
C SUBROUTINE VITAL(X,XI,IPRINT)
DIMENSION X(1),XI(1),DUM(4,4)
DIMENSION D(5,5),D2(5,5),D3(5,6),D4(5),ER(5),EI(5)
COMPLEX EIG(5),EVEC(5,5),CNORM
Appendix C INTODE Computer Program

COMMON/NCUT/NODEV
COMMON/MAT/A(4,4),B(4,3),XK(3,4),NX,NU,NINT,NTOT

C INITIALIZE TIME AND STATE
C
DO 2 I = 1,NINT
  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 COMPUTE CLOSED-LOOP MATRIX
C
DO 10 I=1,NX
  DO 10 J=1,NX
    TEMP = 0.
    DO 5 K=1,NU
      TEMP = TEMP + B(I,K)*XK(K,J)
    5 CONTINUE
    DO 10 J=1,NX
      TEMP = TEMP + XK(I,J)**2
    10 CONTINUE
    TEMP = SQRT(TEMP)
    WRITE(NODEV,102)TEMP
102 FORMAT(/,10X,'|K| = ',G15.5)

C COMPUTE CLOSED-LOOP EIGFNSOLUTIONS
C
IJOB = 1
CALL FIGRF(DUM,NX,4,IJOB,EIG,EVEC,5,32,IER)
D7 24 I=1,NX
  FI(I) = REAL(EIG(I))
  FI(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) = RFAL(EVEC(J,I)*CNORM)
    IF(EI(I).GT.3.)DI(J,I+1) = AIMAG(EVEC(J,I)*CNORM)
  22 CONTINUE
  CONTINUE
DO 30 I = 1,NX
  TEMP = 0.
  CONTINUE
30 CONTINUE
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',
13,' /',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(IX,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,I
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(AMINU(ABS(ARG),1))
38 WRITE(6,104)(I,J,D3(I,J),J=1,I)
104 FORMAT(/,5(' ANG',I1,',',I1,' =',F7.2))
RETURN
END

SUBROUTINE XDOT(X,TIME,F)
DIMENSION X(1),F(1)
COMMUN/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.
3 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.
1 SUM=SUM+A(I,J)*X(J)
DO 5 K=1,3
5 SUM=SUM+B(I,K)*X(NINT+K)
Appendix C INTODE Computer Program

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
SUM=SUM+XK(I,J)*X(J)
1 X(NINT+1)=SUM
RETURN
END

C

FILE WRTMAT FORTRAN A1
C
5/8/79
C
FILE OF UTILITY SUBROUTINES TO SUPPORT VIBE FORTRAN A1
C
WRTMAT - GENERAL MATRIX OUTPUT SUBROUTINE
C

SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
DIMENSION A(IA,1)
COMMON/NOUT/NPRINT
REAL*8 ANAME
WRITE(NPRINT,100)(ANAME,N,M)
100 FORMAT(5X,' MATRIX (',A9.3X,'(*',I3,' ROWS X',I3,' COLS')')
IF(M.LE.0)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

TRANSP - TRANSPOSES A MATRIX
C

SUBROUTINE TRANSP(A,N1,N2,NA)
DIMENSION A(NA,1)
COMMON/NOUT/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
MULT - MATRIX MULTIPLICATION

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/NOUT/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
        TEMP = TEMP + DBLE(A(I,K))*DBLE(B(K,J))
  10  C(I,J) = TEMP
  RETURN
  90 WRITE(NOUT,100)N,NA,NC,L,NB
  100 FORMAT(*** ERROR IN MULT *** N,NA,NC,L,NB=',3I5)
  RETURN
END

C
MSHIFT - TRANSFER VECTORS AND MATRICES

SUBROUTINE MSHIFT(A,B,N,M,NA,NB)
  DIMENSION A(NA,1),B(NB,1)
  COMMON/NOUT/NOUT
  IF(N.GT.MINO(NA,NB))GO TO 90
  DO 10 I=1,N
    DO 10 J=1,M
      R(I,J) = A(I,J)
  10  RETURN
  90 WRITE(NOUT,100)N,NA,NB
  100 FORMAT(*** ERROR IN MSHIFT *** N,NA,NB=',3I5)
  RETURN
END

C
MATADD - MATRIX ADDITION

SUBROUTINE MATADD(A,B,C,N,M,NA,NB,NC)
  DIMENSION A(NA,1),B(NB,1),C(NC,1)
  COMMON/NOUT/NOUT
  IF(N.GT.MINO(NA,NB,NC))GO TO 90
  DO 10 I=1,N
    DO 10 J=1,M
      C(I,J) = A(I,J) + B(I,J)
  10  RETURN
  90 WRITE(NOUT,100)N,NA,NB,NC
  100 FORMAT(*** ERROR IN MATADD *** N,NA,NB,NC=',3I5)
  RETURN
END

C
Appendix C  INTODE Computer Program

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/NOUT/NOUT
IF(N.GT.MIN0(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, MA, NB, NC
100 FORMAT(' *** ERROR IN MATSUB *** N, NA, NB, NC=', 4I5)
RETURN
END

C SWAP - INTERCHANGES TWO VARIABLES
C
SUBROUTINE SWAP(A, B)
C = A
A = B
B = C
RETURN
END

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 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 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
Appendix C  INTOE Computer Program

END

C
C ANORM — CALCULATES THE RSS NORM OF A MATRIX
C
FUNCTION ANORM(A,N1,N2,NA)
DIMENSION A(NA,1)
COMMON/NOUT/OUT
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,N
100 FORMAT(*** ERROR IN ANORM ***,N,NA =',2I5)
RETURN
END

