COMPUTER PROGRAMS DEVELOPED TO SOLVE THE EQUATIONS OF MOTION FOR DRILL DYNAMICS

by

Ozan Tekinalp
Graduate Student

Thesis Chairman: A. G. Ulsoy
Associate Professor

Report No: UM-MEAM-88-02

Department of Mechanical Engineering and Applied Mechanics
The University of Michigan
Ann Arbor, MI 48109-2125

May, 1988

UMR0415

TABLE OF CONTENTS

INTRODUCTION 1

DESCRIPTIONS OF THE PROGRAMS 1

SAMPLE SESSION 2

ACKNOWLEDGEMENTS 6

REFERENCES 6

APPENDIX A

Listings of Finite Element Programs to Solve Equations of Motion

- MAIN 1
- READ3 5
- READ2 8
- DISCRE 9
- ELEMEN 9
- ASEMBO 17
- EIGENS 17
- STATE1 18
- GBOUND 20
- KNCOFF 21
- TIMER 22
- STATE2 24
- READ4 25
- DIFFUN 25
- PEDERV 26
- DRIVE 27

APPENDIX B

Listings of Programs to Calculate Drill Cross Sectional Properties:

- MAIN 1
- READ1 2
- BOUNEQ 3
- DIVL1 4
- DIVL2 5
- DIVL3 5
- DIVL4 6
- MESH 7
- SOR 8
- PLOT1 10
- CONNEC 11
- ASSIGN 12
- PNUM 13
- GAUSS 14
- FUNCTN 15
- PRNCPL 15

INTRODUCTION

The drill dynamics is investigated by a model developed. The results are reported in the thesis submitted as a partial fulfillment of the requirement towards a Ph.D. degeree at the University of Michigan Mechanical Engineering Department [1]. This report complements the thesis, and gives the description, and listings of the programs used to solve the drill dynamics equations.

DESCRIPTIONS OF THE PROGRAMS

Two separate programs are developed to analyze the equations of motion of drill dynamics.

The first program uses the finite element method developed to descritize the equations of motion which are partial differential equations with constant coefficients. The flowchart for this program is given in Fig. 1. Standard beam element shape functions are employed for decritization. The program then utilizes NAAS subroutines for eigenanalysis as well as time response solutions [2]. The program in the most part is interactive. The initial inputs must be read from a file.However, at the end of each run, the user may rerun the program, change the boundary conditions, or various drill parameters interactively. In case of time response, the response duration, cutting forces at the drill tip, and the filters' time constant that filters some of the noise from the random number generater, is input interactively. The NAAS routines are not included in the listings (Appendix A). However the NAAS routine DRIVE is included to show the changes necessary to the "COMMON" blocks in the subroutine, that will accomodate the large dimensions necessary for finite element time response solution. The program is currently set up to solve at most twenty

finite element case. The parameter declerations in the main program, as well as in subroutines, must be changed to accomodate more elements.

The second programs flow chart is given in Fig. 2 and the listings are given in Appendix B. This program first inputs the various points on the drill corss section (see Thesis, Chapter 5). Then calculates the boundary geometry, and the boundary mesh. This boundary mesh is than used to descritize the cross section. The internal mesh generated is saved in a plot file that must be supplied during a run (unit 9). The mesh plot can be viewed by running the OLD:PLOTSEE program on the MTS. The inner mesh generated is then employed to calculate the cross sectional area, and area moments of inertia using three node Gaussian integration scheme.

SAMPLE SESSION

The following is a sample session for eigenanalysis. The compiled and linked program name is "otneigenload". Input data file data file and output data file listings are also given.

```
#r otneigenload 5=otreadf2 6=-61 7=-71
#Execution begins   17:37:34
 WOULD YOU LIKE TO SOLVE THE EIGENPROBLEM OR TIME RESPONSE
 FOR EIGENSOLUTION, ENTER "E"
 FOR TIME RESPONSE, ENTER "T"
 e
 TL=.2,
 NL=5,
 WOULD YOU LIKE TO RERUN WITH DIFFRENT PARAMETERS
 y
 CURRENT BOUNDAY CONDITION IS: CLAMPED-PINNED
 WOULD YOU LIKE TO CHANGE THE BOUNDAY CONDITIONS? (Y/N):
 n
THE CURRENT OPERATING PARAMETERS ARE:
 THRUST FORCE ====>  1730.,  NEWTONS
 ROTATIONAL SPEED =====>62.8,  RADIANS/SECOND
 WOULD YOU LIKE TO CHANGE OPERATING PARAMETERS? (Y/N):
 y
ENTER THE NEW THRUST FORCE :
 1000.
```

```
THE NEW THRUST FORCE IS :1000.,  NEWTONS
 IS THE VALUE CORRECT?(Y/N):
 y
ENTER THE NEW ROTATIONAL SPEED :
 0.0
THE NEW ROTATIONAL SPEED IS:0., RAD/S
 IS THE VALUE CORRECT?(Y/N):
 y
THE CURRENT DRILL SPECS ARE:
 AREA ====>  .3410855000000000E-04,  SQUARE METERS
 PRINCIPAL MOMENT OF INERTIAS ARE
 ====>  .5861930000000000E-10,  METERS!4
 ====>  .2952032000000000E-09,  METERS!4
 HELIX ANGLE IS ====>  47.,  RADIANS
 FREE LENGTH OF DRILL BIT IS ====> .2,  METERS
 DO YOU LIKE TO CHANGE THESE SPECIFICATIONS?(Y/N)
 n
PROGRAM IS BEING RERUN WITH NEW:
 OPERATING PARAMETERS
 WOULD YOU LIKE TO RERUN WITH DIFFRENT PARAMETERS
 n
EXITING THE PROGRAM
#T=0.236
#   $.16,    $.41T
#1 -61

      1     AREA= 0.34108550E-04
      2     S1= 0.58619300E-10
      3     S2= 0.29520320E-09
      4      NL=          5
      5     ****** OMEGA ******      62.80000000
      6     ******  ELAS ******    0.2070000000E+12
      7     ****** BETA  ******      47.00000000
      8     ******  RHO  ******      7700.000000
      9     ******   FZ  ******      1730.000000
     10     ******   TL  ******    0.2000000000
     11     ***** U1=  0      ***** TETHA1=  5
     12     ***** V1=  0      ***** FHI1=  5
     13     ***** U2=  0      ***** TETHA2=  0
     14     ***** V2=  0      ***** FHI2=  0
     15
     16     READ2 COMPLETE
     17
     18
     19     DISCRE COMPLETE
     20
     21
     22     ******* EL=       0.40000000E-01
     23     ******UMASS=       0.26263583
     24     IN ELMEN    DUMK4=    -6587.067603
     25
     26     ELEMEN COMPLETE
     27
     28
     29     ASEMBO COMPLETE
     30
     31
     32     **************  N1==    104**************  MDIM
```

```
***** U1=  0      ***** TETHA1=  5
***** V1=  0      ***** FHI1=  5
***** U2=  0      ***** TETHA2=  0
***** V2=  0      ***** FHI2=  0
ROW COLOUMN    1IS REMOVED
ROW COLOUMN    2IS REMOVED
ROW COLOUMN   19IS REMOVED
ROW COLOUMN   19IS REMOVED
ROW COLOUMN   19IS REMOVED
ROW COLOUMN   19IS REMOVED
UDU DECOMPOSITION IS COMPLETE,
TMINV IS INVERTED
STATE1 IS COMPLETE

 RSG COMPLETE
**2*MDIM******* EIGENVALUES **************
    1     (          888.951    ) i
    2     (          888.951    ) i
    3     (          991.569    ) i
    4     (          991.569    ) i
    5     (          2490.69    ) i
    6     (          2490.69    ) i
    7     (          3117.81    ) i
    8     (          3117.81    ) i
    9     (          5939.31    ) i
   10     (          5939.31    ) i
   11     (          7394.85    ) i
   12     (          7394.85    ) i
   13     (          10850.9    ) i
   14     (          10850.9    ) i
   15     (          12127.6    ) i
   16     (          12127.6    ) i
   17     (          17937.2    ) i
   18     (          17937.2    ) i
   19     (          18978.6    ) i
   20     (          18978.6    ) i
   21     (          25756.2    ) i
   22     (          25756.2    ) i
   23     (          28920.0    ) i
   24     (          28920.0    ) i
   25     (          36029.0    ) i
   26     (          36029.0    ) i
   27     (          40408.4    ) i
   28     (          40408.4    ) i
   29     (          48977.5    ) i
   30     (          48977.5    ) i
   31     (          52725.7    ) i
   32     (          52725.7    ) i
   33     (          66302.7    ) i
   34     (          66302.7    ) i
   35     (          81110.8    ) i
   36     (          81110.8    ) i
****** IER***** =    0
IN ELMEN    DUMK4=    -3807.553527

ELEMEN COMPLETE
```

```
 ASEMBO COMPLETE


 ************** N1==   104************** MDIM==   18
 ***** U1=  0     ***** TETHA1=  5
 ***** V1=  0     ***** FHI1=  5
 ***** U2=  0     ***** TETHA2=  0
 ***** V2=  0     ***** FHI2=  0
 ROW COLOUMN    1IS REMOVED
 ROW COLOUMN    2IS REMOVED
 ROW COLOUMN   13IS REMOVED
 ROW COLOUMN   13IS REMOVED
 ROW COLOUMN   13IS REMOVED
 ROW COLOUMN   13IS REMOVED
 UDU DECOMPOSITION IS COMPLETE,
 TMINV IS INVERTED
 STATE1 IS COMPLETE

  RSG COMPLETE
 **2*MDIM******* EIGENVALUES *************
     1     (         1983.74    ) i
     2     (         1983.74    ) i
     3     (         2556.10    ) i
     4     (         2556.10    ) i
     5     (         4947.46    ) i
     6     (         4947.46    ) i
     7     (         9755.91    ) i
     8     (         9755.91    ) i
     9     (         11688.4    ) i
    10     (         11688.4    ) i
    11     (         17709.0    ) i
    12     (         17709.0    ) i
    13     (         21476.6    ) i
    14     (         21476.6    ) i
    15     (         29917.4    ) i
    16     (         29917.4    ) i
    17     (         34987.8    ) i
    18     (         34987.8    ) i
    19     (         45937.1    ) i
    20     (         45937.1    ) i
    21     (         52724.7    ) i
    22     (         52724.7    ) i
    23     (         76669.7    ) i
    24     (         76669.7    ) i
 ****** IER***** =    0
#   $.03,    $.44T
#1 otreadf2
      1     0.34108550D-4,0.586193D-10,0.2952032D-9,5,
      2     62.8D0,207.D9,47.D0,7700.D0,1730.d0,0.2D0,
      3     0,5,0,5,0,0,0,0,
```

ACKNOWLEDGEMENTS

I would like to thank Mr. Dong for supplying the automatic mesh generation routines. These routines are: ASSIGN, SOR, PLOT1, CONNEC, PNUM.

REFERENCES

[1] Tekinlap, O., Dynamic Modeling of Drill Bit Vibrations, Ph.D. Dissertation, Department of Mechanical Engineering and Applied Mechanics, The University of Michigan Library, Ann Arbor, MI 48109 , June 1988.

[2] Harding, L. J., Numerical Analysis Applications Software Abstracts, The University of Michigan Computing Center, 4th ed., September 1979.

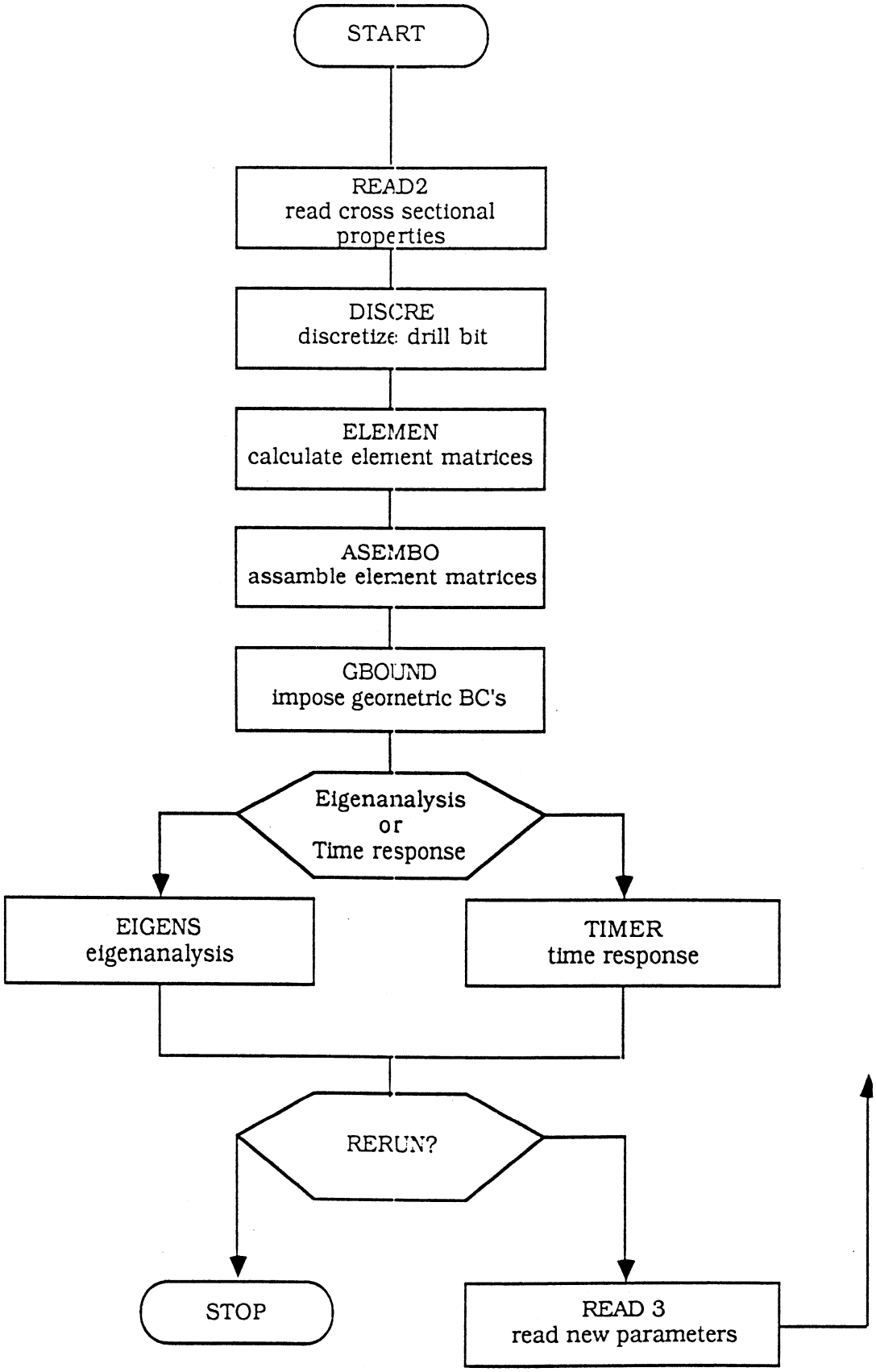

Fig. 1 Flow chart for the finite element program.

