Rensselaer Polytechnic Institute
Troy, New York 12181

Final Report

Contract No. NAS8-21131
Covering period Nov. 4, 1968-Nov. 3, 1969

NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION

# DESCRIPTION AND USER'S GUIDE
# GAIN INITIALIZATION PROGRAMS

by Derek E. McBrinn

Submitted on behalf of

Rob Roy
Professor of Systems Engineering

# DESCRIPTION AND USER'S GUIDE - GAIN INITIALIZATION PROGRAMS GRADGN and LERNGN

## I. Introduction

GRADGN and LERNGN are Fortran IV digital computer programs designed to automatically determine stabilizing feedback gains for linear time-invariant systems with output feedback. Such gains are required, for example, for initialization of the SOCDES[1] design procedure.

Given the system

$$\dot{\underline{z}}(t) = A_1 \underline{z}(t) + B_1 \underline{u}(t) \qquad (1)$$

$$\underline{y}(t) = C\underline{z}(t) \qquad (2)$$

where $\underline{z}(t)$ is an NS-dimensional state vector

$\underline{u}(t)$ is an NC-dimensional control vector

$\underline{y}(t)$ is an NU-dimensional output vector

we wish to determine a linear, time-invariant output-feedback controller

$$\underline{u}(t) = - K_1^T \underline{y}(t) \qquad (3)$$

to stabilize the system.

We consider an entirely equivalent problem, where the first NU states of the system

$$\dot{\underline{x}}(t) = A\,\underline{x}(t) + B\underline{u}(t) \qquad (4)$$

are to be fed back by the controller

$$\underline{u}(t) = - K^T \underline{x}(t) \qquad (5)$$

where K has the form

$$K = \left[\begin{array}{cccc} k_{11} & k_{12} & \cdots & k_{1,NC} \\ k_{21} & k_{22} & \cdots & k_{2,NC} \\ \vdots & \vdots & & \vdots \\ k_{NU,1} & k_{NU,2} & \cdots & k_{NU,NC} \\ \hline & 0 & & \end{array}\right] \Bigg\updownarrow NS \tag{6}$$

This yields

$$\begin{aligned} \underline{\dot{x}}(t) &= \left[A - BK^T\right] \underline{x}(t) \\ &\triangleq \hat{A}\, \underline{x}(t) \end{aligned} \tag{7}$$

Both GRADGN and LERNGN are capable of solving the above problem, provided they are started sufficiently close to a solution region in gain space. That is, both GRADGN and LERNGN are designed to reduce the maximum real part of the set of eigenvalues of Eq. (7). An inopportune starting point may result in the programs "stalling out" on a local minimum and failing to determine a satisfactory set of gains, even though such gains may exist.

Both programs normally are started with all gains zero. If, for a specific problem, there exists prior information which indicates a better choice of starting gains, these gains can be used instead.

In general GRADGN is more efficient than LERNGN. LERNGN, however, has some features which may allow it to produce a solution in cases where GRADGN fails. The recommended procedure is to first try GRADGN. If it fails when a solution is felt to exist, try LERNGN or attempt to determine a set of starting gains close to a stable region.

Note that both GRADGN and LERNGN are programmed only to find stabilizing gains. A trivial modification to either program would allow the degree of stability to be specified.

## II GRADGN

The basis of the GRADGN program, as developed in [2], is the equation

$$\frac{\partial \lambda_i}{\partial k_{jk}} = \frac{\underline{v}_i^T \left[ \underline{0} \cdots \underline{0} \,\Big|\, \underline{b}_k \,\Big|\, \underline{0} \cdots \underline{0} \right] \underline{w}_i}{\underline{v}_i^T \, \underline{w}_i} \qquad (j^{th} \text{ column})$$

where $\lambda_i$ is an eigenvalue of A

$\underline{v}_1^T$ and $\underline{w}_i$ are the corresponding row and column eigenvectors

$\underline{b}_k$ is the $k^{th}$ column of the matrix B

The gradient of the maximum real part of the set of eigenvalues of $\hat{A}$, with respect to the gain matrix K, is obtained from Eq. (8). As an aid to convergence the conjugate gradient [3] is then computed. For the $j^{th}$ iteration, the conjugate gradient is defined by

$$CGRAD_j = GRAD_j + \frac{||GRAD_j||^2}{||GRAD_{j-1}||^2} \; CGRAD_{j-1} \qquad (9)$$

where $GRAD_j$ is the $j^{th}$ gradient

$CGRAD_j$ is the $j^{th}$ conjugate gradient

$CGRAD_1 \triangleq GRAD_1$

Thus the conjugate gradient is a linear combination of the actual gradient and the conjugate gradient of the preceding iteration. As is indicated in [3], this combination provides a better direction of gain adjustment than the true gradient. The gains are adjusted along the negative of the conjugate gradient, thereby increasing the stability of the least stable mode at each iteration.

The program, comprising a MAIN and seven subprograms, is described below. A flow chart is given in Fig. 1 and a program listing, together with a sample problem, are given in Appendix A.

MAIN The MAIN program is used for input-output, control of the various subprograms, computation of the gradient and conjugate gradient matrices and adjustment of the gain matrix.

To run a problem the numbers of states, controls, states available for feedback, and the parameter IGAINS are read-in on the first data card using the 4I10 format. IGAINS = 0 sets the initial gain matrix to zero, while IGAINS = 1 indicates that a guess at a stable set of gains is to be read in.

The A matrix of Eq. (4) (AMAT in the program) is then read-in by rows, followed by the B matrix (BMAT in the program) and, if IGAINS = 1, the initial K matrix. The 7F10.4 format is used for all of these.

AMHT Computes AHAT = AMAT - $[\mathrm{BMAT}]\,[K^T]$

VECT Converts AHAT to single-subscript notation AAAA, for use in variable-dimensional subroutines. This conversion is not necessary at some facilities, but is included for maximum utility of the program.

MSQ Computes ASQR = $[\mathrm{AAAA}]^2$ for subsequent use in subroutine EIGVEC

HSBG Preconditions AAAA to upper Hessenberg form, for subsequent use by subroutines ATEIG.

ATEIG Determines the eigenvalues of AHAT.

MAXRT Selects from the eigenvalues of AHAT the eigenvalue having the maximum real part.

EIGVEC Computes the row and column eigenvectors of AHAT corresponding to the eigenvalue having the maximum real part. Error messages SW1, ITER and DIF indicate the success of this operation.

The program may fail in the case of a repeated value of the eigenvalue with the maximum real part, since subroutine EIGVEC may fail to determine eigenvectors for such a case.

## III LERNGN

Like GRADGN, the LERNGN program is aimed at reduction of the maximum real part of the set of eigenvalues of Eq. (7). Unlike GRADGN, however, LERNGN does not depend on calculation of a gradient. Instead, each gain is adjusted individually to reduce the maximum real part of the set of eigenvalues. The adjustment entails a predetermined variation of the gain followed by an iterative procedure to determine its optimum value. The gains are adjusted in sequence, with the program "learning" better gains at each stage. The process continues until either the system is stabilized or the maximum allowable number of adjustment cycles is reached.

The program comprises a MAIN and six subprograms, several of which are also used in GRADGN. A flow chart is given in Fig.2 and a program listing with a sample problem is given in Appendix B.

### Main

Used for input-output, control of the subprograms, and the predetermined variation of each gain. The data cards required to run a problem are exactly the same as in GRADGN.

### AMHT, VECT, HSBG, ATEIG

Same as in GRADGN.

### Function RMAX

Same as MAXRT in GRADGN.

### GITER

Iteratively computes the optimum value of the gain being adjusted.

References

1. Cassidy, John F. Jr., "Optimal Control With Unavailable States", Final Report, Vol. III, Contract No. NAS8-21131, Rensselaer Polytechnic Institute, Nov. 1968.

2. ___________, "Advanced Control System For A Saturn Booster", Progress Report, June 4, 1969 to July 4, 1969, Contract NAS8-21131, Rensselaer Polytechnic Institute.

3. Lasdon, L.S., S.K. Mitter and A.D. Waren, "The Conjugate Gradient Method For Optimal Control Problems". IEEE Transactions Vol AC-12, No. 2, April 1967.

