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
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. degreee 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 declarations in the main program, as well as in subroutines, must be changed to accommodate 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 cross section (see Thesis, Chapter 5). Then calculates the boundary geometry, and the boundary mesh. This boundary mesh is then used to discretize 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.

```plaintext
# r otnneigenload 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 different parameters
y
Current boundary condition is: clamped-pinned
Would you like to change the boundary conditions? (Y/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 ===> .34108550000000E-04, SQUARE METERS
PRINCIPAL MOMENT OF INERTIAS ARE
  ===> .58619300000000E-10, METERS!4
  ===> .29520320000000E-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 DIFFERENT 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  READ2 COMPLETE
16  DISCRE COMPLETE
17  ELEMEN COMPLETE
18  ELEMEN COMPLETE
19  ASEMBO COMPLETE
20  ********* EL= 0.40000000E-01
21  *********UMASS= 0.26263583
22  IN ELMEN  DUMK4= -6587.067603
23  ELEMEN COMPLETE
24  ASEMBO COMPLETE
25
26
27
28
29
30
31
32  ************* NL= 104************* MDIN.
***** U1= 0 ***** TETHA1= 5
***** V1= 0 ***** FHI1= 5
***** U2= 0 ***** TETHA2= 0
***** V2= 0 ***** FHI2= 0
ROW COLUMN 1 IS REMOVED
ROW COLUMN 2 IS REMOVED
ROW COLUMN 19 IS REMOVED
ROW COLUMN 19 IS REMOVED
ROW COLUMN 19 IS REMOVED
ROW COLUMN 19 IS REMOVED
UDU DECOMPOSITION IS COMPLETE,
TMINV IS INVERTED
STATE1 IS COMPLETE
RSG COMPLETE

**2*MDIM**

**EIGENVALUES**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(888.951) i</td>
</tr>
<tr>
<td>(888.951) i</td>
</tr>
<tr>
<td>(991.569) i</td>
</tr>
<tr>
<td>(991.569) i</td>
</tr>
<tr>
<td>(2490.69) i</td>
</tr>
<tr>
<td>(2490.69) i</td>
</tr>
<tr>
<td>(3117.81) i</td>
</tr>
<tr>
<td>(3117.81) i</td>
</tr>
<tr>
<td>(5939.31) i</td>
</tr>
<tr>
<td>(5939.31) i</td>
</tr>
<tr>
<td>(7394.85) i</td>
</tr>
<tr>
<td>(7394.85) i</td>
</tr>
<tr>
<td>(10850.9) i</td>
</tr>
<tr>
<td>(10850.9) i</td>
</tr>
<tr>
<td>(12127.6) i</td>
</tr>
<tr>
<td>(12127.6) i</td>
</tr>
<tr>
<td>(17937.2) i</td>
</tr>
<tr>
<td>(17937.2) i</td>
</tr>
<tr>
<td>(18978.6) i</td>
</tr>
<tr>
<td>(18978.6) i</td>
</tr>
<tr>
<td>(25756.2) i</td>
</tr>
<tr>
<td>(25756.2) i</td>
</tr>
<tr>
<td>(28920.0) i</td>
</tr>
<tr>
<td>(28920.0) i</td>
</tr>
<tr>
<td>(36029.0) i</td>
</tr>
<tr>
<td>(36029.0) i</td>
</tr>
<tr>
<td>(40408.4) i</td>
</tr>
<tr>
<td>(40408.4) i</td>
</tr>
<tr>
<td>(48977.5) i</td>
</tr>
<tr>
<td>(48977.5) i</td>
</tr>
<tr>
<td>(52725.7) i</td>
</tr>
<tr>
<td>(52725.7) i</td>
</tr>
<tr>
<td>(65302.7) i</td>
</tr>
<tr>
<td>(65302.7) i</td>
</tr>
<tr>
<td>(81110.8) i</td>
</tr>
<tr>
<td>(81110.8) i</td>
</tr>
</tbody>
</table>

***** IER***** = 0
IN ELMEN DUMK4= -3807.553527
ELEMEN COMPLETE
ASEMBO COMPLETE

************** N1== 104************* MDIM== 18

***** U1= 0 ***** TETHA1= 5
***** V1= 0 ***** PHI1= 5
***** U2= 0 ***** TETHA2= 0
***** V2= 0 ***** PHI2= 0

ROW COLOUMN 1 IS REMOVED
ROW COLOUMN 2 IS REMOVED
ROW COLOUMN 13 IS REMOVED
ROW COLOUMN 13 IS REMOVED
ROW COLOUMN 13 IS REMOVED
ROW COLOUMN 13 IS 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,
ACKNOWLEDGEMENTS

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

REFERENCES


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* THIS PROGRAM SOLVES THE DRILL BIT EQUATIONS *
C* THE SOLUTION CAN BE AN EIGENVALUE PROBLEM *
C* OR TIME RESPONSE *
C*****************************************************************************

ASSOCIATED SUBROUTINES-------

C*** READ2(AREA, S1, S2, NL, OMEGA, ELAS, BETA, RHO, FZ, TL, U1, TETHA1, V1, PHI1, U2, TETHA2, V2, PHI2) = Reads data

C*** DISCRE(TL, NL, AREA, RHO, EL, UMASS) = Discretizes drill bit

C*** ELEMEN(S1, S2, EL, BETA, UMASS, FZ, OMEGA, ELAS, SME, SCE, SKE) = Calculates element matrices

C*** ASEMBO(SME, SCE, SKE, NL, NI, MDIM, TM, TC, TK) = Assembles element matrices to a global one

C*** SUBROUTINE GROUND(TM, TC, NI, MDIM, U1, TETHA1, V1, PHI1, U2, TETHA2, V2, PHI2) = Imposes geometric boundary conditions to global TM, TC, TK

C*** EIGENS(NL, TKTOT, TMTOT, W, Z, SV1, SV2) = Solves the generalized symmetric eigenvalue problem of TKTOT & TMTOT

C*** TIMER() = calculates the time response of the system using TKTOT and input forcing function.

C*** READ3(AREA, S1, S2, NL, OMEGA, ELAS, BETA, RHO, FZ, TL, U1, TETHA1, V1, PHI1, U2, TETHA2, V2, PHI2, FLAG2, FLAG3, FLAG4) = prompts the user for changing the parameters to rerun the program.

