T-71-01502

RENSSELAER POLYTECHNIC INSTITUTE

KNOWLEDGE AND THOROUGHNESS

1824

Rensselaer Polytechnic Institute

Troy, New York

Rensselaer Polytechnic Institute
Troy, New York

Final Report Vol. III
Contract No. NAS8-21131
Covering Period Nov. 4, 1969 - April 4, 1971
NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION

Improved Computational Methods for Initial ate Averaging

by

Derek E. McBrinn

Submitted on behalf of

Rob J. Roy
Professor
Systems Engineering Division

I. Introduction

A major obstacle to the implementation of optimal controllers has been the complexity of such devices. For the general problem the optimal controller is most readily determined in the form of required time histories of the system inputs. This contrasts with the classical output-feedback controller.

Optimal feedback controllers may be determined for the linear-quadratic state regulator problem. Even here, however, they may be difficult to mechanize since they require, in general, time-varying feedback of all system states. Efforts have recently been made[1,2,3] to determine optimal time-invariant output-feedback controllers for such systems. In particular the finite-control-time problem was considered in [3]; the theory was developed and computational techniques suitable for low order systems were presented.

In this report new computational methods are developed to mechanize the theory presented in [3]. The increased computational efficiency associated with these new methods allows the application of the theory to systems of higher order. It also facilitates the computation of optimal piecewise-constant output-feedback controllers for time varying systems.

The techniques developed in this report are illustrated by a seven state model of a Saturn V booster rocket. An optimal controller is computed for this time-varying system over a portion of its flight.

II. Review of Theory

The following is a review of the theory developed in [3].

We are concerned with determination of the optimal time-invariant output-feedback controller for the linear system.

$$\dot{\underline{x}}(t) = A\,\underline{x}(t) + B\,\underline{u}(t) \tag{1}$$

$$\underline{y}(t) = C\,\underline{x}(t) \tag{2}$$

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

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

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

The quadratic cost functional is

$$J = \underline{x}^T(T)\,F\underline{x}(T) + \int_0^T \underline{x}^T(\tau)\,Q\underline{x}(\tau) + \underline{u}^T(\tau)\,R\underline{u}(\tau)\,d\tau \tag{3}$$

with R, F and Q suitably positive (semi) definite. Thus we seek the gain matrix K which minimizes (3) where

$$\underline{u}(t) = -K^T\,\underline{y}(t) \tag{4}$$

The above problem cannot be solved for NF $<$ NS unless the state $\underline{x}(t)$ of the system is specified for some time instant. It has been shown however, that the expected value of J can be minimized if the probability distribution of $\underline{x}(0)$, the system initial state, is known. For the particular case where (a) the outputs $\underline{y}(t)$ comprise the first NF system states, and (b) the system initial state $\underline{x}(0)$ is uniformly distributed on the surface of the unit hypersphere in NS-space, minimization of the expected value of J is equivalent to minimization of the function GF defined by

$$GF = \frac{1}{NS} \operatorname{tr}\left\{ F\phi(T)\phi^T(T) + (Q+KRK^T)\int_0^T \phi(\tau)\phi^T(\tau)d\tau \right\} \tag{5}$$

where tr{ } denotes the matrix trace operation

$\phi(t)$ is the state transition matrix of the system

$$\underline{\dot{x}}(t) = (A - BK^T)\,\underline{x}(t) \tag{6}$$

The restrictions (a) and (b) above are not severe.

The necessary conditions for K to minimize (5) are defined by

$$W = 0 \tag{7}$$

where W is an NF x NC matrix whose coefficients are given by

$$w_{ij} = \operatorname{tr}\left\{ F\phi \frac{\partial \phi^T}{\partial k_{ij}}\Bigg|_T + (Q+KRK^T)\int_0^T \phi(\tau)\frac{\partial \phi^T(\tau)}{\partial k_{ij}}\,d\tau \right.$$

$$\left. + \frac{\partial K}{\partial k_{ij}} RK^T \int_0^T \phi(\tau)\,\phi^T(\tau)d\tau \right\} \tag{8}$$

The gradient of W with respect to the variable feedback gains is given by

$$\frac{\partial w_{gh}}{\partial k_{pq}} = \operatorname{tr}\left\{ F\left[\frac{\partial \phi}{\partial k_{pq}}\frac{\partial \phi^T}{\partial k_{gh}} + \phi\frac{\partial^2 \phi}{\partial k_{pq}\partial k_{gh}}\right]_T \right.$$

$$+ (Q+KRK^T)\int_0^T \frac{\partial \phi}{\partial k_{pq}}\frac{\partial \phi^T}{\partial k_{gh}} + \phi\frac{\partial^2 \phi^T}{\partial k_{pq}\partial k_{gh}}\,d\tau$$

$$+ \frac{\partial K}{\partial k_{pq}} RK^T \int_0^T \frac{\partial \phi}{\partial k_{gh}}\phi^T + \phi\frac{\partial \phi^T}{\partial k_{gh}}\,d\tau$$

$$+ \frac{\partial K}{\partial k_{gh}} \quad RK^T \int_0^T \quad \frac{\partial \phi}{\partial k_{pq}} \phi^T + \frac{\partial \phi^T}{\partial k_{pq}} \quad d\tau$$

$$+ \frac{\partial K}{\partial k_{gh}} \quad R \frac{\partial K^T}{\partial k_{pq}} \int_0^T \phi \, \phi^T \, d\tau \Big\} \tag{9}$$

The above equations provide the basis of a Newton-Raphson iterative method for finding the optimal K. The appropriate relationship is

$$K_{n+1} = K_n - \mu_n \nabla_n^{-1} \; {}^{\circ}\underline{w}_n^{\circ} \tag{10}$$

where the suffix denotes the iteration number

${}^{\circ}\underline{w}^{\circ}$ is a vector arrangement of the matrix w of necessary conditions

∇ is a suitable matrix arrangement of the gradient coefficients defined by (9)

μ is a convergence factor.

III. Computational Algorithm

A method is described below for digital computer mechanization of the Newton-Raphson iterative scheme defined in Section II.

It can be seen from (5)-(10) that the computations required involve products and integrals of $\phi(t)$, $\frac{\partial \phi(t)}{\partial k_{gh}}$ and $\frac{\partial^2 \phi(t)}{\partial k_{gh} \partial k_{pq}}$. It was shown in [3] that

$$\frac{\partial \phi(t)}{\partial k_{gh}} = -\int_0^t \phi(t-\tau) B \frac{\partial K^T}{\partial k_{gh}} \phi(\tau)\, d\tau \tag{11}$$

$$\frac{\partial^2 \phi(t)}{\partial k_{gh} \partial k_{pq}} = -\int_0^t \phi(t-\tau) B \left[\frac{\partial K^T}{\partial k_{gh}} \frac{\partial \phi(\tau)}{\partial k_{pq}} + \frac{\partial K^T}{\partial k_{pq}} \frac{\partial \phi(\tau)}{\partial k_{gh}} \right] d\tau \tag{12}$$

Assuming the eigenvalues of $[A-BK^T]$ to be distinct

$$\phi(t) = M e^{\Lambda t} M^{-1} \tag{13}$$

where Λ is the diagonal matrix of eigenvalues of $[A-BK^T]$.

M is a corresponding modal matrix of eigenvectors.

Equation (13) is used to compute $\phi(t)$ at time instants T/32, T/16, T/8, 2T/8, ..., T. Using these values, $\frac{\partial \phi(t)}{\partial k_{gh}}$ can be approximated at times T/16, T/8, 2T/8, ..., T by

$$\frac{\partial \phi}{\partial k_{gh}} (T/16) \approx -\phi(T/32)\, B \frac{\partial K^T}{\partial k_{gh}} \phi(T/32)\, T/16 \tag{14}$$

$$\frac{\partial \phi}{\partial k_{gh}} (T/8) \approx -\phi(T/16)\, B \frac{\partial K^T}{\partial k_{gh}} \phi(T/16)\, T/8 \tag{15}$$

$$\frac{\partial \phi}{\partial k_{gh}} (2T/8) \approx \phi(3T/16)\ B \frac{\partial K^T}{\partial k_{gh}}\ \phi(T/16)\ T/8$$

$$+ \phi(T/16)\ B \frac{\partial K^T}{\partial k_{gh}}\ \phi(3T/16)\ T/8$$

$$= \phi(T/8) \frac{\partial \phi}{\partial k_{gh}} (T/8) + \frac{\partial \phi}{\partial k_{gh}} (T/8)\ \phi(T/8) \qquad (16)$$

. . . .

$$\frac{\partial \phi}{\partial k_{gh}} \left(\frac{nT+T}{8}\right) \approx \phi(T/8) \frac{\partial \phi}{\partial k_{gh}} \left(\frac{nT}{8}\right) + \frac{\partial \phi}{\partial k_{gh}} \left(\frac{nT}{8}\right) \phi(T/8) \qquad (17)$$

In a similar manner $\frac{\partial^2 \phi(t)}{\partial k_{gh} \partial k_{pq}}$ is approximated at times $nT/8$, $n=1,\ldots,8$ by

$$\frac{\partial^2 \phi}{\partial k_{gh} \partial k_{pq}} (T/8) \approx -\phi(T/16)\ B \left[\frac{\partial K^T}{\partial k_{gh}} \frac{\partial \phi}{\partial k_{pq}} (T/16) + \frac{\partial K^T}{\partial k_{pq}} \frac{\partial \phi}{\partial k_{gh}} (T/16) \right] T/8$$

$$\triangleq \text{D1MFE} + \text{D2MFE} \qquad (18)$$

$$\frac{\partial^2 \phi}{\partial k_{gh} \partial k_{pq}} \left(\frac{nT+T}{8}\right) \approx \phi\ (T/8) \frac{\partial^2 \phi}{\partial k_{gh} \partial k_{pq}} \left(\frac{nT}{8}\right)$$

$$+ \text{D1MFE} \frac{\partial \phi}{\partial k_{pq}} \left(\frac{nT}{8}\right) + \text{D2MFE} \frac{\partial \phi}{\partial k_{gh}} \left(\frac{nT}{8}\right) \qquad (19)$$

The above computations, plus the fact that $\phi(0)$ is the identity matrix and both $\frac{\partial \phi}{\partial k_{gh}} (0)$ and $\frac{\partial^2 \phi}{\partial k_{gh} \partial k_{pq}} (0)$ are null matrices, allows the computation of the various terms in (5)-(9) by Runge-Kutta numerical integration. The remaining computations required for (10) are routine matrix operations, except for assignment of a value to the convergence factor γ.

