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

! . .

! . 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 .

! . .

! . Xiong Zhang, (2013) .

! . Computational Dynamics Group, School of Aerospace .

! . Tsinghua Univerity .

! . .

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

! . Define global variables

module GLOBALS

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

integer :: NFIRST

integer :: NLAST

integer :: MIDEST

integer :: MAXEST

integer :: NG

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 .

! . .

! . Xiong Zhang, (2013) .

! . Computational Dynamics Group, School of Aerospace .

! . Tsinghua Univerity .

! . .

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

PROGRAM STAP90

USE GLOBALS

USE MEMALLOCATE

IMPLICIT NONE

INTEGER :: NLCASE, NEQ1, NLOAD, MM

INTEGER :: L, LL, I

REAL :: TT

! OPEN INPUT DATA FILE, RESULTS OUTPUT FILE AND TEMPORARY FILES

CALL OPENFILES()

MAXEST=0

! \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

! \* INPUT PHASE \*

! \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

WRITE(\*,'("Input phase ... ")')

CALL SECOND (TIM(1))

! Read control information

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

! NUMNP - Total number of nodal points

! 0 : program stop

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

! NLCASE - Number of load case (>0)

! MODEX - Solution mode

! 0 : data check only;

! 1 : execution

READ (IIN,'(A80,/,4I5)') HED,NUMNP,NUMEG,NLCASE,MODEX

IF (NUMNP.EQ.0) STOP ! Data check mode

WRITE (IOUT,"(/,' ',A80,//, &

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

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

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

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

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

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

' EQ.1, EXECUTION')") HED,NUMNP,NUMEG,NLCASE,MODEX

! Read nodal point data

! ALLOCATE STORAGE

! ID(3,NUMNP) : Boundary condition codes (0=free,1=deleted)

! X(NUMNP) : X coordinates

! Y(NUMNP) : Y coordinates

! Z(NUMNP) : Z coordinates

CALL MEMALLOC(1,"ID ",3\*NUMNP,1)

CALL MEMALLOC(2,"X ",NUMNP,ITWO)

CALL MEMALLOC(3,"Y ",NUMNP,ITWO)

CALL MEMALLOC(4,"Z ",NUMNP,ITWO)

CALL INPUT (IA(NP(1)),DA(NP(2)),DA(NP(3)),DA(NP(4)),NUMNP,NEQ)

NEQ1=NEQ + 1

! Calculate and store load vectors

! R(NEQ) : Load vector

CALL MEMALLOC(5,"R ",NEQ,ITWO)

WRITE (IOUT,"(//,' L O A D C A S E D A T A')")

REWIND ILOAD

DO L=1,NLCASE

! LL - Load case number

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

READ (IIN,'(2I5)') LL,NLOAD

WRITE (IOUT,"(/,' LOAD CASE NUMBER',7(' .'),' = ',I5,/, &

' NUMBER OF CONCENTRATED LOADS . = ',I5)") LL,NLOAD

IF (LL.NE.L) THEN

WRITE (IOUT,"(' \*\*\* ERROR \*\*\* LOAD CASES ARE NOT IN ORDER')")

STOP

ENDIF

! Allocate storage

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

! IDIRN(NLOAD) : Degree of freedom number for this load component

! 1 : X-direction;

! 2 : Y-direction;

! 3 : Z-direction

! FLOAD(NLOAD) : Magnitude of load

CALL MEMALLOC(6,"NOD ",NLOAD,1)

CALL MEMALLOC(7,"IDIRN",NLOAD,1)

CALL MEMALLOC(8,"FLOAD",NLOAD,ITWO)

CALL LOADS (DA(NP(5)),IA(NP(6)),IA(NP(7)),DA(NP(8)),IA(NP(1)),NLOAD,NEQ)

END DO

! Read, generate and store element data

! Clear storage

! MHT(NEQ) - Vector of column heights

CALL MEMFREEFROM(5)

CALL MEMALLOC(5,"MHT ",NEQ,1)

IND=1 ! Read and generate element information

CALL ELCAL

CALL SECOND (TIM(2))

! \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

! \* SOLUTION PHASE \*

! \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

WRITE(\*,'("Solution phase ... ")')

! Assemble stiffness matrix

! ALLOCATE STORAGE

! MAXA(NEQ+1)