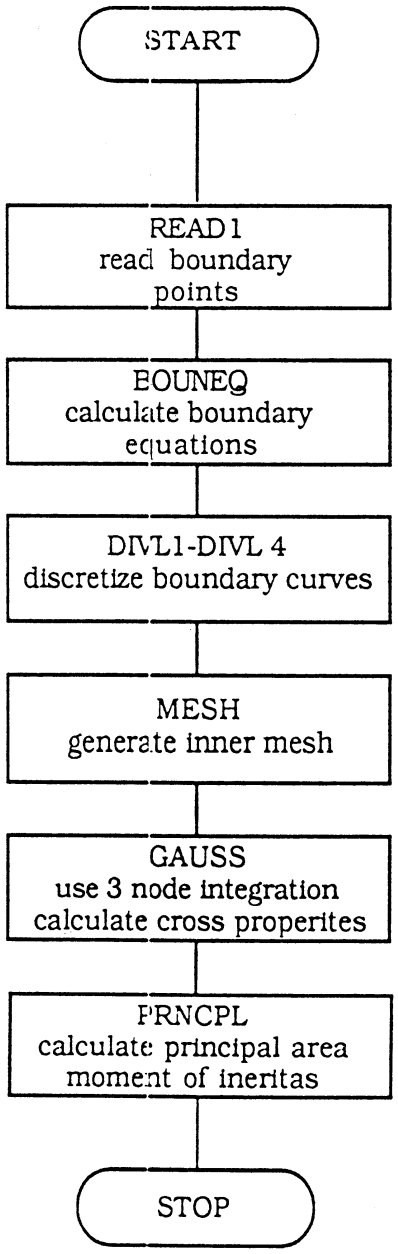

Fig. 2 Flow chart for the automatic mesh generation program

APPENDIX A

PROGRAM LISTING FOR FINITE ELEMENT ROUTINES TO ANALYZE THE DRILL DYNAMICS

```
C/* filename: new7 11/30/87 */
C
C               ************************************************
C               * THIS PROGRAM SOLVES THE DRILL BIT EQUATIONS   *
C               * THE SOLUTION CAN BE AN EIGENVALUE PROBLEM     *
C               * OR TIME RESPONSE                              *
C               ************************************************
C
C                       ASSOCIATED SUBROUTINES----------
C
C***        READ2(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL,
C     &           U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2)= Reads data
C
C***        DISCRE(TL,NL,AREA,RHO,EL,UMASS)= Discretizes drill bit
C
C***        ELEMEN(S1,S2,EL,BETA,UMASS,FZ,OMEGA,ELAS,SME,SCE,SKE)=
C                                     Calculates element matrices
C
C***        ASEMBO(SME,SCE,SKE,NL,N1,MDIM,TM,TC,TK)= Assembles element
C                                     matrices to a global one
C
C***        SUBROUTINE GBOUND(TM,TC,N1,MDIM,U1,TETHA1,V1,FHI1,
C     &      U2,TETHA2,V2,FHI2) = Imposes gemetric boundary conditions
C                       to global TM,TC,TK
C
C
C***    EIGENS(N1,TKTOT,TMTOT,W,Z,SV1,SV2) = Solves the generalized
C               symmetric eigenvalue problem of TKTOT & TMTOT
C
C***    TIMER() = calculates the time response of the system using
C               TKTOT and input forcing function.
C
C***    READ3(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL
C               ,U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2,FLAG2,FLAG3,FLAG4)=
C       prompts the user for changing the parameters to rerun
C       the program.
C
C                       ARGUMENTS----------
C               AREA= Cross-sectional area
C               S1,S2= Area moment of inertias in principal dirctions
C               OMEGA= Drill rotational speed
C               ELAS= Young's modulus of elasticity
C               BETA= Drill helix angle
C               RHO= Material density
C               FZ= Axial force
C               TL= Total free length of the drill bit
C               NL= Number of beam elements
C               EL= Length of the beam element
```

```
C               UMASS= Mass/unit lenth
C               XL= Coordinates of nodes
C               SME,SCE,SKE= Element mass, damping, stiffness matrices
C               TM,TC,TK= Global mass, damping, stiffness matrices
C               MDIM= Dimension of global matrices
C                     --------------------
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
      CHARACTER*1 FLAG1,FLAG2,FLAG3,FLAG4,FLAG5,ALPHA
C
C      Parametrs below are the dimension of the largest
C      system that can be solved.  This translates into a
C      20 elemet system. i.e. N1=MDIM= 4*(20+1) = 84
C      to increase the allowable system change N1
      PARAMETER(N1=84)
C
      DIMENSION SME(8,8), SCE(8,8), SKE(8,8)
      DIMENSION TM(2*N1+1,2*N1),TC(N1,N1),TK(2*N1+1,2*N1+1)
      DIMENSION TMINV(N1,N1),SV1(1),IV(N1),INERT(3)
      DIMENSION TY(1),TWORK(1)
      DIMENSION EW(1),ESV1(1),ESV2(1)
      COMMON /A/TKTOT(2*N1+1,2*N1+1)
      COMMON /C/INIT,M2,TIMEC,F(2*N1+1)
C      ::: note DUMMY is declared in blank COMMON ::::::::::::
C      ::::::hence dimentions are adjustable:::::::::
      COMMON DUMMY(6*N1+1)
C
      EQUIVALENCE (TK(1,1), TKTOT(1,1)),(DUMMY(1),SV1(1))
C
C      :::: following are for subroutine timer ::::::::
C      EQUIVALENCE (DUMMY(1),TY(1)),(DUMMY(2*N1+1),TWORK(1))
C
C      :::: following are for subroutine eigens :::::::
      EQUIVALENCE (EW(1),DUMMY(1)),(ESV1(1),DUMMY(2*N1+1))
     &          ,(DUMMY(4*N1+1),ESV2(1))
C
C
C      ================ initialize the flags ================
      FLAG1 = 'N'
      FLAG2 = 'N'
      FLAG3 = 'N'
      FLAG4 = 'N'
      FLAG5 = 'N'
C
C      ::::::::::::::::::::::::::::::::::::::::::::::::::::::
C      ====== Read the input values =====
      CALL READ2(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL,
     &          U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2)
  160  WRITE(6,150)
  150  FORMAT(/,'READ2 COMPLETE',/)
C
      PRINT*,'WOULD YOU LIKE TO SOLVE THE EIGENPROBLEM OR',
     &       ' TIME RESPONSE'
            PRINT*,'FOR EIGENSOLUTION, ENTER "E"'
            PRINT*,'FOR TIME RESPONSE, ENTER "T"'
      READ(*,'(A1)',END=1800) ALPHA
      IF ((ALPHA .EQ. 'T') .OR. (ALPHA .EQ. 't')) THEN
      ELSE IF ((ALPHA .EQ. 'E') .OR. (ALPHA .EQ. 'e')) THEN
```

```
        ELSE
                PRINT*,'PLEASE REENTER'
                GOTO 160
        ENDIF
C
        MDIM=(NL+1)*4
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C       ================ discretize the drill bit ================
        CALL DISCRE(TL,NL,AREA,RHO,EL,UMASS)
C
         PRINT*,'TL=',TL
         PRINT*,'NL=',NL
         WRITE(6,151)
 151     FORMAT(/,'DISCRE COMPLETE',/)
        WRITE(6,101) EL,UMASS
 101     FORMAT(/,'******* EL=',G 20.8
     &           ,/,'******UMASS=',G20.8)
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C      ========== operating or drill parameters are changed =======
C      ===========  entry point for GOTO 5000 ==================
 5000   CONTINUE
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        ==================== calculate the element matrices ==========
C
         CALL ELEMEN(S1,S2,EL,BETA,UMASS,FZ,OMEGA,
     &                     ELAS,SME,SCE,SKE)
C
         WRITE(6,152)
  152    FORMAT(/,'ELEMEN COMPLETE',/)
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C       ==== boundary conditions are changed FLAG2=YES  ====
C       ===== reassamble and reimpose the geometric boundary
conditions===
C       === entry point for: GOTO 6000 =====
 6000   CONTINUE
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        ======= assemble mass stiffness and coriolis matrices =====
C        ======================== to TM,TK,TC ======================
        CALL ASEMBO(SME,SCE,SKE,NL,N1,MDIM,TM,TC,TK)
C
         WRITE(6,155)
 155     FORMAT(/,'ASEMBO COMPLETE',/)
         WRITE(6,156) N1,MDIM
 156     FORMAT(/,'**************  N1==  ',I10,
     &            '**************  MDIM==',I10)
C
C
        IF((ALPHA .EQ. 'T') .OR. (ALPHA .EQ. 't')) THEN
           PRINT*,'TIME RESPONSE IS SOLVED FOR A CANTILEVER BEAM'
           U1=5
           V1=5
           U2=0
           V2=0
```

```
            TETHA1=5
            FHI1=5
            TETHA2=0
            FHI2=0
         ENDIF
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        ==============   impose the geometric boundary conditions
==============
         CALL GBOUND(TM,TC,TK,N1,MDIM,U1,TETHA1,V1,FHI1,
     &       U2,TETHA2,V2,FHI2)
C
C
C
C        ============invert the symmetric matrix TM ===========
C        :::::::::::::: TMINV = TM ::::::::::::::::::::
         DO 77 I=1,MDIM
         DO 77 J=1,MDIM
         TMINV(I,J) = TM(I,J)
 77      CONTINUE
C
C
C        ::::::::::::::::::::invert TMINV ::::::::::::::::::::
C        ::::::::::::::compute UDU decomposition of TMINV::::::::::
C
         CALL DSICO (TMINV,N1, MDIM, IV, RC, SV1)
         WRITE(6,73)
 73      FORMAT('UDU DECOMPOSITION IS COMPLETE,/,')
         IF(RC .EQ. 1.D0) PRINT*,'WARNING!!!  TM IS ILL CONDITIONED'
C
C        ::::::::::::invert UDU decomposition of TMINV::::::::::::::
         CALL DSIDI (TMINV, N1, MDIM, IV, DET, INERT, SV1, 1)
         WRITE(6,74)
 74      FORMAT('TMINV IS INVERTED')
C        :::::::::::::: make TMINV a full symmmetric matrix ::::::::::::::
C        :::::: only upper triangular part is returned by DSIDI ::::::
         DO 72 I=2, MDIM
          I1=I-1
         DO 72 J=1,I1
         TMINV(I,J) = TMINV(J,I)
 72      CONTINUE
C
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        ========= eigensolution or time response?? ==================
C
         IF((ALPHA .EQ. 'T') .OR. (ALPHA .EQ. 't')) THEN
C
C
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        ================== calculate time response of the system ======
                 CALL TIMER(N1,MDIM,TC,TK,TMINV,UMASS)
         ELSE
C
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        ========== calculate the eigenvalues of the system ========
                 CALL EIGENS(N1,MDIM,TM,TC,TK,TMINV,EW,ESV1,ESV2)
         ENDIF
C
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
C       ============== promt to rerun the program ==============
C       ================  reset the flags   ====================
        FLAG1 = 'N'
        FLAG2 = 'N'
        FLAG3 = 'N'
        FLAG4 = 'N'
        FLAG5 = 'N'
C
        WRITE(*,*) 'WOULD YOU LIKE TO RERUN WITH DIFFRENT PARAMETERS'
        READ(*,'(A1)',END=1800) FLAG1
        IF((FLAG1.EQ.'N') .OR. (FLAG1 .EQ. 'n')) GOTO 1800
C
        CALL READ3(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL
     &  ,U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2,FLAG2,FLAG3,FLAG4)
        IF (((FLAG3 .EQ. 'Y') .OR. (FLAG3 .EQ. 'y')) .OR.
     &  ((FLAG4 .EQ. 'Y') .OR. (FLAG4 .EQ. 'y'))) GOTO 5000
        IF ((FLAG2 .EQ. 'Y') .OR. (FLAG2 .EQ. 'y')) GOTO 6000
 1800   WRITE(*,*) 'EXITING THE PROGRAM'
        STOP
        END
C       ------------------------------------------------
C       ------------------------------------------------
C
C
        SUBROUTINE READ3(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL
     &  ,U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2,FLAG2,FLAG3,FLAG4)
C
       IMPLICIT REAL*8 (A-H,O-Z)
        INTEGER*2 U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
       CHARACTER*1 FLAG1,FLAG2,FLAG3,FLAG4
C
C       ========== change the bounday conditions ==============
        PRINT*, 'CURRENT BOUNDAY CONDITION IS:'
        IF(((U1 .EQ. 0) .AND. (U2 .EQ. 0))) THEN
                IF((TETHA1 .EQ. 0) .AND. (TETHA2 .EQ. 0)) THEN
                 PRINT*, '=======>    CAMPED-CLAMPED'
                ELSE IF((TETHA1 .NE. 0) .AND. (TETHA2 .NE. 0)) THEN
                 PRINT*, '=======>    PIN-PIN'
                ELSE IF(((TETHA1 .NE.0) .AND. (TEHA2 .EQ.0)) .OR.
     &                  ((TETHA1 .EQ. 0) .AND. (TETHA2 .NE. 0))) THEN
                 PRINT*, '=======>    CLAMPED-PINNED'
                END IF
        ELSE IF(((U1 .EQ. 0) .AND. (U2 .NE. 0)) .OR.
     &          ((U1 .NE. 0) .OR. (U2 .EQ. 0))) THEN
                 PRINT*, '=======>    CANTILEVER'
        END IF
C
        PRINT*,'WOULD YOU LIKE TO CHANGE THE BOUNDAY CONDITIONS? (Y/N):'
        READ(*,'(A1)',END=100) FLAG2
        IF ((FLAG2 .EQ. 'Y') .OR. (FLAG2 .EQ. 'y')) THEN
 5            PRINT*, 'ENTER ::'
              PRINT*, '1       FOR CALMPED-CLAMPED BOUNCARY CONDITION'
              PRINT*, '2       FOR PIN-PIN BOUNDAY CONDITION'
              PRINT*, '3       FOR CALMPED-PINNED BOUNDARY CONDITION'
              PRINT*, '4       FOR CANTILEVER BOUNDAY CONDITION'
C
            READ(*,*) I
            IF ((I .LT. 1) .OR. (I .GT. 4)) THEN
```

```
                        GOTO 5
              ELSE IF(I .EQ. 1) THEN
C                       clamped-clamped bc.
                        U1 = 0
                        U2 = 0
                        V1 = 0
                        V2 = 0
                        TETHA1 = 0
                        TETHA2 = 0
                        FHI1 = 0
                        FHI2 = 0
              ELSE IF(I .EQ. 2) THEN
C                       pin-pin bc
                        U1 = 0
                        U2 = 0
                        V1 = 0
                        V2 = 0
                        TETHA1 =5
                        TETHA2 = 5
                        FHI1 = 5
                        FHI2 = 5
              ELSE IF(I .EQ.3) THEN
C                       clamped-pinned bc
                        U1 = 0
                        U2 = 0
                        V1 = 0
                        V2 = 0
                        TETHA1 =0
                        TETHA2 = 5
                        FHI1 = 0
                        FHI2 = 5
              ELSE
C                       cantilever bc
                        U1 = 0
                        U2 = 5
                        V1 = 0
                        V2 = 5
                        TETHA1 =0
                        TETHA2 = 5
                        FHI1 = 0
                        FHI2 = 5
              END IF
            END IF
C
C       ========== change the operating parameters ==========
            PRINT*, 'THE CURRENT OPERATING PARAMETERS ARE:'
            PRINT*, 'THRUST FORCE ====>  ',FZ,'  NEWTONS'
            PRINT*, 'ROTATIONAL SPEED ====>', OMEGA, '  RADIANS/SECOND'
            PRINT*,'WOULD YOU LIKE TO CHANGE OPERATING PARAMETERS? (Y/N):'
            READ(*,'(A1)',ERR=100) FLAG3
C
            IF((FLAG3 .EQ. 'Y') .OR. (FLAG3 .EQ. 'y')) THEN
 11             PRINT*, 'ENTER THE NEW THRUST FORCE :'
                READ(*,*,END=100) FZ
                PRINT*, 'THE NEW THRUST FORCE IS :', FZ, '  NEWTONS'
                    PRINT*, 'IS THE VALUE CORRECT?(Y/N):'
                    READ(*,'(A1)',ERR=100) FLAG3
                    IF((FLAG3 .EQ. 'N') .OR. (FLAG3 .EQ. 'n')) GOTO 11
```

```
C
 12        PRINT*, 'ENTER THE NEW ROTATIONAL SPEED :'
           READ(*,*,END=100) OMEGA
           PRINT*,'THE NEW ROTATIONAL SPEED IS:',OMEGA,' RAD/S'
                PRINT*, 'IS THE VALUE CORRECT?(Y/N):'
                READ(*,'(A1)',ERR=100) FLAG3
                IF((FLAG3 .EQ. 'N') .OR. (FLAG3 .EQ. 'n')) GOTO 12
          END IF
C
C       ================ change the drill specifications ==========
 20      PRINT*, 'THE CURRENT DRILL SPECS ARE:'
