```
      SUBROUTINE ALGOR(IINCS ,IITER ,KRESL ,KUNLD )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C Hyplas database: Global parameters and common blocks
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Numerical constants
      DATA R0   /0.0D0/
C***********************************************************************
C SETS EQUATION RESOLUTION INDEX, KRESL, ACCORDING TO SELECTED ITERATIVE
C ALGORITHM FOR SOLUTION OF THE NON-LINEAR EQUILIBRIUM PROBLEM
C
C REFERENCE: Section 5.4.4
C***********************************************************************
C
C Set KRESL
C ---------
      KRESL=2
      IABSN=IABS(NALGO)
C Initial stiffness method
      IF(IABSN.EQ.1.AND.IINCS.EQ.1.AND.IITER.EQ.1) KRESL=1
C Newton-Raphson tangential stiffness method
      IF(IABSN.EQ.2) KRESL=1
C Modified Newton KT1
      IF(IABSN.EQ.3.AND.IITER.EQ.1) KRESL=1
C Modified Newton KT2
      IF(IABSN.EQ.4.AND.IINCS.EQ.1.AND.IITER.EQ.1) KRESL=1
      IF(IABSN.EQ.4.AND.IITER.EQ.1.AND.KUNLD.EQ.1) KRESL=1
      IF(IABSN.EQ.4.AND.IITER.EQ.2) KRESL=1
C Secant Newton - Initial stiffness
      IF(IABSN.EQ.5.AND.IINCS.EQ.1.AND.IITER.EQ.1) KRESL=1
C Secant Newton - KT1
      IF(IABSN.EQ.6.AND.IITER.EQ.1) KRESL=1
C Secant Newton - KT2
      IF(IABSN.EQ.7.AND.IINCS.EQ.1.AND.IITER.EQ.1) KRESL=1
      IF(IABSN.EQ.7.AND.IITER.EQ.1.AND.KUNLD.EQ.1) KRESL=1
      IF(IABSN.EQ.7.AND.IITER.EQ.2) KRESL=1
C
C Zero prescribed displacements if not first iteration
C ----------------------------------------------------
      IF(IITER.GT.1)THEN
        NRHS=1
        IF(NALGO.LT.0)NRHS=2
        DO 20 ITOTV = 1,NTOTV
          DO 10 IRHS=1,NRHS
            FIXED(ITOTV,IRHS)=R0
   10     CONTINUE
   20   CONTINUE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE ARCLEN
     1(   DFACT      ,DLAMD      ,DLENG      ,DLENM      ,DLENP      ,
     2    IFNEG      ,IINCS      ,IITER      ,INCCUT     ,TFACT      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Arguments
      LOGICAL  INCCUT
C Local logical flags
      LOGICAL  ONEROO ,TWOROO
C Local numerical constants
      PARAMETER
     1(   R0=0.0D0   ,R1=1.0D0   ,R2=2.0D0   )
C***********************************************************************
C SOLVE ARC-LENGTH EQUATIONS AND UPDATE LOAD FACTOR AND VECTOR OF
C ITERATIVE DISPLACEMENTS ACCONDINGLY
C
C REFERENCE: Box 4.4, items (iv)-(vi)
C            Section 4.4
C***********************************************************************
C
      IF(IITER.EQ.1)THEN
C
C For first iteration only
C ========================
C
C evaluate length of tangential solution
        APARM=SCAPRD(DTANG,DTANG,NTOTV)
C
C ...and compute arc-length/load increment factor
        IF(IINCS.EQ.1)THEN
C for the first iteration of the first increment set up the arc-length
C constraint, DLENG, so as to match the user-prescribed initial load
C increment factor, DFACT
          SIGNUM=DBLE(IFNEG)
          DFACT=SIGNUM*DFACT
          DLENG=SQRT(DFACT*DFACT*APARM)
C set up the maximum allowed arc-length according to the specified
C maximum arc-length parameter, DLENP
          DLENM=DLENP*DLENG
        ELSE
C Predictor solution: for the first iteration of other increments
C compute the load increment factor, DFACT, corresponding to the
C arc-length constraint, DLENG, of the current increment
          IF(NARCL.EQ.1)THEN
C Use stiffness determinant sign criterion
            SIGNUM=DBLE(IFNEG)
          ELSEIF(NARCL.EQ.2)THEN
C Use secant path criterion (sign of the internal product between the
C previous converged displacement increment and the current tangential
C solution)
            SCAL=SCAPRD(DINCRO,DTANG,NTOTV)
            IF(SCAL.GT.R0)THEN
              SIGNUM=R1
            ELSE
              SIGNUM=-R1
            ENDIF
          ENDIF
          DFACT=SIGNUM*ABS(DLENG/SQRT(APARM))
        ENDIF
C update current total load factor
        DLAMD=DFACT
        TFACT=TFACT+DLAMD
C
      ELSE
C
C For iterations other than the first: Solve the cylindrical arc-length
C                                      constraint equation to update
C                                      the load factor
C =====================================================================
C
C Compute the coefficients of the arc-length constraint equation
        APARM=R0
        BPARM=R0
        CPARM=R0
        DO 10 ITOTV=1,NTOTV
C Here, DTANG is the current tangential solution, DINCR is the last
C incremental displacement (at the end of the previous iteration) and
C DITER is the current iterative displacement resulting from the
C standard load controlled N-R procedure.
          APARM=APARM+DTANG(ITOTV)*DTANG(ITOTV)
          BPARM=BPARM+R2*(DINCR(ITOTV)+DITER(ITOTV))*DTANG(ITOTV)
          CPARM=CPARM+(DINCR(ITOTV)+DITER(ITOTV))*
     1                (DINCR(ITOTV)+DITER(ITOTV))
   10   CONTINUE
        CPARM=CPARM-(DLENG*DLENG)
C Solve the quadratic equation and decide which root to use
        CALL SOLQUA
     1(   APARM      ,BPARM      ,CPARM      ,ONEROO     ,TWOROO     ,
     2    DLAM1      ,DLAM2      )
        IF(TWOROO)THEN
C Two real roots: Choose the solution that renders minimum angle
C (maximum cosine) between the incremental displacements at the end
C of the previous iteration and the current iteration
          COS1=R0
          COS2=R0
          DO 20 ITOTV=1,NTOTV
            COS1=COS1+(DINCR(ITOTV)+DITER(ITOTV)+DLAM1*DTANG(ITOTV))*
     1            DINCR(ITOTV)
            COS2=COS2+(DINCR(ITOTV)+DITER(ITOTV)+DLAM2*DTANG(ITOTV))*
     1            DINCR(ITOTV)
   20     CONTINUE
          IF(COS1.GT.COS2)THEN
            DLAMD=DLAM1
          ELSE
            DLAMD=DLAM2
          ENDIF
        ELSEIF(ONEROO)THEN
C There is only one root to the equation
```

```
          DLAMD=DLAM1
        ELSE
C There are no real roots: issue warning message and request main
C program to activate increment cutting
          CALL ERRPRT('WE0012')
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
C Update current incremental and total load factors
        DFACT=DFACT+DLAMD
        TFACT=TFACT+DLAMD
C
      ENDIF
C
C For all iterations: compute actual iterative displacement using
C the arc-length expression
C ================================================================
C
      DO 30 ITOTV=1,NTOTV
        DITER(ITOTV)=DITER(ITOTV)+DLAMD*DTANG(ITOTV)
   30 CONTINUE
C
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ARRGO2
     1(   A4TH       ,AMATX      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   NDIM=2     ,NGDIM=4    )
C Arguments
      DIMENSION
     1    A4TH(NDIM,NDIM,NDIM,NDIM)              ,AMATX(NGDIM,NGDIM)
C***********************************************************************
C RE-ARRANGES A FOURTH ORDER TENSOR, STORED AS A 4-INDEX ARRAY, IN
C MATRIX FORM (2-INDEX ARRAY) USING G-MATRIX COMPONENT ORDERING
C (11,21,12,22). FOR 2-D ONLY.
C
C REFERENCE: Section D.2.1
C***********************************************************************
C
      AMATX(1,1)=A4TH(1,1,1,1)
      AMATX(1,2)=A4TH(1,1,2,1)
      AMATX(1,3)=A4TH(1,1,1,2)
      AMATX(1,4)=A4TH(1,1,2,2)
C
      AMATX(2,1)=A4TH(2,1,1,1)
      AMATX(2,2)=A4TH(2,1,2,1)
      AMATX(2,3)=A4TH(2,1,1,2)
      AMATX(2,4)=A4TH(2,1,2,2)
C
      AMATX(3,1)=A4TH(1,2,1,1)
      AMATX(3,2)=A4TH(1,2,2,1)
      AMATX(3,3)=A4TH(1,2,1,2)
      AMATX(3,4)=A4TH(1,2,2,2)
C
      AMATX(4,1)=A4TH(2,2,1,1)
      AMATX(4,2)=A4TH(2,2,2,1)
      AMATX(4,3)=A4TH(2,2,1,2)
      AMATX(4,4)=A4TH(2,2,2,2)
C
      RETURN
      END
```

```
      SUBROUTINE ATMDFB
     1(   AMATX      ,NTYPE      ,QMATX      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MADIM=5    )
      DIMENSION
     1    AMATX(MADIM,MADIM)    ,QMATX(MADIM,MADIM)   ,STRES(*)
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,R3   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,3.0D0/
C***********************************************************************
C COMPUTE THE ADDITIONAL TANGENT MODULUS "q" REQUIRED BY F-BAR
C ELEMENTS:
C                        1                  2
C               q  :=  --- a:(I (x) I) - --- [sigma] (x) I
C                        3                  3
C FOR AXISYMMETRIC CASE, AND
C                        1                  1
C               q  :=  --- a:(I (x) I) - --- [sigma] (x) I
C                        2                  2
C FOR PLANE STRAIN.
C
C REFERENCE: Expressions (15.11) and (15.22)
C***********************************************************************
C
      IF(NTYPE.EQ.2)THEN
C Plane strain
        A=RP5
        B=-RP5
        QMATX(1,1)=A*(AMATX(1,1)+AMATX(1,4))+B*STRES(1)
        QMATX(2,1)=A*(AMATX(2,1)+AMATX(2,4))+B*STRES(3)
        QMATX(3,1)=A*(AMATX(3,1)+AMATX(3,4))+B*STRES(3)
        QMATX(4,1)=A*(AMATX(4,1)+AMATX(4,4))+B*STRES(2)
        QMATX(1,2)=R0
        QMATX(2,2)=R0
        QMATX(3,2)=R0
        QMATX(4,2)=R0
        QMATX(1,3)=R0
        QMATX(2,3)=R0
        QMATX(3,3)=R0
        QMATX(4,3)=R0
        QMATX(1,4)=QMATX(1,1)
        QMATX(2,4)=QMATX(2,1)
        QMATX(3,4)=QMATX(3,1)
        QMATX(4,4)=QMATX(4,1)
      ELSEIF(NTYPE.EQ.3)THEN
C Axisymmetric
        A=R1/R3
        B=-R2/R3
        QMATX(1,1)=A*(AMATX(1,1)+AMATX(1,4)+AMATX(1,5))+B*STRES(1)
        QMATX(2,1)=A*(AMATX(2,1)+AMATX(2,4)+AMATX(2,5))+B*STRES(3)
        QMATX(3,1)=A*(AMATX(3,1)+AMATX(3,4)+AMATX(3,5))+B*STRES(3)
        QMATX(4,1)=A*(AMATX(4,1)+AMATX(4,4)+AMATX(4,5))+B*STRES(2)
        QMATX(5,1)=A*(AMATX(5,1)+AMATX(5,4)+AMATX(5,5))+B*STRES(4)
        QMATX(1,2)=R0
        QMATX(2,2)=R0
        QMATX(3,2)=R0
        QMATX(4,2)=R0
        QMATX(5,2)=R0
        QMATX(1,3)=R0
        QMATX(2,3)=R0
        QMATX(3,3)=R0
        QMATX(4,3)=R0
        QMATX(5,3)=R0
        QMATX(1,4)=QMATX(1,1)
        QMATX(2,4)=QMATX(2,1)
        QMATX(3,4)=QMATX(3,1)
        QMATX(4,4)=QMATX(4,1)
        QMATX(5,4)=QMATX(5,1)
        QMATX(1,5)=QMATX(1,1)
        QMATX(2,5)=QMATX(2,1)
        QMATX(3,5)=QMATX(3,1)
        QMATX(4,5)=QMATX(4,1)
        QMATX(5,5)=QMATX(5,1)
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE BETRIA
     1(   BETRL      ,EEN        ,FINCR      ,NTYPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    BETRL(*)           ,EEN(*)             ,FINCR(3,3)
      DIMENSION
     1    AUXM(2,2)          ,BEN(4)             ,BENMTX(2,2)        ,
     2    BETRLM(2,2)
C***********************************************************************
C COMPUTES THE "ELASTIC TRIAL" LEFT CAUCHY-GREEN STRAIN TENSOR FOR
C HYPERELASTIC-BASED LARGE STRAIN ELASTO-PLASTIC MODELS ACCORDING TO
C THE FORMULA:
C
C                        e trial             e   T
C                       B        :=   F     B   F
C                        n+1           incr  n   incr
C          e
C WHERE  B    IS OBTAINED AS:
C          n
C                        e              e
C                       B   :=  exp[ 2 E  ]
C                        n              n
C         e
C WITH  E   DENOTING THE ELASTIC LOGARITHMIC STRAIN AT  t .
C         n                                              n
C
C REFERENCE: Box 14.3, item (ii)
C***********************************************************************
      BEN(1)=EEN(1)
      BEN(2)=EEN(2)
      BEN(3)=EEN(3)
      BEN(4)=EEN(4)
C Convert engineering elastic logarithmic strain components into the
C corresponding elastic left Cauchy-Green strain tensor components
      CALL SETBE(BEN,NTYPE)
C Convert left Cauchy-Green strain tensor from vector array to matrix
C form
      BENMTX(1,1)=BEN(1)
      BENMTX(2,1)=BEN(3)
      BENMTX(1,2)=BEN(3)
      BENMTX(2,2)=BEN(2)
C
C In-plane components of the elastic trial left Cauchy-Green tensor
C
      CALL RVZERO(AUXM,4)
      DO 30 I=1,2
        DO 20 J=1,2
          DO 10 K=1,2
            AUXM(I,J)=AUXM(I,J)+FINCR(I,K)*BENMTX(K,J)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      CALL RVZERO(BETRLM,4)
      DO 60 I=1,2
        DO 50 J=1,2
          DO 40 K=1,2
            BETRLM(I,J)=BETRLM(I,J)+AUXM(I,K)*FINCR(J,K)
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
C
C         e trial
C Store B        in array form
C         n+1
C
      BETRL(1)=BETRLM(1,1)
      BETRL(2)=BETRLM(2,2)
      BETRL(3)=BETRLM(1,2)
C out-of-plane component
      IF(NTYPE.EQ.2)THEN
        BETRL(4)=BEN(4)
      ELSEIF(NTYPE.EQ.3)THEN
        BETRL(4)=BEN(4)*FINCR(3,3)**2
      ENDIF
C
      RETURN
      END
```

```fortran
      SUBROUTINE CHECK2(MXFRON)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C Hyplas database: Global parameters and common blocks
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Local array
      DIMENSION
     1    NDFRO(MELEM)
C***********************************************************************
C PERFORMS FURTHER CHECKS ON THE INPUT DATA
C***********************************************************************
 1010 FORMAT(/' Check why node',I4,' never appears')
 1020 FORMAT(//' Maximum frontwidth encountered =',I5)
      DO 10 IELEM=1,NELEM
        NDFRO(IELEM)=0
   10 CONTINUE
C Check against two identical nodal coordinates
      IREPCO=0
      DO 40 IPOIN=2,NPOIN
        KPOIN=IPOIN-1
        DO 30 JPOIN=1,KPOIN
          DO 20 IDIME=1,NDIME
            IF(COORD(IDIME,IPOIN,1).NE.COORD(IDIME,JPOIN,1))GOTO 30
   20     CONTINUE
          IREPCO=IREPCO+1
   30   CONTINUE
   40 CONTINUE
C... send warning message if there are nodes with identical coordinates
      IF(IREPCO.NE.0)CALL ERRPRT('WD0001')
C Check for invalid node numbers in connectivity list
      DO 70 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NNODE=IELPRP(3,IELIDN)
        DO 60 INODE=1,NNODE
          IF(LNODS(IELEM,INODE).LE.0.OR.LNODS(IELEM,INODE).GT.NPOIN)
     1    CALL ERRPRT('ED0038')
   60   CONTINUE
   70 CONTINUE
C Change the sign of the last appearance of each degree of freedom
      DO 90 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NNODE=IELPRP(3,IELIDN)
        DO 80 INODE=1,NNODE
          DO 75 IDOFN=2,NDOFN
            LNODS(IELEM,NNODE*(IDOFN-1)+INODE)=
     1                       LNODS(IELEM,INODE)+NPOIN*(IDOFN-1)
   75     CONTINUE
   80   CONTINUE
   90 CONTINUE
      DO 150 ITOTV=1,NTOTV
        KSTAR=0
        DO 130 IELEM=1,NELEM
          IGRUP=IGRPID(IELEM)
          IELIDN=IELTID(IGRUP)
          NNODE=IELPRP(3,IELIDN)
          KZERO=0
          DO 120 INODE=1,NNODE
            DO 110 IDOFN=1,NDOFN
              IEVAB=(IDOFN-1)*NNODE+INODE
              IPOIN=IABS(LNODS(IELEM,INODE))
              IF(MASTER(NDOFN*(IPOIN-1)+IDOFN).NE.ITOTV)GOTO 110
              KZERO=KZERO+1
              IF(KSTAR.NE.0)GOTO 100
              KSTAR=IELEM
              NDFRO(IELEM)=NDFRO(IELEM)+1
  100         CONTINUE
              LELEM=IELEM
              LEVAB=IEVAB
  110       CONTINUE
  120     CONTINUE
  130   CONTINUE
        IF(KSTAR.EQ.0)GOTO 150
        IF(LELEM.LT.NELEM)NDFRO(LELEM+1)=NDFRO(LELEM+1)-1
        LNODS(LELEM,LEVAB)=-LNODS(LELEM,LEVAB)
  150 CONTINUE
C Check for any repetition of a node number within an element
      DO 200 IPOIN=1,NPOIN
        KSTAR=0
        DO 170 IELEM=1,NELEM
          IGRUP=IGRPID(IELEM)
          IELIDN=IELTID(IGRUP)
          NNODE=IELPRP(3,IELIDN)
          KZERO=0
          DO 160 INODE=1,NNODE
            IF(IABS(LNODS(IELEM,INODE)).NE.IPOIN)GOTO 160
            KZERO=KZERO+1
            IF(KZERO.GT.1)CALL ERRPRT('ED0039')
            IF(KSTAR.EQ.0)KSTAR=IELEM
  160     CONTINUE
  170   CONTINUE
C Check that coordinates for an unused node have not been specified
        IF(KSTAR.EQ.0)THEN
```

```
          WRITE(16,1010)IPOIN
          CALL ERRPRT('ED0040')
        ENDIF
  200 CONTINUE
C Calculate the largest frontwidth
      NFRONT=0
      MXFRON=0
      DO 210 IELEM=1,NELEM
        NFRONT=NFRONT+NDFRO(IELEM)
        IF(NFRONT.GT.MXFRON)MXFRON=NFRONT
  210 CONTINUE
      WRITE(16,1020)MXFRON
      IF(MXFRON.GT.MFRON)CALL ERRPRT('ED0041')
C Check data for nodes with prescribed displacements
      DO 230 IVFIX=1,NVFIX
        IF(NOFIX(IVFIX).LE.0.OR.NOFIX(IVFIX).GT.NPOIN)THEN
          CALL ERRPRT('ED0042')
        ENDIF
        KVFIX=IVFIX-1
        DO 220 JVFIX=1,KVFIX
          IF(IVFIX.NE.1.AND.NOFIX(IVFIX).EQ.NOFIX(JVFIX))THEN
            CALL ERRPRT('ED0043')
          ENDIF
  220   CONTINUE
  230 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHKNDB
     1(   FOUND      ,NNODE      ,NEDGEL     ,NODCHK     ,NORDEB     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL FOUND
      DIMENSION
     1    NODCHK(NNODE)      ,NORDEB(NNODE,NEDGEL)
C***********************************************************************
C CHECKS WHETHER A GIVEN SET OF LOCAL ELEMENT NODE NUMBERS CORRESPOND TO
C ONE OF THE ELEMENT BOUNDARIES (EDGES IN 2-D AND FACETS IN 3-D). IF IT
C DOES, RETURNS (STORED IN NODCHK) THE LOCAL NODE NUMBERS ORDERED FOR
C NUMERICAL INTEGRATION ON BOUNDARY.
C***********************************************************************
      FOUND=.FALSE.
C Searches for the element boundary whose nodes coincide with the given
C set
      DO 20 IEDGEL=1,NEDGEL
        DO 10 INODE=1,NNODE
          IF((NODCHK(INODE).NE.0.AND.NORDEB(INODE,IEDGEL).EQ.0).OR.
     1       (NODCHK(INODE).EQ.0.AND.NORDEB(INODE,IEDGEL).NE.0))GOTO 20
   10   CONTINUE
        FOUND=.TRUE.
        GOTO 30
   20 CONTINUE
C
   30 CONTINUE
      IF(FOUND)THEN
C If the given node set corresponds indeed to one of the boundaries of
C the element, stores the node numbers in NODCHK ordered for numerical
C integration on the boundary.
        DO 40 INODE=1,NNODE
          INODEG=NORDEB(INODE,IEDGEL)
          IF(INODEG.NE.0)NODCHK(INODEG)=INODE
   40   CONTINUE
      ENDIF
C
      RETURN
      END
```

```fortran
      SUBROUTINE CONVER
     1(   CONVRG     ,DIVERG     ,IITER      ,TOLER     ,TFACT      )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C Hyplas database: Global parameters and common blocks
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      LOGICAL CONVRG,DIVERG
      SAVE RATOLD ,REMOLD
      DATA R0   ,R20   ,R100   ,R1000   /
     1     0.0D0,20.0D0,100.0D0,1000.0D0/
C***********************************************************************
C COMPUTE GLOBAL RESIDUAL (OUT-OF-BALANCE) FORCE VECTOR AND ITS RELATIVE
C NORM AND SET EQUILIBRIUM CONVERGENCE FLAG
C
C REFERENCE: Expressions (4.72) and (4.77)
C***********************************************************************
 1000 FORMAT(6X,I3,19X,G14.6,15X,G14.6)
 1010 FORMAT(6X,I3,11X,G14.6,6X,G14.6,6X,G14.6)
C Initialize relevant variables
C -----------------------------
      CONVRG=.FALSE.
      DIVERG=.FALSE.
      RESID=R0
      RETOT=R0
      REMAX=R0
C Evaluate global nodal internal and external forces
C --------------------------------------------------
      CALL RVZERO(STFOR,NTOTV)
      CALL RVZERO(TOFOR,NTOTV)
      DO 30 IELEM=1,NELEM
        IGRUP =IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NNODE =IELPRP(3,IELIDN)
        KEVAB=0
        DO 20 INODE=1,NNODE
          LOCNO=IABS(LNODS(IELEM,INODE))
          DO 10 IDOFN=1,NDOFN
            KEVAB=KEVAB+1
            NPOSI=MASTER((LOCNO-1)*NDOFN+IDOFN)
C current internal force
            STFOR(NPOSI)=STFOR(NPOSI)+ELOAD(KEVAB,IELEM)
C current external force
            TOFOR(NPOSI)=TOFOR(NPOSI)+TFACT*RLOAD(KEVAB,IELEM)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C Loop over nodal points
C ----------------------
      DO 70 IPOIN=1,NPOIN
        ISVAB=(IPOIN-1)*NDOFN
C Search for node in prescribed displacements
        DO 60 IVFIX=1,NVFIX
          IF(NOFIX(IVFIX).EQ.IPOIN.AND.ANGLE(IVFIX).NE.R0)THEN
C Rotate forces to local nodal coordinate system for prescribed
C displacements at an angle (for 2-D only)
            C=COS(ANGLE(IVFIX))
            S=SIN(ANGLE(IVFIX))
            ISVAB=ISVAB+1
            JSVAB=ISVAB+1
            GASHI= C*STFOR(ISVAB)+S*STFOR(JSVAB)
            GASHJ=-S*STFOR(ISVAB)+C*STFOR(JSVAB)
            STFOR(ISVAB)=GASHI
            STFOR(JSVAB)=GASHJ
            GASHI= C*TOFOR(ISVAB)+S*TOFOR(JSVAB)
            GASHJ=-S*TOFOR(ISVAB)+C*TOFOR(JSVAB)
            TOFOR(ISVAB)=GASHI
            TOFOR(JSVAB)=GASHJ
C Evaluate reactions
            KSVAB=ISVAB-1
            DO 40 IDOFN=1,NDOFN
            KSVAB=KSVAB+1
            IF(IFFIX(KSVAB).NE.0)THEN
              TREAC(IVFIX,IDOFN)=STFOR(KSVAB)-TOFOR(KSVAB)
              TOFOR(KSVAB)=TOFOR(KSVAB)+TREAC(IVFIX,IDOFN)
            ELSE
              TREAC(IVFIX,IDOFN)=R0
            ENDIF
   40       CONTINUE
C Rotate forces and reactions back to global system
            GASHI= C*STFOR(ISVAB)-S*STFOR(JSVAB)
            GASHJ= S*STFOR(ISVAB)+C*STFOR(JSVAB)
            STFOR(ISVAB)=GASHI
            STFOR(JSVAB)=GASHJ
            GASHI= C*TOFOR(ISVAB)-S*TOFOR(JSVAB)
            GASHJ= S*TOFOR(ISVAB)+C*TOFOR(JSVAB)
            TOFOR(ISVAB)=GASHI
            TOFOR(JSVAB)=GASHJ
            GASHI= C*TREAC(IVFIX,1)-S*TREAC(IVFIX,2)
            GASHJ= S*TREAC(IVFIX,1)+C*TREAC(IVFIX,2)
            TREAC(IVFIX,1)=GASHI
            TREAC(IVFIX,2)=GASHJ
            GOTO 70
          ELSEIF(NOFIX(IVFIX).EQ.IPOIN)THEN
```

```
C Evaluate reactions
            DO 50 IDOFN=1,NDOFN
              ISVAB=ISVAB+1
              IF(IFFIX(ISVAB).NE.0)THEN
                TREAC(IVFIX,IDOFN)=STFOR(ISVAB)-TOFOR(ISVAB)
                TOFOR(ISVAB)=TOFOR(ISVAB)+TREAC(IVFIX,IDOFN)
              ELSE
                TREAC(IVFIX,IDOFN)=R0
              ENDIF
   50       CONTINUE
            GOTO 70
          ENDIF
   60   CONTINUE
   70 CONTINUE
C Evaluate residual and external force norm
C -------------------------------------------
      DO 80 ITOTV=1,NTOTV
        REFOR=TOFOR(ITOTV)-STFOR(ITOTV)
        RESID=RESID+REFOR*REFOR
        RETOT=RETOT+TOFOR(ITOTV)*TOFOR(ITOTV)
C maximum nodal residual
        AGASH=ABS(REFOR)
        IF(AGASH.GT.REMAX)REMAX=AGASH
   80 CONTINUE
C Euclidean norm of residual
      RESID=SQRT(RESID)
C Euclidean norm of external force
      RETOT=SQRT(RETOT)
C compute relative residual norm
      IF(RETOT.EQ.R0)THEN
        RATIO=R0
      ELSE
        RATIO=R100*RESID/RETOT
      ENDIF
      IF(NALGO.GT.0)THEN
        WRITE(16,1000)IITER,RATIO,REMAX
        WRITE(*,1000) IITER,RATIO,REMAX
      ELSE
        WRITE(16,1010)IITER,RATIO,REMAX,TFACT
        WRITE(*,1010) IITER,RATIO,REMAX,TFACT
      ENDIF
C Set convergence/divergence flags
C ----------------------------------
      IF(RATIO.LE.TOLER.OR.ABS(REMAX).LE.(TOLER/R1000))CONVRG=.TRUE.