The optimal value of the convergence factor γ is determined iteratively. It is first set to 1. If GF is reduced γ is doubled, otherwise γ is halved. This process continues until the optimal value of γ is straddled. A quadratic interpolation technique then iterates to the optimal γ. Note that this "best step" procedure requires only the relatively simple computation of the function GF at each step.

A Fortran IV program listing of the algorithm described above comprises Appendix I of this report.

IV. Results and Discussion

Notable computational improvements have resulted from the use of the algorithm described in Section III as compared to that used in [3]. The improvements increase with the dimensionality of the system. Some representative comparisons of computation times are shown in Table I.

Systems Considered	2 state 1 control 1 feedback	2 state 1 control 2 feedbacks	3 state 1 control 1 feedback
Ref 3 algorithm	69 sec.	193 sec.	976 sec.
Present algorithm	8 sec.	82 sec.	17 sec.

Table 1 Comparison of WATFIV Computation Times for Old and New Algorithms

The techniques described in this report are illustrated here by a 7 state model of a Saturn V booster rocket. It is supposed that the objective is to produce piecewise constant output feedback gains for the time varying system.

Some preliminary notes are called for. In optimal control theory, if the control interval is long compared to the time constants of the dominant system modes it is most convenient to consider the control interval to be semi-infinite. For this reason the numerical integration techniques described in Section III were designed for control intervals not longer than about five time constants. If it is desired to consider longer control intervals it will be necessary to increase the number of points used in the numerical integration. This is a trivial modification.

The data for the illustrative example represent the Saturn V booster at a time of 80 seconds after lift-off. The control interval is chosen to be 5 seconds; say from lift-off plus 77.5 seconds to lift-off plus 82.5 seconds. The booster characteristics are adequately represented over this time interval by the data for the 80 second point.

The system matrix is

$$A = \begin{bmatrix} 0. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & .203 & -.6535 & -.0020 & 2.558 & 0. \\ -.0137 & 1. & -.0407 & .0002 & -.0146 & -.0334 & 0. \\ 0. & 0. & 0. & . & 1. & 0. & 0. \\ 0. & 0. & 0. & -44.67 & -.1337 & 254.6 & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. \\ 0. & 0. & 0. & 0. & 0. & -50. & -10. \end{bmatrix} \quad (21)$$

and the control matrix is

$$\underline{b} = \begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \end{bmatrix} \quad (22)$$

The first two states are available for feedback to the single controller. The cost matrices F and Q are chosen to be identity matrices of the appropriate order, and R is equal to .1. The first few pages of program output for this problem comprise Figure 1. The solution achieved is plotted in Figure 2, which shows that the program did indeed attain a minimum expected value of the cost.

STATES	CONTROLS	FEEDBACKS	IGAINS
7	1	2	1

SYSTEM MATRIX A

0.0000000	1.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.2030470	-0.6534950	-0.0019550
2.5580130	0.0000000	-0.0136615	1.0000000
-0.0406825	0.0001999	-0.0146300	-0.0333820
0.0000000	0.0000000	0.0000000	0.0000000
0.0000000	1.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	-44.6681000
-0.1336680	254.6100000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
0.0000000	1.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	-50.0000000
-10.0000000			

CONTROL MATRIX B

0.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	1.0000000	

TERMINAL TIME T = 5.0000000

TERMINAL COST MATRIX F

1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000			

STATE WEIGHTING MATRIX Q

1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000
1.0000000			

CONTROL WEIGHTING MATRIX R

0.1000000

Figure 1 Program Output for Example

```
ITERATION NUMBER          1

GAIN MATRIX
     -15.6600000       -16.5700000

SYSTEM EIGENVALUES
   -0.4844715D 01      0.5445459D 01
   -0.4844715D 01     -0.5445459D 01
   -0.4536563D-01      0.6176736D 01
   -0.4536563D-01     -0.6176736D 01
   -0.1320758D 00      0.3865571D 00
   -0.1320758D 00     -0.3865571D 00
   -0.1300372D 00     -0.0000000D 00

AVERAGE COST =         0.9222255D 03

NECESSARY CONDITIONS VECTOR
    0.2554403D 02     -0.1271373D 03

GRADIENT MATRIX
    0.1487893D 02      0.1133224D 01      0.1133224D 01      0.5122086D 02

INVERSE GRADIENT MATRIX
    0.6732260D-01     -0.1489463D-02     -0.1489463D-02      0.1955625D-01
```


Figure 1 (continued)

PRECEDING PAGE BLANK NOT FILMED

```
ITERATION NUMBER          2

NEW GAINS
      -17.5690567        -14.0456240

SYSTEM EIGENVALUES
  -0.77709120-01      0.62524660 01
  -0.77709120-01     -0.62524660 01
  -0.48514500 01      0.53358073D 01
  -0.48514500 01     -0.53358073D 01
  -0.10474410 00      0.46073590 00
  -0.10474410 00     -0.46073590 00
  -0.10654470 00     -0.00000000 00

AVERAGE COST GF2 =         0.65949820 03

STEP SIZE IS DOUBLED. NEW GAINS ARE
  -0.19478110 02     -0.11521250 02

SYSTEM EIGENVALUES
  -0.10617490 00      0.63291500 01
  -0.10617490 00     -0.63291500 01
  -0.48617540 01      0.52711060 01
  -0.48617540 01     -0.52711060 01
  -0.72371860-01      0.51976800 00
  -0.72371860-01     -0.51976800 00
  -0.93748240-01     -0.00000000 00

AVERAGE COST GF2 =         0.32911980 03

STEP SIZE IS DOUBLED, NEW GAINS ARE
  -0.23296230 02     -0.64724960 01

SYSTEM EIGENVALUES
  -0.15156790 00      0.64833040 01
  -0.15156790 00     -0.64833040 01
  -0.48925430 01      0.51003490 01
  -0.48925430 01     -0.51003490 01
  -0.31482650-02      0.61001830 00
  -0.31482650-02     -0.61001830 00
  -0.79832210-01     -0.00000000 00

AVERAGE COST GF2 =         0.78329980 03

STEP SIZE IS DOUBLED, NEW GAINS ARE
  -0.30932450 02      0.36250080 01

SYSTEM EIGENVALUES
  -0.20079170 00      0.67852840 01
  -0.20079170 00     -0.67852840 01
```

Figure 1 (continued)

```
   -0.4987939D 01     0.4777796D 01
   -0.4987939D 01    -0.4777796D 01
    0.1353055D 00     0.7255425D 00
    0.1353055D 00    -0.7255425D 00
   -0.6750121D-01    -0.0000000D 00

AVERAGE COST GF2 =        0.8597506D 03

STEP SIZE INTERPOLATION, NEW GAINS ARE
   -0.2450952D 02    -0.4868144D 01

SYSTEM EIGENVALUES
   -0.1628903D 00     0.6532388D 01
   -0.1628903D 00    -0.6532388D 01
   -0.4904954D 01     0.5046356D 01
   -0.4904954D 01    -0.5046356D 01
    0.1916654D-01     0.6331545D 00
    0.1916654D-01    -0.6331545D 00
   -0.7699490D-01    -0.0000000D 00

AVERAGE COST GF2 =        0.7717185D 03

STEP SIZE INTERPOLATION, NEW GAINS ARE
   -0.2547031D 02    -0.3597675D 01

SYSTEM EIGENVALUES
   -0.1708504D 00     0.6570928D 01
   -0.1708504D 00    -0.6570928D 01
   -0.4915602D 01     0.5005220D 01
   -0.4915602D 01    -0.5005220D 01
    0.3681169D-01     0.6497780D 00
    0.3681169D-01    -0.6497780D 00
   -0.7506940D-01    -0.0000000D 00

AVERAGE COST GF2 =        0.7660385D 03

STEP SIZE INTERPOLATION, NEW GAINS ARE
   -0.2547031D 02    -0.3597675D 01

ABOVE GAINS ARE BEST STEP FOR THIS ITERATION

     GAIN TOLERANCE ACHIEVED =           0.0057528

REQUIRED STOPPING TOLERANCE =            0.0500000

SYSTEM EIGENVALUES
   -0.1708504D 00     0.6570928D 01
   -0.1708504D 00    -0.6570928D 01
   -0.4915602D 01     0.5005220D 01
```

