! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . S T A P 9 0 .

! . .

! . AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90 .

! . Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . Define global variables

module GLOBALS

integer, parameter :: MTOT = 10000 ! Speed storage available for execution

integer, parameter :: ITWO = 2 ! Double precision indicator

! 1 - Single precision arithmetic

! 2 - Double precision arithmetic

integer, parameter :: IELMNT=1 ! Unit storing element data

integer, parameter :: ILOAD=2 ! Unit storing load vectors

integer, parameter :: IIN=5 ! Unit used for input

integer, parameter :: IOUT=6 ! Unit used for output

integer :: NUMNP ! Total number of nodal points

! = 0 : Program stop

integer :: NEQ ! Number of equations

integer :: NWK ! Number of matrix elements

integer :: MK ! Maximum half bandwidth

integer :: IND ! Solution phase indicator

! 1 - Read and generate element information

! 2 - Assemble structure stiffness matrix

! 3 - Stress calculations

integer :: NPAR(10) ! Element group control data

! NPAR(1) - Element type

! 1 : Truss element

! NPAR(2) - Number of elements

! NPAR(3) - Number of different sets of material and

! cross-sectional constants

integer :: NUMEG ! Total number of element groups, > 0

integer :: MODEX ! Solution mode: 0 - data check only; 1 - execution

real :: TIM(5) ! Timing information

character\*80 :: HED ! Master heading information for use in labeling the output

real :: A(MTOT)

integer :: NFIRST

integer :: NLAST

integer :: NUMEST

integer :: MIDEST

integer :: MAXEST

integer :: NG

! Base addresses of arrays/matrices in array A/IA(MTOT)

integer :: N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15

end module GLOBALS

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . S T A P 9 0 .

! . .

! . AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90 .

! . Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

PROGRAM STAP90

C

USE GLOBALS

C

IMPLICIT NONE

INTEGER :: NLCASE, NEQ1, NLOAD, MM, NNL, KTR

INTEGER :: L, LL, I

REAL :: TT

C

CALL OPENFILES()

C

NUMEST=0

MAXEST=0

C

C \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

C \* INPUT PHASE \*

C \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

C

CALL SECOND (TIM(1))

C

C Read control information

C

C HED - The master heading informaiton for use in labeling the output

C NUMNP - Total number of nodal points

C 0 : program stop

C NUMEG - Total number of element group (>0)

C NLCASE - Number of load case (>0)

C MODEX - Solution mode

C 0 : data check only;

C 1 : execution

READ (IIN,1000) HED,NUMNP,NUMEG,NLCASE,MODEX

IF (NUMNP.EQ.0) STOP

WRITE (IOUT,2000) HED,NUMNP,NUMEG,NLCASE,MODEX

C

C Read nodal point data

C

C ALLOCATE STORAGE

C N1 - ID(3,NUMNP) : Boundary condition codes (0=free,1=deleted)

C N2 - X(NUMNP) : X coordinates

C N3 - Y(NUMNP) : Y coordinates

C N4 - Z(NUMNP) : Z coordinates

N1= 1

N2=N1 + 3\*NUMNP

N2=(N2/2)\*2 + 1

N3=N2 + NUMNP\*ITWO

N4=N3 + NUMNP\*ITWO

N5=N4 + NUMNP\*ITWO

IF (N5.GT.MTOT) CALL ERROR (N5-MTOT,1)

C

CALL INPUT (A(N1),A(N2),A(N3),A(N4),NUMNP,NEQ)

C

NEQ1=NEQ + 1

C

C Calculate and store load vectors

C

C N5 - R(NUMNP) : Load vector

N6=N5 + NEQ\*ITWO

WRITE (IOUT,2005)

C

REWIND ILOAD

C

DO L=1,NLCASE

C

C LL - Load case number

C NLOAD - The number of concentrated loads applied in this load case

READ (IIN,1010) LL,NLOAD

C

WRITE (IOUT,2010) LL,NLOAD

C

IF (LL.NE.L) THEN

WRITE (IOUT,2020)

STOP

ENDIF

C

C Allocate storage

C N6 - NOD(NLOAD) : Node number to which this load is applied (1~NUMNP)

C N7 - IDIRN(NLOAD) : Degree of freedom number for this load component

C 1 : X-direction; 2 : Y-direction; 3 : Z-direction

C N8 - FLOAD(NLOAD) : Magnitude of load

N7=N6 + NLOAD

N8=N7 + NLOAD

N9=N8 + NLOAD\*ITWO

C

IF (N9.GT.MTOT) CALL ERROR (N9-MTOT,2)