      IF(IITER.NE.1.AND.(RATIO.GT.R20*RATOLD.OR.REMAX.GT.R20*REMOLD))
     1  DIVERG=.TRUE.
      RATOLD=RATIO
      REMOLD=REMAX
C Evaluate element residual forces before exit -> store it in ELOAD
C -----------------------------------------------------------------
      DO 100 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NEVAB=IELPRP(5,IELIDN)
C Current element residual (out-of-balance force) = current element
C external load - current element internal force
        DO 90 IEVAB=1,NEVAB
          ELOAD(IEVAB,IELEM)=TFACT*RLOAD(IEVAB,IELEM)-ELOAD(IEVAB,IELEM)
   90   CONTINUE
  100 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE CSTEP2
     1(   AMATX      ,BETRL      ,DMATX      ,STRES      ,DETF       ,
     2    NTYPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MADIM=5    ,MSTRE=4    )
      EXTERNAL  DDLGD2 ,DLGD2
      LOGICAL   OUTOFP
      DIMENSION
     1    AMATX(MADIM,MADIM)        ,BETRL(MSTRE)              ,
     2    DMATX(MSTRE,MSTRE)        ,STRES(MSTRE)
      DIMENSION
     1    AUXMTX(MADIM,MADIM)       ,BMTX(MADIM,MADIM)         ,
     2    DLGAUX(MSTRE,MSTRE)       ,DLGMTX(MADIM,MADIM)       ,
     3    DMATX2(MADIM,MADIM)       ,IG(MADIM)
      DATA
     1    R1   ,R2    /
     2    1.0D0,2.0D0 /
      DATA
     1    IG(1),IG(2),IG(3),IG(4),IG(5)  /
     2    1    ,3    ,3    ,2    ,4      /
C***********************************************************************
C COMPUTE THE CONSISTENT SPATIAL TANGENT MODULUS 'a' FOR LARGE STRAIN
C HYPERELASTIC-BASED ELASTOPLASTIC MATERIAL MODELS
C
C REFERENCE: Section 14.5
C***********************************************************************
      IF(NTYPE.EQ.3)THEN
        OUTOFP=.TRUE.
        NADIM=5
      ELSEIF(NTYPE.EQ.1.OR.NTYPE.EQ.2)THEN
        OUTOFP=.FALSE.
        NADIM=4
      ELSE
        CALL ERRPRT('EI0020')
      ENDIF
C                            e trial   e trial
C Compute the derivative  dE       /dB
      CALL DISO2
     1(   DLGAUX     ,DDLGD2     ,DLGD2      ,OUTOFP     ,BETRL      )
      FACTOR=R1/DETF
      DO 20 I=1,MSTRE
        DO 10 J=1,MSTRE
          DLGAUX(I,J)=FACTOR*DLGAUX(I,J)
   10   CONTINUE
   20 CONTINUE
C                                       e trial   e trial
C Rearrange components of DMATX and  dE       /dB         into the
C ordering (11,21,12,22,33), compatible with the discrete gradient G.
      CALL RVZERO(DMATX2,MADIM*MADIM)
      CALL RVZERO(DLGMTX,MADIM*MADIM)
      DO 40 INEW=1,NADIM
        DO 30 JNEW=1,NADIM
          IOLD=IG(INEW)
          JOLD=IG(JNEW)
          DLGMTX(INEW,JNEW)=DLGAUX(IOLD,JOLD)
          DMATX2(INEW,JNEW)=DMATX(IOLD,JOLD)
   30   CONTINUE
   40 CONTINUE
C Compute remaining needed matrix [DELTA_ik BETRL_jl+DELTA_jl BETRL_il]
      CALL RVZERO(BMTX,MADIM*MADIM)
      BMTX(1,1)=R2*BETRL(1)
      BMTX(1,3)=R2*BETRL(3)
      BMTX(2,1)=BETRL(3)
      BMTX(2,2)=BETRL(1)
      BMTX(2,3)=BETRL(2)
      BMTX(2,4)=BETRL(3)
      BMTX(3,1)=BETRL(3)
      BMTX(3,2)=BETRL(1)
      BMTX(3,3)=BETRL(2)
      BMTX(3,4)=BETRL(3)
      BMTX(4,2)=R2*BETRL(3)
      BMTX(4,4)=R2*BETRL(2)
      IF(OUTOFP)BMTX(5,5)=R2*BETRL(4)
C Assemble the spatial tangent modulus a
C --------------------------------------
C compute the product  D:L:B
      CALL RVZERO(AUXMTX,MADIM*MADIM)
      DO 70 I=1,NADIM
        DO 60 J=1,NADIM
          DO 50 K=1,NADIM
            AUXMTX(I,J)=AUXMTX(I,J)+DMATX2(I,K)*DLGMTX(K,J)
   50     CONTINUE
   60   CONTINUE
   70 CONTINUE
      CALL RVZERO(AMATX,MADIM*MADIM)
      DO 100 I=1,NADIM
        DO 90 J=1,NADIM
          DO 80 K=1,NADIM
            AMATX(I,J)=AMATX(I,J)+AUXMTX(I,K)*BMTX(K,J)
   80     CONTINUE
   90   CONTINUE
  100 CONTINUE
C subtract  [SIGMA_il DELTA_jk]
      AMATX(1,1)=AMATX(1,1)-STRES(1)
      AMATX(1,3)=AMATX(1,3)-STRES(3)
      AMATX(2,1)=AMATX(2,1)-STRES(3)
      AMATX(2,3)=AMATX(2,3)-STRES(2)
      AMATX(3,2)=AMATX(3,2)-STRES(1)
      AMATX(3,4)=AMATX(3,4)-STRES(3)
      AMATX(4,2)=AMATX(4,2)-STRES(3)
      AMATX(4,4)=AMATX(4,4)-STRES(2)
      IF(OUTOFP)AMATX(5,5)=AMATX(5,5)-STRES(4)
      RETURN
      END
```

```
      SUBROUTINE CSTOGD
     1(   AMATX      ,B          ,IPROPS     ,NTYPE      ,RPROPS     ,
     2    STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL OUTOFP ,REPEAT
      PARAMETER
     1(   MADIM=5    ,MSTRE=4    ,NDIM=2     )
      PARAMETER(IPOGDC=2)
      DIMENSION
     1    AMATX(MADIM,MADIM) ,B(MSTRE)           ,IPROPS(*)          ,
     2    RPROPS(*)          ,STRES(MSTRE)
      DIMENSION
     1    DELTA(3,3)         ,DPSTRE(3,3)        ,DTAUDB(MSTRE,MSTRE),
     2    EIGPRJ(MSTRE,NDIM) ,EIGB(NDIM)         ,PSTALP(3)          ,
     3    PSTRES(3)          ,PSTRTC(3)
      DATA
     1    DELTA(1,1)   ,DELTA(1,2)   ,DELTA(1,3)    /
     2    1.0D0        ,0.0D0        ,0.0D0         /
     3    DELTA(2,1)   ,DELTA(2,2)   ,DELTA(2,3)    /
     4    0.0D0        ,1.0D0        ,0.0D0         /
     5    DELTA(3,1)   ,DELTA(3,2)   ,DELTA(3,3)    /
     6    0.0D0        ,0.0D0        ,1.0D0         /
      DATA
     1    R1   ,R2   ,R3   ,R6   /
     2    1.0D0,2.0D0,3.0D0,6.0D0/
C***********************************************************************
C COMPUTATION OF THE CONSISTENT SPATIAL TANGENT MODULUS 'a' FOR
C OGDEN TYPE HYPERELASTIC MATERIAL MODEL.
C PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 13.5.2
C***********************************************************************
C Set Ogden material constants
C ============================
C Number of terms in Ogden's strain-energy function
      NOGTRM=IPROPS(3)
C Bulk modulus (incompressibility penalty parameter)
      BULK=RPROPS(IPOGDC+NOGTRM*2)
C Compute principal stretches
C ===========================
C Perform spectral decomposition of the left Cauchy-Green tensor B
      CALL SPDEC2
     1(   EIGPRJ     ,EIGB       ,REPEAT     ,B          )
C Compute in-plane principal stretches
      PSTRTC(1)=SQRT(EIGB(1))
      PSTRTC(2)=SQRT(EIGB(2))
C and out-of-plane stretches
      IF(NTYPE.EQ.1)THEN
        PSTRTC(3)=R1/(PSTRTC(1)*PSTRTC(2))
        DETF=R1
      ELSEIF(NTYPE.EQ.2)THEN
        PSTRTC(3)=R1
        DETF=PSTRTC(1)*PSTRTC(2)
      ELSEIF(NTYPE.EQ.3)THEN
        PSTRTC(3)=SQRT(B(4))
        DETF=PSTRTC(1)*PSTRTC(2)*PSTRTC(3)
      ENDIF
C Recover principal Kirchhoff stresses (from the given Cauchy stress)
      PSTRES(1)=(STRES(1)*EIGPRJ(1,1)+STRES(2)*EIGPRJ(2,1)+
     1                               R2*STRES(3)*EIGPRJ(3,1))*DETF
      PSTRES(2)=(STRES(1)*EIGPRJ(1,2)+STRES(2)*EIGPRJ(2,2)+
     1                               R2*STRES(3)*EIGPRJ(3,2))*DETF
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)PSTRES(3)=STRES(4)*DETF
C Compute derivatives of principal Kirchhoff stresses
C ===================================================
      CALL RVZERO(DPSTRE,9)
      IF(NTYPE.EQ.1) THEN
C Plane stress: Perfectly incompressibility assumed
C -------------------------------------------------
        NSTRA=2
        DO 10 IP=1,NOGTRM
          ALPHA=RPROPS(IPOGDC+IP*2-1)
          ALPHMU=ALPHA*RPROPS(IPOGDC+IP*2-2)
          PSTALP(1)=PSTRTC(1)**ALPHA
          PSTALP(2)=PSTRTC(2)**ALPHA
          FACTOR=R1/(PSTALP(1)*PSTALP(2))
          DO I=1,NSTRA
            DO J=1,NSTRA
              DPSTRE(I,J)=DPSTRE(I,J)+ALPHMU/(R2*PSTRTC(J)**2)*
     1                    (PSTALP(J)*DELTA(I,J)+FACTOR)
            END DO
          END DO
   10   CONTINUE
      ELSE IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
C Plane strain and axisymmetric: Regularised Ogden model
C ------------------------------------------------------
C compute principal Kirchhoff stresses derivatives
        R1D3=R1/R3
        IF(NTYPE.EQ.2)THEN
          NSTRA=2
        ELSEIF(NTYPE.EQ.3)THEN
          NSTRA=3
        ENDIF
        DO 40 IP=1,NOGTRM
          CMU=RPROPS(IPOGDC-1+IP*2-1)
          ALPHA=RPROPS(IPOGDC-1+IP*2)
          PSTALP(1)=PSTRTC(1)**ALPHA
          PSTALP(2)=PSTRTC(2)**ALPHA
```

```
          PSTALP(3)=PSTRTC(3)**ALPHA
          ALPHMU=ALPHA*CMU
          FACTOR=R1D3*(PSTALP(1)+PSTALP(2)+PSTALP(3))
          FACVOL=DETF**(-ALPHA*R1D3)
          DO 30 I=1,NSTRA
            DO 20 J=1,NSTRA
              DPSTRE(I,J)=DPSTRE(I,J)+ALPHMU*FACVOL/(R6*PSTRTC(J)**2)*
     1                    (FACTOR-PSTALP(I)-PSTALP(J)+R3*PSTALP(I)*
     2                    DELTA(I,J))
   20       CONTINUE
   30     CONTINUE
   40   CONTINUE
        DO 60 I=1,NSTRA
          DO 50 J=1,NSTRA
            DPSTRE(I,J)=DPSTRE(I,J)+BULK/(R2*PSTRTC(J)**2)
   50     CONTINUE
   60   CONTINUE
      ENDIF
C Compute the derivative of the Kirchhoff stress with respect to B
C (use routine for computation of derivative of general isotropic
C tensor functions of one tensor)
C ==================================================================
      IF(NTYPE.EQ.3)THEN
        OUTOFP=.TRUE.
        NADIM=5
      ELSE
        OUTOFP=.FALSE.
        NADIM=4
      ENDIF
      CALL DGISO2
     1(   DPSTRE     ,DTAUDB     ,EIGPRJ     ,EIGB       ,PSTRES     ,
     2    OUTOFP     ,REPEAT     )
C Assemble the spatial tangent modulus 'a'
C ========================================
      R2DDET=R2/DETF
C upper triangle and diagonal terms
      AMATX(1,1)=R2DDET*(DTAUDB(1,1)*B(1)+DTAUDB(1,3)*B(3))-STRES(1)
      AMATX(1,2)=R2DDET*(DTAUDB(1,3)*B(1)+DTAUDB(1,2)*B(3))
      AMATX(1,3)=R2DDET*(DTAUDB(1,1)*B(3)+DTAUDB(1,3)*B(2))-STRES(3)
      AMATX(1,4)=R2DDET*(DTAUDB(1,3)*B(3)+DTAUDB(1,2)*B(2))
      AMATX(2,2)=R2DDET*(DTAUDB(3,3)*B(1)+DTAUDB(3,2)*B(3))
      AMATX(2,3)=R2DDET*(DTAUDB(3,1)*B(3)+DTAUDB(3,3)*B(2))-STRES(2)
      AMATX(2,4)=R2DDET*(DTAUDB(3,3)*B(3)+DTAUDB(3,2)*B(2))
      AMATX(3,3)=R2DDET*(DTAUDB(3,1)*B(3)+DTAUDB(3,3)*B(2))
      AMATX(3,4)=R2DDET*(DTAUDB(3,3)*B(3)+DTAUDB(3,2)*B(2))-STRES(3)
      AMATX(4,4)=R2DDET*(DTAUDB(2,3)*B(3)+DTAUDB(2,2)*B(2))-STRES(2)
      IF(NTYPE.EQ.3) THEN
        AMATX(1,5)=R2DDET*DTAUDB(1,4)*B(4)
        AMATX(2,5)=R2DDET*DTAUDB(3,4)*B(4)
        AMATX(3,5)=R2DDET*DTAUDB(3,4)*B(4)
        AMATX(4,5)=R2DDET*DTAUDB(2,4)*B(4)
        AMATX(5,5)=R2DDET*DTAUDB(4,4)*B(4)-STRES(4)
      ENDIF
C lower triangle
      DO 80 J=1,NADIM
        DO 70 I=J+1,NADIM
          AMATX(I,J)=AMATX(J,I)
 70     CONTINUE
 80   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE CSTPDS
     1(   AMATX      ,DGAM       ,EPFLAG     ,FINCR      ,IPROPS     ,
     2    LALGVA     ,NTYPE      ,RPROPS     ,RSTAVA     ,RSTAVN     ,
     3    STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   IPHARD=6   ,IPHVAR=5   ,MADIM=5    ,NDIM=2     ,NGDIM=4    ,
     2    NIPROP=3   ,NLALGV=6   ,NRALGV=4   ,NRSTAV=5   ,NSTRE=4    ,
     3    NSYST=4    )
C Arguments
      LOGICAL
     1    EPFLAG     ,LALGVA(NLALGV)
      DIMENSION
     1    AMATX(MADIM,MADIM) ,DGAM(NRALGV)       ,FINCR(3,3)         ,
     2    IPROPS(NIPROP)     ,RPROPS(*)          ,RSTAVA(NRSTAV)     ,
     3    RSTAVN(NRSTAV)     ,STRES(NSTRE)
C Local arrays and variables
      LOGICAL
     1    S1ACT  ,S2ACT  ,S3ACT  ,S4ACT
      DIMENSION
     1    APMATX(NGDIM,NGDIM),AUXMTX(NGDIM,NGDIM),AUX2ND(NDIM,NDIM)  ,
     2    AUX4TH(NDIM,NDIM,NDIM,NDIM)            ,BEISO(3,3)         ,
     3    BMATX(NDIM,NDIM,NSYST)                 ,DELKRO(NDIM,NDIM)  ,
     4    DEREXP(3,3,3,3)    ,DEVPRJ(NGDIM,NGDIM),DEVSTR(NGDIM)      ,
     5    DS0M0(NDIM,NDIM)   ,DUMATX(NDIM,NDIM,NDIM,NDIM)            ,
     6    FE(2,2)            ,FEFETR(NGDIM,NGDIM),FEISO(2,2)         ,
     7    FEN(2,2)           ,FETISO(2,2)        ,FETRL(2,2)         ,
     8    FOIDS(NGDIM,NGDIM) ,FPILOG(3,3)        ,GINV(NSYST,NSYST)  ,
     9    GMATX(NSYST,NSYST) ,IACSET(NSYST)      ,
     O    SM0MS0(NDIM,NDIM,NSYST)                ,SOID(NGDIM)        ,
     1    S0M0(NDIM,NDIM,NSYST)                  ,
     2    UMATX(NDIM,NDIM,NDIM,NDIM)             ,VECM(NDIM,NSYST)   ,
     3    VECM0(NDIM,NSYST)  ,VECS(NDIM,NSYST)   ,VECS0(NDIM,NSYST)  ,
     4    VMATX(NGDIM,NGDIM)
C... Kroenecker delta
      DATA
     1    DELKRO(1,1),DELKRO(1,2)/
     2    1.0D0      ,0.D0       /
     3    DELKRO(2,1),DELKRO(2,2)/
     4    0.0D0      ,1.D0       /
C... fourth order (symmetric subspace) identity components stored in
C    matrix form using G matrix ordering (11,21,12,22)
      DATA
     1    FOIDS(1,1),FOIDS(1,2),FOIDS(1,3),FOIDS(1,4)/
     2    1.0D0     ,0.0D0     ,0.0D0     ,0.0D0     /
     3    FOIDS(2,1),FOIDS(2,2),FOIDS(2,3),FOIDS(2,4)/
     4    0.0D0     ,0.5D0     ,0.5D0     ,0.0D0     /
     5    FOIDS(3,1),FOIDS(3,2),FOIDS(3,3),FOIDS(3,4)/
     6    0.0D0     ,0.5D0     ,0.5D0     ,0.0D0     /
     7    FOIDS(4,1),FOIDS(4,2),FOIDS(4,3),FOIDS(4,4)/
     8    0.0D0     ,0.0D0     ,0.0D0     ,1.0D0     /
C... second order identity components in stored in vector form using G
C    matrix ordering
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,0.0D0    ,0.0D0    ,1.D0     /
      DATA
     1    R0   ,R1   ,R2   ,R3   /
     2    0.0D0,1.0D0,2.0D0,3.0D0/
C***********************************************************************
C COMPUTATION OF THE CONSISTENT SPATIAL TANGENT MODULUS 'a' FOR
C THE PLANAR DOULBLE SLIP SINGLE CRYSTAL ELASTO-PLASTIC MODEL.
C MODEL VALID FOR PLANE STRAIN ONLY.