CALL MEMFREEFROM(6)

CALL MEMFREEFROMTO(2,4)

CALL MEMALLOC(2,"MAXA ",NEQ+1,1)

CALL ADDRES (IA(NP(2)),IA(NP(5)))

! ALLOCATE STORAGE

! A(NWK) - Global structure stiffness matrix K

! R(NEQ) - Load vector R and then displacement solution U

MM=NWK/NEQ

CALL MEMALLOC(3,"STFF ",NWK,ITWO)

CALL MEMALLOC(4,"R ",NEQ,ITWO)

CALL MEMALLOC(11,"ELEGP",MAXEST,1)

! Write total system data

WRITE (IOUT,"(//,' TOTAL SYSTEM DATA',//, &

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

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

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

' MEAN HALF BANDWIDTH',14(' .'),'(MM ) = ',I5)") NEQ,NWK,MK,MM

! In data check only mode we skip all further calculations

IF (MODEX.LE.0) THEN

CALL SECOND (TIM(3))

CALL SECOND (TIM(4))

CALL SECOND (TIM(5))

ELSE

IND=2 ! Assemble structure stiffness matrix

CALL ASSEM (A(NP(11)))

CALL SECOND (TIM(3))

! Triangularize stiffness matrix

CALL COLSOL (DA(NP(3)),DA(NP(4)),IA(NP(2)),NEQ,NWK,NEQ1,1)

CALL SECOND (TIM(4))

IND=3 ! Stress calculations

REWIND ILOAD

DO L=1,NLCASE

CALL LOADV (DA(NP(4)),NEQ) ! Read in the load vector

! Solve the equilibrium equations to calculate the displacements

CALL COLSOL (DA(NP(3)),DA(NP(4)),IA(NP(2)),NEQ,NWK,NEQ1,2)

WRITE (IOUT,"(//,' LOAD CASE ',I3)") L

CALL WRITED (DA(NP(4)),IA(NP(1)),NEQ,NUMNP) ! Print displacements

! Calculation of stresses

CALL STRESS (A(NP(11)))

END DO

CALL SECOND (TIM(5))

END IF

! Print solution times

TT=0.

DO I=1,4

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

TT=TT + TIM(I)

END DO

WRITE (IOUT,"(//, &

' S O L U T I O N T I M E L O G I N S E C',//, &

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

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

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

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

' T O T A L S O L U T I O N T I M E . . . . . =',F12.2)") (TIM(I),I=1,4),TT

WRITE (\*,"(//, &

' S O L U T I O N T I M E L O G I N S E C',//, &

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

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

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

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

' T O T A L S O L U T I O N T I M E . . . . . =',F12.2)") (TIM(I),I=1,4),TT

STOP

END PROGRAM STAP90

SUBROUTINE SECOND (TIM)

! USE DFPORT ! Only for Compaq Fortran

IMPLICIT NONE

REAL :: TIM

! This is a Fortran 95 intrinsic subroutine

! Returns the processor time in seconds

CALL CPU\_TIME(TIM)

RETURN

END SUBROUTINE SECOND

SUBROUTINE WRITED (DISP,ID,NEQ,NUMNP)

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

! . .

! . To print displacements .

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

USE GLOBALS, ONLY : IOUT

IMPLICIT NONE

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

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

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

! Print displacements

WRITE (IOUT,"(//,' D I S P L A C E M E N T S',//,' NODE ',10X, &

'X-DISPLACEMENT Y-DISPLACEMENT Z-DISPLACEMENT')")

IC=4

DO II=1,NUMNP

IC=IC + 1

IF (IC.GE.56) THEN

WRITE (IOUT,"(//,' D I S P L A C E M E N T S',//,' NODE ',10X, &

'X-DISPLACEMENT Y-DISPLACEMENT Z-DISPLACEMENT')")

IC=4

END IF

DO I=1,3

D(I)=0.

END DO

DO I=1,3

KK=ID(I,II)

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

END DO

WRITE (IOUT,'(1X,I3,8X,3E18.6)') II,D

END DO

RETURN

END SUBROUTINE WRITED

SUBROUTINE OPENFILES()

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

! . .

! . Open input data file, results output file and temporary files .