         PRINT*, 'AREA ====>  ', AREA, '  SQUARE METERS'
         PRINT*, 'PRINCIPAL MOMENT OF INERTIAS ARE'
         PRINT*, '====>  ',S1,'  METERS!4'
         PRINT*, '====>  ',S2,'  METERS!4'
         PRINT*, 'HELIX ANGLE IS ====>  ', BETA, '  RADIANS'
         PRINT*, 'FREE LENGTH OF DRILL BIT IS ====> ', TL, '  METERS'
         PRINT*, 'DO YOU LIKE TO CHANGE THESE SPECIFICATIONS?(Y/N)'
         READ(*,'(A1)',ERR=100) FLAG4
        IF((FLAG4 .EQ. 'Y') .OR. (FLAG4 .EQ. 'y')) THEN
C
           PRINT*,'WOULD YOU LIKE TO CHANGE ,AREA AND ',
           'AREA MOMENT OF INERTIAS? (Y/N):'
           READ(*,'(A1)', END=100) FLAG4
           IF((FLAG4 .EQ. 'Y') .OR. (FLAG4 .EQ. 'y')) THEN
 21          PRINT*, 'ENTER THE NEW CROSS-SECTIONAL AREA: '
             READ(*,*,END=100) AREA
             PRINT*, 'THE NEW AREA IS : ', AREA, 'SQUARE METERS'
                  PRINT*, 'IS THE VALUE CORRECT?(Y/N):'
                  READ(*,'(A1)',ERR=100) FLAG4
                  IF((FLAG4 .EQ. 'N') .OR. (FLAG4 .EQ. 'n')) GOTO 21
 22          PRINT*, 'ENTER THE NEW AREA MOMENT OF INERTIAS:'
             READ(*,*,END=100) S1,S2
             PRINT*, 'THE NEW MOMENTS ARE:',S1,'  AND  ',S2,'  METERS!4'
                  PRINT*, 'ARE THE VALUES CORRECT?(Y/N):'
                  READ(*,'(A1)',ERR=100) FLAG4
                  IF((FLAG4 .EQ. 'N') .OR. (FLAG4 .EQ. 'n')) GOTO 22
          ENDIF
          PRINT*, 'WOULD YOU LIKE TO CHANGE HELIX ANGLE OR LEGTH? (Y/N):'
          READ(*,'(A1)', END=100) FLAG4
          IF((FLAG4 .EQ. 'Y') .OR. (FLAG4 .EQ. 'y')) THEN
 23          PRINT*, 'ENTER THE NEW DRILL HELIX ANGLE:'
             READ(*,*,END=100) BETA
             PRINT*, 'THE NEW HELIX ANGLE IS: ', BETA, '  RADIANS'
                 PRINT*, 'IS THE VALUE CORRECT?(Y/N):'
                 READ(*,'(A1)',ERR=100) FLAG4
                 IF((FLAG4 .EQ. 'N') .OR. (FLAG4 .EQ. 'n')) GOTO 23
 24          PRINT*, 'ENTER THE NEW FREE LENGTH:'
             READ(*,*,END=100) TL
             PRINT*, 'THE NEW FREE LENGTH IS: ', TL, '  METERS'
                 PRINT*, 'IS THE VALUE CORRECT?(Y/N):'
                 READ(*,'(A1)',ERR=100) FLAG4
                 IF((FLAG4 .EQ. 'N') .OR. (FLAG4 .EQ. 'n')) GOTO 24
           ENDIF
          ENDIF
C
          PRINT*, 'PROGRAM IS BEING RERUN WITH NEW:'
          IF((FLAG2.EQ.'Y') .OR. (FLAG2.EQ.'y')) PRINT*,
```

```
     &                  'BOUNDARY CONDITIONS'
        IF((FLAG3 .EQ. 'Y') .OR. (FLAG3 .EQ. 'y')) PRINT*,
     &                  'OPERATING PARAMETERS'
        IF((FLAG4 .EQ. 'Y') .OR. (FLAG4 .EQ. 'y')) PRINT*,
     &                  'DRILL SPECIFICATIONS'
C
 100    RETURN
        END
C       ------------------------------------------------------------
C       ------------------------------------------------------------
C/* filename: fem7 */11/10/87
C       ------------------------------------------------
C
C       -----   SUBROUTINE
READ2(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL)    -----
C
C                       ARGUMENTS----------
C           AREA=Drill cross-sectional area
C           S1,S2 = Drill principal are moment of inertias
C               NL= Number of beam elements
C               OMEGA= Drill rotational speed
C               ELAS= Young's modulus of elasticity
C               BETA= Drill helix angle
C               RHO= Material density
C               FZ= Axial force
C               TL= Total free length of the drill bit
C
C
        SUBROUTINE READ2(AREA,S1,S2,NL,OMEGA,ELAS,BETA,RHO,FZ,TL
     &       ,U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2)
C
        IMPLICIT REAL*8(A-H,O-Z)
        INTEGER*2 U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
C
         READ(5,1000) AREA,S1,S2,NL
 1000    FORMAT(3G20.5,I5)
         WRITE(6,11) AREA,S1,S2,NL
11       FORMAT('AREA=', G15.8,/,
     &          'S1=',G15.8,/,
     &          'S2=',G15.8,/,
     &        ' NL=',I10)
C
        READ(5,1200) OMEGA,ELAS,BETA,RHO,FZ,TL
 1200   FORMAT(6G20.10,I5)
C
C
        WRITE(6,1202) OMEGA,ELAS,BETA,RHO,FZ,TL
 1202   FORMAT('****** OMEGA ******',G20.10,/,
     &        '******  ELAS ******',G20.10,/,
     &        '****** BETA  ******',G20.10,/,
     &        '******  RHO  ******',G20.10,/,
     &        '******   FZ  ******',G20.10,/,
     &        '******   TL  ******',G20.10)
C
        READ(5,10) U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
 10     FORMAT(8I5)
C
        WRITE(6,15) U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
```

```
 15     FORMAT('***** U1=',I3,5X,'***** TETHA1=',I3,/,
     &         '***** V1=',I3,5X,'***** FHI1=',I3,/,
     &         '***** U2=',I3,5X,'***** TETHA2=',I3,/,
     &         '***** V2=',I3,5X,'***** FHI2=',I3)
C
        RETURN
        END
C
C
C       ---------------------------------------
C
C           SUBROUTINE DISCRE(TL,NL,AREA,RHO,EL,UMASS)
C
C                       ARGUMENTS----------
C               AREA= Cross-sectional area
C               RHO= Material density
C               NL= Number of beam elements
C               EL= Length of the beam element
C               UMASS= Mass/unit lenth
C               XL= Coordinates of nodes
C               TL= Total free length of the drill bit
C
        SUBROUTINE DISCRE(TL,NL,AREA,RHO,EL,UMASS)
C
        IMPLICIT REAL*8(A-H,O-Z)
C
        EL=TL/NL
C
        UMASS=AREA*RHO
C
C
C
        RETURN
        END
C
C
C       ----------------------------------------
C
C             SUBROUTINE ELEMEN(S1,S2,EL,BETA,UMASS,FZ,OMEGA,
C                               ELAS,SME,SCE,SKE)
C
C
C                       ARGUMENTS----------
C               S1,S2= Area moment of inertias in principal dirctions
C               OMEGA= Drill rotational speed
C               ELAS= Young's modulus of elasticity
C               BETA= Drill helix angle
C               FZ= Axial force
C               EL= Length of the beam element
C               UMASS= Mass/unit lenth
C               SME,SCE,SKE= Element mass, damping, stiffness matrices
C                       --------------------
C
        SUBROUTINE ELEMEN(S1,S2,EL,BETA,UMASS,FZ,OMEGA,
     &                    ELAS,SME,SCE,SKE)
C
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION SME(8,8),SCE(8,8),SKE(8,8),
```

```
     &                    &              SKE1(8,8),SKE2(8,8),SKE4(8,8)
C          C       C
C          C       C          SME(I,J)
                     SME(1,1)=(13.*EL)/35.
                     SME(1,2)=(11.*EL**2)/210.
                     SME(1,3)=0.D0
                     SME(1,4)=0.D0
                     SME(1,5)=(9.*EL)/70.
                     SME(1,6)=(-13.*EL**2)/420.
                     SME(1,7)=0.D0
                     SME(1,8)=0.D0
                     SME(2,1)=(11.*EL**2)/210.
                     SME(2,2)=EL**3/105.
                     SME(2,3)=0.D0
                     SME(2,4)=0.D0
                     SME(2,5)=(13.*EL**2)/420.
                     SME(2,6)=(-EL**3)/140.
                     SME(2,7)=0.D0
                     SME(2,8)=0.D0
                     SME(3,1)=0.D0
                     SME(3,2)=0.D0
                     SME(3,3)=(13.*EL)/35.
                     SME(3,4)=(11.*EL**2)/210.
                     SME(3,5)=0.D0
                     SME(3,6)=0.D0
                     SME(3,7)=(9.*EL)/70.
                     SME(3,8)=(-13.*EL**2)/420.
                     SME(4,1)=0.D0
                     SME(4,2)=0.D0
                     SME(4,3)=(11.*EL**2)/210.
                     SME(4,4)=EL**3/105.
                     SME(4,5)=0.D0
                     SME(4,6)=0.D0
                     SME(4,7)=(13.*EL**2)/420.
                     SME(4,8)=(-EL**3)/140.
                     SME(5,1)=(9.*EL)/70.
                     SME(5,2)=(13.*EL**2)/420.
                     SME(5,3)=0.D0
                     SME(5,4)=0.D0
                     SME(5,5)=(13.*EL)/35.
                     SME(5,6)=(-11.*EL**2)/210.
                     SME(5,7)=0.D0
                     SME(5,8)=0.D0
                     SME(6,1)=(-13.*EL**2)/420.
                     SME(6,2)=(-EL**3)/140.
                     SME(6,3)=0.D0
                     SME(6,4)=0.D0
                     SME(6,5)=(-11.*EL**2)/210.
                     SME(6,6)=EL**3/105.
                     SME(6,7)=0.D0
                     SME(6,8)=0.D0
                     SME(7,1)=0.D0
                     SME(7,2)=0.D0
                     SME(7,3)=(9.*EL)/70.
                     SME(7,4)=(13.*EL**2)/420.
                     SME(7,5)=0.D0
                     SME(7,6)=0.D0
                     SME(7,7)=(13.*EL)/35.
```

```
SME(7,8)=(-11.*EL**2)/210.
SME(8,1)=0.D0
SME(8,2)=0.D0
SME(8,3)=(-13.*EL**2)/420.
SME(8,4)=(-EL**3)/140.
SME(8,5)=0.D0
SME(8,6)=0.D0
SME(8,7)=(-11.*EL**2)/210.
SME(8,8)=EL**3/105.


   SCE(I,J)
  DUMCE=2.* OMEGA
SCE(1,1)=0.D0
SCE(1,2)=0.D0
SCE(1,3)=(-13.*EL*DUMCE)/35.
SCE(1,4)=(-11.*EL**2*DUMCE)/210.
SCE(1,5)=0.D0
SCE(1,6)=0.D0
SCE(1,7)=(-9.*EL*DUMCE)/70.
SCE(1,8)=(13.*EL**2*DUMCE)/420.
SCE(2,1)=0.D0
SCE(2,2)=0.D0
SCE(2,3)=(-11.*EL**2*DUMCE)/210.
SCE(2,4)=(-EL**3*DUMCE)/105.
SCE(2,5)=0.D0
SCE(2,6)=0.D0
SCE(2,7)=(-13.*EL**2*DUMCE)/420.
SCE(2,8)=(EL**3*DUMCE)/140.
SCE(3,1)=(13.*EL*DUMCE)/35.
SCE(3,2)=(11.*EL**2*DUMCE)/210.
SCE(3,3)=0.D0
SCE(3,4)=0.D0
SCE(3,5)=(9.*EL*DUMCE)/70.
SCE(3,6)=(-13.*EL**2*DUMCE)/420.
SCE(3,7)=0.D0
SCE(3,8)=0.D0
SCE(4,1)=(11.*EL**2*DUMCE)/210.
SCE(4,2)=(EL**3*DUMCE)/105.
SCE(4,3)=0.D0
SCE(4,4)=0.D0
SCE(4,5)=(13.*EL**2*DUMCE)/420.
SCE(4,6)=(-EL**3*DUMCE)/140.
SCE(4,7)=0.D0
SCE(4,8)=0.D0
SCE(5,1)=0.D0
SCE(5,2)=0.D0
SCE(5,3)=(-9.*EL*DUMCE)/70.
SCE(5,4)=(-13.*EL**2*DUMCE)/420.
SCE(5,5)=0.D0
SCE(5,6)=0.D0
SCE(5,7)=(-13.*EL*DUMCE)/35.
SCE(5,8)=(11.*EL**2*DUMCE)/210.
SCE(6,1)=0.D0
SCE(6,2)=0.D0
SCE(6,3)=(13.*EL**2*DUMCE)/420.
SCE(6,4)=(EL**3*DUMCE)/140.
SCE(6,5)=0.D0
```

```
      SCE(6,6)=0.D0
      SCE(6,7)=(11.*EL**2*DUMCE)/210.
      SCE(6,8)=(-EL**3*DUMCE)/105.
      SCE(7,1)=(9.*EL*DUMCE)/70.
      SCE(7,2)=(13.*EL**2*DUMCE)/420.
      SCE(7,3)=0.D0
      SCE(7,4)=0.D0
      SCE(7,5)=(13.*EL*DUMCE)/35.
      SCE(7,6)=(-11.*EL**2*DUMCE)/210.
      SCE(7,7)=0.D0
      SCE(7,8)=0.D0
      SCE(8,1)=(-13.*EL**2*DUMCE)/420.
      SCE(8,2)=(-EL**3*DUMCE)/140.
      SCE(8,3)=0.D0
      SCE(8,4)=0.D0
      SCE(8,5)=(-11.*EL**2*DUMCE)/210.
      SCE(8,6)=(EL**3*DUMCE)/105.
      SCE(8,7)=0.D0
      SCE(8,8)=0.D0
C
C        SKE1(I,J)
        DUMK1=-(OMEGA**2)
      SKE1(1,1)=DUMK1*(13.*EL)/35.
      SKE1(1,2)=DUMK1*(11.*EL**2)/210.
      SKE1(1,3)=0.D0
      SKE1(1,4)=0.D0
      SKE1(1,5)=DUMK1*(9.*EL)/70.
      SKE1(1,6)=-DUMK1*(13.*EL**2)/420.
      SKE1(1,7)=0.D0
      SKE1(1,8)=0.D0
      SKE1(2,1)=DUMK1*(11.*EL**2)/210.
      SKE1(2,2)=DUMK1*EL**3/105.
      SKE1(2,3)=0.D0
      SKE1(2,4)=0.D0
      SKE1(2,5)=DUMK1*(13.*EL**2)/420.
      SKE1(2,6)=-DUMK1*(EL**3)/140.
      SKE1(2,7)=0.D0
      SKE1(2,8)=0.D0
      SKE1(3,1)=0.D0
      SKE1(3,2)=0.D0
      SKE1(3,3)=DUMK1*(13.*EL)/35.
      SKE1(3,4)=DUMK1*(11.*EL**2)/210.
      SKE1(3,5)=0.D0
      SKE1(3,6)=0.D0
      SKE1(3,7)=DUMK1*(9.*EL)/70.
      SKE1(3,8)=-DUMK1*(13.*EL**2)/420.
      SKE1(4,1)=0.D0
      SKE1(4,2)=0.D0
      SKE1(4,3)=DUMK1*(11.*EL**2)/210.
      SKE1(4,4)=DUMK1*EL**3/105.
      SKE1(4,5)=0.D0
      SKE1(4,6)=0.D0
      SKE1(4,7)=DUMK1*(13.*EL**2)/420.
      SKE1(4,8)=-DUMK1*(EL**3)/140.
      SKE1(5,1)=DUMK1*(9.*EL)/70.
      SKE1(5,2)=DUMK1*(13.*EL**2)/420.
      SKE1(5,3)=0.D0
      SKE1(5,4)=0.D0
```

```
      SKE1(5,5)=DUMK1*(13.*EL)/35.
      SKE1(5,6)=-DUMK1*(11.*EL**2)/210.
      SKE1(5,7)=0.D0
      SKE1(5,8)=0.D0
      SKE1(6,1)=-DUMK1*(13.*EL**2)/420.
      SKE1(6,2)=-DUMK1*(EL**3)/140.
      SKE1(6,3)=0.D0
      SKE1(6,4)=0.D0
      SKE1(6,5)=-DUMK1*(11.*EL**2)/210.
      SKE1(6,6)=DUMK1*EL**3/105.
      SKE1(6,7)=0.D0
      SKE1(6,8)=0.D0
      SKE1(7,1)=0.D0
      SKE1(7,2)=0.D0
      SKE1(7,3)=DUMK1*(9.*EL)/70.
      SKE1(7,4)=DUMK1*(13.*EL**2)/420.
      SKE1(7,5)=0.D0
      SKE1(7,6)=0.D0
      SKE1(7,7)=DUMK1*(13.*EL)/35.
      SKE1(7,8)=-DUMK1*(11.*EL**2)/210.
      SKE1(8,1)=0.D0
      SKE1(8,2)=0.D0
      SKE1(8,3)=-DUMK1*(13.*EL**2)/420.
      SKE1(8,4)=-DUMK1*(EL**3)/140.
      SKE1(8,5)=0.D0
      SKE1(8,6)=0.D0
      SKE1(8,7)=-DUMK1*(11.*EL**2)/210.
      SKE1(8,8)=DUMK1*EL**3/105.
C
C        SKE2(I,J)
        DUMK2 = ELAS / UMASS
      SKE2(1,1)=(13.*EL**4*BETA**4*S1+168.*EL**2*BETA**2*S2+84.*
     . EL**2*BETA**2*S1+420.*S1)/(35.*EL**3) * DUMK2
      SKE2(1,2)=(11.*EL**4*BETA**4*S1+84.*EL**2*BETA**2*S2+252.*
     . EL**2*BETA**2*S1+1260.*S1)/(210.*EL**2) * DUMK2
      SKE2(1,3)=BETA**3*(S2-S1) * DUMK2
      SKE2(1,4)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(1,5)=(3.*(3.*EL**4*BETA**4*S1-112.*EL**2*BETA**2*S2-
     . 56.*EL**2*BETA**2*S1-280.*S1))/(70.*EL**3) * DUMK2
      SKE2(1,6)=(-13.*EL**4*BETA**4*S1+168.*EL**2*BETA**2*S2+84.*
     . EL**2*BETA**2*S1+2520.*S1)/(420.*EL**2) * DUMK2
      SKE2(1,7)=BETA**3*(S2+S1) * DUMK2
      SKE2(1,8)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(2,1)=(11.*EL**4*BETA**4*S1+84.*EL**2*BETA**2*S2+252.*
     . EL**2*BETA**2*S1+1260.*S1)/(210.*EL**2) * DUMK2
      SKE2(2,2)=(EL**4*BETA**4*S1+56.*EL**2*BETA**2*S2+28.*EL**2*
     . BETA**2*S1+420.*S1)/(105.*EL) * DUMK2
      SKE2(2,3)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(2,4)=BETA*(-S2+S1) * DUMK2
      SKE2(2,5)=(13.*EL**4*BETA**4*S1-168.*EL**2*BETA**2*S2-84.*
     . EL**2*BETA**2*S1-2520.*S1)/(420.*EL**2) * DUMK2
      SKE2(2,6)=(-3.*EL**4*BETA**4*S1-56.*EL**2*BETA**2*S2-28.*EL
     . **2*BETA**2*S1+840.*S1)/(420.*EL) * DUMK2
      SKE2(2,7)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
```

```
      SKE2(2,8)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-30.*S2-