C
C REFERENCE: Expression (16.85)
C***********************************************************************
C Stop program if not plane strain
      IF(NTYPE.NE.2)CALL ERRPRT('EI0035')
C Retrieve some state and algorithmic variables
C ---------------------------------------------
C... current hardening internal variable
      HRVAR=RSTAVA(IPHVAR)
C... current elastic deformation gradient
      FE(1,1)=RSTAVA(1)
      FE(2,1)=RSTAVA(2)
      FE(1,2)=RSTAVA(3)
      FE(2,2)=RSTAVA(4)
C... current active slip-systems logical flags
      S1ACT=LALGVA(3)
      S2ACT=LALGVA(4)
      S3ACT=LALGVA(5)
      S4ACT=LALGVA(6)
C... elastic deformation gradient at the beginning of the current load
C    increment
      FEN(1,1)=RSTAVN(1)
      FEN(2,1)=RSTAVN(2)
      FEN(1,2)=RSTAVN(3)
      FEN(2,2)=RSTAVN(4)
C Retrieve material properties
C ----------------------------
C... neo-Hookean constants
      GMODU=RPROPS(2)
      BULK=RPROPS(3)
C... initial system orientation
      THETA=RPROPS(4)
C... relative angle between systems
      BETA=RPROPS(5)
C... number of sampling points on hardening curve
      NHARD=IPROPS(3)
C Set some constants
C ------------------
      R1D3=R1/R3
      R2D3=R2*R1D3
C Assemble deviatoric projection tensor (use G matrix ordering)
      DO 20 I=1,NGDIM
        DO 10 J=1,NGDIM
          DEVPRJ(I,J)=FOIDS(I,J)-R1D3*SOID(I)*SOID(J)
   10   CONTINUE
   20 CONTINUE
C Get current Cauchy deviatoric stress and Cauchy hydrostatic pressure
      P=R1D3*(STRES(1)+STRES(2)+STRES(4))
C... use G matrix component ordering to store in-plane deviatoric
```

```
C    Cauchy stress components
      DEVSTR(1)=STRES(1)-P
      DEVSTR(2)=STRES(3)
      DEVSTR(3)=STRES(3)
      DEVSTR(4)=STRES(2)-P
C Compute isochoric component of Fe and Be
      DETFE=FE(1,1)*FE(2,2)-FE(1,2)*FE(2,1)
      FACTOR=DETFE**(-R1D3)
      FEISO(1,1)=FACTOR*FE(1,1)
      FEISO(1,2)=FACTOR*FE(1,2)
      FEISO(2,1)=FACTOR*FE(2,1)
      FEISO(2,2)=FACTOR*FE(2,2)
      CALL RVZERO(BEISO,9)
      DO 50 I=1,2
        DO 40 J=1,2
          DO 30 K=1,2
            BEISO(I,J)=BEISO(I,J)+FEISO(I,K)*FEISO(J,K)
   30     CONTINUE
   40   CONTINUE
   50 CONTINUE
      BEISO(3,3)=FACTOR*FACTOR
C Trace of isochoric component of Be
      TRBISO=BEISO(1,1)+BEISO(2,2)+BEISO(3,3)
C
C
C Compute ELASTIC tangent modulus
C ===============================
C
      GFAC=R2D3*GMODU*TRBISO/DETFE
      BULFAC=BULK/DETFE
      R2P=R2*P
C... assemble tensorially compact part
      DO 70 I=1,NGDIM
        DO 60 J=1,NGDIM
          AMATX(I,J)=BULFAC*SOID(I)*SOID(J)-R2P*FOIDS(I,J)+
     1               GFAC*DEVPRJ(I,J)-
     2               R2D3*(DEVSTR(I)*SOID(J)+SOID(I)*DEVSTR(J))
   60   CONTINUE
   70 CONTINUE
C... add non-compact part: delta_ik sigma_jl
      AMATX(1,1)=AMATX(1,1)+STRES(1)
      AMATX(3,1)=AMATX(3,1)+STRES(3)
      AMATX(2,2)=AMATX(2,2)+STRES(1)
      AMATX(4,2)=AMATX(4,2)+STRES(3)
      AMATX(1,3)=AMATX(1,3)+STRES(3)
      AMATX(3,3)=AMATX(3,3)+STRES(2)
      AMATX(2,4)=AMATX(2,4)+STRES(3)
      AMATX(4,4)=AMATX(4,4)+STRES(2)
C
C
      IF(EPFLAG)THEN
C
C Compute and add algorithm-consistent PLASTIC contribution to spatial
C tangent modulus
C ===================================================================
C
C Compute individual terms needed to assemble the plastic contribution
C -------------------------------------------------------------------
C
C Last elastic trial deformation gradient
        CALL RVZERO(FETRL,4)
        DO 100 I=1,2
          DO 90 J=1,2
            DO 80 K=1,2
              FETRL(I,J)=FETRL(I,J)+FINCR(I,K)*FEN(K,J)
   80       CONTINUE
   90     CONTINUE
  100   CONTINUE
C... isochoric component
        DETFET=FETRL(1,1)*FETRL(2,2)-FETRL(1,2)*FETRL(2,1)
        VOLFAC=DETFET**(-R1D3)
        FETISO(1,1)=VOLFAC*FETRL(1,1)
        FETISO(2,1)=VOLFAC*FETRL(2,1)
        FETISO(1,2)=VOLFAC*FETRL(1,2)
        FETISO(2,2)=VOLFAC*FETRL(2,2)
C Assemble relevant fourth order tensor
        DO 140 I=1,NDIM
          DO 130 J=1,NDIM
            DO 120 K=1,NDIM
              DO 110 L=1,NDIM
                AUX4TH(I,J,K,L)=FEISO(I,L)*FETISO(J,K)
  110         CONTINUE
  120       CONTINUE
  130     CONTINUE
  140   CONTINUE
C... rearrange in matrix form with G matrix component ordering
        CALL ARRGO2(AUX4TH,FEFETR)
C
C Compute exact exponential map derivative
C ----------------------------------------
C
C retrieve information on current set of active slip-systems and set up
C corresponding initial slip-system vectors
        IACSYS=0
        IF(S1ACT)THEN
C... system 1:
          IACSYS=IACSYS+1
          IACSET(IACSYS)=1
          VECS0(1,1)=COS(THETA)
          VECS0(2,1)=SIN(THETA)
          VECM0(1,1)=-SIN(THETA)
          VECM0(2,1)=COS(THETA)
        ENDIF
        IF(S2ACT)THEN
C... system 2:
          IACSYS=IACSYS+1
          IACSET(IACSYS)=2
          VECS0(1,2)=COS(THETA+BETA)
          VECS0(2,2)=SIN(THETA+BETA)
          VECM0(1,2)=-SIN(THETA+BETA)
          VECM0(2,2)=COS(THETA+BETA)
        ENDIF
```

```
        IF(S3ACT)THEN
          IACSYS=IACSYS+1
          IACSET(IACSYS)=3
C... system 3:
          VECS0(1,3)=-COS(THETA)
          VECS0(2,3)=-SIN(THETA)
          VECM0(1,3)=-SIN(THETA)
          VECM0(2,3)=COS(THETA)
        ENDIF
        IF(S4ACT)THEN
C... system 4:
          IACSYS=IACSYS+1
          IACSET(IACSYS)=4
          VECS0(1,4)=-COS(THETA+BETA)
          VECS0(2,4)=-SIN(THETA+BETA)
          VECM0(1,4)=-SIN(THETA+BETA)
          VECM0(2,4)=COS(THETA+BETA)
        ENDIF
C... number of currently active systems
        NACSYS=IACSYS
C Compute current elastic push forward of the slip-system vectors
C of the active systems
        CALL RVZERO(VECS,NDIM*NSYST)
        CALL RVZERO(VECM,NDIM*NSYST)
        DO 170 I=1,NDIM
          DO 160 J=1,NDIM
            DO 150 II=1,NACSYS
              ISYST=IACSET(II)
              VECS(I,ISYST)=VECS(I,ISYST)+FEISO(I,J)*VECS0(J,ISYST)
              VECM(I,ISYST)=VECM(I,ISYST)+FEISO(I,J)*VECM0(J,ISYST)
  150       CONTINUE
  160     CONTINUE
  170   CONTINUE
C Current slope of Taylor hardening curve
        HSLOPE=DPLFUN(HRVAR,NHARD,RPROPS(IPHARD))
C Compute logarithm of inverse of incremental plastic deformation
C gradient by summing up contributions from each active slip system
        CALL RVZERO(FPILOG,9)
        DO 200 II=1,NACSYS
          ISYST=IACSET(II)
          DO 190 I=1,2
            DO 180 J=1,2
              FPILOG(I,J)=FPILOG(I,J)-
     1                    DGAM(ISYST)*VECS0(I,ISYST)*VECM0(J,ISYST)
  180       CONTINUE
  190     CONTINUE
  200   CONTINUE
C... and the corresponding exact derivative of the exponential map
        CALL DEXPMP(   DEREXP      ,NOCONV     ,FPILOG     )
C
C Compute jacobian of non-linear system of return mapping equations
C -------------------------------------------------------------------
C compute some preliminary matrices
        CALL RVZERO(BMATX,NDIM*NDIM*NSYST)
        DO 300 II=1,NACSYS
          ISYST=IACSET(II)
          DO 220 I=1,2
            DO 210 J=1,2
              S0M0(I,J,ISYST)=VECS0(I,ISYST)*VECM0(J,ISYST)
              SM0MS0(I,J,ISYST)=VECS(I,ISYST)*VECM0(J,ISYST)+
     1                          VECM(I,ISYST)*VECS0(J,ISYST)
  210       CONTINUE
  220     CONTINUE
          CALL RVZERO(DS0M0,NDIM*NDIM)
          DO 260 I=1,2
            DO 250 J=1,2
              DO 240 K=1,2
                DO 230 L=1,2
                  DS0M0(I,J)=DS0M0(I,J)+
     1                       DEREXP(I,J,K,L)*S0M0(K,L,ISYST)
  230           CONTINUE
  240         CONTINUE
  250       CONTINUE
  260     CONTINUE
          DO 290 I=1,2
            DO 280 J=1,2
              DO 270 K=1,2
                BMATX(I,J,ISYST)=BMATX(I,J,ISYST)+
     1                           FETISO(I,K)*DS0M0(K,J)
  270         CONTINUE
  280       CONTINUE
  290     CONTINUE
  300   CONTINUE
C Assemble exact jacobian of non-linear system
        DO 320 II=1,NACSYS
          ISYST=IACSET(II)
          DO 310 JJ=1,NACSYS
            JSYST=IACSET(JJ)
            GMATX(II,JJ)=GMODU*
     1         SCAPRD(SM0MS0(1,1,ISYST),BMATX(1,1,JSYST),NDIM*NDIM)+
     2         HSLOPE
  310     CONTINUE
  320   CONTINUE
C Invert jacobian: Note that for the double slip model only one or two
C systems may be active
        IF(NACSYS.EQ.1)THEN
          IF(GMATX(1,1).EQ.R0)CALL ERRPRT('EE0006')
          GINV(1,1)=R1/GMATX(1,1)
        ELSEIF(NACSYS.EQ.2)THEN
          DETG=GMATX(1,1)*GMATX(2,2)-GMATX(1,2)*GMATX(2,1)
          IF(DETG.EQ.R0)CALL ERRPRT('EE0006')
          DETGIN=R1/DETG
          GINV(1,1)=GMATX(2,2)*DETGIN
          GINV(2,2)=GMATX(1,1)*DETGIN
          GINV(1,2)=-GMATX(1,2)*DETGIN
          GINV(2,1)=-GMATX(2,1)*DETGIN
        ENDIF
C Compute U matrix
C ----------------
        CALL RVZERO(UMATX,NDIM*NDIM*NDIM*NDIM)
        DO 380 I=1,NDIM
          DO 370 J=1,NDIM
```

```
          DO 360 K=1,NDIM
            DO 350 L=1,NDIM
              DO 340 II=1,NACSYS
                ISYST=IACSET(II)
                DO 330 JJ=1,NACSYS
                  JSYST=IACSET(JJ)
                  UMATX(I,J,K,L)=UMATX(I,J,K,L)+S0M0(I,J,ISYST)*
     1                           GINV(II,JJ)*SM0MS0(K,L,JSYST)
  330           CONTINUE
  340         CONTINUE
  350       CONTINUE
  360     CONTINUE
  370   CONTINUE
  380 CONTINUE
C Compute product [D:U]
      CALL RVZERO(DUMATX,NDIM*NDIM*NDIM*NDIM)
      DO 440 I=1,NDIM
        DO 430 J=1,NDIM
          DO 420 K=1,NDIM
            DO 410 L=1,NDIM
              DO 400 M=1,NDIM
                DO 390 N=1,NDIM
                  DUMATX(I,J,K,L)=DUMATX(I,J,K,L)+
     1                            DEREXP(I,J,M,N)*UMATX(M,N,K,L)
  390           CONTINUE
  400         CONTINUE
  410       CONTINUE
  420     CONTINUE
  430   CONTINUE
  440 CONTINUE
C... and the contribution to a^p involving the product D:U
      CALL RVZERO(AUX4TH,NDIM*NDIM*NDIM*NDIM)
      DO 490 I=1,NDIM
        DO 480 J=1,NDIM
          DO 470 K=1,NDIM
            DO 460 L=1,NDIM
              DO 450 M=1,NDIM
                AUX4TH(I,J,K,L)=AUX4TH(I,J,K,L)+
     1                          DUMATX(I,J,K,M)*FEISO(L,M)
  450         CONTINUE
  460       CONTINUE
  470     CONTINUE
  480   CONTINUE
  490 CONTINUE
      CALL RVZERO(AUX2ND,NDIM*NDIM)
      DO 530 I=1,NDIM
        DO 520 J=1,NDIM
          DO 510 K=1,NDIM
            DO 500 L=1,NDIM
              AUX2ND(I,J)=AUX2ND(I,J)+DUMATX(I,J,K,L)*FEISO(K,L)
  500       CONTINUE
  510     CONTINUE
  520   CONTINUE
  530 CONTINUE
      DO 570 I=1,NDIM
        DO 560 J=1,NDIM
          DO 550 K=1,NDIM
            DO 540 L=1,NDIM
              AUX4TH(I,J,K,L)=AUX4TH(I,J,K,L)-
     1                        R1D3*AUX2ND(I,J)*DELKRO(K,L)
  540       CONTINUE
  550     CONTINUE
  560   CONTINUE
  570 CONTINUE
C... rearrange in matrix form
      CALL ARRGO2(AUX4TH,VMATX)
      CALL RVZERO(AUXMTX,NGDIM*NGDIM)
      DO 600 I=1,NGDIM
        DO 590 J=1,NGDIM
          DO 580 K=1,NGDIM
            AUXMTX(I,J)=AUXMTX(I,J)+FEFETR(I,K)*VMATX(K,J)
  580     CONTINUE
  590   CONTINUE
  600 CONTINUE
C Compute plastic contribution
      CALL RVZERO(APMATX,NGDIM*NGDIM)
      AUX=R2*GMODU*GMODU/DETFE
      DO 630 I=1,NGDIM
        DO 620 J=1,NGDIM
          DO 610 K=1,NGDIM
            APMATX(I,J)=APMATX(I,J)-AUX*DEVPRJ(I,K)*AUXMTX(K,J)
  610     CONTINUE
  620   CONTINUE
  630 CONTINUE
C Add plastic contribution to spatial tangent modulus
      DO 650 I=1,NGDIM
        DO 640 J=1,NGDIM
          AMATX(I,J)=AMATX(I,J)+APMATX(I,J)
  640   CONTINUE
  650 CONTINUE
C
    ENDIF
C
    RETURN
    END
```

```
      SUBROUTINE CTDAMA
     1(   DGAMA      ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=6  ,MSTRE=4)
      LOGICAL EPFLAG
      DIMENSION
     1    DMATX(MSTRE,MSTRE),IPROPS(*)           ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+2)   ,STRES(MSTRE)
      DIMENSION
     1    DEVPRJ(MSTRE,MSTRE),FOID(MSTRE,MSTRE)  ,S(MSTRE)           ,
     2    SOID(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    R0   ,R1   ,R2   ,R3   ,R6   ,SMALL ,TOLDGA/
     2    0.0D0,1.0D0,2.0D0,3.0D0,6.0D0,1.D-16,1.D-08/
C***********************************************************************
C COMPUTATION OF THE CONSISTENT TANGENT MODULUS FOR LEMAITRE'S DUCTILE
C DAMAGE ELASTO-PLASTIC MODEL WITH PIECE-WISE LINEAR ISOTROPIC
C HARDENING.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 12.4.2
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0052')
C Retrieve current hardening and damage variables
      HVAR=RSTAVA(MSTRE+1)
      DAMAGE=RSTAVA(MSTRE+2)
      OMEGA=R1-DAMAGE
C Retrieve material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      DAMEXP=RPROPS(4)
      DAMDEN=RPROPS(5)
      NHARD=IPROPS(3)
C Shear and bulk moduli
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R1D3=R1/R3
C Set deviatoric projection tensor
      IF(NTYPE.EQ.2)THEN
        NSTRE=3
      ELSEIF(NTYPE.EQ.3)THEN
        NSTRE=4
      ENDIF
      DO 20 I=1,NSTRE
        DO 10 J=1,NSTRE
          DEVPRJ(I,J)=FOID(I,J)-SOID(I)*SOID(J)*R1D3
   10   CONTINUE
   20 CONTINUE
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent
C ========================================
        R3G=R3*GMODU
        R6G=R6*GMODU
        R2BULK=R2*BULK
        ROO3D2=SQRT(R3/R2)
C Current hydrostatic pressure
        P=(STRES(1)+STRES(2)+STRES(4))*R1D3
C Current deviatoric stress components
        S(1)=STRES(1)-P
        S(2)=STRES(2)-P
        S(3)=STRES(3)
        S(4)=STRES(4)-P
C Recover last (undamaged) elastic trial von Mises effective stress
        SNORM=SQRT(S(1)*S(1)+S(2)*S(2)+R2*S(3)*S(3)+S(4)*S(4))
        Q=ROO3D2*SNORM
        IF(ABS(OMEGA).LT.SMALL)THEN
C... internal error: singular residual derivatives
          CALL ERRPRT('EI0053')
        ENDIF
        IF(DGAMA.EQ.R0)THEN
C... at DGAMA=0 use numerical perturbation
          DGAMA=TOLDGA
        ENDIF
        QTILTR=(Q+R3G*DGAMA)/OMEGA
C.. and last (undamaged) elastic trial pressure
        PTILDE=P/OMEGA
C Get factors required in the assemblage of the elastoplastic tangent
C --------------------------------------------------------------------
        HSLOPE=DPLFUN(HVAR,NHARD,RPROPS(IPHARD))
        SIGMAY=PLFUN(HVAR,NHARD,RPROPS(IPHARD))
        Y=-SIGMAY**2/R6G-PTILDE**2/R2BULK
        AUX=-Y/DAMDEN
        AUXB=(QTILTR-SIGMAY)/R3G
        PHIT=QTILTR-SIGMAY
        DOMEGA=(R3G+OMEGA*HSLOPE)/PHIT
        DY=-HSLOPE*SIGMAY/R3G
        DOMDQT=-OMEGA/PHIT
C... residual derivatives
        DRES=DOMEGA-HSLOPE/R3G*AUX**DAMEXP-
     2       AUXB*DAMEXP*DY/DAMDEN*AUX**(DAMEXP-R1)
        DRDPTL=DAMEXP*AUXB*AUX**(DAMEXP-R1)*PTILDE/
     1         (DAMDEN*BULK)
        DRDQTL=DOMDQT+AUX**DAMEXP/R3G
C Compute some factors
        A1=-DRDQTL/DRES
        A2=-DRDPTL/DRES
        A3=A2*DOMEGA
        A4=A1*DOMEGA+DOMDQT
```

```
        AFACT=R2G*SIGMAY*OMEGA/QTILTR
        BFACT=R2G*(A1*HSLOPE*OMEGA+A4*SIGMAY-SIGMAY*OMEGA/QTILTR)
        BFACT=BFACT/(SNORM*SNORM)
        CFACT=BULK*(A2*HSLOPE*OMEGA+A3*SIGMAY)/ROO3D2
        CFACT=CFACT/SNORM
        DFACT=PTILDE*R2G*ROO3D2*A4
        DFACT=DFACT/SNORM
        EFACT=BULK*(OMEGA+PTILDE*A3)
C Assemble elastoplastic tangent modulus
C --------------------------------------
        DO 40 I=1,NSTRE
          DO 30 J=1,NSTRE
            DMATX(I,J)=AFACT*DEVPRJ(I,J)+BFACT*S(I)*S(J)+
     1                 CFACT*S(I)*SOID(J)+DFACT*SOID(I)*S(J)+
     2                 EFACT*SOID(I)*SOID(J)
   30     CONTINUE
   40   CONTINUE
      ELSE
C Compute damaged elasticity matrix
C ---------------------------------
C... upper triangle
        FACTG=OMEGA*R2G
        FACTK=OMEGA*BULK
        DO 60 I=1,NSTRE
          DO 50 J=I,NSTRE
            DMATX(I,J)=FACTG*DEVPRJ(I,J)+FACTK*SOID(I)*SOID(J)
   50     CONTINUE
   60   CONTINUE
C... lower triangle
        DO 80 J=1,NSTRE-1
          DO 70 I=J+1,NSTRE
            DMATX(I,J)=DMATX(J,I)
   70     CONTINUE
   80   CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE CTDMEL
     1(   DMATX      ,NTYPE      ,RPROPS     ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(MSTRE=4)
C Arguments
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,RPROPS(*)          ,RSTAVA(*)          ,
     2    STRES(MSTRE)
C Local arrays and variables
      LOGICAL
     1    REPEAT             ,OUTOFP
      DIMENSION
     1    DPSTRA(3,3)        ,DPSTRE(3,3)        ,EIGPRJ(MSTRE,2)    ,
     2    PDMINU(3,3)        ,PDPLUS(3,3)        ,PSTRA(3)           ,
     3    PSTRES(3)          ,STRAIN(MSTRE)      ,VI(3)              ,
     4    VIDMIN(3)          ,VIDPLU(3)
      DATA
     1    VI(1),VI(2),VI(3)/
     2    1.D0 ,1.D0 ,1.D0 /
      DATA
     1    R0   ,R1   ,R2   /
     2    0.0D0,1.0D0,2.0D0/
C***********************************************************************
C COMPUTATION OF THE TANGENT MODULUS (ELASTICITY MATRIX) FOR THE
C ISOTROPICALLY DAMAGED ISOTROPIC ELASTIC MATERIAL MODEL WITH
C MICROCRACK/VOID CLOSURE EFFECTS
C
C REFERENCE: Section 12.6.1
C***********************************************************************
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0056')
C Set material constants
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      DAMAGE=RPROPS(4)
      HFACT=RPROPS(5)
C Retrieve current (physical) strains
      STRAIN(1)=RSTAVA(1)
      STRAIN(2)=RSTAVA(2)
      STRAIN(3)=RSTAVA(3)/R2
      STRAIN(4)=RSTAVA(4)
C Perform spectral decomposition of the strain tensor
      CALL SPDEC2(EIGPRJ,PSTRA,REPEAT,STRAIN)
      PSTRA(3)=STRAIN(4)
C Compute the principal stresses (internal product between stress
C tensor and individual eigenprojection tensors)
      PSTRES(1)=STRES(1)*EIGPRJ(1,1)+STRES(2)*EIGPRJ(2,1)+
     1          R2*STRES(3)*EIGPRJ(3,1)
      PSTRES(2)=STRES(1)*EIGPRJ(1,2)+STRES(2)*EIGPRJ(2,2)+
     1          R2*STRES(3)*EIGPRJ(3,2)
      PSTRES(3)=STRES(4)
C Zero relevant arrays
      CALL RVZERO(PDPLUS,9)
      CALL RVZERO(PDMINU,9)
C Construct current projection matrices
C... positive and negative principal stress projection matrices
      IF(PSTRES(1).GE.R0)THEN
        PDPLUS(1,1)=R1/(R1-DAMAGE)
        PDMINU(1,1)=R0
      ELSE
        PDPLUS(1,1)=R0
        PDMINU(1,1)=R1/(R1-HFACT*DAMAGE)
      ENDIF
      IF(PSTRES(2).GE.R0)THEN
        PDPLUS(2,2)=R1/(R1-DAMAGE)
        PDMINU(2,2)=R0
      ELSE
        PDPLUS(2,2)=R0
        PDMINU(2,2)=R1/(R1-HFACT*DAMAGE)
      ENDIF
      IF(PSTRES(3).GE.R0)THEN
        PDPLUS(3,3)=R1/(R1-DAMAGE)
        PDMINU(3,3)=R0
      ELSE
        PDPLUS(3,3)=R0
        PDMINU(3,3)=R1/(R1-HFACT*DAMAGE)
      ENDIF
C... positive and negative trace operators
      TRACE=PSTRES(1)+PSTRES(2)+PSTRES(3)
      IF(TRACE.GE.R0)THEN
        VIDPLU(1)=R1/(R1-DAMAGE)
        VIDPLU(2)=R1/(R1-DAMAGE)
        VIDPLU(3)=R1/(R1-DAMAGE)
        VIDMIN(1)=R0
        VIDMIN(2)=R0
        VIDMIN(3)=R0
      ELSE
        VIDPLU(1)=R0
        VIDPLU(2)=R0
        VIDPLU(3)=R0
        VIDMIN(1)=R1/(R1-HFACT*DAMAGE)
        VIDMIN(2)=R1/(R1-HFACT*DAMAGE)
        VIDMIN(3)=R1/(R1-HFACT*DAMAGE)
      ENDIF
C Inverse elasticity operator that transforms principal stresses into
C principal strains (matrix of derivatives of principal strains with
C respect to principal stresses)
      DO 20 I=1,3
        DO 10 J=1,3
```

```
          DPSTRA(I,J)=(R1+POISS)/YOUNG*(PDPLUS(I,J)+PDMINU(I,J))-
     1                POISS/YOUNG*(VI(I)*(VIDPLU(J)+VIDMIN(J)))
   10   CONTINUE
   20 CONTINUE
C Compute elasticity operator that transforms principal strains into
C principal stresses (matrix of derivatives of principal stresses with
C respect to principal strains)
      CALL INVMT3(DPSTRA,DPSTRE,DET)
C
C Use general routine for isotropic tensor functions of a single tensor
C to compute the tangent operator
C
      IF(NTYPE.EQ.2)THEN
        OUTOFP=.FALSE.
      ELSEIF(NTYPE.EQ.3)THEN
        OUTOFP=.TRUE.
      ENDIF
      CALL DGISO2
     1(   DPSTRE     ,DMATX      ,EIGPRJ     ,PSTRA      ,PSTRES     ,
     2    OUTOFP     ,REPEAT     )
C
      RETURN
      END
```

```
      SUBROUTINE CTDP
     1(   DGAM       ,DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,
     2    NTYPE      ,RPROPS     ,RSTAVA     ,STRAT      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MSTRE=4)
      LOGICAL APEX, EPFLAG, LALGVA(3)
      DIMENSION
     1    DMATX(MSTRE,MSTRE),IPROPS(*)           ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)   ,STRAT(MSTRE)
      DIMENSION
     1    EETD(MSTRE)        ,FOID(MSTRE,MSTRE)  ,SOID(MSTRE)        ,
     2    UNIDEV(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    R0   ,R1   ,RP5  ,R2   ,R3   /
     2    0.0D0,1.0D0,0.5D0,2.0D0,3.0D0/
C***********************************************************************
C COMPUTATION OF CONSISTENT TANGENT MODULUS FOR DRUCKER-PRAGER TYPE
C ELASTO-PLASTIC MATERIAL WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW RULE AND
C PIECE-WISE LINEAR ISOTROPIC HARDENING
C
C REFERENCE: Section 8.3.5
C***********************************************************************
      IF(NTYPE.EQ.2)THEN
        NSTRE=3
      ELSEIF(NTYPE.EQ.3)THEN
        NSTRE=4
      ELSE
        CALL ERRPRT('EI0017')
      ENDIF
C Retrieve accumulated plastic strain, DGAMA and APEX algorithm flag
      EPBAR=RSTAVA(MSTRE+1)
      DGAMA=DGAM
      APEX=LALGVA(3)
C Set some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      ETA=RPROPS(4)
      XI=RPROPS(5)
      ETABAR=RPROPS(6)
      NHARD=IPROPS(3)
C and some constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R1D3=R1/R3
      ROOT2=SQRT(R2)
C
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent
C ========================================
C Hardening slope
        HSLOPE=DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
        IF(APEX)THEN
C Elastoplastic tangent consistent with apex return
C -------------------------------------------------
          ALPHA=XI/ETABAR
          BETA=XI/ETA
          AFACT=BULK*(R1-BULK/(BULK+ALPHA*BETA*HSLOPE))
          DO 20 I=1,NSTRE
            DO 10 J=1,NSTRE
              DMATX(I,J)=AFACT*SOID(I)*SOID(J)
   10       CONTINUE
   20     CONTINUE
        ELSE
C Elastoplastic tangent consistent with smooth cone wall return
C -------------------------------------------------------------
C Elastic trial deviatoric (physical) strain
          EEVD3=(STRAT(1)+STRAT(2)+STRAT(4))*R1D3
          EETD(1)=STRAT(1)-EEVD3
          EETD(2)=STRAT(2)-EEVD3
          EETD(3)=STRAT(3)*RP5
          EETD(4)=STRAT(4)-EEVD3
          ETDNOR=SQRT(EETD(1)*EETD(1)+EETD(2)*EETD(2)+
     1             R2*EETD(3)*EETD(3)+EETD(4)*EETD(4))
C Unit deviatoric flow vector
          IF(ETDNOR.NE.R0)THEN
            EDNINV=R1/ETDNOR
          ELSE
            EDNINV=R0
          ENDIF
          DO 30 I=1,NSTRE
            UNIDEV(I)=EETD(I)*EDNINV
   30     CONTINUE
C Assemble tangent
          AUX=R1/(GMODU+BULK*ETA*ETABAR+XI*XI*HSLOPE)
          AFACT=R2G*(R1-DGAMA/(ROOT2*ETDNOR))
          AFACD3=AFACT*R1D3
          BFACT=R2G*(DGAMA/(ROOT2*ETDNOR)-GMODU*AUX)
          CFACT=-ROOT2*GMODU*BULK*AUX
          DFACT=BULK*(R1-BULK*ETA*ETABAR*AUX)
          DO 50 I=1,NSTRE
            DO 40 J=1,NSTRE
              DMATX(I,J)=AFACT*FOID(I,J)+BFACT*UNIDEV(I)*UNIDEV(J)+
     1                   CFACT*(ETA*UNIDEV(I)*SOID(J)+
     2                           ETABAR*SOID(I)*UNIDEV(J))+
     3                   (DFACT-AFACD3)*SOID(I)*SOID(J)
   40       CONTINUE
   50     CONTINUE
        ENDIF
      ELSE
```

```
C Compute elasticity matrix
C =========================
        FACTOR=BULK-R2G*R1D3
        DO 70 I=1,NSTRE
          DO 60 J=I,NSTRE
            DMATX(I,J)=R2G*FOID(I,J)+FACTOR*SOID(I)*SOID(J)
   60     CONTINUE
   70   CONTINUE
        DO 90 J=1,NSTRE-1
          DO 80 I=J+1,NSTRE
            DMATX(I,J)=DMATX(J,I)
   80     CONTINUE
   90   CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE CTDPPN
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,
     2    NTYPE      ,RPROPS     ,RSTAVA     ,STRAT      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(  MSTRE=4  )
C Arguments
      LOGICAL EPFLAG, LALGVA(3)
      DIMENSION
     1    RALGVA(3)          ,DMATX(MSTRE,MSTRE),IPROPS(*)          ,
     2    RPROPS(*)          ,RSTAVA(MSTRE+1)   ,STRAT(MSTRE)
C Local arrays
      DIMENSION
     1    D12(3)             ,D21(3)
C***********************************************************************
C COMPUTATION OF THE CONSISTENT TANGENT MODULUS FOR THE DRUCKER-PRAGER
C ELASTO-PLASTIC MATERIAL WITH PIECE-WISE LINEAR ISOTROPIC HARDENING.
C PLANE STRESS IMPLEMENTATION ONLY.
C
C REFERENCE: Sections 9.2.4-5
C***********************************************************************
C Stops program if not plane stress
      IF(NTYPE.NE.1)CALL ERRPRT('EI0039')
C Retrieve the elastic trial THICKNESS STRAIN last determined in the
C plane stress enforcement loop of subroutine SUDPPN. The in-plane
C elastic trial components have already been stored in the first
C three components of STRAT before the present routine was called.
