

GLOBALS.FOR

```

1  ! . . . . .
2  ! .
3  ! .           S T A P 9 0
4  ! .
5  ! .       AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .       Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! . . . . .
9
10 ! . Define global variables
11
12 module GLOBALS
13
14     integer, parameter :: MTOT = 10000    ! Speed storage available for execution
15     integer, parameter :: ITWO = 2      ! Double precision indicator
16                                     ! 1 - Single precision arithmetic
17                                     ! 2 - Double precision arithmetic
18
19     integer, parameter :: IELMNT=1     ! Unit storing element data
20     integer, parameter :: ILOAD=2     ! Unit storing load vectors
21     integer, parameter :: IIN=5       ! Unit used for input
22     integer, parameter :: IOUT=6      ! Unit used for output
23
24     integer :: NUMNP      ! Total number of nodal points
25                       ! = 0 : Program stop
26     integer :: NEQ       ! Number of equations
27     integer :: NWK       ! Number of matrix elements
28     integer :: MK        ! Maximum half bandwidth
29
30     integer :: IND       ! Solution phase indicator
31                       ! 1 - Read and generate element information
32                       ! 2 - Assemble structure stiffness matrix
33                       ! 3 - Stress calculations
34     integer :: NPAR(10) ! Element group control data
35                       ! NPAR(1) - Element type
36                       ! 1 : Truss element
37                       ! NPAR(2) - Number of elements
38                       ! NPAR(3) - Number of different sets of material and
39                       ! cross-sectional constants
40     integer :: NUMEG     ! Total number of element groups, > 0
41
42     integer :: MODEX     ! Solution mode: 0 - data check only; 1 - execution
43
44     real :: TIM(5)       ! Timing information
45     character*80 :: HED ! Master heading information for use in labeling the output
46
47     real :: A(MTOT)
48
49     integer :: NFIRST
50     integer :: NLAST
51     integer :: NUMEST
52     integer :: MIDEST
53     integer :: MAXEST
54
55     integer :: NG
56
57     ! Base addresses of arrays/matrices in array A/IA(MTOT)
58     integer :: N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15
59
60 end module GLOBALS

```

STAP90.FOR

```

1  ! . . . . .
2  ! .
3  ! .           S T A P 9 0
4  ! .
5  ! .   AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .   Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! . . . . .
9  C
10 C   PROGRAM STAP90
11 C
12 C   USE GLOBALS
13 C
14 C   IMPLICIT NONE
15 C   INTEGER :: NLCASE, NEQ1, NLOAD, MM, NNL, KTR
16 C   INTEGER :: L, LL, I
17 C   REAL :: TT
18 C
19 C   CALL OPENFILES()
20 C
21 C   NUMEST=0
22 C   MAXEST=0
23 C
24 C   * * * * *
25 C   *           INPUT PHASE           *
26 C   * * * * *
27 C
28 C   CALL SECOND (TIM(1))
29 C
30 C   Read control information
31 C
32 C   HED   - The master heading information for use in labeling the output
33 C   NUMNP - Total number of nodal points
34 C           0 : program stop
35 C   NUMEG - Total number of element group (>0)
36 C   NLCASE - Number of load case (>0)
37 C   MODEX - Solution mode
38 C           0 : data check only;
39 C           1 : execution
40 C   READ (IIN,1000) HED, NUMNP, NUMEG, NLCASE, MODEX
41 C   IF (NUMNP.EQ.0) STOP
42 C   WRITE (IOUT,2000) HED, NUMNP, NUMEG, NLCASE, MODEX
43 C
44 C   Read nodal point data
45 C
46 C   ALLOCATE STORAGE
47 C   N1 - ID(3, NUMNP) : Boundary condition codes (0=free,1=deleted)
48 C   N2 - X(NUMNP) : X coordinates
49 C   N3 - Y(NUMNP) : Y coordinates
50 C   N4 - Z(NUMNP) : Z coordinates
51 C   N1= 1
52 C   N2=N1 + 3*NUMNP
53 C   N2=(N2/2)*2 + 1
54 C   N3=N2 + NUMNP*ITWO
55 C   N4=N3 + NUMNP*ITWO
56 C   N5=N4 + NUMNP*ITWO
57 C   IF (N5.GT.MTOT) CALL ERROR (N5-MTOT,1)
58 C
59 C   CALL INPUT (A(N1), A(N2), A(N3), A(N4), NUMNP, NEQ)
60 C
61 C   NEQ1=NEQ + 1
62 C
63 C   Calculate and store load vectors
64 C
65 C   N5 - R(NUMNP) : Load vector
66 C   N6=N5 + NEQ*ITWO
67 C   WRITE (IOUT,2005)
68 C
69 C   REWIND ILOAD
70 C
71 C   DO L=1, NLCASE
72 C
73 C   LL - Load case number
74 C   NLOAD - The number of concentrated loads applied in this load case