     . 30.*S1))/30. * DUMK2
      SKE2(3,1)=BETA**3*(S2-S1) * DUMK2
      SKE2(3,2)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(3,3)=(13.*EL**4*BETA**4*S2+84.*EL**2*BETA**2*S2+168.*
     . EL**2*BETA**2*S1+420.*S2)/(35.*EL**3) * DUMK2
      SKE2(3,4)=(11.*EL**4*BETA**4*S2+252.*EL**2*BETA**2*S2+84.*
     . EL**2*BETA**2*S1+1260.*S2)/(210.*EL**2) * DUMK2
      SKE2(3,5)=-BETA**3*(S2+S1) * DUMK2
      SKE2(3,6)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(3,7)=(3.*(3.*EL**4*BETA**4*S2-56.*EL**2*BETA**2*S2-
     . 112.*EL**2*BETA**2*S1-280.*S2))/(70.*EL**3) * DUMK2
      SKE2(3,8)=(-13.*EL**4*BETA**4*S2+84.*EL**2*BETA**2*S2+168.*
     . EL**2*BETA**2*S1+2520.*S2)/(420.*EL**2) * DUMK2
      SKE2(4,1)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(4,2)=BETA*(-S2+S1) * DUMK2
      SKE2(4,3)=(11.*EL**4*BETA**4*S2+252.*EL**2*BETA**2*S2+84.*
     . EL**2*BETA**2*S1+1260.*S2)/(210.*EL**2) * DUMK2
      SKE2(4,4)=(EL**4*BETA**4*S2+28.*EL**2*BETA**2*S2+56.*EL**2*
     . BETA**2*S1+420.*S2)/(105.*EL) * DUMK2
      SKE2(4,5)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(4,6)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+30.*S2+
     . 30.*S1))/30. * DUMK2
      SKE2(4,7)=(13.*EL**4*BETA**4*S2-84.*EL**2*BETA**2*S2-168.*
     . EL**2*BETA**2*S1-2520.*S2)/(420.*EL**2) * DUMK2
      SKE2(4,8)=(-3.*EL**4*BETA**4*S2-28.*EL**2*BETA**2*S2-56.*EL
     . **2*BETA**2*S1+840.*S2)/(420.*EL) * DUMK2
      SKE2(5,1)=(3.*(3.*EL**4*BETA**4*S1-112.*EL**2*BETA**2*S2-
     . 56.*EL**2*BETA**2*S1-280.*S1))/(70.*EL**3) * DUMK2
      SKE2(5,2)=(13.*EL**4*BETA**4*S1-168.*EL**2*BETA**2*S2-84.*
     . EL**2*BETA**2*S1-2520.*S1)/(420.*EL**2) * DUMK2
      SKE2(5,3)=-BETA**3*(S2+S1) * DUMK2
      SKE2(5,4)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(5,5)=(13.*EL**4*BETA**4*S1+168.*EL**2*BETA**2*S2+84.*
     . EL**2*BETA**2*S1+420.*S1)/(35.*EL**3) * DUMK2
      SKE2(5,6)=(-11.*EL**4*BETA**4*S1-84.*EL**2*BETA**2*S2-252.*
     . EL**2*BETA**2*S1-1260.*S1)/(210.*EL**2) * DUMK2
      SKE2(5,7)=BETA**3*(-S2+S1) * DUMK2
      SKE2(5,8)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(6,1)=(-13.*EL**4*BETA**4*S1+168.*EL**2*BETA**2*S2+84.*
     . EL**2*BETA**2*S1+2520.*S1)/(420.*EL**2) * DUMK2
      SKE2(6,2)=(-3.*EL**4*BETA**4*S1-56.*EL**2*BETA**2*S2-28.*EL
     . **2*BETA**2*S1+840.*S1)/(420.*EL) * DUMK2
      SKE2(6,3)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(6,4)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+30.*S2+
     . 30.*S1))/30. * DUMK2
      SKE2(6,5)=(-11.*EL**4*BETA**4*S1-84.*EL**2*BETA**2*S2-252.*
     . EL**2*BETA**2*S1-1260.*S1)/(210.*EL**2) * DUMK2
      SKE2(6,6)=(EL**4*BETA**4*S1+56.*EL**2*BETA**2*S2+28.*EL**2*
     . BETA**2*S1+420.*S1)/(105.*EL) * DUMK2
      SKE2(6,7)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
```

```
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(6,8)=BETA*(S2-S1) * DUMK2
      SKE2(7,1)=BETA**3*(S2+S1) * DUMK2
      SKE2(7,2)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(7,3)=(3.*(3.*EL**4*BETA**4*S2-56.*EL**2*BETA**2*S2-
     . 112.*EL**2*BETA**2*S1-280.*S2))/(70.*EL**3) * DUMK2
      SKE2(7,4)=(13.*EL**4*BETA**4*S2-84.*EL**2*BETA**2*S2-168.*
     . EL**2*BETA**2*S1-2520.*S2)/(420.*EL**2) * DUMK2
      SKE2(7,5)=BETA**3*(-S2+S1) * DUMK2
      SKE2(7,6)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(7,7)=(13.*EL**4*BETA**4*S2+84.*EL**2*BETA**2*S2+168.*
     . EL**2*BETA**2*S1+420.*S2)/(35.*EL**3) * DUMK2
      SKE2(7,8)=(-11.*EL**4*BETA**4*S2-252.*EL**2*BETA**2*S2-84.*
     . EL**2*BETA**2*S1-1260.*S2)/(210.*EL**2) * DUMK2
      SKE2(8,1)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-10.*S2-
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(8,2)=(BETA*(-EL**2*BETA**2*S2-EL**2*BETA**2*S1-30.*S2-
     . 30.*S1))/30. * DUMK2
      SKE2(8,3)=(-13.*EL**4*BETA**4*S2+84.*EL**2*BETA**2*S2+168.*
     . EL**2*BETA**2*S1+2520.*S2)/(420.*EL**2) * DUMK2
      SKE2(8,4)=(-3.*EL**4*BETA**4*S2-28.*EL**2*BETA**2*S2-56.*EL
     . **2*BETA**2*S1+840.*S2)/(420.*EL) * DUMK2
      SKE2(8,5)=(BETA*(EL**2*BETA**2*S2+EL**2*BETA**2*S1+10.*S2+
     . 10.*S1))/(5.*EL) * DUMK2
      SKE2(8,6)=BETA*(S2-S1) * DUMK2
      SKE2(8,7)=(-11.*EL**4*BETA**4*S2-252.*EL**2*BETA**2*S2-84.*
     . EL**2*BETA**2*S1-1260.*S2)/(210.*EL**2) * DUMK2
      SKE2(8,8)=(EL**4*BETA**4*S2+28.*EL**2*BETA**2*S2+56.*EL**2*
     . BETA**2*S1+420.*S2)/(105.*EL) * DUMK2
C
C
C        SKE4(I,J)
        DUMK4 = - FZ / UMASS
         WRITE(6,80) DUMK4
 80      FORMAT('IN ELMEN    DUMK4=', G20.10)
      SKE4(1,1)=(DUMK4*(13.*BETA**2*EL**2+42.))/(35.*EL)
      SKE4(1,2)=(DUMK4*(11.*BETA**2*EL**2+21.))/210.
      SKE4(1,3)=0.D0
      SKE4(1,4)=(BETA*EL*DUMK4)/5.
      SKE4(1,5)=(3.*DUMK4*(3.*BETA**2*EL**2-28.))/(70.*EL)
      SKE4(1,6)=(DUMK4*(-13.*BETA**2*EL**2+42.))/420.
      SKE4(1,7)=BETA*DUMK4
      SKE4(1,8)=(-BETA*EL*DUMK4)/5.
      SKE4(2,1)=(DUMK4*(11.*BETA**2*EL**2+21.))/210.
      SKE4(2,2)=(EL*DUMK4*(BETA**2*EL**2+14.))/105.
      SKE4(2,3)=(-BETA*EL*DUMK4)/5.
      SKE4(2,4)=0.D0
      SKE4(2,5)=(DUMK4*(13.*BETA**2*EL**2-42.))/420.
      SKE4(2,6)=(EL*DUMK4*(-3.*BETA**2*EL**2-14.))/420.
      SKE4(2,7)=(BETA*EL*DUMK4)/5.
      SKE4(2,8)=(-BETA*EL**2*DUMK4)/30.
      SKE4(3,1)=0.D0
      SKE4(3,2)=(-BETA*EL*DUMK4)/5.
      SKE4(3,3)=(DUMK4*(13.*BETA**2*EL**2+42.))/(35.*EL)
      SKE4(3,4)=(DUMK4*(11.*BETA**2*EL**2+21.))/210.
      SKE4(3,5)=-BETA*DUMK4
```

```
      SKE4(3,6)=(BETA*EL*DUMK4)/5.
      SKE4(3,7)=(3.*DUMK4*(3.*BETA**2*EL**2-28.))/(70.*EL)
      SKE4(3,8)=(DUMK4*(-13.*BETA**2*EL**2+42.))/420.
      SKE4(4,1)=(BETA*EL*DUMK4)/5.
      SKE4(4,2)=0.D0
      SKE4(4,3)=(DUMK4*(11.*BETA**2*EL**2+21.))/210.
      SKE4(4,4)=(EL*DUMK4*(BETA**2*EL**2+14.))/105.
      SKE4(4,5)=(-BETA*EL*DUMK4)/5.
      SKE4(4,6)=(BETA*EL**2*DUMK4)/30.
      SKE4(4,7)=(DUMK4*(13.*BETA**2*EL**2-42.))/420.
      SKE4(4,8)=(EL*DUMK4*(-3.*BETA**2*EL**2-14.))/420.
      SKE4(5,1)=(3.*DUMK4*(3.*BETA**2*EL**2-28.))/(70.*EL)
      SKE4(5,2)=(DUMK4*(13.*BETA**2*EL**2-42.))/420.
      SKE4(5,3)=-BETA*DUMK4
      SKE4(5,4)=(-BETA*EL*DUMK4)/5.
      SKE4(5,5)=(DUMK4*(13.*BETA**2*EL**2+42.))/(35.*EL)
      SKE4(5,6)=(DUMK4*(-11.*BETA**2*EL**2-21.))/210.
      SKE4(5,7)=0.D0
      SKE4(5,8)=(BETA*EL*DUMK4)/5.
      SKE4(6,1)=(DUMK4*(-13.*BETA**2*EL**2+42.))/420.
      SKE4(6,2)=(EL*DUMK4*(-3.*BETA**2*EL**2-14.))/420.
      SKE4(6,3)=(BETA*EL*DUMK4)/5.
      SKE4(6,4)=(BETA*EL**2*DUMK4)/30.
      SKE4(6,5)=(DUMK4*(-11.*BETA**2*EL**2-21.))/210.
      SKE4(6,6)=(EL*DUMK4*(BETA**2*EL**2+14.))/105.
      SKE4(6,7)=(-BETA*EL*DUMK4)/5.
      SKE4(6,8)=0.D0
      SKE4(7,1)=BETA*DUMK4
      SKE4(7,2)=(BETA*EL*DUMK4)/5.
      SKE4(7,3)=(3.*DUMK4*(3.*BETA**2*EL**2-28.))/(70.*EL)
      SKE4(7,4)=(DUMK4*(13.*BETA**2*EL**2-42.))/420.
      SKE4(7,5)=0.D0
      SKE4(7,6)=(-BETA*EL*DUMK4)/5.
      SKE4(7,7)=(DUMK4*(13.*BETA**2*EL**2+42.))/(35.*EL)
      SKE4(7,8)=(DUMK4*(-11.*BETA**2*EL**2-21.))/210.
      SKE4(8,1)=(-BETA*EL*DUMK4)/5.
      SKE4(8,2)=(-BETA*EL**2*DUMK4)/30.
      SKE4(8,3)=(DUMK4*(-13.*BETA**2*EL**2+42.))/420.
      SKE4(8,4)=(EL*DUMK4*(-3.*BETA**2*EL**2-14.))/420.
      SKE4(8,5)=(BETA*EL*DUMK4)/5.
      SKE4(8,6)=0.D0
      SKE4(8,7)=(DUMK4*(-11.*BETA**2*EL**2-21.))/210.
      SKE4(8,8)=(EL*DUMK4*(BETA**2*EL**2+14.))/105.
C
C
        DO 1300 I=1,8
        DO 1300 J=1,8
        SKE(I,J) = SKE1(I,J) + SKE2(I,J) + SKE4(I,J)
 1300   CONTINUE
C
C
        RETURN
        END
C
C
C
C       ------------------------------------------------
C
```

```
C                       SUBROUTINE
ASEMBO(SME,SCE,SKE,NL,N1,N2,MDIM,TM,TC)
C
C                       ARGUMENTS----------
C
C               SME,SCE,SKE= Element mass, damping, stiffness matrices
C               TM,TC,TK= Global mass, damping, stiffness matrices
C                       note TK is passed in the  common
C                       block A
C               M1,MDIM= Dimension of the global matrices
C               NL= Number of elements
C                       --------------------
C
C
      SUBROUTINE ASEMBO(SME,SCE,SKE,NL,N1,MDIM,TM,TC,TK)
C
      REAL*8 SME(8,8),SCE(8,8),SKE(8,8),TM(2*N1+1,1),
     &       TC(N1,1),TK(2*N1+1,1)
C
C
C
C
       DO 100 I=1,MDIM
       DO 100 J=1,MDIM
       TM(I,J)=0.D0
       TK(I,J)=0.D0
       TC(I,J)=0.D0
 100   CONTINUE
C
      DO 1530 I=1,NL
C
      I1=4*(I-1)
C
               DO 1530 IIN=1,8
               DO 1530 JIN=1,8
               TM(I1+IIN,I1+JIN)=TM(I1+IIN,I1+JIN)+SME(IIN,JIN)
               TK(I1+IIN,I1+JIN)=TK(I1+IIN,I1+JIN)+SKE(IIN,JIN)
               TC(I1+IIN,I1+JIN)=TC(I1+IIN,I1+JIN)+SCE(IIN,JIN)
 1530  CONTINUE
C
C
      RETURN
      END
C
C  ---------------------------------------------------
C  ---------------------------------------------------
C
C     /* filename eigens7 11/19/87 */
C
C     This subroutine calculates the eigenvalues of the
C     syetem of equations using NAAS:EISPACK routine
C     RSG. The routine solves the generalized symmetric eigenvalue
C     problem.
C
C
      SUBROUTINE EIGENS(N1,MDIM,TM,TC,TK,TMINV,W,SV1,SV2)
C
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
C
C       NOTE THE FOLLOWING PARAMETER DECLERATION IS SET FOR N1=120
C       OR 30 ELEMENTS
        PARAMETER(N3=241)
C
        DIMENSION TM(2*N1+1,1),TC(N1,1),TK(2*N1+1,1),TMINV(N1,1)
C       ::::::::: local vector declerations ::::::::::::::
        DIMENSION W(1),Z(N3,N3),SV1(1),SV2(1)
C
         M1 = 2*MDIM
        DO 89 I=1,M1
        W(I) = 0.D0
        SV1(I) = 0.D0
        SV2(I) = 0.D0
        DO 89 J=1,M1
        Z(I,J) = 0.D0
 89     CONTINUE
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C         ======= transform the equations into meirowitch form ======
C       ================= assemble state space form =========
        CALL STATE1(N1,MDIM,TM,TC,TK,TMINV)
        WRITE(6,5)
 5      FORMAT( 'STATE1 IS COMPLETE',/)
C
        MATZ=0
        CALL RSG(2*N1+1,M1,TK,TM,W,MATZ,Z,SV1,SV2,IER)
C                M
         WRITE(6,7)
 7       FORMAT(' RSG COMPLETE')
C
       WRITE(6,1705)
1705   FORMAT('**2*MDIM******* EIGENVALUES *************')
      DO 1710 I=1,M1
        PI=3.141592654D0
        W(I) = DSQRT(DABS(W(I)))/2./PI
       WRITE(6,1706) I, W(I)
 1706   FORMAT(I5,5X,'(',G20.6,') i')
 1710   CONTINUE
C
         WRITE(6,1720) IER
 1720    FORMAT('****** IER***** =', I5)
C
        RETURN
        END
C
C-----------------------------------------------------------------
C-----------------------------------------------------------------
C
        SUBROUTINE STATE1(N1,MDIM,TM,TC,TK,TMINV)
C       input matices:
C            |TM     0|              |TK     0|
C       TM = |        | ,       TK = |        |, TC,TMINV
C            |0      0|              | 0     0|
C
C       matrix descriptions:
C            |TM11     TM12|                |TK11    TK12|
C       TM = |             |           TK = |            |
```

```
C             |TM21     TM22|                  |TK21    TK22|
C
C       state space configuration:
C            |TM     0|           |-TC*TMINV*TC+TK       tr(TK21)|
C       TM = |        |      TK = |                              |
C            |0     TK|           |TK*TMINV*TC         TK*TMINV*TK|
C
C
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION TM(2*N1+1,1),TC(N1,1),TK(2*N1+1,1),TMINV(N1,1)
C
C       place TK stiffness part in TM matrix
        DO 10 I=1,MDIM
        DO 10 J=1,MDIM
        TM(MDIM+I,MDIM+J) = TK(I,J)
 10     CONTINUE
C
C       ::::::initialize  TK::::::::::::::
        DO 20 I=1,2*MDIM
        DO 20 J=1,2*MDIM
        TK(I,J) = 0.D0
 20     CONTINUE
C
C       ::::::TK12 = TMINV * TC ::::::::::::
        DO 30 I=1,MDIM
        DO 30 J=1,MDIM
        DO 30 K=1,MDIM
        TK(I,J+MDIM) = TK(I,J+MDIM) + TMINV(I,K) * TC(K,J)
 30     CONTINUE
C
C       ::::::TK11 = -TC * TK12 = TC * TMINV * TC::::::::
        DO 40 I=1,MDIM
        DO 40 J=1,MDIM
        DO 40 K=1,MDIM
        TK(I,J) = TK(I,J) + TC(I,K) * TK(K,J+MDIM)
 40     CONTINUE
C
C       ::::::TK11 = -TK11 + TM22 = TC * TMINV * TC + TK:::::
        DO 50 I=1,MDIM
        DO 50 J=1,MDIM
        TK(I,J) = -TK(I,J) + TM(I+MDIM,J+MDIM)
 50     CONTINUE
C
C       ::::::TK21 = TM22 * TK12 = TK * TMINV * TC::::::::
        DO 55 I=1,MDIM
        DO 55 J=1,MDIM
        DO 55 K=1,MDIM
    fi  TK(I+MDIM,J) = TK(I+MDIM,J) + TM(I+MDIM,K+MDIM) * TK(K,J+MDIM)
 55     CONTINUE
C
C       ::::::discarding the coriolis matrix!!::::::
C       ::::::TC = TMINV * TK::::::::::::::
        DO 57 I=1,MDIM
        DO 57 J=1,MDIM
        TC(I,J) = 0.D0
 57     CONTINUE
C
        DO 60 I=1,MDIM
```

```
        DO 60 J=1,MDIM
        DO 60 K=1,MDIM
        TC(I,J) = TC(I,J) + TMINV(I,K) * TM(K+MDIM,J+MDIM)
 60     CONTINUE
C
C       ::::::TK22 = TM22 * TC =  TK * TMINV * TK
        DO 70 I=1,MDIM
        DO 70 J=1,MDIM
        DO 70 K=1,MDIM
        TK(I+MDIM,J+MDIM) =TK(I+MDIM,J+MDIM)+TM(I+MDIM,MDIM+K)*TC(K,J)
 70     CONTINUE
C
C       ::::::TK12 = tr(TK21)
        DO 80 I=1,MDIM
        DO 80 J=1,MDIM
        DO 80 K=1,MDIM
        TK(I,J+MDIM) = TK(J+MDIM,I)
 80     CONTINUE
C
C       place zeros to the necessary places im TM
         DO 90 I=1,MDIM
         DO 90 J=1,MDIM
         TM(I,J+MDIM) = 0.D0
         TM(I+MDIM,J) = 0.D0
 90      CONTINUE
C
        RETURN
        END
C/* filename: boundry7 11/10/87 */
C       CALCULATE BOUNDARY CONDITIONS
C       ------------------------------------
C               This file contains the following subroutines:
C               ============================================
C     /*   SUBROUTINE GBOUND(TM,TC,N1,N2,MDIM,U1,TETHA1,V1,FHI1,
C     &      U2,TETHA2,V2,FHI2)
C*/
C     /*   SUBROUTINE KNCOFF(TM,TC,TK,TMMAX,TCMAX,TKMAX,IIN,N1,M1,MDIM)