      STRAT(4)=RALGVA(3)
C Compute the axisymmetric consistent tangent matrix
      CALL CTDP
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,
     2    3          ,RPROPS     ,RSTAVA     ,STRAT      )
C Decompose into submatrices
      D12(1)=DMATX(1,4)
      D12(2)=DMATX(2,4)
      D12(3)=DMATX(3,4)
      D21(1)=DMATX(4,1)
      D21(2)=DMATX(4,2)
      D21(3)=DMATX(4,3)
      D22=DMATX(4,4)
C Assemble plane stress consistent tangent matrix
      DO 20 I=1,3
        DO 10 J=1,3
          DMATX(I,J)=DMATX(I,J)-D12(I)*D21(J)/D22
   10   CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CTEL
     1(   DMATX       ,NTYPE       ,RPROPS      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(MSTRE=4)
      DIMENSION
     1    DMATX(MSTRE,MSTRE),RPROPS(*)
      DIMENSION
     1    FOID(MSTRE,MSTRE)  ,SOID(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    R1   ,R2   ,R3   ,R4   /
     2    1.0D0,2.0D0,3.0D0,4.0D0/
C***********************************************************************
C COMPUTATION OF THE TANGENT MODULUS (ELASTICITY MATRIX) FOR THE LINEAR
C ELASTIC MATERIAL MODEL
C
C REFERENCE: Expression (4.44)
C***********************************************************************
C
C Set shear and bulk modulus
C --------------------------
C
      GMODU=RPROPS(2)
      BULK=RPROPS(3)
C
      R1D3=R1/R3
      R2G=R2*GMODU
      FACTOR=BULK-R2G*R1D3
C
C Compute elasticity matrix
C -------------------------
C
      IF(NTYPE.EQ.1)THEN
C plane stress
        NSTRE=3
        R4GD3=R4*GMODU*R1D3
        FACTOR=(BULK-R2G*R1D3)*(R2G/(BULK+R4GD3))
      ELSEIF(NTYPE.EQ.2)THEN
C plane strain
        NSTRE=3
        FACTOR=BULK-R2G*R1D3
      ELSEIF(NTYPE.EQ.3)THEN
C axisymmetric
        NSTRE=4
        FACTOR=BULK-R2G*R1D3
      ELSE
C stops program if other stress state
        CALL ERRPRT('EI0019')
      ENDIF
C
C Assemble matrix
C
      DO 20 I=1,NSTRE
        DO 10 J=I,NSTRE
          DMATX(I,J)=R2G*FOID(I,J)+FACTOR*SOID(I)*SOID(J)
   10   CONTINUE
   20 CONTINUE
C lower triangle
      DO 40 J=1,NSTRE-1
        DO 30 I=J+1,NSTRE
          DMATX(I,J)=DMATX(J,I)
   30   CONTINUE
   40 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE CTMC
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MDIM=3,  MSTRE=4)
C Arguments
      LOGICAL EPFLAG  ,LALGVA(5)
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(MSTRE)       ,STRES(MSTRE)
C Local arrays and variables
      LOGICAL APEX ,EDGE ,OUTOFP ,RIGHT ,REPEAT
      DIMENSION
     1    DPSTRS(MDIM,MDIM)  ,EIGPRJ(MSTRE,2)    ,FOID(MSTRE,MSTRE)  ,
     2    PSTRS(MDIM)        ,PSTRA(MDIM)        ,SOID(MSTRE)        ,
     3    STRAC(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    RP5  ,R1   ,R2   ,R3   ,R4   /
     2    0.5D0,1.0D0,2.0D0,3.0D0,4.0D0/
C***********************************************************************
C COMPUTATION OF CONSISTENT TANGENT MODULUS FOR MOHR-COULOMB TYPE
C ELASTO-PLASTIC MATERIAL WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW RULE AND
C PIECE-WISE LINEAR ISOTROPIC HARDENING.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 8.2.5
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0026')
C Current accumulated plastic strain
      EPBAR=RSTAVA(MSTRE+1)
C Set material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      SINPHI=RPROPS(4)
      COSPHI=RPROPS(5)
      SINPSI=RPROPS(6)
      NHARD=IPROPS(3)
C Set needed algorithmic variables
      EDGE=LALGVA(3)
      RIGHT=LALGVA(4)
      APEX=LALGVA(5)
C Set some constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R4G=R4*GMODU
      R2BULK=R2*BULK
      R2CPHI=R2*COSPHI
      R4C2PH=R2CPHI*R2CPHI
      R1D3=R1/R3
      R2D3=R2*R1D3
      R2GD3=R2G*R1D3
      R4GD3=R4G*R1D3
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent
C -----------------------------------------
C Spectral decomposition of the elastic trial strain
        STRAC(1)=STRAT(1)
        STRAC(2)=STRAT(2)
        STRAC(3)=STRAT(3)*RP5
        CALL SPDEC2(EIGPRJ,PSTRA,REPEAT,STRAC)
        PSTRA(3)=STRAT(4)
C and current total stress
        PSTRS(1)=STRES(1)*EIGPRJ(1,1)+STRES(2)*EIGPRJ(2,1)+
     1                           R2*STRES(3)*EIGPRJ(3,1)
        PSTRS(2)=STRES(1)*EIGPRJ(1,2)+STRES(2)*EIGPRJ(2,2)+
     1                           R2*STRES(3)*EIGPRJ(3,2)
        PSTRS(3)=STRES(4)
C Identify directions of maximum and minimum principal trial stresses
        II=1
        JJ=1
        PSTMAX=PSTRA(II)
        PSTMIN=PSTRA(JJ)
        DO 10 I=2,3
          IF(PSTRA(I).GE.PSTMAX)THEN
            II=I
            PSTMAX=PSTRA(II)
          ENDIF
          IF(PSTRA(I).LT.PSTMIN)THEN
            JJ=I
            PSTMIN=PSTRA(JJ)
          ENDIF
   10   CONTINUE
        IF(II.NE.1.AND.JJ.NE.1)MM=1
        IF(II.NE.2.AND.JJ.NE.2)MM=2
        IF(II.NE.3.AND.JJ.NE.3)MM=3
        IF(EDGE)THEN
C Tangent consistent with 2-vector return to edge
          SPHSPS=SINPHI*SINPSI
          CONSTA=R4G*(R1+R1D3*SPHSPS)+R4*BULK*SPHSPS
          IF(RIGHT)THEN
            CONSTB=R2G*(R1+SINPHI+SINPSI-R1D3*SPHSPS)+R4*BULK*SPHSPS
          ELSE
            CONSTB=R2G*(R1-SINPHI-SINPSI-R1D3*SPHSPS)+R4*BULK*SPHSPS
          ENDIF
          FACTA=R4C2PH*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          DRVAA=-CONSTA-FACTA
          DRVAB=-CONSTB-FACTA
          DRVBA=-CONSTB-FACTA
```

```fortran
          DRVBB=-CONSTA-FACTA
          AUX1=R2G*(R1+R1D3*SINPSI)+R2BULK*SINPSI
          AUX2=(R4GD3-R2BULK)*SINPSI
          AUX3=R2G*(R1-R1D3*SINPSI)-R2BULK*SINPSI
          R1DDET=R1/(DRVAA*DRVBB-DRVAB*DRVBA)
          IF(RIGHT)THEN
C ...returned to right edge
            DPSTRS(II,II)=BULK+R4GD3+AUX1*(-DRVAB+DRVBB+DRVAA-DRVBA)*
     1                   (R2G+(R2BULK+R2GD3)*SINPHI)*R1DDET
            DPSTRS(II,MM)=BULK-R2GD3+AUX1*(R2G*(DRVAB-DRVAA)+
     1                   ((-DRVAB+DRVBB+DRVAA-DRVBA)*(R2BULK+R2GD3)+
     2                   (DRVBA-DRVBB)*R2G)*SINPHI)*R1DDET
            DPSTRS(II,JJ)=BULK-R2GD3+AUX1*(R2G*(DRVBA-DRVBB)+
     1                   ((-DRVAB+DRVBB+DRVAA-DRVBA)*(R2BULK+R2GD3)+
     2                   (DRVAB-DRVAA)*R2G)*SINPHI)*R1DDET
            DPSTRS(MM,II)=BULK-R2GD3+(AUX2*(DRVAB-DRVBB)+AUX3*(DRVBA-
     1                   DRVAA))*(R2G+(R2BULK+R2GD3)*SINPHI)*R1DDET
            DPSTRS(MM,MM)=BULK+R4GD3+(AUX2*((R2BULK*(DRVAB-DRVBB)+
     1                   (DRVAB*R2GD3+DRVBB*R4GD3))*SINPHI-DRVAB*R2G)+
     2                   AUX3*(DRVAA*R2G+(R2BULK*(DRVBA-DRVAA)-
     3                   (DRVAA*R2GD3+DRVBA*R4GD3))*SINPHI))*R1DDET
            DPSTRS(MM,JJ)=BULK-R2GD3+(AUX2*((R2BULK*(DRVAB-DRVBB)-
     1                   (DRVBB*R2GD3+DRVAB*R4GD3))*SINPHI+DRVBB*R2G)+
     2                   AUX3*((R2BULK*(DRVBA-DRVAA)+(DRVAA*R4GD3+
     3                   DRVBA*R2GD3))*SINPHI-DRVBA*R2G))*R1DDET
            DPSTRS(JJ,II)=BULK-R2GD3+((AUX2*(DRVBA-DRVAA)+AUX3*(DRVAB-
     1                   DRVBB))*((R2BULK+R2GD3)*SINPHI+R2G))*R1DDET
            DPSTRS(JJ,MM)=BULK-R2GD3+(AUX2*(((R2BULK*(DRVBA-DRVAA)-
     1                   (DRVBA*R4GD3+DRVAA*R2GD3))*SINPHI)+DRVAA*R2G)+
     2                   AUX3*(((R2BULK*(DRVAB-DRVBB)+(DRVAB*R2GD3+
     3                   DRVBB*R4GD3))*SINPHI)-DRVAB*R2G))*R1DDET
            DPSTRS(JJ,JJ)=BULK+R4GD3+(AUX2*(((R2BULK*(DRVBA-DRVAA)+
     1                   (DRVAA*R4GD3+DRVBA*R2GD3))*SINPHI)-DRVBA*R2G)+
     2                   AUX3*(((R2BULK*(DRVAB-DRVBB)-(DRVAB*R4GD3+
     3                   DRVBB*R2GD3))*SINPHI)+DRVBB*R2G))*R1DDET
          ELSE
C ...returned to left edge
            DPSTRS(II,II)=BULK+R4GD3+(AUX1*(((R2BULK*(DRVBB-DRVAB)+
     1                   (DRVAB*R4GD3+DRVBB*R2GD3))*SINPHI)+DRVBB*R2G)+
     2                   AUX2*(((R2BULK*(DRVBA-DRVAA)+(DRVAA*R4GD3+
     3                   DRVBA*R2GD3))*SINPHI)+DRVBA*R2G))*R1DDET
            DPSTRS(II,MM)=BULK-R2GD3+(AUX1*(((R2BULK*(DRVBB-DRVAB)-
     1                   (DRVAB*R2GD3+DRVBB*R4GD3))*SINPHI)-DRVAB*R2G)+
     2                   AUX2*(((R2BULK*(DRVBA-DRVAA)-(DRVAA*R2GD3+
     3                   DRVBA*R4GD3))*SINPHI)-DRVAA*R2G))*R1DDET
            DPSTRS(II,JJ)=BULK-R2GD3+((AUX1*(DRVBB-DRVAB)+AUX2*(DRVBA-
     1                   DRVAA))*(((R2BULK+R2GD3)*SINPHI)-R2G))*R1DDET
            DPSTRS(MM,II)=BULK-R2GD3+(AUX1*(((R2BULK*(DRVAA-DRVBA)-
     1                   (DRVAA*R4GD3+DRVBA*R2GD3))*SINPHI)-DRVBA*R2G)+
     2                   AUX2*(((R2BULK*(DRVAB-DRVBB)-(DRVAB*R4GD3+
     3                   DRVBB*R2GD3))*SINPHI)-DRVBB*R2G))*R1DDET
            DPSTRS(MM,MM)=BULK+R4GD3+(AUX1*(((R2BULK*(DRVAA-DRVBA)+
     1                   (DRVAA*R2GD3+DRVBA*R4GD3))*SINPHI)+DRVAA*R2G)+
     2                   AUX2*(((R2BULK*(DRVAB-DRVBB)+(DRVAB*R2GD3+
     3                   DRVBB*R4GD3))*SINPHI)+DRVAB*R2G))*R1DDET
            DPSTRS(MM,JJ)=BULK-R2GD3+((AUX1*(DRVAA-DRVBA)+AUX2*(DRVAB-
     1                   DRVBB))*(((R2BULK+R2GD3)*SINPHI)-R2G))*R1DDET
            DPSTRS(JJ,II)=BULK-R2GD3+(AUX3*(((R2BULK*(DRVAB-DRVBB-DRVAA+
     1                   DRVBA)+(DRVAA-DRVAB)*R4GD3+(DRVBA-DRVBB)*
     2                   R2GD3)*SINPHI)+(DRVBA-DRVBB)*R2G))*R1DDET
            DPSTRS(JJ,MM)=BULK-R2GD3+(AUX3*(((R2BULK*(DRVAB-DRVBB-DRVAA+
     1                   DRVBA)+(DRVAB-DRVAA)*R2GD3+(DRVBB-DRVBA)*
     2                   R4GD3)*SINPHI)+(DRVAB-DRVAA)*R2G))*R1DDET
            DPSTRS(JJ,JJ)=BULK+R4GD3+(AUX3*(DRVAB-DRVBB-DRVAA+DRVBA)*
     1                   (((R2BULK+R2GD3)*SINPHI)-R2G))*R1DDET
          ENDIF
        ELSEIF(APEX)THEN
C Tangent consistent with multi-vector return to apex
          COTPHI=COSPHI/SINPHI
          DSIDEJ=BULK*(R1-(BULK/(BULK+
     1           DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))*COTPHI*COSPHI/
     2           SINPSI)))
          DPSTRS(II,II)=DSIDEJ
          DPSTRS(II,MM)=DSIDEJ
          DPSTRS(II,JJ)=DSIDEJ
          DPSTRS(MM,II)=DSIDEJ
          DPSTRS(MM,MM)=DSIDEJ
          DPSTRS(MM,JJ)=DSIDEJ
          DPSTRS(JJ,II)=DSIDEJ
          DPSTRS(JJ,MM)=DSIDEJ
          DPSTRS(JJ,JJ)=DSIDEJ
        ELSE
C Tangent consistent with 1-vector return to main active plane
          SPHSPS=SINPHI*SINPSI
          CONSTA=R4G*(R1+R1D3*SPHSPS)+R4*BULK*SPHSPS
          DENOM=-CONSTA-R4C2PH*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          B1=(R2G*(R1+R1D3*SINPSI)+R2BULK*SINPSI)/DENOM
          B2=(R4G*R1D3-R2BULK)*SINPSI/DENOM
          B3=(R2G*(R1-R1D3*SINPSI)-R2BULK*SINPSI)/DENOM
          DPSTRS(II,II)=R2G*(R2D3+B1*(R1+R1D3*SINPHI))+
     1                 BULK*(R1+R2*B1*SINPHI)
          DPSTRS(II,MM)=R1D3*(R3*BULK-R2G)*(R1+R2*B1*SINPHI)
          DPSTRS(II,JJ)=R2G*(-R1D3-B1*(R1-R1D3*SINPHI))+
     1                 BULK*(R1+R2*B1*SINPHI)
          DPSTRS(MM,II)=R2G*(-R1D3-B2*(R1+R1D3*SINPHI))+
     1                 BULK*(R1-R2*B2*SINPHI)
          DPSTRS(MM,MM)=R4G*R1D3*(R1+B2*SINPHI)+BULK*(R1-R2*B2*SINPHI)
          DPSTRS(MM,JJ)=R2G*(-R1D3+B2*(R1-R1D3*SINPHI))+
     1                 BULK*(R1-R2*B2*SINPHI)
          DPSTRS(JJ,II)=R2G*(-R1D3-B3*(R1+R1D3*SINPHI))+
     1                 BULK*(R1-R2*B3*SINPHI)
          DPSTRS(JJ,MM)=R1D3*(R3*BULK-R2G)*(R1-R2*B3*SINPHI)
          DPSTRS(JJ,JJ)=R2G*(R2D3+B3*(R1-R1D3*SINPHI))+
     1                 BULK*(R1-R2*B3*SINPHI)
        ENDIF
C
        IF(NTYPE.EQ.2)THEN
          OUTOFP=.FALSE.
        ELSEIF(NTYPE.EQ.3)THEN
          OUTOFP=.TRUE.
        ENDIF
        CALL DGISO2
```

```
     1(   DPSTRS     ,DMATX      ,EIGPRJ     ,PSTRA      ,PSTRS      ,
     2    OUTOFP     ,REPEAT     )
      ELSE
C Compute elasticity matrix
C -------------------------
        IF(NTYPE.EQ.2)THEN
          NSTRE=3
        ELSEIF(NTYPE.EQ.3)THEN
          NSTRE=4
        ENDIF
        FACTOR=BULK-R2G*R1D3
        DO 50 I=1,NSTRE
          DO 40 J=I,NSTRE
            DMATX(I,J)=R2G*FOID(I,J)+FACTOR*SOID(I)*SOID(J)
   40     CONTINUE
   50   CONTINUE
        DO 70 J=1,NSTRE-1
          DO 60 I=J+1,NSTRE
            DMATX(I,J)=DMATX(J,I)
   60     CONTINUE
   70   CONTINUE
      ENDIF
      RETURN
      END
```

```fortran
      SUBROUTINE CTTR
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MDIM=3,  MSTRE=4)
      LOGICAL EPFLAG, LALGVA(4), OUTOFP, RIGHT, REPEAT, TWOVEC
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(*)           ,STRES(*)
      DIMENSION
     1    DPSTRS(MDIM,MDIM)  ,DPSTRE(MDIM,MDIM)  ,EIGPRJ(MSTRE,2)    ,
     2    FOID(MSTRE,MSTRE)  ,PSTRS(MDIM)        ,PSTRA(MDIM)        ,
     3    SOID(MSTRE)        ,STRAC(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,R3   ,R4   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,3.0D0,4.0D0/
C***********************************************************************
C COMPUTATION OF CONSISTENT TANGENT MODULUS FOR TRESCA TYPE
C ELASTO-PLASTIC MATERIAL WITH PIECE-WISE LINEAR ISOTROPIC HARDENING.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 8.1.5
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0028')
C Current accumulated plastic strain
      EPBAR=RSTAVA(MSTRE+1)
C Set material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Set needed algorithmic variables
      TWOVEC=LALGVA(3)
      RIGHT=LALGVA(4)
C Set some constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R4G=R4*GMODU
      R1D3=R1/R3
      R2D3=R2*R1D3
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent
C ---------------------------------------
C Spectral decomposition of the elastic trial strain
        STRAC(1)=STRAT(1)
        STRAC(2)=STRAT(2)
        STRAC(3)=STRAT(3)*RP5
        CALL SPDEC2(EIGPRJ,PSTRA,REPEAT,STRAC)
        PSTRA(3)=STRAT(4)
C and current total stress
        PSTRS(1)=STRES(1)*EIGPRJ(1,1)+STRES(2)*EIGPRJ(2,1)+
     1                           R2*STRES(3)*EIGPRJ(3,1)
        PSTRS(2)=STRES(1)*EIGPRJ(1,2)+STRES(2)*EIGPRJ(2,2)+
     1                           R2*STRES(3)*EIGPRJ(3,2)
        PSTRS(3)=STRES(4)
C Identify directions of maximum and minimum principal trial stresses
        II=1
        JJ=1
        PSTMAX=PSTRA(II)
        PSTMIN=PSTRA(JJ)
        DO 10 I=2,3
          IF(PSTRA(I).GE.PSTMAX)THEN
            II=I
            PSTMAX=PSTRA(II)
          ENDIF
          IF(PSTRA(I).LT.PSTMIN)THEN
            JJ=I
            PSTMIN=PSTRA(JJ)
          ENDIF
   10   CONTINUE
        IF(II.NE.1.AND.JJ.NE.1)MM=1
        IF(II.NE.2.AND.JJ.NE.2)MM=2
        IF(II.NE.3.AND.JJ.NE.3)MM=3
        IF(TWOVEC)THEN
C Tangent consistent with two-vector return algorithm
          HSLOPE=DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          DAA=R4G+HSLOPE
          DAB=R2G+HSLOPE
          DBA=R2G+HSLOPE
          DBB=R4G+HSLOPE
          DET=DAA*DBB-DAB*DBA
          R2GDD=R2G/DET
          R4G2DD=R2G*R2GDD
          IF(RIGHT)THEN
C ...returned to right corner
            DPSTRS(II,II)=R2G*(R1-R2GDD*R4G)
            DPSTRS(II,MM)=R4G2DD*(DAA-DAB)
            DPSTRS(II,JJ)=R4G2DD*(DBB-DBA)
            DPSTRS(MM,II)=R4G2DD*R2G
            DPSTRS(MM,MM)=R2G*(R1-R2GDD*DAA)
            DPSTRS(MM,JJ)=R4G2DD*DBA
            DPSTRS(JJ,II)=R4G2DD*R2G
            DPSTRS(JJ,MM)=R4G2DD*DAB
            DPSTRS(JJ,JJ)=R2G*(R1-R2GDD*DBB)
          ELSE
C ...returned to left corner
            DPSTRS(II,II)=R2G*(R1-R2GDD*DBB)
            DPSTRS(II,MM)=R4G2DD*DAB
            DPSTRS(II,JJ)=R4G2DD*R2G
```

```
            DPSTRS(MM,II)=R4G2DD*DBA
            DPSTRS(MM,MM)=R2G*(R1-R2GDD*DAA)
            DPSTRS(MM,JJ)=R4G2DD*R2G
            DPSTRS(JJ,II)=R4G2DD*(DBB-DBA)
            DPSTRS(JJ,MM)=R4G2DD*(DAA-DAB)
            DPSTRS(JJ,JJ)=R2G*(R1-R2GDD*R4G)
          ENDIF
        ELSE
C Tangent consistent with one-vector return algorithm
          FACTOR=R2G/(R4G+DPLFUN(EPBAR,NHARD,RPROPS(IPHARD)))
          DPSTRS(II,II)=R2G*(R1-FACTOR)
          DPSTRS(II,MM)=R0
          DPSTRS(II,JJ)=R2G*FACTOR
          DPSTRS(MM,II)=DPSTRS(II,MM)
          DPSTRS(MM,MM)=R2G
          DPSTRS(MM,JJ)=R0
          DPSTRS(JJ,II)=DPSTRS(II,JJ)
          DPSTRS(JJ,MM)=DPSTRS(MM,JJ)
          DPSTRS(JJ,JJ)=DPSTRS(II,II)
        ENDIF
        DPSTRE(1,1)=+DPSTRS(1,1)*R2D3-DPSTRS(1,2)*R1D3-DPSTRS(1,3)*R1D3+
     1                                                           BULK
        DPSTRE(2,1)=+DPSTRS(2,1)*R2D3-DPSTRS(2,2)*R1D3-DPSTRS(2,3)*R1D3+
     1                                                           BULK
        DPSTRE(3,1)=+DPSTRS(3,1)*R2D3-DPSTRS(3,2)*R1D3-DPSTRS(3,3)*R1D3+
     1                                                           BULK
        DPSTRE(1,2)=-DPSTRS(1,1)*R1D3+DPSTRS(1,2)*R2D3-DPSTRS(1,3)*R1D3+
     1                                                           BULK
        DPSTRE(2,2)=-DPSTRS(2,1)*R1D3+DPSTRS(2,2)*R2D3-DPSTRS(2,3)*R1D3+
     1                                                           BULK
        DPSTRE(3,2)=-DPSTRS(3,1)*R1D3+DPSTRS(3,2)*R2D3-DPSTRS(3,3)*R1D3+
     1                                                           BULK
        DPSTRE(1,3)=-DPSTRS(1,1)*R1D3-DPSTRS(1,2)*R1D3+DPSTRS(1,3)*R2D3+
     1                                                           BULK
        DPSTRE(2,3)=-DPSTRS(2,1)*R1D3-DPSTRS(2,2)*R1D3+DPSTRS(2,3)*R2D3+
     1                                                           BULK
        DPSTRE(3,3)=-DPSTRS(3,1)*R1D3-DPSTRS(3,2)*R1D3+DPSTRS(3,3)*R2D3+
     1                                                           BULK
        IF(NTYPE.EQ.2)THEN
          OUTOFP=.FALSE.
        ELSEIF(NTYPE.EQ.3)THEN
          OUTOFP=.TRUE.