C

CALL LOADS (A(N5),A(N6),A(N7),A(N8),A(N1),NLOAD,NEQ)

C

END DO

C

C Read, generate and store element data

C

C Clear storage

C N5 - MHT(NUMNP) - Vector of column heights

C N6 - (2\*NUMMAT\*ITWO+7\*NUME+6\*NUME\*ITWO) : Element group data

N6=N5 + NEQ

N6=(N6/2)\*2 + 1

DO I=N5,N6

A(I)=0

END DO

C

IND=1

C

CALL ELCAL

C

CALL SECOND (TIM(2))

C

C \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

C \* SOLUTION PHASE \*

C \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

C

C Assemble stiffness matrix

C

CALL ADDRES (A(N2),A(N5))

C

C N2 - MAXA(NEQ+1)

C N3 - A(NWK) - Global structure stiffness matrix K

C N4 - R(NUMNP) - Load vector R and then displacement solution U

C N5 - (2\*NUMMAT\*ITWO+7\*NUME+6\*NUME\*ITWO) : Element group data

MM=NWK/NEQ

N3=N2 + NEQ + 1

N3=(N3/2)\*2 + 1

N4=N3 + NWK\*ITWO

N5=N4 + NEQ\*ITWO

N6=N5 + MAXEST

IF (N6.GT.MTOT) CALL ERROR (N6-MTOT,4)

C

C Write total system data

C

WRITE (IOUT,2025) NEQ,NWK,MK,MM

C

C In data check only mode we skip all further calculations

C

IF (MODEX.LE.0) THEN

CALL SECOND (TIM(3))

CALL SECOND (TIM(4))

CALL SECOND (TIM(5))

ELSE

C

C Clear storage

C

NNL=NWK + NEQ

CALL CLEAR (A(N3),NNL)

C

IND=2

C

CALL ASSEM (A(N5))

C

CALL SECOND (TIM(3))

C

C Triangularize stiffness matrix

C

KTR=1

CALL COLSOL (A(N3),A(N4),A(N2),NEQ,NWK,NEQ1,KTR)

C

CALL SECOND (TIM(4))

C

KTR=2

IND=3

C

REWIND ILOAD

DO L=1,NLCASE

C

CALL LOADV (A(N4),NEQ)

C

C Calculation of displacements

C

CALL COLSOL (A(N3),A(N4),A(N2),NEQ,NWK,NEQ1,KTR)

C

WRITE (IOUT,2015) L

CALL WRITED (A(N4),A(N1),NEQ,NUMNP)

C

C Calculation of stresses

C

CALL STRESS (A(N5))

C

END DO

C

CALL SECOND (TIM(5))

END IF

C

C Print solution times

C

TT=0.

DO I=1,4

TIM(I)=TIM(I+1) - TIM(I)

TT=TT + TIM(I)

END DO

WRITE (IOUT,2030) HED,(TIM(I),I=1,4),TT

C

STOP

C

1000 FORMAT (A80,/,4I5)

1010 FORMAT (2I5)

C

2000 FORMAT (///,' ',A80,///,

1 ' C O N T R O L I N F O R M A T I O N',//,

2 ' NUMBER OF NODAL POINTS',10(' .'),' (NUMNP) = ',I5,//,

3 ' NUMBER OF ELEMENT GROUPS',9(' .'),' (NUMEG) = ',I5,//,

4 ' NUMBER OF LOAD CASES',11(' .'),' (NLCASE) = ',I5,//,

5 ' SOLUTION MODE ',14(' .'),' (MODEX) = ',I5,/,

6 ' EQ.0, DATA CHECK',/,

7 ' EQ.1, EXECUTION')

2005 FORMAT (//,' L O A D C A S E D A T A')

2010 FORMAT (////,' LOAD CASE NUMBER',7(' .'),' = ',I5,//,

1 ' NUMBER OF CONCENTRATED LOADS . = ',I5)

2015 FORMAT (//,' LOAD CASE ',I3)

2020 FORMAT (' \*\*\* ERROR \*\*\* LOAD CASES ARE NOT IN ORDER')

2025 FORMAT (//,' TOTAL SYSTEM DATA',///,

1 ' NUMBER OF EQUATIONS',14(' .'),'(NEQ) = ',I5,//,

2 ' NUMBER OF MATRIX ELEMENTS',11(' .'),'(NWK) = ',I5,//,

3 ' MAXIMUM HALF BANDWIDTH ',12(' .'),'(MK ) = ',I5,//,

4 ' MEAN HALF BANDWIDTH',14(' .'),'(MM ) = ',I5)