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

USE GLOBALS

! use DFLIB ! for NARGS() ! Only for Compaq Fortran

IMPLICIT NONE

LOGICAL :: EX

CHARACTER\*80 FileInp

! Only for Compaq Fortran

! if(NARGS().ne.2) then

! stop 'Usage: mpm3d InputFileName'

! else

! call GETARG(1,FileInp)

! end if

if(COMMAND\_ARGUMENT\_COUNT().ne.1) then

stop 'Usage: STAP90 InputFileName'

else

call GET\_COMMAND\_ARGUMENT(1,FileInp)

end if

INQUIRE(FILE = FileInp, EXIST = EX)

IF (.NOT. EX) THEN

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

STOP

END IF

OPEN(IIN , FILE = FileInp, STATUS = "OLD")

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

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

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

END SUBROUTINE OPENFILES

SUBROUTINE CLOSEFILES()

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

! . .

! . Close all data files .

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

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 .

! . .

! . Xiong Zhang, (2013) .

! . Computational Dynamics Group, School of Aerospace .

! . Tsinghua Univerity .

! . .

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

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)

INTEGER :: I, J, N

! Read nodal point data

N = 0

DO WHILE (N.NE.NUMNP)

READ (IIN,"(4I5,3F10.0,I5)") N,(ID(I,N),I=1,3),X(N),Y(N),Z(N)

END DO

! Write complete nodal data

WRITE (IOUT,"(//,' N O D A L P O I N T D A T A',/)")

WRITE (IOUT,"(' NODE',10X,'BOUNDARY',25X,'NODAL POINT',/, &

' NUMBER CONDITION CODES',21X,'COORDINATES', /,15X, &

'X Y Z',15X,'X',12X,'Y',12X,'Z')")

DO N=1,NUMNP

WRITE (IOUT,"(I5,6X,3I5,6X,3F13.3)") N,(ID(I,N),I=1,3),X(N),Y(N),Z(N)

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,"(//,' EQUATION NUMBERS',//,' NODE',9X, &

'DEGREES OF FREEDOM',/,' NUMBER',/, &

' N',13X,'X Y Z',/,(1X,I5,9X,3I5))") (N,(ID(I,N),I=1,3),N=1,NUMNP)

RETURN

END SUBROUTINE INPUT

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,\*),NOD(NLOAD),IDIRN(NLOAD)

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

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

WRITE (IOUT,"(/,' NODE DIRECTION LOAD',/, ' NUMBER',19X,'MAGNITUDE')")

READ (IIN,"(2I5,F10.0)") (NOD(I),IDIRN(I),FLOAD(I),I=1,NLOAD)

WRITE (IOUT,"(' ',I6,9X,I4,7X,E12.5)") (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

END SUBROUTINE LOADS

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 SUBROUTINE LOADV

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

! . .

! . 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 .

! . .

! . Xiong Zhang, (2013) .

! . Computational Dynamics Group, School of Aerospace .

! . Tsinghua Univerity .

! . .

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

SUBROUTINE ELCAL

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

! . .

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

! . generating and storing the element data .

! . .

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

USE GLOBALS

USE MEMALLOCATE

IMPLICIT NONE

INTEGER :: N, I

REWIND IELMNT

WRITE (IOUT,"(//,' E L E M E N T G R O U P D A T A',//)")

! Loop over all element groups

DO N=1,NUMEG

IF (N.NE.1) WRITE (IOUT,'(1X)')

READ (IIN,'(10I5)') NPAR

CALL ELEMNT

IF (MIDEST.GT.MAXEST) MAXEST=MIDEST

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

END DO

RETURN

END SUBROUTINE ELCAL

SUBROUTINE ELEMNT

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

! . .

! . To call the appropriate element subroutine .

! . .

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

USE GLOBALS

IMPLICIT NONE

INTEGER :: NPAR1

NPAR1=NPAR(1)

IF (NPAR1 == 1) THEN

CALL TRUSS

ELSE

! Other element types would be called here, identifying each

! element type by a different NPAR(1) parameter

END IF

RETURN

END SUBROUTINE ELEMNT

SUBROUTINE STRESS (AA)

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

! . .

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

! . .

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

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

IMPLICIT NONE

REAL :: AA(\*)

INTEGER N, I

! Loop over all element groups

REWIND IELMNT

DO N=1,NUMEG

NG=N

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

CALL ELEMNT

END DO

RETURN

END subroutine STRESS

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

! . .

