APPENDIX A1 – Fortran program for modified elastic compensation

```
PROGRAM FEA
!
!    CONTROL MAIN PROGRAM  --  FEA.FOR -- PLANE STRESS
!

    DOUBLE PRECISION CO,S,DISP,PLOAD,TH,V,SMAX,YM,LOWLMULT,Z,SM
    INTEGER ELEM,NNODE,NDOF,NELEM,NN,ANA,NITER,ITER
    CHARACTER*1 ANS
    DIMENSION CO(500,2),NN(500,8),SMAX(500)
    DIMENSION S(1000,1000),DISP(1000),PLOAD(1000)
          DOUBLE PRECISION, DIMENSION (0:10) :: SHAKE
          DOUBLE PRECISION, DIMENSION(1:500,10:13) :: VMISES
          DOUBLE PRECISION, DIMENSION(1:500,1:4) :: VMGAUSS
          DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D
          DOUBLE PRECISION, DIMENSION(0:10,1:500,1:4) :: E

          SM=300          ! YIELD STRESS
          NITER=10        ! NUMBER OF ITERATIONS (ALSO CHANGE UPPER LIMIT OF
                                  ![SHAKE] = NITER
                                  ! & DIMENSIONS OF [E]
                                  ! YOU CAN USE ALLOCATABLE ARRAY SIZE HERE
                                  ! niter=0 for elastic analyses

!    TYPES OF ELEMENTS
10   PRINT *,'INPUT TYPES OF ELEMENTS'
    PRINT *
    PRINT *,'1 - FOR 2D-4 NODE QUADRILATERAL'
    PRINT *,'2 - FOR 2D-8 NODE QUADRILATERAL'
    PRINT *,'13 - TO QUIT THIS PROGRAM'
    PRINT *
    PRINT *
!
          ITER = 0                  !ELASTIC SOLUTION

    READ *,ELEM
          PRINT *
          PRINT *

!         DEFINE INITIAL ELASTIC CONSTANTS AND THICKNESS FOR PLANE STRESS
!
    PRINT *,'INPUT YOUNGS MODULUS, POISSON RATIO, THICKNESS'
    PRINT *

    READ *,YM,V,TH
    PRINT *,YM,V,TH
          READ *,NUMBER

!         BUILD GEOMETRY
    PRINT *
    PRINT *
    PRINT *,'INPUT THE MAXIMUM NODE NUMBER'
    PRINT *
    PRINT *
    READ *,NNODE
    PRINT *
    PRINT *
    PRINT *,'INPUT THE TOTAL NUMBER OF ELEMENTS'
```

```
      PRINT *
      PRINT *
      READ *,NELEM
      PRINT *
      PRINT *

      CALL COORD (NNODE,CO)

      PRINT *

      NDOF=NNODE*2
      PRINT *,'NDOF=',NDOF

!           DEFINE ELEMENT BY NODE NUMBERS
      CALL ELEMCON1 (NELEM,NN,ELEM)

DO 2000 ITER=0,NITER

!MODIFY YOUNG'S MODULUS & [D] MATRIX
          CALL PSTRESS (D,TH,NELEM,YM,V,ITER,SMAX,SM,VMGAUSS,VMISES)

      PRINT *
      PRINT *,' ASSEMBLY OF GLOBAL STIFFNESS MATRIX'
      PRINT *

!           FORMULATE GLOBAL STIFFNESS MATRIX
222   CALL STIFFNESS1 (NDOF,NELEM,NN,ELEM,CO,D,TH,S)

!           APPLY BOUNDARY CONDITIONS TO YOUR MODEL DURING ITERATION ZERO
CALL DISPLOAD (NDOF,S,PLOAD,ITER)

330   PRINT *
      PRINT *,' SOLUTION PHASE'
      PRINT *

      CALL SOLUTION1 (NDOF,S,PLOAD,DISP)

      PRINT *
      PRINT *,'  PRINTING OF DISPLACEMENTS'
      PRINT *

      CALL PRDISP (NNODE,DISP)

      PRINT *
      PRINT *,' PRINTING OF STRESSES'
      PRINT *

      CALL PRSTRESS (NELEM,NDOF,NN,ELEM,CO,D,DISP,TH,VMISES,VMGAUSS,iter)

!           FIND LARGEST VMISES STRESS (AT NODES) FOR EACH ELEMENT AND STORE IN
!           ARRAY SMAX
            DO COUNT = 1,NELEM
             SMAX(COUNT) =
MAX(VMises(COUNT,10),VMises(COUNT,11),VMises(COUNT,12),VMises(COUNT,13))
            ENDDO

             SHAKE(ITER) = MAXVAL(SMAX)       !STORE MAX VMISES FROM WHOLE MODEL

            PRINT *, SHAKE(ITER)
            write (10001,*) 'shake(iter)',shake(iter)
```

```
!           READ *,NUMBER

2000  CONTINUE
                  Z=MINVAL(SHAKE)

           LOWLMULT=SM/Z     !CALCULATE BEST LIMIT LOAD MULTIPLIER

          DO 999 I = 0,NITER
           PRINT *, SHAKE(I)
999       ENDDO

     PRINT *
           PRINT *
           PRINT *, 'LOWER LIMIT LOAD MULTIPLIER IS : ', LOWLMULT
           PRINT *

990   PRINT *
    PRINT *,'DO YOU WANT TO START AGAIN? (Y FOR YES, N FOR NO)'
    PRINT *
    READ (5,995) ANS
    IF (ANS.EQ.'Y' .OR. ANS.EQ.'y') THEN
    GOTO 10
    ELSEIF (ANS.EQ.'N' .OR. ANS.EQ.'n') THEN
    GOTO 1000
    ENDIF
995   FORMAT (A1)
1000  END


!    ---------------------------------------------------------------
!    TO READ IN ALL NODAL CO-ORDINATES
!    ---------------------------------------------------------------

    SUBROUTINE COORD (NNODE,CO)
    DOUBLE PRECISION CO
    DIMENSION CO(500,2)
          CHARACTER(15) :: FILENAME
          INTEGER :: OPENSTATUS, INPUTSTATUS

          PRINT 200
          READ *,FILENAME

          OPEN (UNIT = 10, FILE = FILENAME, STATUS = "OLD", &
                POSITION="REWIND",ACTION = "READ", IOSTAT = OPENSTATUS)
           IF (OPENSTATUS > 0) STOP " *** CANNOT OPEN FILE ***"

     DO 100 I=1,NNODE

            READ (UNIT = 10, FMT = *, IOSTAT = INPUTSTATUS) CO(I,1),CO(I,2)

100    CONTINUE

200     FORMAT (' ENTER FILENAME TO READ NODAL CO-ORDINATES',A15)

          CLOSE (10)

    PRINT *
    PRINT *,' NODE NUMBER      X-COORDINATES       Y-COORDINATES'
    PRINT *
    DO 300 I=1,NNODE
```

```
      PRINT 150,I,CO(I,1),CO(I,2)
                    IF (I.EQ.20 .OR. I.EQ.40 .OR. I.EQ.60) THEN
      READ *,NUMBER
       ENDIF
150   FORMAT (2X,I5,2(12X,F10.4))
300   CONTINUE
      PRINT *
      PRINT *

      RETURN
      END

!     ------------------------------------------------------------------------------------------------
!     TO READ IN NODE NUMBER CONNECTING EACH ELEMENT (TYPE NO:1,2)
!     ------------------------------------------------------------------------------------------------
!           REMEMBER :: TO CREATE ELEMENTS ENTER NODE NUMBER IN ANTI-
!CLOCKWISE SENSE

      SUBROUTINE ELEMCON1 (NELEM,NN,ELEM)
      INTEGER NELEM,ELEM,NN
      DIMENSION NN(500,8)
            CHARACTER(15) :: FILENAME
            INTEGER :: OPENSTATUS, INPUTSTATUS

            PRINT 310
            READ *,FILENAME
310   FORMAT (' ENTER FILENAME TO READ ELEMCON',A15)

      OPEN (UNIT = 15, FILE = FILENAME, STATUS = "OLD", &
                        POSITION="REWIND",ACTION = "READ", IOSTAT = OPENSTATUS)
             IF (OPENSTATUS > 0) STOP " *** CANNOT OPEN FILE ***"

       IF (ELEM.EQ.1) THEN

       DO 100 I=1,NELEM

              READ (UNIT = 15, FMT = *, IOSTAT = INPUTSTATUS) &
                              NN(I,1),NN(I,2),NN(I,3),NN(I,4)

100    CONTINUE

!           CLOSE (15)

      PRINT *
      PRINT *,' ELEMENT NUMBER     NODE-I    NODE-J    NODE-K    NODE-L'
      PRINT *
      PRINT *
      DO 300 I=1,NELEM
      PRINT 250,I,NN(I,1),NN(I,2),NN(I,3),NN(I,4)
            IF (I.EQ.20 .OR. I.EQ.40 .OR. I.EQ.60) THEN
      READ *,NUMBER
      ENDIF

250   FORMAT (6X,I3,13X,I3,3(7X,I3))

300   CONTINUE
      PRINT *
      PRINT *

      ELSEIF (ELEM.EQ.2) THEN
```

```
      DO 101 I=1,NELEM

            READ (UNIT = 15, FMT = *, IOSTAT = INPUTSTATUS) &
            NN(I,1),NN(I,2),NN(I,3),NN(I,4),NN(I,5),NN(I,6),NN(I,7),NN(I,8)
101   CONTINUE

      PRINT *
      PRINT *,' ELEM NUM  N-I  N-J  N-K  N-L  N-M  N-N  N-P  N-Q'
      PRINT *
      PRINT *
      DO 301 I=1,NELEM
      PRINT 251,I,NN(I,1),NN(I,2),NN(I,3),NN(I,4),NN(I,5),NN(I,6),NN(I,7 &
      ),NN(I,8)
251   FORMAT (3X,I3,6X,7(I3,2X),I3)

301   CONTINUE
      PRINT *
      PRINT *

      ENDIF

            CLOSE (15)

      RETURN
      END

!     ---------------------------------------------------------------
!     TO CALCULATE THE D-MATRIX FOR 2D - PLANE STRESS
!     ---------------------------------------------------------------

      SUBROUTINE PSTRESS (D,TH,NELEM,YM,V,ITER,SMAX,SM,VMGAUSS,VMISES)
      DIMENSION SMAX(500)
      DOUBLE PRECISION TH,SMAX,SM,YM,V
            DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D
            DOUBLE PRECISION, DIMENSION(0:10,1:500,1:4) :: E
            DOUBLE PRECISION, DIMENSION(1:500,1:4) :: VMGAUSS
            DOUBLE PRECISION, DIMENSION(1:500,10:13) :: VMISES
            INTEGER :: ENUM,ITER,G

            open (unit=9999, file="Efile",status="old")

!     ZERO ALL ELEMENTS IN D MATRIX
            DO 25 K=1,NELEM
      DO 20 I=1,3
      DO 10 J=1,3
            DO 29 G=1,4
      D(I,J,K,G)=0

29          ENDDO
10    CONTINUE
20    CONTINUE
25          CONTINUE
            IF (ITER == 0) THEN
write (9999,*) 'E - Used for Elastic solution'
             DO I = 1,NELEM
              DO 3001 G = 1,4
               E(ITER,I,G)=YM
3001       ENDDO
write(9999,700) E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i
```

```
print 700, E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i

              ENDDO

             ENDIF
!PRINT *,ITER
! PRINT *, VMGAUSS(1,1),VMGAUSS(1,2),VMGAUSS(1,3),VMGAUSS(1,4)
!READ *, NUMBER

             IF (ITER > 0) THEN
write (9999,*) 'E - Used for iteration',iter

              DO 4000 I = 1,NELEM
!!!!!!!!
! TO CHOOSE WHICH STRESS TO USE TO MODIFY [D] FOR ELCOGAUSS
!!!!!!GAUSS STRESSES here
               DO 4001 G = 1,4
               E(ITER,I,G)=E(ITER-1,I,G)*SM/VMGAUSS(I,G)
                     4001      ENDDO
!!!!!!NODAL STRESSES here
!              DO 4001 G = 1,4
!               E(ITER,I,G)=E(ITER-1,I,G)*SM/VMISES(I,(G+9))
!                     4001      ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! use below for standard elastic compensation
!DO 4001 G = 1,4
!E(ITER,I,G)=E(ITER-1,I,G)*SM/smax(i)
!4001      ENDDO
!!!!!!!
write(9999,700) E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i
print 700, E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i
4000   ENDDO
!!!!!!!