```

STAP90.FOR

```

1      READ (IIN,1010) LL,NLOAD
2      C
3      WRITE (IOUT,2010) LL,NLOAD
4      C
5      IF (LL.NE.L) THEN
6          WRITE (IOUT,2020)
7          STOP
8      ENDIF
9      C
10     C Allocate storage
11     C N6 - NOD(NLOAD) : Node number to which this load is applied (1~NUMNP)
12     C N7 - IDIRN(NLOAD) : Degree of freedom number for this load component
13     C 1 : X-direction; 2 : Y-direction; 3 : Z-direction
14     C N8 - FLOAD(NLOAD) : Magnitude of load
15     C N7=N6 + NLOAD
16     C N8=N7 + NLOAD
17     C N9=N8 + NLOAD*ITWO
18     C
19     C IF (N9.GT.MTOT) CALL ERROR (N9-MTOT,2)
20     C
21     C CALL LOADS (A(N5),A(N6),A(N7),A(N8),A(N1),NLOAD,NEQ)
22     C
23     C END DO
24     C
25     C Read, generate and store element data
26     C
27     C Clear storage
28     C N5 - MHT(NUMNP) - Vector of column heights
29     C N6 - (2*NUMMAT*ITWO+7*NUME+6*NUME*ITWO) : Element group data
30     C N6=N5 + NEQ
31     C N6=(N6/2)*2 + 1
32     C DO I=N5,N6
33     C A(I)=0
34     C END DO
35     C
36     C IND=1
37     C
38     C CALL ELCAL
39     C
40     C CALL SECOND (TIM(2))
41     C
42     C * * * * *
43     C * SOLUTION PHASE *
44     C * * * * *
45     C
46     C Assemble stiffness matrix
47     C
48     C CALL ADDRES (A(N2),A(N5))
49     C
50     C N2 - MAXA(NEQ+1)
51     C N3 - A(NWK) - Global structure stiffness matrix K
52     C N4 - R(NUMNP) - Load vector R and then displacement solution U
53     C N5 - (2*NUMMAT*ITWO+7*NUME+6*NUME*ITWO) : Element group data
54     C MM=NWK/NEQ
55     C N3=N2 + NEQ + 1
56     C N3=(N3/2)*2 + 1
57     C N4=N3 + NWK*ITWO
58     C N5=N4 + NEQ*ITWO
59     C N6=N5 + MAXEST
60     C IF (N6.GT.MTOT) CALL ERROR (N6-MTOT,4)
61     C
62     C Write total system data
63     C
64     C WRITE (IOUT,2025) NEQ,NWK,MK,MM
65     C
66     C In data check only mode we skip all further calculations
67     C
68     C IF (MODEX.LE.0) THEN
69     C CALL SECOND (TIM(3))
70     C CALL SECOND (TIM(4))
71     C CALL SECOND (TIM(5))
72     C ELSE
73     C
74     C Clear storage

```

STAP90. FOR

```

1  C
2     >NNL=NWK + NEQ
3      CALL CLEAR (A(N3),NNL)
4  C
5      IND=2
6  C
7      CALL ASSEM (A(N5))
8  C
9      CALL SECOND (TIM(3))
10 C
11 C    Triangularize stiffness matrix
12 C
13     KTR=1
14     CALL COLSOL (A(N3),A(N4),A(N2),NEQ,NWK,NEQ1,KTR)
15 C
16     CALL SECOND (TIM(4))
17 C
18     KTR=2
19     IND=3
20 C
21     REWIND ILOAD
22     DO L=1,NLCASE
23 C
24         CALL LOADV (A(N4),NEQ)
25 C
26 C    Calculation of displacements
27 C
28         CALL COLSOL (A(N3),A(N4),A(N2),NEQ,NWK,NEQ1,KTR)
29 C
30         WRITE (IOUT,2015) L
31         CALL WRITED (A(N4),A(N1),NEQ,NUMNP)
32 C
33 C    Calculation of stresses
34 C
35         CALL STRESS (A(N5))
36 C
37     END DO
38 C
39     CALL SECOND (TIM(5))
40 END IF
41 C
42 C    Print solution times
43 C
44     TT=0.
45     DO I=1,4
46         TIM(I)=TIM(I+1) - TIM(I)
47         TT=TT + TIM(I)
48     END DO
49
50     WRITE (IOUT,2030) HED, (TIM(I), I=1,4), TT
51 C
52     STOP
53 C
54     1000 FORMAT (A80,/,4I5)
55     1010 FORMAT (2I5)
56 C
57     2000 FORMAT (///,' ',A80,///,
58     1 ' CONTROL INFORMATION',//,
59     2 ' NUMBER OF NODAL POINTS',10(' .'),' (NUMNP) = ',I5,//,
60     3 ' NUMBER OF ELEMENT GROUPS',9(' .'),' (NUMEG) = ',I5,//,
61     4 ' NUMBER OF LOAD CASES',11(' .'),' (NLCASE) = ',I5,//,
62     5 ' SOLUTION MODE ',14(' .'),' (MODEX) = ',I5,/,
63     6 ' EQ. 0, DATA CHECK',/,
64     7 ' EQ. 1, EXECUTION')
65     2005 FORMAT (//,' LOAD CASE DATA')
66     2010 FORMAT (////,' LOAD CASE NUMBER',7(' .'),' = ',I5,//,
67     1 ' NUMBER OF CONCENTRATED LOADS . = ',I5)
68     2015 FORMAT (//,' LOAD CASE ',I3)
69     2020 FORMAT (' *** ERROR *** LOAD CASES ARE NOT IN ORDER')
70     2025 FORMAT (//,' TOTAL SYSTEM DATA',///,
71     1 ' NUMBER OF EQUATIONS',14(' .'),' (NEQ) = ',I5,//,
72     2 ' NUMBER OF MATRIX ELEMENTS',11(' .'),' (NWK) = ',I5,//,
73     3 ' MAXIMUM HALF BANDWIDTH ',12(' .'),' (MK) = ',I5,//,
74     4 ' MEAN HALF BANDWIDTH',14(' .'),' (MM) = ',I5)