        ENDIF
        CALL DGISO2
     1(   DPSTRE     ,DMATX      ,EIGPRJ     ,PSTRA      ,PSTRS      ,
     2    OUTOFP     ,REPEAT     )
      ELSE
C Compute elasticity matrix
C -------------------------
        IF(NTYPE.EQ.2)THEN
          NSTRE=3
        ELSEIF(NTYPE.EQ.3)THEN
          NSTRE=4
        ENDIF
C
        FACTOR=BULK-R2G*R1D3
        DO 50 I=1,NSTRE
          DO 40 J=I,NSTRE
            DMATX(I,J)=R2G*FOID(I,J)+FACTOR*SOID(I)*SOID(J)
   40     CONTINUE
   50   CONTINUE
        DO 70 J=1,NSTRE-1
          DO 60 I=J+1,NSTRE
            DMATX(I,J)=DMATX(J,I)
   60     CONTINUE
   70   CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE CTTRPN
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(  MSTRE=4 )
C Arguments
      LOGICAL EPFLAG, LALGVA(4)
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(*)           ,STRES(*)
C Local arrays and variables
      DIMENSION
     1    D12(3)             ,D21(3)
C***********************************************************************
C COMPUTATION OF CONSISTENT TANGENT MODULUS FOR TRESCA TYPE
C ELASTO-PLASTIC MATERIAL WITH PIECE-WISE LINEAR ISOTROPIC HARDENING.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 9.2
C***********************************************************************
      STRAT(4)=RSTAVA(4)
C Compute the axisymmetric tangent matrix
      NTYPAX=3
      CALL CTTR
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,NTYPAX     ,
     2    RPROPS     ,RSTAVA     ,STRAT      ,STRES      )
C Decompose into submatrices
      D12(1)=DMATX(1,4)
      D12(2)=DMATX(2,4)
      D12(3)=DMATX(3,4)
      D21(1)=DMATX(4,1)
      D21(2)=DMATX(4,2)
      D21(3)=DMATX(4,3)
      D22=DMATX(4,4)
C Assemble plane stress consistent tangent matrix
      DO 20 I=1,3
        DO 10 J=1,3
          DMATX(I,J)=DMATX(I,J)-D12(I)*D21(J)/D22
   10   CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CTVM
     1(   DGAMA      ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
      LOGICAL EPFLAG
      DIMENSION
     1    DMATX(MSTRE,MSTRE),IPROPS(*)           ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)   ,STRES(MSTRE)
      DIMENSION
     1    DEVPRJ(MSTRE,MSTRE),FOID(MSTRE,MSTRE)  ,S(MSTRE)           ,
     2    SOID(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    R1   ,R2   ,R3   ,R6   /
     2    1.0D0,2.0D0,3.0D0,6.0D0/
C***********************************************************************
C COMPUTATION OF THE CONSISTENT TANGENT MODULUS FOR VON MISES TYPE
C ELASTO-PLASTIC MATERIAL WITH PIECE-WISE LINEAR ISOTROPIC HARDENING.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 7.4.3
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0030')
C Current accumulated plastic strain
      EPBAR=RSTAVA(MSTRE+1)
C Set material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Shear and bulk moduli
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R1D3=R1/R3
C Set deviatoric projection tensor
      IF(NTYPE.EQ.2)THEN
        NSTRE=3
      ELSEIF(NTYPE.EQ.3)THEN
        NSTRE=4
      ENDIF
      DO 20 I=1,NSTRE
        DO 10 J=1,NSTRE
          DEVPRJ(I,J)=FOID(I,J)-SOID(I)*SOID(J)*R1D3
   10   CONTINUE
   20 CONTINUE
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent
C ----------------------------------------
        R3G=R3*GMODU
        ROO3D2=SQRT(R3/R2)
C Hydrostatic pressure
        P=(STRES(1)+STRES(2)+STRES(4))*R1D3
C Deviatoric stress components
        S(1)=STRES(1)-P
        S(2)=STRES(2)-P
        S(3)=STRES(3)
        S(4)=STRES(4)-P
C Recover last elastic trial von Mises effective stress
        SNORM=SQRT(S(1)*S(1)+S(2)*S(2)+R2*S(3)*S(3)+S(4)*S(4))
        Q=ROO3D2*SNORM
        QTRIAL=Q+R3G*DGAMA
C Assemble elastoplastic tangent (upper triangle only)
        AFACT=R2G*(R1-R3G*DGAMA/QTRIAL)
        BFACT=R6*GMODU*GMODU*(DGAMA/QTRIAL-
     1        R1/(R3G+DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))))/
     2        (SNORM*SNORM)
        DO 40 I=1,NSTRE
          DO 30 J=I,NSTRE
            DMATX(I,J)=AFACT*DEVPRJ(I,J)+BFACT*S(I)*S(J)+
     1                 BULK*SOID(I)*SOID(J)
   30     CONTINUE
   40   CONTINUE
      ELSE
C Compute elasticity matrix (upper triangle only)
C -----------------------------------------------
        DO 60 I=1,NSTRE
          DO 50 J=I,NSTRE
            DMATX(I,J)=R2G*DEVPRJ(I,J)+BULK*SOID(I)*SOID(J)
   50     CONTINUE
   60   CONTINUE
      ENDIF
C Assemble lower triangle
C -----------------------
      DO 80 J=1,NSTRE-1
        DO 70 I=J+1,NSTRE
          DMATX(I,J)=DMATX(J,I)
   70   CONTINUE
   80 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CTVMMX
     1(   DGAMA      ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,RSTAV2     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
C Arguments
      LOGICAL EPFLAG
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(2*MSTRE+1)  ,RSTAV2(2*MSTRE+1)  ,STRES(MSTRE)
C Local arrays and variables
      DIMENSION
     1    BACSTR(MSTRE)      ,DEVPRJ(MSTRE,MSTRE),ETA(MSTRE)         ,
     2    FOID(MSTRE,MSTRE)  ,S(MSTRE)           ,SOID(MSTRE)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    R1   ,R2   ,R3   ,R6   /
     2    1.0D0,2.0D0,3.0D0,6.0D0/
C***********************************************************************
C COMPUTATION OF THE CONSISTENT TANGENT MODULUS FOR VON MISES TYPE
C ELASTO-PLASTIC WITH PIECE-WISE LINEAR MIXED ISOTROPIC/KINEMATIC
C HARDENING.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 7.6.6
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0048')
C Retrieve current accumulated plastic strain
      EPBAR=RSTAVA(MSTRE+1)
C Retrieve last converged accumulated plastic strain
      EPBARN=RSTAV2(MSTRE+1)
C Retrieve current backstress tensor components
      BACSTR(1)=RSTAVA(6)
      BACSTR(2)=RSTAVA(7)
      BACSTR(3)=RSTAVA(8)
      BACSTR(4)=RSTAVA(9)
C Set material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Set pointers to isotropic and kinematic hardening curves
      IPIHAR=IPHARD
      IPKHAR=IPHARD+2*NHARD
C Shear and bulk moduli
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R1D3=R1/R3
C Set deviatoric projection tensor
      IF(NTYPE.EQ.2)THEN
        NSTRE=3
      ELSEIF(NTYPE.EQ.3)THEN
        NSTRE=4
      ENDIF
      DO 20 I=1,NSTRE
        DO 10 J=1,NSTRE
          DEVPRJ(I,J)=FOID(I,J)-SOID(I)*SOID(J)*R1D3
   10   CONTINUE
   20 CONTINUE
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent
C ---------------------------------------
        R3G=R3*GMODU
        ROO3D2=SQRT(R3/R2)
C Hydrostatic pressure
        P=(STRES(1)+STRES(2)+STRES(4))*R1D3
C Deviatoric stress components
        S(1)=STRES(1)-P
        S(2)=STRES(2)-P
        S(3)=STRES(3)
        S(4)=STRES(4)-P
C Relative stress components
        DO 30 ISTRE=1,MSTRE
          ETA(ISTRE)=S(ISTRE)-BACSTR(ISTRE)
   30   CONTINUE
        ETANOR=SQRT(ETA(1)**2+ETA(2)**2+R2*ETA(3)**2+ETA(4)**2)
C Recover last elastic trial relative effective stress
        QBAR=ROO3D2*ETANOR
        BETBAN=PLFUN(EPBARN,NHARD,RPROPS(IPKHAR))
        BETBAR=PLFUN(EPBAR,NHARD,RPROPS(IPKHAR))
        QBARTR=QBAR+R3G*DGAMA+BETBAR-BETBAN
C Assemble elastoplastic tangent (upper triangle only)
        HISLOP=DPLFUN(EPBAR,NHARD,RPROPS(IPIHAR))
        AFACT=R2G*(R1-R3G*DGAMA/QBARTR)
        HKSLOP=DPLFUN(EPBAR,NHARD,RPROPS(IPKHAR))
        BFACT=R6*GMODU*GMODU*(DGAMA/QBARTR-R1/(R3G+HISLOP+HKSLOP))/
     1        (ETANOR*ETANOR)
        DO 50 I=1,NSTRE
          DO 40 J=I,NSTRE
            DMATX(I,J)=AFACT*DEVPRJ(I,J)+BFACT*ETA(I)*ETA(J)+
     1                 BULK*SOID(I)*SOID(J)
   40     CONTINUE
   50   CONTINUE
      ELSE
C Compute elasticity matrix (upper triangle only)
C -----------------------------------------------
        DO 70 I=1,NSTRE
          DO 60 J=I,NSTRE
            DMATX(I,J)=R2G*DEVPRJ(I,J)+BULK*SOID(I)*SOID(J)
   60     CONTINUE
   70   CONTINUE
```

```
      ENDIF
C Assemble lower triangle
C -----------------------
      DO 90 J=1,NSTRE-1
        DO 80 I=J+1,NSTRE
          DMATX(I,J)=DMATX(J,I)
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CTVMPS
     1(   DGAMA      ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
      LOGICAL EPFLAG
C Array arguments
      DIMENSION
     1    DMATX(MSTRE,MSTRE),IPROPS(*)           ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)   ,STRES(MSTRE)
C Local arrays
      DIMENSION
     1    FOID(MSTRE,MSTRE)  ,SOID(MSTRE)        ,VECN(3)
      DATA
     1    FOID(1,1),FOID(1,2),FOID(1,3),FOID(1,4)/
     2    1.0D0    ,0.0D0    ,0.0D0    ,0.0D0    /
     3    FOID(2,1),FOID(2,2),FOID(2,3),FOID(2,4)/
     4    0.0D0    ,1.0D0    ,0.0D0    ,0.0D0    /
     5    FOID(3,1),FOID(3,2),FOID(3,3),FOID(3,4)/
     6    0.0D0    ,0.0D0    ,0.5D0    ,0.0D0    /
     7    FOID(4,1),FOID(4,2),FOID(4,3),FOID(4,4)/
     8    0.0D0    ,0.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    SOID(1)  ,SOID(2)  ,SOID(3)  ,SOID(4)  /
     2    1.0D0    ,1.0D0    ,0.0D0    ,1.0D0    /
      DATA
     1    RP5  ,R1   ,R2   ,R3   ,R4   /
     2    0.5D0,1.0D0,2.0D0,3.0D0,4.0D0/
C***********************************************************************
C COMPUTATION OF THE CONSISTENT TANGENT MODULUS FOR VON MISES TYPE
C ELASTO-PLASTIC MATERIAL WITH PIECE-WISE LINEAR ISOTROPIC HARDENING.
C PLANE STRESS IMPLEMENTATION ONLY.
C
C REFERENCE: Section 9.4.5
C***********************************************************************
C Stops program if neither not plane stress
      IF(NTYPE.NE.1)CALL ERRPRT('EI0032')
C Current accumulated plastic strain
      EPBAR=RSTAVA(MSTRE+1)
C Set material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Shear and bulk moduli
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R1D3=R1/R3
      R2D3=R2*R1D3
      IF(EPFLAG)THEN
C Compute elastoplastic consistent tangent (Box 9.6)
C ===================================================
C Item (i):
C ---------
C Compute XI
        XI=R2D3*(STRES(1)*STRES(1)+STRES(2)*STRES(2)-STRES(1)*STRES(2))+
     1     R2*STRES(3)*STRES(3)
C Hardening slope
        HSLOPE=DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
C Matrix E components
        ESTAR1=R3*YOUNG/(R3*(R1-POISS)+YOUNG*DGAMA)
        ESTAR2=R2G/(R1+R2G*DGAMA)
        ESTAR3=GMODU/(R1+R2G*DGAMA)
        E11=RP5*(ESTAR1+ESTAR2)
        E22=E11
        E12=RP5*(ESTAR1-ESTAR2)
        E33=ESTAR3
C Components of the matrix product EP
        EPSTA1=R1D3*ESTAR1
        EPSTA2=ESTAR2
        EPSTA3=EPSTA2
        EP11=RP5*(EPSTA1+EPSTA2)
        EP22=EP11
        EP12=RP5*(EPSTA1-EPSTA2)
        EP21=EP12
        EP33=EPSTA3
C Vector n
        VECN(1)=EP11*STRES(1)+EP12*STRES(2)
        VECN(2)=EP21*STRES(1)+EP22*STRES(2)
        VECN(3)=EP33*STRES(3)
C Scalar alpha
        DENOM1=STRES(1)*(R2D3*VECN(1)-R1D3*VECN(2))+
     1         STRES(2)*(R2D3*VECN(2)-R1D3*VECN(1))+
     2         STRES(3)*R2*VECN(3)
        DENOM2=R2*XI*HSLOPE/(R3-R2*HSLOPE*DGAMA)
        ALPHA=R1/(DENOM1+DENOM2)
C Item (ii): Assemble elasto-plastic tangent
C ------------------------------------------
        DMATX(1,1)=E11-ALPHA*VECN(1)*VECN(1)
        DMATX(1,2)=E12-ALPHA*VECN(1)*VECN(2)
        DMATX(1,3)=-ALPHA*VECN(1)*VECN(3)
        DMATX(2,1)=DMATX(1,2)
        DMATX(2,2)=E22-ALPHA*VECN(2)*VECN(2)
        DMATX(2,3)=-ALPHA*VECN(2)*VECN(3)
        DMATX(3,1)=DMATX(1,3)
        DMATX(3,2)=DMATX(2,3)
        DMATX(3,3)=E33-ALPHA*VECN(3)*VECN(3)
      ELSE
C Compute plane stress elasticity matrix
C ======================================
        NSTRE=3
        R4GD3=R4*GMODU/R3
        FACTOR=(BULK-R2G/R3)*(R2G/(BULK+R4GD3))
        DO 20 I=1,NSTRE
          DO 10 J=I,NSTRE
            DMATX(I,J)=R2G*FOID(I,J)+FACTOR*SOID(I)*SOID(J)
   10     CONTINUE
   20   CONTINUE
C lower triangle
        DO 40 J=1,NSTRE-1
          DO 30 I=J+1,NSTRE
            DMATX(I,J)=DMATX(J,I)
```

```
 30      CONTINUE
 40    CONTINUE
      ENDIF
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION DDLGD2(X)
C
      DOUBLE PRECISION X
      DOUBLE PRECISION RP5
      DATA   RP5 /0.5D0/
C***********************************************************************
C DERIVATIVE OF THE SCALAR FUNCTION 'DLGD2' THAT RELATES PRINCIPAL
C LOGARITHMIC STRECTHES AND EIGENVALUES OF THE CAUCHY-GREEN TENSOR
C***********************************************************************
      DDLGD2=RP5/X
C
      RETURN
      END
```

```
      SUBROUTINE DEFGRA
     1(   ELDISP     ,F          ,GMATX      ,MDOFN      ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MCOMP=5    )
      DIMENSION
     1    ELDISP(MDOFN,*)    ,F(3,3)             ,GMATX(MGDIM,*)
      DIMENSION
     1    FVEC(MCOMP)
      DATA  R0   ,R1    / 0.0D0,1.0D0 /
C***********************************************************************
C COMPUTES THE DEFORMATION GRADIENT TENSOR ASSOCIATED WITH THE ELEMENT
C DISPLACEMENT 'ELDISP'
C***********************************************************************
C Set total number of deformation gradient components
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2)THEN
        NCOMP=4
      ELSEIF(NTYPE.EQ.3)THEN
        NCOMP=5
      ELSE
        CALL ERRPRT('EI0021')
      ENDIF
C Evaluate the deformation gradient stored in vector form
      CALL RVZERO(FVEC,NCOMP)
      DO 30 ICOMP=1,NCOMP
        IEVAB=0
        DO 20 INODE=1,NNODE
          DO 10 IDOFN=1,NDOFN
            IEVAB=IEVAB+1
            FVEC(ICOMP)=FVEC(ICOMP)+
     1                  GMATX(ICOMP,IEVAB)*ELDISP(IDOFN,INODE)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C Store the deformation gradient in matrix form
      F(1,1)=FVEC(1)+R1
      F(2,1)=FVEC(2)
      F(3,1)=R0
      F(1,2)=FVEC(3)
      F(2,2)=FVEC(4)+R1
      F(3,2)=R0
      F(1,3)=R0
      F(2,3)=R0
      IF(NTYPE.EQ.1)THEN
        F(3,3)=R0
      ELSEIF(NTYPE.EQ.2)THEN
        F(3,3)=R1
      ELSEIF(NTYPE.EQ.3)THEN
        F(3,3)=FVEC(5)+R1
      ENDIF
C
      RETURN
      END
```

```fortran
      SUBROUTINE DEXPMP
     1(   DEXPX      ,NOCONV     ,X          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   NDIM=3     ,NDIM2=9    ,NDIM4=81   ,MAXN=100   )
C Arguments
      LOGICAL  NOCONV
      DIMENSION
     1    DEXPX(NDIM,NDIM,NDIM,NDIM),X(NDIM,NDIM)
C Local arrays and variables
C...matrix of powers of X
      DIMENSION
     1    R1DFAC(MAXN)       ,XMATX(NDIM,NDIM,0:MAXN)
C...initialise identity matrix: X to the power 0
      DATA
     1    XMATX(1,1,0)  ,XMATX(1,2,0)  ,XMATX(1,3,0)  /
     2    1.D0          ,0.D0          ,0.D0          /
     3    XMATX(2,1,0)  ,XMATX(2,2,0)  ,XMATX(2,3,0)  /
     4    0.D0          ,1.D0          ,0.D0          /
     5    XMATX(3,1,0)  ,XMATX(3,2,0)  ,XMATX(3,3,0)  /
     6    0.D0          ,0.D0          ,1.D0          /
      DATA
     1    R0   ,RP5  ,R1   ,TOL    ,OVER    ,UNDER   /
     2    0.0D0,0.5D0,1.0D0,1.0D-10,1.0D+100,1.0D-100/
C***********************************************************************
C COMPUTES THE DERIVATIVE OF THE EXPONENTIAL OF A (GENERALLY
C UNSYMMETRIC) 3-D TENSOR. USES THE SERIES REPRESENTATION OF THE TENSOR
C EXPONENTIAL.
C
C REFERENCE: Section B.2
C            Box B.2
C***********************************************************************
C Initialise convergence flag
      NOCONV=.FALSE.
C X to the power 1
      DO 20 I=1,NDIM
        DO 10 J=1,NDIM
          XMATX(I,J,1)=X(I,J)
   10   CONTINUE
   20 CONTINUE
C Zero remaining powers of X
      CALL RVZERO(XMATX(1,1,2),NDIM*NDIM*(MAXN-1))
C Compute X square
      DO 50 I=1,NDIM
        DO 40 J=1,NDIM
          DO 30 K=1,NDIM
            XMATX(I,J,2)=XMATX(I,J,2)+X(I,K)*X(K,J)
   30     CONTINUE
   40   CONTINUE
   50 CONTINUE
C Compute principal invariants of X
      C1=X(1,1)+X(2,2)+X(3,3)
      C2=RP5*(C1*C1-(XMATX(1,1,2)+XMATX(2,2,2)+XMATX(3,3,2)))
      C3=X(1,1)*X(2,2)*X(3,3)+X(1,2)*X(2,3)*X(3,1)+
     1   X(1,3)*X(2,1)*X(3,2)-X(1,2)*X(2,1)*X(3,3)-
     2   X(1,1)*X(2,3)*X(3,2)-X(1,3)*X(2,2)*X(3,1)
C Compute X to the powers 3,4,...,NMAX using recursive formula
      R1DFAC(1)=R1
      R1DFAC(2)=RP5
      DO 80 N=3,MAXN
        R1DFAC(N)=R1DFAC(N-1)/DBLE(N)
        DO 70 I=1,NDIM
          DO 60 J=1,NDIM
            XMATX(I,J,N)=C1*XMATX(I,J,N-1)-C2*XMATX(I,J,N-2)+
     1                   C3*XMATX(I,J,N-3)
   60     CONTINUE
   70   CONTINUE
        XNNORM=SQRT(SCAPRD(XMATX(1,1,N),XMATX(1,1,N),NDIM2))
C...check number of terms required for series convergence
        IF(XNNORM.GT.OVER.OR.(XNNORM.LT.UNDER.AND.XNNORM.GT.R0)
     1                                 .OR.R1DFAC(N).LT.UNDER)THEN
C...numbers are to small or too big: Exit without computing derivative
          NOCONV=.TRUE.
          GOTO 999
        ELSEIF(XNNORM*R1DFAC(N).LT.TOL)THEN
C...series will converge with NMAX terms:
C   Carry on to derivative computation
          NMAX=N
          GOTO 90
        ENDIF
   80 CONTINUE
C...series will not converge for the currently prescribed tolerance
C   with the currently prescribed maximum number of terms MAXN:
C   Exit without computing derivative
      NOCONV=.TRUE.
      GOTO 999
   90 CONTINUE
C Compute the derivative of exponential map
      CALL RVZERO(DEXPX,NDIM4)
      DO 150 I=1,NDIM
        DO 140 J=1,NDIM
          DO 130 K=1,NDIM
            DO 120 L=1,NDIM
              DO 110 N=1,NMAX
                DO 100 M=1,N
                  DEXPX(I,J,K,L)=DEXPX(I,J,K,L)+
     1                           R1DFAC(N)*XMATX(I,K,M-1)*XMATX(L,J,N-M)
  100           CONTINUE
  110         CONTINUE
  120       CONTINUE
  130     CONTINUE
  140   CONTINUE
  150 CONTINUE
C
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DGISO2
     1(   DEIGY      ,DYDX       ,EIGPRJ     ,EIGX       ,EIGY       ,
     2    OUTOFP     ,REPEAT     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MCOMP=4    ,MDIM=3     ,NDIM=2     )
C Arguments
      LOGICAL OUTOFP, REPEAT
      DIMENSION
     1    DEIGY(MDIM,MDIM)  ,DYDX(MCOMP,MCOMP) ,EIGPRJ(MCOMP,NDIM),
     2    EIGX(NDIM)        ,EIGY(NDIM)
C Local arrays
      DIMENSION
     1    EIGPR3(MCOMP)     ,FOID(MCOMP,MCOMP) ,SOPID(MCOMP)
      DATA
     1    FOID(1,1)     ,FOID(1,2)     ,FOID(1,3)     /
     2    1.0D0         ,0.0D0         ,0.0D0         /
     3    FOID(2,1)     ,FOID(2,2)     ,FOID(2,3)     /
     4    0.0D0         ,1.0D0         ,0.0D0         /
     5    FOID(3,1)     ,FOID(3,2)     ,FOID(3,3)     /
     6    0.0D0         ,0.0D0         ,0.5D0         /
      DATA
     1    SOPID(1)      ,SOPID(2)      ,SOPID(3)      ,SOPID(4)        /
     2    1.0D0         ,1.0D0         ,0.0D0         ,0.0D0           /
      DATA
     1    EIGPR3(1)     ,EIGPR3(2)     ,EIGPR3(3)     ,EIGPR3(4)       /
     2    0.0D0         ,0.0D0         ,0.0D0         ,1.0D0           /
C***********************************************************************
C COMPUTE THE DERIVATIVE OF A GENERAL ISOTROPIC TENSOR FUNCTION OF ONE
C TENSOR IN 2-D (WITH ONE POSSIBLE OUT-OF-PLANE COMPONENT)
C
C REFERENCE: Sections A.3.1-2
C            Box A.2
C***********************************************************************
      CALL RVZERO(DYDX,MCOMP*MCOMP)
      IF(REPEAT)THEN
C Derivative dY/dX for repeated in-plane eigenvalues of X
C -----------------------------------------------------
C In-plane component
        DO 20 I=1,3
          DO 10 J=1,3
            DYDX(I,J)=(DEIGY(1,1)-DEIGY(1,2))*FOID(I,J)+
     1                DEIGY(1,2)*SOPID(I)*SOPID(J)
   10     CONTINUE
   20   CONTINUE
        IF(OUTOFP)THEN
C out-of-plane components required
          DO 40 I=1,4
            DO 30 J=1,4
              IF(I.EQ.4.OR.J.EQ.4)DYDX(I,J)=
     1                  DEIGY(1,3)*SOPID(I)*EIGPR3(J)+
     2                  DEIGY(3,1)*EIGPR3(I)*SOPID(J)+
     3                  DEIGY(3,3)*EIGPR3(I)*EIGPR3(J)
   30       CONTINUE
   40     CONTINUE
        ENDIF
      ELSE
C Derivative dY/dX for distinct in-plane eigenvalues of X
C -----------------------------------------------------
C Assemble in-plane DYDX
        A1=(EIGY(1)-EIGY(2))/(EIGX(1)-EIGX(2))
        DO 70 I=1,3
          DO 60 J=1,3
            DYDX(I,J)=A1*(FOID(I,J)-EIGPRJ(I,1)*EIGPRJ(J,1)-
     1               EIGPRJ(I,2)*EIGPRJ(J,2))+
     2               DEIGY(1,1)*EIGPRJ(I,1)*EIGPRJ(J,1)+
     3               DEIGY(1,2)*EIGPRJ(I,1)*EIGPRJ(J,2)+
     4               DEIGY(2,1)*EIGPRJ(I,2)*EIGPRJ(J,1)+
     5               DEIGY(2,2)*EIGPRJ(I,2)*EIGPRJ(J,2)
   60     CONTINUE
   70   CONTINUE
        IF(OUTOFP) THEN
C out-of-plane components required
          DO 90 I=1,4
            DO 80 J=1,4
              IF(I.EQ.4.OR.J.EQ.4)DYDX(I,J)=
     1               DEIGY(1,3)*EIGPRJ(I,1)*EIGPR3(J)+
     2               DEIGY(2,3)*EIGPRJ(I,2)*EIGPR3(J)+
     3               DEIGY(3,1)*EIGPR3(I)*EIGPRJ(J,1)+
     4               DEIGY(3,2)*EIGPR3(I)*EIGPRJ(J,2)+
     5               DEIGY(3,3)*EIGPR3(I)*EIGPR3(J)
   80       CONTINUE
   90     CONTINUE
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE DISO2
     1(   DYDX       ,DFUNC      ,FUNC       ,OUTOFP     ,X          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL
     1    DFUNC      ,FUNC
      PARAMETER
     1(   MCOMP=4    ,NDIM=2     )
C Arguments
      LOGICAL OUTOFP
      DIMENSION
     1    DYDX(MCOMP,MCOMP) ,X(*)
C Local variables and arrays
      LOGICAL REPEAT
      DIMENSION
     1    DEIGY(NDIM)       ,EIGPRJ(MCOMP,NDIM),EIGX(NDIM)        ,
     2    EIGY(NDIM)        ,FOID(MCOMP,MCOMP)
      DATA
     1    FOID(1,1)     ,FOID(1,2)     ,FOID(1,3)     /
     2    1.0D0         ,0.0D0         ,0.0D0         /
     3    FOID(2,1)     ,FOID(2,2)     ,FOID(2,3)     /
     4    0.0D0         ,1.0D0         ,0.0D0         /
     5    FOID(3,1)     ,FOID(3,2)     ,FOID(3,3)     /
     6    0.0D0         ,0.0D0         ,0.5D0         /
C***********************************************************************
C COMPUTE (AND STORE IN MATRIX FORM) THE DERIVATIVE dY/dX OF AN
C ISOTROPIC TENSOR FUNCTION OF THE TYPE:
C
C                         Y(X) = sum{ y(x_i) E_i }
C
C WHERE Y AND X ARE SYMMETRIC TENSORS, x_i AND E_i ARE, RESPECTIVELY
C THE EIGENVALUES AND EIGENPROJECTIONS OF X, AND y(.) IS A SCALAR
C FUNCTION. THIS ROUTINE IS RESTRICTED TO 2-D TENSORS WITH ONE
C POSSIBLE (TRANSVERSAL) OUT-OF-PLANE COMPONENT.
C
C REFERENCE: Section A.5
C***********************************************************************
C Spectral decomposition of X
      CALL SPDEC2
     1(   EIGPRJ     ,EIGX       ,REPEAT     ,X          )
C In-plane eigenvalues of Y (and derivatives)
      DO 10 IDIR=1,2
        EIGY(IDIR)=FUNC(EIGX(IDIR))
        DEIGY(IDIR)=DFUNC(EIGX(IDIR))
   10 CONTINUE
C
C In-plane components of dY/dX
C ----------------------------
      CALL RVZERO(DYDX,MCOMP*MCOMP)
      IF(REPEAT)THEN
C for repeated in-plane eigenvalues of X
        DO 20 I=1,3
          DYDX(I,I)=DEIGY(1)*FOID(I,I)
   20   CONTINUE
      ELSE
C for distinct in-plane eigenvalues of X
        A1=(EIGY(1)-EIGY(2))/(EIGX(1)-EIGX(2))
        DO 40 I=1,3
          DO 30 J=I,3
            DYDX(I,J)=A1*(FOID(I,J)-EIGPRJ(I,1)*EIGPRJ(J,1)-
     1                EIGPRJ(I,2)*EIGPRJ(J,2))+
     2                DEIGY(1)*EIGPRJ(I,1)*EIGPRJ(J,1)+
     3                DEIGY(2)*EIGPRJ(I,2)*EIGPRJ(J,2)
            IF(I.NE.J)DYDX(J,I)=DYDX(I,J)
   30     CONTINUE
   40   CONTINUE
      ENDIF
C
C Out-of-plane component required
C -------------------------------
      IF(OUTOFP)DYDX(4,4)=DFUNC(X(4))
C
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION DLGD2(X)
C
      DOUBLE PRECISION X
      DOUBLE PRECISION RP5
      DATA   RP5 /0.5D0/
C***********************************************************************
C SCALAR FUNCTION THAT RELATES PRINCIPAL LOGARITHMIC STRECTHES AND
C EIGENVALUES OF THE CAUCHY-GREEN TENSOR
C
C REFERENCE: Section 3.1.7
C***********************************************************************
      DLGD2=RP5*DLOG(X)
C
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION DPLFUN(X, NPOINT, XFX)
C
      INTEGER NPOINT, I
      DOUBLE PRECISION X, XFX(2,NPOINT), R0
      DATA R0 / 0.0D0 /
C***********************************************************************
C DERIVATIVE OF THE PIECEWISE LINEAR FUNCTION 'PLFUN' DEFINED BY A SET
C OF NPOINT PAIRS {X,F(X)} STORED IN THE MATRIX XFX (DIM. 2*NPOINT).
C***********************************************************************
      DO 100 I=1,NPOINT
        IF (X.GE.XFX(1,I)) THEN
          GOTO 100
        ELSE
          IF (I.EQ.1) THEN
C           -- x < x1   --> f(x)=f(x1) --> df(x)/dx=0 ---
            DPLFUN=R0
            GOTO 999
          ELSE
C           -- x(i-1) <= x < x(i) ---
            DPLFUN=(XFX(2,I)-XFX(2,I-1))/
     1             (XFX(1,I)-XFX(1,I-1))
            GOTO 999
          ENDIF
        ENDIF
 100  CONTINUE
C     ---- x >= x(npoint) --> f(x) = f(x(npoint)) --> df/dx=0 ---
      DPLFUN=R0
 999  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ELEIIF
     1(   IELEM      ,IFFAIL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      LOGICAL  IFFAIL
C***********************************************************************
C ELEMENT INTERFACE FOR COMPUTATION OF ELEMENT INTERNAL FORCE VECTOR.
C CALL ELEMENT CLASS-SPECIFIC INTERNAL FORCE VECTOR CALCULATION ROUTINES
C
C REFERENCE: Section 5.5.1
C            Figure 5.4
C***********************************************************************
C Initialise internal force calculation failure flag
      IFFAIL=.FALSE.