Figure 1 (continued)

```
   -0.49156020D 01     -0.59052200D 01
    0.36811690D-01      0.64597600D 00
    0.36811690D-01     -0.64597000D 00
   -0.75069400D-01     -0.00000000D 00

AVERAGE COST =          0.76203850D 03

NECESSARY CONDITIONS VECTOR
   -0.25793130D 02     -0.33453640D 02

GRADIENT MATRIX
    0.16879540D 02      0.10946680D 01      0.10946680D 01      0.19561610D 02

INVERSE GRADIENT MATRIX
    0.59464790D-01     -0.34149380D-02     -0.34149380D-02      0.52657570D-01
```

Figure 1 (continued)

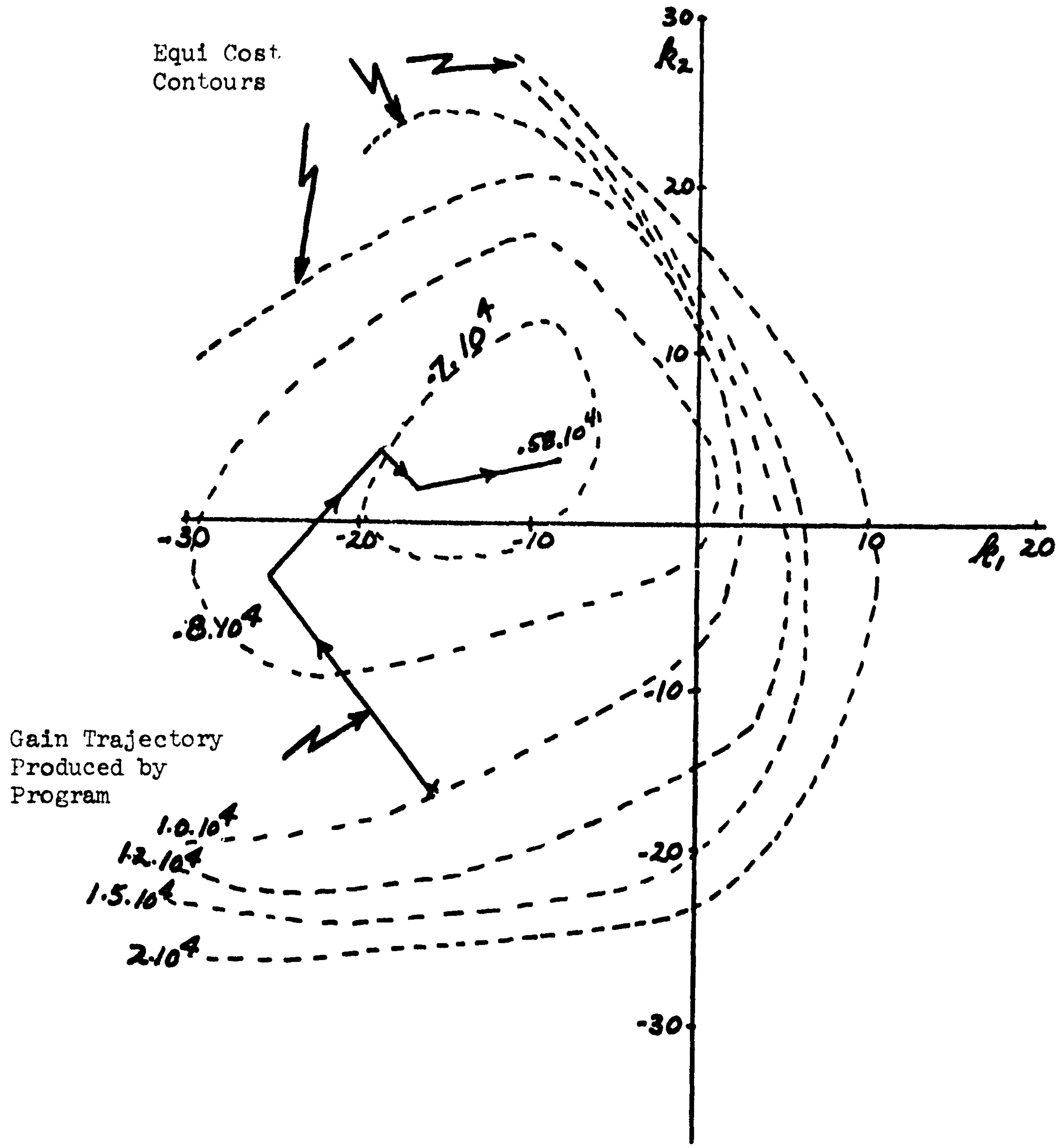

Figure 2 Expected Value of Cost as a Function of Feedback Gains

The example chosen illustrates a perhaps unexpected result. The optimal feedback gains correspond to an unstable system. This arises from the finite control interval used. It simply shows that the cost matrices F, Q and R stressed conservation of control energy at the expense of tightness of control. If stability is necessary then the cost matrices must be chosen accordingly. This differs, of course, from the case of a semi-infinite control interval, where stability of the optimal system is assured.

V. Conclusions

A computational algorithm has been derived to mechanize the theory of optimal time-invariant output-feedback controllers presented in [3]. The new algorithm uses the techniques of numerical integration. It is computationally much faster than the analytic algorithm presented in [3]. This allows its economic use on systems of higher order.

<u>References:</u>

1. Cassidy, J.F., Jr., "Optimal Control with Unavailable States," Ph.D. dissertation, Systems Engineering Division, Rensselaer Polytechnic Institute, Troy, New York, 1969.

2. Levine, W.S., "Optimal Output-Feedback Controllers for Linear Systems," Ph.D. dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1969.