```

```

                                STAP90.FOR
1  2030 FORMAT (//,' S O L U T I O N   T I M E   L O G   I N   S E C',//,
2  1 '           FOR PROBLEM',//,' ',A80,///,
3  2 '           TIME FOR INPUT PHASE ',14(' . '), ' =',F12.2,//,
4  3 '           TIME FOR CALCULATION OF STIFFNESS MATRIX . . . . =',F12.2,
5  4 //,
6  5 '           TIME FOR FACTORIZATION OF STIFFNESS MATRIX . . . =',F12.2,
7  6 //,
8  7 '           TIME FOR LOAD CASE SOLUTIONS ',10(' . '), ' =',F12.2,///,
9  8 '           T O T A L   S O L U T I O N   T I M E . . . . =',F12.2)
10 C
11   END
12
13   SUBROUTINE ERROR (N, I)
14 C . . . . .
15 C . . . . .
16 C .   P R O G R A M . . . . .
17 C .   TO PRINT MESSAGES WHEN HIGH-SPEED STORAGE IS EXCEEDED . . . . .
18 C . . . . .
19 C . . . . .
20   USE GLOBALS, ONLY : IOUT
21 C
22   IMPLICIT NONE
23   INTEGER :: N, I
24 C
25   IF (I == 1) THEN
26     WRITE (IOUT,2000)
27   ELSE IF (I == 2) THEN
28     WRITE (IOUT,2010)
29   ELSE IF (I == 3) THEN
30     WRITE (IOUT,2020)
31   ELSE IF (I == 4) THEN
32     WRITE (IOUT,2030)
33   END IF
34 C
35   WRITE (IOUT,2050) N
36   STOP
37 C
38   2000 FORMAT (//,' NOT ENOUGH STORAGE FOR ID ARRAY AND NODAL POINT ',
39   1 ' COORDINATES')
40   2010 FORMAT (//,' NOT ENOUGH STORAGE FOR DEFINITION OF LOAD VECTORS')
41   2020 FORMAT (//,' NOT ENOUGH STORAGE FOR ELEMENT DATA INPUT')
42   2030 FORMAT (//,' NOT ENOUGH STORAGE FOR ASSEMBLAGE OF GLOBAL ',
43   1 ' STRUCTURE STIFFNESS, AND DISPLACEMENT AND STRESS SOLUTION PHASE')
44   2050 FORMAT (//,' *** ERROR *** STORAGE EXCEEDED BY ', I9)
45 C
46   END
47 C
48   SUBROUTINE SECOND (TIM)
49   USE DFPORT
50   IMPLICIT NONE
51   REAL :: TIM
52 C
53 C   This is a Fortran 95 intrinsic subroutine
54 C   Returns the processor time in seconds
55 C
56   CALL CPU_TIME(TIM)
57 C
58   RETURN
59   END
60
61   SUBROUTINE CLEAR (A, N)
62 C . . . . .
63 C . . . . .
64 C . . . . .
65 C .   To clear double precision array A . . . . .
66 C . . . . .
67 C . . . . .
68 C
69   IMPLICIT NONE
70 C
71   INTEGER :: N, I
72   REAL(8) :: A(N)
73 C
74   DO I=1, N

```

STAP90. FOR

```

1      A(I)=0.
2      END DO
3      RETURN
4      END
5
6
7      SUBROUTINE WRITED (DISP, ID, NEQ, NUMNP)
8      C . . . . .
9      C . . . . .
10     C . To print displacements . . . . .
11     C . . . . .
12     C
13     USE GLOBALS, ONLY : IOUT
14     C
15     IMPLICIT NONE
16     INTEGER :: NEQ, NUMNP, ID(3, NUMNP)
17     REAL(8) :: DISP(NEQ), D(3)
18     INTEGER :: IC, II, I, KK, IL
19     C
20     C Print displacements
21     C
22     WRITE (IOUT, 2000)
23     IC=4
24     C
25     DO II=1, NUMNP
26         IC=IC + 1
27         IF (IC. GE. 56) THEN
28             WRITE (IOUT, 2000)
29             IC=4
30         END IF
31     C
32     DO I=1, 3
33         D(I)=0.
34     END DO
35     C
36     DO I=1, 3
37         KK=ID(I, II)
38         IL=I
39         IF (KK. NE. 0) D(IL)=DISP(KK)
40     END DO
41     C
42     WRITE (IOUT, 2010) II, D
43     END DO
44     C
45     RETURN
46     C
47     2000 FORMAT (///, ' D I S P L A C E M E N T S', //, ' NODE ', 10X,
48     1 ' X-DISPLACEMENT Y-DISPLACEMENT Z-DISPLACEMENT')
49     2010 FORMAT (1X, I3, 8X, 3E18. 6)
50     C
51     END
52
53
54     SUBROUTINE OPENFILES ()
55     USE GLOBALS
56     IMPLICIT NONE
57     LOGICAL :: EX
58     !
59     INQUIRE(FILE = "STAP90. IN", EXIST = EX)
60     IF (. NOT. EX) THEN
61         PRINT *, "*** STOP *** FILE STAP90. IN DOES NOT EXIST !"
62         STOP
63     END IF
64
65     OPEN(IIN , FILE = "STAP90. IN", STATUS = "OLD")
66     OPEN(IOUT , FILE = "STAP90. OUT", STATUS = "REPLACE")
67
68     OPEN(IELMNT, FILE = "ELMNT. TMP", FORM = "UNFORMATTED",
69     1 STATUS = "SCRATCH")
70     OPEN(ILOAD , FILE = "LOAD. TMP", FORM = "UNFORMATTED",
71     1 STATUS = "SCRATCH")
72     END SUBROUTINE OPENFILES
73
74

```

```
1  SUBROUTINE CLOSEFILES ()  
2  USE GLOBALS  
3  IMPLICIT NONE  
4  CLOSE (IIN)  
5  CLOSE (IOUT)  
6  CLOSE (IELMNT)  
7  CLOSE (ILOAD)  
8  END SUBROUTINE CLOSEFILES
```