             ENDIF

!            CALCULATE D MATRIX FOR EACH GAUSS POINT IN EACH ELEMENT
             DO 27 ENUM=1,NELEM
              DO 28 G = 1,4
       D(1,1,ENUM,G)=E(ITER,ENUM,G)/(1.-V*V)
       D(2,2,ENUM,G)=D(1,1,ENUM,G)
       D(1,2,ENUM,G)=V*D(1,1,ENUM,G)
       D(2,1,ENUM,G)=D(1,2,ENUM,G)
       D(3,3,ENUM,G)=.5*(1.-V)*D(1,1,ENUM,G)
28            CONTINUE
27           CONTINUE

     DO 41 I=1,3

    PRINT 30,D(I,1,1,1),D(I,2,1,1),D(I,3,1,1)            !PRINTING [D] FOR IST GAUSS PT OF IST
ELEMENT
30    FORMAT (D10.4,2X,D10.4,2X,D10.4)
41    CONTINUE

     PRINT *
     PRINT *
!            READ *, NUMBER
700 format (4En14.4,2x,i3)
     RETURN
     END
```

```
!    ------------------------------------------------------------
!    TO CALCULATE THE D-MATRIX FOR 2D - PLANE STRAIN
!    ------------------------------------------------------------

     SUBROUTINE PSTRAIN (D,TH)
     DIMENSION D(6,6)
     DOUBLE PRECISION D,TH
!
!    DEFINE ELASTIC CONSTANTS
!
     PRINT *
     PRINT *,'INPUT YOUNGS MODULUS, POISSON RATIO, THICKNESS'
     PRINT *

     READ *,E,V,TH
     PRINT *

     DO 20 I=1,3
     DO 10 J=1,3
     D(I,J)=0.
10   CONTINUE
20   CONTINUE

     D(1,1)=E*(1.-V)/((1.+V)*(1.-2.*V))
     D(2,2)=D(1,1)
     D(1,2)=D(1,1)*V/(1.-V)
     D(2,1)=D(1,2)
     D(3,3)=.5*E/(1.+V)

     DO 40 I=1,3

     PRINT 30,D(I,1),D(I,2),D(I,3)
30   FORMAT (D10.4,2X,D10.4,2X,D10.4)
40   CONTINUE

     PRINT *
     PRINT *

     RETURN
     END

!    ------------------------------------------------------------------
!    TO CALCULATE THE D-MATRIX FOR 3D - ISOTROPIC MATERIALS
!    ------------------------------------------------------------------

     SUBROUTINE THREED (D)
     DIMENSION D(6,6)
     DOUBLE PRECISION D
!
!    DEFINE ELASTIC CONSTANTS
!
     PRINT *
     PRINT *,'INPUT YOUNGS MODULUS, POISSON RATIO'
     PRINT *
     READ *,E,V
     PRINT *

     DO 20 I=1,6
     DO 10 J=1,6
     D(I,J)=0.
```

```
10   CONTINUE
20   CONTINUE

     C=E/((1.+V)*(1.-2*V))
     D(1,1)=(1.-V)*C
     D(2,2)=D(1,1)
     D(3,3)=D(1,1)
     D(1,2)=V*C
     D(1,3)=D(1,2)
     D(2,3)=D(1,2)
     D(4,4)=.5*E/(1.+V)
     D(5,5)=D(4,4)
     D(6,6)=D(4,4)

     DO 40 I=1,5
     DO 30 J=I+1,6
     D(J,I)=D(I,J)
30   CONTINUE
40   CONTINUE

     DO 60 I=1,6
     PRINT 50,D(I,1),D(I,2),D(I,3),D(I,4),D(I,5),D(I,6)
50   FORMAT (D10.4,5(2X,D10.4))
60   CONTINUE

     PRINT *
     PRINT *

     RETURN
     END

!    ---------------------------------------------------------------
!    FORM GLOBAL STIFFNESS MATRIX
!    ---------------------------------------------------------------

     SUBROUTINE STIFFNESS1 (NDOF,NELEM,NN,ELEM,CO,D,TH,S)
     INTEGER NDOF,NELEM,NN,ELEM,II
     DOUBLE PRECISION DJ,EM,C,T,S,CO,S3,A,B,TH
     DIMENSION EM(16,16),C(20,20),T(20,20),S(1000,1000)
     DIMENSION NN(500,8),CO(500,2)
          DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D

!         ZERO S MATRIX
     DO 20 I=1,NDOF
     DO 10 J=1,NDOF
     S(I,J)=0.                ! S IS THE GLOBAL STIFFNESS MATRIX
10   CONTINUE
20   CONTINUE

     S3=0.577350269

     DO 150 N=1,NELEM  !ELEMENT NO. IS TAKEN FROM HERE

!         ZERO EM MATRIX
!         em of previous element is already stored in global matrix
     DO 40 I=1,8*ELEM
     DO 30 J=1,8*ELEM
     EM(I,J)=0.
30   CONTINUE
40   CONTINUE
```

```
!           II is used to calculate K for the 4 Gauss points
      DO 100 II=1,4
      A=S3
      B=S3
      IF (II.EQ.1) THEN
      A=-A
      B=-B
      ELSEIF (II.EQ.2) THEN
      B=-B
      ELSEIF (II.EQ.4) THEN
      A=-A
      ENDIF

      CALL MATRIX1 (NN,N,ELEM,D,A,B,CO,DJ,T,C,II)

      DJ=ABS(DJ)
!           form k for each Gauss point in subroutine MATRIX1 and add to integrate
      DO 90 I=1,8*ELEM
      DO 80 J=1,8*ELEM
      DO 70 K=1,3
      EM(I,J)=EM(I,J)+C(K,I)*T(K,J)*DJ*TH
70    CONTINUE
80    CONTINUE
90    CONTINUE
100   CONTINUE
! REMOVE BELOW TO VIEW MATRICES
!     print *,'element no     em matrix '
!     PRINT *
!     DO 105 I=1,8*ELEM
!     print 85,n,em(i,1),em(i,2),em(i,3),em(i,4),em(i,5),em(i,6),em(i,7) &
!     ,em(i,8)
85    FORMAT (I3,8f9.2)
!105   CONTINUE
!     READ *,NUMBER

      IF (ELEM.EQ.2) THEN
      DO 110 I=1,8*ELEM
      PRINT 85,N,EM(I,9),EM(I,10),EM(I,11),EM(I,12),EM(I,13),EM(I,14) &
      ,EM(I,15),EM(I,16)
110   CONTINUE
      !read *,number
      ENDIF


!
!     DO LOOP 140 DOES THE ASSEMBLY OF ELEMENT STIFFNESS MATRIX TO THE
!     GLOBAL STIFFNESS MATRIX
!
      DO 140 I=1,4*ELEM
      K=NN(N,I)
      II1=2*I-1
      II2=2*I
      KI1=2*K-1
      KI2=2*K
!      IN=NN(N,I)
      DO 130 J=1,4*ELEM
      L=NN(N,J)
      JJ1=2*J-1
      JJ2=2*J
```

```
      LJ1=2*L-1
      LJ2=2*L
!     JN=NN(N,J)
!      DO 125 IL=1,2
!       IE=(I-1)*2+IL
!       NR=(IN-1)*2+IL
!         DO 124 JL=1,2
!          JE=(I-1)*2+JL
!          NC=(JN-1)*2+JL
!          S(NR,NC)=S(NR,NC)+EM(IE,JE)
!124       CONTINUE
!125    CONTINUE
      S(KI1,LJ1)=S(KI1,LJ1)+EM(II1,JJ1)
      S(KI1,LJ2)=S(KI1,LJ2)+EM(II1,JJ2)
      S(KI2,LJ1)=S(KI2,LJ1)+EM(II2,JJ1)
      S(KI2,LJ2)=S(KI2,LJ2)+EM(II2,JJ2)

130   CONTINUE
140   CONTINUE

150   CONTINUE

!     PRINT *,'MATRIX OF S'
!     DO 160 I=1,NDOF
!     PRINT 155,S(I,1),S(I,2),S(I,3),S(I,4),S(I,5),S(I,6),S(I,7),S(I,8)
!    +,S(I,9)
!160   CONTINUE
!     read *,number
!     DO 170 I=1,NDOF
!     PRINT 155,S(I,10),S(I,11),S(I,12),S(I,13),S(I,14),S(I,15)
!    + ,S(I,16),S(I,17),S(I,18)
!170   CONTINUE
!155   FORMAT (9f8.2)
!     read *,number
      RETURN
      END

!     ---------------------------------------------------------------
!     TO GENERATE  INTEGRATION OF [D].[B]  FOR ELEMENT 1
!     ---------------------------------------------------------------

      SUBROUTINE MATRIX1 (NN,N,ELEM,D,A,B,CO,DJ,T,C,II)
      DOUBLE PRECISION DKL,TJ,DJ,XY,CO,T,C,A,B,DD
      INTEGER NN,N,K,ELEM,II
      DIMENSION CO(500,2),NN(500,8),XY(20,2),DKL(2,8),TJ(2,2),T(20,20) &
      ,C(20,20)
            DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D

!
!     THE DO-LOOP 20 DOES THE ASSIGNING OF COORDINATES TO APPROPRIATE
!     NODES WITHIN A SINGLE ELEMENT
!
!     PRINT *,'ELEM=',ELEM
      DO 20 I=1,4*ELEM
      K=NN(N,I)
      XY(I,1)=CO(K,1)
      XY(I,2)=CO(K,2)
20    CONTINUE

      IF (ELEM.EQ.1) THEN
```

```
      DKL(1,1)=(B-1.)/4.
      DKL(1,2)=(1.-B)/4.
      DKL(1,3)=(1.+B)/4.
      DKL(1,4)=-(1.+B)/4.
      DKL(2,1)=(A-1.)/4.
      DKL(2,2)=-(1.+A)/4.
      DKL(2,3)=(1.+A)/4.
      DKL(2,4)=(1.-A)/4.

      ELSEIF (ELEM.EQ.2) THEN

      DKL(1,1)=((B+2.*A)*(1.-B))/4.
      DKL(1,2)=A*(B-1.)
      DKL(1,3)=((B-2.*A)*(B-1.))/4.
      DKL(1,4)=(1.-B*B)/2.
      DKL(1,5)=((B+2.*A)*(1.+B))/4.
      DKL(1,6)=-A*(1.+B)
      DKL(1,7)=((2.*A-B)*(1.+B))/4.
      DKL(1,8)=-(1.-B*B)/2.
      DKL(2,1)=((A+2.*B)*(1.-A))/4.
      DKL(2,2)=-(1.-A*A)/2.
      DKL(2,3)=((1.+A)*(2.*B-A))/4.
      DKL(2,4)=-B*(1.+A)
      DKL(2,5)=((1.+A)*(2.*B+A))/4.
      DKL(2,6)=(1.-A*A)/2.
      DKL(2,7)=((1.-A)*(2.*B-A))/4.
      DKL(2,8)=B*(A-1.)

      ENDIF
!set up Jacobian matrix mine
      DO 50 I=1,2
       DO 40 J=1,2
       TJ(I,J)=0.
                DO 30 K=1,4*ELEM
                  TJ(I,J)=TJ(I,J)+DKL(I,K)*XY(K,J)
 30       CONTINUE
 40     CONTINUE
 50    CONTINUE
!determinant of Jacobian mine
      DJ=TJ(1,1)*TJ(2,2)-TJ(1,2)*TJ(2,1)
      DD=TJ(1,1)
!             inverse of jacobian mine
      TJ(1,1)=TJ(2,2)/DJ
      TJ(2,2)=DD/DJ
      TJ(1,2)=-TJ(1,2)/DJ
      TJ(2,1)=-TJ(2,1)/DJ

      DO 90 I=1,2
       DO 80 J=1,4*ELEM
              T(I,J)=0.
                DO 70 K=1,2
                  T(I,J)=T(I,J)+TJ(I,K)*DKL(K,J)
 70       CONTINUE
 80      CONTINUE
 90    CONTINUE

      DO 100 J=1,4*ELEM
             C(1,2*J-1)=T(1,J)
             C(3,2*J)=T(1,J)
```

```
            C(2,2*J)=T(2,J)
            C(3,2*J-1)=T(2,J)
 100  CONTINUE
!             work out D.B mine /stored in [T]
     DO 150 I=1,3
      DO 130 J=1,8*ELEM
              T(I,J)=0.