C*/
C  ---------------------------------------------------
C  ---------------------------------------------------
C
C        SUBROUTINE GBOUND(TM,TC,TK,N1,MDIM,U1,TETHA1,V1,FHI1,
C     &      U2,TETHA2,V2,FHI2)
C
C        This subroutine assigns the geometric boundary
C       conditions to the global mass, coriolis & stiffness
C       matrices.
C       It reads the values U1,TETHA1,...etc. from the file, then
C       if  U1=0 then U1 is constrained thus subroutine makes
C       the necessary changes in the global mass stiffness and
C       coriolis matrices
C
        SUBROUTINE GBOUND(TM,TC,TK,N1,MDIM,U1,TETHA1,V1,FHI1,
     &      U2,TETHA2,V2,FHI2)
C
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION TM(2*N1+1,1),TC(N1,1),TK(2*N1+1,1)
        INTEGER*2 U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
```

```
C
C
      WRITE(6,15) U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2
 15   FORMAT('***** U1=',I3,5X,'***** TETHA1=',I3,/,
     &       '***** V1=',I3,5X,'***** FHI1=',I3,/,
     &       '***** U2=',I3,5X,'***** TETHA2=',I3,/,
     &       '***** V2=',I3,5X,'***** FHI2=',I3)
C
C
      IIS=0
      IF (U1.NE.0) GOTO 30
       IIN=1
       CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
       IIS=IIS+1
 30   IF (TETHA1.NE.0) GOTO 40
       IIN=2-IIS
       CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
       IIS=IIS+1
 40   IF (V1.NE.0) GOTO 50
       IIN=3-IIS
       CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
       IIS=IIS+1
 50   IF (FHI1.NE.0) GOTO 60
       IIN=4-IIS
       CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
       IIS=IIS+1
 60   IF (U2.NE.0) GOTO 70
      IIN=MDIM-3
      CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
 70   IF (TETHA2.NE.0) GOTO 80
      IIN=MDIM-2
      CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
 80   IF (V2.NE.0) GOTO 90
      IIN=MDIM-1
      CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
 90   IF (FHI2.NE.0) GOTO 100
      IIN=MDIM
      CALL KNCOFF(TM,TC,TK,IIN,N1,MDIM)
 100  CONTINUE
C
      RETURN
      END
C
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
C
C     SUBROUTINE KNCOFF(TM,TC,TK,TMMAX,TCMAX,TKMAX,IIN,N1,M1,MDIM)
C
C     this subroutine removes the IIN'th row and columns
C         and concequently the matrix dimesnsion is reduced by one
C             i.e.  MDIM=MDIM-1
C
C        |        a1n     |
C        |       .a2n.... |
C        |...        .... |
C     TM=|an1 an2   ...ann| ain and ani are removed
```

```
C           |        ain     |
C           |        ..      |
C
        SUBROUTINE KNCOFF(TM,TC,TK,IIN,N1,MDIM)
C
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION TM(2*N1+1,1), TC(N1,1),TK(2*N1+1,1)
C
C
        WRITE(6,199) IIN
 199     FORMAT('ROW COLOUMN',I5,'IS REMOVED')
C
C       reduce the dimension of the global matrices by 1
        MDIM = MDIM - 1
C       Remove the unvanted row IIN
        DO 100 I = IIN,MDIM
        DO 100 J = 1,MDIM+1
        TM(I,J) = TM(I+1,J)
        TC(I,J) = TC(I+1,J)
        TK(I,J) = TK(I+1,J)
 100    CONTINUE
C
C       Remove the unvanted coumn IIN
        DO 200 I = 1,MDIM
        DO 200 J = IIN,MDIM
        TM(I,J) = TM(I,J+1)
        TC(I,J) = TC(I,J+1)
        TK(I,J) = TK(I,J+1)
 200    CONTINUE
C
C
        RETURN
        END
C/*filename: timer9 12/3/87 */
C       ================ Associted Subroutines: ==============
C
C       STATES2 = forms the state space form, results are returned
C                 in TKTOT
C       READ4 = reads the initial conditions, and the magnitude
C                 the cutting forces, FX,FY
C       DRIVE=is a NAAS:NAL integration routine that uses
C                 GEAR METHOD
C
C
C
C       ======================== Variables:====================
C       Y =       solution vector,initial conditions
C       FX,FY = cutting forces
C       FUNCT = is a subroutine called by DRKF45 to calculate
C                 right hand side vector at every integration step
C                 should be declared external in the calling routines
C
        SUBROUTINE TIMER(N1,MDIM,TC,TK,TMINV,UMASS)
C
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER(N2=84)
C
        DIMENSION TMINV(N1,1),TC(N1,1),TK(2*N1+1,1)
```

```
      DIMENSION Y(2*N2+1)
      COMMON /C/TIMEC,F(1)
      COMMON /GEAR2/YMAX(2*N2+1)
      COMMON /GEAR3/ERROR(2*N2+1)
      COMMON /GEAR4/SAVE1(2*N2+1)
      COMMON /GEAR5/SAVE2(2*N2+1)
      COMMON /GEAR6/PW(400)
      COMMON /GEAR7/IPIV(2*N2+1)
       COMMON /BB/ICOUNT
       ICOUNT = 0
C
      M2=MDIM
      INIT = 3
C
C
C     :::::: read the corresponding data :::::::::
       IDIM=2*MDIM+1
      CALL READ4(MDIM,Y,FX,FY,T,TFINAL,NSTEPS,TIMEC1)
C
C     :::::: formulate the state space represantation ::::::::::
      CALL STATE2(N1,MDIM,TMINV,TC,TK,TIMEC1)
C
C     ::::::: setup the forcing function ::::::::::::::::
      TIMEC=TIMEC1
      FX = FX/UMASS
      FY = FY/UMASS
C
C     :::::: forcing function in state space representation :::::::
      DO 5 I=1,MDIM
              F(I) = TMINV(I,1) * FX + TMINV(I,3) * FY
 5    CONTINUE
C
      DO 6 I=1+MDIM,2*MDIM
              F(I) = 0.D0
 6    CONTINUE
C
C
      TDELTA = (TFINAL - T)/NSTEPS
      PRINT*,'TDELTA=',TDELTA
      PRINT*,'T = ', T
      TOUT = T + TDELTA
      PRINT*,'TOUT=',TOUT
C
       EPS=1.D-6
       INDEX=1
       HO=1.D-5
      DO 50 I=1,NSTEPS
              RN = FNORM(INIT)
             PRINT*,'RANDOM NUMBER = ',RN
             WRITE(7,*) 'RANDOM NUMBER = ',RN
              F(2*MDIM+1) = RN/TIMEC1
              CALL DRIVE(IDIM,T,HO,Y,TOUT,EPS,20,INDEX)
              PRINT*,'DRIVE'
              WRITE(7,*) 'STEP # ',I,'  ','SOLUTION POINT  ',TOUT
              PRINT*,'STEP NO:',I,'   INDEX: ',INDEX
              PRINT*,'SOULUTION POINT:  ',TOUT
C
              TOUT=TOUT+TDELTA
```

```
C
              WRITE(7,30) (Y(J) ,J=MDIM+1,2*MDIM,4)
 30           FORMAT(5G15.7)
 50     CONTINUE
C
        RETURN
        END
C
C       ------------------------------------------------------------
C       ------------------------------------------------------------
        SUBROUTINE STATE2(N1,MDIM,TMINV,TC,TK,TIMEC)
C
C            |-TMINV*TC   -TMINV*TK        0     |
C       TK = |                             .     |
C            | I                0          0     |
C            | 0 ....           0       -1/TIMEC|
C
C            |TK11   TK12|
C       TK = |           |
C            |TK21   TK22|
C
C
        IMPLICIT REAL*8(A-H,O-Z)
C
        DIMENSION TMINV(N1,1),TC(N1,1),TK(2*N1+1,1)
C
C       TK22 = TK = TK11,  TK11 = 0.D0,  TK12 = 0.D0, TK21 = 0.D0
C
        DO 10 I=1,MDIM
        DO 10 J=1,MDIM
        TK(I+MDIM,J+MDIM) = TK(I,J)
        TK(I,J) = 0.D0
        TK(I,J+MDIM) = 0.D0
        TK(I+MDIM,J) = 0.D0
 10     CONTINUE
C
C       TK11 = -TMINV * TC
        DO 20 I=1,MDIM
        DO 20 J=1,MDIM
        DO 20 K=1,MDIM
        TK(I,J) = TK(I,J) - TMINV(I,K) * TC(K,J)
 20     CONTINUE
C
C       TK12 = -TMINV * TK
        DO 30 I=1,MDIM
        DO 30 J=1,MDIM
        DO 30 K=1,MDIM
        TK(I,J+MDIM) = TK(I,J+MDIM) - TMINV(I,K) * TK(K+MDIM,J+MDIM)
 30     CONTINUE
C
C       discard TK at TK22
        DO 40 I=MDIM+1,2*MDIM
        DO 40 J=MDIM+1,2*MDIM
        TK(I,J) = 0.D0
 40     CONTINUE
C
C       TK21 = I
        DO 50 I=1,MDIM
```

```
      TK(I+MDIM,I) = 1.D0
 50   CONTINUE
C
C     :::::place the filter part into the TK
      DO 60 I=1,2*MDIM
      TK(2*MDIM+1,I) = 0.D0
      TK(I,2*MDIM+1) = 0.D0
 60   CONTINUE
C
      TK(2*MDIM+1,2*MDIM+1) = -1/TIMEC
      RETURN
      END
C
C     ---------------------------------------------------------------
C     ---------------------------------------------------------------
C
      SUBROUTINE READ4(MDIM,Y,FX,FY,T,TFINAL,NSTEPS,TIMEC)
      REAL*8 Y(1),FX,FY,T,TFINAL,TIMEC
C
      DO 10 I=1,2*MDIM+1
      Y(I) = 0.D0
 10   CONTINUE
C
      PRINT*,'PLEASE ENTER THE CUTTING FORCE IN THE X DIRECTION'
      READ(*,*) FX
      PRINT*,'PLEASE ENTER THE CUTTING FORCE IN THE Y DIRECTION'
      READ(*,*) FY
      PRINT*,'PLEASE ENTER THE INITIAL TIME'
      READ(*,*) T
      PRINT*,'PLEASE ENTER THE FINAL TIME'
      READ(*,*) TFINAL
      PRINT*,'PLEASE ENTER THE NUMBER OF INTEGRATION STEPS'
      READ(*,*) NSTEPS
      PRINT*,'ENTER THE TIME CONSTANT FOR THE FILTER'
      READ(*,*) TIMEC
       PRINT*,'FX = ',FX,'   FY = ',FY
       PRINT*,'INITIAL TIME = ',T
       PRINT*,'FINAL TIME = ',TFINAL
       PRINT*, 'NUMBER OF STEPS = ',NSTEPS
       PRINT*,'TIME CONSTANT FOR THE FILTER IS = ',TIMEC
C
      WRITE(7,*) 'INITIAL CONDITIONS','TIME=',T
         WRITE(7,30) (Y(J), J=MDIM+1,2*MDIM,4)
 30      FORMAT(5G15.7)
C
      RETURN
      END
C
C     ---------------------------------------------------------------
C     ---------------------------------------------------------------
C
C
C     The forcing function in the following subroutine is obtained
C     throuh a random number generator.  Then it is filterd at a given
C     time constant.   Hence richness of the input is insured without
C     exciting too high frequencies.  The last equation in the
C     the system of equations is for the filter.
      SUBROUTINE DIFFUN(M1,T,Y,YDOT)
```

```
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1),YDOT(1)
C
      PARAMETER(N1=84)
      COMMON /A/TK(2*N1+1,1)
      COMMON /C/TIMEC,F(1)
       COMMON /BB/ICOUNT
C
      IF (MOD(ICOUNT,60) .EQ.0) PRINT*,'Y(M1) =',Y(M1)
       ICOUNT = ICOUNT+1
C
       DO 5 I=1,M1
       YDOT(I) = 0.D0
 5     CONTINUE
C
      DO 10 I=1,M1
      DO 10 J=1,M1
      YDOT(I) = YDOT(I) +TK(I,J) * Y(J)
 10   CONTINUE
C
      DO 20 I=1,M1-1
      YDOT(I) = YDOT(I) +F(I)*Y(M1)
 20   CONTINUE
C
      YDOT(M1) = YDOT(M1) + F(M1)
C
      RETURN
      END
C
C     ------------------------------------------
C     ------------------------------------------
      SUBROUTINE PEDERV(M1,T,Y,PD,NO)
      DIMENSION Y(168,1),PD(1)
      RETURN
      END
C
```

```
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION Y(1),YDOT(1)
C
      PARAMETER(N1=84)
      COMMON /A/TK(2*N1+1,1)
      COMMON /C/TIMEC,F(1)
       COMMON /BB/ICOUNT
C
      IF (MOD(ICOUNT,60) .EQ.0) PRINT*,'Y(M1) =',Y(M1)
       ICOUNT = ICOUNT+1
C
       DO 5 I=1,M1
       YDOT(I) = 0.D0
 5     CONTINUE
C
      DO 10 I=1,M1
      DO 10 J=1,M1
      YDOT(I) = YDOT(I) +TK(I,J) * Y(J)
 10   CONTINUE
C
      DO 20 I=1,M1-1
      YDOT(I) = YDOT(I) +F(I)*Y(M1)
 20   CONTINUE
C
      YDOT(M1) = YDOT(M1) + F(M1)
C
      RETURN
      END
C
C     ------------------------------------------
C     ------------------------------------------
      SUBROUTINE PEDERV(M1,T,Y,PD,NO)
      DIMENSION Y(168,1),PD(1)
      RETURN
      END
C
```

```
C NAASA  8.1.010 DGEAR    FTN   05-02-78     THE UNIV OF MICH COMP CTR
      SUBROUTINE DRIVE (N, T0, H0, Y0, TOUT, EPS, MF, INDEX)
C        THIS IS THE JANUARY 13, 1975 VERSION OF
C        GEAR, A PACKAGE FOR THE SOLUTION OF THE INITIAL VALUE
C        PROBLEM FOR SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS,
C                  DY/DT = F(Y,T),     Y = (Y(1),Y(2),...,Y(N)).
C        SUBROUTINE DRIVE IS A DRIVER ROUTINE FOR THE GEAR PACKAGE.
C
C                        REFERENCES
C       1.  A. C. HINDMARSH, GEAR.. ORDINARY DIFFERENTIAL EQUATION
C             SYSTEM SOLVER, UCID-30001 REV. 3, LAWRENCE LIVERMORE
C             LABORATORY, P.O.BOX 808, LIVERMORE, CA 94550, DEC. 1974.
C
C       2.  A. C. HINDMARSH, LINEAR MULTISTEP METHODS FOR ORDINARY
C             DIFFERENTIAL EQUATIONS.. METHOD FORMULATIONS,
C             STABILITY, AND THE METHODS OF NORDSIECK AND GEAR,
C             UCRL-51186 REV. 1, L.L.L., MARCH 1972.
C
C       3.  A. C. HINDMARSH, CONSTRUCTION OF MATHEMATICAL SOFTWARE,
C             PART III.. THE CONTROL OF ERROR IN THE GEAR PACKAGE
C             FOR ORDINARY DIFFERENTIAL EQUATIONS, UCID-30050 PART 3,
C             L.L.L., AUGUST 1972.
C
C-----------------------------------------------------------------------
C  THE ORIGINAL VERSION OF THIS PROGRAM WAS WRITTEN AT LLL BY
C  A. C. HINDMARSH FOR CDC COMPUTERS.  THE CDC VERSION WAS MODIFIED
C  FOR USE ON IBM COMPUTERS IN DOUBLE PRECISION AT ARGONNE
C  NATIONAL LABORATORY IN JANUARY 1975.
C-----------------------------------------------------------------------
C  DRIVE IS TO BE CALLED ONCE FOR EACH OUTPUT VALUE OF T, AND
C  IN TURN MAKES REPEATED CALLS TO THE CORE INTEGRATOR, STIFF.
C
C  THE INPUT PARAMETERS ARE..
C    N     =  THE NUMBER OF FIRST-ORDER DIFFERENTIAL EQUATIONS.
C               N CAN BE REDUCED, BUT NEVER INCREASED, DURING A PROBLEM.
C    T0    =  THE INITIAL VALUE OF T, THE INDEPENDENT VARIABLE
C               (USED ONLY ON FIRST CALL).
C    H0    =  THE NEXT STEP SIZE IN T (USED FOR INPUT ONLY ON THE
C               FIRST CALL).
C    Y0    =  A VECTOR OF LENGTH N CONTAINING THE INITIAL VALUES OF
C               Y (USED FOR INPUT ONLY ON FIRST CALL).
C    TOUT  =  THE VALUE OF T AT WHICH OUTPUT IS DESIRED NEXT.
C               INTEGRATION WILL NORMALLY GO SLIGHTLY BEYOND TOUT
C               AND THE PACKAGE WILL INTERPOLATE TO T = TOUT.
C    EPS   =  THE RELATIVE ERROR BOUND  (USED ONLY ON THE
C               FIRST CALL, UNLESS INDEX = -1).  SINGLE STEP ERROR
C               ESTIMATES DIVIDED BY YMAX(I) WILL BE KEPT LESS THAN
C               EPS IN ROOT-MEAN-SQUARE NORM (I.E. EUCLIDEAN NORM
C               DIVIDED BY DSQRT(N) ).  THE VECTOR YMAX OF
C               WEIGHTS IS COMPUTED IN DRIVE.  INITIALLY YMAX(I) IS
C               DABS(Y(I)), WITH A DEFAULT VALUE OF 1 IF Y(I) = 0
C               INITIALLY.  THEREAFTER, YMAX(I) IS THE LARGEST VALUE
C               OF DABS(Y(I)) SEEN SO FAR, OR THE INITIAL YMAX(I) IF