ARGUMENTS---------

AREA= Cross-sectional area
S1,S2= Area moment of inertia in principal directions
OMEGA= Drill rotational speed
ELAS= Young's modulus of elasticity
BETA= Drill helix angle
RHO= Material density
FZ= Axial force
TL= Total free length of the drill bit
NL= Number of beam elements
EL= Length of the beam element
UMASS= Mass/unit length
XL= Coordinates of nodes
SME,SCE,SKE= Element mass, damping, stiffness matrices
TM,TC,TK= Global mass, damping, stiffness matrices
MDIM= Dimension of global matrices

--------------
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

Parameters below are the dimension of the largest system that can be solved. This translates into a 20 element system. i.e. \( N1=MDIM=4\times(20+1)=84 \) to increase the allowable system change \( N1 \)
PARAMETER(N1=84)

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(I),TWORK(I)
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)

::: note DUMMY is declared in blank COMMON :::::::::::::
::::: hence dimensions are adjustable:::::::::::::::
COMMON DUMMY(6*N1+1)

EQUIVALENCE (TK(1,1), TKTOT(1,1)),(DUMMY(1),SV1(1))

::: following are for subroutine timer :::::::::::::
EQUIVALENCE (DUMMY(1),TY(I)),(DUMMY(2*N1+1),TWORK(I))

::: following are for subroutine eigens ::::::::::::
EQUIVALENCE (EW(1),DUMMY(1)),(ESV1(1),DUMMY(2*N1+1))
&
,(DUMMY(4*N1+1),ESV2(1))

--------------- initialize the flags ---------------

FLAG1 = 'N'
FLAG2 = 'N'
FLAG3 = 'N'
FLAG4 = 'N'
FLAG5 = 'N'

-------------------------------------

CALL READ2(ARE5,S1,S2,SL,CM5,E5,B5,R5,FZ,TL,
&
U1,TETHA1,V1,FHI1,U2,TETHA2,V2,FHI2)

160 WRITE(6,150)
150 FORMAT(',//READ2 COMPLETE',//)

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
::: discretize the drill bit :::
CALL DISCRE(TL, NL, AREA, RHO, EL, UMASE)
C
PRINT*, 'TL=', TL
PRINT*, 'NL=', NL
WRITE(6,151)
FORMAT(//,'DISCRE COMPLETE',//)
WRITE(6,101) EL, UMASS
101   FORMAT(//,'****** EL=',G20.8
      &
      '/','******UMASS=','G20.8)

::: operating or drill parameters are changed :::
::: entry point for GOTO 5000 :::
5000 CONTINUE

::: calculate the element matrices :::
CALL ELEMEN(S1,S2,EL,BETA,UMASS,FZ,CMEGA,
      &
      ELAS,SME,SCE,SKE)
C
WRITE(6,152)
152   FORMAT(//,'ELEMEN COMPLETE',//)

::: boundary conditions are changed FLAG2=YES :::
::: reassemble and reimpose the geometric boundary conditions :::
::: entry point for: GOTO 6000 :::
6000 CONTINUE

::: assemble mass stiffness and coriolis matrices :::
::: 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) NL, 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

A-3
TETHA1=5
FHI1=5
TETHA2=0
FHI2=0
ENDIF

C
:impose the geometric boundary conditions:


CALL GROUND(TM, TC, TK, N1, MDIM, U1, TETHA1, V1, FHI1,
&
U2, TETHA2, V2, FHI2)

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)
CONTINUE

C

C
:invert TMINV

C
:compute UDU decomposition of TMINV

CALL DSICO(TMINV, N1, MDIM, IV, RC, SV1)
WRITE(6,73)
FORMAT('UDU DECOMPOSITION IS COMPLETE,/',)
IF(RC .EQ. 1) PRINT*, 'WARNING!!! TM IS ILL CONDITIONED'

C

C
:inverted UDU decomposition of TMINV

CALL DSIDI(TMINV, N1, MDIM, IV, DET, INERT, SV1, I)
WRITE(6,74)
FORMAT('TMINV IS INVERTED!')

C

C
:make TMINV a full symmetric 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)
CONTINUE

C

C
:eigensolution or time response??

IF((ALPHA .EQ. 'T') .OR. (ALPHA .EQ. 'T')) THEN

C

C
:calculate time response of the system

CALL TIMER(N1, MDIM, TC, TK, TMINV, UMASS)

ELSE

C

C
:calculate the eigenvalues of the system

CALL EIGENS(N1, MDIM, TM, TC, TK, TMINV, EW, ESV1, ESV2)

ENDIF
--- PROMPT TO RERUN THE PROGRAM ---

FLAG1 = 'N'
FLAG2 = 'N'
FLAG3 = 'N'
FLAG4 = 'N'
FLAG5 = 'N'

WRITE(*,*) 'WOULD YOU LIKE TO RERUN WITH DIFFERENT PARAMETERS?'
READ(*,'(A1)',END=1800) FLAG1
IF(((FLAG1.EQ.'N') .OR. (FLAG1.EQ. 'n')) ) GOTO 1800

CALL READ3(area,s1,s2,nl,omega,elas,beta,rho,fz,tl & ,u1,tetha1,v1,phi1,u2,tetha2,v2,phi2,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

--- SUBROUTINE READ3 ---

SUBROUTINE READ3(area,s1,s2,nl,omega,elas,beta,rho,fz,tl & ,u1,tetha1,v1,phi1,u2,tetha2,v2,phi2)
CHARACTER*1 flag1,flag2,flag3,flag4

--- CHANGE THE BOUNDARY CONDITIONS ---

PRINT*, 'CURRENT BOUNDARY 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. (TETHA2 .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

PRINT*, 'WOULD YOU LIKE TO CHANGE THE BOUNDARY CONDITIONS? (Y/N):'
READ(*,'(A1)',END=100) FLAG2
5 PRINT*, 'ENTER ::'
PRINT*, '1 FOR CAMPED-CLAMPED BOUNDARY CONDITION'
PRINT*, '2 FOR PIN-PIN BOUNDARY CONDITION'
PRINT*, '3 FOR CAMPED-PINNED BOUNDARY CONDITION'
PRINT*, '4 FOR CANTILEVER BOUNDARY CONDITION'

READ(*,*), I
IF ((I .LT. 1) .OR. (I .GT. 4)) THEN

A-5
GOTO 5
ELSE IF(I .EQ. 1) THEN
  clamped-clamped bc.
  U1 = 0
  U2 = 0
  V1 = 0
  V2 = 0
  TETHA1 = 0
  TETHA2 = 0
  FHI1 = 0
  FHI2 = 0
END IF
ELSE IF(I .EQ. 2) THEN
  pin-pin bc
  U1 = 0
  U2 = 0
  V1 = 0
  V2 = 0
  TETHA1 = 5
  TETHA2 = 5
  FHI1 = 5
  FHI2 = 5
END IF
ELSE IF(I .EQ. 3) THEN
  clamped-pinned bc
  U1 = 0
  U2 = 0
  V1 = 0
  V2 = 0
  TETHA1 = 0
  TETHA2 = 5
  FHI1 = 0
  FHI2 = 5
ELSE
  cantilever bc
  U1 = 0
  U2 = 5
  V1 = 0
  V2 = 5
  TETHA1 = 0
  TETHA2 = 5
  FHI1 = 0
  FHI2 = 5
END IF
END IF

---------- 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

IF((FLAG3 .EQ. 'Y') .OR. (FLAG3 .EQ. 'y')) THEN
  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
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
CHANGE THE DRILL SPECIFICATIONS

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

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
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

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

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

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*,
A-7
'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,THETA1,V1,FHI1,U2,THETA2,V2,FHI2)
C
IMPLICIT REAL*8(A-H,O-Z)
INTEGER*2 U1,THETA1,V1,FHI1,U2,THETA2,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
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,THETA1,V1,FHI1,U2,THETA2,V2,FHI2
10  FORMAT(8I5)
C
WRITE (6,15) U1,THETA1,V1,FHI1,U2,THETA2,V2,FHI2
A-8
FORMAT(**** U1=','I3,5X,**** TETHA1=','I3,/, &
  '**** V1=','I3,5X,**** FHI1=','I3,/, &
  '**** U2=','I3,5X,**** TETHA2=','I3,/, &
  '**** V2=','I3,5X,**** FHI2=','I3)
C
RETURN
END

-------------------------

SUBROUTINE DISCRE(TL, NL, AREA, RHO, EL, UMASS)

ARGUMENTS--------

AREA= Cross-sectional area
RHO= Material density
NL= Number of beam elements
EL= Length of the beam element
UMASS= Mass/unit length
XL= Coordinates of nodes
TL= Total free length of the drill bit

SUBROUTINE DISCRE(TL, NL, AREA, RHO, EL, UMASS)

IMPLICIT REAL*8(A-H,O-Z)

EL=TL/NL

UMASS=AREA*RHO

RETURN
END

-------------------------

SUBROUTINE ELEMEN(S1, S2, EL, BETA, UMASS, FZ, OMEGA, 
  ELAS, SME, SCE, SKE)

ARGUMENTS--------

S1, S2= Area moment of inertias in principal directions
OMEGA= Drill rotational speed
ELAS= Young's modulus of elasticity
BETA= Drill helix angle
FZ= Axial force
EL= Length of the beam element
UMASS= Mass/unit length
SME, SCE, SKE= Element mass, damping, stiffness matrices

-------------------------

SUBROUTINE ELEMEN(S1, S2, EL, BETA, UMASS, FZ, OMEGA, 
  ELAS, SME, SCE, SKE)

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION SME(8,8), SCE(8,8), SKE(8,8),
SME (I,J)
SME (1,1)=(-13.*EL)/35.
SME (1,2)=(-11.*EL**2)/210.
SME (1,3)=0.00
SME (1,4)=0.00
SME (1,5)=(-9.*EL)/70.
SME (1,6)=(-13.*EL**2)/420.
SME (1,7)=0.00
SME (1,8)=0.00
SME (2,1)=(-11.*EL**2)/210.
SME (2,2)=EL**3/105.