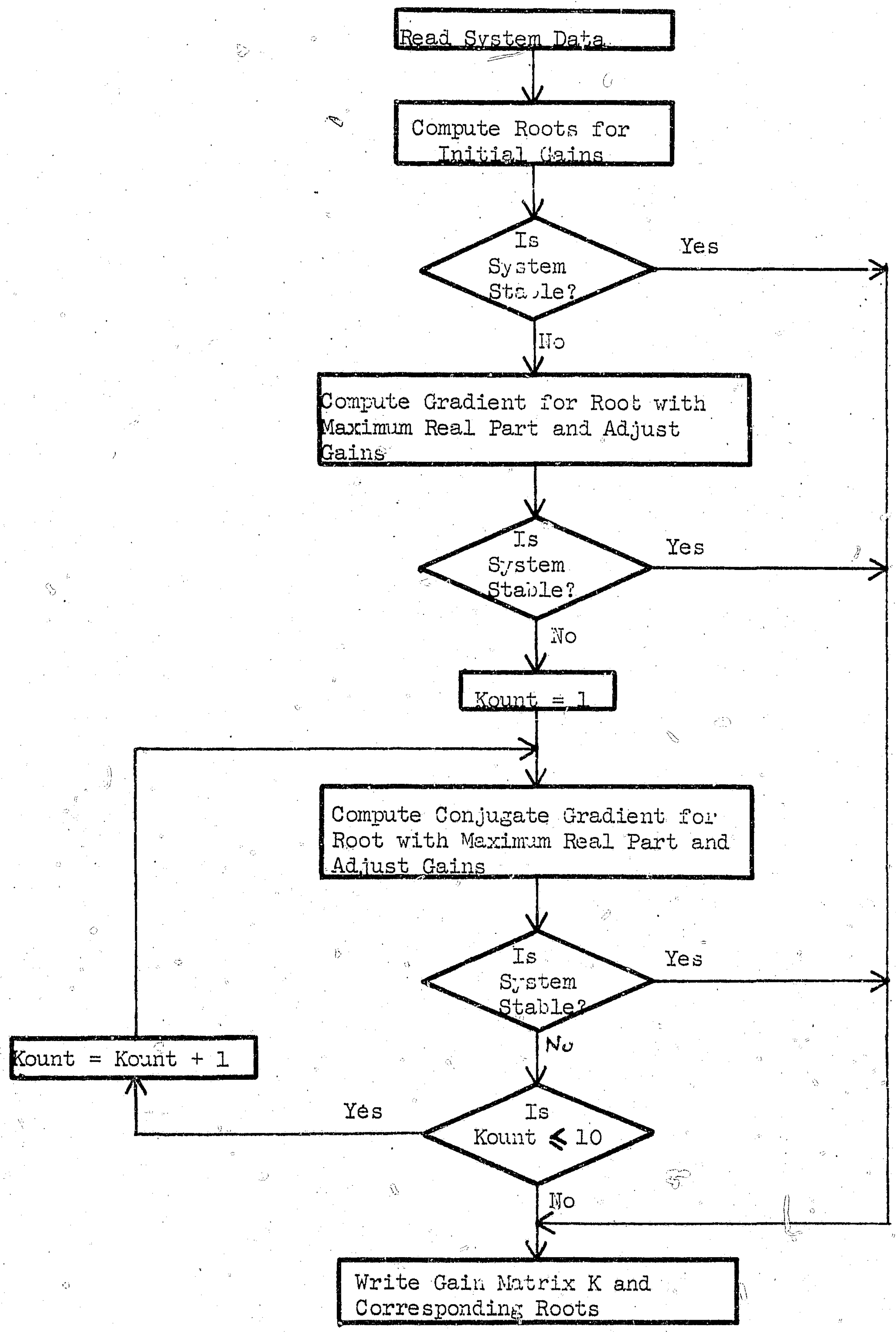


FIGURE 1 FLOW CHART FOR GRADCN

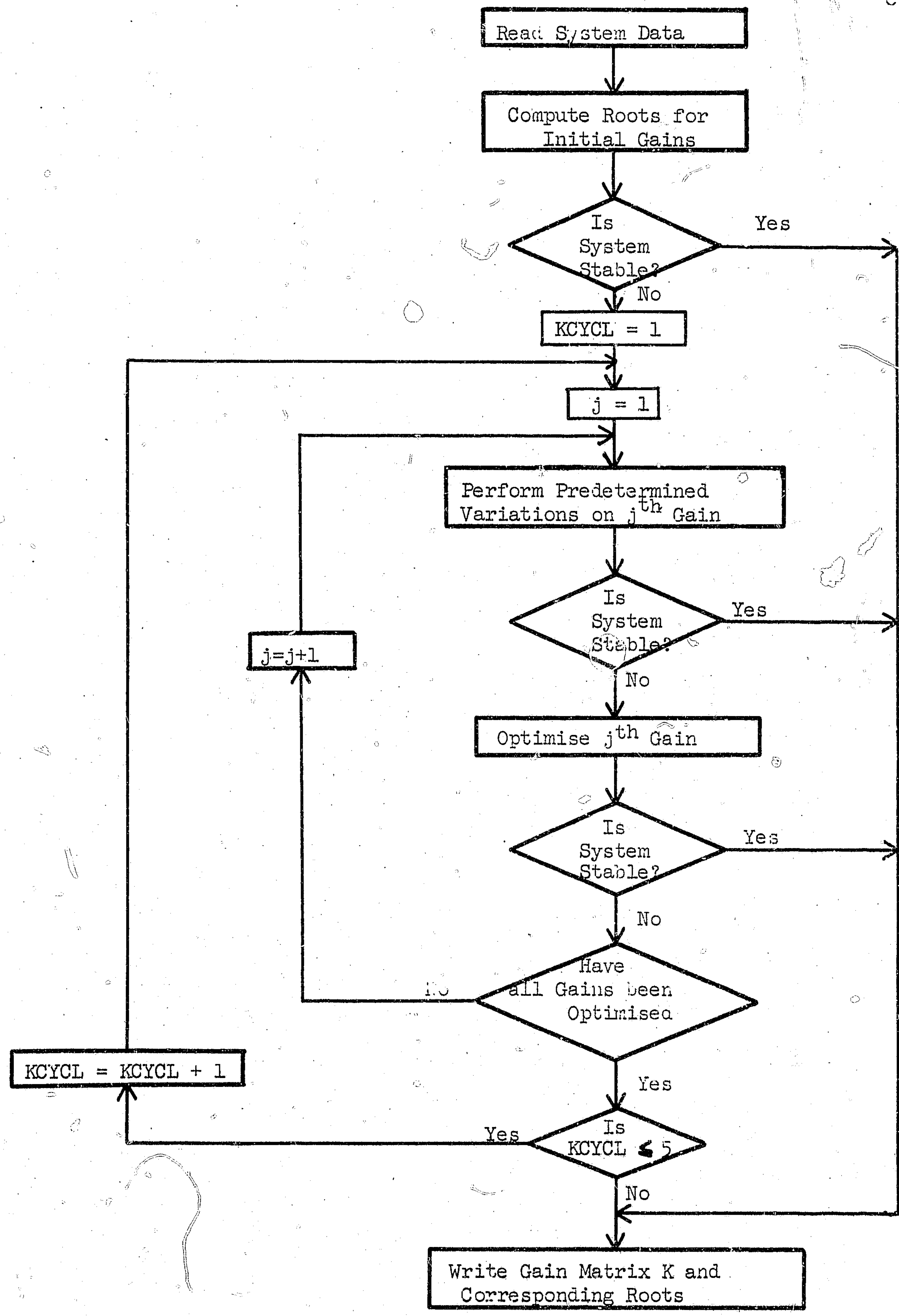


FIGURE 2 FLOW CHART FOR LERNGN

# APPENDIX A

A sample problem was run to illustrate the use of GRADGN. The plant was a seven state model of the Saturn V booster, with the first two states fed back to a single controller. Complete ignorance of stabilizing gains was assumed, so the gain matrix was initialized to zero. The input cards required for the problem are shown in Fig. A-1. A program listing, together with the outputs for the sample problem, is given below.