C               THAT IS LARGER.  TO ALTER EITHER OF THESE, CHANGE THE
C               APPROPRIATE STATEMENTS IN THE DO-LOOPS ENDING AT
C               STATEMENTS 10 AND 70 BELOW.
C    MF    =  THE METHOD FLAG  (USED ONLY ON FIRST CALL, UNLESS
C               INDEX = -1).  ALLOWED VALUES ARE 10, 11, 12, 13,
```

```
C               20, 21, 22, 23.  MF HAS TWO DECIMAL DIGITS, METH
C               AND MITER  (MF = 10*METH + MITER).
C               METH IS THE BASIC METHOD INDICATOR..
C                 METH = 1  MEANS THE ADAMS METHODS.
C                 METH = 2  MEANS THE STIFF METHODS OF GEAR, OR THE
C                           BACKWARD DIFFERENTIATION FORMULAS.
C               MITER IS THE ITERATION METHOD INDICATOR..
C                 MITER = 0 MEANS FUNCTIONAL ITERATION (NO PARTIAL
C                           DERIVATIVES NEEDED).
C                 MITER = 1 MEANS CHORD METHOD WITH ANALYTIC JACOBIAN.
C                           FOR THIS USER SUPPLIES SUBROUTINE
C                           PEDERV  (SEE DESCRIPTION BELOW).
C                 MITER = 2 MEANS CHORD METHOD WITH JACOBIAN CALCULATED
C                           INTERNALLY BY FINITE DIFFERENCES.
C                 MITER = 3 MEANS CHORD METHOD WITH JACOBIAN REPLACED
C                           BY A DIAGONAL APPROXIMATION BASED ON A
C                           DIRECTIONAL DERIVATIVE.
C    INDEX =  INTEGER USED ON INPUT TO INDICATE TYPE OF CALL,
C               WITH THE FOLLOWING VALUES AND MEANINGS..
C                  1    THIS IS THE FIRST CALL FOR THIS PROBLEM.
C                  0    THIS IS NOT THE FIRST CALL FOR THIS PROBLEM,
C                       AND INTEGRATION IS TO CONTINUE.
C                 -1    THIS IS NOT THE FIRST CALL FOR THE PROBLEM,
C                       AND THE USER HAS RESET N, EPS, AND/OR MF.
C                  2    SAME AS 0 EXCEPT THAT TOUT IS TO BE HIT
C                       EXACTLY (NO INTERPOLATION IS DONE).
C                       ASSUMES TOUT .GE. THE CURRENT T.
C                  3    SAME AS 0 EXCEPT CONTROL RETURNS TO CALLING
C                       PROGRAM AFTER ONE STEP.  TOUT IS IGNORED.
C               SINCE THE NORMAL OUTPUT VALUE OF INDEX IS 0,
C               IT NEED NOT BE RESET FOR NORMAL CONTINUATION.
C
C  AFTER THE INITIAL CALL, IF A NORMAL RETURN OCCURRED AND A NORMAL
C  CONTINUATION IS DESIRED, SIMPLY RESET TOUT AND CALL AGAIN.
C  ALL OTHER PARAMETERS WILL BE READY FOR THE NEXT CALL.
C  A CHANGE OF PARAMETERS WITH INDEX = -1 CAN BE MADE AFTER
C  EITHER A SUCCESSFUL OR AN UNSUCCESSFUL RETURN.
C
C  THE OUTPUT PARAMETERS ARE..
C    H0     =  THE STEP SIZE H USED LAST, WHETHER SUCCESSFULLY OR NOT.
C    Y0     =  THE COMPUTED VALUES OF Y AT T = TOUT.
C    TOUT   =  THE OUTPUT VALUE OF T.  IF INTEGRATION WAS SUCCESSFUL,
C               AND THE INPUT VALUE OF INDEX WAS NOT 3, TOUT IS
C               UNCHANGED FROM ITS INPUT VALUE.  OTHERWISE, TOUT
C               IS THE CURRENT VALUE OF T TO WHICH INTEGRATION
C               HAS BEEN COMPLETED.
C    INDEX =  INTEGER USED ON OUTPUT TO INDICATE RESULTS,
C               WITH THE FOLLOWING VALUES AND MEANINGS..
C          0    INTEGRATION WAS COMPLETED TO TOUT OR BEYOND.
C         -1    THE INTEGRATION WAS HALTED AFTER FAILING TO PASS THE
C               ERROR TEST EVEN AFTER REDUCING H BY A FACTOR OF
C               1.E10 FROM ITS INITIAL VALUE.
C         -2    AFTER SOME INITIAL SUCCESS, THE INTEGRATION WAS
C               HALTED EITHER BY REPEATED ERROR TEST FAILURES OR BY
C               A TEST ON EPS.  TOO MUCH ACCURACY HAS BEEN REQUESTED.
C         -3    THE INTEGRATION WAS HALTED AFTER FAILING TO ACHIEVE
C               CORRECTOR CONVERGENCE EVEN AFTER REDUCING H BY A
C               FACTOR OF 1.E10 FROM ITS INITIAL VALUE.
```

```
C         -4    IMMEDIATE HALT BECAUSE OF ILLEGAL VALUES OF INPUT
C               PARAMETERS.  SEE PRINTED MESSAGE.
C         -5    INDEX WAS -1 ON INPUT, BUT THE DESIRED CHANGES OF
C               PARAMETERS WERE NOT IMPLEMENTED BECAUSE TOUT
C               WAS NOT BEYOND T.  INTERPOLATION TO T = TOUT WAS
C               PERFORMED AS ON A NORMAL RETURN.  TO TRY AGAIN,
C               SIMPLY CALL AGAIN WITH INDEX = -1 AND A NEW TOUT.
C
C   IN ADDITION TO DRIVE, THE FOLLOWING ROUTINES ARE PROVIDED IN
C   THE PACKAGE..
C     INTERP(TOUT,Y,N0,Y0)  INTERPOLATES TO GET THE OUTPUT VALUES
C                           AT T = TOUT, FROM THE DATA IN THE Y ARRAY.
C     STIFF(Y,N0)  IS THE CORE INTEGRATOR ROUTINE.  IT PERFORMS A
C                           SINGLE STEP AND ASSOCIATED ERROR CONTROL.
C     COSET(METH,NQ,EL,TQ,MAXDER)  SETS COEFFICIENTS FOR USE IN
C                           THE CORE INTEGRATOR.
C     PSET(Y,N0,CON,MITER,IER)  COMPUTES AND PROCESSES THE JACOBIAN
C                           MATRIX J = DF/DY.
C     DEC(N,N0,A,IP,IER)  PERFORMS AN LU DECOMPOSITION ON A MATRIX.
C     SOL(N,N0,A,B,IP)  SOLVES LINEAR SYSTEMS A*X = B AFTER DEC
C                           HAS BEEN CALLED FOR THE MATRIX A.
C   NOTE.. PSET, DEC, AND SOL ARE CALLED ONLY IF MITER = 1 OR 2.
C
C   THE FOLLOWING ROUTINES ARE TO BE SUPPLIED BY THE USER..
C     DIFFUN(N,T,Y,YDOT)  COMPUTES THE FUNCTION YDOT = F(Y,T), THE
C                           RIGHT-HAND SIDE OF THE O.D.E.
C                           HERE Y AND YDOT ARE VECTORS OF LENGTH N.
C     PEDERV(N,T,Y,PD,N0)  COMPUTES THE N BY N JACOBIAN MATRIX OF
C                           PARTIAL DERIVATIVES, AND STORES IT IN PD
C                           AS AN N0 BY N0 ARRAY.  PD(I,J) IS TO BE
C                           SET TO THE PARTIAL DERIVATIVE OF YDOT(I)
C                           WITH RESPECT TO Y(J).  PEDERV IS CALLED
C                           ONLY IF MITER = 1.  OTHERWISE A DUMMY
C                           ROUTINE CAN BE SUBSTITUTED.
C
C  THE DIMENSIONS IN THE FOLLOWING DECLARATIONS ARE SET FOR A
C  MAXIMUM OF 20 EQUATIONS.  IF THE PACKAGE IS TO USED FOR A LARGER
C  VALUE OF N, THE DIMENSIONS SHOULD BE INCREASED ACCORDINGLY.  THE
C  DIMENSION OF PW BELOW MUST BE AT LEAST N**2 IF MITER = 1 OR 2,
C  BUT CAN BE REDUCED TO N IF MITER = 3, OR TO 1 IF MITER = 0.
C  THE DIMENSIONS OF YMAX, ERROR, SAVE1, SAVE2, IPIV, AND THE FIRST
C  DIMENSION OF Y SHOULD ALL BE AT LEAST N.  THE COLUMN LENGTH OF
C  THE Y ARRAY AS USED ELSEWHERE IS N0, NOT 20.  THE ROW LENGTH OF Y
C  CAN BE REDUCED FROM 13 TO 6 IF METH = 2.
C  THE IPIV ARRAY IS USED ONLY IF MITER IS 1 OR 2.
C
C  THE COMMON BLOCK GEAR9 CAN BE ACCESSED EXTERNALLY BY THE USER
C  IF DESIRED.  IT CONTAINS THE STEP SIZE LAST USED (SUCCESSFULLY),
C  THE ORDER LAST USED (SUCCESSFULLY), THE NUMBER OF STEPS TAKEN
C  SO FAR, THE NUMBER OF F EVALUATIONS (DIFFUN CALLS) SO FAR,
C  AND THE NUMBER OF JACOBIAN EVALUATIONS SO FAR.
C
C  IN THE FOLLOWING DATA STATEMENT, SET..
C    LOUT   =  THE LOGICAL UNIT NUMBER FOR THE OUTPUT OF MESSAGES
C              DURING THE INTEGRATION.
C-----------------------------------------------------------------------
C
C     NOTE THE FOLLOWING PARAMETER CHANGE IS FOR 20 ELEMENT SOLUTION
```

```
C     COMMON BLOCKS ARE CHANGED ACCORDINGLY FROM 20 TO 2*N2+1
      PARAMETER (N2=82)
C-----------------------------------------------------------------------
-
      INTEGER N,MF,INDEX
      INTEGER NC,MFC,KFLAG,JSTART,IPIV,NSQ,NQUSED,NSTEP,NFE,NJE
      INTEGER LOUT,I,N0,NHCUT,KGO
      DOUBLE PRECISION T0,H0,Y0,TOUT,EPS
      DOUBLE PRECISION T,H,HMIN,HMAX,EPSC,UROUND,YMAX,ERROR,
     1   SAVE1,SAVE2,PW,EPSJ,HUSED
      DOUBLE PRECISION Y,TOUTP,AYI,D
      COMMON /GEAR1/ T,H,HMIN,HMAX,EPSC,UROUND,NC,MFC,KFLAG,JSTART
      COMMON /GEAR2/ YMAX(2*N2+1)
      COMMON /GEAR3/ ERROR(2*N2+1)
      COMMON /GEAR4/ SAVE1(2*N2+1)
      COMMON /GEAR5/ SAVE2(2*N2+1)
      COMMON /GEAR6/ PW(400)
      COMMON /GEAR7/ IPIV(2*N2+1)
      COMMON /GEAR8/ EPSJ,NSQ
      COMMON /GEAR9/ HUSED,NQUSED,NSTEP,NFE,NJE
      DATA LOUT/6/
      DIMENSION Y0(N)
      DIMENSION Y(20,13)
      IF (INDEX .EQ. 0) GO TO 20
      IF (INDEX .EQ. 2) GO TO 25
      IF (INDEX .EQ. -1) GO TO 30
      IF (INDEX .EQ. 3) GO TO 40
      IF (INDEX .NE. 1) GO TO 430
      IF (EPS .LE. 0.D0) GO TO 400
      IF (N .LE. 0) GO TO 410
      IF ((T0-TOUT)*H0 .GE. 0.D0) GO TO 420
C-----------------------------------------------------------------------
C  IF INITIAL VALUES OF YMAX OTHER THAN THOSE SET BELOW ARE DESIRED,
C  THEY SHOULD BE SET HERE.  ALL YMAX(I) MUST BE POSITIVE.
C  IF VALUES FOR HMIN OR HMAX, THE BOUNDS ON DABS(H), OTHER THAN
C  THOSE BELOW ARE DESIRED, THEY SHOULD BE SET BELOW.
C  IN THE FOLLOWING STATEMENT, SET..
C    UROUND =  THE UNIT ROUNDOFF OF THE MACHINE, I.E. THE SMALLEST
C              POSITIVE U SUCH THAT 1. + U .NE. 1. ON THE MACHINE.
C-----------------------------------------------------------------------
      UROUND = 2.22D-16
      DO 10 I = 1,N
        YMAX(I) = DABS(Y0(I))
        IF (YMAX(I) .EQ. 0.D0) YMAX(I) = 1.D0
 10     Y(I,1) = Y0(I)
      NC = N
      T = T0
      H = H0
      IF ((T+H) .EQ. T) WRITE(LOUT,15)
 15   FORMAT(35H WARNING..  T + H = T ON NEXT STEP.)
      HMIN = DABS(H0)
      HMAX = DABS(T0-TOUT)*10.D0
      EPSC = EPS
      MFC = MF
      JSTART = 0
      N0 = N
      NSQ = N0*N0
      EPSJ = DSQRT(UROUND)
```

```
      NHCUT = 0
      GO TO 50
C
C  TOUTP IS THE PREVIOUS VALUE OF TOUT FOR USE IN HMAX.----------------
 20   HMAX = DABS(TOUT-TOUTP)*10.
      GO TO 80
C
 25   HMAX = DABS(TOUT-TOUTP)*10.D0
      IF ((T-TOUT)*H .GE. 0.D0) GO TO 500
      GO TO 85
C
 30   IF ((T-TOUT)*H .GE. 0.D0) GO TO 440
      JSTART = -1
      NC = N
      EPSC = EPS
      MFC = MF
C
 40   IF ((T+H) .EQ. T) WRITE(LOUT,15)
C
 50   CALL STIFF (Y, N0)
C
      KGO = 1 - KFLAG
      GO TO (60, 100, 200, 300), KGO
C  KFLAG  =   0,  -1,  -2,  -3
C
 60   CONTINUE
C-----------------------------------------------------------------------
C  NORMAL RETURN FROM INTEGRATOR.
C
C  THE WEIGHTS YMAX(I) ARE UPDATED.  IF DIFFERENT VALUES ARE DESIRED,
C  THEY SHOULD BE SET HERE.  A TEST IS MADE FOR EPS BEING TOO SMALL
C  FOR THE MACHINE PRECISION.
C
C  ANY OTHER TESTS OR CALCULATIONS THAT ARE REQUIRED AFTER EVERY
C  STEP SHOULD BE INSERTED HERE.
C
C  IF INDEX = 3, Y0 IS SET TO THE CURRENT Y VALUES ON RETURN.
C  IF INDEX = 2, H IS CONTROLLED TO HIT TOUT (WITHIN ROUNDOFF
C  ERROR), AND THEN THE CURRENT Y VALUES ARE PUT IN Y0 ON RETURN.
C  FOR ANY OTHER VALUE OF INDEX, CONTROL RETURNS TO THE INTEGRATOR
C  UNLESS TOUT HAS BEEN REACHED.  THEN INTERPOLATED VALUES OF Y ARE
C  COMPUTED AND STORED IN Y0 ON RETURN.
C  IF INTERPOLATION IS NOT DESIRED, THE CALL TO INTERP SHOULD BE
C  REMOVED AND CONTROL TRANSFERRED TO STATEMENT 500 INSTEAD OF 520.
C-----------------------------------------------------------------------
      D = 0.D0
      DO 70 I = 1,N
        AYI = DABS(Y(I,1))
        YMAX(I) = DMAX1(YMAX(I), AYI)
 70     D = D + (AYI/YMAX(I))**2
      D = D*(UROUND/EPS)**2
      IF (D .GT. DFLOAT(N)) GO TO 250
      IF (INDEX .EQ. 3) GO TO 500
      IF (INDEX .EQ. 2) GO TO 85
 80   IF ((T-TOUT)*H .LT. 0.D0) GO TO 40
      CALL INTERP (TOUT, Y, N0, Y0)
      GO TO 520
 85   IF (((T+H)-TOUT)*H .LE. 0.D0) GO TO 40
```

```
      IF (DABS(T-TOUT) .LE. 100.D0*UROUND*HMAX) GO TO 500
      IF ((T-TOUT)*H .GE. 0.D0) GO TO 500
      H = (TOUT - T)*(1.D0 - 4.D0*UROUND)
      JSTART = -1
      GO TO 40
C-----------------------------------------------------------------------
C  ON AN ERROR RETURN FROM INTEGRATOR, AN IMMEDIATE RETURN OCCURS IF
C  KFLAG = -2, AND RECOVERY ATTEMPTS ARE MADE OTHERWISE.
C  TO RECOVER, H AND HMIN ARE REDUCED BY A FACTOR OF .1 UP TO 10
C  TIMES BEFORE GIVING UP.
C-----------------------------------------------------------------------
 100  WRITE (LOUT,105) T
 105  FORMAT(//35H KFLAG = -1 FROM INTEGRATOR AT T = ,E16.8/
     1   39H  ERROR TEST FAILED WITH DABS(H) = HMIN/)
 110  IF (NHCUT .EQ. 10) GO TO 150
      NHCUT = NHCUT + 1
      HMIN = HMIN*.1D0
      H = H*.1D0
      WRITE (LOUT,115) H
 115  FORMAT(24H  H HAS BEEN REDUCED TO ,E16.8,
     1   26H  AND STEP WILL BE RETRIED//)
      JSTART = -1
      GO TO 40
C
 150  WRITE (LOUT,155)
 155  FORMAT(//44H PROBLEM APPEARS UNSOLVABLE WITH GIVEN INPUT//)
      GO TO 500
C
 200  WRITE (LOUT,205) T,H
 205  FORMAT(//35H KFLAG = -2 FROM INTEGRATOR AT T = ,E16.8,5H  H =,
     1  E16.8/52H  THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED//)
      GO TO 500
C
 250  WRITE (LOUT,255) T
 255  FORMAT(//37H INTEGRATION HALTED BY DRIVER AT T = ,E16.8/
     1   56H  EPS TOO SMALL TO BE ATTAINED FOR THE MACHINE PRECISION/)
      KFLAG = -2
      GO TO 500
C
 300  WRITE (LOUT,305) T
 305  FORMAT(//35H KFLAG = -3 FROM INTEGRATOR AT T = ,E16.8/
     1   45H  CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED/)
      GO TO 110
C
 400  WRITE (LOUT,405)
 405  FORMAT(//28H ILLEGAL INPUT.. EPS .LE. 0.//)
      INDEX = -4
      RETURN
C
 410  WRITE (LOUT,415)
 415  FORMAT(//25H ILLEGAL INPUT.. N .LE. 0//)
      INDEX = -4
      RETURN
C
 420  WRITE (LOUT,425)
 425  FORMAT(//36H ILLEGAL INPUT.. (T0-TOUT)*H .GE. 0.//)
      INDEX = -4
      RETURN
```

```
C
 430  WRITE (LOUT,435) INDEX
 435  FORMAT(//24H ILLEGAL INPUT.. INDEX =,I5//)
      INDEX = -4
      RETURN
C
 440  WRITE(LOUT,445) T,TOUT,H
 445  FORMAT(//44H INDEX = -1 ON INPUT WITH (T-TOUT)*H .GE. 0./
     1   4H T =,E16.8,9H   TOUT =,E16.8,6H   H =,E16.8/
     1   44H INTERPOLATION WAS DONE AS ON NORMAL RETURN./
     2   41H DESIRED PARAMETER CHANGES WERE NOT MADE.)
      CALL INTERP (TOUT, Y, N0, Y0)
      INDEX = -5
      RETURN
C
 500  TOUT = T
      DO 510 I = 1,N
 510    Y0(I) = Y(I,1)
 520  INDEX = KFLAG
      TOUTP = TOUT
      H0 = HUSED
      IF (KFLAG .NE. 0) H0 = H
      RETURN
C----------------------- END OF SUBROUTINE DRIVE ----------------------
```