SME (2,3)=0.00
SME (2,4)=0.00
SME (2,5)=(-13.*EL**2)/420.
SME (2,6)=(-EL**3)/140.
SME (2,7)=0.00
SME (2,8)=0.00
SME (3,1)=0.00
SME (3,2)=0.00
SME (3,3)=(-13.*EL)/35.
SME (3,4)=(-11.*EL**2)/210.
SME (3,5)=0.00
SME (3,6)=0.00
SME (3,7)=(-9.*EL)/70.
SME (3,8)=(-13.*EL**2)/420.
SME (4,1)=0.00
SME (4,2)=0.00
SME (4,3)=(-11.*EL**2)/210.
SME (4,4)=EL**3/105.
SME (4,5)=0.00
SME (4,6)=0.00
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)=0.00
SME (5,4)=0.00
SME (5,5)=(-13.*EL)/35.
SME (5,6)=(-11.*EL**2)/210.
SME (5,7)=0.00
SME (5,8)=0.00
SME (6,1)=(-13.*EL**2)/420.
SME (6,2)=(-EL**3)/140.
SME (6,3)=0.00
SME (6,4)=0.00
SME (6,5)=(-11.*EL**2)/210.
SME (6,6)=EL**3/105.
SME (6,7)=0.00
SME (6,8)=0.00
SME (7,1)=0.00
SME (7,2)=0.00
SME (7,3)=(-9.*EL)/70.
SME (7,4)=(-13.*EL**2)/420.
SME (7,5)=0.00
SME (7,6)=0.00
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(4,2)=(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) = 0.0 D
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) = 0.0 D
SCE(7, 4) = 0.0 D
SCE(7, 5) = (13.*EL*DUMCE)/35.
SCE(7, 6) = (-11.*EL**2*DUMCE)/210.
SCE(7, 7) = 0.0 D
SCE(7, 8) = 0.0 D
SCE(8, 1) = (-13.*EL**2*DUMCE)/420.
SCE(8, 2) = (-EL**3*DUMCE)/140.
SCE(8, 3) = 0.0 D
SCE(8, 4) = 0.0 D
SCE(8, 5) = (-11.*EL**2*DUMCE)/210.
SCE(8, 6) = (EL**3*DUMCE)/105.
SCE(8, 7) = 0.0 D
SCE(8, 8) = 0.0 D

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) = 0.0 D
  SKE1(1, 4) = 0.0 D
  SKE1(1, 5) = DUMK1*(9.*EL)/70.
  SKE1(1, 6) = -DUMK1*(13.*EL**2)/420.
  SKE1(1, 7) = 0.0 D
  SKE1(1, 8) = 0.0 D
  SKE1(2, 1) = DUMK1*(11.*EL**2)/210.
  SKE1(2, 2) = DUMK1*EL**3/105.
  SKE1(2, 3) = 0.0 D
  SKE1(2, 4) = 0.0 D
  SKE1(2, 5) = DUMK1*(13.*EL**2)/420.
  SKE1(2, 6) = DUMK1*(EL**3)/140.
  SKE1(2, 7) = 0.0 D
  SKE1(2, 8) = 0.0 D
  SKE1(3, 1) = 0.0 D
  SKE1(3, 2) = 0.0 D
  SKE1(3, 3) = DUMK1*(13.*EL)/35.
  SKE1(3, 4) = DUMK1*(11.*EL**2)/210.
  SKE1(3, 5) = 0.0 D
  SKE1(3, 6) = 0.0 D
  SKE1(3, 7) = DUMK1*(9.*EL)/70.
  SKE1(3, 8) = -DUMK1*(13.*EL**2)/420.
  SKE1(4, 1) = 0.0 D
  SKE1(4, 2) = 0.0 D
  SKE1(4, 3) = DUMK1*(11.*EL**2)/210.
  SKE1(4, 4) = DUMK1*EL**3/105.
  SKE1(4, 5) = 0.0 D
  SKE1(4, 6) = 0.0 D
  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) = 0.0 D
  SKE1(5, 4) = 0.0 D

A-12
SKE1(5, 5) = DUMK1*(13.*EL)/35.
SKE1(5, 6) = DUMK1*(11.*EL**2)/210.
SKE1(5, 7) = 0.00
SKE1(5, 8) = 0.00
SKE1(6, 1) = DUMK1*(13.*EL**2)/420.
SKE1(6, 2) = DUMK1*(EL**3)/140.
SKE1(6, 3) = 0.00
SKE1(6, 4) = 0.00
SKE1(6, 5) = DUMK1*(11.*EL**2)/210.
SKE1(6, 6) = DUMK1*EL**3/105.
SKE1(6, 7) = 0.00
SKE1(6, 8) = 0.00
SKE1(7, 1) = 0.00
SKE1(7, 2) = 0.00
SKE1(7, 3) = DUMK1*(9.*EL)/70.
SKE1(7, 4) = DUMK1*(13.*EL**2)/420.
SKE1(7, 5) = 0.00
SKE1(7, 6) = 0.00
SKE1(7, 7) = DUMK1*(13.*EL)/35.
SKE1(7, 8) = DUMK1*(11.*EL**2)/210.
SKE1(8, 1) = 0.00
SKE1(8, 2) = 0.00
SKE1(8, 3) = DUMK1*(13.*EL**2)/420.
SKE1(8, 4) = DUMK1*(EL**3)/140.
SKE1(8, 5) = 0.00
SKE1(8, 6) = 0.00
SKE1(8, 7) = DUMK1*(11.*EL**2)/210.
SKE1(8, 8) = DUMK1*EL**3/105.

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+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+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+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+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
. 10. *S1)) / (5. *EL) * DUMK2
SKE2(6, 8) = BETA *(S2-S1) * DUMK2
SKE2(7, 1) = BETA **3 *(S2+1) * DUMK2
SKE2(7, 2) = (BETA * (EL**2 * BETA**2 * S2+EL**2 * BETA**2 * S2+10. * S2+)
. 10. *S1)) / (5. *EL) * DUMK2
SKE2(7, 3) = (3. * (S2+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
. 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

SKE4(1, 7)
DUMMY = - FZ / UMASS
WRITE (6, 80) DUMMY
FORMAT ('IN ELMN  DUMMY=', G20.10)
SKE4(1, 1) = (DUMMY**(13. * BETA**2 * EL**2+42.1)) / (35. * EL)
SKE4(1, 3) = 0.0
SKE4(1, 4) = (BETA * EL * DUMMY4 / 5.
SKE4(1, 5) = (3. * DUMMY4 * (3. * BETA**2 * EL**2-28.)) / (70. * EL)
SKE4(1, 6) = (DUMMY4**(-13. * BETA**2 * EL**2+42.1)) / 420.
SKE4(1, 7) = BETA * DUMMY4
SKE4(1, 8) = (-BETA * EL * DUMMY4) / 5.
SKE4(2, 2) = (EL * DUMMY4 * (BETA**2 * EL**2+14.1)) / 105.
SKE4(2, 3) = (-BETA * EL * DUMMY4) / 5.
SKE4(2, 4) = 0.0
SKE4(2, 7) = (BETA * EL * DUMMY4) / 5.
SKE4(2, 8) = (-BETA * EL**2 * DUMMY4) / 30.
SKE4(3, 1) = 0.0
SKE4(3, 2) = (-BETA * EL * DUMMY4 / 5.
SKE4(3, 3) = (DUMMY4**(11. * BETA**2 * EL**2+42.1)) / (35. * EL)
SKE4(3, 5) = -BETA * DUMMY4

A-15
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, 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, 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, 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(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, 8) = (EL*DUMK4* (BETA**2*EL**2+14. ))/105.