               DO 120 K=1,3
               T(I,J)=T(I,J)+D(I,K,N,II)*C(K,J)
 120       CONTINUE
 130   CONTINUE
 150  CONTINUE

     RETURN
     END

!    ----------------------------------------------------------------
!    INPUT NODAL DISPLACEMENTS AND FORCES
!    ----------------------------------------------------------------

     SUBROUTINE DISPLOAD (NDOF,S,PLOAD,ITER)

     INTEGER NUM,INUM,JNUM,A,DIR,INPUTSTATUS,OPENSTATUS,ITER
     DOUBLE PRECISION VALUE,S,PLOAD
     CHARACTER*1 ANS, FILELOAD*15
             REAL :: P

     DIMENSION S(1000,1000),PLOAD(1000)

     DO 10 I=1,NDOF
      PLOAD(I)=0.
10    CONTINUE

     P=1E30

!     'INPUT NODAL FORCES " 1,N(NODE NUMBER),DIR(1,2),VALUE "'
!     ,' DIR IN X DIRECTION AS 1 AND Y DIRECTION AS 2.'
!     ,'NODAL FORCES AS 1'
!
!                    OR'
!
!     ,'INPUT NODAL DISPLACEMENTS " 2,N(NODE NUMBER),DIR(1,2,12), &
!    VALUE."'
!     ,'NODAL DISPLACEMENT AS 2'
!
            IF (ITER == 0) THEN
             PRINT 410
             READ *,FILELOAD
410   FORMAT (' ENTER FILENAME TO READ LOADS & B.C.',A15)
            ENDIF


    OPEN (UNIT = 25, FILE = FILELOAD, STATUS = "OLD", &
                      POSITION="REWIND",ACTION = "READ", IOSTAT = OPENSTATUS)
            IF (OPENSTATUS > 0) STOP " *** CANNOT OPEN FILE ***"

            DO 56

            READ (UNIT = 25, FMT = *, IOSTAT = INPUTSTATUS) &
                              A,NUM,DIR,VALUE
```

```
        IF (INPUTSTATUS < 0) EXIT  ! END OF FILE

    PRINT *,A,NUM,DIR,VALUE

    IF (A.EQ.1) THEN

        IF (DIR.EQ.1) THEN
        INUM=1
        ELSEIF (DIR.EQ.2) THEN
        INUM=0
        ENDIF

        JNUM=2*NUM-INUM
        PLOAD(JNUM)=VALUE
        PRINT *
        PRINT *,'FORCE AT NODE NUMBER',NUM
        PRINT *,'IN DIRECTION',DIR
        PRINT *,'IS OF VALUE',PLOAD(JNUM)
        PRINT *

    ELSEIF (A.EQ.2) THEN

        IF (DIR.EQ.1) THEN
        INUM=1
        JNUM=2*NUM-INUM
        S(JNUM,JNUM)=S(JNUM,JNUM)*P
        PLOAD(JNUM)=S(JNUM,JNUM)*VALUE


        ELSEIF (DIR.EQ.2) THEN
        JNUM=2*NUM
        S(JNUM,JNUM)=S(JNUM,JNUM)*P
        PLOAD(JNUM)=S(JNUM,JNUM)*VALUE


        ELSEIF (DIR.EQ.12) THEN
        DO 50 I=1,2
        INUM=I-1
        JNUM=2*NUM-INUM
        S(JNUM,JNUM)=S(JNUM,JNUM)*P
        PLOAD(JNUM)=S(JNUM,JNUM)*VALUE
50      CONTINUE

        ENDIF


    ENDIF

56 end do
close (25)

    RETURN
    END

!   ------------------------------------------------------------
!   SOLUTION OF SIMULTANEOUS EQUATIONS BY GAUSS ELIMINATION METHOD
!
!   CODING FROM "COMPUTATIONAL METHODS FOR THE SOLUTION OF ENGINEERING
!   PROBLEMS" BY C.A.BREBBIA AND A.J.FERRANTE
!   THIRD REVISED EDITION, PENTECH PRESS   -- PAGE 42
```

```
!     ------------------------------------------------------------

      SUBROUTINE SOLUTION1 (NDOF,S,PLOAD,DISP)

      INTEGER NDOF
      DOUBLE PRECISION S,PLOAD,DISP,C,CALOAD
!     DOUBLE PRECISION SS
      DIMENSION S(1000,1000),PLOAD(1000),DISP(1000)
!     DIMENSION ss(100,100)

      do 99 i=1,ndof
!      do 98 j=1,ndof
!      ss(i,j)=s(i,j)
!98    continue
      DISP(I)=0.
99    continue

!     DO 30 I=1,NDOF
!     PRINT *,S(I,I)
!30    CONTINUE
!     READ *,NUMBER

      N1=NDOF-1
      DO 100 K=1,N1
      C=S(K,K)
      K1=K+1
      IF (ABS(C) .LT.  0.0000000001) THEN
            DO 7 J=K1,NDOF
!
!     TRY TO INTERCHANGE ROWS TO GET NON ZERO DIAGONAL COEFFICIENT
!
              IF (ABS(S(J,K)) .GT. 0.0000000001) THEN
                DO 6 L=K,NDOF
                  C=S(K,L)
                  S(K,L)=S(J,L)
                  S(J,L)=C
6           CONTINUE
              C=PLOAD(K)
              PLOAD(K)=PLOAD(J)
              PLOAD(J)=C
              C=S(K,K)
              PRINT *, ' ROW INTERCHANGED ONCE',S(K,K)
              GOTO 3
             ENDIF
             PRINT *,'  A LOOP ON J'
7       CONTINUE
1     PRINT 2, K
2     FORMAT ('****** SINGULARITY IN ROW',I5)
      GOTO 300
      ENDIF
!
!     DIVIDE ROW BY DIAGONAL COEFFICIENT
!

3     C=S(K,K)
      DO 4 J=K1,NDOF
!      IJ=IJ+1
!      IF (IJ.EQ.20 .OR. IJ.EQ.40 .OR. IJ.EQ.60) THEN
!      READ *,NUMBER
!      ENDIF
```

```
!     PRINT *,' S(K,J) AND C',K,J,S(K,J),C
      S(K,J)=S(K,J)/C
4     CONTINUE
      PLOAD(K)=PLOAD(K)/C

!
!     ELIMINATION OF UNKNOWNS FROM EACH ROW
!

      DO 10 I=K1,NDOF
      C=S(I,K)
      DO 9 J=K1,NDOF
      S(I,J)=S(I,J)-C*S(K,J)
9     CONTINUE
      PLOAD(I)=PLOAD(I)-C*PLOAD(K)
10    CONTINUE
100   CONTINUE

      IF (ABS(S(NDOF,NDOF)) .LT. 0.000001) THEN
      PRINT 2,NDOF
      GOTO 300
      ENDIF

!
!     COMPUTE LAST UNKNOWN
!

101   DISP(NDOF)=PLOAD(NDOF)/S(NDOF,NDOF)

!
!     BACK SUBSTITUTION PROCESS TO COMPUTE REMAINING UNKNOWNS
!
      DO 200 L=1,N1
      K=NDOF-L
      K1=K+1
      DO 150 J=K1,NDOF
      DISP(K)=DISP(K)+ PLOAD(K)-S(K,J)*DISP(J)
      PLOAD(K)=0.
150   CONTINUE
200   CONTINUE

!     DO 250 I=1,NDOF
!     PRINT *,' DISPLACEMENT AT ',I,'  =  ',DISP(I)
!250   CONTINUE

!     READ *,NUMBER

!     print *,'  i     pload(i)'
!     do 199 i=1,ndof
!     pload(i)=0.
!     do 198 j=1,ndof
!     pload(i)=pload(i)+ss(i,j)*disp(j)
!198   continue
!     print *,i, pload(i)
!199   continue
!     read *,number

300   RETURN
      END
```

```
!    ------------------------------------------------------------
!    TO PRINT X AND Y DISPLACEMENT
!    ------------------------------------------------------------

     SUBROUTINE PRDISP (NNODE,DISP)

     INTEGER NNODE
     DOUBLE PRECISION DISP
     DIMENSION DISP(1000)

     PRINT *
     PRINT *, ' THE FOLLOWING X AND Y DISPLACEMENT ARE IN GLOBAL COORD'
     PRINT *
     PRINT *, ' NODE                UX                UY'
     PRINT *
     PRINT *
     DO 50 I=1,NNODE
     I1=2*I-1
     I2=2*I
     PRINT 30, I,DISP(I1),DISP(I2)
     IF (I.EQ.20 .OR. I.EQ.40 .OR. I.EQ.60) THEN
 !    READ *,NUMBER
     ENDIF
30    FORMAT (I3,10X,D18.8,5X,D18.8)
50    CONTINUE
 !    READ *,NUMBER
     RETURN
     END

!    ------------------------------------------------------------------------------------------
!    TO PRINT ELEMENT STRESS AT FOUR GAUSS POINT AND CENTROID
!    ------------------------------------------------------------------------------------------

     SUBROUTINE PRSTRESS
(NELEM,NDOF,NN,ELEM,CO,D,DISP,TH,VMISESD,VMGAUSSD,iter)

     DOUBLE PRECISION CO,DISP,ED,T,ES,A,B,DJ,C,C1,C2,C3,C4,C5,TH,GSX,GSY,GSXY
     INTEGER NELEM,NDOF,NN,ELEM,N,II
     DIMENSION CO(500,2),DISP(1000),NN(500,8),ED(20),T(20,20)
          DOUBLE PRECISION, DIMENSION(10:13) :: SX,SY,SXY
          DOUBLE PRECISION, DIMENSION(1:500,10:13), INTENT(OUT) :: VMISESD !VMISES
AT NODES
          DOUBLE PRECISION, DIMENSION(1:500,1:4), INTENT(OUT) :: VMGAUSSD  !VMISES
AT GAUSS POINTS
          DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D

     DIMENSION ES(17,3),C(20,20)

          open (unit=10000, file="stressG",status="old")
          open (unit=10001, file="stressN",status="old")
          open (unit=10002, file="sgansy",status="old")
          open (unit=10003, file="snansy",status="old")

!
!    TO DETERMINE NUMBER OF EXTRAPOLATED STRESSES REQUIRED
!
     IF (ELEM.EQ.2) THEN
     IEXTRA=8
     ELSEIF (ELEM.EQ.1) THEN
     IEXTRA=4
```

```
    ENDIF

    PRINT *
    PRINT *,'THICKNESS = ',TH
    PRINT *
    PRINT *
    PRINT *,'STRESS 1-4 ARE UTIMATE STRESSES LOCATION +/- 0.57735'
    PRINT *,'STRESS 5 IS CENTROIDAL STRESS'
    PRINT *,'STRESS 6-9 ARE STRESSES AT CORNER NODES'
    PRINT *,'STRESS 10-13 ARE EXTRAPOLATED STRESSES AT CORNER NODES'
    PRINT *,'STRESS 14-17 ARE EXTRAPOLATED STRESSES AT MID-SIDE NODES'
    PRINT *

    DO 100 N=1,NELEM              !TAKES ELEMENT NO. FROM HERE

    PRINT *,' ELEM    G.P.    SIGMA-X        SIGMA-Y        SIGMA-XY'

          DO 30 I=1,4*ELEM

                K=NN(N,I)
                ED(2*I-1)=DISP(2*K-1)
                ED(2*I)  =DISP(2*K)

30        CONTINUE

          DO 95 II=1,4      !N.B. HERE I AM NOT GETTING STRESSES AT CORNER NODES (NOT
NEEDED TILL NOW)

                A=0.577350269
                B=A

                IF (II.EQ.1) THEN
                A=-A
                B=-B

                ELSEIF (II.EQ.2) THEN
                B=-B

                ELSEIF (II.EQ.4) THEN
                A=-A

                ELSEIF (II.EQ.5) THEN
                A=0.
                B=0.

                ELSEIF (II.EQ.6) THEN
                A=-1.
                B=-1.

                ELSEIF (II.EQ.7) THEN
                A=1.
                B=-1.

                ELSEIF (II.EQ.8) THEN
                A=1.
                B=1.

                ELSEIF (II.EQ.9) THEN
                A=-1.
                B=1.
```

```
            ENDIF

          CALL MATRIX1 (NN,N,ELEM,D,A,B,CO,DJ,T,C,II)          !WORK OUT DB FOR EACH
!ELEMENT

            DO 80 I=1,3

                  ES(II,I)=0.