DATAIN90.FOR

```

1  ! .....
2  ! .
3  ! .           S T A P 9 0
4  ! .
5  ! .   AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .   Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! .....
9  !
10 ! SUBROUTINE INPUT (ID, X, Y, Z, NUMNP, NEQ)
11 ! .....
12 ! .
13 ! .   To read, generate, and print nodal point input data
14 ! .   To calculate equation numbers and store them in id array
15 ! .
16 ! .       N = Element number
17 ! .       ID = Boundary condition codes (0=free,1=deleted)
18 ! .       X,Y,Z = Coordinates
19 ! .       KN = Generation code
20 ! .           i.e. increment on nodal point number
21 ! .
22 ! .....
23 !
24 ! USE GLOBALS, ONLY : IIN, IOUT
25 !
26 ! IMPLICIT NONE
27 ! INTEGER :: NUMNP, NEQ, ID(3, NUMNP)
28 ! REAL(8) :: X(NUMNP), Y(NUMNP), Z(NUMNP)
29 ! REAL(8) :: XNUM, DX, DY, DZ
30 ! INTEGER :: KNOLD, NOLD, NUM, NUMN
31 ! INTEGER :: I, J, K, KN, N, KK
32 !
33 ! Read and generate nodal point data
34 !
35 ! WRITE (IOUT,2000)
36 ! WRITE (IOUT,2010)
37 ! WRITE (IOUT,2020)
38 !
39 ! KNOLD=0
40 ! NOLD=0
41 !
42 ! N = 0
43 ! DO WHILE (N.NE.NUMNP)
44 !   READ (IIN,1000) N, (ID(I,N), I=1,3), X(N), Y(N), Z(N), KN
45 !   WRITE (IOUT,2030) N, (ID(I,N), I=1,3), X(N), Y(N), Z(N), KN
46 !   IF (KNOLD.NE.0) THEN
47 !     NUM=(N-NOLD)/KNOLD
48 !     NUMN=NUM - 1
49 !     IF (NUMN.GE.1) THEN
50 !       XNUM=NUM
51 !       DX=(X(N)-X(NOLD))/XNUM
52 !       DY=(Y(N)-Y(NOLD))/XNUM
53 !       DZ=(Z(N)-Z(NOLD))/XNUM
54 !       K=NOLD
55 !       DO J=1, NUMN
56 !         KK=K
57 !         K=K + KNOLD
58 !         X(K)=X(KK) + DX
59 !         Y(K)=Y(KK) + DY
60 !         Z(K)=Z(KK) + DZ
61 !         DO I=1, 3
62 !           ID(I, K)=ID(I, KK)
63 !         END DO
64 !       END DO
65 !     END IF
66 !   ENDIF
67 !
68 !   50 NOLD=N
69 !     KNOLD=KN
70 !   END DO
71 !
72 ! Write complete nodal data
73 !
74 ! WRITE (IOUT,2015)

```


DATAIN90. FOR

```

1      WRITE (IOUT, 2020)
2      DO N=1, NUMNP
3          WRITE (IOUT, 2030) N, (ID(I, N), I=1, 3), X(N), Y(N), Z(N), KN
4      END DO
5      !
6      !      Number unknowns
7      !
8      NEQ=0
9      DO N=1, NUMNP
10         DO I=1, 3
11             IF (ID(I, N) .EQ. 0) THEN
12                 NEQ=NEQ + 1
13                 ID(I, N)=NEQ
14             ELSE
15                 ID(I, N)=0
16             END IF
17         END DO
18     END DO
19     !
20     !      Write equation numbers
21     !
22     WRITE (IOUT, 2040) (N, (ID(I, N), I=1, 3), N=1, NUMNP)
23     !
24     RETURN
25     !
26     1000 FORMAT (4I5, 3F10.0, I5)
27     2000 FORMAT(//, ' N O D A L   P O I N T   D A T A ', //)
28     2010 FORMAT(' INPUT NODAL DATA', //)
29     2015 FORMAT(//, ' GENERATED NODAL DATA', //)
30     2020 FORMAT('  NODE', 10X, ' BOUNDARY', 25X, ' NODAL POINT', 17X, ' MESH', /,
31         1' NUMBER      CONDITION  CODES', 21X, ' COORDINATES', 14X, ' GENERATING',
32         2/, 77X, ' CODE', /,
33         315X, ' X      Y      Z', 15X, ' X', 12X, ' Y', 12X, ' Z', 10X, ' KN')
34     2030 FORMAT (I5, 6X, 3I5, 6X, 3F13.3, 3X, I6)
35     2040 FORMAT(//, ' EQUATION NUMBERS', //, '  NODE', 9X,
36         1' DEGREES OF FREEDOM', /, ' NUMBER', //,
37         2'      N', 13X, ' X      Y      Z', /, (1X, I5, 9X, 3I5))
38     !
39     END
40
41
42     SUBROUTINE LOADS (R, NOD, IDIRN, FLOAD, ID, NLOAD, NEQ)
43     ! . . . . .
44     ! .
45     ! . To read nodal load data .
46     ! . To calculate the load vector r for each load case and .
47     ! . write onto unit ILOAD .
48     ! .
49     ! . . . . .
50     USE GLOBALS, ONLY : IIN, IOUT, ILOAD, MODEX
51     !
52     IMPLICIT NONE
53     INTEGER :: NLOAD, NEQ, ID(3, 1), NOD(1), IDIRN(1)
54     REAL(8) :: R(NEQ), FLOAD(1)
55     INTEGER :: I, L, LI, LN, II
56     !
57     WRITE (IOUT, 2000)
58     READ (IIN, 1000) (NOD(I), IDIRN(I), FLOAD(I), I=1, NLOAD)
59     WRITE (IOUT, 2010) (NOD(I), IDIRN(I), FLOAD(I), I=1, NLOAD)
60     !
61     IF (MODEX.EQ. 0) RETURN
62     !
63     DO I=1, NEQ
64         R(I)=0.
65     END DO
66     !
67     DO L=1, NLOAD
68         LN=NOD(L)
69         LI=IDIRN(L)
70         II=ID(LI, LN)
71         IF (II > 0) R(II)=R(II) + FLOAD(L)
72     END DO
73     !
74     WRITE (ILOAD) R