DO 1300 I=1, 8
DO 1300 J=1, 8
SKE(I, J) = SKE1(I, J) + SKE2(I, J) + SKE4(I, J)
CONTINUE

RETURN
END

------------------------------------------------------------------------

A-16
SUBROUTINE ASEMBO(SME, SCE, SKE, NL, N1, N2, MDIM, TM, TC)

ARGUMENTS--------

SME, SCE, SKE = Element mass, damping, stiffness matrices
TM, TC, TK = Global mass, damping, stiffness matrices
note TK is passed in the common block A
ML, MDIM = Dimension of the global matrices
NL = Number of elements

------------------------

SUBROUTINE ASEMBO(SME, SCE, SKE, NL, N1, MDIM, TM, TC, TK)

REAL*8 SME(8,8), SCE(8,8), SKE(8,8), TM(2*N1+1,1),
&
TC(N1,1), TK(2*N1+1,1)

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

DO 1530 I=1,NL
C
II=4*(I-1)

DO 1530 IIN=1,8
DO 1530 JIN=1,8
TM(I1+IIN,II1+JIN)=TM(I1+IIN,II1+JIN)+SME(IIN,JIN)
TK(I1+IIN,II1+JIN)=TK(I1+IIN,II1+JIN)+SKE(IIN,JIN)
TC(I1+IIN,II1+JIN)=TC(I1+IIN,II1+JIN)+SCE(IIN,JIN)
1530 CONTINUE

RETURN
END

---------------------------------------------

/* filename eigens7 11/19/87 */

This subroutine calculates the eigenvalues of the system of equations using NAAS:EISPACK routine RSG. The routine solves the generalized symmetric eigenvalue problem.

SUBROUTINE EIGENS(NL, MDIM, TM, TC, TK, TMINV, W, SV1, SV2)

IMPLICIT REAL*8 (A-H, O-Z)
NOTE THE FOLLOWING PARAMETER DECLARATION IS SET FOR N1=120
OR 30 ELEMENTS
PARAMETER(N3=241)

DIMENSION TM(2*N1+1,1), TC(N1,1), TK(2*N1+1,1), TMINV(N1,1)

::: local vector declarations ::
DIMENSION W(1), Z(N3,N3), SV1(1), SV2(1)

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
CONTINUE

::: transform the equations into moirorwitch form ::
::: assemble state space form ::
CALL STATE1(N1,MDIM,TM,TC,TK,TMINV)
WRITE(6,5)
FORMAT( 'STATE1 IS COMPLETE',/

MATZ=0
CALL RSG(2*N1+1,M1,TK,TM,W,MATZ,Z,SV1,SV2,IER)
M
WRITE(6,7)
FORMAT( ' RSG COMPLETE'

WRITE(6,1705)
FORMAT( '***2*MDIM***** EIGENVALUES ************'
DO 1710 I=1,M1
PI=3.141592654D0
W(I) = DABS(W(I)) / 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

SUBROUTINE STATE1(N1,MDIM,TM,TC,TK,TMINV)
input matrices:
C
|TM  0|
TM = |   |
|0   0|

|TK  0|
TK = |
|0   0|

matrix descriptions:
C
|TM11  TM12|
TM = |
|TK11  TK12|

C
|TM21   TM22| |TK21   TK22|

state space configuration:

\[
\begin{pmatrix}
|TM & 0| \\
| 0 & TK|
\end{pmatrix}
\begin{pmatrix}
|-TC*TMINV*TC+TK| \\
|TK*TMINV*TC TMINV*TK|
\end{pmatrix}
\]

\(\text{tr}(TK21)\)

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION TM(2*N1+1,1), TC(N1,1), TK(2*N1+1,1), TMINV(N1,1)

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)
CONTINUE

10

:::initialize TK:::----------------

DO 20 I=1, 2*MDIM
DO 20 J=1, 2*MDIM
TK(I, J) = 0.0
CONTINUE

20

:::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)
CONTINUE

30

:::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)
CONTINUE

40

:::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)
CONTINUE

50

:::TK21 = TM22 * TK12 = TK * TMINV * TC:::::::::

DO 55 I=1, MDIM
DO 55 J=1, MDIM
DO 55 K=1, MDIM
TK(I+MDIM, J) = TK(I+MDIM, J) + TM(I+MDIM, J+MDIM) * TK(K, J+MDIM)
CONTINUE

55

:::discarding the coriolis matrix!!:::::::

:::TC = TMINV * TK:::::::::::::::

DO 57 I=1, MDIM
DO 57 J=1, MDIM
TC(I, J) = 0.0
CONTINUE

57

DO 60 I=1, MDIM

A-19
DO 60 J=1,MDIM
DO 60 K=1,MDIM
TC(I,J) = TC(I,J) + TMINV(I,K) * TM(K+MDIM,J+MDIM)
CONTINUE

:::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)
CONTINUE

:::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)
CONTINUE

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
CONTINUE

RETURN
END

C/* filename: boundry7 11/10/87 */
C CALCULATE BOUNDARY CONDITIONS
----------------------------------------
This file contains the following subroutines:
----------------------------------------
C /* SUBROUTINE GBOUND(TM,TC,N1,N2,MDIM,U1,TETHA1,V1,PHI1,
& U2,TETHA2,V2,PHI2)
C*/
C/* SUBROUTINE KNCOFF(TM,TC,TK,TKMAX,TCMAX,TKMAX,IIN,N1,M1,MDIM)
C*/
----------------------------------------
C SUBROUTINE GBOUND(TM,TC,TK,N1,MDIM,U1,TETHA1,V1,PHI1,
& U2,TETHA2,V2,PHI2)
This subroutine assigns the geometric boundary
conditions to the global mass, coriolis & stiffness
matrices.
It reads the values U1,TETHA1,...etc. from the file, then
if U1=0 then U1 is constrained thus subroutine makes
the necessary changes in the global mass stiffness and
coriolis matrices
C SUBROUTINE GBOUND(TM,TC,TK,N1,MDIM,U1,TETHA1,V1,PHI1,
& U2,TETHA2,V2,PHI2)
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,PHI1,U2,TETHA2,V2,PHI2

A-20
WRITE(6,15) U1,TETHA1,V1,PHI1,U2,TETHA2,V2,PHI2
15 FORMAT ('***** U1=',I3,5X,'***** TETHA1=',I3,/, '***** V1=',I3,5X,'***** PHI1=',I3,/, '***** U2=',I3,5X,'***** TETHA2=',I3,/, '***** V2=',I3,5X,'***** PHI2=',I3)

IIS=0
IF (U1.NE.0) GOTO 30
IIN=1
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
IIS=IIS+1
30 IF (TETHA1.NE.0) GOTO 40
IIN=2-IIS
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
IIS=IIS+1
40 IF (V1.NE.0) GOTO 50
IIN=3-IIS
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
IIS=IIS+1
50 IF (PHI1.NE.0) GOTO 60
IIN=4-IIS
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
IIS=IIS+1
60 IF (U2.NE.0) GOTO 70
IIN=MDIM-3
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
70 IF (TETHA2.NE.0) GOTO 80
IIN=MDIM-2
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
80 IF (V2.NE.0) GOTO 90
IIN=MDIM-1
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
90 IF (PHI2.NE.0) GOTO 100
IIN=MDIM
CALL KNCOFF(TM,TC,TK,IIN,NL,MDIM)
100 CONTINUE
C
RETURN
END

---------------------------------------------
---------------------------------------------
---------------------------------------------
SUBROUTINE KNCOFF(TM,TC,TK,TMMAK,TCMAX,TKMAK,IIN,NL,ML,MDIM)
---------------------------------------------
---------------------------------------------
---------------------------------------------
this subroutine removes the IIN'th row and columns
and consequently the matrix dimensions is reduced by one
i.e. MDIM=MDIM-1
---------------------------------------------
---------------------------------------------
---------------------------------------------
| a1n |
| ... |
| a2n.... |
| ... | .... |
| TM=| an1 an2 ...ann|
ain and an1 are removed
SUBROUTINE KNCOFF(TM, TC, TK, IIN, NL, MDIM)

IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION TM(2*NL+1,1), TC(NL,1), TK(2*NL+1,1)

WRITE(6,199) IIN
199 FORMAT('ROW COLUMN', I5, 'IS REMOVED')

reduce the dimension of the global matrices by 1
MDIM = MDIM - 1
Remove the unwanted 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
Remove the unwanted column 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

RETURN
END

C/*filename: timer9 12/3/87 */

C                      Associated 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                      ______________________ 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(NL, MDIM, TC, TK, TMINV, UMASS)

IMPLICIT REAL*8 (A-H, O-Z)
PARAMETER(NZ=84)

DIMENSION TMINV(NL,1), TC(NL,1), TK(2*NL+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

::: read the corresponding data :::

C
IDIM=2*MDIM+1
CALL READ4(MDIM,Y,FX,FY,T,TFINAL,NSTEPS,TIMEC1)

C
C

::: formulate the state space representation :::

C
CALL STATE2(N1,MDIM,TMINV,TC,TK,TIMEC1)

C
C

::: setup the forcing function :::

C
TIMEC=TIMEC1
FX = FX/UMASS
FY = FY/UMASS

C
C

::: forcing function in state space representation :::

C
DO 5 I=1,MDIM
   F(I) = TMINV(I,1) * FX + TMINV(I,3) * FY
   CONTINUE

C
C

DO 6 I=1+MDIM,2*MDIM
   F(I) = 0.0D0
   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
   PRINT*, 'SOLUTION POINT: ',TOUT

C
TOUT=TOUT+TDELTA

A-23
WRITE (7,30) (Y(J), J=MDIM+1,2*MDIM, 4)
FORMAT (5G15.7)

CONTINUE

RETURN
END

SUBROUTINE STATE2 (N1, MDIM, TMINV, TC, TK, TIMEC)

TK = | -TMINV*TC -TMINV*TK  0 |
    |    I    0  0   |
    |  0 .... 0  -1/TIMEC|

TK = | TK11 TK12 |
    | TK21 TK22 |

IMPLICIT REAL*8 (A-H,O-Z)

DIMENSION TMINV(N1,1), TC(N1,1), TK(2*N1+1,1)

TK22 = TK = TK11, TK11 = 0.D0, TK12 = 0.D0, TK21 = 0.D0

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

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

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

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

TK21 = I
DO 50 I=1, MDIM

A-24
TK(I+MDIM,I) = 1.D0
CONTINUE

C
C : : : plac e 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
CONTINUE

C

TK(2*MDIM+1,2*MDIM+1) = -1/TIMEC
RETURN
END

C
C
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
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)
FORMAT(5G15.7)

C
RETURN
END
C
C
C --------------------------------------------------
C
C
C The forcing function in the following subroutine is obtained
C through a random number generator. Then it is filtered 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 system of equations is for the filter.