C Recover element and material type group identification numbers
C ---------------------------------------------------------------
      IGRUP =IGRPID(IELEM)
      IELIDN=IELTID(IGRUP)
      MATIDN=MATTID(IGRUP)
C Identify element class
C ----------------------
      IELCLS=IELPRP(2,IELIDN)
C
C Call internal force computation routine according to element class
C -------------------------------------------------------------------
C
      IF(IELCLS.EQ.STDARD)THEN
C Standard 2-D displacement-based isoparametric elements
        CALL IFSTD2
     1(   IELEM      ,IFFAIL     ,MDIME      ,MELEM      ,MPOIN      ,
     2    MSTRE      ,MTOTV      ,NAXIS      ,NLARGE     ,NTYPE      ,
     3    COORD(1,1,1)       ,DINCR              ,
     4    ELOAD(1,IELEM)     ,IELPRP(1,IELIDN)   ,IPROPS(1,MATIDN)   ,
     5    LALGVA(1,1,IELEM,1),LNODS              ,RALGVA(1,1,IELEM,1),
     6    RELPRP(1,IELIDN)   ,RPROPS(1,MATIDN)   ,RSTAVA(1,1,IELEM,1),
     7    STRSG(1,1,IELEM,1) ,THKGP(1,IELEM,1)   ,TDISP              )
C 2-D F-bar elements (for large strain formulation only)
      ELSEIF(IELCLS.EQ.FBAR)THEN
        CALL IFFBA2
     1(   IELEM      ,IFFAIL     ,MDIME      ,MELEM      ,MPOIN      ,
     2    MSTRE      ,MTOTV      ,NAXIS      ,NTYPE      ,
     3    COORD(1,1,1)       ,DINCR              ,
     4    ELOAD(1,IELEM)     ,IELPRP(1,IELIDN)   ,IPROPS(1,MATIDN)   ,
     5    LALGVA(1,1,IELEM,1),LNODS              ,RALGVA(1,1,IELEM,1),
     6    RELPRP(1,IELIDN)   ,RPROPS(1,MATIDN)   ,RSTAVA(1,1,IELEM,1),
     7    STRSG(1,1,IELEM,1) ,THKGP(1,IELEM,1)   ,TDISP              )
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ELEIST
     1(   ESTIF      ,IELEM      ,KUNLD      ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas global database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      LOGICAL UNSYM
      DIMENSION
     1    ESTIF(MEVAB,MEVAB)
C***********************************************************************
C ELEMENT INTERFACE ROUTINE FOR COMPUTATION OF ELEMENT TANGENT STIFFNESS
C MATRIX.
C CALL ELEMENT CLASS-SPECIFIC STIFFNESS MATRIX CALCULATION ROUTINES
C
C REFERENCE: Section 5.6.2
C            Figure 5.5
C***********************************************************************
C
C Recover element and material type group identification numbers
C ---------------------------------------------------------------
      IGRUP=IGRPID(IELEM)
      IELIDN=IELTID(IGRUP)
      MATIDN=MATTID(IGRUP)
C Identify element class
      IELCLS=IELPRP(2,IELIDN)
C
C Call stiffness computation routine according to the element class
C ------------------------------------------------------------------
      IF(IELCLS.EQ.STDARD)THEN
        CALL STSTD2
     1(   IELEM      ,KUNLD      ,MDIME      ,MELEM      ,
     2    MPOIN      ,MSTRE      ,MTOTV      ,NAXIS      ,NLARGE     ,
     3    NTYPE      ,UNSYM      ,
     4    COORD(1,1,1)       ,DINCR              ,ESTIF              ,
     5    IELPRP(1,IELIDN)   ,IPROPS(1,MATIDN)   ,LALGVA(1,1,IELEM,1),
     6    LNODS              ,RALGVA(1,1,IELEM,1),RELPRP(1,IELIDN)   ,
     7    RPROPS(1,MATIDN)   ,RSTAVA(1,1,IELEM,1),RSTAVA(1,1,IELEM,2),
     8    STRSG(1,1,IELEM,1) ,THKGP(1,IELEM,1)   ,TDISP              )
      ELSEIF(IELCLS.EQ.FBAR)THEN
        CALL STFBA2
     1(   IELEM      ,KUNLD      ,MDIME      ,MELEM      ,
     2    MPOIN      ,MSTRE      ,MTOTV      ,NAXIS      ,
     3    NTYPE      ,UNSYM      ,
     4    COORD(1,1,1)       ,DINCR              ,ESTIF              ,
     5    IELPRP(1,IELIDN)   ,IPROPS(1,MATIDN)   ,LALGVA(1,1,IELEM,1),
     6    LNODS              ,RALGVA(1,1,IELEM,1),RELPRP(1,IELIDN)   ,
     7    RPROPS(1,MATIDN)   ,RSTAVA(1,1,IELEM,1),RSTAVA(1,1,IELEM,2),
     8    STRSG(1,1,IELEM,1) ,TDISP              )
      ENDIF
C
      RETURN
      END
```

```fortran
      SUBROUTINE ERRPRT(ERRCOD)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL       AVAIL ,FOUND
      CHARACTER*6   ERRCOD
      CHARACTER*22  HEADER
      CHARACTER*72  INLINE
      CHARACTER*256 HYPLASHOME
      DIMENSION IWBEG(128), IWEND(128)
C***********************************************************************
C PRINTS (TO THE STANDARD OUTPUT AND RESULTS FILE) MOST ERROR/WARNING
C MESSAGES OF HYPLAS.
C THE ARGUMENT "ERRCOD" CONTAINS THE ERROR/WARNING MESSAGE CODE. THE
C TEXT OF THE MESSAGE ASSOCIATED WITH THE GIVEN CODE MUST BE IN FILE
C "ERROR.RUN". THIS FILE IS KEPT IN THE FILE SYSTEM DIRECTORY WHOSE
C NAME IS STORED IN THE OPERATING SYSTEM ENVIRONMENT VARIABLE
C "HYPLASHOME" (SEE COMMENTS BELOW).
C***********************************************************************
 1000 FORMAT(///' ',74('*')/' *',72X,'*'/' *',25X,A22,25X,'*'/' *',72X,
     1       '*'/' ',74('*')/' *',72X,'*'/' * Code:         ',A6,51X,'*')
 1010 FORMAT(' *',A72,'*')
 1020 FORMAT(' *',72X,'*'/' ',74('*'))
 1030 FORMAT(//'   ASSOCIATED MESSAGES WILL NOT BE PRINTED !',//,
     1 '    The above error code was not found in file ERROR.RUN.',//)
 1040 FORMAT(//'   ASSOCIATED MESSAGES WILL NOT BE PRINTED !',//,
     1    '    File ERROR.RUN does not exist in HYPLASHOME directory.',//)
C
      IF(ERRCOD(1:2).EQ.'ED')THEN
        HEADER='  INPUT  DATA  ERROR  '
      ELSEIF(ERRCOD(1:2).EQ.'WD')THEN
        HEADER=' INPUT  DATA  WARNING '
      ELSEIF(ERRCOD(1:2).EQ.'EI')THEN
        HEADER='   INTERNAL   ERROR   '
      ELSEIF(ERRCOD(1:2).EQ.'EE')THEN
        HEADER='    EXECUTION ERROR   '
      ELSEIF(ERRCOD(1:2).EQ.'WE')THEN
        HEADER='   EXECUTION WARNING  '
      ELSE
        HEADER='  UNKNOWN ERROR TYPE  '
      ENDIF
      WRITE(*,1000)HEADER,ERRCOD
      WRITE(16,1000)HEADER,ERRCOD
C
C
C WARNING: GETENV is a non-standard FORTRAN 77 instruction, used here to
C          obtain the value of the operating system environment variable
C          HYPLASHOME - A character string containig the name of the
C          directory where the file ERROR.RUN (containing the error/
C          warning messages of HYPLAS) is kept in the file system. You
C          may need to change this if your FORTRAN compiler does not
C          support the instruction GETENV.
      CALL GETENV('HYPLASHOME',HYPLASHOME)
C
C Opens file ERROR.RUN
      NWRD=NWORD(HYPLASHOME,IWBEG,IWEND)
      INQUIRE(FILE=HYPLASHOME(1:IWEND(NWRD))//'/ERROR.RUN',EXIST=AVAIL)
      IF(.NOT.AVAIL)THEN
        WRITE(*,1040)
        WRITE(16,1040)
        GOTO 999
      ENDIF
      OPEN(23,FILE=HYPLASHOME(1:IWEND(NWRD))//'/ERROR.RUN',STATUS='OLD')
C
C Finds the character string with the given error code in file ERROR.RUN
      CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,ERRCOD     ,INLINE     ,
     2    23         ,NWRD       )
      IF(.NOT.FOUND)THEN
        WRITE(*,1030)
        WRITE(16,1030)
        GOTO 998
      ENDIF
C
C Reads and echoes the associated error/warning message
      NLINES=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      DO 10 I=1,NLINES
        READ(23,'(A72)')INLINE
        WRITE(*,1010)INLINE
        WRITE(16,1010)INLINE
   10 CONTINUE
      WRITE(*,1020)
      WRITE(16,1020)
  998 CONTINUE
      CLOSE(23,STATUS='KEEP')
  999 CONTINUE
C
C All codes WE???? and WD???? are WARNING codes (execution warnings and
C input data warnings, respectively) and do not stop the execution
C of HYPLAS. Any other type of message code is seen as an ERROR code and
C will cause HYPLAS to stop its execution. Refer to comments in the
C file ERROR.RUN.
      IF(ERRCOD(1:2).NE.'WE'.AND.ERRCOD(1:2).NE.'WD')CALL PEXIT
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION EXP2X(X)
C
      DOUBLE PRECISION X
      DOUBLE PRECISION R2
      DATA   R2  /2.0D0/
C***********************************************************************
C SCALAR FUNCTION THAT RELATES EIGENVALUES OF THE CAUCHY-GREEN
C TENSOR TO THE PRINCIPAL LOGARITHMIC STRECTHES
C
C REFERENCE: Section 3.1.7
C***********************************************************************
      EXP2X=DEXP(R2*X)
C
      RETURN
      END
```

```fortran
      SUBROUTINE EXPMAP
     1(   EXPX       ,NOCONV     ,X          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   NDIM=3     ,NDIM2=9    )
C Arguments
      LOGICAL  NOCONV
      DIMENSION
     1    EXPX(NDIM,NDIM)           ,X(NDIM,NDIM)
C Local arrays and variables
      DIMENSION
     1    XN(NDIM,NDIM)     ,XNM1(NDIM,NDIM)     ,XNM2(NDIM,NDIM)     ,
     2    XNM3(NDIM,NDIM)   ,X2(NDIM,NDIM)
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,TOL    ,OVER    ,UNDER   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,1.0D-10,1.0D+100,1.0D-100/
      DATA
     1    NMAX / 100 /
C***********************************************************************
C COMPUTES THE EXPONENTIAL OF A (GENERALLY UNSYMMETRIC) 3-D TENSOR. USES
C THE SERIES REPRESENTATION OF THE TENSOR EXPONENTIAL
C
C REFERENCE: Section B.1
C            Box B.1
C***********************************************************************
C Initialise series convergence flag
      NOCONV=.FALSE.
C Compute X square
      CALL RVZERO(X2,NDIM2)
      DO 30 I=1,NDIM
        DO 20 J=1,NDIM
          DO 10 K=1,NDIM
            X2(I,J)= X2(I,J)+X(I,K)*X(K,J)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C Compute principal invariants of X
      C1=X(1,1)+X(2,2)+X(3,3)
      C2=RP5*(C1*C1-(X2(1,1)+X2(2,2)+X2(3,3)))
      C3=X(1,1)*X(2,2)*X(3,3)+X(1,2)*X(2,3)*X(3,1)+
     1   X(1,3)*X(2,1)*X(3,2)-X(1,2)*X(2,1)*X(3,3)-
     2   X(1,1)*X(2,3)*X(3,2)-X(1,3)*X(2,2)*X(3,1)
C Start computation of exponential using its series definition
C =============================================================
      DO 50 I=1,NDIM
        DO 40 J=1,NDIM
          XNM1(I,J)=X2(I,J)
          XNM2(I,J)=X(I,J)
   40   CONTINUE
   50 CONTINUE
      XNM3(1,1)=R1
      XNM3(1,2)=R0
      XNM3(1,3)=R0
      XNM3(2,1)=R0
      XNM3(2,2)=R1
      XNM3(2,3)=R0
      XNM3(3,1)=R0
      XNM3(3,2)=R0
      XNM3(3,3)=R1
C Add first three terms of series
C -------------------------------
      DO 70 I=1,NDIM
        DO 60 J=1,NDIM
          EXPX(I,J)=RP5*XNM1(I,J)+XNM2(I,J)+XNM3(I,J)
   60   CONTINUE
   70 CONTINUE
C Add remaining terms (with X to the powers 3 to NMAX)
C ----------------------------------------------------
      FACTOR=R2
      DO 140 N=3,NMAX
C Use recursive formula to obtain X to the power N
        DO 90 I=1,NDIM
          DO 80 J=1,NDIM
            XN(I,J)=C1*XNM1(I,J)-C2*XNM2(I,J)+C3*XNM3(I,J)
   80     CONTINUE
   90   CONTINUE
C Update factorial
        FACTOR=DBLE(N)*FACTOR
        R1DFAC=R1/FACTOR
C Add Nth term of the series
        DO 110 I=1,NDIM
          DO 100 J=1,NDIM
            EXPX(I,J)=EXPX(I,J)+R1DFAC*XN(I,J)
  100     CONTINUE
  110   CONTINUE
C Check convergence of series
        XNNORM=SQRT(SCAPRD(XN(1,1),XN(1,1),NDIM2))
        IF(XNNORM.GT.OVER.OR.(XNNORM.LT.UNDER.AND.XNNORM.GT.R0)
     1                                     .OR.R1DFAC.LT.UNDER)THEN
C...first check possibility of overflow or underflow.
C...numbers are to small or too big: Break (unconverged) loop and exit
          NOCONV=.TRUE.
          GOTO 999
        ELSEIF(XNNORM*R1DFAC.LT.TOL)THEN
C...converged: Break series summation loop and exit with success
          GOTO 999
        ENDIF
        DO 130 I=1,NDIM
          DO 120 J=1,NDIM
            XNM3(I,J)=XNM2(I,J)
            XNM2(I,J)=XNM1(I,J)
            XNM1(I,J)=XN(I,J)
  120     CONTINUE
  130   CONTINUE
  140 CONTINUE
C Re-set convergence flag if series did not converge
      NOCONV=.TRUE.
C
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EXQ4
     1(   NGAUSP      ,EXMATX      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NNODE=4)
      DIMENSION EXMATX(NNODE,NGAUSP)
      DATA R1     /
     1     1.0D0 /
      DATA
     1    A4                ,B4                  ,C4                 /
     2    1.866025404D0     ,-0.5D0              ,0.133974596D0      /
C***********************************************************************
C SETS COEFFICIENTS MATRIX (EXMATX) FOR EXTRAPOLATION FROM GAUSS POINTS
C TO NODES FOR ELEMENT TYPE 'QUAD_4' (STANDARD 4-NODED BI-LINEAR
C QUADRILATERAL)
C
C REFERENCE: Section 5.6.1
C            E Hinton & JS Campbel. Local and global Smoothing of
C            discontinuous finite element functions using a least
C            squares method. Int. J. Num. meth. Engng., 8:461-480, 1974.
C            E Hinton & DRJ Owen. An introduction to finite element
C            computations. Pineridge Press, Swansea, 1979.
C***********************************************************************
      IF(NGAUSP.EQ.1)THEN
        EXMATX(1,1)=R1
        EXMATX(2,1)=R1
        EXMATX(3,1)=R1
        EXMATX(4,1)=R1
      ELSEIF(NGAUSP.EQ.4)THEN
        EXMATX(1,1)=A4
        EXMATX(1,2)=B4
        EXMATX(1,3)=B4
        EXMATX(1,4)=C4
        EXMATX(2,1)=B4
        EXMATX(2,2)=C4
        EXMATX(2,3)=A4
        EXMATX(2,4)=B4
        EXMATX(3,1)=C4
        EXMATX(3,2)=B4
        EXMATX(3,3)=B4
        EXMATX(3,4)=A4
        EXMATX(4,1)=B4
        EXMATX(4,2)=A4
        EXMATX(4,3)=C4
        EXMATX(4,4)=B4
      ENDIF
      RETURN
      END
```

```fortran
      SUBROUTINE EXQ4FB( EXMATX )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(     NGAUSP=4    ,NNODE=4     )
      DIMENSION EXMATX(NNODE,NGAUSP)
      DATA
     1    A4                ,B4                  ,C4                 /
     2    1.866025404D0     ,-0.5D0              ,0.133974596D0      /
C***********************************************************************
C SET COEFFICIENTS MATRIX (EXMATX) FOR EXTRAPOLATION FROM GAUSS POINTS
C TO NODES FOR ELEMENT TYPE 'QUAD_4_FBAR' (F-BAR 4-NODED BI-LINEAR
C QUADRILATERAL)
C
C REFERENCE: Section 5.6.1
C            E Hinton & JS Campbel. Local and global Smoothing of
C            discontinuous finite element functions using a least
C            squares method. Int. J. Num. meth. Engng., 8:461-480, 1974.
C            E Hinton & DRJ Owen. An introduction to finite element
C            computations. Pineridge Press, Swansea, 1979.
C***********************************************************************
      EXMATX(1,1)=A4
      EXMATX(1,2)=B4
      EXMATX(1,3)=B4
      EXMATX(1,4)=C4
      EXMATX(2,1)=B4
      EXMATX(2,2)=C4
      EXMATX(2,3)=A4
      EXMATX(2,4)=B4
      EXMATX(3,1)=C4
      EXMATX(3,2)=B4
      EXMATX(3,3)=B4
      EXMATX(3,4)=A4
      EXMATX(4,1)=B4
      EXMATX(4,2)=A4
      EXMATX(4,3)=C4
      EXMATX(4,4)=B4
      RETURN
      END
```

```
      SUBROUTINE EXQ8
     1(   NGAUSP     ,EXMATX     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NNODE=8)
      DIMENSION EXMATX(NNODE,NGAUSP)
      DATA R0    ,RP5   ,R1    /
     1     0.0D0 ,0.5D0 ,1.0D0 /
      DATA
     1    A4                ,B4                  ,C4                  /
     2    1.866025404D0     ,-0.5D0              ,0.133974596D0       /
      DATA
     1    A5                ,B5                  ,C5                  /
     2    1.290994449D0     ,-0.290994449D0      ,0.645497224D0       /
      DATA
     1    A9                ,B9                  ,C9                  ,
     2    D9                ,E9                  ,F9                  ,
     3    G9                ,H9                  ,P9                  /
     4    2.186939819D0     ,0.277777778D0       ,0.035282404D0       ,
     5    1.478830558D0     ,0.187836109D0       ,-0.985887039D0      ,
     6    -0.125224073D0    ,0.444444444D0       ,-0.666666667D0      /
C***********************************************************************
C SET COEFFICIENTS MATRIX (EXMATX) FOR EXTRAPOLATION FROM GAUSS POINTS
C TO NODES FOR ELEMENT TYPE 'QUAD_8' (STANDARD 8-NODED QUADRILATERAL)
C
C REFERENCE: Section 5.6.1
C            E Hinton & JS Campbel. Local and global Smoothing of
C            discontinuous finite element functions using a least
C            squares method. Int. J. Num. meth. Engng., 8:461-480, 1974.
C            E Hinton & DRJ Owen. An introduction to finite element
C            computations. Pineridge Press, Swansea, 1979.
C***********************************************************************
      IF(NGAUSP.EQ.1)THEN
        EXMATX(1,1)=R1
        EXMATX(2,1)=R1
        EXMATX(3,1)=R1
        EXMATX(4,1)=R1
        EXMATX(5,1)=R1
        EXMATX(6,1)=R1
        EXMATX(7,1)=R1
        EXMATX(8,1)=R1
      ELSEIF(NGAUSP.EQ.4)THEN
        EXMATX(1,1)=A4
        EXMATX(1,2)=B4
        EXMATX(1,3)=B4
        EXMATX(1,4)=C4
        EXMATX(2,1)=RP5*(A4+B4)
        EXMATX(2,2)=RP5*(B4+C4)
        EXMATX(2,3)=RP5*(B4+A4)
        EXMATX(2,4)=RP5*(C4+B4)
        EXMATX(3,1)=B4
        EXMATX(3,2)=C4
        EXMATX(3,3)=A4
        EXMATX(3,4)=B4
        EXMATX(4,1)=RP5*(B4+C4)
        EXMATX(4,2)=RP5*(C4+B4)
        EXMATX(4,3)=RP5*(A4+B4)
        EXMATX(4,4)=RP5*(B4+A4)
        EXMATX(5,1)=C4
        EXMATX(5,2)=B4
        EXMATX(5,3)=B4
        EXMATX(5,4)=A4
        EXMATX(6,1)=RP5*(C4+B4)
        EXMATX(6,2)=RP5*(B4+A4)
        EXMATX(6,3)=RP5*(B4+C4)
        EXMATX(6,4)=RP5*(A4+B4)
        EXMATX(7,1)=B4
        EXMATX(7,2)=A4
        EXMATX(7,3)=C4
        EXMATX(7,4)=B4
        EXMATX(8,1)=RP5*(B4+A4)
        EXMATX(8,2)=RP5*(A4+B4)
        EXMATX(8,3)=RP5*(C4+B4)
        EXMATX(8,4)=RP5*(B4+C4)
      ELSEIF(NGAUSP.EQ.5)THEN
        EXMATX(1,1)=A5
        EXMATX(1,2)=R0
        EXMATX(1,3)=R0
        EXMATX(1,4)=R0
        EXMATX(1,5)=B5
        EXMATX(2,1)=C5
        EXMATX(2,2)=R0
        EXMATX(2,3)=C5
        EXMATX(2,4)=R0
        EXMATX(2,5)=B5
        EXMATX(3,1)=R0
        EXMATX(3,2)=R0
        EXMATX(3,3)=A5
        EXMATX(3,4)=R0
        EXMATX(3,5)=B5
        EXMATX(4,1)=R0
        EXMATX(4,2)=R0
        EXMATX(4,3)=C5
        EXMATX(4,4)=C5
        EXMATX(4,5)=B5
        EXMATX(5,1)=R0
        EXMATX(5,2)=R0
        EXMATX(5,3)=R0
        EXMATX(5,4)=A5
```

```
        EXMATX(5,5)=B5
        EXMATX(6,1)=R0
        EXMATX(6,2)=C5
        EXMATX(6,3)=R0
        EXMATX(6,4)=C5
        EXMATX(6,5)=B5
        EXMATX(7,1)=R0
        EXMATX(7,2)=A5
        EXMATX(7,3)=R0
        EXMATX(7,4)=R0
        EXMATX(7,5)=B5
        EXMATX(8,1)=C5
        EXMATX(8,2)=C5
        EXMATX(8,3)=R0
        EXMATX(8,4)=R0
        EXMATX(8,5)=B5
      ELSEIF(NGAUSP.EQ.9)THEN
        EXMATX(1,1)=A9
        EXMATX(1,2)=F9
        EXMATX(1,3)=B9
        EXMATX(1,4)=F9
        EXMATX(1,5)=H9
        EXMATX(1,6)=G9
        EXMATX(1,7)=B9
        EXMATX(1,8)=G9
        EXMATX(1,9)=C9
        EXMATX(2,1)=R0
        EXMATX(2,2)=R0
        EXMATX(2,3)=R0
        EXMATX(2,4)=D9
        EXMATX(2,5)=P9
        EXMATX(2,6)=E9
        EXMATX(2,7)=R0
        EXMATX(2,8)=R0
        EXMATX(2,9)=R0
        EXMATX(3,1)=B9
        EXMATX(3,2)=G9
        EXMATX(3,3)=C9
        EXMATX(3,4)=F9
        EXMATX(3,5)=H9
        EXMATX(3,6)=G9
        EXMATX(3,7)=A9
        EXMATX(3,8)=F9
        EXMATX(3,9)=B9
        EXMATX(4,1)=R0
        EXMATX(4,2)=E9
        EXMATX(4,3)=R0
        EXMATX(4,4)=R0
        EXMATX(4,5)=P9
        EXMATX(4,6)=R0
        EXMATX(4,7)=R0
        EXMATX(4,8)=D9
        EXMATX(4,9)=R0
        EXMATX(5,1)=C9
        EXMATX(5,2)=G9
        EXMATX(5,3)=B9
        EXMATX(5,4)=G9
        EXMATX(5,5)=H9
        EXMATX(5,6)=F9
        EXMATX(5,7)=B9
        EXMATX(5,8)=F9
        EXMATX(5,9)=A9
        EXMATX(6,1)=R0
        EXMATX(6,2)=R0
        EXMATX(6,3)=R0
        EXMATX(6,4)=E9
        EXMATX(6,5)=P9
        EXMATX(6,6)=D9
        EXMATX(6,7)=R0
        EXMATX(6,8)=R0
        EXMATX(6,9)=R0
        EXMATX(7,1)=B9
        EXMATX(7,2)=F9
        EXMATX(7,3)=A9
        EXMATX(7,4)=G9
        EXMATX(7,5)=H9
        EXMATX(7,6)=F9
        EXMATX(7,7)=C9
        EXMATX(7,8)=G9
        EXMATX(7,9)=B9
        EXMATX(8,1)=R0
        EXMATX(8,2)=D9
        EXMATX(8,3)=R0
        EXMATX(8,4)=R0
        EXMATX(8,5)=P9
        EXMATX(8,6)=R0
        EXMATX(8,7)=R0
        EXMATX(8,8)=E9
        EXMATX(8,9)=R0
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE EXT3
     1(   EXMATX      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(NNODE=3,NGAUSP=1)
      DIMENSION EXMATX(NNODE,NGAUSP)
      DATA R1    /
     1     1.0D0 /
C***********************************************************************
C SET COEFFICIENTS MATRIX (EXMATX) FOR EXTRAPOLATION FROM GAUSS POINTS
C TO NODES FOR ELEMENT TYPE 'TRI_3' (STANDARD 3-NODED LINEAR TRIANGLE)
C
C REFERENCE: Section 5.6.1
C            E Hinton & JS Campbel. Local and global Smoothing of
C            discontinuous finite element functions using a least
C            squares method. Int. J. Num. meth. Engng., 8:461-480, 1974.
C            E Hinton & DRJ Owen. An introduction to finite element
C            computations. Pineridge Press, Swansea, 1979.
C***********************************************************************
      EXMATX(1,1)=R1
      EXMATX(2,1)=R1
      EXMATX(3,1)=R1
C
      RETURN
      END
```

```
      SUBROUTINE EXTNOD
     1(   EXMATX      ,VARGP      ,VARNOD     ,NVAR       ,NGAUSP     ,
     2    NNODE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE '../ELEMENTS.INC'
      DIMENSION
     1    EXMATX(NNODE,NGAUSP),VARGP(NVAR,NGAUSP),VARNOD(NVAR,NNODE)
C***********************************************************************
C EXTRAPOLATES GAUSS POINT VALUES OF A GIVEN FIELD TO NODES
C
C REFERENCE: Section 5.6.1
C            E Hinton & JS Campbel. Local and global Smoothing of
C            discontinuous finite element functions using a least
C            squares method. Int. J. Num. meth. Engng., 8:461-480, 1974.
C            E Hinton & DRJ Owen. An introduction to finite element
C            computations. Pineridge Press, Swansea, 1979.