```

DATAIN90. FOR

```

1  !
2      RETURN
3  !
4      1000 FORMAT (2I5, F10. 0)
5      2000 FORMAT (//, '      NODE      DIRECTION      LOAD', /,
6          1      '      NUMBER', 19X, ' MAGNITUDE')
7      2010 FORMAT (' ', I6, 9X, I4, 7X, E12. 5)
8  !
9      END
10
11
12      SUBROUTINE LOADV (R, NEQ)
13  ! . . . . .
14  ! .
15  ! .   To obtain the load vector
16  ! . . . . .
17  !
18      USE GLOBALS, ONLY : ILOAD
19  !
20      IMPLICIT NONE
21      INTEGER :: NEQ
22      REAL(8) :: R(NEQ)
23  !
24      READ (ILOAD) R
25  !
26      RETURN
27      END

```

ELCAL90. FOR

```

1  ! . . . . .
2  ! .
3  ! .           S T A P 9 0
4  ! .
5  ! .   AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .   Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! . . . . .
9  C
10 C   SUBROUTINE ELCAL
11 C . . . . .
12 C .
13 C .   To loop over all element groups for reading,
14 C .   generating and storing the element data
15 C .
16 C . . . . .
17 C   USE GLOBALS
18 C
19 C   IMPLICIT NONE
20 C   INTEGER :: N, I
21 C
22 C   REWIND IELMNT
23 C   WRITE (IOUT,2000)
24 C
25 C   Loop over all element groups
26 C
27 C   DO N=1,NUMEG
28 C     IF (N.NE.1) WRITE (IOUT,2010)
29 C
30 C     READ (IIN,1000) NPAR
31 C
32 C     CALL ELEMNT
33 C
34 C     IF (MIDEST.GT.MAXEST) MAXEST=MIDEST
35 C
36 C     WRITE (IELMNT) MIDEST, NPAR, (A(I), I=NFIRST, NLAST)
37 C
38 C   END DO
39 C
40 C   RETURN
41 C
42 C   1000 FORMAT (10I5)
43 C   2000 FORMAT (//,' E L E M E N T   G R O U P   D A T A',//)
44 C   2010 FORMAT (' ')
45 C
46 C   END
47
48
49 C   SUBROUTINE ELEMNT
50 C . . . . .
51 C .
52 C .   To call the appropriate element subroutine
53 C .
54 C . . . . .
55 C
56 C   USE GLOBALS
57 C
58 C   IMPLICIT NONE
59 C   INTEGER :: NPAR1
60 C
61 C   NPAR1=NPAR(1)
62 C
63 C   IF (NPAR1 == 1) THEN
64 C     CALL TRUSS
65 C   ELSE
66 C
67 C     Other element types would be called here, identifying each
68 C     element type by a different NPAR(1) parameter
69 C
70 C   END IF
71 C
72 C   RETURN
73 C   END
74

```

ELCAL90. FOR

```

1      SUBROUTINE TRUSS
2      C . . . . .
3      C .
4      C .   To set up storage and call the truss element subroutine   .
5      C .
6      C . . . . .
7      C
8      USE GLOBALS
9      C
10     IMPLICIT NONE
11     INTEGER :: NUME, NUMMAT, N101, N102, N103, N104, N105, N106
12     C
13     NUME = NPAR(2)
14     NUMMAT = NPAR(3)
15     C
16     C   Allocate storage
17     C
18     NFIRST=N6
19     IF (IND.GT.1) NFIRST=N5
20     N101=NFIRST
21     N102=N101 + NUMMAT*ITWO
22     N103=N102 + NUMMAT*ITWO
23     N104=N103 + 6*NUME
24     N105=N104 + 6*NUME*ITWO
25     N106=N105 + NUME
26     NLAST=N106
27     C
28     IF (IND.LE.1) THEN
29         IF (NLAST.GT.MTOT) CALL ERROR (NLAST-MTOT, 3)
30     ELSE
31         IF (NLAST.GT.MTOT) CALL ERROR (NLAST-MTOT, 4)
32     END IF
33     C
34     MIDEST=NLAST - NFIRST
35     C
36     CALL RUSS (A(N1), A(N2), A(N3), A(N4), A(N4), A(N5), A(N101), A(N102),
37     1         A(N103), A(N104), A(N105))
38     C
39     RETURN
40     C
41     END
42
43
44     SUBROUTINE RUSS (ID, X, Y, Z, U, MHT, E, AREA, LM, XYZ, MATP)
45     C . . . . .
46     C .
47     C .   TRUSS element subroutine   .
48     C .
49     C . . . . .
50     C
51     USE GLOBALS
52     C
53     IMPLICIT NONE
54     INTEGER :: ID(3, 1), LM(6, 1), MATP(1), MHT(1)
55     REAL(8) :: X(1), Y(1), Z(1), E(1), AREA(1), XYZ(6, 1), U(1)
56     REAL(8) :: S(21), ST(6), D(3)
57     C
58     INTEGER :: NPAR1, NUME, NUMMAT, ND, I, J, K, L, N, M, II, JJ
59     INTEGER :: MTYP, MTYPE, KG, KL, KKK, IPRINT
60     REAL(8) :: XL2, XL, SQRT, XX, YY, STR, P
61     C
62     NPAR1 = NPAR(1)
63     NUME = NPAR(2)
64     NUMMAT = NPAR(3)
65     C
66     ND=6
67     C
68     GO TO (300, 610, 800), IND
69     C
70     C   Read and generate element information
71     C
72     C   Read material information
73     C
74     300 WRITE (IOUT, 2000) NPAR1, NUME