C********** SUBROUTINE PPLOT **********
C
SUBROUTINE PPLOT(X,Y,NPOINT,XYLIMS,NCHARS,NLINES)
C
PPLOT GENERATES A PRINTER PLOT UP TO 130 CHARACTERS WIDE
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(1) = 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 IBLANK,ICROSS/* "/,",+",/"
COMMON/NOUT/NODEV
WRITE(NODEV,100)NPOINT,NCHARS,NLINES
100 FORMAT(/' PPLOT CALLED ... TOTAL NUMBER OF POINTS =','I6,
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
IX = -XST/DX*NCHARS + 1.000001
IXAXIS = 0
IF((IX .GE. 1 .AND. IX .LE. NCHARS)) IXAXIS = 1
CALL SCALE(YMIN, YMAX, YST, YDEL, NYP)
YFIN = YST + YDEL*NYP
DY = YFIN - YST
IY = YFIN/DY*NLINES + 1.000001
IYAXIS = 0
IF((IY .GE. 1 .AND. IY .LE. NLINES)) IYAXIS = 1
WRITE(NODEV, 105)XST, XFIN, YST, YFIN, IX, IY
105 FORMAT(2X, 'XST, XFIN, YST, YFIN =', 1P4E12.4, I0P2I6)
INDEX FOR X ANY Y ZERO REF AXES =*, OP2I6)
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, IY) = IHOR
20 IFLD(I, NLINES) = IHOR
DO 30 I = 1, NLINES
IF(IXAXIS .EQ. 1) IFLD(IX, 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 = MIN0(IX, NCHARS)
45 YGRID(J) = YST + (J-1)*YDEL
IY = ((J-1)*YDEL/DY)*NLINES + 1.00001
IY = MIN0(IY, NLINES)
45 IFLD(I, IY) = ICROSS
WRITE(NODEV, 101)(XGRID(I), I=1, NX)
101 FORMAT(2X, 'X-GRID =', 7F10.3, 2X, 9X, 7F10.3)
WRITE(NODEV, 102)(YGRID(I), I=1, NY)
102 FORMAT(2X, 'Y-GRID =', 7F10.3, 2X, 9X, 7F10.3)
DO 50 I = 1, NPOINT
IX = ((XI - XST)/DX)*NCHARS + 1
IY = ((YFIN - Y(I))/DY)*NLINES + 1
50 IF(I .LE. 50) WRITE(NODEV, 107)X(I), Y(I), IX, IY
107 FORMAT(2X, 'X(I), Y(I) =', 1P2E12.4, 2X, IX, IY =*, OP2I6)
IX = MAX0(MIN0(IX, NCHARS), 1)
IY = MAX0(MIN0(IY, NLINES), 1)
50 IFLD(I, IY) = ISTAR
WRITE(NODEV, 103)
103 FORMAT(141)
DO 60 IY = 1, NLINES
60 WRITE(NODEV, 104)(IFLD(I, IY), I=1, NCHARS)
Appendix C INTOE Computer Program

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********** SUBROUTINE MINMAX **********
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********** SUBROUTINE SCALE **********
SUBROUTINE SCALE(XMIN,XMAX,START1,DEL1,NPTS1)
INTX(X) = IFIX(X-.5+SIGN(.5,X))
IF(XMAX.LE.XMIN)GO TO 90
X=IZ = P
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.4.8.8.XNORM.LE.8.)XMOD = 1.
IF(XNORM.GT.8.)XMOD = 2.
IEXP = 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 WRITF(6,100)XMIN,XMAX
100 FORMAT(//' ERROR IN SCALE XMIN,XMAX = ',2E12.4)
RETURN
END
C********** SUBROUTINE A PLOT VERSION 1 11/26/80 **********
SUBROUTINE APLOT(X,Y,NPT,NPLOT,XYLIMS,WIDTH,HEIGHT,TICKL,
   1 NCASF,XLBL,NXC,YLBL,NYC,TITL,NTTL,CNAMES,NCINDEX)

NOTE THE X( ), Y( ) ARRAYS ARE UNCHANGED IN THIS SUBROUTINE

APLOT INPUTS:

X( ) = ARRAY OF X-COORDINATES OF POINTS
Y( ) = ARRAY OF Y-COORDINATES OF POINTS
NPT( ) = NO. OF POINTS(X,Y PAIRS) IN EACH CURVE
NPLOT = NUMBER OF CURVES
XYLIMS(4) = LIMITS OF X AND Y AXES IN THE ORDER X_MIN, X_MAX, Y_MIN, Y_MAX
IF XYLIMS( ) = 0, SCALING IS AUTOMATIC
WIDTH = WIDTH OF PLOT IN INCHES
HEIGHT = HEIGHT OF PLOT IN INCHES
TICKL = LENGTH OF TICK MARKS ON AXES IN INCHES AND HEIGHT OF SCALE NOS
NCASE = A NO. PRINTED AT THE UPPER RIGHT HAND CORNER OF PLOT FOR
INDEXING PURPOSES.
XLBL( ) = ALPHANUMERIC STRING FOR X-AXIS LABEL
NXC = NUMBER OF CHARACTERS IN X AXIS LABEL
YLBL( ), NYC = LIKewise FOR Y AXIS LABEL
TITL( ) = ALPHANUMERIC STRING FOR TITLE
NTTL = NUMBER OF CHARACTERS IN TITLE
CNAMES( ) = ARRAY OF FOUR-CHARACTER IDENTIFIERS FOR THE NPLOT CURVES
NCINDEX( ) = INTEGER (GE.1. AND LE. NPT( )) INDICATING LOCATION
OF NAMES IDENTIFIERS FOR EACH CURVE

COMMON/NOUT/NODEV
DATA EPS/1.E-9/
WRITE(NODEV,100)NPLOT,NCASE,(NPT(I),I=1,NPLOT)
100 FORMAT(/" APLOT CALLED 	 NPLOT = ",15,3X,",NCASE = ",13,3X,"NPT( ) = ",
6I14,/,10X,10I4))

C IF(NPLOT.LE.0)RETURN
TICK = ABS(TICKL)
H1 = 12*TICK
H2 = H1 + WIDTH
V1 = 9*TICK
V2 = V1 + HEIGHT

C FIND TOTAL NUMBER OF POINTS ON PLOT
NPONT = 0
20 DO 2 I=1,NPLOT
   2 NPONT = NPONT + NPT(I)
SCALE X-AXES
Appendix C INTODE Computer Program

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
NX = XFIN - XST
XF = (H2-H1)/DX
WRITE(NODEV,101)XMIN,XMAX,XST,XDEL,XFIN,NXP
101 FORMAT(/$ XMIN, XMAX = *, 1PE14.7, 5X, 'XSTART, XDEL, XFINISH = *', 1 3E10.3, 5X, 'NXP = *, I4)
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
NY = YFIN - YST
YF = (V2-V1)/DY
WRITE(NODEV,102)YMIN,YMAX,YST,YDEL,YFIN,NYP
102 FORMAT(/$ YMIN, YMAX = *, 1PE14.7, 5X, 'YSTART, YDEL, YFINISH = *', 1 3E10.3, 5X, 'NYP = *, I4)
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
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
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)
KP = (H1+H2)/2. - .5*NXC*WD
IF(NXC.GT.0)CALL SYMBOL(XP,V1-3.*HT,HT,XL3L,ANG,NXC)
YP = (V1+V2)/2. - .5*NYC*WD
IF(NYC.GT.0)CALL SYMBOL(H1+11.*HT,YP,HT,YLBL,ANG90,NYC)

ADD CASE NUMBER (IF NONZERO) AND DATE

NDG = 0
IF(NCASE.GT.0)CALL NUMBER(H2+HT,V2,HT,FLOAT(NCASE),ANG,NDG)
CALL DATIMP(H1,V2+2*HT,HT,*APLOT *)

ADD X-AXIS TICK MARKS AND SCALE

N = NXP + 1
YY = V1 - 4.*TICK
DO 22 I=1,N
XA = HI + (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)

ADD Y-AXIS TICK MARKS AND SCALE

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)