3. McBrinn, D.E., "Optimal Time-Invariant Output Feedback Controllers," Final Report, Vol. 1, Contract No. NAS8-21131, Rensselaer Polytechnic Institute, Troy, New York, 1970.

```
/JOB            4257        MCBRINN,PAGES=100,KP=MIX
      C
      C         PROGRAM IS.FT
      C         DETERMINES OPTIMAL CONSTANT GAIN OUTPUT FEEDBACK CONTROLLERS
      C         FOR FINITE TIME STATE REGULATOR PROBLEMS
      C         OUTPUTS ARE ASSUMED TO BE FIRST NF STATES
      C
            COMMON M(7,7),MI(7,7),RC(7),AHAT(7,7),T,K(7,2),G(7,2),GHAT(7,7),VG
           1RAD(4,4),GRADI(4,4),VW(4),B(7,2),F(7,7),R(2,2),NS,NC,NF
            DIMENSION A(7,7),Q(7,7)
            COMPLEX*16 M,MI,RC
            DOUBLE PRECISION AHAT,T,K,G,GHAT,GF,GF2,VW,VGRAD,GRADI
            DOUBLE PRECISION TEST,TEST1,GNSTOP
            DOUBLE PRECISION DMAX1,DABS
            DOUBLE PRECISION GF3,GF4
            DOUBLE PRECISION G1(7,2)
       10   FORMAT (4F18.7)
       11   FORMAT (4D16.7)
       15   FORMAT (7I10)
       20   FORMAT (14I5)
       25   FORMAT ('1',T4,'STATES',4X,'CONTROLS',4X,'FEEDBACKS',4X,'IGAINS')
       30   FORMAT (//T3,'INVERSE GRADIENT MATRIX')
       35   FORMAT (//T3,'SYSTEM MATRIX A')
       40   FORMAT (//T3,'NEW GAINS')
       45   FORMAT (//T3,'CONTROL MATRIX B')
       50   FORMAT (//T3,'GAIN MATRIX')
       55   FORMAT (//T3,'TERMINAL TIME T =',F18.7)
       60   FORMAT (//T7,'GAIN TOLERANCE ACHIEVED =',F18.7)
       62   FORMAT (//T3,'AVERAGE COST =',D20.7)
       65   FORMAT (//T3,'TERMINAL COST MATRIX F')
       70   FORMAT (//T3,'REQUIRED STOPPING TOLERANCE =',F18.7)
       75   FORMAT (//T3,'STATE WEIGHTING MATRIX Q')
       80   FORMAT (//T3,'CONTROL WEIGHTING MATRIX R')
       85   FORMAT ('1',T,'ITERATION NUMBER',I10)
       86   FORMAT (//T3,'STEP SIZE IS HALVED. NEW GAINS ARE')
       87   FORMAT (//T3,'STEP SIZE IS DOUBLED. NEW GAINS ARE')
       88   FORMAT (//T3,'STEP SIZE INTERPOLATION. NEW GAINS ARE')
       89   FORMAT (//T3,'ABOVE GAINS ARE BEST STEP FOR THIS ITERATION')
       90   FORMAT (//T3,'SOLUTION IS COMPLETE. FOLLOWING GAINS ARE OPTIMAL')
      100   READ (1,20) NS,NC,NF,IGAINS
            READ (1,10) ((A(I,J),J=1,NS),I=1,NS)
            READ (1,10) ((B(I,J),J=1,NC),I=1,NS)
            READ (1,11) T
            READ (1,10) ((F(I,J),J=1,NS),I=1,NS)
            READ (1,10) ((Q(I,J),J=1,NS),I=1,NS)
            READ (1,10) ((R(I,J),J=1,NC),I=1,NC)
            READ (1,11) GNSTOP
            WRITE (3,25)
            WRITE (3,15) NS,NC,NF,IGAINS
            WRITE (3,35)
            WRITE (3,10) ((A(I,J),J=1,NS),I=1,NS)
            WRITE (3,45)
            WRITE (3,10) ((B(I,J),J=1,NC),I=1,NS)
            WRITE (3,55) T
            WRITE (3,65)
            WRITE (3,10) ((F(I,J),J=1,NS),I=1,NS)
            WRITE (3,75)
            WRITE (3,10) ((Q(I,J),J=1,NS),I=1,NS)
            WRITE (3,80)
            WRITE (3,10) ((R(I,J),J=1,NC),I=1,NC)
            NFP1=NF+1
```

```
      NFC=NF*NC
      DO 1000 I=1,NS
      DO 1000 J=1,NC
      G(I,J)=0.0D0
 1000 K(I,J)=0.0D0
      IF(IGAINS) 1200,1200,1100
 1100 READ (1,11) ((K(I,J),J=1,NC),I=1,NF)
 1200 CONTINUE
      DO 1220 I=1,NS
      DO 1220 J=NF,NS
      CHAT(I,J)=C(I,J)
 1220 AHAT(I,J)=A(I,J)
      IT=1
      WRITE (3,85) IT
      WRITE (3,50)
      WRITE (3,10) ((K(I,J),J=1,NC),I=1,NF)
      DO 1225 I=1,NF
      DO 1225 J=1,NC
 1225 G(I,J)=K(I,J)
 1230 CONTINUE
      CALL MATHAT(A,Q)
      CALL STRAM
 1260 CALL FEFN(GF)
      IT=IT+1
      CALL INVERT(VGRAD,GRADI,NFC)
      WRITE (3,30)
      WRITE (3,11) ((GRADI(I,J),J=1,NFC),I=1,NFC)
      CALL NEWRIT
      WRITE (3,85) IT
      WRITE (3,40)
      WRITE (3,10) ((G(I,J),J=1,NC),I=1,NF)
 1280 VMU=1.
      DO 1290 I=1,NF
      DO 1290 J=1,NC
 1290 G1(I,J)=G(I,J)
      GF1=GF
      VMU1=0.
      CALL MATHAT(A,Q)
      CALL STRAM
      CALL GAIN2(GF2)
      IF(GF-GF2.GT.0.0D0) GO TO 1380
      IHALF=1
 1310 GF3=GF2
      IHALF=IHALF+1
      IF(IHALF.GT.5) GO TO 1500
      WRITE (3,86)
      VMU=VMU/2.
      DO 1320 I=1,NF
      DO 1320 J=1,NC
 1320 G(I,J)=(K(I,J)+G(I,J))*0.5
      WRITE (3,11) ((G(I,J),J=1,NC),I=1,NF)
      CALL MATHAT(A,Q)
      CALL STRAM
      CALL GAIN2(GF2)
      IF(GF2-GF.GT.0.0D0) GO TO 1310
      GO TO 1500
 1380 CONTINUE
      WRITE (3,87)
      DO 1390 I=1,NF
      DO 1390 J=1,NC
 1390 G(I,J)=2.*G(I,J)-K(I,J)
```

```
      WRITE (3,11) ((G(I,J),J=1,NC),I=1,NF)
      CALL MATHAT(A,Q)
      CALL STRAM
      CALL GAIN2(GF3)
      IF(GF3-GF2.GT.0.000) GO TO 1500
      VMU1=VMU
      VMU=2.*VMU
      GF1=GF2
      GF2=GF3
      GO TO 1380
 1500 VMU2=VMU
      VMU3=2.*VMU
      KQUAD=1
 1510 KQUAD=KQUAD+1
      IF(KQUAD-3.GT.0) GO TO 1800
      D1MU=VMU2-VMU1
      D2MU=VMU3-VMU1
      DGF1=GF2-GF1
      DGF2=GF3-GF1
      D3MU=.5*(D2MU*D2MU*DGF1-D1MU*D1MU*DGF2)/(DGF1*D2MU-DGF2*D1MU)
      VMU4=VMU1+D3MU
      IF(ABS(D3MU-D1MU).LT.0.001) GO TO 1800
      WRITE (3,88)
      DO 1520 I=1,NF
      DO 1520 J=1,NC
 1520 G(I,J)=VMU4*GI(I,J)+(1.-VMU4)*K(I,J)
      WRITE (3,11) ((G(I,J),J=1,NC),I=1,NF)
      CALL MATHAT(A,Q)
      CALL STRAM
      CALL GAIN2(GF4)
      IF(D3MU.GT.D1MU) GO TO 1600
      IF(GF4.GT.GF2) GO TO 1550
      GF3=GF2
      VMU3=VMU2
      GF2=GF4
      VMU2=VMU4
      GO TO 1510
 1550 GF1=GF4
      VMU1=VMU4
      GO TO 1510
 1600 IF(GF4.GT.GF2) GO TO 1650
      GF1=GF2
      VMU1=VMU2
      GF2=GF4
      VMU2=VMU4
      GO TO 1510
 1650 GF3=GF4
      VMU3=VMU4
      GO TO 1510
 1800 DO 1810 I=1,NF
      DO 1810 J=1,NC
 1810 G(I,J)=VMU2*GI(I,J)+(1.-VMU2)*K(I,J)
      WRITE (3,88)
      WRITE(3,11) ((G(I,J),J=1,NC),I=1,NF)
      WRITE (3,89)
      TEST1=0.000
      DO 2000 I=1,NF
      DO 2000 J=1,NC
      IF (G(I,J).EQ.0.000) GO TO 1900
      TEST=DABS((G(I,J)-K(I,J))/G(I,J))
      TEST1=DMAX1(TEST,TEST1)
```

```
      GO TO 2000
 1900 IF(G(I,J)-K(I,J).EQ.0.000) GO TO 2000
      TEST1=1000.*GNSTOP
 2000 K(I,J)=G(I,J)
      WRITE (3,60) TEST1
      WRITE (3,70) GNSTOP
      IF(TEST1-GNSTOP.GT.0.000) GO TO 1230
      WRITE (3,90)
      WRITE (3,10) ((G(I,J),J=1,NC),I=1,NF)
      WRITE (3,62) GF2
      GO TO 6000
 8000 CONTINUE
 9000 CONTINUE
      STOP
      END

      SUBROUTINE MATHAT(A,Q)
C
C        COMPUTES MATRICES AHAT AND QHAT
C
      COMMON M(7,7),MI(7,7),RC(7),AHAT(7,7),T,K(7,2),G(7,2),QHAT(7,7),VG
     1RAD(4,4),GRADI(4,4),VW(4),B(7,2),F(7,7),R(2,2),NS,NC,NF
      DIMENSION A(7,7),Q(7,7)
      COMPLEX*16 M,MI,RC
      DOUBLE PRECISION AHAT,T,K,G,QHAT,GF,GF2,VW,VGRAD,GRADI
      DO 100 I=1,NS
      DO 100 J=1,NF
      AHAT(I,J)=A(I,J)
      QHAT(I,J)=Q(I,J)
      DO 100 N1=1,NC
       AHAT(I,J)=AHAT(I,J)-B(I,N1)*G(J,N1)
      DO 100 N2=1,NC
  100 QHAT(I,J)=QHAT(I,J)+G(I,N1)*R(N1,N2)*G(J,N2)
      RETURN
      END

      SUBROUTINE STRAM
C
C        COMPUTES THE STATE TRANSITION MATRIX
C
      COMMON M(7,7),MI(7,7),RC(7),AHAT(7,7),T,K(7,2),G(7,2),QHAT(7,7),VG
     1RAD(4,4),GRADI(4,4),VW(4),B(7,2),F(7,7),R(2,2),NS,NC,NF
      DIMENSION AAAA(49),RR(7),RI(7),ASQR(7,7),ASQ2(7,7),XR(7),XI(7),