2030 FORMAT (//,' S O L U T I O N T I M E L O G I N S E C',//,

1 ' FOR PROBLEM',//,' ',A80,///,

2 ' TIME FOR INPUT PHASE ',14(' .'),' =',F12.2,//,

3 ' TIME FOR CALCULATION OF STIFFNESS MATRIX . . . . =',F12.2,

4 //,

5 ' TIME FOR FACTORIZATION OF STIFFNESS MATRIX . . . =',F12.2,

6 //,

7 ' TIME FOR LOAD CASE SOLUTIONS ',10(' .'),' =',F12.2,///,

8 ' T O T A L S O L U T I O N T I M E . . . . . =',F12.2)

C

END

SUBROUTINE ERROR (N,I)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . P R O G R A M .

C . TO PRINT MESSAGES WHEN HIGH-SPEED STORAGE IS EXCEEDED .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

USE GLOBALS, ONLY : IOUT

C

IMPLICIT NONE

INTEGER :: N, I

C

IF (I == 1) THEN

WRITE (IOUT,2000)

ELSE IF (I == 2) THEN

WRITE (IOUT,2010)

ELSE IF (I == 3) THEN

WRITE (IOUT,2020)

ELSE IF (I == 4) THEN

WRITE (IOUT,2030)

END IF

C

WRITE (IOUT,2050) N

STOP

C

2000 FORMAT (//,' NOT ENOUGH STORAGE FOR ID ARRAY AND NODAL POINT ',

1 'COORDINATES')

2010 FORMAT (//,' NOT ENOUGH STORAGE FOR DEFINITION OF LOAD VECTORS')

2020 FORMAT (//,' NOT ENOUGH STORAGE FOR ELEMENT DATA INPUT')

2030 FORMAT (//,' NOT ENOUGH STORAGE FOR ASSEMBLAGE OF GLOBAL ',

1'STRUCTURE STIFFNESS, AND DISPLACEMENT AND STRESS SOLUTION PHASE')

2050 FORMAT (//,' \*\*\* ERROR \*\*\* STORAGE EXCEEDED BY ', I9)

C

END

C

SUBROUTINE SECOND (TIM)

USE DFPORT

IMPLICIT NONE

REAL :: TIM

C

C This is a Fortran 95 intrinsic subroutine

C Returns the processor time in seconds

C

CALL CPU\_TIME(TIM)

C

RETURN

END

SUBROUTINE CLEAR (A,N)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To clear double precision array A .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

IMPLICIT NONE

C

INTEGER :: N, I

REAL(8) :: A(N)

C

DO I=1,N

A(I)=0.

END DO

RETURN

END

SUBROUTINE WRITED (DISP,ID,NEQ,NUMNP)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To print displacements .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS, ONLY : IOUT

C

IMPLICIT NONE

INTEGER :: NEQ,NUMNP,ID(3,NUMNP)

REAL(8) :: DISP(NEQ),D(3)

INTEGER :: IC,II,I,KK,IL

C

C Print displacements

C

WRITE (IOUT,2000)

IC=4

C

DO II=1,NUMNP

IC=IC + 1

IF (IC.GE.56) THEN

WRITE (IOUT,2000)

IC=4

END IF

C

DO I=1,3

D(I)=0.

END DO

C

DO I=1,3

KK=ID(I,II)

IL=I

IF (KK.NE.0) D(IL)=DISP(KK)

END DO

C

WRITE (IOUT,2010) II,D

END DO

C

RETURN

C

2000 FORMAT (///,' D I S P L A C E M E N T S',//,' NODE ',10X,

1 'X-DISPLACEMENT Y-DISPLACEMENT Z-DISPLACEMENT')

2010 FORMAT (1X,I3,8X,3E18.6)

C

END

SUBROUTINE OPENFILES()

USE GLOBALS

IMPLICIT NONE

LOGICAL :: EX

!

INQUIRE(FILE = "STAP90.IN", EXIST = EX)

IF (.NOT. EX) THEN

PRINT \*, "\*\*\* STOP \*\*\* FILE STAP90.IN DOES NOT EXIST !"

STOP

END IF

OPEN(IIN , FILE = "STAP90.IN", STATUS = "OLD")

OPEN(IOUT , FILE = "STAP90.OUT", STATUS = "REPLACE")

OPEN(IELMNT, FILE = "ELMNT.TMP", FORM = "UNFORMATTED",

1 STATUS = "SCRATCH")

OPEN(ILOAD , FILE = "LOAD.TMP", FORM = "UNFORMATTED",

1 STATUS = "SCRATCH")

END SUBROUTINE OPENFILES

SUBROUTINE CLOSEFILES()

USE GLOBALS

IMPLICIT NONE

CLOSE(IIN)

CLOSE(IOUT)

CLOSE(IELMNT)

CLOSE(ILOAD)

END SUBROUTINE CLOSEFILES

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . S T A P 9 0 .