! . 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 .

! . .

! . Xiong Zhang, (2013) .

! . Computational Dynamics Group, School of Aerospace .

! . Tsinghua Univerity .

! . .

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

SUBROUTINE TRUSS

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

! . .

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

! . .

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

USE GLOBALS

USE MEMALLOCATE

IMPLICIT NONE

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

NUME = NPAR(2)

NUMMAT = NPAR(3)

! Allocate storage for element group data

IF (IND == 1) THEN

MM = 2\*NUMMAT\*ITWO + 7\*NUME + 6\*NUME\*ITWO

CALL MEMALLOC(11,"ELEGP",MM,1)

END IF

NFIRST=NP(11) ! Pointer to the first entry in the element group data array

! in the unit of single precision (corresponding to A)

! Calculate the pointer to the arrays in the element group data

! N101: E(NUMMAT)

! N102: AREA(NUMMAT)

! N103: LM(6,NUME)

! N104: XYZ(6,NUME)

! N105: MTAP(NUME)

N101=NFIRST

N102=N101+NUMMAT\*ITWO

N103=N102+NUMMAT\*ITWO

N104=N103+6\*NUME

N105=N104+6\*NUME\*ITWO

N106=N105+NUME

NLAST=N106

MIDEST=NLAST - NFIRST

CALL RUSS (IA(NP(1)),DA(NP(2)),DA(NP(3)),DA(NP(4)),DA(NP(4)),IA(NP(5)), &

A(N101),A(N102),A(N103),A(N104),A(N105))

RETURN

END SUBROUTINE TRUSS

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

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

! . .

! . TRUSS element subroutine .

! . .

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

USE GLOBALS

USE MEMALLOCATE

IMPLICIT NONE

INTEGER :: ID(3,NUMNP),LM(6,NPAR(2)),MATP(NPAR(2)),MHT(NEQ)

REAL(8) :: X(NUMNP),Y(NUMNP),Z(NUMNP),E(NPAR(3)),AREA(NPAR(3)), &

XYZ(6,NPAR(2)),U(NEQ)

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

INTEGER :: NPAR1, NUME, NUMMAT, ND, I, J, L, N

INTEGER :: MTYPE, IPRINT

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

NPAR1 = NPAR(1)

NUME = NPAR(2)

NUMMAT = NPAR(3)

ND=6

! Read and generate element information

IF (IND .EQ. 1) THEN

WRITE (IOUT,"(' E L E M E N T D E F I N I T I O N',//, &

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

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

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

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

' NUMBER OF ELEMENTS.',10(' .'),'( NPAR(2) ) . . =',I5,/)") NPAR1,NUME

IF (NUMMAT.EQ.0) NUMMAT=1

WRITE (IOUT,"(' M A T E R I A L D E F I N I T I O N',//, &

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

' AND CROSS-SECTIONAL CONSTANTS ', &

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

WRITE (IOUT,"(' SET YOUNG''S CROSS-SECTIONAL',/, &

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

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

DO I=1,NUMMAT

READ (IIN,'(I5,2F10.0)') N,E(N),AREA(N) ! Read material information

WRITE (IOUT,"(I5,4X,E12.5,2X,E14.6)") N,E(N),AREA(N)

END DO

WRITE (IOUT,"(//,' E L E M E N T I N F O R M A T I O N',//, &

' ELEMENT NODE NODE MATERIAL',/, &

' NUMBER-N I J SET NUMBER')")

N=0

DO WHILE (N .NE. NUME)

READ (IIN,'(5I5)') N,I,J,MTYPE ! Read in element information

! Save element information

XYZ(1,N)=X(I) ! Coordinates of the element's left node

XYZ(2,N)=Y(I)

XYZ(3,N)=Z(I)

XYZ(4,N)=X(J) ! Coordinates of the element's right node

XYZ(5,N)=Y(J)

XYZ(6,N)=Z(J)