     1          VR(7),VI(7),IANA(7),IROW(7,2),VRN(7),VIN(7),W(7,4)
      COMPLEX*16 M,MI,RC
      COMPLEX*16 DCMPLX,DCONJG
      DOUBLE PRECISION AHAT,T,K,G,QHAT,GF,GF2,VW,VGRAD,GRADI
      DOUBLE PRECISION AAAA,RR,RI,ASQR,ASQ2,XR,XI,VR,VI,VRN,VIN,W,VECMGR
     1                   ,VECMGI,VECMGS,SW1
   10 FORMAT (2D15.7)
   30 FORMAT (//T3,'SYSTEM EIGENVALUES')
      CALL VECT(AHAT,AAAA,NS)
      CALL HSBG(NS,AAAA,NS)
      CALL ATEIG(NS,AAAA,RR,RI,IANA,NS)
      WRITE (3,30)
      WRITE (3,10) (RR(I),RI(I),I=1,NS)
      CALL MSQ(AHAT,NS,ASQR)
      NEIG=0
   50 CONTINUE
      DO 100 I=1,NS
      RC(I)=DCMPLX(RR(I),RI(I))
```

```
            DO 100 J=1,NS
      100   ASQ2(I,J)=ASQR(I,J)
            CALL EIGVEC(3,AHAT,ASQR,W,IRCW,XR,XI,VR,VI,RR(1),RI(1),NS,NS,0,
     1           SWI,ITER,DIF,2)
            NEIG=NEIG+1
            IF(NEIG.GT.3) GO TO 105
            IF(ITER.LT.15) GO TO 105
            CALL RAYL(AHAT,RR(1),RI(1),XR,XI,VR,VI,NS,NS)
            GO TO 50
      105   CONTINUE
            VECMGR=0.0D0
            VECMGI=0.0D0
            DO 110 I=1,NS
            VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
      110   VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
            VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
            DO 120 I=1,NS
            VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
            VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
            M(I,1)=DCMPLX(XR(I),XI(I))
      120   MI(1,I)=DCMPLX(VRN(I),VIN(I))
            DO 1000 KOUNT=2,NS
            KOUNT1=KOUNT-1
            IF(RR(KOUNT)-RR(KOUNT1)) 200,140,200
      140   CONTINUE
            DO 150 I=1,NS
            M(I,KOUNT)=DCONJG(M(I,KOUNT1))
      150   MI(KOUNT,I)=DCONJG(MI(KOUNT1,I))
            GO TO 1000
      200   DO 210 I=1,NS
            DO 210 J=1,NS
      210   ASQR(I,J)=ASQ2(I,J)
            CALL EIGVEC(3,AHAT,ASQR,W,IRCW,XR,XI,VR,VI,RR(KOUNT),RI(KOUNT),NS,
     1           NS,0,SWI,ITER,DIF,2)
            VECMGR=0.0D0
            VECMGI=0.0D0
            DO 220 I=1,NS
            VECMGR=VECMGR+VR(I)*XR(I)-VI(I)*XI(I)
      220   VECMGI=VECMGI+VR(I)*XI(I)+VI(I)*XR(I)
            VECMGS=VECMGR*VECMGR+VECMGI*VECMGI
            DO 230 I=1,NS
            VRN(I)=(VR(I)*VECMGR+VI(I)*VECMGI)/VECMGS
            VIN(I)=(VI(I)*VECMGR-VR(I)*VECMGI)/VECMGS
            M(I,KOUNT)=DCMPLX(XR(I),XI(I))
      230   MI(KOUNT,I)=DCMPLX(VRN(I),VIN(I))
     1000   CONTINUE
            RETURN
            END

            SUBROUTINE FEHN(GF)
            COMMON M(7,7),MI(7,7),RC(7),AHAT(7,7),T,K(7,2),G(7,2),QHAT(7,7),VG
     1RADI(4,4),GRADI(4,4),VW(4),B(7,2),F(7,7),R(2,2),NS,NC,NF
            COMPLEX*16 M,MI,RC,CDEXP
            COMPLEX*16 EXS(7,8)
            DOUBLE PRECISION AHAT,T,K,G,QHAT,GF,GF2,VW,VGRAD,GRADI,
     1           FEE(7,7,8),DFEE(7,7,2,2,8),D2FEE(7,7,2,2,2,2,8),FFTI(7,7),
     2           WRK(7,7,8),WRK1(7,7),FDFTI(7,7,2,2),WRK2(7,7,8),WRK3(7,7,8)
     3           ,WRK4(7,7)
            DOUBLE PRECISION FEQTR(7,7),FEHLF(7,7),DFEHLF(7,7,2,2),
     1                   D1MFE(7,7,2,2,2,2),D2MFE(7,7,2,2,2,2)
            CALL TRAPS(0,0,100000,0,0)
```

```
10    FORMAT (//T3,'NECESSARY CONDITIONS VECTOR')
20    FORMAT(4D18.7)
30    FORMAT(//T3,'GRADIENT MATRIX')
62    FORMAT(//T3,'AVERAGE COST =',D20.7)
      NFC=NF*NC
      DO 100 I=1,NS
      DO 100 KT=1,8
100   EX3(I,KT)=CDEXP(RC(I)*T*KT/8.0D0)
      DO 200 I=1,NS
      DO 200 J=1,NS
      FEQTR(I,J)=0.0D0
      FEHLF(I,J)=0.0D0
      DO 190 N1=1,NS
      FEQTR(I,J)=FEQTR(I,J)+M(I,N1)*MI(N1,J)*CDEXP(RC(N1)*T/32.0D0)
190   FEHLF(I,J)=FEHLF(I,J)+M(I,N1)*MI(N1,J)*CDEXP(RC(N1)*T/16.0D0)
      DO 200 KT=1,8
      FEE(I,J,KT)=0.0D0
      DO 200 N1=1,NS
200   FEE(I,J,KT)=FEE(I,J,KT)+M(I,N1)*MI(N1,J)*EX3(N1,KT)
      DO 300 K1=1,NF
      DO 300 K2=1,NC
      DO 300 I=1,NS
      DO 300 J=1,NS
      DFEHLF(I,J,K1,K2)=0.0D0
      DFEE(I,J,K1,K2,1)=0.0D0
      DO 300 N1=1,NS
      DFEHLF(I,J,K1,K2)=DFEHLF(I,J,K1,K2)-FEQTR(I,N1)*B(N1,K2)*FEQTR(K1,J)
     1J)*T/16.0D0
300   DFEE(I,J,K1,K2,1)=DFEE(I,J,K1,K2,1)-FEHLF(I,N1)*B(N1,K2)*FEHLF(K1,J)*T/8
     1J)*T/8.0D0
      DO 400 KT=2,8
      KT1=KT-1
      DO 400 K1=1,NF
      DO 400 K2=1,NC
      DO 400 I=1,NS
      DO 400 J=1,NS
      DFEE(I,J,K1,K2,KT)=0.0D0
      DO 400 N1=1,NS
400   DFEE(I,J,K1,K2,KT)=DFEE(I,J,K1,K2,KT)+FEE(I,N1,1)*DFEE(N1,J,K1,K2,KT1)
     1KT1)+DFEE(I,N1,K1,K2,1)*FEE(N1,J,KT1)
      DO 500 K1=1,NF
      DO 500 K2=1,NC
      DO 500 K3=1,NF
      DO 500 K4=1,NC
      DO 500 I=1,NS
      DO 500 J=1,NS
      D1MFE(I,J,K1,K2,K3,K4)=0.0D0
      D2MFE(I,J,K1,K2,K3,K4)=0.0D0
      DO 450 N1=1,NS
      D1MFE(I,J,K1,K2,K3,K4)=D1MFE(I,J,K1,K2,K3,K4)-
     1      FEHLF(I,N1)*B(N1,K2)*DFEHLF(K1,J,K3,K4)*T/8.0D0
450   D2MFE(I,J,K1,K2,K3,K4)=D2MFE(I,J,K1,K2,K3,K4)-
     1      FEHLF(I,N1)*B(N1,K4)*DFEHLF(K3,J,K1,K2)*T/8.0D0
500   D2FEE(I,J,K1,K2,K3,K4,1)=D1MFE(I,J,K1,K2,K3,K4)+D2MFE(I,J,K1,K2,K3
     1,K4)
      DO 600 KT=2,8
      KT1=KT-1
      DO 600 K1=1,NF
      DO 600 K2=1,NC
      DO 600 K3=1,NF
      DO 600 K4=1,NC
```

```
      DO 600 I=1,NS
      DO 600 J=1,NS
      D2FEE(I,J,K1,K2,K3,K4,KT)=0.0D0
      DO 600 N1=1,NS
  600 D2FEE(I,J,K1,K2,K3,K4,KT)=D2FEE(I,J,K1,K2,K3,K4,KT)+
     1     FEE(I,N1,1)*D2FEE(N1,J,K1,K2,K3,K4,KT1)+D1MFE(I,N1,K1,K2,K3,K
     14)*DFEE(N1,J,K3,K4,KT1)+D2MFE(I,N1,K1,K2,K3,K4)*DFEE(N1,J,K1,K2,KT1)
     11)
      CALL SIMPRD(FLE,FEE,FFTI,T,NS)
      DO 620 I=1,NS
  620 FFTI(I,I)=FFTI(I,I)+T/24.0D0
      DO 700 K1=1,NF
      DO 700 K2=1,NC
      DO 650 I=1,NS
      DO 650 J=1,NS
      DO 650 KT=1,8
  650 WRK(I,J,KT)=DFEE(I,J,K1,K2,KT)
      CALL SIMPRD(FLE,WRK,WRK1,T,NS)
      DO 700 I=1,NS
      DO 700 J=1,NS
  700 FDFTI(I,J,K1,K2)=WRK1(I,J)
      GF=0.0D0
      DO 1000 N1=1,NS
      DO 1000 N2=1,NS
      GF=GF+QHAT(N1,N2)*FFTI(N2,N1)
      DO 1000 N3=1,NS
 1000 GF=GF+F(N1,N2)*FEE(N2,N3,8)*FEE(N1,N3,8)
      GF=GF/NS
      WRITE(3,62) GF
      DO 2100 I=1,NF
      DO 2100 J=1,NC
      IN=NF*(J-1)+I
      VW(IN)=0.0D0
      DO 2100 N1=1,NS
      DO 2000 N2=1,NS
      VW(IN)=VW(IN)+QHAT(N1,N2)*FDFTI(N2,N1,I,J)
      DO 2000 N3=1,NS
 2000 VW(IN)=VW(IN)+F(N1,N2)*FEE(N2,N3,8)*DFEE(N1,N3,I,J,8)
      DO 2100 N4=1,NC
 2100 VW(IN)=VW(IN)+R(J,N4)*K(N1,N4)*FFTI(N1,I)
      WRITE(3,10)
      WRITE(3,20) (VW(I),I=1,NFC)
      DO 3500 I=1,NF
      DO 3500 J=1,NC
      DO 3000 N1=1,NS
      DO 3000 N2=1,NS
      DO 3000 KT=1,8
 3000 WRK(N1,N2,KT)=DFEE(N1,N2,I,J,KT)
      IN=NF*(J-1)+I
      DO 3500 K1=1,NF
      DO 3500 K2=1,NC
      ID=NF*(K2-1)+K1
      VGRAD(IN,ID)=R(J,K2)*FFTI(K1,I)
      DO 3100 N1=1,NS
      DO 3100 N2=1,NS
      DO 3100 KT=1,8
      WRK2(N1,N2,KT)=DFEE(N1,N2,K1,K2,KT)
 3100 WRK3(N1,N2,KT)=D2FEE(N1,N2,I,J,K1,K2,KT)
      CALL SIMPRD(WRK2,WRK,WRK1,T,NS)
      CALL SIMPRD(FLE,WRK3,WRK4,T,NS)
      DO 3500 N1=1,NS
```

```
      DO 3200 N2=1,NS
      VGRAD(IN,ID)=VGRAD(IN,ID)+QHAT(N1,N2)*(WRK1(N2,N1)+WRK4(N2,N1))
      DO 3200 N3=1,NS
 3200 VGRAD(IN,ID)=VGRAD(IN,ID)+F(N1,N2)*(DFEE(N2,N3,K1,K2,8)*DFEE(N1,N3