! . .

! . AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90 .

! . Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

!

SUBROUTINE INPUT (ID,X,Y,Z,NUMNP,NEQ)

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . To read, generate, and print nodal point input data .

! . To calculate equation numbers and store them in id arrray .

! . .

! . N = Element number .

! . ID = Boundary condition codes (0=free,1=deleted) .

! . X,Y,Z = Coordinates .

! . KN = Generation code .

! . i.e. increment on nodal point number .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

!

USE GLOBALS, ONLY : IIN, IOUT

!

IMPLICIT NONE

INTEGER :: NUMNP,NEQ,ID(3,NUMNP)

REAL(8) :: X(NUMNP),Y(NUMNP),Z(NUMNP)

REAL(8) :: XNUM, DX, DY, DZ

INTEGER :: KNOLD, NOLD, NUM, NUMN

INTEGER :: I, J, K, KN, N, KK

!

! Read and generate nodal point data

!

WRITE (IOUT,2000)

WRITE (IOUT,2010)

WRITE (IOUT,2020)

!

KNOLD=0

NOLD=0

!

N = 0

DO WHILE (N.NE.NUMNP)

READ (IIN,1000) N,(ID(I,N),I=1,3),X(N),Y(N),Z(N),KN

WRITE (IOUT,2030) N,(ID(I,N),I=1,3),X(N),Y(N),Z(N),KN

IF (KNOLD.NE.0) THEN

NUM=(N-NOLD)/KNOLD

NUMN=NUM - 1

IF (NUMN.GE.1) THEN

XNUM=NUM

DX=(X(N)-X(NOLD))/XNUM

DY=(Y(N)-Y(NOLD))/XNUM

DZ=(Z(N)-Z(NOLD))/XNUM

K=NOLD

DO J=1,NUMN

KK=K

K=K + KNOLD

X(K)=X(KK) + DX

Y(K)=Y(KK) + DY

Z(K)=Z(KK) + DZ

DO I=1,3

ID(I,K)=ID(I,KK)

END DO

END DO

END IF

ENDIF

!

50 NOLD=N

KNOLD=KN

END DO

!

! Write complete nodal data

!

WRITE (IOUT,2015)

WRITE (IOUT,2020)

DO N=1,NUMNP

WRITE (IOUT,2030) N,(ID(I,N),I=1,3),X(N),Y(N),Z(N),KN

END DO

!

! Number unknowns

!

NEQ=0

DO N=1,NUMNP

DO I=1,3

IF (ID(I,N) .EQ. 0) THEN

NEQ=NEQ + 1

ID(I,N)=NEQ

ELSE

ID(I,N)=0

END IF

END DO

END DO

!

! Write equation numbers

!

WRITE (IOUT,2040) (N,(ID(I,N),I=1,3),N=1,NUMNP)

!

RETURN

!

1000 FORMAT (4I5,3F10.0,I5)

2000 FORMAT(//,' N O D A L P O I N T D A T A',/)

2010 FORMAT(' INPUT NODAL DATA',//)

2015 FORMAT(//,' GENERATED NODAL DATA',//)

2020 FORMAT(' NODE',10X,'BOUNDARY',25X,'NODAL POINT',17X,'MESH',/,

1' NUMBER CONDITION CODES',21X,'COORDINATES',14X,'GENERATING',

2/,77X,'CODE',/,

315X,'X Y Z',15X,'X',12X,'Y',12X,'Z',10X,'KN')

2030 FORMAT (I5,6X,3I5,6X,3F13.3,3X,I6)

2040 FORMAT(//,' EQUATION NUMBERS',//,' NODE',9X,

1 'DEGREES OF FREEDOM',/,' NUMBER',//,

2 ' N',13X,'X Y Z',/,(1X,I5,9X,3I5))

!

END

SUBROUTINE LOADS (R,NOD,IDIRN,FLOAD,ID,NLOAD,NEQ)

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . To read nodal load data .

! . To calculate the load vector r for each load case and .

! . write onto unit ILOAD .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

USE GLOBALS, ONLY : IIN, IOUT, ILOAD, MODEX

!