                  DO 70 J=1,8*ELEM

                        ES(II,I)=ES(II,I)+T(I,J)*ED(J)

70             CONTINUE
80         CONTINUE

!                 WORK OUT V.MISES STRESS AT EACH GAUSS POINT
                  GSX=(ES(II,1))
                  GSY=(ES(II,2))
                  GSXY=(ES(II,3))

                  IF (II <= 4) THEN
                   VMGAUSSD(N,II)=SQRT((GSX**2) + (GSY**2) - (GSX*GSY) + 3*(GSXY**2))
                  ENDIF

            PRINT 85,N,II,ES(II,1),ES(II,2),ES(II,3)
85         FORMAT (I3,5X,I3,3X,F15.5,2(2X,F15.5))

95       CONTINUE

            C1=1.866
            C2=0.5
            C3=0.133975
            C4=0.683
            C5=0.183

            DO 98 I=1,3
            ES(10,I)=C1*ES(1,I)-C2*ES(2,I)+C3*ES(3,I)-C2*ES(4,I)
            ES(11,I)=-C2*ES(1,I)+C1*ES(2,I)-C2*ES(3,I)+C3*ES(4,I)
            ES(12,I)=C3*ES(1,I)-C2*ES(2,I)+C1*ES(3,I)-C2*ES(4,I)
            ES(13,I)=-C2*ES(1,I)+C3*ES(2,I)-C2*ES(3,I)+C1*ES(4,I)

!           ES(10,I)=C1*ES(1,I)-C2*ES(2,I)+C3*ES(3,I)-C2*ES(4,I)
!           ES(13,I)=-C2*ES(1,I)+C1*ES(2,I)-C2*ES(3,I)+C3*ES(4,I)
!           ES(12,I)=C3*ES(1,I)-C2*ES(2,I)+C1*ES(3,I)-C2*ES(4,I)
!           ES(11,I)=-C2*ES(1,I)+C3*ES(2,I)-C2*ES(3,I)+C1*ES(4,I)

            IF (ELEM.EQ.2) THEN
            ES(14,I)=C4*ES(1,I)+C4*ES(2,I)-C5*ES(3,I)-C5*ES(4,I)
            ES(15,I)=-C5*ES(1,I)+C4*ES(2,I)+C4*ES(3,I)-C5*ES(4,I)
            ES(16,I)=-C5*ES(1,I)-C5*ES(2,I)+C4*ES(3,I)+C4*ES(4,I)
            ES(17,I)=C4*ES(1,I)-C5*ES(2,I)-C5*ES(3,I)+C4*ES(4,I)
            ENDIF
98         CONTINUE

!CALC VMISES AT NODES USING EXTRAPOLATED CORNER NODES STRESSES
                  DO 103 II = 10,13
                   SX(II)=(ES(II,1))
                   SY(II)=(ES(II,2))
```

```
                    SXY(II)=(ES(II,3))
                    VMISESD(N,II)=SQRT(((SX(II))**2) + ((SY(II))**2) - ((SX(II))*(SY(II))) +
3*((SXY(II))**2))
                103 CONTINUE
!
             DO 99 II=10,9+IEXTRA
             PRINT 85,N,II,ES(II,1),ES(II,2),ES(II,3)
99         CONTINUE

! PRINT VMISES STRESSES AT THE CORNER NODES
              PRINT *, 'VON MISES STRESSES AT NODES (DERIVED FROM EXTRAPOLATED
STRESSES)'
            DO 998 I = 10,13
             PRINT *, VMISESD(N,I)
998         ENDDO
write (10001,*) 'nodal vmises stresses +++++ iteration ',iter,'  elem ',n
write (10001,701) VMisesD(N,10),VMisesD(N,11),VMisesD(N,12),VMisesD(N,13)
write (10003,701) VMisesD(N,10),VMisesD(N,11),VMisesD(N,12),VMisesD(N,13)

! PRINT VMISES STRESSES AT THE gauss pts
write (10000,*) 'Gauss point vmises stresses +++++ iteration ',iter,'  elem ',n
write (10000,701) VMGAUSSD(N,1),VMGAUSSD(N,2),VMGAUSSD(N,3),VMGAUSSD(N,4)
write (10002,701) VMGAUSSD(N,1),VMGAUSSD(N,2),VMGAUSSD(N,3),VMGAUSSD(N,4)

            DO 999 I = 1,4
             PRINT *, 'VON MISES STRESSES AT gauss points'
             PRINT *, VMGAUSSD(N,I)

999         ENDDO

100   CONTINUE

701 format(4en16.4)

    RETURN
    END
```

APPENDIX A2 – Fortran program for 'software interface'

```
program coords

implicit none

real :: xnode,ynode,dummyr
integer :: elenum,n1,n2,n3,n4,n5,n6,n7,n8,mat
integer :: nodenum,dummyi,increment,i,maxnodes,maxelem

!cdwrite,geom,filename,ext

increment =0
mat=1
!maxnodes=2521        !Obtain from Ansys model
!maxelem=800          !Obtain from Ansys model

print *,'enter number of nodes'
read (*,35) maxnodes
print *,'enter number of elements'
read (*,36) maxelem
```

```
open(unit=12, file="po", status="old") !use for pcfeap
!open(unit=12, file="ans15", status="old")   !use for yew

open(unit=13, file="mudell", status="new")

write(13,*) 'coor'
do 24 i=1,maxnodes
read(12,31) nodenum,dummyi,dummyi,xnode,ynode,dummyr,dummyr,dummyr,dummyr

!write(13,*) xnode,',',ynode  !use for yew
write(13,34) nodenum,increment,xnode,ynode  !use for pcfeap

24 continue

write(13,*)
write(13,*) 'elem'
do 25 i=1,maxelem
read(12,32) dummyi,dummyi,dummyi,dummyi,dummyi,dummyi,dummyi,dummyi, &
                    dummyi,dummyi,elenum,n1,n2,n3,n4,n5,n6,n7,n8

!write(13,*) n1,',',n2,',',n3,',',n4                                    !use for yew
!write(13,33) elenum,increment,elenum,n1,n2,n3,n4,n5,n6,n7,n8          !use for pcfeap
write(13,33) elenum,increment,mat,n1,n2,n3,n4,n5,n6,n7,n8    !use for pcfeap

25 continue

      31 format(3i8,6e16.9)
      41 format(3e16.9)
      34 format(2i10,6f10.3)
      32 format(19i7)
      35 format(i10)
      36 format(i10)
      33 format(16i5)

end program coords
```

APPENDIX A3 – Fortran program for dividing each element in 4 areas used to plot stress contours

```
program gaussplot

implicit none

real :: xnode,ynode,dummyr
integer :: elenum,mat,nn,a,b,c,d,e,f,g,h,m,num,ii,jj,count
integer :: nodenum,dummyi,increment,i,j,k,maxnodes,maxelem
integer,dimension (1:4) :: n,no
real,dimension (1:9) :: xn
real,dimension (1:9) :: yn
integer,dimension (1:1000,1:12) :: pair
logical :: exist(4)
integer,dimension (1:9) :: s

increment =0
mat=1
maxnodes=496    !Obtain from Ansys model
```

```
maxelem=450                !Obtain from Ansys model
nn=maxnodes


open(unit=12, file="oldelem", status="old") !obtained from ansys elem connectivity
open(unit=14, file="newnodes", status="old", position="rewind")
open(unit=15, file="newelem", status="old")


do 25 j=1,maxelem
 read(12,32) dummyi,dummyi,dummyi,dummyi,dummyi,dummyi,dummyi,dummyi, &
           dummyi,dummyi,elenum,n(1),n(2),n(3),n(4)
!  if ((n(1)==ready(pair,)) and (n(2)==ready(pair,1))

 do 70 i=1,4
  open(unit=13, file="oldnodes", status="old", position="rewind")        !obtained from Ansys nodal
coords
   do 72 k=1,maxnodes
    read(13,31) nodenum,dummyi,dummyi,xnode,ynode,dummyr,dummyr,dummyr,dummyr
    if (nodenum==n(i)) then
     xn(i)=xnode
     yn(i)=ynode
    endif
   72 continue
  close(unit=13)
 70 continue

 do 71 i=5,7
  xn(i)=(xn(i-4)+xn(i-3))/2
  yn(i)=(yn(i-4)+yn(i-3))/2
 71 continue
 xn(8)=(xn(1)+xn(4))/2
 yn(8)=(yn(1)+yn(4))/2

 xn(9)=(xn(1)+xn(2)+xn(3)+xn(4))/4
 yn(9)=(yn(1)+yn(2)+yn(3)+yn(4))/4

 !print *,xn(5),xn(6),xn(7),xn(8)
 !read *,num

! check must be made now
do 1001 i=1,4
 exist(i)=.false.
1001 continue

do 73 i=1,4
 if (i==4) then
  ii=1
  else
   ii=i+1
 endif
 do 103 jj=1,maxelem
   if ((pair(jj,1)==n(i)) .and. (pair(jj,2)==n(ii))) then
    exist(i)=.true.
           no(i)=pair(jj,3)
   endif
    if ((pair(jj,1)==n(ii)) .and. (pair(jj,2)==n(i))) then
      exist(i)=.true.
             no(i)=pair(jj,3)
    endif
```

```
 if ((pair(jj,4)==n(i)) .and. (pair(jj,5)==n(ii))) then
  exist(i)=.true.
           no(i)=pair(jj,6)
 endif
  if ((pair(jj,4)==n(ii)) .and. (pair(jj,5)==n(i))) then
    exist(i)=.true.
           no(i)=pair(jj,6)
  endif

 if ((pair(jj,7)==n(i)) .and. (pair(jj,8)==n(ii))) then
  exist(i)=.true.
           no(i)=pair(jj,9)
 endif
  if ((pair(jj,7)==n(ii)) .and. (pair(jj,8)==n(i))) then
    exist(i)=.true.
           no(i)=pair(jj,9)
  endif

 if ((pair(jj,10)==n(i)) .and. (pair(jj,11)==n(ii))) then
  exist(i)=.true.
           no(i)=pair(jj,12)
 endif
  if ((pair(jj,10)==n(ii)) .and. (pair(jj,11)==n(i))) then
    exist(i)=.true.