APPENDIX B

PROGRAM LISTING FOR THE ROUTINES THAT CALCULATES THE DRILL CROSS SECTIONAL PROPERTIES

```
C
C
C            ********************************************************
C
C
C                    AUTOMATIC MESH GENERATION PROGRAM
C
C
C           ************************************************************
C
C    Purposes:    This program calculates the IXX, IYY and area
C    ________    of the drill cross-section after generating the
C                boundary mesh. 5 points that specify the drill
C                cross-section is input to the program.
C
C
C Purpose 1: Program generates the mesh points along the boundary
C ---------    using the following subroutines:
C               READ1, BOUNEQ, DIVL1, DIVL2, DIVL3, DIVL4
C READ1: Reads the 5 points thst specify the drill boundary
C BOUNEQ: Calculates the equations along the boundary
C DIVL1:  Generates the mesh points along the first boundary
C DIVL2:   >>  >>      >>       >>       >>   second boundary
C DIVL3:   >>  >>      >>       >>       >>   third boundary
C DIVL4:   >>  >>      >>       >>       >>   fourth boundary
C
C Purpose 2: Program generates the inner triangular mesh
C ---------    using Subroutine mesh This subroutine calls the
C              following subroutines:
C ASSIGN: Assigns the boundary mesh vector to X,Y matrices
C SOR:    Generates the inner mesh using Poisson's equation
C CONNEC: Calculates the triangular element connectivities
C PLOT1:  Plots the triangular grid
C PNUM:   Numbers the elements on the plot
C
C               ***********************************
C
C     Arguments:
C        XA,YA:  The vectors that keep 5 points of drill geometry
C        NN,MM:  Number of mesh divisions along the boundaries
C A,B,C,D,E,F,R: Coefficients of boundary equations
C        XB,YB:  Stores the drill boundary mesh coordinates
C           KJ:  Controls the boundary mesh storage order
C        XE,YE:  Drill triangular mesh point coordinates
C          IJK:  Element connectivities
C          NEL:  Number of elements
C        NNODE:  Number of nodes
C        S1,S2:  Principal Area moment of inertias
C         AREA:  Drill cross-sectional area
```

```
C              ------------------------------------
C
C
C       SUBROUTINE COEFF(S1,S2,AREA)
C
        IMPLICIT REAL*8(A-H, O-Z)
        DIMENSION XA(5), YA(5), XB(400),YB(400),
     &           XE(10000),YE(10000),IJK(3,1000)
C
        KJ=0
C
        CALL READ1(NN,MM,XA,YA)
C
        CALL BOUNEQ(XA,YA,A,B,C,D,E,F,R)
C
        CALL DIVL1(XA,YA,XB,YB,KJ,NN,MM,A,B,C)
C
        CALL DIVL2(XA,YA,XB,YB,KJ,NN,MM)
C
        CALL DIVL3(XA,YA,XB,YB,KJ,NN,MM,D,E,F)
C
        CALL DIVL4(XA,YA,XB,YB,KJ,NN,MM,R)
C
        WRITE(6,1001)
1001    FORMAT('**************XB,YB****************')
        DO 1003 J=1,KJ
        WRITE(6,1002) J,XB(J),YB(J)
1002    FORMAT(5X,I3,2G20.10)
1003    CONTINUE
C
        CALL MESH(XB,YB,NN,MM,XE,YE,IJK,N,M,NEL,NNODE)
C
        CALL GAUSS(XE,YE,IJK,NEL,NNODE,SXX,SYY,AREA)
        STOP
        END
C
C
C     ----------------------------------------
C
C                  SUBROUTINE READ1
C
C        Purpose: This subroutine reads the 5 points that is
C                 required to define the drill geometry and the
C                 number of divisions in x and y directions.
C        Arguments:
C   XA,YA:   The vectors that keep the 5 points.
C   NN,MM:   Number of divisions along the boundary in X & Y
C            directions.
C
        SUBROUTINE READ1(NN,MM,XA,YA)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION XA(5),YA(5)
        READ(5,100) NN,MM
 100     FORMAT(2I5)
        WRITE(6,102)
102     FORMAT('*********  NN,MM  *******')
        WRITE(6,100)NN,MM
C
C
```

```
        DO 101 J=1,5
        XA(J)=0.D0
        YA(J)=0.D0
 101     CONTINUE
C
        WRITE(6,106)
106     FORMAT('********  XA,YA  ********')
        DO 105 J=1,5
        READ(5,103) XA(J),YA(J)
        WRITE(6,103)XA(J),YA(J)
 103     FORMAT(2F10.5)
 105     CONTINUE
C
        RETURN
        END
C
C
C      ---------------------------------------------
C
C                SUBROUTINE BOUNEQ
C
C         Purpose:   Calculates the boundary equations
C                    using the 5 points specified along the boundary
C         Arguments:
C        XA,YA:  5 points that specify the drill boundary
C        A,B,C:  Coefficients of the first boundary equation which
C                is specified as a circle.
C        D,E,F:  Coefficients of the third boundary equation which
C                is specified as a parabola.
C        R:      Radius of the circle that specifies the fourth
C                boundary equation.
C         Local Variables:
C        DET, F1,F2,F3
C
        SUBROUTINE BOUNEQ(XA,YA,A,B,C,D,E,F,R)
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION XA(5),YA(5)
        A=0.D0
        B=0.D0
        C=0.D0
        D=0.D0
        E=0.D0
        F=0.D0
        R=0.D0
        F1=0.D0
        F2=0.D0
        F3=0.D0
C     CALCULATE THE PARAMETERS A,B,C
        DET=XA(2)*YA(3)-XA(3)*YA(2)-XA(1)*YA(3)+XA(3)*YA(1)
     &  +XA(1)*YA(2)-YA(1)*XA(2)
        F1=-( XA(1)**2 + YA(1)**2)
        F2=-( XA(2)**2 + YA(2)**2 )
        F3=-( XA(3)**2 + YA(3)**2 )
C
        A=((YA(3)-YA(1))*(F2-F1)+(YA(1)-YA(2))*(F3-F1))/DET
        B=((XA(1)-XA(3))*(F2-F1)+(XA(2)-XA(1))*(F3-F1))/DET
        C= F1-A*XA(1)-B*YA(1)
C       CALCULATE PARAMETERS D,E,F
```

```
      D=((YA(4)-YA(5))+(2.*XA(3)+A)/(2.*YA(3)+B)*(XA(4)-XA(5)))/
     &     (-XA(4)**2-XA(5)**2+2.*XA(4)*XA(5))
      E=-(2.*XA(3)+A)/(2.*YA(3)+B)-2.*XA(4)*D
      F=YA(4)-E*XA(4)-D*XA(4)**2
C     CALCULATE R
      R=(XA(5)**2+YA(5)**2)**.5
      RETURN
      END
C
C
C     ------------------------------------------------
C
C                SUBROUTINE DIVL1
C
C        Purpose:  Generates the boundary mesh for the first
C                  boundary.
C        Arguments:
C        XA,YA:    5 points that specify the drill boundary
C        XB,YB:    Stores the boundary mesh coordinates genmerated
C        KJ:       Controls the boundary mesh storage
C        NN,MM:    Number of divisions along the boundaries
C        A,B,C:    Coefficients of the first boundary equation.
C
C
      SUBROUTINE DIVL1(XA,YA,XB,YB,KJ,NN,MM,A,B,C)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XA(5),YA(5),XB(400),YB(400)
C
      WRITE(6,200) XA(1),YA(1)
200   FORMAT(2F10.5)
      KJ=1
      XB(KJ)=XA(1)
      YB(KJ)=YA(1)
C
      X0=-A/2.
      Y0=-B/2.
      R=DSQRT(X0**2+Y0**2-C)
      TETHA1=DATAN((YA(1)-Y0)/(XA(1)-X0))
      TETHA2=DATAN((YA(3)-Y0)/(X0-XA(3)))
      PI=3.141592654D0
      BETA=PI-(TETHA2+TETHA1)
      DBETA1=BETA/(NN-1)
      BETA1=TETHA1
C
      N1=NN-2
      DO 205 J=1,N1
      BETA1=BETA1+DBETA1
      XS=X0+R*DCOS(BETA1)
      YS=Y0+R*DSIN(BETA1)
      WRITE(6,200) XS,YS
      KJ=KJ+1
      XB(KJ)=XS
      YB(KJ)=YS
205   CONTINUE
      RETURN
      END
C
C     ----------------------------------------
```

```
C
C               SUBROUTINE DIVL2
C
C           Purpose:  Generate the boundary mesh for the
C                     second boundary.
C           Variables:
C           XA,YA:    5 points that specify the drill geometry
C           XB,YB:    Boundary mesh
C           KJ:       Controls the storage of boundary mesh
C           NN,MM:    Number of divisions along the boundaries
C
C
          SUBROUTINE DIVL2(XA,YA,XB,YB,KJ,NN,MM)
          IMPLICIT REAL*8(A-H,O-Z)
          DIMENSION XA(5),YA(5),XB(400),YB(400)
C
          DX2=(XA(4)-XA(3))/(MM-1)
          WRITE(6,300) XA(3),YA(3)
          KJ=KJ+1
          XB(KJ)=XA(3)
          YB(KJ)=YA(3)
 300      FORMAT(2F10.5)
          XS=XA(3)
          YS=YA(3)
          M1=MM-1
C
          DO 305 J=1,M1
          XS=XS+DX2
          WRITE(6,300) XS,YS
          KJ=KJ+1
          XB(KJ)=XS
          YB(KJ)=YS
 305      CONTINUE
          RETURN
          END
C
C       ------------------------------------------------
C
C            SUBROUTINE DIVL3
C
C          Purpose:  Generates the boundary mesh for the
C                    third boundary
C
C           Variables:
C           XA,YA:    5 points that specify the drill geometry
C           XB,YB:    Boundary mesh
C           KJ:       Controls the storage of boundary mesh
C           NN,MM:    Number of divisions along the boundaries
C           D,E,F:    Coefficients of the third boundary equation
C
C
          SUBROUTINE DIVL3(XA,YA,XB,YB,KJ,NN,MM,D,E,F)
          IMPLICIT REAL*8(A-H,O-Z)
          DIMENSION XA(5),YA(5),XB(400),YB(400)
C
          DY3=(YA(5)-YA(4))/(NN-1)
          YS=YA(4)
          N1=NN-2
```

```
C
      DO 400 J=1,N1
      YS=YS+DY3
      E1=E/D
      D1=(F-YS)/D
      DET=DABS(E1)**2.-4.*D1
      XS=(-E1+DET**0.5)/2.
      WRITE(6,405) XS,YS
      KJ=KJ+1
      XB(KJ)=XS
      YB(KJ)=YS
 405  FORMAT(2F10.5)
 400  CONTINUE
C
      RETURN
      END
C
C
C     ------------------------------------------
C
C              SUBROUTINE DIVL4
C
C     Purpose:  Generates the boundary mesh for the
C               fourth boundary
C
C       Variables:
C       XA,YA:    5 points that specify the drill geometry
C       XB,YB:    Boundary mesh
C       KJ:       Controls the storage of boundary mesh
C       NN,MM:    Number of divisions along the boundaries
C       R:        Radius of the fourth boundary equation
C
      SUBROUTINE DIVL4(XA,YA,XB,YB,KJ,NN,MM,R)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XA(5),YA(5),XB(400),YB(400)
C
      WRITE(6,500) XA(5),YA(5)
      KJ=KJ+1
      XB(KJ)=XA(5)
      YB(KJ)=YA(5)
 500  FORMAT(2F10.5)
C
      TETHA1=DATAN(YA(1)/XA(1))
      PI2=3.141592654D0/2.
      BETA=PI2-TETHA1
      BETAD=BETA/(MM-1)
      BETA1=0.D0
C
      M1=MM-1
      DO 505 J=1,M1
      BETA1=BETA1+BETAD
      XS=R*DSIN(BETA1)
      YS=R*DCOS(BETA1)
      WRITE(6,500) XS,YS
      KJ=KJ+1
      XB(KJ)=XS
      YB(KJ)=YS
 505  CONTINUE
```

```
      RETURN
      END
C
C     ------------------------------------------------
C
C                          SUBROUTINE MESH
C                          ---------------
C
C
C     Purpose: This program is for the finite element grid genera-
C              tion by using the Poisson's equations(in the  rec-
C              tangular coordinate form).
C
C     Variables:
C                       NEN=the number of nodes within one element
C                       FACTOR=the multiplication factor of the plot
C                       W=the constant in the SOR method
C                       MN=the upper limit of the iteration
C                       TOL=the tolerance of the iteration
C                       N=Number of divisions in X direction
C                       M=Number of divisions in Y direction
C                       NEL=Total number or elements
C                       NNODE=Total number of nodes
C
      SUBROUTINE MESH(XB,YB,NN,MM,XE,YE,IJK,N,M,NEL,NNODE)
C
      REAL*8 X,Y,XE,YE,XB,YB
      REAL*4 XEP,YEP,XP,YP
      DIMENSION X(100,100),Y(100,100),XE(10000),YE(10000)
      DIMENSION XEP(10000),YEP(10000),XP(100,100),YP(100,100)
     & ,IJK(3,1000),XB(400),YB(400)
C
      READ(5,109) N,M
  109 FORMAT(3I5)
      IF((N+M).NE.(NN+MM)) GOTO 111
      NODE=3
      FACTOR=1.
      NELD=1
C
C     Assign the boundary mesh to X & Y matrices
      CALL ASSIGN(XB,YB,X,Y,N,M)
C
C     Compute the mesh points for X(I,J),Y(I,J):
      CALL SOR(X,Y,N,M)