C
C SUBROUTINE DIFFUN(ML,T,Y,YDOT)

A-25
C

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION Y(1), YDOT(1)

C

PARAMETER (N1=84)
COMMON /A/T(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.DO
5 CONTINUE

C

DO 10 I=1,M1
DO 10 J=1,M1
YDOT(I) = YDOT(I) + T(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

-----------------------------------------------------------
SUBROUTINE PDEERV(M1,T,Y,PD,NO)
DIMENSION Y(168,1), PD(1)
RETURN
END
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION Y(1),YDOT(1)

PARAMETER(NL=84)
COMMON /A/TM(2*N1+1,1)
COMMON /C/TIMEC,F1(1)
COMMON /BB/ICOUNT

IF (MOD(ICOUNT,60) .EQ.0) PRINT*, 'Y(M1) =', Y(M1)
ICOUNT = ICOUNT+1

DO 5 I=1,M1
YDOT(I) = 0.D0
CONTINUE

5

DO 10 I=1,M1
DO 10 J=1,M1
YDOT(I) = YDOT(I) + TM(I,J) * Y(J)
CONTINUE

10

DO 20 I=1,M1-1
YDOT(I) = YDOT(I) + F(I)*Y(M1)
CONTINUE

20

YDOT(M1) = YDOT(M1) + F(M1)

RETURN
END

SUBROUTINE PEDERV(M1,T,Y,PD,NO)
DIMENSION Y(168,1),PD(1)
RETURN
END
SUBROUTINE DRIVE (N, T0, H0, Y0, TOUT, EPS, MF, INDEX)

THIS IS THE JANUARY 13, 1975 VERSION OF
GEAR, A PACKAGE FOR THE SOLUTION OF THE INITIAL VALUE
PROBLEM FOR SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS,
DY/DT = F(Y,T), Y = (Y(1),Y(2),...,Y(N)).
SUBROUTINE DRIVE IS A DRIVER ROUTINE FOR THE GEAR PACKAGE.

REFERENCES
1. A. C. HINDMARSH, GEAR: ORDINARY DIFFERENTIAL EQUATION
SYSTEM SOLVER, UCID-30001 REV. 3, LAWRENCE LIVERMORE
LABORATORY, P.O. BOX 808, LIVERMORE, CA 94550, DEC. 1974.

2. A. C. HINDMARSH, LINEAR MULTISTEP METHODS FOR ORDINARY
DIFFERENTIAL EQUATIONS: METHOD FORMULATIONS,
STABILITY, AND THE METHODS OF NORDSTIECK AND GEAR,
UCRL-51186 REV. 1, L.L.L., MARCH 1972.

3. A. C. HINDMARSH, CONSTRUCTION OF MATHEMATICAL SOFTWARE,
PART III: THE CONTROL OF ERROR IN THE GEAR PACKAGE
FOR ORDINARY DIFFERENTIAL EQUATIONS, UCID-30050 PART 3,
L.L.L., AUGUST 1972.

THE ORIGINAL VERSION OF THIS PROGRAM WAS WRITTEN AT LLL BY
A. C. HINDMARSH FOR CDC COMPUTERS. THE CDC VERSION WAS MODIFIED
FOR USE ON IBM COMPUTERS IN DOUBLE PRECISION AT ARGONNE
NATIONAL LABORATORY IN JANUARY 1975.

DRIVE IS TO BE CALLED ONCE FOR EACH OUTPUT VALUE OF T, AND
IN TURN MAKES REPEATED CALLS TO THE CORE INTEGRATOR, STIFF.

THE INPUT PARAMETERS ARE..
N = THE NUMBER OF FIRST-ORDER DIFFERENTIAL EQUATIONS.
N CAN BE REDUCED, BUT NEVER INCREASED, DURING A PROBLEM.
T0 = THE INITIAL VALUE OF T, THE INDEPENDENT VARIABLE
(USED ONLY ON FIRST CALL).
H0 = THE NEXT STEP SIZE IN T (USED FOR INPUT ONLY ON THE
FIRST CALL).
Y0 = A VECTOR OF LENGTH N CONTAINING THE INITIAL VALUES OF
Y (USED FOR INPUT ONLY ON FIRST CALL).
TOUT = THE VALUE OF T AT WHICH OUTPUT IS DESIRED NEXT.
INTEGRATION WILL NORMALLY GO SLIGHTLY BEYOND TOUT
AND THE PACKAGE WILL INTERPOLATE TO T = TOUT.
EPS = THE RELATIVE ERROR BOUND (USED ONLY ON THE
FIRST CALL, UNLESS INDEX = -1). SINGLE STEP ERROR
ESTIMATES DIVIDED BY YMAX(I) WILL BE KEPT LESS THAN
EPS IN ROOT-MEAN-SQUARE NORM (I.E. EUCLIDEAN NORM
DIVIDED BY DSQRT(N)). THE VECTOR YMAX OF
WEIGHTS IS COMPUTED IN DRIVE. INITIALLY YMAX(I) IS
DABS(Y(I)), WITH A DEFAULT VALUE OF 1 IF Y(I) = 0
INITIALLY. THEREAFTER, YMAX(I) IS THE LARGEST VALUE
OF DABS(Y(I)) SEEN SO FAR, OR THE INITIAL YMAX(I) IF
THAT IS LARGER. TO ALTER EITHER OF THESE, CHANGE THE
APPROPRIATE STATEMENTS IN THE DO-LOOPS ENDING AT
STATEMENTS 10 AND 70 BELOW.
MF = THE METHOD FLAG (USED ONLY ON FIRST CALL, UNLESS
INDEX = -1). ALLOWED VALUES ARE 10, 11, 12, 13,
20, 21, 22, 23. MF HAS TWO DECIMAL DIGITS, METH
AND MITER (MF = 10*METH + MITER).
METH IS THE BASIC METHOD INDICATOR.
METH = 1 MEANS THE ADAMS METHODS.
METH = 2 MEANS THE STIFF METHODS OF GEAR, OR THE
BACKWARD DIFFERENTIATION FORMULAS.
MITER IS THE ITERATION METHOD INDICATOR.