             no(i)=pair(jj,12)
  endif
103 continue
73 continue

do 730 i=1,4
 write(14,35) n(i),xn(i),yn(i)
730 continue

 if (exist(1)==0) then
  nn=nn+1
           no(1)=nn
  write(14,35) nn,xn(5),yn(5)
           else
            write(14,35) no(1),xn(5),yn(5)
 endif
 if (exist(2)==0) then
  nn=nn+1
           no(2)=nn
  write(14,35) nn,xn(6),yn(6)
           else
   write(14,35) no(2),xn(6),yn(6)
 endif
 if (exist(3)==0) then
  nn=nn+1
           no(3)=nn
  write(14,35) nn,xn(7),yn(7)
           else
            write(14,35) no(3),xn(7),yn(7)
 endif
 if (exist(4)==0) then
  nn=nn+1
           no(4)=nn
  write(14,35) nn,xn(8),yn(8)
           else
```

```
    write(14,35) no(4),xn(8),yn(8)
   endif

    nn=nn+1
    write(14,35) nn,xn(9),yn(9)

  do 78 i=5,8

            if (i==5) then
             pair(j,1)=n(1)
     pair(j,2)=n(2)
     pair(j,3)=no(1)
            endif
    if (i==6) then
             pair(j,4)=n(2)
     pair(j,5)=n(3)
     pair(j,6)=no(2)
            endif
    if (i==7) then
             pair(j,7)=n(3)
     pair(j,8)=n(4)
     pair(j,9)=no(3)
            endif
    if (i==8) then
             pair(j,10)=n(4)
     pair(j,11)=n(1)
     pair(j,12)=no(4)
            endif
 78 continue

25 continue
close(unit=14)

!count=maxelem*9
open(unit=14, file="newnodes", status="old", position="rewind")
do i=1,maxelem
 do count=1,9
  read(14,37) s(count)
 enddo
  write(15,36) s(1),s(5),s(9),s(8)
  write(15,36) s(5),s(2),s(6),s(9)
  write(15,36) s(9),s(6),s(3),s(7)
  write(15,36) s(8),s(9),s(7),s(4)
enddo

31 format(3i8,6e16.9)
41 format(3e16.9)
34 format(2i10,6f10.3)
32 format(19i7)
33 format(16i5)
35 format ('n,',i4,',',f10.4,',',f10.4)
36 format ('e,',i4,',',i4,',',i4,',',i4)
37 format (2x,i4)
end program gaussplot
```

APPENDIX A4 – FORTRAN program used for viewing stress contour plots in ANSYS

```
program gausssplot

implicit none

real :: a,b,c,d
integer :: i,maxelem,iter,j,k


!cdwrite,geom,filename,ext

i=0
maxelem=8                    !Obtain from Ansys model
iter=10

open(unit=13, file="gplot", status="old")
open(unit=12, file="sgansy", status="old")

do 5 k=0,iter
 write(13,*) 'iteration: ',k
i=0
 do 10 j=1,maxelem
  read(12,31) a,b,c,d
  i=i+1
  write(13,*) 'stgauss(',i,')=',a
  i=i+1
  write(13,*) 'stgauss(',i,')=',b
  i=i+1
  write(13,*) 'stgauss(',i,')=',c
  i=i+1
  write(13,*) 'stgauss(',i,')=',d
 10 continue
5 continue

31 format(4en16.4)


end program gausssplot
```

APPENDIX A5 – Sample APDL Ansys file for viewing stress contours

```
/prep7
/title, ndiv=4  iteration 0  elcogauss
et,1,plane42
mp,ex,1,207e3
n,   1,   10.0000,    0.0000
n,   3,   18.0000,    0.0000
n,   9,   16.2264,    7.4565
n,   8,    9.2388,    3.8268
n,  16,   14.0000,    0.0000
n,  17,   17.1132,    3.7283
```

n, 18, 12.7326, 5.6417
n, 19, 9.6194, 1.9134
n, 20, 13.3663, 2.8208
n, 3, 18.0000, 0.0000
n, 2, 50.0000, 0.0000
n, 5, 50.0000, 25.0000
n, 9, 16.2264, 7.4565
n, 21, 34.0000, 0.0000
n, 22, 50.0000, 12.5000
n, 23, 33.1132, 16.2283
n, 17, 17.1132, 3.7283
n, 24, 33.5566, 8.1141
n, 8, 9.2388, 3.8268
n, 9, 16.2264, 7.4565
n, 7, 13.2038, 13.2038
n, 6, 7.0711, 7.0711
n, 18, 12.7326, 5.6417
n, 25, 14.7151, 10.3301
n, 26, 10.1374, 10.1374
n, 27, 8.1549, 5.4490
n, 28, 11.4350, 7.8895
n, 9, 16.2264, 7.4565
n, 5, 50.0000, 25.0000
n, 4, 50.0000, 50.0000
n, 7, 13.2038, 13.2038
n, 23, 33.1132, 16.2283
n, 29, 50.0000, 37.5000
n, 30, 31.6019, 31.6019
n, 25, 14.7151, 10.3301
n, 31, 32.3576, 23.9151
n, 6, 7.0711, 7.0711
n, 7, 13.2038, 13.2038
n, 15, 7.4565, 16.2264
n, 14, 3.8268, 9.2388
n, 26, 10.1374, 10.1374
n, 32, 10.3301, 14.7151
n, 33, 5.6417, 12.7326
n, 34, 5.4490, 8.1549
n, 35, 7.8895, 11.4350
n, 7, 13.2038, 13.2038
n, 4, 50.0000, 50.0000
n, 11, 25.0000, 50.0000
n, 15, 7.4565, 16.2264
n, 30, 31.6019, 31.6019
n, 36, 37.5000, 50.0000
n, 37, 16.2283, 33.1132
n, 32, 10.3301, 14.7151
n, 38, 23.9151, 32.3576
n, 14, 3.8268, 9.2388
n, 15, 7.4565, 16.2264
n, 13, 0.0000, 18.0000
n, 12, 0.0000, 10.0000
n, 33, 5.6417, 12.7326
n, 39, 3.7283, 17.1132
n, 40, 0.0000, 14.0000
n, 41, 1.9134, 9.6194
n, 42, 2.8208, 13.3663
n, 15, 7.4565, 16.2264
n, 11, 25.0000, 50.0000
n, 10, 0.0000, 50.0000

```
n,  13,    0.0000,   18.0000
n,  37,   16.2283,   33.1132
n,  43,   12.5000,   50.0000
n,  44,    0.0000,   34.0000
n,  39,    3.7283,   17.1132
n,  45,    8.1141,   33.5566
e,   1,  16,  20,  19
e,  16,   3,  17,  20
e,  20,  17,   9,  18
e,  19,  20,  18,   8
e,   3,  21,  24,  17
e,  21,   2,  22,  24
e,  24,  22,   5,  23
e,  17,  24,  23,   9
e,   8,  18,  28,  27
e,  18,   9,  25,  28
e,  28,  25,   7,  26
e,  27,  28,  26,   6
e,   9,  23,  31,  25
e,  23,   5,  29,  31
e,  31,  29,   4,  30
e,  25,  31,  30,   7
e,   6,  26,  35,  34
e,  26,   7,  32,  35
e,  35,  32,  15,  33
e,  34,  35,  33,  14
e,   7,  30,  38,  32
e,  30,   4,  36,  38
e,  38,  36,  11,  37
e,  32,  38,  37,  15
e,  14,  33,  42,  41
e,  33,  15,  39,  42
e,  42,  39,  13,  40
e,  41,  42,  40,  12
e,  15,  37,  45,  39
e,  37,  11,  43,  45
e,  45,  43,  10,  44
e,  39,  45,  44,  13

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!CHANGE MAXELEM TO SUIT MODEL
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
maxelem=4*8
*dim,stgauss,,maxelem
/solu
solve
/post1
etable,gaustr,volu

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!INSERT GAUSS STRESS VALUES HERE FROM FILE 'GPLOT'
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
stgauss(          1)=   218.7544
 stgauss(          2)=   148.1189
 stgauss(          3)=   139.6788
 stgauss(          4)=   209.5525
 stgauss(          5)=   115.8249
 stgauss(          6)=   98.25230
 stgauss(          7)=   98.26340
 stgauss(          8)=   114.6127
```

```
 stgauss(         9)=   176.4737
 stgauss(        10)=   135.7063
 stgauss(        11)=   128.6696
 stgauss(        12)=   166.2536
 stgauss(        13)=   123.6831
 stgauss(        14)=   103.6043
 stgauss(        15)=   104.3010
 stgauss(        16)=   123.1548
 stgauss(        17)=   118.5638
 stgauss(        18)=   98.82230
 stgauss(        19)=   93.10940
 stgauss(        20)=   112.1199
 stgauss(        21)=   122.2709
 stgauss(        22)=   103.7729
 stgauss(        23)=   98.05620
 stgauss(        24)=   114.0659
 stgauss(        25)=   69.87940
 stgauss(        26)=   53.27060
 stgauss(        27)=   59.20920
 stgauss(        28)=   77.61290
 stgauss(        29)=   94.07790
 stgauss(        30)=   89.01460
 stgauss(        31)=   86.04170
 stgauss(        32)=   89.10200

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*vput,stgauss(1),elem,,etab,gaustr,,,0
/show,nplot4,grph
pletab,gaustr
```

APPENDIX A6 – Fortran program for performing elastic compensation using a bilinear continuous stiffness function across elements

```
PROGRAM FEA
!
!    CONTROL MAIN PROGRAM  --  FEA.FOR -- PLANE STRESS
!

    DOUBLE PRECISION CO,S,DISP,PLOAD,TH,V,SMAX,YM,LOWLMULT,Z,SM
    INTEGER ELEM,NNODE,NDOF,NELEM,NN,ANA,NITER,ITER
    CHARACTER*1 ANS
    DIMENSION CO(500,2),NN(500,8),SMAX(500)
    DIMENSION S(1000,1000),DISP(1000),PLOAD(1000)
                        DOUBLE PRECISION, DIMENSION (0:10) :: SHAKE
                        DOUBLE PRECISION, DIMENSION(1:500,10:13) :: VMISES
                        DOUBLE PRECISION, DIMENSION(1:500,1:4) :: VMGAUSS
                        DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D
                        DOUBLE PRECISION, DIMENSION(0:10,1:500,1:4) :: E

                        DOUBLE PRECISION, DIMENSION(1:500) :: AVSTRESS

                        SM=300 ! YIELD STRESS
                        NITER=10        ! NUMBER OF ITERATIONS (ALSO CHANGE UPPER
                       !LIMIT OF [SHAKE] = NITER
                                ! & DIMENSIONS OF [E]
                                ! YOU CAN USE ALLOCATABLE ARRAY SIZE HERE
                                ! niter=0 for elastic analyses
```

```
      PRINT *
      PRINT *,'HAVE A NICE DAY'

!     TYPES OF ELEMENTS

10    PRINT *,'INPUT TYPES OF ELEMENTS'
      PRINT *
      PRINT *,'1 - FOR 2D-4 NODE QUADRILATERAL'
      PRINT *,'2 - FOR 2D-8 NODE QUADRILATERAL'
PRINT *,'13 - TO QUIT THIS PROGRAM'
      PRINT *
      PRINT *
!
ITER = 0                    !ELASTIC SOLUTION

READ *,ELEM
                            PRINT *
                            PRINT *

!                           DEFINE INITIAL ELASTIC CONSTANTS AND THICKNESS FOR PLANE
                           !STRESS
!