```

```

1      IF (NUMMAT.EQ.0) NUMMAT=1
2      WRITE (IOUT,2010) NUMMAT
3      C
4      WRITE (IOUT,2020)
5      DO I=1, NUMMAT
6          READ (IIN,1000) N, E(N), AREA(N)
7          WRITE (IOUT,2030) N, E(N), AREA(N)
8      END DO
9      C
10     C      Read element information
11     C
12     WRITE (IOUT,2040)
13     N=1
14     100 READ (IIN,1020) M, II, JJ, MTYP, KG
15     IF (KG.EQ.0) KG=1
16
17     DO WHILE (.TRUE.)
18         IF (M.EQ.N) THEN
19             I=II
20             J=JJ
21             MTYPE=MTYP
22             KKK=KG
23         END IF
24     C
25     C      Save element information
26     C
27         XYZ(1,N)=X(I)
28         XYZ(2,N)=Y(I)
29         XYZ(3,N)=Z(I)
30     C
31         XYZ(4,N)=X(J)
32         XYZ(5,N)=Y(J)
33         XYZ(6,N)=Z(J)
34     C
35         MATP(N)=MTYPE
36     C
37         DO L=1,6
38             LM(L,N)=0
39         END DO
40
41         DO L=1,3
42             LM(L,N)=ID(L,I)
43             LM(L+3,N)=ID(L,J)
44         END DO
45     C
46     C      Update column heights and bandwidth
47     C
48         CALL COLHT (MHT,ND,LM(1,N))
49     C
50         WRITE (IOUT,2050) N, I, J, MTYPE
51         IF (N.EQ.NUME) RETURN
52     C
53         N=N + 1
54         I=I + KKK
55         J=J + KKK
56         IF (N.GT.M) GO TO 100
57     END DO
58     C
59     C      Assemble structure stiffness matrix
60     C
61     610 DO N=1, NUME
62         MTYPE=MATP(N)
63         XL2=0.
64         DO L=1,3
65             D(L)=XYZ(L,N) - XYZ(L+3,N)
66             XL2=XL2 + D(L)*D(L)
67         END DO
68         XL=SQRT(XL2)
69         XX=E(MTYPE)*AREA(MTYPE)*XL
70         DO L=1,3
71             ST(L)=D(L)/XL2
72             ST(L+3)=-ST(L)
73         END DO
74     C

```

```

1      KL=0
2      DO L=1, 6
3          YY=ST(L)*XX
4          DO K=L, 6
5              KL=KL + 1
6              S(KL)=ST(K)*YY
7          END DO
8      END DO
9      CALL ADDBAN (A(N3), A(N2), S, LM(1, N), ND)
10     END DO
11
12     RETURN
13 C
14 C     Stress calculations
15 C
16     800 IPRINT=0
17     DO 830 N=1, NUME
18         IPRINT=IPRINT + 1
19         IF (IPRINT.GT. 50) IPRINT=1
20         IF (IPRINT.EQ. 1) WRITE (IOUT, 2060) NG
21         MTYPE=MATP(N)
22         XL2=0.
23         DO L=1, 3
24             D(L) = XYZ(L, N) - XYZ(L+3, N)
25             XL2=XL2 + D(L)*D(L)
26         END DO
27         DO L=1, 3
28             ST(L)=(D(L)/XL2)*E(MTYPE)
29             ST(L+3)=-ST(L)
30         END DO
31         STR=0. 0
32         DO L=1, 3
33             I=LM(L, N)
34             IF (I.GT. 0) STR=STR + ST(L)*U(I)
35             J=LM(L+3, N)
36             IF (J.GT. 0) STR=STR + ST(L+3)*U(J)
37         END DO
38         P=STR*AREA(MTYPE)
39         WRITE (IOUT, 2070) N, P, STR
40     830 CONTINUE
41 C
42     1000 FORMAT (I5, 2F10. 0)
43     1010 FORMAT (2F10. 0)
44     1020 FORMAT (5I5)
45     2000 FORMAT (' E L E M E N T   D E F I N I T I O N', ///,
46         1      ' ELEMENT TYPE ', 13(' . '), ' ( NPAR(1) ) . . . =', I5, /,
47         2      '      EQ. 1, TRUSS ELEMENTS', /,
48         3      '      EQ. 2, ELEMENTS CURRENTLY', /,
49         4      '      EQ. 3, NOT AVAILABLE', //,
50         5      ' NUMBER OF ELEMENTS.', 10(' . '), ' ( NPAR(2) ) . . . =', I5, //)
51     2010 FORMAT (' M A T E R I A L   D E F I N I T I O N', ///,
52         1      ' NUMBER OF DIFFERENT SETS OF MATERIAL', /,
53         2      ' AND CROSS-SECTIONAL CONSTANTS ',
54         3      '      4(' . '), ' ( NPAR(3) ) . . . =', I5, //)
55     2020 FORMAT (' SET          YOUNG'S      CROSS-SECTIONAL', /,
56         1      ' NUMBER          MODULUS', 10X, ' AREA', /,
57         2      ' 15X, ' E', 14X, ' A')
58     2030 FORMAT (/, I5, 4X, E12. 5, 2X, E14. 6)
59     2040 FORMAT (//, ' E L E M E N T   I N F O R M A T I O N', ///,
60         1      ' ELEMENT      NODE      NODE      MATERIAL', /,
61         2      ' NUMBER=N      I      J      SET NUMBER', /)
62     2050 FORMAT (I5, 6X, I5, 4X, I5, 7X, I5)
63     2060 FORMAT (///, ' S T R E S S   C A L C U L A T I O N S   F O R ',
64         1      ' E L E M E N T   G R O U P', I4, //,
65         2      ' ELEMENT', 13X, ' FORCE', 12X, ' STRESS', /,
66         3      ' NUMBER', /)
67     2070 FORMAT (1X, I5, 11X, E13. 6, 4X, E13. 6)
68 C
69     END
70
71
72     SUBROUTINE STRESS (AA)
73 C . . . . .
74 C .