IMPLICIT NONE

INTEGER :: NLOAD,NEQ,ID(3,1),NOD(1),IDIRN(1)

REAL(8) :: R(NEQ),FLOAD(1)

INTEGER :: I,L,LI,LN,II

!

WRITE (IOUT,2000)

READ (IIN,1000) (NOD(I),IDIRN(I),FLOAD(I),I=1,NLOAD)

WRITE (IOUT,2010) (NOD(I),IDIRN(I),FLOAD(I),I=1,NLOAD)

!

IF (MODEX.EQ.0) RETURN

!

DO I=1,NEQ

R(I)=0.

END DO

!

DO L=1,NLOAD

LN=NOD(L)

LI=IDIRN(L)

II=ID(LI,LN)

IF (II > 0) R(II)=R(II) + FLOAD(L)

END DO

!

WRITE (ILOAD) R

!

RETURN

!

1000 FORMAT (2I5,F10.0)

2000 FORMAT (//,' NODE DIRECTION LOAD',/,

1 ' NUMBER',19X,'MAGNITUDE')

2010 FORMAT (' ',I6,9X,I4,7X,E12.5)

!

END

SUBROUTINE LOADV (R,NEQ)

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . To obtain the load vector .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

!

USE GLOBALS, ONLY : ILOAD

!

IMPLICIT NONE

INTEGER :: NEQ

REAL(8) :: R(NEQ)

!

READ (ILOAD) R

!

RETURN

END

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . S T A P 9 0 .

! . .

! . AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90 .

! . Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

SUBROUTINE ELCAL

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To loop over all element groups for reading, .

C . generating and storing the element data .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

USE GLOBALS

C

IMPLICIT NONE

INTEGER :: N, I

C

REWIND IELMNT

WRITE (IOUT,2000)

C

C Loop over all element groups

C

DO N=1,NUMEG

IF (N.NE.1) WRITE (IOUT,2010)

C

READ (IIN,1000) NPAR

C

CALL ELEMNT

C

IF (MIDEST.GT.MAXEST) MAXEST=MIDEST

C

WRITE (IELMNT) MIDEST,NPAR,(A(I),I=NFIRST,NLAST)

C

END DO

C

RETURN

C

1000 FORMAT (10I5)

2000 FORMAT (//,' E L E M E N T G R O U P D A T A',//)

2010 FORMAT (' ')

C

END

SUBROUTINE ELEMNT

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To call the appropriate element subroutine .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS

C

IMPLICIT NONE

INTEGER :: NPAR1

C

NPAR1=NPAR(1)

C

IF (NPAR1 == 1) THEN

CALL TRUSS

ELSE

C

C Other element types would be called here, identifying each

C element type by a different NPAR(1) parameter

C

END IF

C

RETURN

END

SUBROUTINE TRUSS

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To set up storage and call the truss element subroutine .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS

C

IMPLICIT NONE

INTEGER :: NUME, NUMMAT, N101, N102, N103, N104, N105, N106

C

NUME = NPAR(2)

NUMMAT = NPAR(3)

C

C Allocate storage

C

NFIRST=N6

IF (IND.GT.1) NFIRST=N5

N101=NFIRST

N102=N101 + NUMMAT\*ITWO

N103=N102 + NUMMAT\*ITWO

N104=N103 + 6\*NUME

N105=N104 + 6\*NUME\*ITWO

N106=N105 + NUME

NLAST=N106

C

IF (IND.LE.1) THEN

IF (NLAST.GT.MTOT) CALL ERROR (NLAST-MTOT,3)

ELSE

IF (NLAST.GT.MTOT) CALL ERROR (NLAST-MTOT,4)

END IF

C

MIDEST=NLAST - NFIRST

C

CALL RUSS (A(N1),A(N2),A(N3),A(N4),A(N4),A(N5),A(N101),A(N102),

1 A(N103),A(N104),A(N105))

C

RETURN

C

END

SUBROUTINE RUSS (ID,X,Y,Z,U,MHT,E,AREA,LM,XYZ,MATP)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . TRUSS element subroutine .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS

C

IMPLICIT NONE

INTEGER :: ID(3,1),LM(6,1),MATP(1),MHT(1)

REAL(8) :: X(1),Y(1),Z(1),E(1),AREA(1),XYZ(6,1),U(1)

REAL(8) :: S(21),ST(6),D(3)

C

INTEGER :: NPAR1, NUME, NUMMAT, ND, I, J, K, L, N, M, II, JJ

INTEGER :: MTYP, MTYPE, KG, KL, KKK, IPRINT

REAL(8) :: XL2, XL, SQRT, XX, YY, STR, P

C

NPAR1 = NPAR(1)