C***********************************************************************
      CALL RVZERO(VARNOD,NVAR*NNODE)
      DO 30 IVAR=1,NVAR
        DO 20 INODE=1,NNODE
          DO 10 IGAUSP=1,NGAUSP
            VARNOD(IVAR,INODE)=VARNOD(IVAR,INODE)+
     1                         EXMATX(INODE,IGAUSP)*VARGP(IVAR,IGAUSP)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE FCLOSE
C*********************************************************************
C CLOSES DATA AND RESULTS FILES
C*********************************************************************
      CLOSE(UNIT=15,STATUS='KEEP')
      CLOSE(UNIT=16,STATUS='KEEP')
      RETURN
      END
```

```
      SUBROUTINE FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,KEYWRD     ,INLINE     ,
     2    NFILE      ,NWRD       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL FOUND
      CHARACTER*80 INLINE
      CHARACTER*(*) KEYWRD
      DIMENSION
     1    IWBEG(40), IWEND(40)
C***********************************************************************
C FINDS AND READS A LINE CONTAINING A SPECIFIED KEYWORD FROM A FILE.
C THIS ROUTINE SEARCHES FOR A GIVEN KEYWORD POSITIONED AS THE FIRST
C WORD OF A LINE IN A FILE.
C IF THE GIVEN KEYWORD IS FOUND THEN THE CORRESPONDING LINE IS READ AND
C RETURNED TOGETHER WITH THE NUMBER OF WORDS IN THE LINE AND TWO INTEGER
C ARRAYS CONTAINING THE POSITION OF THE BEGINNING AND END OF EACH WORD.
C***********************************************************************
 1000 FORMAT(A80)
C
      FOUND=.TRUE.
      IEND=0
   10 READ(NFILE,1000,END=20)INLINE
      NWRD=NWORD(INLINE,IWBEG,IWEND)
      IF(NWRD.NE.0)THEN
        IF(INLINE(IWBEG(1):IWEND(1)).EQ.KEYWRD)THEN
          GOTO 999
        ENDIF
      ENDIF
      GOTO 10
   20 IF(IEND.EQ.0)THEN
        IEND=1
        REWIND NFILE
        GOTO 10
      ELSE
        FOUND=.FALSE.
      ENDIF
  999 RETURN
      END
```

```
      SUBROUTINE FOPEN
     1(   DATFIL      ,RESFIL     ,RSTOUT     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER DATFIL*256,RESFIL*256,RSTOUT*256
      CHARACTER BELL*1
      LOGICAL AVAIL
C***********************************************************************
C READ DATA FILE NAME FROM STANDARD INPUT, SET NAMES FOR AND OPEN DATA
C AND RESULTS FILES AND SET RE-START FILE NAME
C
C REFERENCE: Figure 5.1
C***********************************************************************
C Read data file name from standard input
C ---------------------------------------
      WRITE(*,'(////15X,A,/15X,A)')
     1'Data file name must have extension .dat or .DAT',
     2'and must not contain blank spaces.'
      WRITE(*,'(/15X,A)')'(Type EXIT or QUIT to terminate)'
   10 WRITE(*,'(//15X,A$)')'Input data file name --------> '
      READ(*,'(A)',ERR=10)DATFIL
C Sort out data, result and re-start file names and open data and
C result files
C ------------------------------------------------------------------
      BELL=CHAR(7)
C Find end of data file name
      I=INDEX(DATFIL,' ')-1
      IF(I.EQ.0)THEN
        WRITE(*,'(/15X,A)')
     1          'Data file name must NOT begin with blank space !'
        GOTO 10
      ELSEIF(I.EQ.4.AND.
     1         (DATFIL(1:4).EQ.'EXIT'.OR.DATFIL(1:4).EQ.'exit'
     2      .OR.DATFIL(1:4).EQ.'QUIT'.OR.DATFIL(1:4).EQ.'quit'))THEN
        WRITE(*,'(///15X,A,///)')'Program HYPLAS terminated by user.'
        STOP ' '
      ENDIF
C Check data file name extension: it must be either .dat or .DAT
      IF((I-3.LT.1).OR.(DATFIL(I-3:I).NE.'.dat'.AND.
     1                  DATFIL(I-3:I).NE.'.DAT'))THEN
        WRITE(*,'(/15X,A,/15X,A)')
     1          'Data file name does not have extension .dat or .DAT !',
     2          'Please try again'
        WRITE(*,'(1X,A$)')BELL
        GOTO 10
      ENDIF
C Check existence of data file
      INQUIRE(FILE=DATFIL,EXIST=AVAIL)
      IF(.NOT.AVAIL)THEN
        WRITE(*,'(/A,A,A,A)')
     1'               File "',DATFIL(1:I),'" not found !  ',
     2              ' Please try again'
        WRITE(*,'(1X,A$)')BELL
        GOTO 10
      ENDIF
C give name to results file (with extension .res)
      RESFIL(1:I-3)=DATFIL(1:I-3)
      RESFIL(I-3:I)='.res'
      RESFIL(I+1:256)=DATFIL(I+1:256)
C give name to output re-start file (with extension .rst)
      RSTOUT(1:I-3)=DATFIL(1:I-3)
      RSTOUT(I-3:I)='.rst'
      RSTOUT(I+1:256)=DATFIL(I+1:256)
C Open data and results file
      OPEN(UNIT=15,FILE=DATFIL,STATUS='OLD')
      OPEN(UNIT=16,FILE=RESFIL,STATUS='UNKNOWN')
C
      RETURN
      END
```

```
      SUBROUTINE FRONT
     1(   IITER      ,KRESL     ,IFNEG      ,KUNLD      ,MXFRON     ,
     2    UNSYM      ,INCCUT    )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas global database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Common block of arrays used only by the frontal solver. This common
C block is firstly defined in HYPLAS main program
      COMMON / FRONTA /
     1    EQRHS(MTOTV,2)     ,EQROW(MFRON,MTOTV) ,EQCOL(MFRON,MTOTV) ,
     2    DECAY(MFRON)       ,GLOAD(MFRON,2)     ,VECRV(MFRON,2)     ,
     3    LOCEL(MEVAB,MELEM) ,NACVA(MFRON,MELEM) ,NAMEV(MTOTV)       ,
     4    NDEST(MEVAB,MELEM) ,NPIVO(MTOTV)       ,NFRON(MELEM)
C Arguments
      LOGICAL UNSYM, INCCUT
C Local arrays and variables
      DIMENSION
     1    ESTIF(MEVAB,MEVAB)
      DIMENSION
     1    GSTIF(MFRON*MFRON),LACVA(MFRON)
C Numerical constants
      DATA R0,R2,E5,E10/0.0D0,2.0D0,1.0D5,1.0D10/
C***********************************************************************
C ASSEMBLES AND SOLVES THE GLOBAL SYSTEM OF LINEAR ALGEBRAIC FINITE
C ELEMENT EQUILIBRIUM EQUATIONS (LINEARISED EQUILIBRIUM EQUATIONS FOR
C NON-LINEAR PROBLEMS) BY THE FRONTAL METHOD
C
C REFERENCE: Section 5.4.4
C***********************************************************************
 1010 FORMAT(//' Zero pivot encountered for variable no.',
     1I5,' of value',G14.6/)
 1020 FORMAT(//' *** WARNING ***'/' Diagonal decay of',G14.6,
     1' for variable no.',I5/
     2' probable mechanism or flying structure'//)
 1030 FORMAT(//' *** WARNING ***'/' Diagonal decay of',G14.6,
     1' for variable no.',I5/
     2' roundoff errors likely'//)
C Initialise increment cutting flag
      INCCUT=.FALSE.
C Decide solution required
      IF(NALGO.LT.0.AND.IITER.EQ.1.AND.KRESL.EQ.2)THEN
        NRHS=0
        MODE=0
      ELSE IF(NALGO.LT.0.AND.IITER.EQ.1.AND.KRESL.EQ.1)THEN
        NRHS=1
        MODE=2
      ELSE IF(NALGO.LT.0.AND.IITER.GT.1.AND.KRESL.EQ.1)THEN
        NRHS=2
        MODE=3
      ELSE IF(NALGO.LT.0.AND.IITER.GT.1.AND.KRESL.EQ.2)THEN
        NRHS=1
        MODE=1
      ELSE
        NRHS=1
        MODE=1
      ENDIF
      IF(MODE.EQ.0)GOTO 900
C Frontal stiffness (GSTIF)
C Start by initializing everything that matters to zero
      IF(IITER.GT.1)KUNLD=0
      IF(KRESL.EQ.1)THEN
        IF(.NOT.UNSYM)THEN
          MSTIF=(MXFRON*(MXFRON+1))/2
          DO 150 ISTIF=1,MSTIF
            GSTIF(ISTIF)=R0
  150     CONTINUE
        ENDIF
      ENDIF
      DO 160 IFRON=1,MXFRON
        DO 152 IRHS=1,NRHS
          GLOAD(IFRON,IRHS)=R0
          VECRV(IFRON,IRHS)=R0
  152   CONTINUE
        IF(KRESL.EQ.1)THEN
          DECAY(IFRON)=R0
          IF(UNSYM)THEN
            DO 155 JFRON=1,MXFRON
            GSTIF((IFRON-1)*MXFRON+JFRON)=R0
  155       CONTINUE
          ENDIF
          DO 156 IBUFA=1,NTOTV
            EQROW(IFRON,IBUFA)=R0
            EQCOL(IFRON,IBUFA)=R0
  156     CONTINUE
        ENDIF
  160 CONTINUE
      IF(KRESL.EQ.1)IFNEG=1
      NBUFA=0
      NVARB=0
C
C Main element assembly-reduction loop
C ====================================
      KELVA=0
      DO 320 IELEM=1,NELEM
      IGRUP=IGRPID(IELEM)
      IELIDN=IELTID(IGRUP)
      NNODE=IELPRP(3,IELIDN)
      NEVAB=IELPRP(5,IELIDN)
      IF(KRESL.GT.1) GOTO 400
      IF(IELEM.EQ.1)THEN
        NFRON(IELEM)=0
        DO 161 IFRON=1,MXFRON
          NACVA(IFRON,IELEM)=0
  161   CONTINUE
      ELSE
        NFRON(IELEM)=NFRON(IELEM-1)
        DO 162 IFRON=1,MXFRON
          NACVA(IFRON,IELEM)=LACVA(IFRON)
  162   CONTINUE
```

```
      ENDIF
C
C Call element interface routine for computation of element stiffness
C -------------------------------------------------------------------
      CALL ELEIST
     1(   ESTIF      ,IELEM      ,KUNLD      ,UNSYM      )
C
C transform the stiffness matrix into the local nodal coordinate system
C for prescribed displacements at an angle (for 2-D only)
      DO 168 INODE=1,NNODE
        LNODE=IABS(LNODS(IELEM,INODE))
        DO 167 IVFIX=1,NVFIX
          IF(NOFIX(IVFIX).EQ.LNODE)THEN
            IF(ANGLE(IVFIX).EQ.R0)GOTO 168
            C=COS(ANGLE(IVFIX))
            S=SIN(ANGLE(IVFIX))
            IEVAB=(INODE-1)*NDOFN+1
            JEVAB=IEVAB+1
            DO 165 KEVAB=1,NEVAB
              GASHI= C*ESTIF(IEVAB,KEVAB)+S*ESTIF(JEVAB,KEVAB)
              GASHJ=-S*ESTIF(IEVAB,KEVAB)+C*ESTIF(JEVAB,KEVAB)
              ESTIF(IEVAB,KEVAB)=GASHI
              ESTIF(JEVAB,KEVAB)=GASHJ
  165       CONTINUE
            DO 166 KEVAB=1,NEVAB
              GASHI= ESTIF(KEVAB,IEVAB)*C+ESTIF(KEVAB,JEVAB)*S
              GASHJ=-ESTIF(KEVAB,IEVAB)*S+ESTIF(KEVAB,JEVAB)*C
              ESTIF(KEVAB,IEVAB)=GASHI
              ESTIF(KEVAB,JEVAB)=GASHJ
  166       CONTINUE
            GOTO 168
          ENDIF
  167   CONTINUE
  168 CONTINUE
C
      DO 175 INODE=1,NNODE
        DO 170 IDOFN=1,NDOFN
          IEVAB=(IDOFN-1)*NNODE+INODE
          NPOSI=(INODE-1)*NDOFN+IDOFN
          IPOIN=IABS(LNODS(IELEM,INODE))
          LOCEL(NPOSI,IELEM)=
     1        SIGN(MASTER(NDOFN*(IPOIN-1)+IDOFN),LNODS(IELEM,IEVAB))
  170   CONTINUE
  175 CONTINUE
C
C Start by looking for existing destinations
      KEVAB=0
      DO 210 IEVAB=1,NEVAB
        NIKNO=IABS(LOCEL(IEVAB,IELEM))
        KEXIS=0
        DO 180 IFRON=1,NFRON(IELEM)
          IF(NIKNO.NE.NACVA(IFRON,IELEM)) GOTO 180
          KEVAB=KEVAB+1
          KEXIS=1
          NDEST(KEVAB,IELEM)=IFRON
  180   CONTINUE
        IF(KEXIS.NE.0) GOTO 210
C
C Now seek new empty places for destination vector
        DO 190 IFRON=1,MXFRON
          IF(NACVA(IFRON,IELEM).NE.0) GOTO 190
          NACVA(IFRON,IELEM)=NIKNO
          KEVAB=KEVAB+1
          NDEST(KEVAB,IELEM)=IFRON
          GOTO 200
  190   CONTINUE
C
C The new places may demand an increase in current frontwidth
  200   IF(NDEST(KEVAB,IELEM).GT.NFRON(IELEM)) NFRON(IELEM)=
     1                                         NDEST(KEVAB,IELEM)
  210 CONTINUE
  400 CONTINUE
      LFRON=NFRON(IELEM)
      DO 205 IFRON=1,MXFRON
        LACVA(IFRON)=NACVA(IFRON,IELEM)
  205 CONTINUE
C Assemble element loads in local nodal coordinate system if node has
C prescribed displacements at an angle (for 2-D only)
      DO 215 INODE=1,NNODE
        IEVAB=(INODE-1)*NDOFN
        LNODE=IABS(LNODS(IELEM,INODE))
        DO 211 IVFIX=1,NVFIX
          IF(NOFIX(IVFIX).EQ.LNODE.AND.ANGLE(IVFIX).NE.R0)THEN
            C=COS(ANGLE(IVFIX))
            S=SIN(ANGLE(IVFIX))
            IEVAB=IEVAB+1
            JEVAB=IEVAB+1
            IDEST=NDEST(IEVAB,IELEM)
            JDEST=NDEST(JEVAB,IELEM)
            IF(MODE.EQ.1)THEN
              GLOAD(IDEST,1)=GLOAD(IDEST,1)+
     1                     C*ELOAD(IEVAB,IELEM)+S*ELOAD(JEVAB,IELEM)
              GLOAD(JDEST,1)=GLOAD(JDEST,1)-
     1                     S*ELOAD(IEVAB,IELEM)+C*ELOAD(JEVAB,IELEM)
            ELSE IF(MODE.EQ.2)THEN
              GLOAD(IDEST,1)=GLOAD(IDEST,1)+
     1                     C*RLOAD(IEVAB,IELEM)+S*RLOAD(JEVAB,IELEM)
              GLOAD(JDEST,1)=GLOAD(JDEST,1)-
     1                     S*RLOAD(IEVAB,IELEM)+C*RLOAD(JEVAB,IELEM)
            ELSE IF(MODE.EQ.3)THEN
              GLOAD(IDEST,1)=GLOAD(IDEST,1)+
     1                     C*RLOAD(IEVAB,IELEM)+S*RLOAD(JEVAB,IELEM)
              GLOAD(JDEST,1)=GLOAD(JDEST,1)-
     1                     S*RLOAD(IEVAB,IELEM)+C*RLOAD(JEVAB,IELEM)
              GLOAD(IDEST,2)=GLOAD(IDEST,2)+
     1                     C*ELOAD(IEVAB,IELEM)+S*ELOAD(JEVAB,IELEM)
              GLOAD(JDEST,2)=GLOAD(JDEST,2)-
     1                     S*ELOAD(IEVAB,IELEM)+C*ELOAD(JEVAB,IELEM)
            ENDIF
            GOTO 215
          ENDIF
  211   CONTINUE
        DO 212 IDOFN=1,NDOFN
```

```
          IEVAB=IEVAB+1
          IDEST=NDEST(IEVAB,IELEM)
          IF(MODE.EQ.1)THEN
            GLOAD(IDEST,1)=GLOAD(IDEST,1)+ELOAD(IEVAB,IELEM)
          ELSE IF(MODE.EQ.2)THEN
            GLOAD(IDEST,1)=GLOAD(IDEST,1)+RLOAD(IEVAB,IELEM)
          ELSE IF(MODE.EQ.3)THEN
            GLOAD(IDEST,1)=GLOAD(IDEST,1)+RLOAD(IEVAB,IELEM)
            GLOAD(IDEST,2)=GLOAD(IDEST,2)+ELOAD(IEVAB,IELEM)
          ENDIF
  212   CONTINUE
  215 CONTINUE
C
C Assemble the element stiffnesses - but not in resolution
      IF(KRESL.GT.1) GOTO 402
      DO 220 IEVAB=1,NEVAB
        IDEST=NDEST(IEVAB,IELEM)
        IF(UNSYM)THEN
          LEVAB=NEVAB
        ELSE
          LEVAB=IEVAB
        ENDIF
        DO 222 JEVAB=1,LEVAB
          JDEST=NDEST(JEVAB,IELEM)
          IF(UNSYM)THEN
            NGESH=(IDEST-1)*MXFRON+JDEST
          ELSE
            IF((IDEST.EQ.JDEST).AND.(IEVAB.NE.JEVAB))
     1        ESTIF(IEVAB,JEVAB)=R2*ESTIF(IEVAB,JEVAB)
            NGASH=NFUNC(IDEST,JDEST)
            NGISH=NFUNC(JDEST,IDEST)
            IF(JDEST.GE.IDEST)NGESH=NGASH
            IF(JDEST.LT.IDEST)NGESH=NGISH
          ENDIF
          GSTIF(NGESH)=GSTIF(NGESH)+ESTIF(IEVAB,JEVAB)
  222   CONTINUE
C If diagonal term modified evaluate contribution to diagonal decay
        DECAY(IDEST)=DECAY(IDEST)+GSTIF(NGESH)*GSTIF(NGESH)
  220 CONTINUE
  402 CONTINUE
C
C Re-examine each element node, to enquire which can be eliminated
      DO 310 IEVAB=1,NEVAB
      NIKNO=-LOCEL(IEVAB,IELEM)
      IF(NIKNO.LE.0) GOTO 310
C
C Find positions of variables ready for elimination
      DO 300 IFRON=1,LFRON
        IF(LACVA(IFRON).NE.NIKNO) GOTO 300
        NBUFA=NBUFA+1
        NVARB=NVARB+1
C
C Extract the coefficients of the new equation for elimination
        IF(KRESL.GT.1) GOTO 404
        DO 230 JFRON=1,MXFRON
          IF(UNSYM)THEN
            NLOCA=(IFRON-1)*MXFRON+JFRON
          ELSE
            IF(IFRON.LT.JFRON) NLOCA=NFUNC(IFRON,JFRON)
            IF(IFRON.GE.JFRON) NLOCA=NFUNC(JFRON,IFRON)
          ENDIF
          EQROW(JFRON,NBUFA)=GSTIF(NLOCA)
          GSTIF(NLOCA)=R0
          IF(UNSYM)THEN
            NLOCA=(JFRON-1)*MXFRON+IFRON
            EQCOL(JFRON,NBUFA)=GSTIF(NLOCA)
            GSTIF(NLOCA)=R0
          ENDIF
  230   CONTINUE
  404   CONTINUE
C
C ...and extract the corresponding right hand sides
        DO 235 IRHS=1,NRHS
          EQRHS(NVARB,IRHS)=GLOAD(IFRON,IRHS)
          GLOAD(IFRON,IRHS)=R0
  235   CONTINUE
        KELVA=KELVA+1
        NAMEV(NVARB)=NIKNO
        NPIVO(NVARB)=IFRON
C
C Deal with pivot
        PIVOT=EQROW(IFRON,NBUFA)
        IF(KRESL.EQ.1.AND.PIVOT.LT.R0)THEN
          IFNEG=-1*IFNEG
        ENDIF
C
C Enquire whether present variable is free or prescribed
        IF(IFFIX(NIKNO).EQ.0) GOTO 250
C
C Deal with a prescribed nodal displacement
        DO 240 JFRON=1,LFRON
          IF(JFRON.EQ.IFRON)GOTO 240
          IF(.NOT.UNSYM)EQCOL(JFRON,NBUFA)=EQROW(JFRON,NBUFA)
          DO 237 IRHS=1,NRHS
            GLOAD(JFRON,IRHS)=GLOAD(JFRON,IRHS)-
     1                        FIXED(NIKNO,IRHS)*EQCOL(JFRON,NBUFA)
  237     CONTINUE
  240   CONTINUE
        GOTO 280
C
C Eliminate a free variable - Deal with the right hand side first
  250   CONTINUE
        IF(KRESL.EQ.1)THEN
          IF(PIVOT.EQ.R0)THEN
            WRITE(16,1010) NIKNO,PIVOT
            CALL ERRPRT('WE0008')
            INCCUT=.TRUE.
            GOTO 900
          ENDIF
C Check diagonal decay
          DECAY(IFRON)=SQRT(DECAY(IFRON))/PIVOT
          IF(ABS(DECAY(IFRON)).GE.E10)THEN
C Print warning of mechanism or flying structure
```

```
          WRITE(16,1020)DECAY(IFRON),NIKNO
        ELSE IF(ABS(DECAY(IFRON)).GE.E5)THEN
C Print warning of roundoff errors
          WRITE(16,1030)DECAY(IFRON),NIKNO
        ENDIF
      ENDIF
      DO 270 JFRON=1,LFRON
        IF(JFRON.EQ.IFRON)GOTO 270
        IF(.NOT.UNSYM)EQCOL(JFRON,NBUFA)=EQROW(JFRON,NBUFA)
        DO 255 IRHS=1,NRHS
          GLOAD(JFRON,IRHS)=GLOAD(JFRON,IRHS)-EQCOL(JFRON,NBUFA)*
     1                                    EQRHS(NVARB,IRHS)/PIVOT
  255   CONTINUE
C
C Now deal with the coefficients in core
        IF(KRESL.GT.1) GOTO 418
        IF(EQCOL(JFRON,NBUFA).EQ.R0) GOTO 270
        CUREQ=EQCOL(JFRON,NBUFA)
        IF(UNSYM)THEN
          NJFRON=LFRON
        ELSE
          NLOCA=NFUNC(0,JFRON)
          NJFRON=JFRON
        ENDIF
        DO 260 KFRON=1,NJFRON
          IF(KFRON.EQ.IFRON)GOTO 260
          IF(UNSYM)THEN
            NGASH=(JFRON-1)*MXFRON+KFRON
          ELSE
            NGASH=KFRON+NLOCA
          ENDIF
          GSTIF(NGASH)=GSTIF(NGASH)-CUREQ*EQROW(KFRON,NBUFA)/PIVOT
  260   CONTINUE
C If diagonal term modified evaluate contribution to diagonal decay
        DECAY(JFRON)=DECAY(JFRON)+GSTIF(NGASH)*GSTIF(NGASH)
  418   CONTINUE
  270 CONTINUE
  280 CONTINUE
C
C Record the new vacant space, and reduce frontwidth if possible
      LACVA(IFRON)=0
C Initialize diagonal decay
      DECAY(IFRON)=R0
      GOTO 290
C
C Complete the element loop in the forward elimination
C
  300 CONTINUE
  290 IF(LACVA(LFRON).NE.0) GOTO 310
      LFRON=LFRON-1
      IF(LFRON.GT.0) GOTO 290
  310 CONTINUE
  320 CONTINUE
C
C Back-substitution phase. Loop backwards through variables
C ==========================================================
      DO 340 IELVA=1,KELVA
C
C Prepare to back-substitute from the current equation
        IFRON=NPIVO(NVARB)
        NIKNO=NAMEV(NVARB)
        PIVOT=EQROW(IFRON,NBUFA)
        DO 325 IRHS=1,NRHS
          IF(IFFIX(NIKNO).NE.0) VECRV(IFRON,IRHS)=FIXED(NIKNO,IRHS)
  325   CONTINUE
        IF(IFFIX(NIKNO).EQ.0)SEQROW=EQROW(IFRON,NBUFA)
        IF(IFFIX(NIKNO).EQ.0)EQROW(IFRON,NBUFA)=R0
C
C Back-substitute in the current equation
        DO 331 JFRON=1,MXFRON
          DO 330 IRHS=1,NRHS
            EQRHS(NVARB,IRHS)=EQRHS(NVARB,IRHS)-
     1                       VECRV(JFRON,IRHS)*EQROW(JFRON,NBUFA)
  330     CONTINUE
  331   CONTINUE
        IF(IFFIX(NIKNO).EQ.0) EQROW(IFRON,NBUFA)=SEQROW
C
C Put the final values where they belong
        DO 335 IRHS=1,NRHS
          IF(IFFIX(NIKNO).EQ.0)VECRV(IFRON,IRHS)=EQRHS(NVARB,IRHS)/PIVOT
          IF(IFFIX(NIKNO).NE.0)FIXED(NIKNO,IRHS)=-EQRHS(NVARB,IRHS)
  335   CONTINUE
        NBUFA=NBUFA-1
        NVARB=NVARB-1
        IF(MODE.EQ.1)THEN
          DITER(NIKNO)=VECRV(IFRON,1)
        ELSE IF(MODE.EQ.2)THEN
          DTANG(NIKNO)=VECRV(IFRON,1)
        ELSE IF(MODE.EQ.3)THEN
          DTANG(NIKNO)=VECRV(IFRON,1)
          DITER(NIKNO)=VECRV(IFRON,2)
        ENDIF
  340 CONTINUE
C Copy displacements of master degrees of freedom to slaves
      DO 530 ITOTV=1,NTOTV
        IF(MODE.EQ.1)THEN
          DITER(ITOTV)=DITER(MASTER(ITOTV))
        ELSE IF(MODE.EQ.2)THEN
          DTANG(ITOTV)=DTANG(MASTER(ITOTV))
        ELSE IF(MODE.EQ.3)THEN
          DTANG(ITOTV)=DTANG(MASTER(ITOTV))
          DITER(ITOTV)=DITER(MASTER(ITOTV))
        ENDIF
  530 CONTINUE
C Transform local nodal displacements back to global values for
C prescribed displacements at an angle (for 2-D only)
      DO 370 IPOIN=1,NPOIN
        ISVAB=(IPOIN-1)*NDOFN
        DO 360 IVFIX=1,NVFIX
          IF(NOFIX(IVFIX).EQ.IPOIN.AND.ANGLE(IVFIX).NE.R0)THEN
            C=COS(ANGLE(IVFIX))
            S=SIN(ANGLE(IVFIX))
            ISVAB=ISVAB+1
```

```
          JSVAB=ISVAB+1
          IF(MODE.EQ.1)THEN
            GASHI= C*DITER(ISVAB)-S*DITER(JSVAB)
            GASHJ= S*DITER(ISVAB)+C*DITER(JSVAB)
            DITER(ISVAB)=GASHI
            DITER(JSVAB)=GASHJ
          ELSE IF(MODE.EQ.2)THEN
            GASHI= C*DTANG(ISVAB)-S*DTANG(JSVAB)
            GASHJ= S*DTANG(ISVAB)+C*DTANG(JSVAB)
            DTANG(ISVAB)=GASHI
            DTANG(JSVAB)=GASHJ
          ELSE IF(MODE.EQ.3)THEN
            GASHI= C*DTANG(ISVAB)-S*DTANG(JSVAB)
            GASHJ= S*DTANG(ISVAB)+C*DTANG(JSVAB)
            DTANG(ISVAB)=GASHI
            DTANG(JSVAB)=GASHJ
            GASHI= C*DITER(ISVAB)-S*DITER(JSVAB)
            GASHJ= S*DITER(ISVAB)+C*DITER(JSVAB)
            DITER(ISVAB)=GASHI
            DITER(JSVAB)=GASHJ
          ENDIF
          GOTO 370
        ENDIF
360   CONTINUE
370 CONTINUE
900 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE GAUS1D
     1(   NGAUS      ,POSGP      ,WEIGP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    POSGP(*)           ,WEIGP(*)
C***********************************************************************
C SET SAMPLING POINTS POSITIONS AND WEIGHTS FOR GAUSSIAN NUMERICAL
C INTEGRATION RULES IN 1-D (INTEGRATION OVER LIMITS [-1,1]).