MITER = 0 MEANS FUNCTIONAL ITERATION (NO PARTIAL
DERIVATIVES NEEDED).
MITER = 1 MEANS CHORD METHOD WITH ANALYTIC JACOBIAN.
FOR THIS USER SUPPLIES SUBROUTINE
FEDERV (SEE DESCRIPTION BELOW).
MITER = 2 MEANS CHORD METHOD WITH JACOBIAN CALCULATED
INTERNALLY BY FINITE DIFFERENCES.
MITER = 3 MEANS CHORD METHOD WITH JACOBIAN REPLACED
BY A DIAGONAL APPROXIMATION BASED ON A
DIRECTIONAL DERIVATIVE.
INDEX = INTEGER USED ON INPUT TO INDICATE TYPE OF CALL,
WITH THE FOLLOWING VALUES AND MEANINGS:
1 THIS IS THE FIRST CALL FOR THIS PROBLEM.
0 THIS IS NOT THE FIRST CALL FOR THIS PROBLEM,
AND INTEGRATION IS TO CONTINUE.
-1 THIS IS NOT THE FIRST CALL FOR THE PROBLEM,
AND THE USER HAS RESET N, EPS, AND/OR MF.
2 SAME AS 0 EXCEPT THAT TOUT IS TO BE HIT
EXACTLY (NO INTERPOLATION IS DONE).
ASSUMES TOUT \geq \text{THE CURRENT T}.
3 SAME AS 0 EXCEPT CONTROL RETURNS TO CALLING
PROGRAM AFTER ONE STEP. TOUT IS IGNORED.
SINCE THE NORMAL OUTPUT VALUE OF INDEX IS 0,
IT NEED NOT BE RESET FOR NORMAL CONTINUATION.

AFTER THE INITIAL CALL, IF A NORMAL RETURN OCCURRED AND A NORMAL
CONTINUATION IS DESIRED, SIMPLY RESET TOUT AND CALL AGAIN.
ALL OTHER PARAMETERS WILL BE READY FOR THE NEXT CALL.
A CHANGE OF PARAMETERS WITH INDEX = -1 CAN BE MADE AFTER
EITHER A SUCCESSFUL OR AN UNSUCCESSFUL RETURN.

THE OUTPUT PARAMETERS ARE..
H0 = THE STEP SIZE H USED LAST, WHETHER SUCCESSFULLY OR NOT.
Y0 = THE COMPUTED VALUES OF Y AT T = TOUT.
TOUT = THE OUTPUT VALUE OF T. IF INTEGRATION WAS SUCCESSFUL,
AND THE INPUT VALUE OF INDEX WAS NOT 3, TOUT IS
UNCHANGED FROM ITS INPUT VALUE. OTHERWISE, TOUT
IS THE CURRENT VALUE OF T TO WHICH INTEGRATION
HAS BEEN COMPLETED.
INDEX = INTEGER USED ON OUTPUT TO INDICATE RESULTS,
WITH THE FOLLOWING VALUES AND MEANINGS:
0 INTEGRATION WAS COMPLETED TO TOUT OR BEYOND.
-1 THE INTEGRATION WAS HALTED AFTER FAILING TO PASS THE
ERROR TEST EVEN AFTER REDUCING H BY A FACTOR OF
1.E10 FROM ITS INITIAL VALUE.
-2 AFTER SOME INITIAL SUCCESS, THE INTEGRATION WAS
HALTED EITHER BY REPEATED ERROR TEST FAILURES OR BY
A TEST ON EPS. TOO MUCH ACCURACY HAS BEEN REQUESTED.
-3 THE INTEGRATION WAS HALTED AFTER FAILING TO ACHIEVE
CORRECTOR CONVERGENCE EVEN AFTER REDUCING H BY A
FACTOR OF 1.E10 FROM ITS INITIAL VALUE.
IMMEDIATE HALT BECAUSE OF ILLEGAL VALUES OF INPUT PARAMETERS. SEE PRINTED MESSAGE.
INDEX WAS -1 ON INPUT, BUT THE DESIRED CHANGES OF PARAMETERS WERE NOT IMPLEMENTED BECAUSE TOUT WAS NOT BEYOND T. INTERPOLATION TO T = TOUT WAS PERFORMED AS ON A NORMAL RETURN. TO TRY AGAIN, SIMPLY CALL AGAIN WITH INDEX = -1 AND A NEW TOUT.

IN ADDITION TO DRIVE, THE FOLLOWING ROUTINES ARE PROVIDED IN THE PACKAGE.
INTERP(TOUT,Y,N0,Y0) INTERPOLATES TO GET THE OUTPUT VALUES AT T = TOUT, FROM THE DATA IN THE Y ARRAY.
STIFF(N0) IS THE CORE INTEGRATOR ROUTINE. IT PERFORMS A SINGLE STEP AND ASSOCIATED ERROR CONTROL.
COSER(METH,NQ,EL,TQ,MAXDER) SETS COEFFICIENTS FOR USE IN THE CORE INTEGRATOR.
PSET(Y,N0,CON,MITER,IER) COMPUTES AND PROCESSES THE JACOBIAN MATRIX J = DF/DY.
DEC(N,N0,A,IP,IER) PERFORMS AN LU DECOMPOSITION ON A MATRIX.
SOL(N,N0,A,B,IF) SOLVES LINEAR SYSTEMS A*X = B AFTER DEC HAS BEEN CALLED FOR THE MATRIX A.
NOTE.. PSET, DEC, AND SOL ARE CALLED ONLY IF MITER = 1 OR 2.

THE FOLLOWING ROUTINES ARE TO BE SUPPLIED BY THE USER.
HERE Y AND YDOT ARE VECTORS OF LENGTH N.
PEDERV(N,T,Y,PD,N0) COMPUTES THE N BY N JACOBIAN MATRIX OF PARTIAL DERIVATIVES, AND STORES IT IN PD AS AN N0 BY N0 ARRAY. PD(I,J) IS TO BE SET TO THE PARTIAL DERIVATIVE OF YDOT(I) WITH RESPECT TO Y(J). PEDERV IS CALLED ONLY IF MITER = 1. OTHERWISE A DUMMY ROUTINE CAN BE SUBSTITUTED.

THE DIMENSIONS IN THE FOLLOWING DECLARATIONS ARE SET FOR A MAXIMUM OF 20 EQUATIONS. IF THE PACKAGE IS TO BE USED FOR A LARGER VALUE OF N, THE DIMENSIONS SHOULD BE INCREASED ACCORDINGLY. THE DIMENSION OF PW BELOW MUST BE AT LEAST N**2 IF MITER = 1 OR 2, BUT CAN BE REDUCED TO N IF MITER = 3, OR TO 1 IF MITER = 0.
THE IPIV ARRAY IS USED ONLY IF MITER IS 1 OR 2.

THE COMMON BLOCK GEAR9 CAN BE ACCESSED EXTERNALLY BY THE USER IF DESIRED. IT CONTAINS THE STEP SIZE LAST USED (SUCCESSFULLY), THE ORDER LAST USED (SUCCESSFULLY), THE NUMBER OF STEPS TAKEN SO FAR, THE NUMBER OF F EVALUATIONS (DIFFUN CALLS) SO FAR, AND THE NUMBER OF JACOBIAN EVALUATIONS SO FAR.

IN THE FOLLOWING DATA STATEMENT, SET...
LOUT = THE LOGICAL UNIT NUMBER FOR THE OUTPUT OF MESSAGES DURING THE INTEGRATION.

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)

 INTEGER N,MF,INDEX
 INTEGER NC,MFC,KFLAG,JSTART,IPIV,NSQ,NQUSED,NSTEP,NFE,NJE
 INTEGER LOUT,I,N0,NHCU1T,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,AAYI,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 .EQ. 1) GO TO 430
 IF (EPS .LE. 0.0D0) GO TO 400
 IF (N .LE. 0) GO TO 410
 IF ((T0-TOUT)*H0 .GE. 0.0D0) GO TO 420

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. 0.0D0) YMAX(I) = 1.0D0
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)*10.0D0
EPSC = EPS
MFC = MF
JSTART = 0
N0 = N
NSQ = N0*N0
EPSJ = DSQRT(UROUND)
NHCUT = 0
GO TO 50

C TOUP IS THE PREVIOUS VALUE OF TOUT FOR USE IN HMAX.