      PRINT *,'INPUT YOUNGS MODULUS, POISSON RATIO, THICKNESS'
      PRINT *

      READ *,YM,V,TH
      PRINT *,YM,V,TH
                            READ *,NUMBER

!                           BUILD GEOMETRY
      PRINT *
      PRINT *
      PRINT *,'INPUT THE MAXIMUM NODE NUMBER'
      PRINT *
      PRINT *
      READ *,NNODE
      PRINT *
      PRINT *
      PRINT *,'INPUT THE TOTAL NUMBER OF ELEMENTS'
      PRINT *
      PRINT *
      READ *,NELEM
      PRINT *
      PRINT *

      CALL COORD (NNODE,CO)

      PRINT *

      NDOF=NNODE*2
      PRINT *,'NDOF=',NDOF


!                           DEFINE ELEMENT BY NODE NUMBERS
      CALL ELEMCON1 (NELEM,NN,ELEM)

DO 2000 ITER=0,NITER

          CALL PSTRESS (D,TH,NELEM,YM,V,ITER,SMAX,SM,VMGAUSS,VMISES,nnode, &
                        nn,avstress)   !MODIFY YOUNG'S MODULUS & [D] MATRIX
```

```
!PRINT *, 'SO FAR'
!READ *,NUMBER

PRINT *
PRINT *,' ASSEMBLY OF GLOBAL STIFFNESS MATRIX'
!            FORMULATE GLOBAL STIFFNESS MATRIX
222   CALL STIFFNESS1 (NDOF,NELEM,NN,ELEM,CO,D,TH,S)

!APPLY BOUNDARY CONDITIONS TO YOUR MODEL DURING ITERATION ZERO

CALL DISPLOAD (NDOF,S,PLOAD,ITER)

330   PRINT *
      PRINT *,' SOLUTION PHASE'
      PRINT *

      CALL SOLUTION1 (NDOF,S,PLOAD,DISP)

      PRINT *
      PRINT *,'  PRINTING OF DISPLACEMENTS'
      PRINT *

      CALL PRDISP (NNODE,DISP)

      PRINT *
      PRINT *,' PRINTING OF STRESSES'
      PRINT *

      CALL PRSTRESS (NELEM,NDOF,NN,ELEM,CO,D,DISP,TH,VMISES, &
                     VMGAUSS,iter,avstress,nnode)

!FIND LARGEST VMISES STRESS (AT NODES) FOR EACH ELEMENT AND STORE IN ARRAY
                     !SMAX
DO COUNT = 1,NELEM
SMAX(COUNT)=MAX(VMises(COUNT,10),VMises(COUNT,11),VMises(COUNT,12),VMises(COU
                     NT,13))
ENDDO

SHAKE(ITER) = MAXVAL(SMAX)          !STORE MAX VMISES FROM WHOLE MODEL
PRINT *, SHAKE(ITER)
write (10001,*) 'shake(iter)',shake(iter)

2000  CONTINUE
Z=MINVAL(SHAKE)

LOWLMULT=SM/Z      !CALCULATE BEST LIMIT LOAD MULTIPLIER

DO 999 I = 0,NITER
PRINT *, SHAKE(I)
999 ENDDO

 PRINT *
 PRINT *, 'LOWER LIMIT LOAD MULTIPLIER IS : ', LOWLMULT
 PRINT *

990   PRINT *
      PRINT *,'DO YOU WANT TO START AGAIN? (Y FOR YES, N FOR NO)'
      PRINT *
      READ (5,995) ANS
      IF (ANS.EQ.'Y' .OR. ANS.EQ.'y') THEN
```

```
      GOTO 10
      ELSEIF (ANS.EQ.'N' .OR. ANS.EQ.'n') THEN
      GOTO 1000
      ENDIF
995   FORMAT (A1)
1000  END

!     ---------------------------------------------------------------
!     TO READ IN ALL NODAL CO-ORDINATES
!     ---------------------------------------------------------------

      SUBROUTINE COORD (NNODE,CO)
      DOUBLE PRECISION CO
      DIMENSION CO(500,2)
 CHARACTER(15) :: FILENAME
                  INTEGER :: OPENSTATUS, INPUTSTATUS

                  PRINT 200
                  READ *,FILENAME
                  OPEN (UNIT = 10, FILE = FILENAME, STATUS = "OLD", &
                  POSITION="REWIND",ACTION = "READ", IOSTAT = OPENSTATUS)
                  IF (OPENSTATUS > 0) STOP " *** CANNOT OPEN FILE ***"

DO 100 I=1,NNODE

READ (UNIT = 10, FMT = *, IOSTAT = INPUTSTATUS) CO(I,1),CO(I,2)

100    CONTINUE

200      FORMAT (' ENTER FILENAME TO READ NODAL CO-ORDINATES',A15)

CLOSE (10)

      PRINT *
      PRINT *,' NODE NUMBER      X-COORDINATES       Y-COORDINATES'
      PRINT *
      DO 300 I=1,NNODE
      PRINT 150,I,CO(I,1),CO(I,2)
                                IF (I.EQ.20 .OR. I.EQ.40 .OR. I.EQ.60) THEN
      READ *,NUMBER
       ENDIF
150   FORMAT (2X,I5,2(12X,F10.4))
300   CONTINUE
      PRINT *
      PRINT *

      RETURN
      END

!     ---------------------------------------------------------------
!     TO READ IN NODE NUMBER CONNECTING EACH ELEMENT (TYPE NO:1,2)
!     ---------------------------------------------------------------
!        REMEMBER :: TO CREATE ELEMENTS ENTER NODE NUMBER IN ANTI-CLOCKWISE
        !SENSE

      SUBROUTINE ELEMCON1 (NELEM,NN,ELEM)
      INTEGER NELEM,ELEM,NN
      DIMENSION NN(500,8)
       CHARACTER(15) :: FILENAME
       INTEGER :: OPENSTATUS, INPUTSTATUS
```

```
      PRINT 310
      READ *,FILENAME
310   FORMAT (' ENTER FILENAME TO READ ELEMCON',A15)

 OPEN (UNIT = 15, FILE = FILENAME, STATUS = "OLD", &
                              POSITION="REWIND",ACTION = "READ", IOSTAT = OPENSTATUS)
                             IF (OPENSTATUS > 0) STOP " *** CANNOT OPEN FILE ***"

 IF (ELEM.EQ.1) THEN

  DO 100 I=1,NELEM

         READ (UNIT = 15, FMT = *, IOSTAT = INPUTSTATUS) &
                NN(I,1),NN(I,2),NN(I,3),NN(I,4)

100    CONTINUE

    PRINT *
    PRINT *,' ELEMENT NUMBER     NODE-I    NODE-J    NODE-K    NODE-L'
    PRINT *
    PRINT *
    DO 300 I=1,NELEM
    PRINT 250,I,NN(I,1),NN(I,2),NN(I,3),NN(I,4)
 IF (I.EQ.20 .OR. I.EQ.40 .OR. I.EQ.60) THEN
    READ *,NUMBER
 ENDIF

250   FORMAT (6X,I3,13X,I3,3(7X,I3))

300   CONTINUE
    PRINT *
    PRINT *

    ELSEIF (ELEM.EQ.2) THEN

    DO 101 I=1,NELEM

                             READ (UNIT = 15, FMT = *, IOSTAT = INPUTSTATUS) &
                             NN(I,1),NN(I,2),NN(I,3),NN(I,4),NN(I,5),NN(I,6),NN(I,7),NN(I,8)
101   CONTINUE

    PRINT *
    PRINT *,' ELEM NUM  N-I  N-J  N-K  N-L  N-M  N-N  N-P  N-Q'
    PRINT *
    PRINT *
    DO 301 I=1,NELEM
    PRINT 251,I,NN(I,1),NN(I,2),NN(I,3),NN(I,4),NN(I,5),NN(I,6),NN(I,7 &
    ),NN(I,8)
251   FORMAT (3X,I3,6X,7(I3,2X),I3)

301   CONTINUE
    PRINT *
    PRINT *

    ENDIF

                            CLOSE (15)

    RETURN
```

```
      END

!     ----------------------------------------------------------------
!     TO CALCULATE THE D-MATRIX FOR 2D - PLANE STRESS
!     ----------------------------------------------------------------

      SUBROUTINE PSTRESS (D,TH,NELEM,YM,V,ITER,SMAX,SM,VMGAUSS,VMISES,&
                                  nnode,nn,avstress)
      DIMENSION SMAX(500)
      DOUBLE PRECISION TH,SMAX,SM,YM,V,nni,nnj,nnk,nnl,xi,ni
                        DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D
                        DOUBLE PRECISION, DIMENSION(0:10,1:500,1:4) :: E
                        DOUBLE PRECISION, DIMENSION(0:10,1:500) :: Exy
                        DOUBLE PRECISION, DIMENSION(1:500,1:4) :: VMGAUSS
                        DOUBLE PRECISION, DIMENSION(1:500,10:13) :: VMISES
                        INTEGER :: ENUM,ITER,G,nn,node
                        DIMENSION NN(500,8)
                        DOUBLE PRECISION, DIMENSION(1:500) :: AVSTRESS
                        DOUBLE PRECISION, DIMENSION(1:4) :: von

                        open (unit=9999, file="Efile",status="old")

!     ZERO ALL ELEMENTS IN D MATRIX
                        DO 25 K=1,NELEM
      DO 20 I=1,3
      DO 10 J=1,3
                        DO 29 G=1,4
      D(I,J,K,G)=0

29                      ENDDO
10    CONTINUE
20    CONTINUE
25     CONTINUE
 IF (ITER == 0) THEN
write (9999,*) 'E - Used for Elastic solution'
DO I = 1,NELEM
DO 3001 G = 1,4
 E(ITER,I,G)=YM
3001ENDDO
write(9999,700) E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i
print 700, E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i

do node=1,nnode
exy(iter,node)=YM
enddo

  ENDDO

ENDIF

! TO CHOOSE WHICH STRESS TO USE TO MODIFY [D]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!GAUSS STRESSES here
!!!                           DO 4001 G = 1,4
!!                             E(ITER,I,G)=E(ITER-1,I,G)*SM/VMGAUSS(I,G)
!                            4001      ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! NODAL STRESSES here
!                             DO 4001 G = 1,4
!                              E(ITER,I,G)=E(ITER-1,I,G)*SM/VMISES(I,(G+9))
!                            4001      ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!! use below for standard elastic compensation to check fortran coding
!DO 4001 G = 1,4
!E(ITER,I,G)=E(ITER-1,I,G)*SM/smax(i)
!4001                          ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!write(9999,700) E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i
!print 700, E(ITER,I,1),E(ITER,I,2),E(ITER,I,3),E(ITER,I,4),i
!4000   ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! END CHANGE

   IF (ITER > 0) THEN !Interpolate Young's modulus value from nodes to gauss points
     do node=1,nnode
       exy(iter,node)=exy(iter-1,node)*sm/avstress(node)
      write(9999,*) Exy(ITER,node),node
     enddo

     do 11 enum=1,nelem
     i=nn(enum,1)
     j=nn(enum,2)
     k=nn(enum,3)
     l=nn(enum,4)
     do g=1,4
     xi=0.57735
     ni=xi
     if (g.eq.1) then
     xi=-xi
     ni=-ni
     elseif (g.eq.2) then
     ni=-ni
     elseif (g.eq.4) then
     xi=-xi
     endif
     !bilinear interpolation functions
       nni=0.25*(1-xi)*(1-ni)
       nnj=0.25*(1+xi)*(1-ni)
       nnk=0.25*(1+xi)*(1+ni)
       nnl=0.25*(1-xi)*(1+ni)

von(g)=(nni*avstress(i))+(nnj*avstress(j))+(nnk*avstress(k))+(nnl*avstress(l))

 E(ITER,enum,g)=(nni*exy(iter,i))+(nnj*exy(iter,j))+(nnk*exy(iter,k))+(nnl*exy(iter,l))
    enddo

!read *, number

!                               DO 4001 G = 1,4
!                                 E(ITER,enum,G)=E(ITER-1,enum,G)*SM/von(G)
!                               4001      ENDDO
!write(9999,700) E(ITER,enum,1),E(ITER,enum,2),E(ITER,enum,3),E(ITER,enum,4),enum
 11 enddo

ENDIF

!CALCULATE D MATRIX FOR EACH GAUSS POINT IN EACH ELEMENT
      DO 27 ENUM=1,NELEM
       DO 28 G = 1,4
        D(1,1,ENUM,G)=E(ITER,ENUM,G)/(1.-V*V)
        D(2,2,ENUM,G)=D(1,1,ENUM,G)
        D(1,2,ENUM,G)=V*D(1,1,ENUM,G)
        D(2,1,ENUM,G)=D(1,2,ENUM,G)
```

```
      D(3,3,ENUM,G)=.5*(1.-V)*D(1,1,ENUM,G)
28  CONTINUE
27  CONTINUE

    DO 41 I=1,3

   PRINT 30,D(I,1,1,1),D(I,2,1,1),D(I,3,1,1) !PRINTING [D] FOR IST GAUSS PT OF IST !ELEMENT
30   FORMAT (D10.4,2X,D10.4,2X,D10.4)
41   CONTINUE

PRINT *
 PRINT *
700 format (4En14.4,2x,i3)
    RETURN
    END

!    ---------------------------------------------------------------
!    FORM GLOBAL STIFFNESS MATRIX
!    ---------------------------------------------------------------

    SUBROUTINE STIFFNESS1 (NDOF,NELEM,NN,ELEM,CO,D,TH,S)
    INTEGER NDOF,NELEM,NN,ELEM,II
    DOUBLE PRECISION DJ,EM,C,T,S,CO,S3,A,B,TH
    DIMENSION EM(16,16),C(20,20),T(20,20),S(1000,1000)
    DIMENSION NN(500,8),CO(500,2)
     DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D

!                           ZERO S MATRIX
    DO 20 I=1,NDOF
    DO 10 J=1,NDOF
    S(I,J)=0.                ! S IS THE GLOBAL STIFFNESS MATRIX
10   CONTINUE
20   CONTINUE

    S3=0.577350269

    DO 150 N=1,NELEM  !ELEMENT NO. IS TAKEN FROM HERE

! ZERO EM MATRIX
! em of previous element is already stored in global matrix
    DO 40 I=1,8*ELEM
    DO 30 J=1,8*ELEM
    EM(I,J)=0.
30   CONTINUE
40   CONTINUE


!II is used to calculate K for the 4 Gauss points
    DO 100 II=1,4

    A=S3
    B=S3
    IF (II.EQ.1) THEN
    A=-A
    B=-B
    ELSEIF (II.EQ.2) THEN
    B=-B
    ELSEIF (II.EQ.4) THEN
    A=-A
    ENDIF
```

```
      CALL MATRIX1 (NN,N,ELEM,D,A,B,CO,DJ,T,C,II)

      DJ=ABS(DJ)
! form k for each Gauss point in subroutine MATRIX1 and add to integrate
      DO 90 I=1,8*ELEM
      DO 80 J=1,8*ELEM
      DO 70 K=1,3
      EM(I,J)=EM(I,J)+C(K,I)*T(K,J)*DJ*TH
70    CONTINUE
80    CONTINUE
90    CONTINUE
100   CONTINUE
! REMOVE ! BELOW TO VIEW MATRICES
!      print *,'element no     em matrix '
!      PRINT *
!      DO 105 I=1,8*ELEM
!      print 85,n,em(i,1),em(i,2),em(i,3),em(i,4),em(i,5),em(i,6),em(i,7) &
!     ,em(i,8)
85    FORMAT (I3,8f9.2)
!105   CONTINUE
!      READ *,NUMBER

      IF (ELEM.EQ.2) THEN
      DO 110 I=1,8*ELEM
      PRINT 85,N,EM(I,9),EM(I,10),EM(I,11),EM(I,12),EM(I,13),EM(I,14) &
      ,EM(I,15),EM(I,16)
110   CONTINUE
      !read *,number
      ENDIF
! UP TO HERE

!