Solution time for this problem on the IBM-360/50 computer was 10.4 sec. compared with 28.2 sec. for solution of the same problem by the LERNGN program.

```
/JOB            4188        MCBRINN,LINECNT=55
      C
      C       PROGRAM GRADGN
      C       COMPUTES STABILISING OUTPUT FEEDBACK GAINS FOR LINEAR SYSTEMS
      C       WRITTEN BY DEREK E. MC BRINN - SYSTEMS ENGINEERING DIVISION
      C                                      RENSSELAER POLYTECHNIC INSTITUTE
      C                                      TROY, NEW YORK, 12181
      C                                      TELEPHONE 518-270-6324
      C       2ND. SEPTEMBER 1969
      C       COMPLETE USER'S MANUAL AVAILABLE FROM AUTHOR
      C
 1            DIMENSION AMAT(30,30),BMAT(30,30),AHAT(30,30),AAAA(900),ASQR(30,
             130),RR(30),RI(30),IANA(30),W(30,4),XR(30),XI(30),VR(30),VI(30),
             2IROW(30,2),GRADR(30,30),GRADI(30,30),GRAD(30,30),CGRAD(30,30),
             3VRN(30),VIN(30)
 2            REAL K(30,30)
 3       10   FORMAT (7I10)
      C
      C       IGAINS CONTROLS INITIALIZATION OF GAIN MATRIX K
      C
 4            READ (1,10) NS,NC,NU,IGAINS
 5       14   FORMAT (1H1)
 6       15   FORMAT (//T4,'STATES',4X,'CONTROLS',4X,'FEEDBACKS')
 7            WRITE (3,14)
 8            WRITE (3,15)
 9            WRITE (3,10) NS,NC,NU
10            IA=NS
11            NS2=NS*NS
12            DO 40 I=1,NS
13            DO 40 J=1,NC
14         40 K(I,J)=0.
15       50   FORMAT (4E18.7)
16       55   FORMAT (2E20.6)
17         60 FORMAT (7F10.4)
      C
      C       SYSTEM DATA MATRICES ARE READ IN BY  ROWS
      C
18            READ (1,60) ((AMAT(I,J), J=1,NS), I=1,NS)
19       70   FORMAT (//T3,'SYSTEM MATRIX AMAT')
20            WRITE (3,70)
21            WRITE (3,50) ((AMAT(I,J),J=1,NS),I=1,NS)
22            READ (1,60) ((BMAT(I,J), J=1,NC), I=1,NS)
23       80   FORMAT (//T3,'CONTROL MATRIX BMAT')
24            WRITE (3,80)
25            WRITE (3,50) ((BMAT(I,J),J=1,NC),I=1,NS)
      C
      C       INITIALIZATION OF GAIN MATRIX K
      C
26            IF (IGAINS) 94,94,92
27       92   READ (1,60) ((K(I,J),J=1,NC),I=1,NU)
28       94   CONTINUE
29       90   FORMAT (//T3,'GAIN MATRIX K')
30            WRITE (3,90)
31            WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
32            CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
33            CALL VECT(AHAT,NS,AAAA)
34            CALL MSC(AAAA,NS,ASQR)
      C
      C       COMPUTATION OF EIGENVALUES
      C
```

```
      CALL HSBG(NS,AAAA,IA)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
  100 FORMAT (//T2,'ROOTS',5X,'REAL PART',11X,'IMAG PART')
      WRITE (3,100)
      WRITE (3,55) (RR(I),RI(I),I=1,NS)
      CALL MAXRT(NS,RR,RI)
      ROOTR=RR(1)
      ROOTI=RI(1)
  120 FORMAT (//T2,'MAXIMUM REAL PART OF ROOTS')
      WRITE (3,120)
      WRITE (3,50) ROOTR
C
C     CHECK FOR SYSTEM STABILITY
C
      IF (ROOTR) 8000,190,190
 190  CONTINUE
C
C     COMPUTATION OF EIGENVECTORS
C
      CALL EIGVEC(3,AHAT,ASQR,W,IROW,XR,XI,VR,VI,ROOTR,ROOTI,NS,30,0,
     1SW1,ITER,DIF,2)
C
C     SW1 = 0 FOR AN EXACT EIGENVALUE AND NO ROUND-OFF ERROR
C     ITER = NUMBER OF ITERATIONS USED TO FIND EIGENVECTORS.
C     IF TOLERANCE IS NOT ACHIEVED, PROGRAM ACCEPTS VALUES AT ITER = 15.
C     DIF = LARGEST CHANGE IN ANY EIGENVECTOR COMPONENT AT FINAL ITER.
C
 200  FORMAT (//T3,'EIGVEC ERROR MESSAGES')
 210  FORMAT (T3,'SW1=',E10.4,10X,'ITER=',I5,10X,'DIF=',E10.4)
      WRITE (3,200)
      WRITE (3,210) SW1,ITER,DIF
 220  FORMAT (//T3,'EIGENVECTORS CORRES TO MRP EIGENVALUE')
 230  FORMAT (T6,'ROW REAL PT',7X,'ROW IMAG PT',7X,'COL REAL PT',7X,'COL
     1 IMAG PT')
      WRITE (3,220)
      WRITE (3,230)
      WRITE (3,50) (VR(I),VI(I),XR(I),XI(I),I=1,NS)
C
C     NORMALISE EIGENVECTORS INNER PRODUCT
C
      VECMGR=0.
      VECMGI=0.
      DO 240 I=1,NS
      VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
 240  VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
      VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
      DO 250 I=1,NS
      VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
  250 VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
C
C     COMPUTE GRADIENT MATRIX
C
      DO 300 J=1,NU
      DO 300 L=1,NC
      GRADR(J,L)=0.
      GRADI(J,L)=0.
      DO 290 I=1,NS
      GRADR(J,L)=GRADR(J,L)-VRN(I)*BMAT(I,L)
  290 GRADI(J,L)=GRADI(J,L)+VIN(I)*BMAT(I,L)
  300 GRAD(J,L)=GRADR(J,L)*XR(J)+GRADI(J,L)*XI(J)
```

```
  310 FORMAT (//T3,'GRADIENT MATRIX')
      WRITE (3,310)
      WRITE (3,50) ((GRAD(I,J),J=1,NC),I=1,NU)
      GRDSC=0.
      DO 320 I=1,NU
      DO 320 J=1,NC
  320 GRDSC=GRDSC+GRAD(I,J)*GRAD(I,J)
C
C     ADJUST GAIN MATRIX K
C
      DELK=-(RCCTR+.01)/GRDSC
      DO 340 I=1,NU
      DO 340 J=1,NC
  340 K(I,J)=K(I,J)+DELK*GRAD(I,J)
      WRITE (3,90)
      WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
C
C     CONJUGATE GRADIENT IS CCMPUTED FCR SECCNC ANC SUBSEQUENT ITER'S.
C
      DO 350 I=1,NU
      DO 350 J=1,NC
  350 CGRAD(I,J)=GRAD(I,J)
      GRDSC1=GRDSC
      DO 7000 KOUNT=1,10
      CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
      CALL VECT(AHAT,NS,AAAA)
      CALL MSQ(AAAA,NS,ASQR)
C
C     CCMPUTATION CF EIGENVALUES
C
      CALL HSBG(NS,AAAA,IA)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
      WRITE (3,100)
      WRITE (3,55) (RR(I),RI(I),I=1,NS)
      CALL MAXRT(NS,RR,RI)
      RCOTR=RR(1)
      RCOTI=RI(1)
      WRITE (3,120)
      WRITE (3,50) RCCTR
C
C     CHECK FCR SYSTEM STABILITY
C
      IF (RCOTR) 8000,380,380
  380 CONTINUE
      IF (KOUNT-10) 390,7500,7500
  390 CONTINUE
C
C     COMPUTATION CF EIGENVECTCRS
C
      CALL EIGVEC(3,AHAT,ASQR,W,IRCW,XR,XI,VR,VI,RCCTR,RCOTI,NS,30,0,
     1SW1,ITER,DIF,2)
      WRITE (3,200)
      WRITE (3,210) SW1,ITER,DIF
      WRITE (3,220)
      WRITE (3,230)
      WRITE (3,50) (VR(I),VI(I),XR(I),XI(I),I=1,NS)
C
C     NORMALISE EIGENVECTCRS INNER PRODUCT
C
      VECMGR=0.
```

```
      VECMGI=0.
      DO 420 I=1,NS
      VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
  420 VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
      VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
      DO 440 I=1,NS
      VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
  440 VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
C
C     COMPUTE GRADIENT MATRIX
C
      DO 460 J=1,NU
      DO 460 L=1,NC
      GRADR(J,L)=0.
      GRADI(J,L)=0.
      DO 450 I=1,NS
      GRADR(J,L)=GRADR(J,L)-VRN(I)*BMAT(I,L)
  450 GRADI(J,L)=GRADI(J,L)+VIN(I)*BMAT(I,L)
  460 GRAD(J,L)=GRADR(J,L)*XR(J)+GRADI(J,L)*XI(J)
      WRITE (3,310)
      WRITE (3,50) ((GRAD(I,J),J=1,NC),I=1,NU)
      GRDSC=0.
      DO 480 I=1,NU
      DO 480 J=1,NC
  480 GRDSC=GRDSC+GRAD(I,J)*GRAD(I,J)
C
C     COMPUTE CONJUGATE GRADIENT MATRIX
C
      BETA=GRDSC/GRDSC1
      CGRSC=0.
      DO 500 I=1,NU
      DO 500 J=1,NC
      CGRAD(I,J)=BETA*CGRAD(I,J)+GRAD(I,J)
  500 CGRSC=CGRSC+CGRAD(I,J)*CGRAD(I,J)
      GRDSC1=GRDSC
  510 FORMAT (//T3,'CONJUGATE GRADIENT MATRIX')
      WRITE (3,510)
      WRITE (3,50) ((CGRAD(I,J),J=1,NC),I=1,NU)
C
C     ADJUST GAIN MATRIX K
C
      DELK=-(RCCTR+.01)/CGRSC
      DO 520 I=1,NU
      DO 520 J=1,NC
  520 K(I,J)=K(I,J)+DELK*CGRAD(I,J)
      WRITE (3,90)
      WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
 7000 CONTINUE
 7500 WRITE (3,7600)
 7600 FORMAT (//T1,'NO STABILISING GAINS FOUND. THIS DOES NOT IMPLY THAT
     1 NO SUCH GAINS EXIST')
      GO TO 9000
 8000 WRITE (3,8100)
 8100 FORMAT (//T3,'LAST GAINS SHOWN GIVE STABLE ROOTS AS LISTED')
 9000 CONTINUE
      STOP
      END
```

```
      SUBROUTINE AMHT(AMAT,BMAT,K,NS,NC,AHAT)
C
C     COMPUTES AHAT = AMAT - BMAT*K(TRANSPOSE)
C
      DIMENSION AMAT(30,30),BMAT(30,30),AHAT(30,30)
      REAL K(30,30)
      DO 100 I=1,NS
      DO 100 J=1,NS
      AHAT(I,J)=AMAT(I,J)
      DO 100 L=1,NC
  100 AHAT(I,J)=AHAT(I,J)-BMAT(I,L)*K(J,L)
      RETURN
      END
```

```
      SUBRCUTINE VECT(AHAT,NS,AAAA)
C
C     CONVERTS AHAT TC SINGLE SUBSCRIPT FCRM AAAA
C
      DIMENSICN AHAT(30,30),AAAA(900)
      DO 1CO J=1,NS
      DO 1CO I=1,NS
      K=(J-1)*NS+I
  100 AAAA(K)=AHAT(I,J)
      RETURN
      END
```

```
      SUBROUTINE MSQ(AAAA,NS,ASQR)
C
C     COMPUTES ASQR = AAAA*AAAA
C
      DIMENSION AAAA(900),ASQR(30,30)
      DO 100 I=1,NS
      DO 100 J=1,NS
      ASQR(I,J)=0.
      DO 100 K=1,NS
      K1=NS*(J-1)+K
      K2=NS*(K-1)+I
  100 ASQR(I,J)=ASQR(I,J)+AAAA(K1)*AAAA(K2)
      RETURN
      END
```

```
      SUBROUTINE HSBG(N,A,IA)
C
C     CONVERTS A TO UPPER HESSENBERG FORM
C
      DOUBLE PRECISION DABS,DFLOAT,DSIGN,DBLE,DEXP,DLOG,DLOG10,DATAN
     1,DSIN,DCOS,DSQRT,DTANH,DMOD,DMAX1,DMIN1
      DOUBLE PRECISION S
      DIMENSION A(900)
      L=N
      NIA=L*IA
      LIA=NIA-IA
   20 IF(L-3) 360,40,40
   40 LIA=LIA-IA
      L1=L-1
      L2=L1-1
      ISUB=LIA+L
      IPIV=ISUB-IA
      PIV=ABS(A(IPIV))
      IF(L-3) 90,90,50
   50 M=IPIV-IA
      DO 80 I=L,M,IA
      T=ABS(A(I))
      IF(T-PIV) 80,80,60
   60 IPIV=I
      PIV=T
   80 CONTINUE
   90 IF(PIV) 100,320,100
  100 IF(PIV-ABS(A(ISUB))) 180,180,120
  120 M=IPIV-L
      DO 140 I=1,L
      J=M+I
      T=A(J)
      K=LIA+I
      A(J)=A(K)
  140 A(K)=T
      M=L2-M/IA
      DO 160 I=L1,NIA,IA
      T=A(I)
      J=I-M
      A(I)=A(J)
  160 A(J)=T
  180 DO 200 I=L,LIA,IA
  200 A(I)=A(I)/A(ISUB)
      J=-IA
      DO 240 I=1,L2
      J=J+IA
      LJ=L+J
      DO 220 K=1,L1
      KJ=K+J
      KL=K+LIA
  220 A(KJ)=A(KJ)-A(LJ)*A(KL)
  240 CONTINUE
      K=-IA
      DO 300 I=1,N
      K=K+IA
      LK=K+L1
      S=A(LK)
      LJ=L-IA
      DO 280 J=1,L2
      JK=K+J
```

```
      LJ=LJ+IA
  280 S=S+A(LJ)*A(JK)*1.0D0
  300 A(LK)=S
      DO 310 I=L,LIA,IA
  310 A(I)=0.0
  320 L=LL
      GO TO 20
  360 RETURN
      END
```

```
      SUBROUTINE ATEIG(M,A,RR,RI,IANA,IA)
C
C     COMPUTES ROOTS OF UPPER HESSENBERG MATRIX A
C
      DOUBLE PRECISION DABS,DFLOAT,DSIGN,DBLE,DEXP,DLOG,DLOG10,DATAN
     1,DSIN,DCOS,DSQRT,DTANH,DMOD,DMAX1,DMIN1
      DIMENSION A(900),RR(30),RI(30),PRR(2),PRI(2),IANA(30)
      INTEGER P,P1,Q
      E7=1.0E-8
      E6=1.0E-6
      E10=1.0E-10
      DELTA=0.5
      MAXIT=30
      N=M
   20 N1=N-1
      IN=N1*IA
      NN=IN+N
      IF(N1) 30,1300,30
   30 NP=N+1
      IT=0
      DO 40 I=1,2
      PRR(I)=0.0
   40 PRI(I)=0.0
      PAN=0.0
      PAN1=0.0
      R=0.0
      S=0.0
      N2=N1-1
      IN1=IN-IA
      NN1=IN1+N
      N1N=IN+N1
      N1N1=IN1+N1
   60 T=A(N1N1)-A(NN)
      U=T*T
      V=4.0*A(N1N)*A(NN1)
      IF(ABS(V)-U*E7) 100,100,65
   65 T=U+V
      IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67,67,68
   67 T=0.0
   68 U=(A(N1N1)+A(NN))/2.0
      V=SQRT(ABS(T))/2.0
      IF(T)140,70,70
   70 IF(U) 80,75,75
   75 RR(N1)=U+V
      RR(N)=U-V
      GO TO 130
   80 RR(N1)=U-V
      RR(N)=U+V
      GO TO 130
  100 IF(T)120,110,110
  110 RR(N1)=A(N1N1)
      RR(N)=A(NN)
      GO TO 130
  120 RR(N1)=A(NN)
      RR(N)=A(N1N1)
  130 RI(N)=0.0
      RI(N1)=0.0
      GO TO 160
  140 RR(N1)=U
      RR(N)=U
```

```
      RI(N1)=V
      RI(N)=-V
  160 IF(N2)1280,1280,180
  180 N1N2=N1N1-IA
      RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
      EPS=E10*SQRT(RMOD)
      IF(ABS(A(N1N2))-EPS)1280,1280,240
  240 IF(ABS(A(NN1))-E10*ABS(A(NN))) 1300,1300,250
  250 IF(ABS(PAN1-A(N1N2))-ABS(A(N1N2))*E6) 1240,1240,260
  260 IF(ABS(PAN-A(NN1))-ABS(A(NN1))*E6)1240,1240,300
  300 IF(IT-MAXIT) 320,1240,1240
  320 J=1
      DO 360 I=1,2
      K=NP-I
      IF(ABS(RR(K)-PRR(I))+ABS(RI(K)-PRI(I))-DELTA*(ABS(RR(K))
     1     +ABS(RI(K)))) 340,360,360
  340 J=J+I
  360 CONTINUE
      GO TO (440,460,460,480),J
  440 R=0.0
      S=0.0
      GO TO 500
  460 J=N+2-J
      R=RR(J)*RR(J)
      S=RR(J)+RR(J)
      GO TO 500