20 HMAX = DABS(TOUT-TOUP)*10.
GO TO 80

C 25 HMAX = DABS(TOUT-TOUP)*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, NO)

C 60 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 C THEY SHOULD BE SET HERE. A TEST IS MADE FOR EPS BEING TOO SMALL
C C FOR THE MACHINE PRECISION.

C C ANY OTHER TESTS OR CALCULATIONS THAT ARE REQUIRED AFTER EVERY
C C STEP SHOULD BE INSERTED HERE.

C C IF INDEX = 3, YO IS SET TO THE CURRENT Y VALUES ON RETURN.
C C IF INDEX = 2, H IS CONTROLLED TO HIT TOUT (WITHIN ROUNDOFF
C C ERROR), AND THEN THE CURRENT Y VALUES ARE PUT IN Y0 ON RETURN.
C C FOR ANY OTHER VALUE OF INDEX, CONTROL RETURNS TO THE INTEGRATOR
C C UNLESS TOUT HAS BEEN REACHED. THEN INTERPOLATED VALUES OF Y ARE
C C COMPUTED AND STORED IN Y0 ON RETURN.
C C IF INTERPOLATION IS NOT DESIRED, THE CALL TO INTERP SHOULD BE
C 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)
   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
70   IF ((T-TOUT)*H .LT. 0.D0) GO TO 40
CALL INTERP (TOUT, Y, NO, Y0)
GO TO 520
80   IF ((T+H)-TOUT)*H .LE. 0.D0) GO TO 40

A-31
IF (DABS(T-OUT) .LE. 100.D0*UROUND*HMAX) GO TO 500
IF (((T-OUT)*H .GE. 0.D0) GO TO 500
H = (TOUT - T)*(1.D0 - 4.D0*UROUND)
JSTART = -1
GO TO 40

---------------------------------------------------------------------
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.
---------------------------------------------------------------------

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 (NHPUT .EQ. 10) GO TO 150
110   NHPUT = NHPUT + 1
110   HMIN = HMIN*.1D0
110   H = H*.1D0
110   WRITE (LOUT,115) H
115 FORMAT(24H H HAS BEEN REDUCED TO ,E16.8,/
1    26H AND STEP WILL BE RETRIED//)
115   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    16H .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    16H EPS TOO SMALL TO BE ATTAINED FOR THE MACHINE PRECISION//)
255   KFLAG = -2
GO TO 500

C
300 WRITE (LOUT,305) T
305 FORMAT('/35H KFLAG = -3 FROM INTEGRATOR AT T = ,E16.8/
1    16H CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED//)
GO TO 110

C
400 WRITE (LOUT,405)
405 FORMAT('/28H ILLEGAL INPUT.. EPS .LE. 0.//)
405   INDEX = -4
RETURN

C
410 WRITE (LOUT,415)
415 FORMAT('/25H ILLEGAL INPUT.. N .LE. 0.//)
415   INDEX = -4
RETURN

C
420 WRITE (LOUT,425)
425 FORMAT('/36H ILLEGAL INPUT.. (T0-OUT)*H .GE. 0.//)
425   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  4IH DESIRED PARAMETER CHANGES WERE NOT MADE.)
      CALL INTERP (TOUT, Y, NO, Y0)
      INDEX = -5
      RETURN
C
500  TOUT = T
500  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

********************************************************************************

AUTOMATIC MESH GENERATION PROGRAM

********************************************************************************

Purposes: This program calculates the IXX, IYY and area of the drill cross-section after generating the boundary mesh. 5 points that specify the drill cross-section is input to the program.

Purpose 1: Program generates the mesh points along the boundary using the following subroutines:

READ1, BOUNEQ, DIVL1, DIVL2, DIVL3, DIVL4

READ1: Reads the 5 points that specify the drill boundary
BOUNEQ: Calculates the equations along the boundary
DIVL1: Generates the mesh points along the first boundary
DIVL2: >> >> >> >> second boundary
DIVL3: >> >> >> >> third boundary
DIVL4: >> >> >> >> fourth boundary

Purpose 2: Program generates the inner triangular mesh using Subroutine mesh This subroutine calls the following subroutines:
ASSIGN: Assigns the boundary mesh vector to X,Y matrices
SOR: Generates the inner mesh using Poisson's equation
CONNEC: Calculates the triangular element connectivities
PLOT1: Plots the triangular grid
PNUM: Numbers the elements on the plot

********************************************************************************

Arguments:
XA,YA: The vectors that keep 5 points of drill geometry
NN,MM: Number of mesh divisions along the boundaries
A,B,C,D,E,F,R: Coefficients of boundary equations
XB,YB: Stores the drill boundary mesh coordinates
KJ: Controls the boundary mesh storage order
XE,YE: Drill triangular mesh point coordinates
IJK: Element connectivities
NEL: Number of elements
NNODE: Number of nodes
S1,S2: Principal Area moment of inertias
AREA: Drill cross-sectional area

B-1
SUBROUTINE COEFF(S1,S2,AREA)

IMPLICIT REAL*8(A-H, O-Z)
DIMENSION XA(5), YA(5), XB(400), YB(400),
&
   XE(10000), YE(10000), IJK(3,1000)

KJ=0

CALL READ1(NN,MM,XA,YA)

CALL BOUNEQ(XA,YA,A,B,C,D,E,F,R)

CALL DIVL1(XA,YA,XB,YB,KJ,NN,MM,A,B,C)

CALL DIVL2(XA,YA,XB,YB,KJ,NN,MM)

CALL DIVL3(XA,YA,XB,YB,KJ,NN,MM,D,E,F)

CALL DIVL4(XA,YA,XB,YB,KJ,NN,MM,R)

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

CALL MESH(XB,YB,NN,MM,XE,YE,IJK,N,M,NEL,NNODE)

CALL GAUSS(XE,YE,IJK,NEL,NNODE,SXX,SYY,AREA)
STOP
END

-----------------------------------

SUBROUTINE READ1

Purpose: This subroutine reads the 5 points that is
required to define the drill geometry and the
number of divisions in x and y directions.

Arguments:
XA,YA:    The vectors that keep the 5 points.
NN,MM:    Number of divisions along the boundary in X & Y
directions.

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
DO 101 J=1,5
XA(J)=0.D0
YA(J)=0.D0
101 CONTINUE
C
WRITE (6,106)
106 FORMAT(1******** 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
-----------------------------

SUBROUTINE BOUQE

Purpose: Calculates the boundary equations
using the 5 points specified along the boundary
Arguments:
XA,YA: 5 points that specify the drill boundary
A,B,C: Coefficients of the first boundary equation which
is specified as a circle.
D,E,F: Coefficients of the third boundary equation which
is specified as a parabola.
R: Radius of the circle that specifies the fourth
boundary equation.
Local Variables:
DET, F1,F2,F3

SUBROUTINE BOUQE(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

B-3
\[ D = \left( (YA(4) - YA(5)) + (2 \cdot XA(3) + A) / (2 \cdot YA(3) + B) \cdot (XA(4) - XA(5)) \right) / \]
\[ (-XA(4) - 2 \cdot XA(5) + 2 \cdot XA(4) + XA(5)) \]
\[ E = (2 \cdot XA(3) + A) / (2 \cdot YA(3) + B) - 2 \cdot XA(4) \cdot D \]
\[ F = YA(4) - E \cdot XA(4) - D \cdot XA(4) \cdot 2 \]

C
CALCULATE R
R = \left( XA(5) \cdot 2 + YA(5) \cdot 2 \right) \cdot 0.5
RETURN
END

SUBROUTINE DIV1L

Purpose: Generates the boundary mesh for the first boundary.

Arguments:
XA, YA: 5 points that specify the drill boundary
XB, YB: Stores the boundary mesh coordinates generated
KJ: Controls the boundary mesh storage
NN, MM: Number of divisions along the boundaries
A, B, C: Coefficients of the first boundary equation.

SUBROUTINE DIV1L(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)

WRITE(6, 200) XA(1), YA(1)
200 FORMAT (2F10.5)
KJ = 1
XB(KJ) = XA(1)
YB(KJ) = YA(1)

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

NN = NN - 2
DO 205 J = 1, NN
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

-----------------------------------------------

B-4
SUBROUTINE DIVL2

Purpose: Generate the boundary mesh for the second boundary.