ADD LABEL FOR EACH CURVE

K = 0
DO 30 I=1,NPLOT
IF(NCIND(I).LE.0)GO TO 32
JPT = MAX0(MIN0(NCIND(I),NPT(I)),1)
XX = A4LIN1(A4AX1(H1,(X(K+JPT)-XST)*XF+H1),H2)
YY = A4LIN1(A4AX1(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+3.5*HT,YY+4.*HT,HT,CNAMES(I),ANG,4)
32 N = NPT(I) - 1

PLOT EACH CURVE

K = K + 1
XX = A4LIN1(A4AX1(H1,(X(K)-"ST)*XF+H1),H2)
Appendix C  INTOE Computer Program

YY = AMIN1(AMAX1(VI,(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.9.)XMOD = 1.
IF(XNORM.GT.9.)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
  XMIN = AMAX1(XMIN,X(I))
10 XMIN = AMIN1(XMIN,X(I))
RETURN
END

C************* SUBROUTINE DATIME *************
C SUBROUTINE DATIME(IMO,IDAY,IYR,IHOURS,IHOURP,PNAME,NOFDE)
REAL*8 PNAME
DATA AM/* AM*/, PM*/ PM*/
CALL DATE(IMO, IDAY, IYR)
CALL TIME(ITIME)
XHOURS = FLOAT(ITIME)/10000.
AMPM = AM
IF(XHOURS.GE.12.)AMPM = PM
IF(XHOURS.GE.13.)XHOURS = XHOURS - 12.
IHOURS = XHOURS
XMIN = (XHOURS - IHOURS)*60.
I’MIN = XMIN
IF(NODEV.GT.0)WRITE(NODEV,100)PNAME, IMO, IDAY, IYR, IHOURS, IMIN, AMPM
100 FORMAT(’TIME IN ’',A8,’ IS ’,A2,’/’ ,A2,’/’,A2,’/’,A2,5X,12,’;’,
1 12,3X,A4)
RETURN
END

C********** SUBROUTINE DATIMP **********

SUBROUTINE DATIMP(XP, YP, HT, PNAME)
REAL*8 PNAME
COMMON/NHOUT/NODEV
ANG = 0.
WD = .9*HT
CALL DATE(IMO, IDAY, IYR, IHOURS, IMIN, AMPM, PNAME, NODEV)
XHOURS = IHOURS
XMIN = I’MIN
CALL SYMBOL(XP, YP, HT, IMO, ANG, 2)
CALL SYMBOL(XP+2.*WD, YP, HT, LH/ANG, 1)
CALL SYMBOL(XP+3.*WD, YP, HT, IDAY, ANG, 2)
CALL SYMBOL(XP+5.*WD, YP, HT, IH/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, IH/ANG, 1)
CALL NUMBER(XP+15.*WD, YP, HT, XMIN, ANG, -1)
CALL SYMBOL(XP+18.*WD, YP, HT, AMPM, ANG, 4)
RETURN
END

FILE INTODE1 INPUT

INPUT FOR MONTGOMERY’S AIRCRAFT LATERAL SIMULATION
JUL 29, 1980

&INTODE
XI = 0.,1.,.1,6*0.,
TMAX = 5.,
IPRINT = 2,
NODEV = 6,
DT = .075,
TPRINT = .05,
NXPR = 1,2,3,4,5,6,7,8,2*0,
A=-3.67994E-1,1.96959,0.,2.59319E-1,2.4209F-2,2.61375E1,
-3.2279E-2,2.67949E-1,-1.10395E-1,-9.65926E-1,2.61375E1,0.,4.401072E-2
R=-7.57133,0.,1.96959,0.,2.56549,0.,-2.33343E0,0.,
\[ \begin{align*}
XK &= 1.494116 \times 10^{-01} + 4.1928029 \times 10^{-01} + 1.0535228 \times 10^{-01} + 2.0296771 \times 10^{-02} + 1.3 \times 10^{00} + 3.6583920 \times 10^{-00} + 2.3151433 \times 10^{-01} + 1.47352 \times 10^{-02} + 4.0296771 \times 10^{-02} + 2.8092501 \times 10^{-01} + 4.892501 \times 10^{-02} + 3.2358761 \times 10^{-01} + 1.5113544 \times 10^{-00} + 4.4778833 \times 10^{00} + 3.2358761 \times 10^{02} + 1.5113544 \times 10^{00}, \\
XK &= 1.702794 \times 10^{01} - 6.0164034 \times 10^{-01} + 2.6196051 \times 10^{-01} + 2.7013290 \times 10^{-02} + 2.9 \times 10^{00} + 4.0537797 \times 10^{-00} + 1.856604 \times 10^{-00} + 4.892501 \times 10^{-01} + 4.912501 \times 10^{-00}, \\
XK &= 1.2956323 \times 10^{00} - 3.1967741 \times 10^{-01} + 8.50401 \times 10^{-02} + 1.563799 \times 10^{01} + 1.9 \times 10^{00} + 4.912501 \times 10^{-01} + 4.892501 \times 10^{-00}, \\
XK &= 1.64359 - 3.31017 + 0.5764 + 0.244 + 2.07949 + 3.5064 + 0.741 + 0.432244, \\
XK &= 1.6368 - 3.5991 + 2.27027 - 3.222 + 2.05683 + 3.41589 + 0.743469 - 0.520 \\
XK &= 1.6368 - 3.5991 + 2.27027 - 3.222 + 2.05683 + 3.41589 + 0.743469 - 0.520 \\
\end{align*} \]
Appendix C INTODE Computer Program

```
TITL(9)=4H(LQR),
TITL(10)=4H(RH),
TITL(11)=4H0=,
TITL(12)=4H100=,
NTTL=48,
XK=-.3306,-.1784,-.10918,-.03913,-.09998,.48297,2.6222,-1.4068,
END

INTODE
TITL(12)=4H1=,
XK=1.2588,-.17087,1.0046,.0753,-.30636,1.0189,5.9393,-1.734,
END

INTODE
ILOCAL=0,
DT=.01,
TITL(12)=4H0.04,
XK=5.17,-.059375,.48719,1.1309,1.065,.065,4.72623,-2.5309,
END

C FILE INTODE INPUT
C INPUT FOR HALL'S AIRCRAFT LATERAL SIMULATION
C JUL 31, 1980

INTODE
XI = 0.0, 1.0, 1.6*0.0,
TMAX = 5.0,
IPRINT = 2,
NODEV = 6,
DT = .025,
TPRINT = .1,
N:PR = 1,2,3,4,5,6,7,8,9,0,
A=-3.18,1.06,.022,3.064,.064,.63,0.0,-.27,.998,-10.6,0,.45,
B=-14.4,.301,1.5,0.0,-2.59,.037,.290,.046,0.0,
XK =.39712631E-02,-.43598451E-03,.19987654E-02,.1202696E-02,-.959E-01,
2.5593357E+00,2.1611822E-01,-.3611261E-03,.47516279E+00,-.94E-01,
-.5477920E-01,-.51410699E-01,
NX=4,
NU=3,
IPLON = 2,
NXPL = 2,4,7,9,5*0
NCHARS = 80, NLLINES = 48,
NCHARS = 130, NLLINES = 62,
XNAMES = 4HRORA, 4HBANK, 4HYAWR, 4HYAW, 4HINST, 4HINU2, 4HAILE, 4HRYAWC, 0,
XLBLS(1,1) = 4HROLL, 4HRAT, 4H (R, 4HAD/S, 4HEC),
XLBLS(1,2) = 4HBN, 4HANG, 4HLE (, 4HRAD),
XLBLS(1,3) = 4HYAW, 4HRATE, 4H (RA, 4HD/SE, 4HC),
XLBLS(1,4) = 4HYAW, 4HANGL, 4HE (R, 4HAD),
XLBLS(1,5) = 4HINT, 4HGRAL, 4HXT*X,
XLBLS(1,6) = 4HINT, 4HGRAL, 4HUT*U,
XLBLS(1,7) = 4HAILE, 4HRON, 4H(RAO, 4H),
XLBLS(1,8) = 4HRUD, 4HER (, 4HRAD),
XLBLS(1,9) = 4HYAW, 4HCONT, 4HROL, 4H(RAD, 4H),
NINT = 9,
NINT = 6,
X0LBC = 4HTIME, 4H (SE, 4HC), 7*0.,
```
Appendix C  INTODE Computer Program