  480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
      S=RR(N)+RR(N1)
  500 PAN=A(NN1)
      PAN1=A(N1N2)
      DO 520 I=1,2
      K=NP-I
      PRR(I)=RR(K)
  520 PRI(I)=RI(K)
      P=N2
      IF(N-3)600,600,525
  525 IPI=N1N2
      DO 580 J=2,N2
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS) 600,600,530
  530 IPIP=IPI+IA
      IPIP2=IPIP+IA
      D=A(IPIP)*(A(IPIP)-S)+A(IPIP2)*A(IPIP+1)+R
      IF(D)540,560,540
  540 IF(ABS(A(IPI)*A(IPIP+1))*(ABS(A(IPIP)+A(IPIP2+1)-S)+ABS(A(IPIP2+2
     1 ))) -ABS(D)*EPS) 620,620,560
  560 P=N1-J
  580 CONTINUE
  600 Q=P
      GO TO 680
  620 P1=P-1
      Q=P1
      IF(P1-1)680,680,650
  650 DO 660 I=2,P1
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS)680,680,660
  660 Q=Q-1
  680 II=(P-1)*IA+P
      DO 1220 I=P,N1
      II1=II-IA
```

```
      IIP=II+IA
      IF(I-P)720,700,720
  700 IPI=II+1
      IPIP=IIP+1
      G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
      G2=A(IPI)*(A(IPIP)+A(II)-S)
      G3=A(IPI)*A(IPIP+1)
      A(IPI+1)=0.0
      GO TO 780
  720 G1=A(II1)
      G2=A(II1+1)
      IF(I-N2)740,740,760
  740 G3=A(II1+2)
      GO TO 780
  760 G3=0.0
  780 CAP=SQRT(G1*G1+G2*G2+G3*G3)
      IF(CAP)800,860,800
  800 IF(G1)820,840,840
  820 CAP=-CAP
  840 T=G1+CAP
      PSI1=G2/T
      PSI2=G3/T
      ALPHA=2.0/(1.0+PSI1*PSI1+PSI2*PSI2)
      GO TO 880
  860 ALPHA=2.0
      PSI1=0.0
      PSI2=0.0
  880 IF(I-C)900,960,900
  900 IF(I-P)920,940,920
  920 A(II1)=-CAP
      GO TO 960
  940 A(II1)=-A(II1)
  960 IJ=II
      DO 1040 J=I,N
      T=PSI1*A(IJ+1)
      IF(I-N1)980,1000,1000
  980 IP2J=IJ+2
      T=T+PSI2*A(IP2J)
 1000 ETA=ALPHA*(T+A(IJ))
      A(IJ)=A(IJ)-ETA
      A(IJ+1)=A(IJ+1)-PSI1*ETA
      IF(I-N1)1020,1040,1040
 1020 A(IP2J)=A(IP2J)-PSI2*ETA
 1040 IJ=IJ+IA
      IF(I-N1)1080,1060,1060
 1060 K=N
      GO TO 1100
 1080 K=I+2
 1100 IP=IIP-I
      DO 1180 J=C,K
      JIP=IP+J
      JI=JIP-IA
      T=PSI1*A(JIP)
      IF(I-N1)1120,1140,1140
 1120 JIP2=JIP+IA
      T=T+PSI2*A(JIP2)
 1140 ETA=ALPHA*(T+A(JI))
      A(JI)=A(JI)-ETA
      A(JIP)=A(JIP)-ETA*PSI1
      IF(I-N1)1160,1180,1180
```

```
1160 A(JIP2)=A(JIP2)-ETA*PSI2
1180 CONTINUE
     IF(I-N2)1200,1220,1220
1200 JI=II+3
     JIP=JI+IA
     JIP2=JIP+IA
     ETA=ALPHA*PSI2*A(JIP2)
     A(JI)=-ETA
     A(JIP)=-ETA*PSI1
     A(JIP2)=A(JIP2)-ETA*PSI2
1220 II=IIP+1
     IT=IT+1
     GO TC 6C
1240 IF(ABS(A(NN1))-ABS(A(N1N2))) 13C0,1280,1280
1280 IANA(N)=0
     IANA(N1)=2
     N=N2
     IF(N2)1400,14C0,20
1300 RR(N)=A(NN)
     RI(N)=0.0
     IANA(N)=1
     IF(N1)14C0,14C0,1320
1320 N=N1
     GO TC 2C
1400 RETURN
     END
```

```
      SUBROUTINE MAXRT(NS,RR,RI)
C
C     COMPUTES ROOT HAVING MAXIMUM REAL PART
C
      DIMENSION RR(30),RI(30)
      DO 100 I=2,NS
      IF (RR(1)-RR(I)) 50,100,100
   50 RI(1)=RI(I)
      RR(1)=RR(I)
  100 CONTINUE
      RETURN
      END
```

```
      SUBRCUTINE EIGVEC(IVC,  A, B, W, IRCW, XR, XI, VR, VI, RCCTRE,
     1   RCCTIE, NE, NMAX, T2, SW1, COUNTE, ERR,MMM)
C        SUBRCUTINE TO FIND THE EIGENVECTORS OF A NCN-SYMMETRIC MATRIX
C        BY A MCDIFIED WILKINSCN'S INVERSE ITERATION METHOD.
C        CONTRCL IVC CODE IS
C               1  FIND CNLY THE REGULAR EIGENVECTORS    (A X = LAMBCA X)
C               2  FIND CNLY THE TRANSPCSED EIGENVECTCRS (AT V = LAMBDA
C               3  FIND BCTH TYPES CF EIGENVECTCRS.
      DIMENSICN A(30,30),B(30,30),W(30,4),XR(3C),XI(30),VR(30),VI(30),
     1IROW(30,2)
      INTEGER COUNT, COUNTE, T2
      IO1=1
      IO3=3
      RCCTR = RCCTRE
      RCCTI = RCCTIE
      N = NE
      MM = MMP - 1
      N1 = N - 1
      NP1 = N + 1
      IVC1 = IVC - 1
      IVC2 = IVC1 - 1
      COUNT = 1
      DO 400 I=1,N
      W(I,1)=0.CE0
      XR(I)=0.CE0
  400 CONTINUE
      CLIM = 1.0E-4
      IF(RCOTI) 1, 60, 1
C
C        COMPLEX EIGENVALUE.
C
    1 TEMP = - RCCTR - RCCTR
      ISW = 2
      TEMP2=RCCTR*RCCTR+RCCTI*RCCTI
      JJ = 300
      DO 6C6 I = 1, N
      IF(T2) 6C0, 603, 6C0
  600 DO 602 J = 1, N
      JJ = JJ + 1
      IF(JJ - 251) 602, 601, 6C1
  601 JJ = 1
      READ (T2) (W(LL,1), LL = 1,250)
  602 B(I,J) = A(I,J)*TEMP + W(JJ,1)
      GO TC 605
  603 DO 604 J = 1, N
  604 B(I,J) = A(I,J)*TEMP + B(I,J)
  605 B(I,I) = B(I,I) + TEMP2
  606 A(I,I) = A(I,I) - RCCTR
      IF(T2 .NE. 0) REWIND T2
      GO TC 7C0
  607 IF(ICC) 622, 608, 622
C
C        MATRIX SINGULAR.
C
  622 IF(IVC2) 623, 625, 623
  623 DO 624 LL = 1, N
      W(LL,2)=0.0
  624 XI(LL)=0.0
      IF(IVC1) 625, 514, 625
  625 DO 626 LL = 1, N
```

```
      W(LL,4)=0.0
  626 VI(LL)=0.0
      GO TC 511
C
C         MATRIX NCT SINGULAR.
C
  608 DO 609 LL = 1, N
      W(LL,1)=1.0
      W(LL,2)=1.0
      W(LL,3)=1.0
 609  W(LL,4)=1.0
  699 IF(IVC2) 610, 612, 610
  610 DO 611 I = 1, N
      I2 = IROW(I,2)
      XI(I2) = W(I,1)*RCCTI
      DO 611 J = 1, N
  611 XI(I2) = XI(I2) + A(I,J)*W(J,2)
      IF(IVC1) 612, 500, 612
  612 DO 613 I = 1, N
      VI(I) = W(I,3)*RCCTI
      DO 613 J = 1,N
  613 VI(I) = VI(I) + A(J,I)*W(J,4)
      GO TC 499
  615 CERR = 0.0
      IF(IVC2) 616, 619, 616
  616 DO 618 I = 1, N
      XR(I) = -W(I,2)
      DO 617 J = 1, N
  617 XR(I) = XR(I) + A(I,J)*XI(J)
  618 XR(I) = XR(I)/RCCTI
      IF(IVC1) 619, 633, 619
  619 DO 621 I = 1, N
      VR(I) = -W(I,4)
      DO 620 J = 1, N
  620 VR(I) = VR(I) + A(J,I)*VI(J)
  621 VR(I) = VR(I)/RCCTI
C
C         SEARCH VECTORS FCR LARGEST ELEMENT ANC NCRMALIZE.
C
  627 AMAX = 0.0
      DO 629 L = 1, N
      TEMP = VR(L)**2 + VI(L)**2
      IF(TEMP - AMAX) 629, 629, 628
  628 AMAX = TEMP
      I2 = L
  629 CONTINUE
      C1 = VR(I2)/AMAX
      C2 = -VI(I2)/AMAX
      DO 630 L = 1, N
      TEMP = VI(L)
      VI(L) = VR(L)*C2 + TEMP*C1
  630 VR(L) = VR(L)*C1 - TEMP*C2
      IF(CCUNT .EC. 1) GC TC 632
      DO 631 LL = 1, N
  631 CERR = AMAX1(CERR, ABS(VR(LL) - W(LL,3)), ABS(VI(LL) - W(
  632 IF(IVC2) 633, 638, 633
  633 AMAX = C.0
      DO 635 L = 1, N
      TEMP = XR(L)**2 + XI(L)**2
      IF(TEMP - AMAX) 635, 635, 634
```

```
      634 AMAX = TEMP
          I2 = L
      635 CONTINUE
          C1 = XR(I2)/AMAX
          C2 = -XI(I2)/AMAX
          DO 636 L = 1, N
          TEMP = XI(L)
          XI(L) = XR(L)*C2 + TEMP*C1
      636 XR(L) = XR(L)*C1 - TEMP*C2
          IF(CCUNT .EC. 1) GC TC 646
          DO 637 LL = 1, N
      637 CERR = AMAX1(CERR, ABS(XR(LL) - W(LL,1)), ABS(XI(LL) - W(LL,2)))
C
C            TEST FCR CCNVERGENCE.
C
      638 IF(CCUNT .EC. 1) GC TO 646
          IF(CERR .GE. 1.0E-4) GC TC 639
          IF(CERR .GE. CLIM) GC TC 648
          CLIM = CERR
          IF(CLIM .LE. 1.0E-8) GC TC 648
      639 IF(CCUNT .GE. 15) GO TC 68
      647 COUNT = CCUNT + 1
          IF(RCOTI) 642, 673, 642
      642 IF(IVC2) 640, 644, 640
      640 DO 641 LL = 1, N
          W(LL,1) = XR(LL)
      641 W(LL,2) = XI(LL)
          IF(IVC1) 644, 610, 644
      644 DO 645 LL = 1, N
          W(LL,3) = VR(LL)
      645 W(LL,4) = VI(LL)
          GO TC 699
      646 CERR = 0.0
          IF(ICC) 648, 647, 648
      648 ERR = CERR
          COUNTE = CCUNT
          IF(RCCTI) 667, 668, 667
      667 DC 649 I = 1, N
      649 A(I,I) = A(I,I) + RCCTR
          RETURN
       68 PRINT 101, RCCTR, ROCTI, CERR
          GO TC 648
C
C            REAL EIGENVECTCRS.
C
       60 ISW = 1
          DO 651 I = 1, N
          DO 650 J = 1, N
      650 B(I,J) = A(I,J)
      651 B(I,I) = B(I,I) - RCCTR
          GO TC 7CC
      652 IF(ICC) 680, 685, 680
C
C            SINGULAR MATRIX.
C
      680 IF(IVC2) 681, 683, 681
      681 DO 682 L = 1, N
      682 XI(L) = 0.0
          IF(IVC1) 683, 514, 683
      683 DO 684 L = 1, N
```

```
  684 VI(L) = C.0
      GO TC 511
C
C         MATRIX NCT SINGULAR.
C
  685 IF(IVC2) 653, 656, 653
  653 DC 654 L = 1, N
  654 XI(L) = 1.0
      IF(IVC1) 656, 500, 656
  656 DO 657 L = 1, N
  657 VI(L) = 1.0
      GO TC 499
C
C         NCRMALIZE REAL VECTCRS.
C
  655 CERR = 0.0
      IF(IVC2) 658, 662, 658
  658 C1=0.0
      C2=0.0
      DC 660 L = 1, N
      TEMP = ABS(XI(L))
      IF(TEMP - C1) 660, 660, 659
  659 C1 = TEMP
      C2 = XI(L)
  660 CCNTINUE
      DO 661 L = 1, N
      XI(L) = XI(L)/C2
      CERR = AMAX1(CERR, ABS(XI(L) - XR(L)))
  661 XR(L) = XI(L)
      IF(IVC1) 662, 638, 662
  662 C2=0.0
      C1=0.0
      DC 664 L = 1, N
      TEMP = ABS(VI(L))
      IF(TEMP - C1) 664, 664, 663
  663 C1 = TEMP
      C2 = VI(L)
  664 CONTINUE
      DO 665 LL = 1, N
      VI(LL) = VI(LL)/C2
      CERR = AMAX1(CERR, ABS(VI(LL) - W(LL,1)))
      W(LL,1)=VI(LL)
  665 VR(LL)=W(LL,1)
      GO TC 638
  668 IF(IVC2) 669, 671, 669
  669 DC 670 L = 1, N
  670 XI(L) = 0.0
      IF(IVC1) 671, 70, 671
  671 DC 672 L = 1, N
  672 VI(L) = C.0
   70 RETURN
  673 IF(IVC2) 674, 502, 674
  674 DO 675 I = 1, N
      I2 = IRCW(I,2)
  675 XI(I2) = XR(I)
      GO TC 5C0
C
C         BACK SUBSTITUTICN SECTION.
C
  499 IF(IVC2) 5C0, 502, 5C0
```

```
  500 DO 501 I = 2, N
      I1 = I - 1
      DO 501 J = 1, I1
  501 XI(I) = XI(I) - B(I,J)*XI(J)
  511 IF(IVC1) 502, 514, 502
  502 DO 510 I = 1, N
      I1 = I - 1
      IF(I1) 503, 505, 503
  503 DO 504 J = 1, I1
  504 VI(I) = VI(I) - B(J,I)*VI(J)
      IF(ICC) 505, 506, 505
  505 IF(B(I,I)) 506, 507, 506
  506 VI(I) = VI(I)/B(I,I)
      GO TO 510
  507 IF(VI(I)) 508, 509, 508
  508 VI(I) = VI(I)*1.0E+15
      GO TO 510
  509 VI(I) = 1.0
  510 CONTINUE
      IF(IVC2) 514, 525, 514
  514 DO 522 I = 1, N
      IR = NP1 - I
      IF(I - 1) 515, 517, 515
  515 I2 = IR + 1
      DO 516 J = I2, N
  516 XI(IR) = XI(IR) - B(IR,J)*XI(J)
      IF(ICC) 517, 518, 517
  517 IF(B(IR,IR)) 518, 519, 518
  518 XI(IR) = XI(IR)/B(IR,IR)
      GO TO 522
  519 IF(XI(IR)) 520, 521, 520
  520 XI(IR) = XI(IR)*1.0E+15
      GO TO 522
  521 XI(IR) = 1.0
  522 CONTINUE
      IF(IVC1) 525, 529, 525
  525 DO 526 I = 2, N
      IR = NP1 - I
      I2 = IR + 1
      DO 526 J = I2, N
  526 VI(IR) = VI(IR) - B(J,IR)*VI(J)
      DO 527 L = 1, N
      I2 = IROW(L,1)
  527 VR(I2) = VI(L)
      DO 528 L = 1, N
  528 VI(L) = VR(L)
  529 IF(ROOTI) 615, 655, 615
C
C        FACTOR MATRIX.
C
  700 ICC = 0
      SW1=1.0E72
      DO 701 LL = 1, N
  701 IROW(LL,1) = LL
      DO 708 K = 1, N1
      AMAX = ABS(B(K,K))
      IMAX = K
      K1 = K + 1
      DO 702 I = K1, N
      IF(AMAX .GT. ABS(B(I,K))) GO TO 702
```

```
      AMAX = ABS(B(I,K))
      IMAX = I
  702 CONTINUE
      IF(AMAX .LT. SW1) SW1 = AMAX
      IF(AMAX .GE. 1.0E-25) GO TO 723
      B(K,K) = 0.0
      ICC = ICC + 1
      GO TO 708
  723 IF(IMAX .EQ. K) GO TO 704
      DO 703 J = 1, N
      AMAX = B(K,J)
      B(K,J) = B(IMAX,J)
  703 B(IMAX,J) = AMAX
      I2 = IROW(K,1)
      IROW(K,1) = IROW(IMAX,1)
      IROW(IMAX,1) = I2
  704 DO 707 I = K1, N
      IF(B(I,K)) 705, 707, 705
  705 B(I,K) = B(I,K)/B(K,K)
      DO 706 J = K1, N
  706 B(I,J) = B(I,J) - B(K,J)*B(I,K)
  707 CONTINUE
  708 CONTINUE
      AMAX = ABS(B(N,N))
      IF(AMAX - 1.0E-25) 712, 712, 713
 712  B(N,N)=0.0
      SW1=0.0
      ICC = ICC + 1
      GO TO 709
  713 IF(AMAX .LT. SW1) SW1 = AMAX
  709 IF(ICC .LE. ISW) GO TO 710
      IF(MP) 1050,1050,1051
 1051 WRITE(IC3,102) ICC
      COUNTE = 0
      RETURN
 1050 WRITE(IC3,1052) ICC
  710 DO 711 LL = 1, N
      I2 = IROW (LL,1)
  711 IROW(I2,2) = LL
      IF(RCOTI) 607, 652, 607
 1052 FORMAT(///23H ****** WARNING ******    ,'   SUBROUTINE EIGVEC HA
     1FOUND AN EIGENVALUE OF APPARENT MULTIPLICITY',
     1                                            I4,/23X,'   COMPUTATION OF
     2GENVECTOR(S) CONTINUES AT USER S OPTION'//)
  101 FORMAT(38HOMORE THAN 15 LOOPS FOR EIGENVECTOR OF,2E12.4,
     2   14H DIFFERENCE OF,E12.4)
  102 FORMAT(16HO****WARNING**** , I4, 71H ZEROS ON DIAGONAL OF FACTOR
     1 MATRIX.  CHECK FOR MULTIPLE EIGENVALUES./20X,
     2' SUBROUTINE EIGVEC WILL NOT PERFORM COMPUTATION FOR THIS EIGENV
     3TOR '//)
      END

/DATA
```