```

ELCAL90.FOR

```

1  C .   To call the element subroutine for the calculation of stresses .
2  C .
3  C . . . . .
4  C
5      USE GLOBALS, ONLY : IELMNT, NG, NUMEST, NPAR, NUMEG
6  C
7      IMPLICIT NONE
8      REAL :: AA(1)
9      INTEGER N, I
10 C
11 C   Loop over all element groups
12 C
13     REWIND IELMNT
14 C
15     DO N=1, NUMEG
16         NG=N
17 C
18         READ (IELMNT) NUMEST, NPAR, (AA(I), I=1, NUMEST)
19 C
20         CALL ELEMNT
21     END DO
22 C
23     RETURN
24     END

```

ASSEM90. FOR

```

1  ! . . . . .
2  ! .
3  ! .           S T A P 9 0
4  ! .
5  ! .   AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90
6  ! .   Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose
7  ! .
8  ! . . . . .
9  C
10 SUBROUTINE COLHT (MHT,ND,LM)
11 C . . . . .
12 C .
13 C .   To calculate column heights
14 C .
15 C . . . . .
16 C
17   IMPLICIT NONE
18   INTEGER :: ND, LM(1),MHT(1)
19   INTEGER :: I, LS, II, ME
20 C
21   LS=100000
22   DO I=1,ND
23     IF (LM(I) .NE. 0) THEN
24       IF (LM(I)-LS .LT. 0) LS=LM(I)
25     END IF
26   END DO
27 C
28   DO I=1,ND
29     II=LM(I)
30     IF (II.NE. 0) THEN
31       ME=II - LS
32       IF (ME. GT. MHT(II)) MHT(II)=ME
33     END IF
34   END DO
35 C
36   RETURN
37   END
38
39
40 SUBROUTINE ADDRES (MAXA,MHT)
41 C . . . . .
42 C .
43 C .   To calculate addresses of diagonal elements in banded
44 c .   matrix whose column heights are known
45 C .
46 C .   MHT = Active column heights
47 C .   MAXA = Addresses of diagonal elements
48 C .
49 C . . . . .
50 C
51   USE GLOBALS, ONLY : NEQ, MK, NWK
52 C
53   IMPLICIT NONE
54   INTEGER :: MAXA(*),MHT(*)
55   INTEGER :: NN, I
56 C
57 C   Clear array maxa
58 C
59   NN=NEQ + 1
60   DO I=1,NN
61     MAXA(I)=0.0
62   END DO
63 C
64   MAXA(1)=1
65   MAXA(2)=2
66   MK=0
67   IF (NEQ.GT.1) THEN
68     DO I=2,NEQ
69       IF (MHT(I).GT.MK) MK=MHT(I)
70       MAXA(I+1)=MAXA(I) + MHT(I) + 1
71     END DO
72   END IF
73   MK=MK + 1
74   NWK=MAXA(NEQ+1) - MAXA(1)

```


ASSEM90. FOR

```

1  C
2      RETURN
3      END
4
5
6      SUBROUTINE ASSEM (AA)
7  C . . . . .
8  C .
9  C . To call element subroutines for assemblage of the
10 C . structure stiffness matrix
11 C .
12 C . . . . .
13 C
14     USE GLOBALS, ONLY : IELMNT, NUMEG, NUMEST, NPAR
15 C
16     IMPLICIT NONE
17     REAL :: AA(1)
18     INTEGER :: N, I
19 C
20     REWIND IELMNT
21     DO N=1, NUMEG
22         READ (IELMNT) NUMEST, NPAR, (AA(I), I=1, NUMEST)
23         CALL ELEMNT
24     END DO
25 C
26     RETURN
27     END
28
29
30     SUBROUTINE ADDBAN (A, MAXA, S, LM, ND)
31 C . . . . .
32 C .
33 C . To assemble upper triangular element stiffness into
34 C . compacted global stiffness
35 C .
36 C .     A = GLOBAL STIFFNESS
37 C .     S = ELEMENT STIFFNESS
38 C .     ND = DEGREES OF FREEDOM IN ELEMENT STIFFNESS
39 C .
40 C .           S(1)      S(2)      S(3)      . . .
41 C .     S   =      S(ND+1)  S(ND+2)  S(ND+3)  . . .
42 C .           S(2*ND)  . . .
43 C .
44 C .
45 C .
46 C .           A(1)      A(3)      A(6)      . . .
47 C .     A   =      A(2)      A(5)      A(4)      . . .
48 C .           A(4)      . . .
49 C .
50 C .
51 C .
52 C . . . . .
53     IMPLICIT NONE
54     REAL(8) :: A(1), S(1)
55     INTEGER :: MAXA(1), LM(1)
56     INTEGER :: NDI, I, ND, II, MI, KS, J, JJ, IJ, KK, KSS
57 C
58     NDI=0
59     DO I=1, ND
60         II=LM(I)
61         IF (II .GT. 0) THEN
62             MI=MAXA(II)
63             KS=I
64             DO J=1, ND
65                 JJ=LM(J)
66                 IF (JJ .GT. 0) THEN
67                     IJ=II - JJ
68                     IF (IJ .GE. 0) THEN
69                         KK=MI + IJ
70                         KSS=KS
71                         IF (J .GE. I) KSS=J + NDI
72                         A(KK)=A(KK) + S(KSS)
73                     END IF
74                 END IF