     1,I,J,8)+FEE(N2,N3,8)*D2FEE(N1,N3,K1,K2,I,J,8))
      DO 3500 N4=1,NC
 3500 VGRAD(IN,ID)=VGRAD(IN,ID)+R(K2,N4)*K(N1,N4)*(FDFTI(N1,K1,I,J)+FDFT
     1I(K1,N1,I,J))+R(J,N4)*K(N1,N4)*(FDFTI(N1,I,K1,K2)+FDFTI(I,N1,K1,K2
     2))
      WRITE(3,30)
      WRITE(3,20) ((VGRAD(I,J),J=1,NFC),I=1,NFC)
      RETURN
      END

      SUBROUTINE GAIN2(GF2)
      COMMON M(7,7),MI(7,7),RC(7),AHAT(7,7),T,K(7,2),G(7,2),QHAT(7,7),VG
     1RAD(4,4),GRADI(4,4),VW(4),B(7,2),F(7,7),R(2,2),NS,NC,NF
      DOUBLE PRECISION AHAT,T,K,G,QHAT,GF,GF2,VW,VGRAD,GRADI
      DOUBLE PRECISION FEE(7,7,8),FFTI(7,7)
      COMPLEX*16 M,MI,RC,CDEXP
      COMPLEX*16 EX3(7,8)
   62 FORMAT(//T3,'AVERAGE COST GF2 =',D20.7)
      DO 100 I=1,NS
      DO 100 KT=1,8
  100 EX3(I,KT)=CDEXP(RC(I)*T*KT/8.0D0)
      DO 200 I=1,NS
      DO 200 J=1,NS
      DO 200 KT=1,8
      FEE(I,J,KT)=0.0D0
      DO 200 N1=1,NS
  200 FEE(I,J,KT)=FEE(I,J,KT)+M(I,N1)*MI(N1,J)*EX3(N1,KT)
      CALL SIMPRD(FEE,FEE,FFTI,T,NS)
      DO 210 I=1,NS
  210 FFTI(I,I)=FFTI(I,I)+T/24.0D0
      GF2=0.0D0
      DO 1000 N1=1,NS
      DO 1000 N2=1,NS
      GF2=GF2+QHAT(N1,N2)*FFTI(N2,N1)
      DO 1000 N3=1,NS
 1000 GF2=GF2+F(N1,N2)*FEE(N2,N3,8)*FEE(N1,N3,8)
      GF2=GF2/NS
      WRITE(3,62) GF2
      RETURN
      END

      SUBROUTINE SIMPRD(A,B,AB,T,N)
      DOUBLE PRECISION A(7,7,8),B(7,7,8),AB(7,7),T
      DO 100 I=1,N
      DO 100 J=1,N
      AB(I,J)=0.0D0
      DO 100 N1=1,N
  100 AB(I,J)=AB(I,J)+(4.0D0*(A(I,N1,1)*B(J,N1,1)+A(I,N1,3)*B(J,N1,3)+A(
     1I,N1,5)*B(J,N1,5)+A(I,N1,7)*B(J,N1,7))+2.0D0*(A(I,N1,2)*B(J,N1,2)+
     2A(I,N1,4)*B(J,N1,4)+A(I,N1,6)*B(J,N1,6))+A(I,N1,8)*B(J,N1,8))*T/
     324.0D0
      RETURN
      END

      SUBROUTINE INVERT(A,B,N)
C
C        INVERTS A TO GIVE B
```

```
C
      DIMENSION A(4,4),B(4,4)
      DOUBLE PRECISION A,B,C,D,X
      DOUBLE PRECISION DABS
      IF(N-1) 100,100,101
  100 B(1,1)=1.0D0/A(1,1)
      RETURN
  101 DO 102 I=1,N
      DO 102 J=1,N
  102 B(I,J)=0.0D0
      DO 103 I=1,N
  103 B(I,I)=1.0D0
C     PICK UP PIVOT ELEMENT
      DO 114 K=1,N
      L=K
      IF(N-K) 110,110,104
  104 I=K+1
      DO 106 JJ=I,N
      IF(DABS(A(JJ,K))-DABS(A(L,K)))106,106,105
  105 L=JJ
  106 CONTINUE
      IF(L-K) 107,110,107
C     PERFORM ROW INTERCHANGE
  107 DO 108 J=K,N
      C=A(K,J)
      A(K,J)=A(L,J)
  108 A(L,J)=C
      DO 109 J=1,N
      C=B(K,J)
      B(K,J)=B(L,J)
  109 B(L,J)=C
C     COLUMN ELIMINATION
  110 DO 114 I=1,N
      IF(K-I) 111,114,111
  111 D=A(I,K)/A(K,K)
      DO 112 J=K,N
  112 A(I,J)=A(I,J)-D*A(K,J)
      A(I,K)=0.0D0
      DO 113 J=1,N
  113 B(I,J)=B(I,J)-D*B(K,J)
  114 CONTINUE
C     SOLVE FOR INVERSE
      DO 115 J=1,N
      DO 115 I=1,N
      X=B(I,J)/A(I,I)
  115 B(I,J)=X
      RETURN
      END

      SUBROUTINE NEWRIT
C
C         PERFORMS NEWTON RAPHSON ITERATION
C
      COMMON M(7,7),MI(7,7),RC(7),AHAT(7,7),T,K(7,2),G(7,2),QHAT(7,7),VG
     1RAD(4,4),GRADI(4,4),VW(4),B(7,2),F(7,7),R(2,2),NS,NC,NF
      COMPLEX*16 M,MI,RC
      DOUBLE PRECISION AHAT,T,K,G,QHAT,GF,GF2,VW,VGRAD,GRADI
      DO 1000 I=1,NF
      DO 1000 J=1,NC
      G(I,J)=K(I,J)
      IN=NF*(J-1)+I
```

```
      DO 1000 K1=1,NF
      DO 1000 K2=1,NC
      ID=NF*(K2-1)+K1
 1000 G(I,J)=G(I,J)-GRACT(IN,ID)*VA(ID)
      RETURN
      END

      SUBROUTINE VECT(AHAT,AAAA,NS)
C
C     CONVERTS AHAT TO SINGLE SUBSCRIPT FORM AAAA
C
      DIMENSION AHAT(7,7),AAAA(49)
      DOUBLE PRECISION AHAT,AAAA
      DO 100 J=1,NS
      DO 100 I=1,NS
      K=(J-1)*NS+I
  100 AAAA(K)=AHAT(I,J)
      RETURN
      END

      SUBROUTINE MSQ(AHAT,NS,ASCR)
C
C     COMPUTES ASCR=AHAT*AHAT
C
      DIMENSION AHAT(7,7),ASCR(7,7)
      DOUBLE PRECISION AHAT,ASCR
      DO 100 I=1,NS
      DO 100 J=1,NS
      ASCR(I,J)=0.0D0
      DO 100 K=1,NS
  100 ASCR(I,J)=ASCR(I,J)+AHAT(I,K)*AHAT(K,J)
      RETURN
      END

      SUBROUTINE RAYL(A,E,EI,X,XI,V,VI,N,MD)
      REAL*8 E,EI,X(MD),XI(MD),V(MD),VI(MD),A(MD,MD),
     1                DVXR,DVXI,DVAXR,DVAXI,A1,A2,A3
      REAL*8 DXDR(12),DXDI(12)
C
C     FOR SYSTEMS OF ORDER HIGHER THAN 12 CHANGE THE FOLLOWING REAL*8
C         STATEMENT
C
      REAL*8 DA(12,12)
C
  800 FORMAT(1X,5D12.4)
      DO 10 I=1,N
      DO 10 J=1,N
   10 DA(I,J)=A(I,J)
      DO 20 I=1,N
   20 DA(I,I)=DA(I,I)-E
      DVXR=0.0
      DO 30 I=1,N
      DVXR=DVXR+V(I)*X(I)
      DXDR(I)=0.0
      DO 30 L=1,N
   30 DXDR(I)=DXDR(I)+DA(I,L)*X(L)
      DVAXR=0.0
      DO 40 I=1,N
   40 DVAXR=DVAXR+V(I)*DXDR(I)
      IF(EI) 60,50,60
   50 E=E+DVAXR/DVXR
```

```
      RETURN
   60 DVXI=0.0
      WRITE(3,800) DVXR
      WRITE(3,800) (DXDR(LL),LL=1,N)
      DO 70 I=1,N
      DVXR=DVXR-VI(I)*XI(I)
   70 DVXI=DVXI+VI(I)*X(I)+V(I)*XI(I)
      WRITE(3,800) DVXR,DVXI
      DO 80 I=1,N
      DXDI(I)=0.0
      DO 80 J=1,N
   80 DXDI(I)=DXDI(I)+DA(I,J)*XI(J)
      WRITE(3,800) (DXDI(LL),LL=1,N)
      A1=0.0
      A2=0.0
      DO 90 I=1,N
      A1=A1+VI(I)*DXDI(I)
      A2=A2+VI(I)*DXDR(I)
      A3=0.0
   90 A3=A3+V(I)*DXDI(I)
      WRITE(3,800) A1,A2,A3
      DVAXR=DVAXR+EI*DVXI-A1
      DVAXI=A2+A3-EI*DVXR
      A1=DVXR*DVXR+DVXI*DVXI
      WRITE(3,800) DVAXR,DVAXI
      WRITE(3,800) A1
      E=E+(DVXR*DVAXR+DVXI*DVAXI)/A1
      EI=EI+(DVAXI*DVXR-DVAXR*DVXI)/A1
      RETURN
      END
      SUBROUTINE HSBG(N,A,IA)
C
C     CONVERTS A TO UPPER HESSENBERG FORM
C
      DIMENSION A(49)
      DOUBLE PRECISION A,PIV,T,S
      DOUBLE PRECISION DABS
      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=DABS(A(IPIV))
      IF(L-3) 90,50,50
   50 M=IPIV-IA
      DO 80 I=L,M,IA
      T=DABS(A(I))
      IF(T-PIV) 80,60,60
   60 IPIV=I
      PIV=T
   80 CONTINUE
   90 IF(PIV) 100,320,100
  100 IF(PIV-DABS(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=I,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)=0.0D0
      320 L=L1
          GO TO 20
      360 RETURN
          END

          SUBROUTINE ATEIG(M,A,RR,RI,IANA,IA)
C
C         COMPUTES ROOTS OF UPPER HESSENBERG MATRIX A
C
          DIMENSION A(40),RR(7),RI(7),PRR(2),PRI(2),IANA(7)
          DOUBLE PRECISION E7,E6,E10,DELTA,PRR,PRI,PAN,PAN1,R,S,T,W,U,V,RR,
         1                 RI,RMOD,EPS,D,G1,G2,G3,CAP,PSI1,PSI2,ALPHA,ETA
          DOUBLE PRECISION DABS,DSQRT,DMAX1
          INTEGER P,P1,Q
          E7=1.0D-8
          E6=1.0D-6
          E10=1.0D-10
          DELTA=0.5D0
          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)=0.0D0
   40 PRI(I)=0.0D0
      PAN=0.0D0
      PAN1=0.0D0
      R=0.0D0
      S=0.0D0
      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=4.0D0*A(N1N)*A(NN1)
      IF(DABS(V)-U*.7) 100,100,65
   65 T=U+V
      IF(DABS(T)-DMAX1(U,DABS(V))*E6) 67,67,68
   67 T=0.0D0
   68 U=(A(N1N1)+A(NN))/2.0D0
      V=DSQRT(DABS(T))/2.0D0
      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)=0.0D0
      RI(N1)=0.0D0
      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*DSQRT(RMOD)
      IF(DABS(A(N1N2))-EPS) 1280,1280,240
  240 IF(DABS(A(NN1))-E10*DABS(A(NN))) 1300,1300,250
  250 IF(DABS(PAN1-A(N1N2))-DABS(A(N1N2))*E6) 1240,1240,260