C
C     Get the element connectivities:
      CALL CONNEC(X,Y,XE,YE,IJK,N,M,NEL,NODE,NNODE)
C
C     Assign double precission X,Y,XE,YE arrays to
C     to  XP,YP,XEP,YEP single precision arrays
C     for ploting purposes.
C
      DO 80 I=1,N
      DO 80 J=1,M
      XP(I,J)=X(I,J)
      YP(I,J)=Y(I,J)
80    CONTINUE
C
```

```
      MAX=N*M
      DO 90 I=1,MAX
      XEP(I)=XE(I)
      YEP(I)=YE(I)
90    CONTINUE
C
C     Plot the grids:
      CALL PLOT1(XP,YP,N,M,NODE,FACTOR,XEP,YEP,IJK,NEL,NELD)
C
      CALL PNUM(IJK,XEP,YEP,3,NEL,NNODE,NELD)
C
      WRITE(8,105) NNODE,NEL
  105 FORMAT(' ',2I5)
      DO 108 JJ=1,NNODE
      WRITE(8,106) JJ, XE(JJ),YE(JJ)
  106 FORMAT(' ',I5,2F10.4)
  108 CONTINUE
C
      WRITE(8,107) (II,(IJK(I,II),I=1,NODE),II=1,NEL)
  107 FORMAT(' ',4I5)
C
      GOTO 113
  111 WRITE(8,112)
  112 FORMAT('WARNING! ** NN+MM .NE. N+M ')
  113 CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------
C
C               SUBROUTINE SOR
C
C
C     Purpose: This routine provides the solution to the Poisson's
C              equations by using the SOR iteration method.
C
C
      SUBROUTINE SOR(X,Y,N,M)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XO(100,100),X(100,100)
      DIMENSION YO(100,100),Y(100,100)
C
      W=1.24
      MN=400
      TOL=0.01
      NN=N-1
      MM=M-1
C
C     Input the initial approximations XO(I,J),YO(I,J):
C
      DO 719 I=1,N
      DO 719 J=1,M
         XO(I,1)=X(I,1)
         XO(I,M)=X(I,M)
         XO(N,J)=X(N,J)
         XO(1,J)=X(1,J)
  719 CONTINUE
```

```
C
      DO 855 I=1,N
      DO 855 J=1,M
         YO(I,1)=Y(I,1)
         YO(I,M)=Y(I,M)
         YO(N,J)=Y(N,J)
         YO(1,J)=Y(1,J)
  855 CONTINUE
C
C     Input XO(I,J),YO(I,J) for the internal points:
C
      DO 750 I=2, NN
      DO 750 J=2, MM
         XO(I,J)=2.0+J*0.1
         YO(I,J)=0.0+J*0.2
  750 CONTINUE
C
C     This sets an iteration limit.
      DO 800 K=1,MN
      DO 700 I=2, NN
      DO 600 J=2, MM
C
         AF=0.+1.*(XO(I,J+1)*XO(I,J+1)
     1      -2.*XO(I,J+1)*XO(I,J-1)+XO(I,J-1)*XO(I,J-1))
     2      +1.*(YO(I,J+1)*YO(I,J+1)
     1      -2.*YO(I,J+1)*YO(I,J-1)+YO(I,J-1)*YO(I,J-1))
C
         BF=(-2.)*(XO(I+1,J)-XO(I-1,J))*(XO(I,J+1)-XO(I,J-1))
     1      -2.*(YO(I+1,J)-YO(I-1,J))*(YO(I,J+1)-YO(I,J-1))
C
         GF=0.+1.*(XO(I+1,J)*XO(I+1,J)
     1      -2.*XO(I+1,J)*XO(I-1,J)+XO(I-1,J)*XO(I-1,J))
     2      +(0.+1.)*(YO(I+1,J)*YO(I+1,J)
     1      -2.*YO(I+1,J)*YO(I-1,J)+YO(I-1,J)*YO(I-1,J))
C
         AIJ=1./(2.*(AF+GF))
C
         X(I,J)=(1.-W)*XO(I,J)+(W*AIJ)*(AF*X(I-1,J)
     1          -0.25*BF*(X(I-1,J+1)-X(I-1,J-1))+GF*X(I,J-1)
     2          +AF*XO(I+1,J)+0.25*BF*(XO(I+1,J+1)-XO(I+1,J-1))
     3          +GF*XO(I,J+1))
         Y(I,J)=(1.-W)*YO(I,J)+(W*AIJ)*(AF*Y(I-1,J)
     1          -0.25*BF*(Y(I-1,J+1)-Y(I-1,J-1))+GF*Y(I,J-1)
     2          +AF*YO(I+1,J)+0.25*BF*(YO(I+1,J+1)-YO(I+1,J-1))
     3          +GF*YO(I,J+1))
C
  600 CONTINUE
  700 CONTINUE
C
      DO 500 I=1,N
      DO 500 J=1,M
C
      IF(DABS(X(I,J)-XO(I,J)).GT.TOL) GO TO 890
      IF(DABS(Y(I,J)-YO(I,J)).GT.TOL) GO TO 890
C
  500 CONTINUE
C
      GO TO 901
```

```
  890 DO 800 I=2,NN
      DO 800 J=2,MM
         XO(I,J)=X(I,J)
         YO(I,J)=Y(I,J)
  800 CONTINUE
C
      WRITE(6,111)
  111 FORMAT('-','EXCEEDS THE ITERATION LIMIT MN')
C
C     Transfer the values of X(I,J) to Y(I,J):
C
  901 KK=1
      WRITE(6,109) K
  109 FORMAT('-','THE NUMBER OF ITERATIONS=',I4)
C
      RETURN
      END
C
C-------------------------------------------------------------
C
C             SUBROUTINE PLOT1
C
C
C     Purpose: This routine plots the grids generated by SOR.
C
C
      SUBROUTINE PLOT1(X,Y,N,M,NEN,FACTOR,XE,YE,IJK,NE,NELD)
C
      DIMENSION X(100,100),Y(100,100),XX(100),YY(100)
      DIMENSION XP(100),YP(100),XP1(100),YP1(100)
      DIMENSION XE(10000),YE(10000),IJK(3,1000)
C
C     CALL PGRID(0.0,0.0,8.5,11.0,1,1)
C     CALL PAXIS(2.0,1.65,'X-AXIS (INCHES)',-15,6.0,0.0,0.0,1.0,1.0)
C     CALL PAXIS(2.0,1.65,'Y-AXIS (INCHES)',15,6.0,90.0,0.0,1.0,1.0)
C
      DO 400 I=1,N
      DO 500 J=1,M
         XX(J)=X(I,J)
         YY(J)=Y(I,J)
  500 CONTINUE
      CALL PLTOFS(0.0,FACTOR,0.0,FACTOR,2.0,2.0)
      CALL PLINE(XX,YY,M,1,0,0,1)
  400 CONTINUE
C
      DO 550 J=1,M
      DO 450 I=1,N
         XX(I)=X(I,J)
         YY(I)=Y(I,J)
  450 CONTINUE
      CALL PLTOFS(0.0,FACTOR,0.0,FACTOR,2.0,2.0)
      CALL PLINE(XX,YY,N,1,0,0,1)
  550 CONTINUE
C
      CALL PSYM(2.0,1.2,0.17,'MESH GENERATED BY ELLIPTIC
EQUATIONS',0.,35,0)
C
C     Plot the generated elements:
```

```
      IF(NEN.NE.3) GO TO 900
C
         NN=N-1
      DO 600 I=1,NN
         IC=I+1
         IIC=IC
      DO 650 J=1,IIC
         XP(J)=X(IC-J+1,J)
         YP(J)=Y(IC-J+1,J)
         IF (IIC.GE.M) GO TO 779
         XP1(J)=X(N-J+1,M+J-I-1)
         YP1(J)=Y(N-J+1,M+J-I-1)
      GO TO 650
  779    IIC=M
  650 CONTINUE
C
      CALL PLTOFS(0.0,FACTOR,0.0,FACTOR,2.0,2.0)
      CALL PLINE(XP,YP,IIC,1,0,0,1)
      IF(IIC.GE.M) GO TO 600
      CALL PLINE(XP1,YP1,IIC,1,0,0,1)
  600 CONTINUE
C
      GO TO 999
  900    NEEE=(N-1)*(M-1)
C
  999 CALL PSYM(3.0,0.8,0.15,'THE NUMBER OF ELEMENTS =',0.0,24,0)
      RETURN
      END
C
C ---------------------------------------------------------------
C
C               SUBROUTINE CONNEC
C
C
C     Purpose: This routine defines the element connectivities for
C              the generated elements.
C
C
      SUBROUTINE CONNEC(X,Y,XE,YE,IJK,N,M,NE,NEN,NNODE)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(100,100),Y(100,100)
      DIMENSION IJK(3,1000),XE(10000),YE(10000)
C
      NN=N-1
      MM=M-1
C
      DO 300 J=1,M
      DO 300 I=1,N
         II=I+(J-1)*N
         XE(II)=X(I,J)
         YE(II)=Y(I,J)
  300 CONTINUE
      NNODE=II
C
C
      DO 500 I=1,MM
      DO 500 J=2,N
```

```
          II=J+(I-1)*N
          NE2=2*(II-1)-(I-1)*2
          NE1=NE2-1
C
          IJK(1,NE1)=II-1
          IJK(2,NE1)=II
          IJK(3,NE1)=II+N-1
C
          IJK(1,NE2)=II
          IJK(2,NE2)=II+N
          IJK(3,NE2)=II+N-1
C
          NE=NE2
  500 CONTINUE
C
      WRITE(6,101)    (JJ,(IJK(I,JJ),I=1,3),JJ=1,NE)
  101 FORMAT(' ','ELEMENT NO.',I4,10X,3I8)
C
      GO TO 709
C
  709 WRITE(6,103) NE
  103 FORMAT('-','THE NUMBER OF ELEMENTS = ',I8)
C
      RETURN
      END
C
C-       ------------------------------------------------------
C                SUBROUTINE ASSIGN
C
C                This program assigns the boundary mesh
C           XB,YB, to the matrices X,Y where the
C           inner mesh be generated later on.
C
C
      SUBROUTINE ASSIGN(XB,YB,X,Y,N,M)
C
      REAL*8 XB,YB,X,Y
      DIMENSION X(100,100),Y(100,100),XB(400),YB(400)
C
      DO 298 I=1,N
      DO 298 J=1,M
      X(I,J)=0.D0
      Y(I,J)=0.D0
298   CONTINUE
C
      M2=M-1
C
      DO 300 I=1,N
      X(I,1)=XB(I)
      Y(I,1)=YB(I)
300   CONTINUE
C
      DO 310 J=2,M
      J1=N+J-1
  19  X(N,J)=XB(J1)
      Y(N,J)=YB(J1)
310   CONTINUE
C
```

```
      DO 320 I=2,N
      I1=N-I+1
      I2=N+M+I-2
      X(I1,M)=XB(I2)
      Y(I1,M)=YB(I2)
320   CONTINUE
C
      DO 330 J=2,M2
      J1=M-J+1
      J2=2*N+M+J-3
      X(1,J1)=XB(J2)
      Y(1,J1)=YB(J2)
330   CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------
C
C         SUBROUTINE PNUM
C
C   Purpose:This routine plot the node and element number by using
C             the connectivities.
C           NELD: control number for plotting the element number;
C
      SUBROUTINE PNUM(IJK,XE,YE,NEN,NEL,NNODE,NELD)
C
      DIMENSION IJK(3,1000),XE(10000),YE(10000),XX(8),YY(8)
      DO 399 I=1,NEL
      DO 299 J=1,NEN
C
         IA=IJK(J,I)
       XX(J)=XE(IA)
      YY(J)=YE(IA)
      X=XE(IA)-0.2+2.0
      Y=YE(IA)-0.1+2.0
C
      CALL PNUMBR(X,Y,0.08,IA,0.0,'I3*',0)
C
  299 CONTINUE
      IF(NELD.NE.1) GO TO 399
         XM=0.
         YM=0.0
C
      DO 345 J=1,NEN
        XM=XM+XX(J)
        YM=YM+YY(J)
  345 CONTINUE
        XM=XM/NEN+2.0
        YM=YM/NEN+2.0
      XM1=XM-0.11
      CALL PNUMBR(XM1,YM,0.08,I,0.0,'I3*',0)
      CALL PCIRCL(XM,YM,0.0,360.0,0.15,0.15,0.09,0)
  399 CONTINUE
C
      CALL PNUMBR(6.,0.8,0.15,NEL,0.0,'I7*',0)
      CALL PSYM(3.0,0.5,0.15,'THE NUMBER OF NODES = ',0.0,22,0)
      CALL PNUMBR(6.0,0.5,0.15,NNODE,0.0,'I7*',0)
      CALL PLTEND
```

```
      RETURN
      END
C
C        ------------------------------------------------
C
C
C             SUBROUTINE GAUSS
C
C     Puppose:     This subroutine caculates the area,
C     area moment of inertias of a cross-section using
C     2nd order gaussian quadrature for triangular elements.
C     Variables:
C     NODE:      Number of nodes
C     NEL:       Number of elements
C      XE,YE :    Nodal coordinates
C     IJK :      Element connectivity
C     AREA :     Area of an element
C     S1,S2. :  Principal area moment of inertias
C     ..................................................
C     SUBROUTINES:
C     FUNC :   Result of the integrations over each element
C
C     ..................................................
C
C
      SUBROUTINE GAUSS(XE,YE,IJK,NEL,NNODE,SXX,SYY,AREA)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XE(10000),YE(10000),IJK(3,1000),AX(3),AY(3)
C
C
      SXX=0.D0
      SYY=0.D0
      SXY=0.D0
      AREA=0.
C
      DO 100 J=1,NEL
           DO 80 I=1,3
           IP=IJK(I,J)
           AX(I)=XE(IP)
           AY(I)=YE(IP)
 80         CONTINUE
C
      CALL FUNCTN(AX,AY,SXX,SYY,SXY,AREA)
C
 100   CONTINUE
          SXX=2.*SXX
         SYY=2.*SYY
         SXY=2.*SXY
         AREA=2.*AREA
         WRITE(8,101) SXX,SXY,SYY,AREA
 101     FORMAT('SXX=    ',G20.10,/,
     &          'SXY=    ',G20.10,/,
     &          'SYY=    ',G20.10,/,
     &          'AREA=   ',G20.10,/)
C
C
         CALL PRNCPL(SXX,SYY,SXY,S1,S2,GAMMA)
         WRITE(8,102) S1,S2,GAMMA
```

```
 102     FORMAT('S1=',G20.10,/,
     &          'S2=',G20.10,/,
     &          'GAMMA=',G20.10)
      RETURN
      END
C
C        ------------------------------------------------
C
C
C              SUBROUTINE FUNCTN
C
C       Purpose: This subroutine calculates the area and moment
C                of inertias over each element.
C
C
      SUBROUTINE FUNCTN(AX,AY,SXX,SYY,SXY,AREA)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION AX(3),AY(3),BX(3),BY(3)
C
      BX(1)=(AX(1)+AX(2))/2.
      BY(1)=(AY(1)+AY(2))/2.
C
      BX(2)=(AX(2)+AX(3))/2.
      BY(2)=(AY(2)+AY(3))/2.
C
      BX(3)=(AX(3)+AX(1))/2.
      BY(3)=(AY(3)+AY(1))/2.
C
      EYY=(BX(1)*BX(1)+BX(2)*BX(2)+BX(3)*BX(3))/3.
      EXY=(BX(1)*BY(1)+BX(2)*BY(2)+BX(3)*BY(3))/3.
      EXX=(BY(1)*BY(1)+BY(2)*BY(2)+BY(3)*BY(3))/3.
C
      AREA1=((AX(2)*AY(3)-AY(2)*AX(3))+(AY(1)*AX(3)-AX(1)*AY(3))
     &      +(AX(1)*AY(2)-AY(1)*AX(2)))/2.
C
      SXX=SXX+DABS(AREA1*EXX)
      SYY=SYY+DABS(AREA1*EYY)
      SXY=SXY+DABS(AREA1*EXY)
      AREA=AREA+DABS(AREA1)
C
      RETURN
      END
C     ----------------------------------------
C
C          SUBROUTINE PRNCPL
C
C
C    Purpose:  This subroutine calculates the area moment
C              of inertias at the principal directions of the
C              drill cross-section
C
C
      SUBROUTINE PRNCPL(SXX,SYY,SXY,S1,S2,GAMMA)
C
      IMPLICIT REAL*8(A-H,O-Z)
C

      GAMMA=DATAN(2.*SXY/DABS(SXX-SYY))
      R=SXY/DSIN(GAMMA)
```

```
C
      S2=R+(SXX+SYY)/2.
      S1=S2-2.*R
C
      RETURN
      END
```