MATP(N)=MTYPE ! Material type

DO L=1,6

LM(L,N)=0

END DO

DO L=1,3

LM(L,N)=ID(L,I) ! Connectivity matrix

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

END DO

! Update column heights and bandwidth

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

WRITE (IOUT,"(I5,6X,I5,4X,I5,7X,I5)") N,I,J,MTYPE

END DO

RETURN

! Assemble stucture stiffness matrix

ELSE IF (IND .EQ. 2) THEN

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) ! Length of element N

XX=E(MTYPE)\*AREA(MTYPE)\*XL ! E\*A\*l

DO L=1,3

ST(L)=D(L)/XL2

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

END DO

DO J=1,ND

YY=ST(J)\*XX

DO I=1,J

S(I,J)=ST(I)\*YY

END DO

END DO

CALL ADDBAN (DA(NP(3)),IA(NP(2)),S,LM(1,N),ND)

END DO

RETURN

! Stress calculations

ELSE IF (IND .EQ. 3) THEN

IPRINT=0

DO N=1,NUME

IPRINT=IPRINT + 1

IF (IPRINT.GT.50) IPRINT=1

IF (IPRINT.EQ.1) WRITE (IOUT,"(//,' S T R E S S C A L C U L A T I O N S F O R ', &

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

' ELEMENT',13X,'FORCE',12X,'STRESS',/,' NUMBER')") 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,"(1X,I5,11X,E13.6,4X,E13.6)") N,P,STR

END DO

ELSE

STOP "\*\*\* ERROR \*\*\* Invalid IND value."

END IF

END SUBROUTINE RUSS

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

! . .

! . 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 .

! . .

! . Xiong Zhang, (2013) .

! . Computational Dynamics Group, School of Aerospace .

! . Tsinghua Univerity .

! . .

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

SUBROUTINE COLHT (MHT,ND,LM)

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

! . .

! . To calculate column heights .

! . .

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

USE GLOBALS, ONLY : NEQ

IMPLICIT NONE

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

INTEGER :: I, LS, II, ME

LS=HUGE(1) ! The largest integer number

DO I=1,ND

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

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

END IF

END DO

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

RETURN

END SUBROUTINE COLHT

SUBROUTINE ADDRES (MAXA,MHT)

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

! . .

! . To calculate addresses of diagonal elements in banded .

! . matrix whose column heights are known .

! . .

! . MHT = Active column heights .

! . MAXA = Addresses of diagonal elements .

! . .

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

USE GLOBALS, ONLY : NEQ, MK, NWK

IMPLICIT NONE

INTEGER :: MAXA(NEQ+1),MHT(NEQ)

INTEGER :: NN, I

! Clear array maxa

NN=NEQ + 1

DO I=1,NN

MAXA(I)=0.0

END DO

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)

RETURN

END SUBROUTINE ADDRES

SUBROUTINE ASSEM (AA)

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

! . .

! . To call element subroutines for assemblage of the .

! . structure stiffness matrix .

! . .

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

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

IMPLICIT NONE

REAL :: AA(\*)

INTEGER :: N, I

REWIND IELMNT

DO N=1,NUMEG

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

CALL ELEMNT

END DO

RETURN

END SUBROUTINE ASSEM

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

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

! . .

! . To assemble element stiffness into compacted global stiffness .

! . .

! . A = GLOBAL STIFFNESS (1D skyline storage) .

! . S = ELEMENT STIFFNESS .

! . ND = DEGREES OF FREEDOM IN ELEMENT STIFFNESS .

! . .

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

USE GLOBALS, ONLY : NWK, NEQ

IMPLICIT NONE

REAL(8) :: A(NWK),S(ND,ND)

INTEGER :: MAXA(NEQ+1),LM(ND)

INTEGER :: ND, I, J, II, JJ, KK

DO J=1,ND

JJ=LM(J)

IF (JJ .GT. 0) THEN

DO I=1,J

II=LM(I)

IF (II .GT. 0) THEN

IF (JJ .GE. II) THEN

KK= MAXA(JJ) + JJ - II

ELSE

KK= MAXA(II) + II - JJ

END IF

A(KK)=A(KK) + S(I,J)

END IF

END DO

END IF

END DO

RETURN

END SUBROUTINE ADDBAN

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

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

! . .

! . To solve finite element static equilibrium equations in .

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

! . .

! . - - Input variables - - .

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

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

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

! . elements of stiffness matrix in a .

! . NN = Number of equations .

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

! . NNM = NN + 1 .