NXC = 10,
YPLFL = 4HSTAT, 4HE, 8*0,
NYC = 5,
    TITL=4HHALL, 4H AIR, 4HCRAF, 4HT LA, 4HTERA, 4HL DY, 4HNAMI, 4HCS(M, 4WIN K, 4HT) 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, -.030165, .02299, 4.14655, .014765, -.3113,
&END
&INTODE
    TITL(12)=4H0.25 ,
    XK= 3.5695368E-01, -.9497081E-02, 1.2787379E-02, 1.4500313E+00, 2.9E-01, 
     -.7618792E-01, -.1 .2922382E+00, 2.6900148E+00, 3.4010111E+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, 1.37833, -.933, 2.535, 2.175, -.02644, 5.530 
     -.9673, -.6965, -.525889,
&END
&INTODE
    TITL(12)=4H 0.5 ,
    XK= .03819, -.0016245, -.012548, 4.133, .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, 1.12387, -.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= .34513, .00574, .003798, .0916, .0222, .011679, .056979, .09242, 
     .03633, -.0794, -.0208, -.008925,
&END
&INTODE
    TITL(11)=4H 1.0 ,
    DT=0.01,
    TPLT=0.02,
    XK= .9653, -.067577, .007833, .79586, -.058595, 0.016246, 1.175, .93939, 
     .35148, -.55619, -.27992, -.1,

124
Appendix C  INTODE Computer Program

&END
&INTODE
  ICNT=0,
  TITL(11)=400.04,
  XK=4.8341,-.4264,.02787,4.98,-.5015,.01787,.41809,4.748,1.7639,
-73614,-3.153,-.86026,
&END
FILE PERTB FORTRAN A1

PROGRAM TO TEST ROBUSTNESS OF LINEAR CONTROL SYSTEMS

DOUBLE PRECISION DSEED
DIMENSION FVR1(4), EV1(4), EVEC1(4,4), EVR2(4), EV2(4), EVEC2(4,4)
DIMENSION XX(4004), YY(4004), XYL(4), NPTS(1)
DIMENSION CNAMES(1), XLBL(LO), YLBL(10), TITL(20), NCINDEX(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/NOUT/NOUT
NAMELIST/PFR,P/N,M, ICONT, A, B, XK, P, NSAMP, IPRINT, IPRINT,
IXLBL, NXC, YLBL, NYC, TITL, NTTL, CNAMES, NCINDEX

INPUT VARIABLES IN THE NAMELIST

N = NO. OF STATE VARIABLES
M = NO. OF CONTROL VARIABLES
ICONT = INDEX TO CONTINUE OR STOP MULTIPLE CASE RUNS
0 , STOP AFTER THIS CASE
1 , CONTINUE TO NEXT CASE
A = N BY N SYSTEM MATRIX
B = N BY M CONTROL MATRIX
XK = M BY N FEEDBACK MATRIX
P = FRACTION GIVING THE MAXIMUM PERTURBATION
NSAMP = NO. OF SAMPLES
IPRINT = INDEX TO DETERMINE THE AMOUNT OF PRINTOUT
IPLOT = INDEX THAT CONTROLS THE GENERATION OF PLOTS
XLBL = ARRAY CONTAINING THE X—AXIS LABEL FOR PLOTS
NXC = NO. OF CHARACTERS (LETTERS) IN X—AXIS LABEL
YLBL = ARRAY CONTAINING THE Y—AXIS LABEL FOR PLOTS
NYC = NO. OF CHARACTERS IN Y—AXIS LABEL
TITL = ARRAY CONTAINING A LABEL TO APPEAR IN PLOTS
NTTL = NO. OF CHARACTERS IN TITLE

DATA NCASE/0/, ISTART/1/
DATA XYL/-10.,0.,-3.,3./
DATA WIDTH, HEIGHT, TICKL/5.5, 3.3, .06/
READ(7,PERT)
DSEED=11DO
NCASE=NCASE+1
NOUT=6
NPTS(1)=NSAMP*N+N
NCURV=1

PRINT BANNER

WRITE(6,100) NCASE
FORMAT(1H1,'****PROGRAM PERTURB, CASE',I5,'****')
WRITE(6,PERT)
CALL DATIIE(IM,ID,IY,IM,AP,'PERTB ',6)
WRITE(6,101) TITL
FORMAT(/1X,20A4//)

PRINT MATRICES A, B, K AND A+B*K

126
Appendix D PERTB Computer Program

CALL WRTMAT(A,N,N,N,A= )
CALL WRTMAT(B,N,N,N,B= )
CALL WRTMAT(XK,M,N,N,K= )
CALL MATM(A,B,XK,CA,N,M)
I=1
CALL WRTMAT(CA,N,N,N,'A+B*K' )

CALCULATE EIGENVALUES AND EIGENVECTORS
CALL GENE(CA,N, EVR1, EVII, EVPC1, I, IPRINT)
IP=1
IF(IPLOT .GT. 0) CALL XYSTO(XX, YY, EVR1, EVII, N, IP)

START MONTE CARLO RUNS
DO 10 I=1, NSAMP
CALL PERTB
CALL MATM(?A, PB, XK, CA, N, M)
IF( (I-1)/IPRINT*IPRINT .GT. 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,'PERT A*B*K' )

CALL GENE(CA,N, EVR2, EVII, EVPC2, I, IPRINT)
IF(IPLOT .GT. 0) CALL XYSTO(XX, YY, EVR2, EVII, N, IP)