| STATES | CONTROLS | FEEDBACKS |
|---|---|---|
| 7 | 1 | 2 |

SYSTEM MATRIX AMAT

| | | | |
|---|---|---|---|
| 0.0000000E 00 | 0.1000000E 01 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.2030470E 00 | -0.6534950E 00 | -0.1555000E-02 |
| 0.2558020E 01 | 0.0000000E 00 | -0.1366150E-01 | 0.1000000E 01 |
| -0.4068250E-01 | 0.1998600E-03 | -0.1463000E-01 | -0.3338200E-01 |
| 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.1000000E 01 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 | -0.4466811E 02 |
| -0.1336680E 00 | 0.2546100E 03 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.1000000E 01 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 | -0.5000000E 02 |
| -0.1000000E 02 | | | |

CONTROL MATRIX BMAT

| | | | |
|---|---|---|---|
| 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 | 0.0000000E 00 |
| 0.0000000E 00 | 0.0000000E 00 | 0.5000000E 02 | |

GAIN MATRIX K

| | |
|---|---|
| 0.0000000E 00 | 0.0000000E 00 |

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.477798E 00 | 0.000000E 00 |
| | 0.423402E 00 | 0.000000E 00 |
| | 0.137118E-01 | 0.000000E 00 |
| | -0.668340E-01 | 0.668309E 01 |
| | -0.668340E-01 | -0.668309E 01 |
| | -0.500000E 01 | 0.500000E 01 |
| | -0.500000E 01 | -0.500000E 01 |

MAXIMUM REAL PART OF ROOTS
0.4234025E 00

EIGVEC ERROR MESSAGES
SW1=0.8345E-06 ITER= 4 DIF=0.3576E-06

EIGENVECTORS CORRES TO MRP EIGENVALUE

| ROW REAL PT | ROW IMAG PT | COL REAL PT | COL IMAG PT |
|---|---|---|---|
| -0.1411699E-01 | 0.0000000E 00 | 0.1000000E 01 | 0.0000000E 00 |
| 0.1000000E 01 | 0.0000000E 00 | 0.4234034E 00 | 0.0000000E 00 |
| 0.4375202E 00 | 0.0000000E 00 | 0.8829014E 00 | 0.0000000E 00 |
| 0.2059637E-03 | 0.0000000E 00 | -0.2525228E-26 | 0.0000000E 00 |
| -0.1463000E-01 | 0.0000000E 00 | 0.8821868E-26 | 0.0000000E 00 |
| -0.2263338E 00 | 0.0000000E 00 | -0.1349601E-27 | 0.0000000E 00 |
| -0.2171400E-01 | 0.0000000E 00 | -0.4707256E-27 | 0.0000000E 00 |

GRADIENT MATRIX

| | |
|---|---|
| 0.1364676E 01 | 0.5778083E 00 |

GAIN MATRIX K

| | |
|---|---|
| -0.2693076E 00 | -0.1140257E 00 |

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.108505E 00 | 0.651177E 01 |
| | -0.108505E 00 | -0.651177E 01 |
| | -0.492104E 01 | 0.511454E 01 |
| | -0.492104E 01 | -0.511454E 01 |
| | 0.130264E-01 | 0.357715E 00 |
| | 0.130264E-01 | -0.357715E 00 |
| | -0.141324E 00 | 0.000000E 00 |

MAXIMUM REAL PART OF ROOTS
0.1302638E-01

EIGVEC ERROR MESSAGES
SW1=0.2023E-04 ITER= 6 DIF=0.6199E-05

EIGENVECTORS CORRES TO MRP EIGENVALUE

| ROW REAL PT | ROW IMAG PT | COL REAL PT | COL IMAG PT |
|---|---|---|---|
| 0.6533355E-01 | 0.9011871E 00 | 0.6386257E 00 | -0.4902583E-01 |
| 0.9999998E 00 | 0.1192093E-06 | 0.2584410E-01 | 0.2278044E 00 |
| 0.8332038E-01 | -0.5551251E 00 | 0.6149467E 00 | 0.6074160E-01 |
| 0.1014182E-02 | -0.1339428E-01 | 0.9999993E 00 | 0.2384186E-06 |
| -0.1473719E-01 | -0.6705523E-05 | 0.1301199E-01 | 0.3577101E 00 |
| -0.2389640E 00 | 0.1194525E-01 | 0.1749420E 00 | 0.2243631E-03 |
| -0.2379218E-01 | 0.2042998E-02 | 0.2205972E-02 | 0.6258488E-01 |