! . KKK = Input flag .

! . EQ. 1 Triangularization of stiffness matrix .

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

! . IOUT = UNIT used for output .

! . .

! . - - OUTPUT - - .

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

! . V(NN) = Displacement vector .

! . .

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

USE GLOBALS, ONLY : IOUT

IMPLICIT NONE

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

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

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

INTEGER :: MIN0

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

IF (KKK == 1) THEN

DO N=1,NN

KN=MAXA(N)

KL=KN + 1

KU=MAXA(N+1) - 1

KH=KU - KL

IF (KH > 0) THEN

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

ENDIF

IF (KH >= 0) THEN

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

ENDIF

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

WRITE (IOUT,"(//' STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE',//, &

' NONPOSITIVE PIVOT FOR EQUATION ',I8,//,' PIVOT = ',E20.12 )") N,A(KN)

STOP

END IF

END DO

ELSE IF (KKK == 2) THEN

! REDUCE RIGHT-HAND-SIDE LOAD VECTOR

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

! BACK-SUBSTITUTE

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

END IF

END SUBROUTINE COLSOL

! --------------------------------------------------------------------------

! - -

! - MEMALLOCATE : A storage manage package for finite element code -

! - -

! - Xiong Zhang, (2013) -

! - Computational Dynamics Group, School of Aerospace -

! - Tsinghua Univerity -

! - -

! - List of subroutine -

! - -

! - memalloca - allocate an array in the shared storage -

! - memfree - deallocate the specified array -

! - memfreefrom - deallocate all arrays from the specified array -

! - memfreefromto - deallocate all arrays between the specified arrays -

! - memprint - print the contents of the specified array -

! - memprintptr - print a subset of the storage in given format -

! - meminfo - list all allocated arrays -

! - -

! --------------------------------------------------------------------------

module memAllocate

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

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

! 1 - Single precision arithmetic

! 2 - Double precision arithmetic

real(4) :: A(MTOT)

real(8) :: DA(MTOT/ITWO)

integer :: IA(MTOT)

equivalence (A,IA), (A,DA) ! A, DA, and IA share the same storage units

integer, parameter :: amax = 200 ! Maximum number of arrays allowed

integer :: np(amax) = 0 ! Pointer to each array

integer :: alen(amax) = 0 ! Length of each array

integer :: aprec(amax) = 0 ! Precision of each array

character\*8 :: aname(amax) = ""

integer :: nplast = 0 ! Pointer to the last allocated element in A

! nplast is in the unit of single precision

contains

subroutine memalloc(num, name, len, prec)

! -----------------------------------------------------------------------------

! - Purpose -

! - Allocate an array in the storage of A -

! - -

! - Input -

! - num - Number of the array allocated -

! - name - Name of the array -

! - len - Length of the array (total number of elements of the array) -

! - prec - Precision of the array -

! - 1: Single precision -

! - 2 : Double precesion -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: num, len, prec

character\*5 name

integer :: i, npfirst

if (num < 1 .or. num > amax) then

write(\*,'("\*\*\* Error \*\*\* Invalid array number: ",I3)') num

stop

end if

if (prec < 1 .or. prec > 2) then

write(\*,'("\*\*\* Error \*\*\* Invalid array type: ",I3)') prec

stop

end if

if (np(num) > 0) call memfree(num) ! array num exists

if (nplast+len\*prec > MTOT) then

write(\*,'("\*\*\* Error \*\*\* No adequate storage available in A",/, &

" Required :", I10, /, &

" Available :", I10)') len\*prec, MTOT - nplast

stop

end if

npfirst = nplast + 1

np(num) = nplast/prec + 1 ! In the unit of allocated array

aname(num) = name

alen(num) = len

aprec(num) = prec

nplast = nplast + len\*prec

if (mod(nplast,2) == 1) nplast = nplast+1 ! Make nplast an even number

do i = npfirst, nplast

A(i) = 0

end do

end subroutine memalloc

subroutine memfree(num)

! -----------------------------------------------------------------------------

! - Purpose -

! - Free the array num and compact the storage if necessary -

! - -

! - Input -

! - num - Number of the array to be deallocated -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: i, num, npbase, nplen

if (np(num) <= 0) return ! The array has not been allocated

! Base address of the array num in the single precision unit

npbase = (np(num)-1)\*aprec(num)