CALCULATE AND PRINT STATISTICS ON REMAX, REMIN AND RTOMAX
CALL STATS(EVR1, EVII, EVR2, EVII, N, I, NSAMP, IPRINT)
CONTINUE
IF(I-PRINT .LT. 1) GO TO 99
IF(I-START .GE. 1) CALL PLOTS(0,0,50)
IF(I-START .LT. 0) CALL PLOT(WIDTH+2,0,-3)
ISTART=0

DO PLOTTING
CALL APLOT(XX, YY, NPTS, NCURV, XYL, WIDTH, HEIGHT, TICKL, NCASE,
1  XLRL, NXC, YLRL, NCYT, NTTL, CNAMES, NCINDEX)

GO TO NEXT CASE IF ICONT IS GREATER THAN ZERO

99 IF(ICONT .GE. 1) GO TO 999
IF(I-PRINT .GT. 0) CALL PLOT(0,0,999)
STOP
END

MATM - COMPUTES CA = A + B * XK
WHERE A, B AND XK ARE GIVEN MATRICES

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
77 DO 10 J=1,N
TEMP=0.

127
Appendix D PERTB Computer Program

DO 5 K=1,M
   TEMP=TEMP*B(I,K)*XK(K,J)
10  CA(I,J)=TEMP+A(I,J)
RETURN
END

C

PERTB - RANDOMLY PERTURBS THE ELEMENTS OF MATRICES A AND B

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/ISFED/DSEED,RAND(28),P
NN=N*(N+1)
CALL GGUBS(DSEED,NN,RAND)
DO 6 I=1,NN
   RAND(I)=(-1+2*RAND(I))
6  NNN=O
DO 1 I=1,N
   DO 2 J=1,N
      NNN=NNN+1
      PA(I,J)=A(I,J)
5   IF(A(I,J) .EQ. 0 .OR. A(I,J) .EQ. 1.) GO TO 7
      PA(I,J)=A(I,J)*(1.+RAND(NNN)*P)
2  CONTINUE
   CONTINUE
1  CONTINUE
RETURN
END

C

STATS - COMPUTES AND PRINTS STATISTICS ON REMAX, REMIN AND RTOMAX

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 CREX0(1000),CREM0(1000),CRTX0(1000)
DATA RMAXM,RMINM,RTMXM,SRXA,SRMA,SRTXD/S*0.*/
DATA RMAXM,RMINM,RTMXM,SRXN,SRNM,SRTXD/S*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
   RT(I)=EVI1(I)/EVR1(I)
3  RETURN
END
Appendix D PERTB Computer Program

```
RTMAX1=AMAX1(RT(1),RT(2),RT(3),RT(4))
CALL MINMAX(EVR2,N,REMIN2,REMAX2)
DO 2 I=1,N
2 RT(I)=EVI2(I)/EVR2(I)
RTMAX2=AMAX1(RT(I),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,'REMIN=',F10.4,'/',/2X,'RTOMAX=',F10.4,'/',/2X,'STD DEVN=',F10.4)
123 FORMAT(2X,'UNPERTURBED VALUES ARE ',/2X,'REMA=',F10.4,'/',/2X,'REMIN=',F10.4,'
CRTX(IS)=(RTMAX2-RTMAX1)/RTMAX1*100.
CRTX(IS)=0.
129 CALL MINMAX(CREX,NSAMP,CREXN,CREXX)
```

129
CALL MINMAX(CRTXArNSAMP,CRTXNA,CRTXXA)
WRITE(6,120)
120 FORMAT('/25X,'STATISTICS FROM THE MONTE CARLO SIMULATIONS')
WRITE(6,123) ANAME1,CREXX,CREXN,RMAXX,SRXX
CALL HISTO(CREX,NSAMP,CREXN,CREXX,D,20,RY,NT)
WRITE(6,121) ANAME1,D
121 FORMAT(10X,A8,* HISTOGRAM, EACH INTERVAL IS *,'F10.4,' UNITS'/)
WRITE(6,122) (I,I=1,20), (RY(J),J=1,20),NT
122 FORMAT(5X,2016,1/4X,2016,1/4X,'TOTAL NUMBER OF SAMPLES=','I6)
WRITE(6,123) ANAME2,CREMX,CREMN,RMINM,SRMX
CALL HISTO(CREX,NSAMP,CREXN,CREMX,D,20,RY,NT)
WRITE(6,121) ANAME2,D
WRITE(6,122) (I,I=1,20), (RY(J),J=1,20),NT
WRITE(6,123) ANAME3,CRTXX,CRTXN,RTMXM,SRTXX
CALL HISTO(CRTX,NSAMP,CRTXN,CRTXX,D,20,RY,NT)
WRITE(6,121) ANAME3,D
WRITE(6,122) (I,I=1,20), (RY(J),J=1,20),NT
RMAXM=0.
WRITE(6,124) CREXA(IS),CREMA(IS),CRTXA(IS)
WRITE(6,125) ANAME1,CREXXA,CREXNA,RMAXA,SRXA
CALL HISTO(CREX,NSAMP,CREXNA,CREXXA,D,20,RY,NT)
WRITE(6,121) ANAME1,D
WRITE(6,122) (I,I=1,20), (RY(J),J=1,20),NT
WRITE(6,125) ANAME2,CREXXA,CREXMA,RMINMA,SRMMA
CALL HISTO(CREX,NSAMP,CREXMA,CREXXA,D,20,RY,NT)
WRITE(6,121) ANAME2,D
WRITE(6,122) (I,I=1,20), (RY(J),J=1,20),NT
WRITE(6,125) ANAME3,CRTXXA,CRTXNA,RTMXMA,SRTXXA
CALL HISTO(CRTX,NSAMP,CRTXNA,CRTXXA,D,20,RY,NT)
WRITE(6,121) ANAME3,D
WRITE(6,122) (I,I=1,20), (RY(J),J=1,20),NT
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 SEED
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)
130
Appendix D PERTB Computer Program

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
NT=NT+NY(I)
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 FIGER(A,N,IA,MODE,EVR,EVI,EVEC,IE,WORK,IPRINT,'A+B*K ',IIS,IPRT)
RETURN
END

C
SUBROUTINE ESHIFT(EVAL,EVEC,N,NC,EVALR,EVALI,EVEC,WORK,NE)
COMPLEX CEVAL(1),CEVEC(N,1),CNORM
DIMENSION EVAL(1),EVAL(1),EVEC(N,E),WORK(NE,1)
DIMENSION INDX(16),EVALA(16)
DO 10 I=1,N
EVALR(I) = REAL(CEVAL(I))
Appendix D PERTB Computer Program

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
FMAX = FMAG
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 FVEC(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(FVALI(I))
I = I + 1
FVALI(I) = -FVALI(I-1)
GO TO 40
END