!     DO LOOP 140 DOES THE ASSEMBLY OF ELEMENT STIFFNESS MATRIX TO THE
!     GLOBAL STIFFNESS MATRIX
!

      DO 140 I=1,4*ELEM
      K=NN(N,I)
      II1=2*I-1
      II2=2*I
      KI1=2*K-1
      KI2=2*K
!      IN=NN(N,I)
      DO 130 J=1,4*ELEM
      L=NN(N,J)
      JJ1=2*J-1
      JJ2=2*J
      LJ1=2*L-1
      LJ2=2*L
!      JN=NN(N,J)
!       DO 125 IL=1,2
!        IE=(I-1)*2+IL
!        NR=(IN-1)*2+IL
!          DO 124 JL=1,2
!           JE=(I-1)*2+JL
!           NC=(JN-1)*2+JL
!           S(NR,NC)=S(NR,NC)+EM(IE,JE)
!124        CONTINUE
!125     CONTINUE
```

```
      S(KI1,LJ1)=S(KI1,LJ1)+EM(II1,JJ1)
      S(KI1,LJ2)=S(KI1,LJ2)+EM(II1,JJ2)
      S(KI2,LJ1)=S(KI2,LJ1)+EM(II2,JJ1)
      S(KI2,LJ2)=S(KI2,LJ2)+EM(II2,JJ2)

130   CONTINUE
140   CONTINUE

150   CONTINUE

!     PRINT *,'MATRIX OF S'
!     DO 160 I=1,NDOF
!     PRINT 155,S(I,1),S(I,2),S(I,3),S(I,4),S(I,5),S(I,6),S(I,7),S(I,8)
!    +,S(I,9)
!160   CONTINUE
!     read *,number
!     DO 170 I=1,NDOF
!     PRINT 155,S(I,10),S(I,11),S(I,12),S(I,13),S(I,14),S(I,15)
!    + ,S(I,16),S(I,17),S(I,18)
!170   CONTINUE
!155   FORMAT (9f8.2)
!     read *,number
      RETURN
      END

!     ---------------------------------------------------------------
!     TO GENERATE  INTEGRATION OF [D].[B]  FOR ELEMENT 1
!     ---------------------------------------------------------------

      SUBROUTINE MATRIX1 (NN,N,ELEM,D,A,B,CO,DJ,T,C,II)
      DOUBLE PRECISION DKL,TJ,DJ,XY,CO,T,C,A,B,DD
      INTEGER NN,N,K,ELEM,II,number
      DIMENSION CO(500,2),NN(500,8),XY(20,2),DKL(2,8),TJ(2,2),T(20,20) &
      ,C(20,20)
                              DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D

!
!     THE DO-LOOP 20 DOES THE ASSIGNING OF COORDINATES TO APPROPRIATE
!     NODES WITHIN A SINGLE ELEMENT
!

!     PRINT *,'ELEM=',ELEM
      DO 20 I=1,4*ELEM
      K=NN(N,I)
      XY(I,1)=CO(K,1)
      XY(I,2)=CO(K,2)
20    CONTINUE

      IF (ELEM.EQ.1) THEN

      DKL(1,1)=(B-1.)/4.
      DKL(1,2)=(1.-B)/4.
      DKL(1,3)=(1.+B)/4.
      DKL(1,4)=-(1.+B)/4.
      DKL(2,1)=(A-1.)/4.
      DKL(2,2)=-(1.+A)/4.
      DKL(2,3)=(1.+A)/4.
      DKL(2,4)=(1.-A)/4.

      ELSEIF (ELEM.EQ.2) THEN
```

```
      DKL(1,1)=((B+2.*A)*(1.-B))/4.
      DKL(1,2)=A*(B-1.)
      DKL(1,3)=((B-2.*A)*(B-1.))/4.
      DKL(1,4)=(1.-B*B)/2.
      DKL(1,5)=((B+2.*A)*(1.+B))/4.
      DKL(1,6)=-A*(1.+B)
      DKL(1,7)=((2.*A-B)*(1.+B))/4.
      DKL(1,8)=-(1.-B*B)/2.
      DKL(2,1)=((A+2.*B)*(1.-A))/4.
      DKL(2,2)=-(1.-A*A)/2.
      DKL(2,3)=((1.+A)*(2.*B-A))/4.
      DKL(2,4)=-B*(1.+A)
      DKL(2,5)=((1.+A)*(2.*B+A))/4.
      DKL(2,6)=(1.-A*A)/2.
      DKL(2,7)=((1.-A)*(2.*B-A))/4.
      DKL(2,8)=B*(A-1.)

      ENDIF
!                             set up Jacobian matrix mine
      DO 50 I=1,2
       DO 40 J=1,2
       TJ(I,J)=0.
                                DO 30 K=1,4*ELEM
                                  TJ(I,J)=TJ(I,J)+DKL(I,K)*XY(K,J)
 30      CONTINUE
 40     CONTINUE
 50    CONTINUE
!                             determinant of Jacobian mine
      DJ=TJ(1,1)*TJ(2,2)-TJ(1,2)*TJ(2,1)
      DD=TJ(1,1)
!                             inverse of jacobian mine
      TJ(1,1)=TJ(2,2)/DJ
      TJ(2,2)=DD/DJ
      TJ(1,2)=-TJ(1,2)/DJ
      TJ(2,1)=-TJ(2,1)/DJ

      DO 90 I=1,2
       DO 80 J=1,4*ELEM
                               T(I,J)=0.
                                DO 70 K=1,2
                                  T(I,J)=T(I,J)+TJ(I,K)*DKL(K,J)
 70      CONTINUE
 80     CONTINUE
 90    CONTINUE

      DO 100 J=1,4*ELEM
                              C(1,2*J-1)=T(1,J)
                              C(3,2*J)=T(1,J)
                              C(2,2*J)=T(2,J)
                              C(3,2*J-1)=T(2,J)
 100  CONTINUE
!                             work out D.B mine /stored in [T]
      DO 150 I=1,3
       DO 130 J=1,8*ELEM
                               T(I,J)=0.
                                DO 120 K=1,3
                                 T(I,J)=T(I,J)+D(I,K,N,II)*C(K,J)
 120      CONTINUE
 130   CONTINUE
```

```
 150  CONTINUE

      RETURN
      END

!     ------------------------------------------------------------
!     INPUT NODAL DISPLACEMENTS AND FORCES
!     ------------------------------------------------------------

      SUBROUTINE DISPLOAD (NDOF,S,PLOAD,ITER)

      INTEGER NUM,INUM,JNUM,A,DIR,INPUTSTATUS,OPENSTATUS,ITER
      DOUBLE PRECISION VALUE,S,PLOAD
      CHARACTER*1 ANS, FILELOAD*15
                              REAL :: P

      DIMENSION S(1000,1000),PLOAD(1000)

      DO 10 I=1,NDOF
       PLOAD(I)=0.
10    CONTINUE

      P=1E30

       IF (ITER == 0) THEN
        PRINT 410
        READ *,FILELOAD
410   FORMAT (' ENTER FILENAME TO READ LOADS & B.C.',A15)
                              ENDIF


    OPEN (UNIT = 25, FILE = FILELOAD, STATUS = "OLD", &
                                 POSITION="REWIND",ACTION = "READ", IOSTAT = OPENSTATUS)
                              IF (OPENSTATUS > 0) STOP " *** CANNOT OPEN FILE ***"

                              DO 56

                              READ (UNIT = 25, FMT = *, IOSTAT = INPUTSTATUS) &
                                       A,NUM,DIR,VALUE
                              IF (INPUTSTATUS < 0) EXIT  ! END OF FILE

      PRINT *,A,NUM,DIR,VALUE

      IF (A.EQ.1) THEN

                              IF (DIR.EQ.1) THEN
                              INUM=1
                              ELSEIF (DIR.EQ.2) THEN
                              INUM=0
                              ENDIF

                              JNUM=2*NUM-INUM
                              PLOAD(JNUM)=VALUE
                              PRINT *
                              PRINT *,'FORCE AT NODE NUMBER',NUM
                              PRINT *,'IN DIRECTION',DIR
                              PRINT *,'IS OF VALUE',PLOAD(JNUM)
                              PRINT *

      ELSEIF (A.EQ.2) THEN
```

```
                        IF (DIR.EQ.1) THEN
                        INUM=1
                        JNUM=2*NUM-INUM
                        S(JNUM,JNUM)=S(JNUM,JNUM)*P
                        PLOAD(JNUM)=S(JNUM,JNUM)*VALUE


                        ELSEIF (DIR.EQ.2) THEN
                        JNUM=2*NUM
                        S(JNUM,JNUM)=S(JNUM,JNUM)*P
                        PLOAD(JNUM)=S(JNUM,JNUM)*VALUE


                        ELSEIF (DIR.EQ.12) THEN
                        DO 50 I=1,2
                        INUM=I-1
                        JNUM=2*NUM-INUM
                        S(JNUM,JNUM)=S(JNUM,JNUM)*P
                        PLOAD(JNUM)=S(JNUM,JNUM)*VALUE
50      CONTINUE

                        ENDIF


      ENDIF

56 end do
close (25)

      RETURN
      END

!     ---------------------------------------------------------------
!     SOLUTION OF SIMULTANEOUS EQUATIONS BY GAUSS ELIMINATION METHOD
!
!     CODING FROM "COMPUTATIONAL METHODS FOR THE SOLUTION OF ENGINEERING
!     PROBLEMS" BY C.A.BREBBIA AND A.J.FERRANTE
!     THIRD REVISED EDITION, PENTECH PRESS   -- PAGE 42
!     ---------------------------------------------------------------

      SUBROUTINE SOLUTION1 (NDOF,S,PLOAD,DISP)

      INTEGER NDOF
      DOUBLE PRECISION S,PLOAD,DISP,C,CALOAD
!     DOUBLE PRECISION SS
      DIMENSION S(1000,1000),PLOAD(1000),DISP(1000)
!     DIMENSION ss(100,100)


      do 99 i=1,ndof
!      do 98 j=1,ndof
!      ss(i,j)=s(i,j)
!98    continue
      DISP(I)=0.
99    continue

!     DO 30 I=1,NDOF
!     PRINT *,S(I,I)
!30    CONTINUE
```

```
!     READ *,NUMBER

      N1=NDOF-1
      DO 100 K=1,N1
      C=S(K,K)
      K1=K+1
      IF (ABS(C) .LT.  0.0000000001) THEN
                              DO 7 J=K1,NDOF
!
!     TRY TO INTERCHANGE ROWS TO GET NON ZERO DIAGONAL COEFFICIENT
!
                             IF (ABS(S(J,K)) .GT. 0.0000000001) THEN
                               DO 6 L=K,NDOF
                                 C=S(K,L)
                                 S(K,L)=S(J,L)
                                 S(J,L)=C
6           CONTINUE
                             C=PLOAD(K)
                             PLOAD(K)=PLOAD(J)
                             PLOAD(J)=C
                             C=S(K,K)
                             PRINT *, ' ROW INTERCHANGED ONCE',S(K,K)
                             GOTO 3
                            ENDIF
                            PRINT *,'  A LOOP ON J'
7       CONTINUE
1     PRINT 2, K
2     FORMAT ('****** SINGULARITY IN ROW',I5)
      GOTO 300
      ENDIF
!
!     DIVIDE ROW BY DIAGONAL COEFFICIENT
!

3     C=S(K,K)
      DO 4 J=K1,NDOF
!     IJ=IJ+1
!     IF (IJ.EQ.20 .OR. IJ.EQ.40 .OR. IJ.EQ.60) THEN
!     READ *,NUMBER
!     ENDIF
!     PRINT *,' S(K,J) AND C',K,J,S(K,J),C
      S(K,J)=S(K,J)/C
4     CONTINUE
      PLOAD(K)=PLOAD(K)/C

!
!     ELIMINATION OF UNKNOWNS FROM EACH ROW
!

      DO 10 I=K1,NDOF
      C=S(I,K)
      DO 9 J=K1,NDOF
      S(I,J)=S(I,J)-C*S(K,J)
9     CONTINUE
      PLOAD(I)=PLOAD(I)-C*PLOAD(K)
10    CONTINUE
100   CONTINUE

      IF (ABS(S(NDOF,NDOF)) .LT. 0.000001) THEN
      PRINT 2,NDOF
```

```
      GOTO 300
      ENDIF

!