```

ASSEM90. FOR

```

1         KS=KS + ND - J
2         END DO
3         END IF
4         NDI=NDI + ND - I
5     END DO
6 C
7     RETURN
8     END
9
10
11     SUBROUTINE COLSOL (A, V, MAXA, NN, NWK, NNM, KKK)
12 C . . . . .
13 C .
14 C .   To solve finite element static equilibrium equations in
15 C .   core, using compacted storage and column reduction scheme
16 C .
17 C .   -- Input variables --
18 C .       A (NWK)   = Stiffness matrix stored in compacted form
19 C .       V (NN)    = Right-hand-side load vector
20 C .       MAXA (NNM) = Vector containing addresses of diagonal
21 C .                   elements of stiffness matrix in a
22 C .                   NN      = Number of equations
23 C .                   NWK     = Number of elements below skyline of matrix
24 C .                   NNM    = NN + 1
25 C .                   KKK    = Input flag
26 C .       EQ. 1     Triangularization of stiffness matrix
27 C .       EQ. 2     Reduction and back-substitution of load vector
28 C .       IOUT     = UNIT used for output
29 C .
30 C .   -- OUTPUT --
31 C .       A (NWK)   = D and L - Factors of stiffness matrix
32 C .       V (NN)    = Displacement vector
33 C .
34 C . . . . .
35 C
36     USE GLOBALS, ONLY : IOUT
37 C
38     IMPLICIT NONE
39     INTEGER :: MAXA(1), NN, NWK, NNM, KKK
40     REAL(8) :: A(NWK), V(1), C, B
41     INTEGER :: N, K, KN, KL, KU, KH, IC, KLT, KI, J, ND, KK, L
42     INTEGER :: MINO
43 C
44 C     Perform L*D*L(T) factorization of stiffness matrix
45 C
46     IF (KKK-2) 40, 150, 150
47
48     40 DO N=1, NN
49         KN=MAXA(N)
50         KL=KN + 1
51         KU=MAXA(N+1) - 1
52         KH=KU - KL
53
54         IF (KH) 110, 90, 50
55
56     50     K=N - KH
57         IC=0
58         KLT=KU
59         DO J=1, KH
60             IC=IC + 1
61             KLT=KLT - 1
62             KI=MAXA(K)
63             ND=MAXA(K+1) - KI - 1
64             IF (ND .GT. 0) THEN
65                 KK=MINO(IC, ND)
66                 C=0.
67                 DO L=1, KK
68                     C=C + A(KI+L)*A(KLT+L)
69                 END DO
70                 A(KLT)=A(KLT) - C
71             END IF
72             K=K + 1
73         END DO
74

```

ASSEM90. FOR

```

1      90      K=N
2          B=0.
3          DO KK=KL, KU
4              K=K - 1
5              KI=MAXA(K)
6              C=A(KK)/A(KI)
7              B=B + C*A(KK)
8              A(KK)=C
9          END DO
10         A(KN)=A(KN) - B
11
12      110     IF (A(KN) .LE. 0) THEN
13          WRITE (IOUT, 2000) N, A(KN)
14          STOP
15      END IF
16  END DO
17  RETURN
18  C
19  C      REDUCE RIGHT-HAND-SIDE LOAD VECTOR
20  C
21  150 DO N=1, NN
22      KL=MAXA(N) + 1
23      KU=MAXA(N+1) - 1
24      IF (KU-KL .GE. 0) THEN
25          K=N
26          C=0.
27          DO KK=KL, KU
28              K=K - 1
29              C=C + A(KK)*V(K)
30          END DO
31          V(N)=V(N) - C
32      END IF
33  END DO
34  C
35  C      BACK-SUBSTITUTE
36  C
37      DO N=1, NN
38          K=MAXA(N)
39          V(N)=V(N)/A(K)
40      END DO
41
42      IF (NN.EQ. 1) RETURN
43
44      N=NN
45      DO L=2, NN
46          KL=MAXA(N) + 1
47          KU=MAXA(N+1) - 1
48          IF (KU-KL .GE. 0) THEN
49              K=N
50              DO KK=KL, KU
51                  K=K - 1
52                  V(K)=V(K) - A(KK)*V(N)
53              END DO
54          END IF
55          N=N - 1
56      END DO
57  C
58  RETURN
59  C
60  2000 FORMAT (//' STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE' , //,
61      1          ' , NONPOSITIVE PIVOT FOR EQUATION ' , I8, //,
62      2          ' , PIVOT = ' , E20. 12 )
63  C
64  END

```

INDEX

函数索引

ADDBAN, 17	INPUT, 8
ADDRES, 16	LOADS, 9
ASSEM, 17	LOADV, 10
CLEAR, 5	OPENFILES, 6
CLOSEFILES, 7	RUSS, 12
COLHT, 16	SECOND, 5
COLSOL, 18	STAP90, 2
ELCAL, 11	STRESS, 14
ELEMNT, 11	TRUSS, 12
ERROR, 5	WRITED, 6
GLOBALS, 1	