NUME = NPAR(2)

NUMMAT = NPAR(3)

C

ND=6

C

GO TO (300,610,800),IND

C

C Read and generate element information

C

C Read material information

C

300 WRITE (IOUT,2000) NPAR1,NUME

IF (NUMMAT.EQ.0) NUMMAT=1

WRITE (IOUT,2010) NUMMAT

C

WRITE (IOUT,2020)

DO I=1,NUMMAT

READ (IIN,1000) N,E(N),AREA(N)

WRITE (IOUT,2030) N,E(N),AREA(N)

END DO

C

C Read element information

C

WRITE (IOUT,2040)

N=1

100 READ (IIN,1020) M,II,JJ,MTYP,KG

IF (KG.EQ.0) KG=1

DO WHILE (.TRUE.)

IF (M.EQ.N) THEN

I=II

J=JJ

MTYPE=MTYP

KKK=KG

END IF

C

C Save element information

C

XYZ(1,N)=X(I)

XYZ(2,N)=Y(I)

XYZ(3,N)=Z(I)

C

XYZ(4,N)=X(J)

XYZ(5,N)=Y(J)

XYZ(6,N)=Z(J)

C

MATP(N)=MTYPE

C

DO L=1,6

LM(L,N)=0

END DO

DO L=1,3

LM(L,N)=ID(L,I)

LM(L+3,N)=ID(L,J)

END DO

C

C Update column heights and bandwidth

C

CALL COLHT (MHT,ND,LM(1,N))

C

WRITE (IOUT,2050) N,I,J,MTYPE

IF (N.EQ.NUME) RETURN

C

N=N + 1

I=I + KKK

J=J + KKK

IF (N.GT.M) GO TO 100

END DO

C

C Assemble stucture stiffness matrix

C

610 DO N=1,NUME

MTYPE=MATP(N)

XL2=0.

DO L=1,3

D(L)=XYZ(L,N) - XYZ(L+3,N)

XL2=XL2 + D(L)\*D(L)

END DO

XL=SQRT(XL2)

XX=E(MTYPE)\*AREA(MTYPE)\*XL

DO L=1,3

ST(L)=D(L)/XL2

ST(L+3)=-ST(L)

END DO

C

KL=0

DO L=1,6

YY=ST(L)\*XX

DO K=L,6

KL=KL + 1

S(KL)=ST(K)\*YY

END DO

END DO

CALL ADDBAN (A(N3),A(N2),S,LM(1,N),ND)

END DO

RETURN

C

C Stress calculations

C

800 IPRINT=0

DO 830 N=1,NUME

IPRINT=IPRINT + 1

IF (IPRINT.GT.50) IPRINT=1

IF (IPRINT.EQ.1) WRITE (IOUT,2060) NG

MTYPE=MATP(N)

XL2=0.

DO L=1,3

D(L) = XYZ(L,N) - XYZ(L+3,N)

XL2=XL2 + D(L)\*D(L)

END DO

DO L=1,3

ST(L)=(D(L)/XL2)\*E(MTYPE)

ST(L+3)=-ST(L)

END DO

STR=0.0

DO L=1,3

I=LM(L,N)

IF (I.GT.0) STR=STR + ST(L)\*U(I)

J=LM(L+3,N)

IF (J.GT.0) STR=STR + ST(L+3)\*U(J)

END DO

P=STR\*AREA(MTYPE)

WRITE (IOUT,2070) N,P,STR

830 CONTINUE

C

1000 FORMAT (I5,2F10.0)

1010 FORMAT (2F10.0)

1020 FORMAT (5I5)

2000 FORMAT (' E L E M E N T D E F I N I T I O N',///,

1 ' ELEMENT TYPE ',13(' .'),'( NPAR(1) ) . . =',I5,/,

2 ' EQ.1, TRUSS ELEMENTS',/,

3 ' EQ.2, ELEMENTS CURRENTLY',/,

4 ' EQ.3, NOT AVAILABLE',//,

5 ' NUMBER OF ELEMENTS.',10(' .'),'( NPAR(2) ) . . =',I5,//)

2010 FORMAT (' M A T E R I A L D E F I N I T I O N',///,

1 ' NUMBER OF DIFFERENT SETS OF MATERIAL',/,

2 ' AND CROSS-SECTIONAL CONSTANTS ',

3 4(' .'),'( NPAR(3) ) . . =',I5,//)

2020 FORMAT (' SET YOUNG''S CROSS-SECTIONAL',/,

1 ' NUMBER MODULUS',10X,'AREA',/,

2 15X,'E',14X,'A')