GRADIENT MATRIX

| | |
|---|---|
| 0.2796646E 00 | 0.5744416E 00 |

CONJUGATE GRADIENT MATRIX

| | |
|---|---|
| 0.5333089E 00 | 0.6818354E 00 |

GAIN MATRIX K

| | |
|---|---|
| -0.2856960E 00 | -0.1349784E 00 |

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.107712E 00 | 0.648022E 01 |
| | -0.107712E 00 | -0.648022E 01 |
| | -0.491035E 01 | 0.514299E 01 |
| | -0.491035E 01 | -0.514299E 01 |
| | -0.328955E-02 | 0.381773E 00 |
| | -0.328955E-02 | -0.381773E 00 |
| | -0.131366E 00 | 0.000000E 00 |

```
MAXIMUM REAL PART OF RCCTS
   -0.3289552E-02

 LAST GAINS SHOWN GIVE STABLE RCCTS AS LISTEC

COMPILE TIME=     34.22 SEC,EXECUTION TIME=     10.43 SEC,OBJECT CCDE=   37944 BY
```

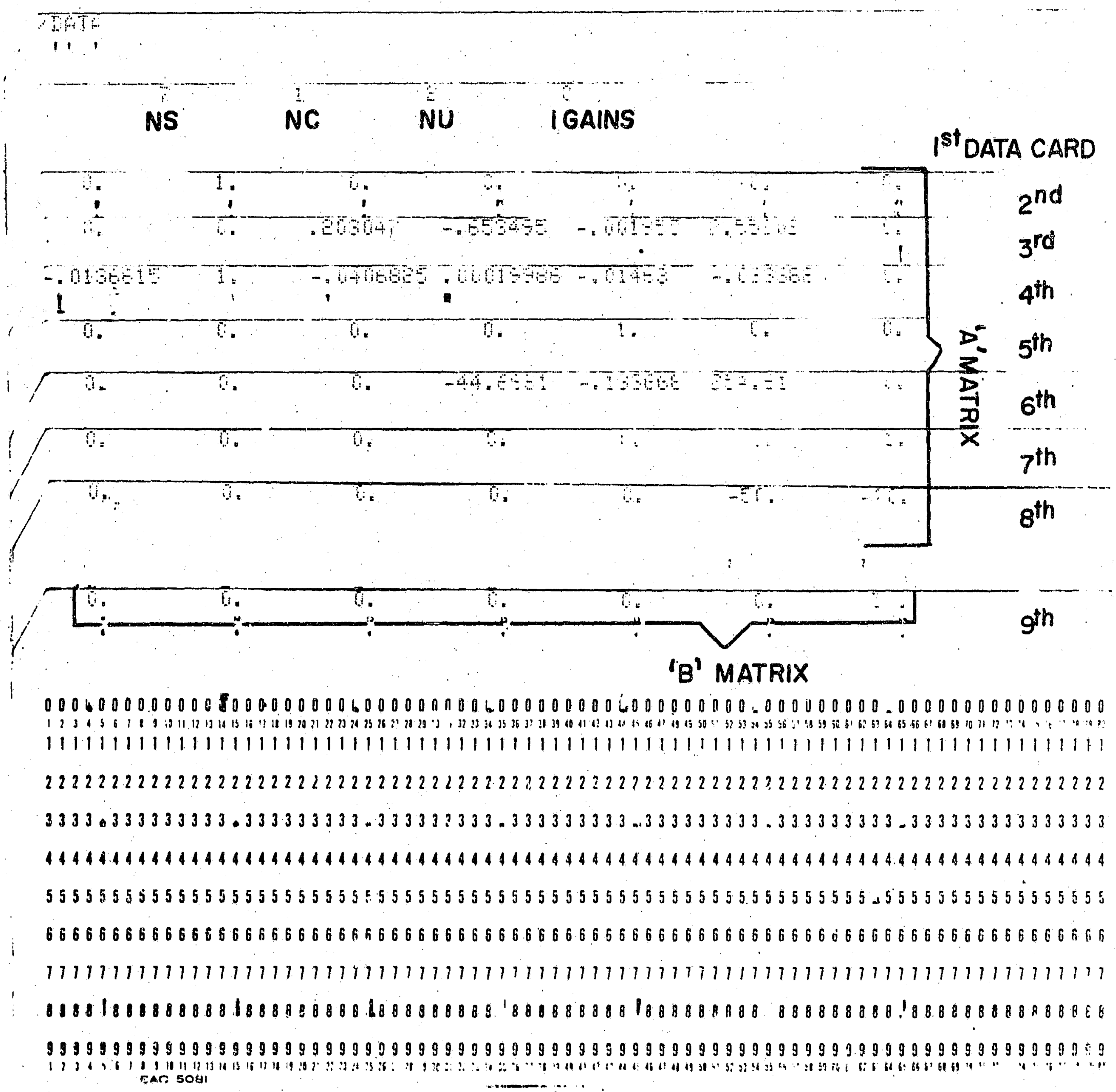


FIG. A-1 INPUT CARDS REQUIRED TO RUN SAMPLE PROBLEM ON EITHER 'GRADGN' OR 'LERNGN' PROGRAMS

# APPENDIX B

The use of LERNGN is illustrated by the same sample problem as was used for GRADGN (see Appendix A). The input cards are unchanged, and are shown in Fig. A-1. We note again that LERNGN took 28.2 sec. to solve this problem, compared with 10.4 sec for GRADGN. A program listing and the outputs for the sample problem follow.

```
/JOB            4188       MCBRINN,LINECNT=55
C
C         PROGRAM LERNGN
C         COMPUTES STABILISING OUTPUT FEEDBACK GAINS FOR LINEAR SYSTEMS
C         WRITTEN BY DEREK E. MC BRINN - SYSTEMS ENGINEERING DIVISION
C                                        RENSSELAER POLYTECHNIC INSTITUT
C                                        TROY, NEW YORK, 12181
C                                        TELEPHONE 518-270-6324
C         2ND. SEPTEMBER 1969
C         COMPLETE USER'S MANUAL AVAILABLE FROM AUTHOR
C
          DIMENSION AMAT(30,30),BMAT(30,30),GV(4),RM(4),RR(30),RI(30),
         1AHAT(30,30),AAAA(900),IANA(30)
          REAL K(30,30)
   10     FORMAT (7I10)
C
C         IGAINS CONTROLS INITIALIZATION OF GAIN MATRIX K
C
          READ (1,10) NS,NC,NU,IGAINS
   14     FORMAT (1H1)
   15     FORMAT (//T4,'STATES',4X,'CONTROLS',4X,'FEEDBACKS')
          WRITE (3,14)
          WRITE (3,15)
          WRITE (3,10) NS,NC,NU
          IA=NS
          NS2=NS*NS
   50     FORMAT (4E18.7)
   55     FORMAT (2E20.6)
   60     FORMAT (7F10.4)
C
C         SYSTEM DATA MATRICES ARE READ IN BY rows
C
          READ (1,60) ((AMAT(I,J), J=1,NS), I=1,NS)
   70     FORMAT (//T3,'SYSTEM MATRIX AMAT')
          WRITE (3,70)
          WRITE (3,50) ((AMAT(I,J),J=1,NS),I=1,NS)
          READ (1,60) ((BMAT(I,J), J=1,NC), I=1,NS)
   80     FORMAT (//T3,'CONTROL MATRIX BMAT')
          WRITE (3,80)
          WRITE (3,50) ((BMAT(I,J),J=1,NC),I=1,NS)
C
C         INITIALIZATION OF GAIN MATRIX K
C
          DO 40 I=1,NS
          DO 40 J=1,NC
   40     K(I,J)=0.
          IF (IGAINS) 94,94,92
   92     READ (1,60) ((K(I,J),J=1,NC),I=1,NU)
   94     CONTINUE
   90     FORMAT (//T3,'GAIN MATRIX K')
          WRITE (3,90)
          WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
          CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
          CALL VECT(AHAT,NS,AAAA)
C
C         COMPUTATION OF EIGENVALUES
C
          CALL HSBG(NS,AAAA,IA)
          CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
  100     FORMAT (//T2,'ROOTS',5X,'REAL PART',11X,'IMAG PART')
```

```
      WRITE (3,100)
      WRITE (3,55) (RR(I),RI(I),I=1,NS)
C
C     COMPUTATION OF ROOT HAVING MAXIMUM REAL PART
C
      RTM=RMAX(NS,RR)
  120 FORMAT (//T2,'MAXIMUM REAL PART OF ROOTS')
      WRITE (3,120)
      WRITE (3,50) RTM
      DO 3000 KCYCL=1,5
      DO 3000 K1=1,NU
      DO 3000 K2=1,NC
C
C     CHECK FOR SYSTEM STABILITY
C
      IF (RTM) 8000,200,200
C
C     PERFORM PREDETERMINED VARIATIONS OF GAIN
C
  200 RM(3)=RTM
      GV(3)=K(K1,K2)
      K(K1,K2)=K(K1,K2)+1.
      WRITE (3,90)
      WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
      CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
      CALL VECT(AHAT,NS,AAAA)
      CALL HSBG(NS,AAAA,IA)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
      WRITE (3,100)
      WRITE (3,55) (RR(I),RI(I),I=1,NS)
      RTM=RMAX(NS,RR)
      WRITE (3,120)
      WRITE (3,50) RTM
      IF (RTM) 8000,400,400
  400 RM(4)=RTM
      GV(4)=K(K1,K2)
      K(K1,K2)=K(K1,K2)-1.5
      WRITE (3,90)
      WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
      CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
      CALL VECT(AHAT,NS,AAAA)
      CALL HSBG(NS,AAAA,IA)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
      WRITE (3,100)
      WRITE (3,55) (RR(I),RI(I),I=1,NS)
      RTM=RMAX(NS,RR)
      WRITE (3,120)
      WRITE (3,50) RTM
      IF (RTM) 8000,600,600
  600 RM(2)=RTM
      GV(2)=K(K1,K2)
      K(K1,K2)=K(K1,K2)-0.5
      WRITE (3,90)
      WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
      CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
      CALL VECT(AHAT,NS,AAAA)
      CALL HSBG(NS,AAAA,IA)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
      WRITE (3,100)
      WRITE (3,55) (RR(I),RI(I),I=1,NS)
```

```
          RTM=RMAX(NS,RR)
          WRITE (3,120)
          WRITE (3,50) RTM
          IF (RTM) 8000,800,800
  800     RM(1)=RTM
          GV(1)=K(K1,K2)
C
C         OPTIMISE GAIN BEING VARIED
C
          DO 2000 KITER=1,6
          CALL GITER(GV,RM,NNG)
          IF (NNG-5) 1000,3000,3000
 1000     K(K1,K2)=GV(NNG)
          WRITE (3,90)
          WRITE (3,50) ((K(I,J),J=1,NC),I=1,NU)
          CALL AMHT(AMAT,BMAT,K,NS,NC,AHAT)
          CALL VECT(AHAT,NS,AAAA)
          CALL HSBG(NS,AAAA,IA)
          CALL ATEIG(NS,AAAA,RR,RI,IANA,IA)
          WRITE (3,100)
          WRITE (3,55) (RR(I),RI(I),I=1,NS)
          RTM=RMAX(NS,RR)
          WRITE (3,120)
          WRITE (3,50) RTM
C
C         CHECK FOR SYSTEM STABILITY
C
          IF (RTM) 8000,1100,1100
 1100     RM(NNG)=RTM
 2000     CONTINUE
 3000     CONTINUE
 7500     WRITE (3,7600)
 7600     FORMAT (//T1,'NO STABILISING GAINS FOUND. THIS DOES NOT IMPLY TH
         1 NO SUCH GAINS EXIST')
          GO TO 9000
 8000     WRITE (3,8100)
 8100     FORMAT (//T3,'ABOVE GAINS GIVE A STABLE SYSTEM WITH LISTED ROOTS
 9000     CONTINUE
          STOP
          END
```

```
      SUBROUTINE AMHT(AMAT,BMAT,K,NS,NC,AHAT)
C
C     COMPUTES AHAT = AMAT - BMAT*K(TRANSPOSE)
C
      DIMENSION AMAT(30,30),BMAT(30,30),AHAT(30,30)
      REAL K(30,30)
      DO 100 I=1,NS
      DO 100 J=1,NS
      AHAT(I,J)=AMAT(I,J)
      DO 100 L=1,NC
  100 AHAT(I,J)=AHAT(I,J)-BMAT(I,L)*K(J,L)
      RETURN
      END
```

```
      SUBRCUTINE VECT(AHAT,NS,AAAA)
C
C     CONVERTS AHAT TC SINGLE SUBSCRIPT FCRM AAAA
C
      DIMENSICN AHAT(30,30),AAAA(9CO)
      DO 100 J=1,NS
      DO 1CC I=1,NS
      K=(J-1)*NS+I
  100 AAAA(K)=AHAT(I,J)
      RETURN
      END
```

```
      SUBROUTINE HSBG(N,A,IA)
C
C     CONVERTS A TO UPPER HESSENBERG FORM
C
      DOUBLE PRECISION DABS,DFLOAT,DSIGN,DBLE,DEXP,DLOG,DLOG10,DATAN
     1,DSIN,DCOS,DSQRT,DTANH,DMOD,DMAX1,DMIN1
      DOUBLE PRECISION S
      DIMENSION A(900)
      L=N
      NIA=L*IA
      LIA=NIA-IA
   20 IF(L-3) 360,40,40
   40 LIA=LIA-IA
      L1=L-1
      L2=L1-1
      ISUB=LIA+L
      IPIV=ISUB-IA
      PIV=ABS(A(IPIV))
      IF(L-3) 90,90,50
   50 M=IPIV-IA
      DO 80 I=L,M,IA
      T=ABS(A(I))
      IF(T-PIV) 80,80,60
   60 IPIV=I
      PIV=T
   80 CONTINUE
   90 IF(PIV) 100,320,100
  100 IF(PIV-ABS(A(ISUB))) 180,180,120
  120 M=IPIV-L
      DO 140 I=1,L
      J=M+I
      T=A(J)
      K=LIA+I
      A(J)=A(K)
  140 A(K)=T
      M=L2-M/IA
      DO 160 I=L1,NIA,IA
      T=A(I)
      J=I-M
      A(I)=A(J)
  160 A(J)=T
  180 DO 200 I=L,LIA,IA
  200 A(I)=A(I)/A(ISUB)
      J=-IA
      DO 240 I=1,L2
      J=J+IA
      LJ=L+J
      DO 220 K=1,L1
      KJ=K+J
      KL=K+LIA
  220 A(KJ)=A(KJ)-A(LJ)*A(KL)
  240 CONTINUE
      K=-IA
      DO 300 I=1,N
      K=K+IA
      LK=K+L1
      S=A(LK)
      LJ=L-IA
      DO 280 J=1,L2
      JK=K+J
```

```
      LJ=LJ+IA
  280 S=S+A(LJ)*A(JK)*1.0D0
  300 A(LK)=S
      DO 310 I=L,LIA,IA
  310 A(I)=0.0
  320 L=L1
      GO TC 20
  360 RETURN
      END
```

```
      SUBRCUTINE ATEIG(M,A,RR,RI,IANA,IA)
C
C     COMPUTES RCCTS CF UPPER HESSENBERG MATRIX A
C
      DOUBLE PRECISION DABS,DFLOAT,DSIGN,DBLE,DEXP,DLOG,DLOG10,DATAN
     1,DSIN,DCOS,DSQRT,DTANH,DMOD,DMAX1,DMIN1
      DIMENSION A(900),RR(30),RI(30),PRR(2),PRI(2),IANA(30)
      INTEGER P,P1,Q
      E7=1.0E-8
      E6=1.0E-6
      E10=1.0E-10
      DELTA=0.5
      MAXIT=30
      N=M
   20 N1=N-1
      IN=N1*IA
      NN=IN+N
      IF(N1) 30,1300,30
   30 NP=N+1
      IT=0
      DO 40 I=1,2
      PRR(I)=0.0
   40 PRI(I)=0.0
      PAN=0.0
      PAN1=0.0
      R=0.0
      S=0.0
      N2=N1-1
      IN1=IN-IA
      NN1=IN1+N
      N1N=IN+N1
      N1N1=IN1+N1
   60 T=A(N1N1)-A(NN)
      U=T*T
      V=4.0*A(N1N)*A(NN1)
      IF(ABS(V)-U*E7) 100,100,65
   65 T=U+V
      IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67,67,68
   67 T=0.0
   68 U=(A(N1N1)+A(NN))/2.0
      V=SQRT(ABS(T))/2.0
      IF(T)140,70,70
   70 IF(U) 80,75,75