ORDER — ORDERS THE ELEMENTS OF A VECTOR BY ALGEBRAIC SIZE. THE
RESULTING PERMUTATION OF THE INDICES IS RETURNED IN VECTOR IV.
IF MODE = 1, THE ELEMENTS OF VECTOR V1 ARE ALSO REORDERED
IF MODE = 2, BOTH VECTORS V1 AND V2 ARE REORDERED

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
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
SUBROUTINE EIGER(A,N,IA,MODE,EVR,EVI,EVEC,IF,WORK,IPRINT,ANAME)

DIMENSION A(IA,1),EVEC(IF,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 FIGRF(A,N,IA,IJOB,CEVAL,CEVEC,IC,WORK,IER)

IF(IER.GT.0)WRITE(NPRINT,102)

CALL ESHIFT(CEVAL,CEVEC,N,IC,EVR,WORK,IE)

RETURN

END

SUBROUTINE CEVPRT(CEVAL,CEVEC,N,IC,WORK,MODE)

DIMENSION CEVAL(1),CEVEC(IC,1)

CALL WRTMAT(WORK,N,N,'E-VALS')

RETURN

END
DO 30 J=1,N
30  WORK(I,J) = AIMAG(CEVEC(I,J))
CALL WRTMAT(4ORK,N,N,N,*I'M E-VEC*)
RETURN
END

C
C WRTMAT — WRITES MATRIX A
C
SUBROUTINE WRTMAT(A,N,M,IA,ANAME)
DIMENSION A(N,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 — 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
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 VARIABLES
C
SUBROUTINE SWAP(A,B)
C = A
A = B
B = C
RETURN
END

C********** SUBROUTINE APLOT **********
C
SUBROUTINE APLOT(X,Y,NPT,NPLOT,XYLIMS,WIDTH,HEIGHT,TICKL,
1 NCLASS,XLRL,NXC,YLBL,YLC,TITL,NTTL,NCAMES,NCINDX)
C
C NOTE: THE X( ) , Y( ) ARRAYS ARE UNCHANGED IN THIS SUBROUTINE
Appendix D PERTB Computer Program

APLOT INPUTS:

X( ) = ARRAY OF X-COORDINATES OF POINTS
Y( ) = ARRAY OF Y COORDINATES OF POINTS
NPT( ) = NO. OF POINTS (X,Y PAIRS) IN EACH CURVE
NPLT = NUMBER OF CURVES
XYLIMS(4) = LIMITS OF X AND Y AXES IN THE ORDER XMIN, XMAX, YMIN, YMAX
IF XYLIMS( ) = 0, SCALING IS AUTOMATIC
WIDTH = WIDTH OF PLOT IN INCHES
HEIGHT = HEIGHT OF PLOT IN INCHES
TICKLE = LENGTH OF TICK MARKS ON AXES IN INCHES AND HEIGHT OF SCALE NOS
IN INCHES. IF TICKL IS NEGATIVE, PUT THE TICK MARKS INSIDE AXES
NCASE = A NO. PRINTED AT THE UPPER RIGHT HAND CORNER OF PLOT FOR
INDEXING PURPOSES.
XLBL( ) = ALPHANUMERIC STRING FOR X-AXIS LABEL
NXC = NUMBER OF CHARACTERS IN X AXIS LABEL
YLBL( ), NYC = LIKEWISE FOR Y AXIS LABEL
TITL( ) = ALPHANUMERIC STRING FOR TITLE
NTTL = NUMBER OF CHARACTERS IN TITLE
CNA( ) = ARRAY OF FOUR-CHARACTER IDENTIFIERS FOR THE NPLT CURVES
NCINDX( ) = INTEGER (.GE.1. AND .LE.NPT( )) INDICATING LOCATION
OF CNA IDENTIFIERS FOR EACH CURVE

DIMENSION X(1), Y(1), XAR(10), NPT(1), XYLIMS(1)
DIMENSION XLBL(1), YLBL(1), TITL(1), CNA(1), NCINDX(1)
COMMON/NOUT/NODEV
DATA EPS/1.E-8/
WRITE(NODEV,100)(NPLT, NCASE, (NPT(I), I=1, NPLT))
100 FORMAT(' APLOT CALLED NPLT = ', I5, ' NCASE = ', I3, ' NPT( ) = ', I4, ', 10I4,/, (10X, 10I4))
C
DO 110 I=1, 10
C
110 NNP = NPT(I)
C109 FORMAT(2X, I5, 'COORDINATES=', 6F10.4)
IF(NPLT.LE.0)RETURN
XMIN = XYLIMS(1)
XMX = XYLIMS(2)
YMN = XYLIMS(3)
YMX = XYLIMS(4)
TICK = ABS(TICKL)
HL = 12*TICK
H2 = HL + WIDTH
V1 = 9*TICK
V2 = V1 + HEIGHT
C
FIND TOTAL NUMBER OF POINTS ON PLOT

NP0INT = 0
DO 2 I=1, NPLT
2 NP0INT = NPOINT + NPT(I)
IF(XMIN.GE.XMX)GO TO 4
XMIN = XMIN
XMAX = XMX

135
Appendix D PERTB Computer Program

4 GO TO 6
6 CALL MINMAX(X,NPOINT,XMIN,XMAX)
6 IF(YMN.GE.YMX)GO TO 8
7 YMIN = YMN
7 YMAX = YMIX
7 GO TO 10
8 CALL MINMAX(Y,NPOINT,YMIN,YMAX)

C SCALE X-AXES
C
10 CALL SCALE(XMIN,XMAX,XST,XDEL,NXP)
10 XFIN = XST + XDEL*XST
7 DX = XFIN - XST
7 XF = (H2-H1)/DX

C SCALE Y-AXES
C
7 CALL SCALE(YMIN,YMAX,YST,YDEL,NYP)
7 YFIN = YST + YDEL*YFIN
7 DY = YFIN - YST
7 YF = (V2-V1)/DY
7 WRITE(NODEV,101)XMIN,XMAX,XST,XDEL,XFIN,NXP
7 WRITE(NODEV,102)YMIN,YMAX,YST,YDEL,YFIN,NYP
101 FORMAT(‘ XMIN,XMAX =’,1PE14.7,5X,’XSTART,XDEL,XFINISH =’,
1 3F10.3,5X,’NXP =’,4)
102 FORMAT(’ YMIN,YMAX =’,1PE14.7,5X,’YSTART,YDEL,YFINISH =’,
1 3F10.3,5X,’NYP =’,4)
C DRAW OUTER BORDER FOR PLOTS
C
C IF APPROPRIATE ADD X = 0, Y = 0 AXES
C
77 IF(XST.GT.0. OR.XFIN.LT.0.)GO TO 12
77 XX = -XST*XF + H1
77 CALL PLOT(XX,V1,3)
77 CALL PLOT(XX,V2,2)
12 IF(YST.GT.0. OR.YFIN.LT.0.)GO TO 14
12 YY = -YST*YF + V1
12 CALL PLOT(H1,YY,3)
12 CALL PLOT(H2,YY,2)
14 CONTINUE
C ADD TITLE, X-AXIS LABEL AND Y-AXIS LABEL
C
7 HT = TICK
7 ANG = 0.
7 ANGS90 = 90.
7 WD = HT
7 XP = (H1+H2)/2. - .5*NTTL*1.5*WD