  260 IF(DABS(PAN-A(NN1))-DABS(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(DABS(RR(K)-PRR(I))+DABS(RI(K)-PRI(I))-DELTA*(DABS(RR(K))
     1   +DABS(RI(K)))) 340,360,360
  340 J=J+I
  360 CONTINUE
      GO TO (440,460,460,430),J
  440 R=0.0D0
      S=0.0D0
```

```
      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(DABS(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(DABS(A(IPI)*A(IPIP+1))*(DABS(A(IPIP)+A(IPIP2+1)-S)+DABS(A(IPIP2
     1  +2)))-DABS(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(DABS(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) =0.0D0
      GO TO 780
  720 G1=A(II1)
      G2=A(II1+1)
      IF(I-Q2)740,740,760
  740 G3=A(II1+2)
      GO TO 780
 760  G3=0.0D0
 780  CAP=DSQRT(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=2.0D0/(1.0D0+PSI1*PSI1+PSI2*PSI2)
      GO TO 880
  860 ALPHA=2.0D0
      PSI1=0.0D0
      PSI2=0.0D0
  880 IF(I-1)900,960,900
  900 IF(I-N)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=1,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(DABS(A(NN1))-DABS(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)=0.0D0
      IANA(N)=1
      IF(N1)1400,1400,1320
 1320 N=N1
```

```
      GO TO 20
 1400 RETURN
      END
      SUBROUTINE EIGVEC(IVC,  A, B, W, IROW, XR, XI, VR, VI, ROOTRE,
     1   ROOTIE, NE, NMAX, T2, SW1, COUNTE, ERR,MMM)
C        SUBROUTINE TO FIND THE EIGENVECTORS OF A NON-SYMMETRIC MATRIX
C        BY A MODIFIED WILKINSON'S INVERSE ITERATION METHOD.
C        CONTROL IVC CODE IS
C              1  FIND ONLY THE REGULAR EIGENVECTORS    (A X = LAMBDA X)
C              2  FIND ONLY THE TRANSPOSED EIGENVECTORS (AT V = LAMBDA V)
C              3  FIND BOTH TYPES OF EIGENVECTORS.
      DIMENSION A(7,7),B(7,7),W(7,4),XR(7),XI(7),VR(7),VI(7),IROW(7,2)
      DOUBLE PRECISION ROOTR,ROOTI,ROOTRE,ROOTIE,TEMP,TEMP2,AMAX,C1,C2,
     1                 SW1,W,XR,XI,VR,VI,B,ZERO,DCERR,A
      DOUBLE PRECISION DABS,DSIGN,DSQRT,DMAX1
      INTEGER COUNT, COUNTE, T2
      I01=1
      I03=3
      ROOTR = ROOTRE
      ROOTI = ROOTIE
      N = NE
      MM = MMM - 1
      N1 = N - 1
      NP1 = N + 1
      IVC1 = IVC - 1
      IVC2 = IVC1 - 1
      COUNT = 1
      DO 400 I=1,N
      W(I,1)=0.0D0
      XR(I)=0.0D0
  400 CONTINUE
      CLIM = 1.0E-4
      IF(ROOTI) 1, 60, 1
C
C        COMPLEX EIGENVALUE.
C
    1 TEMP = - ROOTR - ROOTR
      ISW = 2
      TEMP2=ROOTR*ROOTR+ROOTI*ROOTI
      JJ = 300
      DO 606 I = 1, N
      IF(T2) 600, 603, 600
  600 DO 602 J = 1, N
      JJ = JJ + 1
      IF(JJ - 251) 602, 601, 601
  601 JJ = 1
      READ (T2) (W(LL,1), LL = 1,250)
  602 B(I,J) = A(I,J)*TEMP + W(JJ,1)
      GO TO 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) - ROOTR
      IF(T2 .NE. 0) REWIND T2
      GO TO 700
  607 IF(ICC) 622, 608, 622
C
C        MATRIX SINGULAR.
C
  622 IF(IVC2) 623, 625, 623
```

```
      624 DO 624 LL = 1, N
          W(LL,2)=0.0D0
  624     XI(LL)=0.0D0
          IF(IVC1) 625, 514, 625
      625 DO 626 LL = 1, N
          W(LL,4)=0.0D0
  626     VI(LL)=0.0D0
          GO TO 511
C
C         MATRIX NOT SINGULAR.
C
      608 DO 609 LL = 1, N
          W(LL,1)=1.0D0
          W(LL,2)=1.0D0
          W(LL,3)=1.0D0
  609     W(LL,4)=1.0D0
      699 IF(IVC2) 610, 612, 610
      610 DO 611 I = 1, N
          I2 = IROW(I,2)
          XI(I2) = W(I,1)*ROOTI
          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)*ROOTI
          DO 613 J = 1,N
      613 VI(I) = VI(I) + A(J,I)*W(J,4)
          GO TO 499
      615 CERR = 0.0
          DCERR=0.0D0
          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)/ROOTI
          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)/ROOTI
C
C         SEARCH VECTORS FOR LARGEST ELEMENT AND NORMALIZE.
C
  627     AMAX=0.0D0
          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(COUNT .EQ. 1) GO TO 632
          DO 631 LL = 1, N
  631     DCERR=DMAX1(DCERR,DABS(VR(LL)-W(LL,3)),DABS(VI(LL)-W(LL,4)))
```

```
  632 IF(IVC2) 633, 638, 633
  633  AMAX=0.0D0
       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(COUNT .EQ. 1) GO TO 646
       DO 637 LL = 1, N
  637  DCERR=DMAX1(DCERR,DABS(XR(LL)-W(LL,1)),DABS(XI(LL)-W(LL,2)))
C
C        TEST FOR CONVERGENCE.
C
  638 IF(COUNT .EQ. 1) GO TO 646
       CERR=DCERR
       IF(CERR .GE. 1.0E-4) GO TO 639
       IF(CERR .GE. CLIM) GO TO 648
       CLIM = CERR
       IF(CLIM .LE. 1.0E-8) GO TO 648
  639 IF(COUNT .GE. 15) GO TO 68
  647 COUNT = COUNT + 1
       IF(ROOTI) 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 TO 699
  646 CERR = 0.0
       DCERR=0.0D0
       IF(ICC) 648, 647, 648
  648 ERR = CERR
       COUNTE = COUNT
       IF(ROOTI) 667, 668, 667
  667 DO 649 I = 1, N
  649 A(I,I) = A(I,I) + ROOTR
       RETURN
   68 PRINT 101, ROOTR, ROOTI, CERR
       GO TO 648
C
C        REAL EIGENVECTORS.
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) - ROOTR
       GO TO 700
  652 IF(ICC) 680, 685, 680
C
C        SINGULAR MATRIX.
```

```
C                                                                       ESY11630
  680 IF(IVC2) 681, 683, 681                                            ESY11640
  681 DO 682 L = 1, N                                                   ESY11650
  682  XI(L)=0.0D0
       IF(IVC1) 683, 514, 683                                           ESY11670
  683 DO 684 L = 1, N                                                   ESY11680
  684  VI(L)=0.0D0
       GO TO 511                                                        ESY11700
C                                                                       ESY11710
C          MATRIX NOT SINGULAR.                                         ESY11720
C                                                                       ESY11730
  685 IF(IVC2) 653, 656, 653                                            ESY11740
  653 DO 654 L = 1, N                                                   ESY11750
  654  XI(L)=1.0D0
       IF(IVC1) 656, 500, 656                                           ESY11770
  656 DO 657 L = 1, N                                                   ESY11780
  657  VI(L)=1.0D0
       GO TO 499                                                        ESY11800
C                                                                       ESY11810
C          NORMALIZE REAL VECTORS.                                      ESY11820
C                                                                       ESY11830
  655 CERR = 0.0                                                        ESY11840
       DCERR=0.0D0
       IF(IVC2) 658, 662, 658                                           ESY11850
  658  C1=0.0D0
       C2=0.0D0
       DO 660 L = 1, N                                                  ESY11870
       TEMP=DABS(XI(L))
       IF(TEMP - C1) 660, 660, 659                                      ESY11890
  659 C1 = TEMP                                                         ESY11900
       C2 = XI(L)                                                       ESY11910