   75 RR(N1)=U+V
      RR(N)=U-V
      GO TO 130
   80 RR(N1)=U-V
      RR(N)=U+V
      GO TO 130
  100 IF(T)120,110,110
  110 RR(N1)=A(N1N1)
      RR(N)=A(NN)
      GO TO 130
  120 RR(N1)=A(NN)
      RR(N)=A(N1N1)
  130 RI(N)=0.0
      RI(N1)=0.0
      GO TO 160
  140 RR(N1)=U
      RR(N)=U
```

```
      RI(N1)=V
      RI(N)=-V
  160 IF(N2)1280,1280,180
  180 N1N2=N1N1-IA
      RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
      EPS=E10*SQRT(RMOD)
      IF(ABS(A(N1N2))-EPS)1280,1280,240
  240 IF(ABS(A(NN1))-E10*ABS(A(NN))) 13C0,13C0,25C
  250 IF(ABS(PAN1-A(N1N2))-ABS(A(N1N2))*E6) 1240,1240,260
  260 IF(ABS(PAN-A(NN1))-ABS(A(NN1))*E6)1240,1240,30C
  300 IF(IT-MAXIT) 320,1240,1240
  320 J=1
      DO 360 I=1,2
      K=NP-I
      IF(ABS(RR(K)-PRR(I))+ABS(RI(K)-PRI(I))-DELTA*(ABS(RR(K))
     1     +ABS(RI(K)))) 340,360,36C
  340 J=J+I
  360 CONTINUE
      GO TC (440,460,460,480),J
  440 R=0.0
      S=0.0
      GO TC 5CC
  460 J=N+2-J
      R=RR(J)*RR(J)
      S=RR(J)+RR(J)
      GO TC 5CC
  480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
      S=RR(N)+RR(N1)
  500 PAN=A(NN1)
      PAN1=A(N1N2)
      DO 520 I=1,2
      K=NP-I
      PRR(I)=RR(K)
  520 PRI(I)=RI(K)
      P=N2
      IF(N-3)600,600,525
  525 IPI=N1N2
      DO 580 J=2,N2
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS) 600,6C0,530
  530 IPIP=IPI+IA
      IPIP2=IPIP+IA
      D=A(IPIP)*(A(IPIP)-S)+A(IPIP2)*A(IPIP+1)+R
      IF(D)54C,560,540
  540 IF(ABS(A(IPI)*A(IPIP+1))*(ABS(A(IPIP)+A(IPIP2+1)-S)+ABS(A(IPIP
     1 )) -ABS(D)*EPS) 620,620,560
  560 P=N1-J
  580 CONTINUE
  6C0 Q=P
      GO TC 680
  620 P1=P-1
      Q=P1
      IF(P1-1)680,680,650
  650 DO 660 I=2,P1
      IPI=IPI-IA-1
      IF(ABS(A(IPI))-EPS)680,680,660
  660 Q=Q-1
  680 II=(P-1)*IA+P
      DO 1220 I=P,N1
      II1=II-IA
```

```
      IIP=II+IA
      IF(I-P)720,700,720
  700 IPI=II+1
      IPIP=IIP+1
      G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
      G2=A(IPI)*(A(IPIP)+A(II)-S)
      G3=A(IPI)*A(IPIP+1)
      A(IPI+1)=0.0
      GO TO 780
  720 G1=A(II1)
      G2=A(II1+1)
      IF(I-N2)740,740,760
  740 G3=A(II1+2)
      GO TO 780
  760 G3=0.0
  780 CAP=SQRT(G1*G1+G2*G2+G3*G3)
      IF(CAP)800,860,800
  800 IF(G1)820,840,840
  820 CAP=-CAP
  840 T=G1+CAP
      PSI1=G2/T
      PSI2=G3/T
      ALPHA=2.0/(1.0+PSI1*PSI1+PSI2*PSI2)
      GO TO 880
  860 ALPHA=2.0
      PSI1=0.0
      PSI2=0.0
  880 IF(I-Q)900,960,900
  900 IF(I-P)920,940,920
  920 A(II1)=-CAP
      GO TO 960
  940 A(II1)=-A(II1)
  960 IJ=II
      DO 1040 J=I,N
      T=PSI1*A(IJ+1)
      IF(I-N1)980,1000,1000
  980 IP2J=IJ+2
      T=T+PSI2*A(IP2J)
 1000 ETA=ALPHA*(T+A(IJ))
      A(IJ)=A(IJ)-ETA
      A(IJ+1)=A(IJ+1)-PSI1*ETA
      IF(I-N1)1020,1040,1040
 1020 A(IP2J)=A(IP2J)-PSI2*ETA
 1040 IJ=IJ+IA
      IF(I-N1)1080,1060,1060
 1060 K=N
      GO TO 1100
 1080 K=I+2
 1100 IP=IIP-I
      DO 1180 J=Q,K
      JIP=IP+J
      JI=JIP-IA
      T=PSI1*A(JIP)
      IF(I-N1)1120,1140,1140
 1120 JIP2=JIP+IA
      T=T+PSI2*A(JIP2)
 1140 ETA=ALPHA*(T+A(JI))
      A(JI)=A(JI)-ETA
      A(JIP)=A(JIP)-ETA*PSI1
      IF(I-N1)1160,1180,1180
```

```
 1160 A(JIP2)=A(JIP2)-ETA*PSI2
 1180 CONTINUE
      IF(I-N2)1200,1220,1220
 1200 JI=II+3
      JIP=JI+IA
      JIP2=JIP+IA
      ETA=ALPHA*PSI2*A(JIP2)
      A(JI)=-ETA
      A(JIP)=-ETA*PSI1
      A(JIP2)=A(JIP2)-ETA*PSI2
 1220 II=IIP+1
      IT=IT+1
      GO TO 60
 1240 IF(ABS(A(NN1))-ABS(A(N1N2))) 1300,1280,1280
 1280 IANA(N)=0
      IANA(N1)=2
      N=N2
      IF(N2)1400,1400,20
 1300 RR(N)=A(NN)
      RI(N)=0.0
      IANA(N)=1
      IF(N1)1400,1400,1320
 1320 N=N1
      GO TO 20
 1400 RETURN
      END
```

```
      FUNCTION RMAX(NS,RR)
C
C     COMPUTES ROOT HAVING MAXIMUM REAL PART
C
      DIMENSION RR(30)
      DO 100 I=2,NS
      IF (RR(1)-RR(I)) 50,100,100
   50 RR(1)=RR(I)
  100 CONTINUE
      RMAX=RR(1)
      RETURN
      END
```

```
      SUBROUTINE GITER(GV,RM,NNG)
C
C     OPTIMISES GAIN BEING VARIED
C
      DIMENSION GV(4),RM(4)
      IF (RM(1)-RM(2)) 100,200,300
  100 IF (RM(1)-RM(3)) 120,120,160
  120 IF (RM(1)-RM(4)) 500,500,600
  160 IF (RM(3)-RM(4)) 500,500,600
  200 IF (RM(1)-RM(3)) 220,240,260
  220 IF (RM(1)-RM(4)) 500,600,600
  240 IF (RM(1)-RM(4)) 500,700,600
  260 IF (RM(3)-RM(4)) 500,500,600
  300 IF (RM(2)-RM(3)) 320,320,360
  320 IF (RM(2)-RM(4)) 800,600,600
  360 IF (RM(3)-RM(4)) 900,900,600
  500 NNG=1
      DO 510 I=1,3
      GV(5-I)=GV(4-I)
  510 RM(5-I)=RM(4-I)
      GV(1)=2.*GV(2)
      RETURN
  600 NNG=4
      DO 610 I=1,3
      GV(I)=GV(I+1)
  610 RM(I)=RM(I+1)
      GV(4)=2.*GV(3)
      RETURN
  700 NNG=5
      RETURN
  800 GV(4)=GV(3)
      RM(4)=RM(3)
      IF (RM(1)-RM(3)) 810,810,860
  810 GVT=.8*GV(1)+.2*GV(3)
      IF (GV(2)-GVT) 820,820,830
  820 NNG=3
      GV(3)=GVT
      RETURN
  830 NNG=2
      GV(3)=GV(2)
      RM(3)=RM(2)
      GV(2)=GVT
      RETURN
  860 GVT=.2*GV(1)+.8*GV(3)
      IF (GV(2)-GVT) 870,870,880
  870 NNG=3
      GV(3)=GVT
      RETURN
  880 NNG=2
      GV(3)=GV(2)
      RM(3)=RM(2)
      GV(2)=GVT
      RETURN
  900 GV(1)=GV(2)
      RM(1)=RM(2)
      IF (GV(2)-GV(4)) 910,910,960
  910 GVT=.8*GV(2)+.2*GV(4)
      IF (GV(3)-GVT) 920,920,930
  920 NNG=3
      GV(2)=GV(3)
```

```
      RM(2)=RM(3)
      GV(3)=GVT
      RETURN
  930 NNG=2
      GV(2)=GVT
      RETURN
  960 GVT=.2*GV(2)+.8*GV(4)
      IF (GV(3)-GVT) 970,970,980
  970 NNG=3
      GV(2)=GV(3)
      RM(2)=RM(3)
      GV(3)=GVT
      RETURN
  980 NNG=2
      GV(2)=GVT
      RETURN
      END

  /DATA
```

```
 STATES     CONTROLS     FEEDBACKS
      7            1             2


 SYSTEM MATRIX AMAT
    0.0000000E 00     0.1000000E 01     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.0000000E 00     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.2030470E 00    -0.6534950E 00    -0.1955000E-02
    0.2558020E 01     0.0000000E 00    -0.1366150E-01     0.1000000E 01
   -0.4068250E-01     0.1998600E-03    -0.1463000E-01    -0.3338200E-01
    0.0000000E 00     0.0000000E 00     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.1000000E 01     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.0000000E 00     0.0000000E 00    -0.4466811E 02
   -0.1336680E 00     0.2546100E 03     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.0000000E 00     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.1000000E 01     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.0000000E 00     0.0000000E 00    -0.5000000E 02
   -0.1000000E 02


 CONTROL MATRIX BMAT
    0.0000000E 00     0.0000000E 00     0.0000000E 00     0.0000000E 00
    0.0000000E 00     0.0000000E 00     0.5000000E 02


 GAIN MATRIX K
    0.0000000E 00     0.0000000E 00

ROOTS       REAL PART           IMAG PART
         -0.477798E 00       0.000000E 00
          0.423402E 00       0.000000E 00
          0.137118E-01       0.000000E 00
         -0.668340E-01       0.668309E 01
         -0.668340E-01      -0.668309E 01
         -0.500000E 01       0.500000E 01
         -0.500000E 01      -0.500000E 01


MAXIMUM REAL PART OF ROOTS
    0.4234025E 00


 GAIN MATRIX K
    0.1000000E 01     0.0000000E 00


ROOTS       REAL PART           IMAG PART
          0.132216E 00       0.670811E 01
          0.132216E 00      -0.670811E 01
         -0.509800E 01       0.517533E 01
         -0.509800E 01      -0.517533E 01
         -0.124851E 01       0.000000E 00
          0.104275E 01       0.000000E 00
         -0.370012E-01       0.000000E 00
```

```
MAXIMUM REAL PART OF ROOTS
    0.1042751E 01


 GAIN MATRIX K
   -0.5000000E 00     0.0000000E 00


ROOTS       REAL PART              IMAG PART
         -0.174675E 00          0.667893E 01
         -0.174675E 00         -0.667893E 01
         -0.495482E 01          0.490275E 01
         -0.495482E 01         -0.490275E 01
          0.800452E-01          0.632677E 00
          0.800452E-01         -0.632677E 00
         -0.753986E-01          0.000000E 00


MAXIMUM REAL PART OF ROOTS
    0.8004516E-01


 GAIN MATRIX K
   -0.1000000E 01     0.0000000E 00


ROOTS       REAL PART              IMAG PART
         -0.287612E 00          0.668237E 01
         -0.287612E 00         -0.668237E 01
         -0.491385E 01          0.479901E 01
         -0.491385E 01         -0.479901E 01
          0.143153E 00          0.101065E 01
          0.143153E 00         -0.101065E 01
         -0.576860E-01          0.000000E 00


MAXIMUM REAL PART OF ROOTS
    0.1431528E 00


 GAIN MATRIX K
   -0.8000000E 00     0.0000000E 00


ROOTS       REAL PART              IMAG PART
         -0.241874E 00          0.667997E 01
         -0.241874E 00         -0.667997E 01
         -0.492960E 01          0.484130E 01
         -0.492960E 01         -0.484130E 01
          0.115094E 00          0.877612E 00
          0.115094E 00         -0.877612E 00
         -0.614523E-01          0.000000E 00


MAXIMUM REAL PART OF ROOTS
    0.1150939E 00


 GAIN MATRIX K
   -0.6400000E 00     0.0000000E 00
```