136
IF(NTTL.GT.0) CALL SYMBOL(XP, V2+4.*HT, L, 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)

ADD CASE NUMBER (IF NONZERO) AND DATE
NNG = 0
IF(NCASE.GT.0) CALL NUMBER(H2+HT, V2, HT', FLOAT(NCASE), ANG, NDG)
CALL DATIMP(H1, V2+2*HT, HT, 'APLOT *)

ADD X-AXIS TICK MARKS AND SCALE
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

ADD Y-AXIS TICK MARKS AND SCALE
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

ADD LABEL FOR EACH CURVE
K = 0
DO 30 I=1,NPLGT
IF(NCINDX(I).LE.0) GO TO 29
JPT = MAXO(MINO(NCINDX(I), NPT(I)), 1)
XX = AMINL(A4AX1(H1, (X(K+JPT)-XST)*XF+H1), H2)
YY = AMINL(A4AX1(V1, (Y(K+JPT)-YST)*YF+V1), V2)
CALL PLOT(XX, YY, 3)
30 CONTINUE

137
CALL PLOT(XX+2.*HT,YY+4.*HT,2)
CALL SYMBOL(XX+3.*HT,YY+4.*HT,HT,CNAMES(I),ANG,4)
N = NPT(I) - 1

PLOT EACH CURVE

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 SYMBOL(XX,YY,.04,1H.,0.,0.)
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)
CALL SYMBOL(XX,YY,.04,1H.,0.,0.)
  IF(J .LE. 3) CALL CIRCLE(XX,YY,.02)
30 CONTINUE
RETURN
END

SUBROUTINE SCALE(XMIN,XMAX,START1,DELI,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
XORD = 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.
DELI = 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 IF(ISHIFT = (XPOINT-XMIN)/DELI
  START1 = XPOINT - OF.1*ISHIFT
  IF(START1.GT.XMIN)START1 = START1 - DELI
  NPTS1 = (XMAX-START1)/DELI
  IF(START1+DELI*(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
SUBROUTINE MINMAX(X,N,XMIN,XMAX)
DIMENSION X(1)
XMIN = X(1)
XMAX = X(1)
IF(N.LT.1)RETURN
DO 10 I=2,N
XMAX = AMAX1(XMAX,X(I))
10 XMIN = AMIN1(XMIN,X(I))
RETURN
END

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.
AMP = AM
IF(XHOURS.GE.12.)AMP = 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,'/',A2,'/',A2,' 5X,12,'','
1 I2,3X,A4)
RETURN
END

C********** SUBROUTINE DATIMP **********
C
SUBROUTINE DATIMP(XP,YP,HT,PNAME)
REAL*8 PNAME
COMMON/NOUT/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, IDAY, ANG, 2)
CALL SYMBOL(XP+3.*WD, YP, HT, IDAY, ANG, 2)
CALL SYMBOL(XP+5.*WD, YP, HT, IHOURS, ANG, 2)
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, IHOURS, 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

FILE PERT INPUT
C
IT CONTAINS THE INPUT FOR THE MONTGOMERY AIRCRAFT
Appendix D  PERTB Computer Program

C &PERT
N=4,
M=2,  
ICON=1,  
PRINT=100,  
IPL=2,  
XL=4HREAL, 4H(LAM, 4HDA) ,  
YL=4HIMAG, 4H(NAR, 4H(Y(LA, 4HDA),  
NXC=12, NYC=16,  
TITL=4HMONT, 4HGOME, 4HRY A, 4H/C C, 4HLOSE, 4HD LO, 4HOP E, 4HIGEN, 4HE PE,  
4HRTUR, 4HBATI, 4HONS, 4HP=1.4H (Ml, 4HN K), 4HTAU=4H 1.0,  
NTTL=72, CNAMES=4H  
NCINDX=0;  
NSAMP=5,  
P=.10,  
A=-3.67984E-1, 1.0E0, -2.4209E-2, 2.58819E-1, 1.7835E-2,  
-3.2279E-2, 2.67949E-1, -1.10395E-1, -9.65926E-1, 2.61875E1, 0., 4.46107E2-2,  
B=7.67183, 0., 1.96959, 0., 2.06549, 0., -2.33843E0, 0.,  
XK = 1.494116E-01, -4.1928029E-01, 1.0535226E-01, 2.0296771E-02, 1.6E+00,  
3.6583920E+00, -2.3151433E-01, -4.7954880E-02,  
&END
&PERT
TITL(19)=4HO.5 ,  
XK = 1.7022794E-01, -6.0164034E-01, 2.6196051E-01, -2.7018290E-02, 2.9F+00,  
4.0597797E+00, -1.185604E+00, 4.8925018E-01,  
&END
&PERT
TITL(19)=4HO.25,  
XK = 1.2936323E+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)=4H1(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 = 1.0369, -.3599, 1.27027, -.322, 2.05683, 3.81589, -.743469, -.52096,  
&END
&PERT
TITL(15)=4H1(LQ),  
TITL(16)=4HR1RH,  
TITL(17)=4H0=  
TITL(13)=4H100  
NTTL=72,  
XK = .3306, -.1794, 1.0913, -.03913, -.69998, 4.8287, 2.6222, -1.4068,  
&END
&PERT
TITL(19)=4H1
Appendix D PERTB Computer Program