2030 FORMAT (/,I5,4X,E12.5,2X,E14.6)

2040 FORMAT (//,' E L E M E N T I N F O R M A T I O N',///,

1 ' ELEMENT NODE NODE MATERIAL',/,

2 ' NUMBER-N I J SET NUMBER',/)

2050 FORMAT (I5,6X,I5,4X,I5,7X,I5)

2060 FORMAT (///,' S T R E S S C A L C U L A T I O N S F O R ',

1 'E L E M E N T G R O U P',I4,//,

2 ' ELEMENT',13X,'FORCE',12X,'STRESS',/,

3 ' NUMBER',/)

2070 FORMAT (1X,I5,11X,E13.6,4X,E13.6)

C

END

SUBROUTINE STRESS (AA)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To call the element subroutine for the calculation of stresses .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS, ONLY : IELMNT, NG, NUMEST, NPAR, NUMEG

C

IMPLICIT NONE

REAL :: AA(1)

INTEGER N, I

C

C Loop over all element groups

C

REWIND IELMNT

C

DO N=1,NUMEG

NG=N

C

READ (IELMNT) NUMEST,NPAR,(AA(I),I=1,NUMEST)

C

CALL ELEMNT

END DO

C

RETURN

END

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

! . .

! . S T A P 9 0 .

! . .

! . AN IN-CORE SOLUTION STATIC ANALYSIS PROGRAM IN FORTRAN 90 .

! . Adapted from STAP (KJ Bath, FORTRAN IV) for teaching purpose .

! . .

! . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

SUBROUTINE COLHT (MHT,ND,LM)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To calculate column heights .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

IMPLICIT NONE

INTEGER :: ND, LM(1),MHT(1)

INTEGER :: I, LS, II, ME

C

LS=100000

DO I=1,ND

IF (LM(I) .NE. 0) THEN

IF (LM(I)-LS .LT. 0) LS=LM(I)

END IF

END DO

C

DO I=1,ND

II=LM(I)

IF (II.NE.0) THEN

ME=II - LS

IF (ME.GT.MHT(II)) MHT(II)=ME

END IF

END DO

C

RETURN

END

SUBROUTINE ADDRES (MAXA,MHT)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To calculate addresses of diagonal elements in banded .

c . matrix whose column heights are known .

C . .

C . MHT = Active column heights .

C. MAXA = Addresses of diagonal elements .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS, ONLY : NEQ, MK, NWK

C

IMPLICIT NONE

INTEGER :: MAXA(\*),MHT(\*)

INTEGER :: NN, I

C

C Clear array maxa

C

NN=NEQ + 1

DO I=1,NN

MAXA(I)=0.0

END DO

C

MAXA(1)=1

MAXA(2)=2

MK=0

IF (NEQ.GT.1) THEN

DO I=2,NEQ

IF (MHT(I).GT.MK) MK=MHT(I)

MAXA(I+1)=MAXA(I) + MHT(I) + 1

END DO

END IF

MK=MK + 1

NWK=MAXA(NEQ+1) - MAXA(1)

C

RETURN

END

SUBROUTINE ASSEM (AA)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To call element subroutines for assemblage of the .

C . structure stiffness matrix .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS, ONLY : IELMNT, NUMEG, NUMEST, NPAR

C

IMPLICIT NONE

REAL :: AA(1)

INTEGER :: N, I

C

REWIND IELMNT

DO N=1,NUMEG

READ (IELMNT) NUMEST,NPAR,(AA(I),I=1,NUMEST)

CALL ELEMNT

END DO

C

RETURN

END

SUBROUTINE ADDBAN (A,MAXA,S,LM,ND)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To assemble upper triangular element stiffness into .

C . compacted global stiffness .

C . .

C . A = GLOBAL STIFFNESS .

C . S = ELEMENT STIFFNESS .

C . ND = DEGREES OF FREEDOM IN ELEMENT STIFFNESS .

C . .

C . S(1) S(2) S(3) . . . .

C . S = S(ND+1) S(ND+2) . . . .

C . S(2\*ND) . . . .

C . . . . .

C . .

C . .

C . A(1) A(3) A(6) . . . .

C . A = A(2) A(5) . . . .

C . A(4) . . . .

C . . . . .