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.205817E 00 | 0.667905E 01 |
| | -0.205817E 00 | -0.667905E 01 |
| | -0.494289E 01 | 0.487430E 01 |
| | -0.494289E 01 | -0.487430E 01 |
| | 0.949065E-01 | 0.756081E 00 |
| | 0.949065E-01 | -0.756081E 00 |
| | -0.667472E-01 | 0.000000E 00 |

MAXIMUM REAL PART OF RCCTS
0.9490651E-01

GAIN MATRIX K
-0.5120000E 00    0.0000000E 00

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.177361E 00 | 0.667887E 01 |
| | -0.177361E 00 | -0.667887E 01 |
| | -0.495378E 01 | 0.490034E 01 |
| | -0.495378E 01 | -0.490034E 01 |
| | 0.811626E-01 | 0.644128E 00 |
| | 0.811626E-01 | -0.644128E 00 |
| | -0.743992E-01 | 0.000000E 00 |

MAXIMUM REAL PART OF RCCTS
0.8116257E-01

GAIN MATRIX K
-0.4096000E 00    0.0000000E 00

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.154808E 00 | 0.667915E 01 |
| | -0.154808E 00 | -0.667915E 01 |
| | -0.496271E 01 | 0.492085E 01 |
| | -0.496271E 01 | -0.492085E 01 |
| | 0.733345E-01 | 0.539701E 00 |
| | 0.733345E-01 | -0.539701E 00 |
| | -0.859719E-01 | 0.000000E 00 |

MAXIMUM REAL PART OF RCCTS
0.7333446E-01

GAIN MATRIX K
-0.4000000E 00    0.0000000E 00

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.152725E 00 | 0.667923E 01 |
| | -0.152725E 00 | -0.667923E 01 |
| | -0.496355E 01 | 0.492278E 01 |

| | |
|---|---|
| -0.496355E 01 | -C.492278E 01 |
| 0.728598E-01 | 0.528975E 00 |
| 0.728598E-01 | -C.528975E 00 |
| -0.875382E-01 | C.CCOCCOE CO |

MAXIMUM REAL PART OF RCCTS
0.7285982E-01

GAIN MATRIX K
-0.3276799E 00 C.CCCCCOOE 00

| ROOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.136940E 00 | 0.667955E 01 |
| | -0.136940E 00 | -0.667955E 01 |
| | -0.496999E 01 | 0.493684E 01 |
| | -0.496999E 01 | -0.493684E 01 |
| | 0.719894E-01 | 0.441165E C0 |
| | 0.719894E-01 | -0.441165E 00 |
| | -0.104508E 00 | C.CCC000E 00 |

MAXIMUM REAL PART OF RCCTS
0.7198936E-01

GAIN MATRIX K
-0.3276799E 00 C.1CCCC00E 01

| RCOTS | REAL PART | IMAG PART |
|---|---|---|
| | 0.332208E-01 | 0.783429E 01 |
| | 0.332208E-01 | -C.783429E 01 |
| | -0.559947E 01 | 0.384622E 01 |
| | -0.559947E 01 | -0.384622E 01 |
| | 0.810661E 00 | 0.CCC000E 00 |
| | 0.233089E 00 | 0.CC0000E C0 |
| | -0.854441E-01 | 0.0000C0E 00 |

MAXIMUM REAL PART OF RCCTS
0.8106608E 00

GAIN MATRIX K
-0.3276799E 00 -0.5CCCC00E C0

| RCOTS | REAL PART | IMAG PART |
|---|---|---|
| | -0.482469E 01 | 0.571536E 01 |
| | -0.482469E 01 | -C.571536E 01 |
| | 0.541153E-01 | 0.591901E 01 |
| | 0.541153E-01 | -0.591901E 01 |
| | -0.248026E 00 | 0.329629E C0 |
| | -0.248026E 00 | -C.329629E C0 |
| | -0.137090E 00 | C.CCC000E 00 |

```
MAXIMUM REAL PART OF RCCTS
    0.5411530E-01


 GAIN MATRIX K
   -0.3276799E 00    -C.1CCCC00E 01


ROOTS       REAL PART              IMAG PART
       -0.494302E 01          0.645551E 01
       -0.494302E 01         -0.645551E 01
        0.542241E 00          0.530754E 01
        0.542241E 00         -0.530754E 01
       -0.114563E 01          0.C000C0E 00
       -0.113634E 00          0.911074E-01
       -0.113634E 00         -0.911074E-01

MAXIMUM REAL PART OF RCCTS
    0.5422406E 00


 GAIN MATRIX K
   -0.3276799E 00    -C.2CCCC00E C0


ROOTS       REAL PART              IMAG PART
       -0.100922E 00          0.637914E 01
       -0.100922E 00         -0.637914E 01
       -0.488015E 01          0.523478E 01
       -0.488015E 01         -0.523478E 01
       -0.498866E-01          0.438476E 00
       -0.498866E-01         -0.438476E 00
       -0.112543E 00          0.C0000C0E C0

MAXIMUM REAL PART OF RCCTS
   -0.4988664E-01


 ABCVE GAINS GIVE A STABLE SYSTEM WITH LISTED RCOTS

COMPILE TIME=    19.36 SEC,EXECUTION TIME=    28.20 SEC,CBJECT CCDE=   230
```