\[
\begin{align*}
X_k &= 1.2589, -1.7087, 1.0046, 0.0953, -3068.6, 1.0189, 5.9393, -1.734, \\
&\text{&END} \\
&\text{PERT} \\
&\text{ICONT}=0, \\
&TITL(18)=4H0.04, \\
&X_k = 5.17, -0.59375, 4.8719, 1.1309, 106.5, 0.0654, 7.2623, -2.5309, \\
&\text{&END} \\
&C \\
&C \text{FILE PERT2 INPUT} \\
&C \text{IT CONTAINS THE INPUT FOR THE HALL AIRCRAFT} \\
&C \\
&\text{&PERT} \\
&N=4, m=3, ICONT=1, IPRINT=1000, IPRINT=2, \\
&XLBL=4HREAL, 4H(LAM), 4H(DOA), \\
&YLBL=4HIMAG, 4H(INAR), 4H(YILA), 4H(DOA), \\
&NXC=12, NYC=16, \\
&TITL=4H HALL, 4H AIR, 4H CRAFT, 4H CL, 4H L00, 4H EI, 4HENV, 4HALUE, \\
&4HPER, 4HTURB, 4HATIU, 4HNS, P, 4H H, 4HMIN, 4H K, TITL=4H AUX, 4H1.0, \\
&NNTL=72, CNAMES=4H, NCINDX=0, \\
&NSAMP=5, \\
P=1, \\
A=-3.18, 0.06, 0.022, 3.0, 0.064, 0.63, 0, -27, -99.8, -10.6, 0, 4.5, \\
B=-1.4, 4.3, 0, 1.5, 0, -2.59, 0.037, 2.0, 0, -96, 0, \\
X_k = -1.9732631E-02, -1.598451E-03, 1.9987654E-02, 1.2030686E-02, -9.9E-01, \\
2.5593357E+00, 2.161182E-01, -3.5611261E-03, 4.5716279E+00, -9.4E-01, \\
-5.4777920E-01, -5.1410699E-01, \\
&\text{&END} \\
&\text{&PERT} \\
&TITL(18)=4H0.5, \\
&X_k = 0.2469, -0.001394, -0.01166, 2.4055, 0.0384, -0.0438, -0.30165, 0.02299, 4.14655, \\
&0.0114765, -0.3113, \\
&\text{&END} \\
&\text{&PERT} \\
&TITL(18)=4H0.25, \\
&X_k = 3.5695368E-01, -3.9497081E-02, -1.2787379E-02, 1.4500313E+00, 2.9E-01, \\
&-1.8761892E-01, 1.2922392E-00, 2.6900148E+00, 3.6100111E+00, -4.1E+00, \\
&-1.2453620E+01, 1.4150048E+01, \\
&\text{&END} \\
&\text{&PERT} \\
&TITL(15)=4H(RDB), TITL(16)=4HUST), TITL(17)=4H TAU=.1, TITL(18)=4H1.0, \\
&X_k = 0.03489, -0.1519, -1.2633, 2.137833, -9.33, 2.535, 2.175, 0.02644, 5.5306, \\
&-0.96733, -0.969, -0.525899, \\
&\text{&END} \\
&\text{&PERT} \\
&TITL(18)=4H 0.5, \\
&X_k = -0.3819, -0.016245, -0.012548, 4.133, 0.0386788, -0.0435, -0.27592, \\
&0.0253255, 4.15544, -0.12295, 0.007238, -0.3055, \\
&\text{&END} \\
&\text{&PERT} \\
&TITL(18)=4H0.25, \\
&X_k = 2.0974, 3.30131, -3.344, 1.46229, -0.32129, -6.79063, 1.12387, -2.87326, \\
&-5.17919, -1.04167, -2.05667, -1.36633, \\
&\text{&END} \\
&\text{&PERT} \\
&TITL(15)=4H(LQV), TITL(16)=4H), RH, TITL(17)=4H=, TITL(19)=4HL00,
Appendix D PERTB Computer Program

\[
\begin{align*}
XK &= 0.4513, 0.0574, 0.03798, 0.0916, 0.0222, 0.011679, 0.056979, 0.092429, 0.036339, 0.0754, -0.0208, -0.008925, \\
\end{align*}
\]

\&END

\&PERT

TITL(18) = 4H 1.0,
XK = 0.8653, -0.067577, 0.007833, 0.99586, -0.058595, 0.016246, 0.1175, 0.9399, 0.35148, -0.55619, -0.27992, -0.1,

\&END

\&PERT

ICONT = 0,
TITL(18) = 4H 0.04,
XK = 4.8341, -4.264, 0.02787, 4.98, -5.015, 0.01237, 4.1809, 4.7481, 1.7639, -7.3614, -3.153, -8.6026,

\&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), and more complete discussions of matrix perturbation theory are available in Horscholder (1964) and Wilkinson (1965).

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

$$\tilde{A} v_i = \lambda_i v_i \quad (E1)$$

and

$$(\tilde{A} + E) v'_i = \lambda'_i v'_i \quad (E2)$$

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

$$v^H v = 1, \; w^H v = 1. \quad (E3)$$

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

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

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

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

The eigenvalues of $U^H \tilde{A} w$ will be those of $\tilde{A}$ less $\lambda$ since we have reduced $\tilde{A}$ to block-triangular form by a similarity transformation. Next expand (E2) as
\[(A + E) (v + Up) = (\lambda + \mu) (v + Up) \quad (E4)\]

with
\[Ev + AUp = \nu v + \lambda Up \quad (E5)\]

since
\[EUp = O(\varepsilon^2) \text{ and } uUp = O(\varepsilon^2).\]

Premultiply (E5) by \(U^H\) to obtain an expression for \(p\);
\[U^H Ev + U^H AUp \approx \lambda p\]

or
\[p \approx (\lambda I - U^H A U)^{-1} U^H Ev \quad (E6)\]

which yields equation (17) in Chapter II.

To derive equation (16), premultiply (E4) by \(v^H\) to obtain
\[(v^H A^2 + v^H E) (v + Up) = (\lambda + \mu)\]

\[
\lambda + v^H A^2 Up + v^H Ev + v^H E Up = \lambda + \mu.\]

Use the previous result (E6) to obtain
\[\mu = v^H Ev + v^H AU (\lambda I - U^H AU)^{-1} U^H Ev\]

\[+ v^H EU (\lambda I - U^H AU)^{-1} U^H Ev\]

\[
\mu = [1 v^H AU (\lambda I - U^H AU)^{-1} \begin{bmatrix} v^H \\ U^H \end{bmatrix}] Ev \quad (E7)\]

\[+ v^H EU (\lambda I - U^H AU)^{-1} U^H Ev\]

\[\mu = w^H Ev + v^H EU (\lambda I - U^H AU)^{-1} U^H Ev \quad (E8)\]

\[|\mu| \leq ||w^H|| ||E|| + ||E||^2 ||(\lambda I - U^H AU)^{-1}||. \quad (16)\]

To proceed from step (E7) to (E8) we used the representation for the left-eigenvector
\[w^H = [1 v^H AU (\lambda I - U^H AU)^{-1} \begin{bmatrix} v^H \\ U^H \end{bmatrix}] \text{ of } \tilde{A}. \quad (E9)\]

This identity can be derived as follows. Let \([1 z] \text{ be a left-eigenvector of the matrix}
\[\begin{bmatrix} vU \end{bmatrix}^H \tilde{A} [vU],\]

\[\tilde{A} \]
and solve for $z^H$. Once we find $z^H$, $y^H$ is given by

$$y^H = \begin{bmatrix} 1 & z^H \end{bmatrix} \begin{bmatrix} v^H \\ u^H \end{bmatrix}.$$  \hfill (E10)

so

$$\begin{bmatrix} 1 & z^H \end{bmatrix} \begin{bmatrix} \lambda & v^H u^H \\ 0 & u^H u^H \end{bmatrix} = \lambda \begin{bmatrix} 1 & z^H \end{bmatrix}$$

$$[\lambda \ (v^H + z^H u^H)(u^H u^H)] = [\lambda \ z^H]$$

or

$$z^H (\lambda I - u^H u^H) = v^H u^H$$

$$z^H = v^H u^H (\lambda I - u^H u^H)^{-1}$$

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