! Length of the array num in the single precision unit

nplen = ceiling(alen(num)\*aprec(num)/2.0)\*2 ! Make nplen an even number

! Compact the storage if neccessary

if (npbase+nplen < nplast) then

! Move arrays behind the array num forward to reuse its storage

do i = npbase+nplen+1, nplast

A(i-nplen) = A(i)

end do

! Update the pointer of arrays behind the array num

do i = 1, amax

if ((np(i)-1)\*aprec(i) > npbase) np(i) = np(i) - nplen/aprec(i)

end do

end if

np(num) = 0

aname(num) = ""

alen(num) = 0

aprec(num) = 0

nplast = nplast - nplen

end subroutine memfree

subroutine memfreefrom(num)

! -----------------------------------------------------------------------------

! - Purpose -

! - Free all arrays from num to the end -

! - -

! - Input -

! - num - Number of the array to be deallocated from -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: i, num

do i=amax,num,-1

call memfree(i)

end do

end subroutine memfreefrom

subroutine memfreefromto(n1,n2)

! -----------------------------------------------------------------------------

! - Purpose -

! - Free all arrays from n1 to n2 -

! - -

! - Input -

! - n1 - Number of the array to be deallocated from -

! - n2 - Number of the array to be deallocated to -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: i, n1, n2

do i=n2,n1,-1

call memfree(i)

end do

end subroutine memfreefromto

subroutine memprint(num)

! -----------------------------------------------------------------------------

! - Purpose -

! - Print the contents of the array num -

! - -

! - Input -

! - num - Number of the array to be printed -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: num,i

if (np(num) <= 0) then

write(\*,'("\*\*\* Error \*\*\* Array ", I3, " has not been allocated.")') num

return

end if

write(\*,'("Contents of Array ", A5, ":")') aname(num)

if (aprec(num) == 1) then

write(\*,'(8I10)') (IA(i), i=np(num),np(num)+alen(num)-1)

else

write(\*,'(8E10.2)') (DA(i), i=np(num),np(num)+alen(num)-1)

end if

end subroutine memprint

subroutine memprintptr(ptr, len, atype)

! -----------------------------------------------------------------------------

! - Purpose -

! - Print the contents of the stroage starting from ptr -

! - -

! - Input -

! - ptr - Pointer to the first entry (in single precision unit) -

! - len - Total number of entries to be printed -

! - atype - Type of the entries (0 - integer; 1 - float; 2 - double) -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: i, ptr, len, atype

character\*8 dtype(3)

data dtype/"integer","real","double"/

write(\*,'("Contents of storage starting from ", I5, " in ", A8, ":")') ptr, dtype(atype+1)

if (atype == 0) then

write(\*,'(8I10)') (IA(i), i=ptr,ptr+len-1)

else if (atype == 1) then

write(\*,'(8E10.2)') (A(i), i=ptr,ptr+len-1)

else if (atype == 2) then

write(\*,'(8E10.2)') (DA(i), i=(ptr-1)/ITWO+1, (ptr-1)/ITWO+len)

end if

end subroutine memprintptr

subroutine meminfo

! -----------------------------------------------------------------------------

! - Purpose -

! - Print the information of the storage -

! - -

! -----------------------------------------------------------------------------

implicit none

integer :: i

write(\*,'("List of all arrays:")')

write(\*,'(" Number Name Length Pointer Precision")')

do i=1,amax

if (np(i) == 0) cycle

write(\*,'(I7, 4X, A5, I9, I10, I12)') i, aname(i), alen(i), np(i), aprec(i)

end do

end subroutine meminfo

end module memAllocate

函数索引

ADDBAN, 16

ADDRES, 15

ASSEM, 16

CLOSEFILES, 6

COLHT, 15

COLSOL, 16

ELCAL, 9

ELEMNT, 9

GLOBALS, 1

INPUT, 7

LOADS, 8

LOADV, 8

memalloc, 19

memAllocate, 19

memfree, 20

memfreefrom, 20

memfreefromto, 21

meminfo, 22

memprint, 21

memprintptr, 21

OPENFILES, 5

RUSS, 11

SECOND, 5

STAP90, 2

STRESS, 10

TRUSS, 11

WRITED, 5