!     COMPUTE LAST UNKNOWN
!
101   DISP(NDOF)=PLOAD(NDOF)/S(NDOF,NDOF)

!
!     BACK SUBSTITUTION PROCESS TO COMPUTE REMAINING UNKNOWNS
!
      DO 200 L=1,N1
      K=NDOF-L
      K1=K+1
      DO 150 J=K1,NDOF
      DISP(K)=DISP(K)+ PLOAD(K)-S(K,J)*DISP(J)
      PLOAD(K)=0.
150   CONTINUE
200   CONTINUE

!     DO 250 I=1,NDOF
!     PRINT *,' DISPLACEMENT AT ',I,'  =  ',DISP(I)
!250   CONTINUE

!     READ *,NUMBER

!     print *,'  i     pload(i)'
!     do 199 i=1,ndof
!     pload(i)=0.
!     do 198 j=1,ndof
!     pload(i)=pload(i)+ss(i,j)*disp(j)
!198   continue
!     print *,i, pload(i)
!199   continue
!     read *,number

300   RETURN
      END

!     ---------------------------------------------------------------
!     TO PRINT X AND Y DISPLACEMENT
!     ---------------------------------------------------------------

      SUBROUTINE PRDISP (NNODE,DISP)

      INTEGER NNODE
      DOUBLE PRECISION DISP
      DIMENSION DISP(1000)

      PRINT *
      PRINT *, ' THE FOLLOWING X AND Y DISPLACEMENT ARE IN GLOBAL COORD'
      PRINT *
      PRINT *, ' NODE                 UX                  UY'
      PRINT *
      PRINT *
      DO 50 I=1,NNODE
      I1=2*I-1
      I2=2*I
      PRINT 30, I,DISP(I1),DISP(I2)
      IF (I.EQ.20 .OR. I.EQ.40 .OR. I.EQ.60) THEN
```

```
!    READ *,NUMBER
     ENDIF
30   FORMAT (I3,10X,D18.8,5X,D18.8)
50   CONTINUE
!    READ *,NUMBER
     RETURN
     END

!    ---------------------------------------------------------------
!    TO PRINT ELEMENT STRESS AT FOUR GAUSS POINT AND CENTROID
!    ---------------------------------------------------------------

     SUBROUTINE PRSTRESS (NELEM,NDOF,NN,ELEM,CO,D,DISP,TH,VMISESD,&
                                    VMGAUSSD,iter,avstress,nnode)

     DOUBLE PRECISION CO,DISP,ED,T,ES,A,B,DJ,C,C1,C2,C3,C4,C5,TH,GSX,GSY,GSXY
     INTEGER NELEM,NDOF,NN,ELEM,N,II,number
     DIMENSION CO(500,2),DISP(1000),NN(500,8),ED(20),T(20,20)
                              DOUBLE PRECISION, DIMENSION(10:13) :: SX,SY,SXY
                              DOUBLE PRECISION, DIMENSION(1:500,10:13), intent(out) :: VMISESD
                             !VMISES AT NODES
                              DOUBLE PRECISION, DIMENSION(1:500,1:4), INTENT(OUT) ::
                             VMGAUSSD  !VMISES AT GAUSS POINTS
                              DOUBLE PRECISION, DIMENSION(1:6,1:6,1:500,1:4) :: D
                              DOUBLE PRECISION, DIMENSION(1:500) :: AVSTRESS

     DIMENSION ES(17,3),C(20,20)

        open (unit=10000, file="stressG",status="old")
        open (unit=10001, file="stressN",status="old")
        open (unit=10002, file="sgansy",status="old")
        open (unit=10003, file="snansy",status="old")
        open (unit=991, file="avstress",status="old")

!    TO DETERMINE NUMBER OF EXTRAPOLATED STRESSES REQUIRED
!
     IF (ELEM.EQ.2) THEN
     IEXTRA=8
     ELSEIF (ELEM.EQ.1) THEN
     IEXTRA=4
     ENDIF

     PRINT *
     PRINT *,'THICKNESS = ',TH
     PRINT *
     PRINT *
     PRINT *,'STRESS 1-4 ARE UTIMATE STRESSES LOCATION +/- 0.57735'
     PRINT *,'STRESS 5 IS CENTROIDAL STRESS'
     PRINT *,'STRESS 6-9 ARE STRESSES AT CORNER NODES'
     PRINT *,'STRESS 10-13 ARE EXTRAPOLATED STRESSES AT CORNER NODES'
     PRINT *,'STRESS 14-17 ARE EXTRAPOLATED STRESSES AT MID-SIDE NODES'
     PRINT *

     DO 100 N=1,NELEM              !TAKES ELEMENT NO. FROM HERE

     PRINT *,' ELEM    G.P.    SIGMA-X        SIGMA-Y        SIGMA-XY'


DO 30 I=1,4*ELEM
            K=NN(N,I)
```

```
            ED(2*I-1)=DISP(2*K-1)
            ED(2*I)  =DISP(2*K)
30          CONTINUE

DO 95 II=1,4

            A=0.577350269
            B=A
            IF (II.EQ.1) THEN
            A=-A
            B=-B
            ELSEIF (II.EQ.2) THEN
            B=-B
            ELSEIF (II.EQ.4) THEN
            A=-A
            ELSEIF (II.EQ.5) THEN
            A=0.
            B=0.
            ELSEIF (II.EQ.6) THEN
            A=-1.
            B=-1.
            ELSEIF (II.EQ.7) THEN
            A=1.
            B=-1.
            ELSEIF (II.EQ.8) THEN
            A=1.
            B=1.
            ELSEIF (II.EQ.9) THEN
            A=-1.
            B=1.
            ENDIF


 CALL MATRIX1 (NN,N,ELEM,D,A,B,CO,DJ,T,C,II) !WORK OUT DB FOR EACH ELEMENT

DO 80 I=1,3
   ES(II,I)=0.
    DO 70 J=1,8*ELEM
      ES(II,I)=ES(II,I)+T(I,J)*ED(J)

    70 CONTINUE
80 CONTINUE

!WORK OUT V.MISES STRESS AT EACH GAUSS POINT
            GSX=(ES(II,1))
            GSY=(ES(II,2))
            GSXY=(ES(II,3))


IF (II <= 4) THEN
 VMGAUSSD(N,II)=SQRT((GSX**2) + (GSY**2) - (GSX*GSY) + 3*(GSXY**2))
ENDIF

  PRINT 85,N,II,ES(II,1),ES(II,2),ES(II,3)
85          FORMAT (I3,5X,I3,3X,F15.5,2(2X,F15.5))

95       CONTINUE
```

```
            C1=1.866
            C2=0.5
            C3=0.133975
            C4=0.683
            C5=0.183

            DO 98 I=1,3
            ES(10,I)=C1*ES(1,I)-C2*ES(2,I)+C3*ES(3,I)-C2*ES(4,I)
            ES(11,I)=-C2*ES(1,I)+C1*ES(2,I)-C2*ES(3,I)+C3*ES(4,I)
            ES(12,I)=C3*ES(1,I)-C2*ES(2,I)+C1*ES(3,I)-C2*ES(4,I)
            ES(13,I)=-C2*ES(1,I)+C3*ES(2,I)-C2*ES(3,I)+C1*ES(4,I)

             !ES(10,I)=C1*ES(1,I)-C2*ES(2,I)+C3*ES(3,I)-C2*ES(4,I)
             !ES(13,I)=-C2*ES(1,I)+C1*ES(2,I)-C2*ES(3,I)+C3*ES(4,I)
             !ES(12,I)=C3*ES(1,I)-C2*ES(2,I)+C1*ES(3,I)-C2*ES(4,I)
             !ES(11,I)=-C2*ES(1,I)+C3*ES(2,I)-C2*ES(3,I)+C1*ES(4,I)

               IF (ELEM.EQ.2) THEN
               ES(14,I)=C4*ES(1,I)+C4*ES(2,I)-C5*ES(3,I)-C5*ES(4,I)
               ES(15,I)=-C5*ES(1,I)+C4*ES(2,I)+C4*ES(3,I)-C5*ES(4,I)
               ES(16,I)=-C5*ES(1,I)-C5*ES(2,I)+C4*ES(3,I)+C4*ES(4,I)
               ES(17,I)=C4*ES(1,I)-C5*ES(2,I)-C5*ES(3,I)+C4*ES(4,I)
               ENDIF
98          CONTINUE

!CALC VMISES AT NODES USING EXTRAPOLATED CORNER NODES STRESSES
DO 103 II = 10,13
  SX(II)=(ES(II,1))
  SY(II)=(ES(II,2))
  SXY(II)=(ES(II,3))
  VMISESD(N,II)=SQRT(((SX(II))**2) + ((SY(II))**2) - ((SX(II))*(SY(II))) + 3*((SXY(II))**2))
103 CONTINUE

DO 99 II=10,9+IEXTRA
  PRINT 85,N,II,ES(II,1),ES(II,2),ES(II,3)
99          CONTINUE

! PRINT VMISES STRESSES AT THE CORNER NODES
PRINT *, 'VON MISES STRESSES AT NODES (DERIVED FROM EXTRAPOLATED STRESSES)'
DO 998 I = 10,13
  PRINT *, VMISESD(N,I)
998 ENDDO
write (10001,*) 'nodal vmises stresses +++++ iteration ',iter,'  elem ',n
write (10001,701) VMisesD(N,10),VMisesD(N,11),VMisesD(N,12),VMisesD(N,13)
write (10003,701) VMisesD(N,10),VMisesD(N,11),VMisesD(N,12),VMisesD(N,13)

! PRINT VMISES STRESSES AT THE gauss pts
write (10000,*) 'Gauss point vmises stresses +++++ iteration ',iter,'  elem ',n
write (10000,701) VMGAUSSD(N,1),VMGAUSSD(N,2),VMGAUSSD(N,3),VMGAUSSD(N,4)
write (10002,701) VMGAUSSD(N,1),VMGAUSSD(N,2),VMGAUSSD(N,3),VMGAUSSD(N,4)

      DO 999 I = 1,4
       PRINT *, 'VON MISES STRESSES AT gauss points'
       PRINT *, VMGAUSSD(N,I)
      999 ENDDO

100   CONTINUE

 CALL NEIGHBOURS(VMISESD,NN,NELEM,NNODE,AVSTRESS)
```

```
 do 321 i=1,nnode
   write(991,705) i,avstress(i)
 321 enddo

701 format(4en16.4)
705 format ('avstress node',i6,en16.4)
 RETURN
 END

!-----------------------------------------------------------------------------------------------
SUBROUTINE NEIGHBOURS(VMISESD,NN,NELEM,NNODE,AVSTRESS)
!-----------------------------------------------------------------------------------------------
           !AVERAGES !STRESSES AT NODES

         integer NN,NELEM,NNODE,NODE,COUNT,I,number
         DIMENSION NN(500,8)
         DOUBLE PRECISION, DIMENSION(1:500,10:13):: VMISESd !VMISES AT NODES
         DOUBLE PRECISION, DIMENSION(0:10,1:500,1:4) :: E
         DOUBLE PRECISION, DIMENSION(1:4) :: STRESS
         DOUBLE PRECISION, DIMENSION(1:500) :: AVSTRESS

         DO 1 NODE=1,NNODE      !SEARCH FOR ELEMENTS SHARING NODE(NODE)
          COUNT=0
          DO I=1,4
           STRESS(I)=0
          ENDDO
           DO 2 ELE=1,NELEM

            IF (NODE==NN(ELE,1)) THEN
             STRESS(1)=VMISESd(ELE,10)
             COUNT=COUNT+1
            ENDIF
            IF (NODE==NN(ELE,2)) THEN
             STRESS(2)=VMISESd(ELE,11)
             COUNT=COUNT+1
            ENDIF
            IF (NODE==NN(ELE,3)) THEN
             STRESS(3)=VMISESd(ELE,12)
             COUNT=COUNT+1
            ENDIF
            IF (NODE==NN(ELE,4)) THEN
             STRESS(4)=VMISESd(ELE,13)
             COUNT=COUNT+1
            ENDIF
            IF (COUNT==4) EXIT
           2 CONTINUE
              AVSTRESS(NODE)=(STRESS(1)+STRESS(2)+STRESS(3)+STRESS(4))/COUNT
!           write (*,*) 'avstress node',node,'=',AVSTRESS(NODE)

           1 CONTINUE

RETURN
END
```