C . .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

IMPLICIT NONE

REAL(8) :: A(1),S(1)

INTEGER :: MAXA(1),LM(1)

INTEGER :: NDI, I, ND, II, MI, KS, J, JJ, IJ, KK, KSS

C

NDI=0

DO I=1,ND

II=LM(I)

IF (II .GT. 0) THEN

MI=MAXA(II)

KS=I

DO J=1,ND

JJ=LM(J)

IF (JJ .GT. 0) THEN

IJ=II - JJ

IF (IJ .GE. 0) THEN

KK=MI + IJ

KSS=KS

IF (J.GE.I) KSS=J + NDI

A(KK)=A(KK) + S(KSS)

END IF

END IF

KS=KS + ND - J

END DO

END IF

NDI=NDI + ND - I

END DO

C

RETURN

END

SUBROUTINE COLSOL (A,V,MAXA,NN,NWK,NNM,KKK)

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C . .

C . To solve finite element static equilibrium equations in .

C . core, using compacted storage and column reduction scheme .

C . .

C . - - Input variables - - .

C . A(NWK) = Stiffness matrix stored in compacted form .

C . V(NN) = Right-hand-side load vector .

C . MAXA(NNM) = Vector containing addresses of diagonal .

C . elements of stiffness matrix in a .

C . NN = Number of equations .

C . NWK = Number of elements below skyline of matrix .

C . NNM = NN + 1 .

C . KKK = Input flag .

C . EQ. 1 Triangularization of stiffness matrix .

C . EQ. 2 Reduction and back-substitution of load vector .

C . IOUT = UNIT used for output .

C . .

C . - - OUTPUT - - .

C . A(NWK) = D and L - Factors of stiffness matrix .

C . V(NN) = Displacement vector .

C . .

C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

C

USE GLOBALS, ONLY : IOUT

C

IMPLICIT NONE

INTEGER :: MAXA(1),NN,NWK,NNM,KKK

REAL(8) :: A(NWK),V(1),C,B

INTEGER :: N,K,KN,KL,KU,KH,IC,KLT,KI,J,ND,KK,L

INTEGER :: MIN0

C

C Perform L\*D\*L(T) factorization of stiffness matrix

C

IF (KKK-2) 40,150,150

40 DO N=1,NN

KN=MAXA(N)

KL=KN + 1

KU=MAXA(N+1) - 1

KH=KU - KL

IF (KH) 110,90,50

50 K=N - KH

IC=0

KLT=KU

DO J=1,KH

IC=IC + 1

KLT=KLT - 1

KI=MAXA(K)

ND=MAXA(K+1) - KI - 1

IF (ND .GT. 0) THEN

KK=MIN0(IC,ND)

C=0.

DO L=1,KK

C=C + A(KI+L)\*A(KLT+L)

END DO

A(KLT)=A(KLT) - C

END IF

K=K + 1

END DO

90 K=N

B=0.

DO KK=KL,KU

K=K - 1

KI=MAXA(K)

C=A(KK)/A(KI)

B=B + C\*A(KK)

A(KK)=C

END DO

A(KN)=A(KN) - B

110 IF (A(KN) .LE. 0) THEN

WRITE (IOUT,2000) N,A(KN)

STOP

END IF

END DO

RETURN

C

C REDUCE RIGHT-HAND-SIDE LOAD VECTOR

C

150 DO N=1,NN

KL=MAXA(N) + 1

KU=MAXA(N+1) - 1

IF (KU-KL .GE. 0) THEN

K=N

C=0.

DO KK=KL,KU

K=K - 1

C=C + A(KK)\*V(K)

END DO

V(N)=V(N) - C

END IF

END DO

C

C BACK-SUBSTITUTE

C

DO N=1,NN

K=MAXA(N)

V(N)=V(N)/A(K)

END DO

IF (NN.EQ.1) RETURN

N=NN

DO L=2,NN

KL=MAXA(N) + 1

KU=MAXA(N+1) - 1

IF (KU-KL .GE. 0) THEN

K=N

DO KK=KL,KU

K=K - 1

V(K)=V(K) - A(KK)\*V(N)

END DO

END IF

N=N - 1

END DO

C

RETURN

C

2000 FORMAT (//' STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE',//,

1 ' NONPOSITIVE PIVOT FOR EQUATION ',I8,//,

2 ' PIVOT = ',E20.12 )

C

END

函数索引

ADDBAN, 17

ADDRES, 16

ASSEM, 17

CLEAR, 5

CLOSEFILES, 7

COLHT, 16

COLSOL, 18

ELCAL, 11

ELEMNT, 11

ERROR, 5

GLOBALS, 1

INPUT, 8

LOADS, 9

LOADV, 10

OPENFILES, 6

RUSS, 12

SECOND, 5

STAP90, 2

STRESS, 14

TRUSS, 12

WRITED, 6