  660 CONTINUE                                                          ESY11920
       DO 661 L = 1, N                                                  ESY11930
       XI(L) = XI(L)/C2                                                 ESY11940
       DCERR=DMAX1(DCERR,DABS(XI(L)-XR(L)))
  661 XR(L) = XI(L)                                                     ESY11960
       IF(IVC1) 662, 638, 662                                           ESY11970
  662  C2=0.0D0
       C1=0.0D0
       DO 664 L = 1, N                                                  ESY11990
       TEMP=DABS(VI(L))
       IF(TEMP - C1) 664, 664, 663                                      ESY12010
  663 C1 = TEMP                                                         ESY12020
       C2 = VI(L)                                                       ESY12030
  664 CONTINUE                                                          ESY12040
       DO 665 LL = 1, N                                                 ESY12050
       VI(LL) = VI(LL)/C2                                               ESY12060
       DCERR=DMAX1(DCERR,DABS(VI(LL)-W(LL,1)))
       W(LL,1)=VI(LL)
  665 VR(LL)=W(LL,1)
       GO TO 638                                                        ESY12090
  668 IF(IVC2) 669, 671, 669                                            ESY12100
  669 DO 670 L = 1, N                                                   ESY12110
  670  XI(L)=0.0D0
       IF(IVC1) 671, 70, 671                                            ESY12130
  671 DO 672 L = 1, N                                                   ESY12140
  672  VI(L)=0.0D0
   70 RETURN                                                            ESY12160
  673 IF(IVC2) 674, 502, 674                                            ESY12170
  674 DO 675 I = 1, N                                                   ESY12180
       I2 = IROW(I,2)                                                   ESY12190
```

```
  675 XI(I2) = XR(I)                                    ESY12200
      GO TO 500                                         ESY12210
C                                                       ESY12220
C          BACK SUBSTITUTION SECTION.                   ESY12230
C                                                       ESY12240
  499 IF(IVC2) 500, 502, 500                            ESY12250
  500 DO 501 I = 2, N                                   ESY12260
      I1 = I - 1                                        ESY12270
      DO 501 J = 1, I1                                  ESY12280
  501 XI(I) = XI(I) - B(I,J)*XI(J)                      ESY12290
  511 IF(IVC1) 502, 514, 502                            ESY12300
  502 DO 510 I = 1, N                                   ESY12310
      I1 = I - 1                                        ESY12320
      IF(I1) 503, 505, 503                              ESY12330
  503 DO 504 J = 1, I1                                  ESY12340
  504 VI(I) = VI(I) - B(J,I)*VI(J)                      ESY12350
      IF(ICC) 505, 506, 505                             ESY12360
  505 IF(B(I,I)) 506, 507, 506                          ESY12370
  506 VI(I) = VI(I)/B(I,I)                              ESY12380
      GO TO 510                                         ESY12390
  507 IF(VI(I)) 508, 509, 508                           ESY12400
  508  VI(I)=VI(I)*1,0D+15
      GO TO 510                                         ESY12420
  509  VI(I)=1.0D0
  510 CONTINUE                                          ESY12440
      IF(IVC2) 514, 525, 514                            ESY12450
  514 DO 522 I = 1, N                                   ESY12460
      IR = NP1 - I                                      ESY12470
      IF(I - 1) 515, 517, 515                           ESY12480
  515 I2 = IR + 1                                       ESY12490
      DO 516 J = I2, N                                  ESY12500
  516 XI(IR) = XI(IR) - B(IR,J)*XI(J)                   ESY12510
      IF(ICC) 517, 518, 517                             ESY12520
  517 IF(B(IR,IR)) 518, 519, 518                        ESY12530
  518 XI(IR) = XI(IR)/B(IR,IR)                          ESY12540
      GO TO 522                                         ESY12550
  519 IF(XI(IR)) 520, 521, 520                          ESY12560
  520  XI(IR)=XI(IR)*1.0D+15
      GO TO 522                                         ESY12580
  521  XI(IR)=1.0DC
  522 CONTINUE                                          ESY12600
      IF(IVC1) 525, 529, 525                            ESY12610
  525 DO 526 I = 2, N                                   ESY12620
      IR = NP1 - I                                      ESY12630
      I2 = IR + 1                                       ESY12640
      DO 526 J = I2, N                                  ESY12650
  526 VI(IR) = VI(IR) - B(J,IR)*VI(J)                   ESY12660
      DO 527 L = 1, N                                   ESY12670
      I2 = IROW(L,1)                                    ESY12680
  527 VR(I2) = VI(L)                                    ESY12690
      DO 528 L = 1, N                                   ESY12700
  528 VI(L) = VR(L)                                     ESY12710
  529 IF(RCOTI) 615, 655, 615                           ESY12720
C                                                       ESY12730
C          FACTOR MATRIX.                               ESY12740
C                                                       ESY12750
  700 ICC = 0                                           ESY12760
      SWI=1.0D72
      DO 701 LL = 1, N                                  ESY12780
  701 IROW(LL,1) = LL                                   ESY12790
      DO 708 K = 1, N1                                  ESY12800
```

```
      AMAX=DABS(B(K,K))
      IMAX = K
      K1 = K + 1
      DO 702 I = K1, N
      IF(AMAX.GT.DABS(B(I,K))) GO TO 702
      AMAX=DABS(B(I,K))
      IMAX = I
  702 CONTINUE
      IF(AMAX .LT. SW1) SW1 = AMAX
      IF(AMAX.GE.1.0D-25) GO TO 723
      B(K,K)=0.0DC
      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=DABS(B(N,N))
      IF(AMAX-1.0D-25) 712,712,713
  712 B(N,N)=0.0DC
      SW1=0.0D0
      ICC = ICC + 1
      GO TO 709
  713 IF(AMAX .LT. SW1) SW1 = AMAX
  709 IF(ICC .LE. ISW) GO TO 710
      IF(MM) 1050,1050,1051
 1050 WRITE(IO3,1C2) ICC
      COUNTE = 0
      RETURN
 1051 WRITE(IO3,1C52) ICC
  710 DO 711 LL = 1, N
      I2 = IROW (LL,1)
  711 IROW(I2,2) = LL
      IF(ROOTI) 6C7, 652, 607
 1052 FORMAT(///23H ****** WARNING ******      ,'   SUBROUTINE EIGVEC HAS
     1FOUND AN EIGENVALUE OF APPARENT MULTIPLICITY',
     1                                I4,/23X,'   COMPUTATION OF EI
     2GENVECTOR(S) CONTINUES AT USER S OPTION'//)
  101 FORMAT(38H0MORE THAN 15 LOOPS FOR EIGENVECTOR OF,2E12.4,
     2   14H DIFFERENCE OF,E12.4)
  102 FORMAT(16H0****WARNING**** , I4, 71H ZEROS ON DIAGONAL OF FACTORED
     1 MATRIX,  CHECK FOR MULTIPLE EIGENVALUES./20X,
     2' SUBROUTINE EIGVEC WILL NOT PERFORM COMPUTATION FOR THIS EIGENVEC
     3TOR '//)
      END

/DATA
```