Variables:
XA, YA: 5 points that specify the drill geometry
XB, YB: Boundary mesh
KJ: Controls the storage of boundary mesh
NN, MM: Number of divisions along the boundaries

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)

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

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

SUBROUTINE DIVL3

Purpose: Generates the boundary mesh for the third boundary

Variables:
XA, YA: 5 points that specify the drill geometry
XB, YB: Boundary mesh
KJ: Controls the storage of boundary mesh
NN, MM: Number of divisions along the boundaries
D, E, F: Coefficients of the third boundary equation

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)

DY3=(YA(5)-YA(4))/(NN-1)
YS=YA(4)
N1=NN-2
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
SUBROUTINE DIVL4
C
Purpose: Generates the boundary mesh for the fourth boundary
C
Variables:
XA,YA: 5 points that specify the drill geometry
XB,YB: Boundary mesh
KJ: Controls the storage of boundary mesh
NN,MM: Number of divisions along the boundaries
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
ML=MM-1
DO 505 J=1,ML
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

B-6
SUBROUTINE MESH
---------

Purpose: This program is for the finite element grid generation by using the Poisson's equations (in the rectangular coordinate form).

Variables:
- NEN = the number of nodes within one element
- FACTOR = the multiplication factor of the plot
- W = the constant in the SOR method
- MN = the upper limit of the iteration
- TOL = the tolerance of the iteration
- N = Number of divisions in X direction
- M = Number of divisions in Y direction
- NEL = Total number or elements
- NNODE = Total number of nodes

SUBROUTINE MESH (XB, YB, NN, MM,XE,YE,IJK,N,M,NEL,NNODE)

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),YP(400)

READ (5,109) N,M
109 FORMAT (3I5)
IF ((N+M) .NE. (NN+MM)) GOTO 111
NODE = 3
FACTOR = 1.
NELD = 1

Assign the boundary mesh to X & Y matrices
CALL ASSIGN(XB,YB,X,Y,N,M)

Compute the mesh points for X(I,J), Y(I,J):
CALL SOR(X,Y,N,M)

Get the element connectivities:
CALL CONNEC(X,Y,XE,YE,IJK,N,M,NEL,NODE,NNODE)

Assign double precision X,Y,XE,YE arrays to XP,YP,XEP,YEP single precision arrays for plotting purposes.
DO 80 I=1,N
DO 80 J=1,M
XP(I,J)=X(I,J)
YP(I,J)=Y(I,J)
80 CONTINUE
MAX=N*M
DO 90 I=1,MAX
XEP(I)=XE(I)
YEP(I)=YE(I)
90 CONTINUE

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(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

SUBROUTINE SOR

Purpose: This routine provides the solution to the Poisson's
equations by using the SOR iteration method.

SUBROUTINE SOR(X,Y,N,M)

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION XO(100,100),X(100,100)
DIMENSION YO(100,100),Y(100,100)

W=1.24
MN=400
TOL=0.01
NN=N-1
MM=M-1

Input the initial approximations XO(I,J),YO(I,J):

DO 719 I=1,N
DO 719 J=1,M
   XO(I,J)=X(I,J)
   YO(I,J)=Y(I,J)
719 CONTINUE
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

Input XO(I,J),YO(I,J) for the internal points:
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

This sets an iteration limit.
DO 800 K=1, MN
DO 700 I=2, NN
DO 600 J=2, MM

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))

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))

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))

AIJ=1./((2. *(AF+GF))

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))

600 CONTINUE
700 CONTINUE

DO 500 I=1,N
DO 500 J=1,M

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
500 CONTINUE

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('NO ITERATION LIMIT EXCEEDED')
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
SUBROUTINE PLOT1
C
Purpose: This routine plots the grids generated by SOR.
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(100000),YE(100000),IJK(3,1000)
C
CALL PGGRID(0.0,0.0,0.0,8.5,11.0,1,1)
CALL PAXIS(2.0,1.65,'X-AXIS (INCHES)',-15,6.0,0.0,0.0,1.0,1.0)
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 PLOTOFS(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 PLOTOFS(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',0.1,35,0)
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 PLOFS(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

-------------------------------
SUBROUTINE CONNEC

Purpose: This routine defines the element connectivities for
the generated elements.

SUBROUTINE CONNEC(X,Y,XE,YE,IJK,N,M,NEN,NNODE)

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION X(100,100),Y(100,100)
DIMENSION IJK(3,10000),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

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

---------------------------------------------------------------------
SUBROUTINE ASSIGN
C

This program assigns the boundary mesh XB,YB, to the matrices X,Y where the
inner mesh be generated later on.
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
  II=N-I+1
  I2=N+M+I-2
  X(II,M)=XB(I2)
  Y(II,M)=YB(I2)
  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)
  CONTINUE
RETURN
END

C
SUBROUTINE PNUM
C
Purpose: This routine plot the node and element number by using
the connectivities.
  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=0.0
  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
  XM=XM-0.11
  CALL PNUMBR(XM,YM,0.08,1,0.0,'I3*',0)
  CALL PCIRCL(XM,YM,0.0,360.0,0.15,0.15,0.09,0)
399 CONTINUE
C
  CALL PNUMBR(6.0,0.8,0.15,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 PRTEND
RETURN
END

---------------------------------

SUBROUTINE GAUSS

Purpose: This subroutine calculates the area, area moment of inertias of a cross-section using 2nd order gaussian quadrature for triangular elements.

Variables:

NODE: Number of nodes
NEL: Number of elements
XE,YE: Nodal coordinates
IJK: Element connectivity
AREA: Area of an element
S1,S2: Principal area moment of inertias

SUBRoutines:

FUNC: Result of the integrations over each element

SUBROUTINE GAUSS(XE,YE,IJK,NEL,NNODE,SXX,SYY,AREA)

IMPLICIT REAL*8(A-H,O-Z)
DIMENSION XE(10000),YE(10000),IJK(3,1000),AX(3),AY(3)

SXX=0.D0
SYY=0.D0
SXY=0.D0
AREA=0.

DO 100 J=1,NEL
   DO 80 I=1,3
      IP=IJK(I,J)
      AX(I)=XE(IP)
      AY(I)=YE(IP)

 80 CONTINUE

CALL FUNCTN(AX,AY,SXX,SYY,SXY,AREA)

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,/)
SUBROUTINE FUNCTN

Purpose: This subroutine calculates the area and moment of inertias over each element.

SUBROUTINE FUNCTN(AX,AY,SXX,SYY,XYX,AREA)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION AX(3),AY(3),BX(3),BY(3)

BX(1)=(AX(1)+AX(2))/2.
BY(1)=(AY(1)+AY(2))/2.

BX(2)=(AX(2)+AX(3))/2.
BY(2)=(AY(2)+AY(3))/2.

BX(3)=(AX(3)+AX(1))/2.
BY(3)=(AY(3)+AY(1))/2.

EYX=(BX(1)*BX(1)+BX(2)*BX(2)+BX(3)*BX(3))/3.
EXY=(BY(1)*BY(1)+BY(2)*BY(2)+BY(3)*BY(3))/3.
EXX=(BY(1)*BY(1)+BY(2)*BY(2)+BY(3)*BY(3))/3.

AREAL=((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.

SXX=SXX+DABS(AREAL*EXX)
SYY=SYY+DABS(AREAL*EXY)
XYX=XYX+DABS(AREAL*EXY)
AREA=AREA+DABS(AREAL)

RETURN
END

SUBROUTINE PRNCPL

Purpose: This subroutine calculates the area moment of inertias at the principal directions of the drill cross-section

SUBROUTINE PRNCPL(SXX,SYY,XYX,S1,S2,GAMMA)
IMPLICIT REAL*8(A-H,O-Z)

GAMMA=DATAN(2.*SYX/DABS(SXX-SYY))
R=SYX/DSIN(GAMMA)
C

\[ S_2 = R + \frac{(S_{XX} + S_{YY})}{2}. \]
\[ S_1 = S_2 - 2.0 \times R \]

C

RETURN

END