C
C REFERENCE: Expression (4.36)
C            OC Zienkiewicz & RL Taylor. The finite element method,
C            Volume 1: The basis. 5th Edn. Butterworth Heinemann, 2000.
C            J Fish & T Belytschko. A first course in finite element
C            analysis. Wiley, Chichester, 2007.
C***********************************************************************
      IF(NGAUS.EQ.1)THEN
        POSGP(1)=0.0D0
        WEIGP(1)=2.0D0
      ELSEIF(NGAUS.EQ.2)THEN
        POSGP(1)=-0.577350269189626D0
        WEIGP(1)=1.0D0
        POSGP(2)=+0.577350269189626D0
        WEIGP(2)=1.0D0
      ELSEIF(NGAUS.EQ.3)THEN
        POSGP(1)=-0.774596669241483D0
        WEIGP(1)=0.555555555555556D0
        POSGP(2)=+0.0D0
        WEIGP(2)=0.888888888888889D0
        POSGP(3)=+0.774596669241483D0
        WEIGP(3)=0.555555555555556D0
      ELSEIF(NGAUS.EQ.4)THEN
        POSGP(1)=-0.861136311594053D0
        WEIGP(1)=0.347854845137454D0
        POSGP(2)=-0.339981043584856D0
        WEIGP(2)=0.652145154862546D0
        POSGP(3)=+0.339981043584856D0
        WEIGP(3)=0.652145154862546D0
        POSGP(4)=+0.861136311594053D0
        WEIGP(4)=0.347854845137454D0
      ELSE
        CALL ERRPRT('EI0004')
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE GAUS2D
     1(   DOMAIN     ,NGAUS      ,POSGP      ,WEIGP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*3 DOMAIN
      DIMENSION
     1    POSGP(2,*)         ,WEIGP(*)
C***********************************************************************
C SET SAMPLING POINTS POSITIONS AND WEIGHTS FOR GAUSSIAN NUMERICAL
C INTEGRATION RULES IN 2-D
C
C REFERENCE: Expression (4.31)
C            OC Zienkiewicz & RL Taylor. The finite element method,
C            Volume 1: The basis. 5th Edn. Butterworth Heinemann, 2000.
C            J Fish & T Belytschko. A first course in finite element
C            analysis. Wiley, Chichester, 2007.
C***********************************************************************
      IF(DOMAIN.EQ.'QUA')THEN
C
C Integration over quadrilateral domain with vertices
C                                          {(1,1),(1,-1),(-1,-1),(-1,1)}
C
        IF(NGAUS.EQ.1)THEN
          POSGP(1,1)=0.0D0
          POSGP(2,1)=0.0D0
          WEIGP(1)=4.0D0
        ELSEIF(NGAUS.EQ.4)THEN
          POSGP(1,1)=-0.577350269189626D0
          POSGP(2,1)=-0.577350269189626D0
          WEIGP(1)=1.0D0
          POSGP(1,2)=-0.577350269189626D0
          POSGP(2,2)=+0.577350269189626D0
          WEIGP(2)=1.0D0
          POSGP(1,3)=+0.577350269189626D0
          POSGP(2,3)=-0.577350269189626D0
          WEIGP(3)=1.0D0
          POSGP(1,4)=+0.577350269189626D0
          POSGP(2,4)=+0.577350269189626D0
          WEIGP(4)=1.0D0
        ELSEIF(NGAUS.EQ.5)THEN
          POSGP(1,1)=-0.774596669241483D0
          POSGP(2,1)=-0.774596669241483D0
          WEIGP(1)=0.555555555555556D0
          POSGP(1,2)=-0.774596669241483D0
          POSGP(2,2)=+0.774596669241483D0
          WEIGP(2)=0.555555555555556D0
          POSGP(1,3)=+0.774596669241483D0
          POSGP(2,3)=-0.774596669241483D0
          WEIGP(3)=0.555555555555556D0
          POSGP(1,4)=+0.774596669241483D0
          POSGP(2,4)=+0.774596669241483D0
          WEIGP(4)=0.555555555555556D0
          POSGP(1,5)=+0.0D0
          POSGP(2,5)=+0.0D0
          WEIGP(5)=1.777777777777778D0
        ELSEIF(NGAUS.EQ.9)THEN
          POSGP(1,1)=-0.774596669241483D0
          POSGP(2,1)=-0.774596669241483D0
          WEIGP(1)=0.308641975308643D0
          POSGP(1,2)=-0.774596669241483D0
          POSGP(2,2)=+0.0D0
          WEIGP(2)=0.493827160493828D0
          POSGP(1,3)=-0.774596669241483D0
          POSGP(2,3)=+0.774596669241483D0
          WEIGP(3)=0.308641975308643D0
          POSGP(1,4)=+0.0D0
          POSGP(2,4)=-0.774596669241483D0
          WEIGP(4)=0.493827160493828D0
          POSGP(1,5)=+0.0D0
          POSGP(2,5)=+0.0D0
          WEIGP(5)=0.790123456790124D0
          POSGP(1,6)=+0.0D0
          POSGP(2,6)=+0.774596669241483D0
          WEIGP(6)=0.493827160493828D0
          POSGP(1,7)=+0.774596669241483D0
          POSGP(2,7)=-0.774596669241483D0
          WEIGP(7)=0.308641975308643D0
          POSGP(1,8)=+0.774596669241483D0
          POSGP(2,8)=+0.0D0
          WEIGP(8)=0.493827160493828D0
          POSGP(1,9)=+0.774596669241483D0
          POSGP(2,9)=+0.774596669241483D0
          WEIGP(9)=0.308641975308643D0
        ELSE
          CALL ERRPRT('EI0001')
        ENDIF
      ELSEIF(DOMAIN.EQ.'TRI')THEN
C
C Integration over triangular domain with vertices {(0,0),(1,0),(0,1)}
C
        IF(NGAUS.EQ.1)THEN
          POSGP(1,1)=0.333333333333333D0
          POSGP(2,1)=0.333333333333333D0
          WEIGP(1)=0.5D0
        ELSEIF(NGAUS.EQ.3)THEN
          POSGP(1,1)=0.166666666666667D0
          POSGP(2,1)=0.166666666666667D0
          WEIGP(1)=0.166666666666667D0
          POSGP(1,2)=0.666666666666667D0
```

```
          POSGP(2,2)=0.166666666666667D0
          WEIGP(2)=0.166666666666667D0
          POSGP(1,3)=0.166666666666667D0
          POSGP(2,3)=0.666666666666667D0
          WEIGP(3)=0.166666666666667D0
        ELSE
          CALL ERRPRT('EI0002')
        ENDIF
      ELSE
        CALL ERRPRT('EI0003')
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE GETBMX
     1(   BMATX      ,CARTCO     ,CARTD      ,NDIME      ,MBDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    BMATX(MBDIM,*)     ,CARTCO(NDIME)      ,CARTD(NDIME,*)     ,
     2    SHAPE(*)
      DATA R0/0.0D0/
C***********************************************************************
C EVALUATES THE DISCRETE SYMMETRIC GRADIENT OPERATOR 'B' (SMALL
C STRAIN-DISPLACEMENT MATRIX) FOR PLANE STRESS/STRAIN AND AXISYMMETRIC
C PROBLEMS
C
C REFERENCE: Expression (4.30)
C***********************************************************************
C Plane strain/stress
C -------------------
      IY=0
      DO 10 INODE=1,NNODE
        IX=IY+1
        IY=IX+1
        BMATX(1,IX)=CARTD(1,INODE)
        BMATX(1,IY)=R0
        BMATX(2,IX)=R0
        BMATX(2,IY)=CARTD(2,INODE)
        BMATX(3,IX)=CARTD(2,INODE)
        BMATX(3,IY)=CARTD(1,INODE)
   10 CONTINUE
      IF(NTYPE.EQ.3)THEN
C Axisymmetric problem
C --------------------
        IY=0
        DO 20 INODE=1,NNODE
          IX=IY+1
          IY=IX+1
          IF(NAXIS.EQ.1)THEN
C Axisymmetric about Y axis
            BMATX(4,IX)=SHAPE(INODE)/CARTCO(NAXIS)
            BMATX(4,IY)=R0
          ELSE IF(NAXIS.EQ.2)THEN
C Axisymmetric about X axis
            BMATX(4,IX)=R0
            BMATX(4,IY)=SHAPE(INODE)/CARTCO(NAXIS)
          ENDIF
   20   CONTINUE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE GETGCO
     1(   CARTCO     ,ELCOD      ,MDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    CARTCO(NDIME)      ,ELCOD(MDIME,NNODE) ,SHAPE(NNODE)
      DATA R0/0.0D0/
C***********************************************************************
C EVALUATES THE GLOBAL CARTESIAN COORDINATES OF A POINT WITHIN AN
C ELEMENT BY INTERPOLATION OF THE ELEMENT NODAL COORDINATES
C
C REFERENCE: Expressions (4.39-40)
C***********************************************************************
      DO 20 IDIME=1,NDIME
        CARTCO(IDIME)=R0
        DO 10 INODE=1,NNODE
          CARTCO(IDIME)=CARTCO(IDIME)+ELCOD(IDIME,INODE)*SHAPE(INODE)
   10   CONTINUE
   20 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE GETGMX
     1(   CARTCO     ,CARTD      ,GMATX      ,MDIME      ,MGDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    CARTCO(MDIME)      ,CARTD(MDIME,*)     ,GMATX(MGDIM,*)     ,
     2    SHAPE(*)
      DATA R0/0.0D0/
C***********************************************************************
C EVALUATES THE DISCRETE (FULL) GRADIENT OPERATOR 'G' FOR PLANE
C STRESS/STRAIN AND AXISYMMETRIC PROBLEMS. COMPONENT ORDERING
C (11,21,12,22,33).
C
C REFERENCE: Expression (4.97)
C***********************************************************************
C Plane strain/stress
C -------------------
      IY=0
      DO 10 INODE=1,NNODE
        IX=IY+1
        IY=IX+1
        GMATX(1,IX)=CARTD(1,INODE)
        GMATX(1,IY)=R0
        GMATX(2,IX)=R0
        GMATX(2,IY)=CARTD(1,INODE)
        GMATX(3,IX)=CARTD(2,INODE)
        GMATX(3,IY)=R0
        GMATX(4,IX)=R0
        GMATX(4,IY)=CARTD(2,INODE)
   10 CONTINUE
      IF(NTYPE.EQ.3)THEN
C Axisymmetric problem
C --------------------
        IY=0
        DO 20 INODE=1,NNODE
          IX=IY+1
          IY=IX+1
          IF(NAXIS.EQ.1)THEN
C Axisymmetric about Y axis
            GMATX(5,IX)=SHAPE(INODE)/CARTCO(NAXIS)
            GMATX(5,IY)=R0
          ELSE IF(NAXIS.EQ.2)THEN
C Axisymmetric about X axis
            GMATX(5,IX)=R0
            GMATX(5,IY)=SHAPE(INODE)/CARTCO(NAXIS)
          ENDIF
   20   CONTINUE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE GREET
 1000 FORMAT(////////////////,
     1'  __________________________________________________________',
     2'______________ '/
     3' |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|                            ',
     4'              |'/
     5' |_|_|_|_|_|_|_|_|_|_|                                      ',
     6'              |'/
     7' |_|_|_|_|_|_|                                              ',
     8'              |'/
     9' |_|_|_|_|                                                  ',
     O'              |'/
     1' |_|_|_|                 H Y P L A S    version 2.0         ',
     2'              |'/
     3' |_|_|                  ===========================         ',
     4'              |'/
     5' |_|_|                                                      ',
     6'             _|'/
     7' |_|_|                                                      ',
     8'            |_|')
 1010 FORMAT(
     1' |_|_|_                                                     ',
     2'            |_|'/
     3' |_|_|_|_     SMALL AND LARGE STRAIN FINITE ELEMENT ANALYSIS',
     4'           _|_|'/
     5' |_|_|_|_|       OF HYPERELASTIC AND ELASTO-PLASTIC SOLIDS   ',
     6'          |_|_|'/
     7' |_|_|_|_|_                                                 ',
     8'       _ _|_|_|'/
     9' |_|_|_|_|_|                                                ',
     O'     _|_|_|_|_|'/
     1' |_|_|_|_|_|_ _                                             ',
     2' _ _|_|_|_|_|_|'/
     3' |_|_|_|_|_|_|_|_|____________________________________________',
     4'|_|_|_|_|_|_|_|_|')
 1020 FORMAT(
     1' |                                                      ,   ',
     2'              |'/
     3' |    Copyright (c) 1996-2008   EA de Souza Neto, D Peric & ',
     4'DRJ Owen      |'/
     5' |__________________________________________________________',
     6'______________|'/
     7' |                                                          ',
     8'              |'/
     9' |    Companion to the textbook:                            ',
     O'              |'/
     1' |    EA de Souza Neto, D Peric & DRJ Owen. Computational Me',
     2'thods for     |'/
     3' |    Plasticity: Theory and Applications. Wiley, Chichester',
     4', 2008.       |'/
     3' |__________________________________________________________',
     4'______________|'///)
      WRITE(*,1000)
      WRITE(*,1010)
      WRITE(*,1020)
      RETURN
      END
```

```
C***********************************************************************
C----------------------------------------------------------------------*
C|                                                                    |*
C|                                                                    |*
C|     H Y P L A S    version 2.0                                      |*
C|                                                                    |*
C|                                                                    |*
C|     Program for implicit small and large strain                    |*
C|     finite element analysis of hyperelastic and                    |*
C|     elastoplastic solids.                                          |*
C|                                                                    |*
C|                                                                    |*
C|     Copyright (c) 1996-2008  EA de Souza Neto, D Peric & DRJ Owen  |*
C|                              Civil and Computational Eng. Centre   |*
C|                              School of Engineering                 |*
C|                              Swansea University                    |*
C|                                                                    |*
C|                                                                    |*
C|     This program is a companion to the textbook:                   |*
C|     EA de Souza Neto, D Peric & DRJ Owen. Computational            |*
C|     Methods for Plasticity: Theory and Applications. Wiley,        |*
C|     Chichester, 2008.                                              |*
C|                                                                    |*
C|                                                                    |*
C|     Please send BUG REPORTS to                                     |*
C|                                                                    |*
C|                         hyplas_v2.0@live.co.uk                     |*
C|                                                                    |*
C|     NOTE: Messages sent to the authors' personal email addresses   |*
C|           will NOT be answered.                                    |*
C|                                                                    |*
C----------------------------------------------------------------------*
C***********************************************************************
C----------------------------------------------------------------------*
C                                                                      *
C     COPYRIGHT STATEMENT                                              *
C                                                                      *
C     You may only use this program for your own private purposes.     *
C     You are not allowed, in any circumstances, to distribute this    *
C     program (including its source code, executable and any other     *
C     files related to it, either in their original version or any     *
C     modifications introduced by you, the authors or any other party) *
C     in whole or in part, either freely or otherwise, in any medium,  *
C     without the prior written consent of the copyright holders.      *
C                                                                      *
C     DISCLAIMER                                                       *
C                                                                      *
C     This program (including its source code, executable and any      *
C     other files related to it) is provided "as is" without warranty  *
C     of any kind, either expressed or implied, including, but not     *
C     limited to, any implied warranties of fitness for purpose.       *
C     In particular, THIS PROGRAM IS BY NO MEANS GUARANTEED TO BE FREE *
C     FROM ERRORS.                                                     *
C     The results produced by this program are in no way garanteed to  *
C     be fit for any purpose.                                          *
C     This program (or any modification incorporated to it by you, the *
C     authors or any other party) will be used entirely at your own    *
C     risk.                                                            *
C     Under no circumstances will the authors/copyright holders be     *
C     liable to anyone for damages, including any general, special,    *
C     incidental or consequential damages arising from the use or      *
C     inability to use the program (including, but not limited to,     *
C     loss or corruption of data, failure of the program to operate in *
C     any particular way as well as damages arising from the use of    *
C     any results produced by the program for any purpose).            *
C                                                                      *
C     CONDITIONS OF USE                                                *
C                                                                      *
C     You may only use this program if you fully understand and agree  *
C     with the terms of the above disclaimer. You must not use this    *
C     program if you do not agree with or do not understand (fully or  *
C     in part) these conditions of use.                                *
C                                                                      *
C----------------------------------------------------------------------*
C***********************************************************************
C
C
      PROGRAM HYPLAS
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C Hyplas database: Global parameters and common blocks
      INCLUDE 'MAXDIM.INC'
      INCLUDE 'MATERIAL.INC'
      INCLUDE 'ELEMENTS.INC'
      INCLUDE 'GLBDBASE.INC'
C Common block of arrays used only by the frontal solver
      COMMON / FRONTA /
     1    EQRHS(MTOTV,2)     ,EQROW(MFRON,MTOTV) ,EQCOL(MFRON,MTOTV) ,
     2    DECAY(MFRON)       ,GLOAD(MFRON,2)     ,VECRV(MFRON,2)     ,
     3    LOCEL(MEVAB,MELEM) ,NACVA(MFRON,MELEM) ,NAMEV(MTOTV)       ,
     4    NDEST(MEVAB,MELEM) ,NPIVO(MTOTV)       ,NFRON(MELEM)
C Logical control flags for main program
      LOGICAL
     1    CONVRG     ,DIVERG     ,INCCUT     ,RSTRT      ,UNSYM
C File names
      CHARACTER*256
     1    DATFIL     ,RESFIL     ,RSTINP     ,RSTOUT
```

```
C Increment control arrays for main program
      DIMENSION
     1    DFACTV(MINCS)      ,DFSUB(MSUBIN)      ,MITERV(MINCS)      ,
     2    NOUTP(5)           ,NOUTPV(5,MINCS)    ,TOLERV(MINCS)
C Numerical constants
      PARAMETER
     1(   R0=0.0D0   ,RP5=0.5D0  ,RP7=0.7D0  )
C***********************************************************************
C
 1000 FORMAT(//20X,' H Y P L A S   ANALYSIS  RESULTS'/
     1          20X,'=================================='/)
 1010 FORMAT(
     1 7X,' _________________________________________________________ '/
     2 7X,'|                                                         |'/
     3 7X,'| Program compiled with the dimensioning parameters       |'/
     4 7X,'|_________________________________________________________|'/
     5 7X,'|                                                 |       |'/
     6 7X,'| Maximum number of elements               (MELEM) |',I6,' |'/
     7 7X,'| Maximum frontwidth allowed in solution   (MFRON) |',I6,' |'/
     8 7X,'| Maximum number of element groups         (MGRUP) |',I6,' |'/
     9 7X,'| Maximum number of load increments        (MINCS) |',I6,' |'/
     O 7X,'| Maximum number of nodal points           (MPOIN) |',I6,' |'/
     1 7X,'| Size of increment cutting stack array   (MSUBIN)|',I6,' |'/
     2 7X,'| Max. number of nodes with prescr.displ.  (MVFIV) |',I6,' |'/
     3 7X,'|_________________________________________________|_______|'/
     4                                                               )
 1015 FORMAT(/,' Data file name:'/
     1          ' ==============='//,1X,A/)
 1020 FORMAT(////15X,'Reading data...')
 1030 FORMAT(////15X,'H Y P L A S   ANALYSIS starting...'/)
 1050 FORMAT(////
     1' INCREMENT NUMBER',I5,19X,'TOTAL LOAD FACTOR =',G15.6/
     2' -------------------------------------------------------------',
     3'------------'/
     4 4X,'         ',13X,'relative residual',13X,'maximum residual'/
     5 4X,'iteration',13X,'    norm (%)     ',13X,'     norm       '/
     6' -------------------------------------------------------------',
     7'------------')
 1055 FORMAT(////
     1' INCREMENT NUMBER',I5,19X,'       ARC LENGTH =',G15.6/
     2' -------------------------------------------------------------',
     3'------------'/
     4 4X,'         ',6X,'relative residual',4X,'maximum residual',
     5 5X,'   total'/
     6 4X,'iteration',6X,'    norm (%)     ',4X,'     norm       ',
     7 5X,'load factor'/
     8' -------------------------------------------------------------',
     9'------------')
 1060 FORMAT(
     1' -------------------------------------------------------------',
     2'------------')
 1063 FORMAT(34X,'INCREMENTAL LOAD FACTOR =',G15.6)
 1065 FORMAT(30X,'CONVERGED TOTAL LOAD FACTOR =',G15.6)
 1067 FORMAT(24X,'CONVERGED INCREMENTAL LOAD FACTOR =',G15.6)
 1070 FORMAT(/////15X,'Program  H Y P L A S  successfully completed.')
 1080 FORMAT(///15X,'Data file name was ------> ',A)
 1090 FORMAT(// 15X,'Results file name is ----> ',A///)
 1095 FORMAT(   15X,'Re-start file name is ---> ',A//
     1          15X,'Last increment written -->',I5///)
 1040 FORMAT(//' Iterations not converged.')
 1100 FORMAT(//' Iterations diverging.')
 1110 FORMAT(/ ' Re-trying with reduced increment size...'/)
 1120 FORMAT(/ ' Re-trying with reduced arc length...'/)
C
C
C
C Start up. Read data, initialise variables, etc...
C
C REFERENCE: Flowchart of Figure 5.1
C
C **************************************************
C
C Send greeting message to standard output
      CALL GREET
C Read names and open relevant files
      CALL FOPEN(   DATFIL     ,RESFIL     ,RSTOUT     )
C Echo dimensioning parameters defined in file MAXDIM.INC
      WRITE(16,1000)
      WRITE(16,1010)MELEM,MFRON,MGRUP,MINCS,MPOIN,MSUBIN,MVFIX
C Echo data file name
      I=INDEX(DATFIL,' ')-1
      WRITE(16,1015)DATFIL(1:I)
C
C Read relevant data from input data/re-start file
C ------------------------------------------------
C Check if main data is to be read from the input data file or from an
C input re-start file
      CALL RSTCHK(  RSTINP     ,RSTRT      )
      WRITE(*,1020)
C
      IF(RSTRT)THEN
C Re-start mode: Read main data from input re-start file
         CALL RSTART
     1(  DFOLD      ,DLENG      ,DLENGO     ,DLENM      ,DLAMD      ,
     2   IFNEG      ,IINCS      ,MXFRON     ,NOUTP      ,TFACT      ,
     3   TFACTO     ,UNSYM      ,RSTINP     ,RSTOUT     ,0          ,
     4   IDUMMY     )
      ELSE
```

```
C Not re-start mode: Read main data from input data file
C ...read most of the problem data
        CALL INDATA(MXFRON ,UNSYM)
C ...read and evaluate the applied external loads
        CALL INLOAD
      ENDIF
C
C For any mode: Read load incrementation data from input data file
      CALL ININCR
     1(   DFACT      ,DLENP      ,FSTOP      ,ITDES      ,MINCS      ,
     2    MITER      ,NALGO      ,NINCS      ,TOLER      ,
     3    DFACTV     ,MITERV     ,NOUTP      ,NOUTPV     ,TOLERV     )
C
C Initialise some variables and arrays if not in re-start mode
C -------------------------------------------------------------
      INCRST=0
      IF(.NOT.RSTRT)THEN
        CALL INITIA
     1(   DLAMD      ,IFNEG      ,KUNLD      ,TFACT      )
      ENDIF
C
C
C
C Start incremental finite element analysis...
C ********************************************
C
      WRITE(*,1030)
C
C========================================================================
C                                                                       |
C                  Start loop over load increments                      |
C                                                                       |
C REFERENCE: Chapter 4 (Boxes 4.1-4) of the companion textbook.         |
C            Section 5.4.                                               |
C            The load incrementation loops carried out here are those   |
C            of the Flowcharts of Figures 5.2-3.                        |
C                                                                       |
C========================================================================
C
      IF(.NOT.RSTRT)IINCS=0
C
      DO 50 ICOUNT=1,NINCS
C
        IPSUB=1
        IF(NALGO.GT.0)THEN
          DFSUB(1)=DFACTV(ICOUNT)
          TOLER=TOLERV(ICOUNT)
          MITER=MITERV(ICOUNT)
          NOUTP(1)=NOUTPV(1,ICOUNT)
          NOUTP(2)=NOUTPV(2,ICOUNT)
          NOUTP(3)=NOUTPV(3,ICOUNT)
          NOUTP(4)=NOUTPV(4,ICOUNT)
          NOUTP(5)=NOUTPV(5,ICOUNT)
        ENDIF
C
C Reset converged problem variables
C ---------------------------------
        CALL SWITCH( 1 )
C
   10   CONTINUE
C
C Update increment counter
C ------------------------
        IINCS=IINCS+1
C
C For fixed increments option only: Increment external load according
C to user-prescribed incremental proportional load factor
C -------------------------------------------------------------------
        IF(NALGO.GT.0)THEN
          DFACT=DFSUB(IPSUB)
          CALL INCREM
     1(   IINCS      ,TFACT      ,TOLER      ,MITER      ,NOUTP      ,
     2    DFACT      ,DFOLD      ,KUNLD      )
        ENDIF
C
C_______________________________________________________________________
C                                                                       |
C            Start loop over equilibrium iterations                     |
C_______________________________________________________________________|
C
        DO 20 IITER=1,MITER
C
C Select solution alorithm variable KRESL
          CALL ALGOR(IINCS ,IITER ,KRESL ,KUNLD )
C
          IF(NALGO.LT.0)THEN
C Set up prescribed displacements for tangential solution for the
C arc-length method
            CALL TANGEN
          ENDIF
C
C Assemble stiffness matrix and solve for iterative displacements
C (tangential solution for the arc-length method) the linearised system
C of discretised equilibrium equations using the frontal algorithm
C ---------------------------------------------------------------------
          CALL FRONT
     1(   IITER      ,KRESL      ,IFNEG      ,KUNLD      ,MXFRON     ,
     2    UNSYM      ,INCCUT     )
```

```
C
          IF(INCCUT)THEN
C System solution failed due to zero pivot: break equilibrium iteration
C loop and activate increment cutting
            GOTO 30
          ENDIF
C
C For Arc-Length method only: Compute iterative displacement according
C to the arc-length constraint and update the incremental and total
C load factors
C ---------------------------------------------------------------------
          IF(NALGO.LT.0)THEN
            CALL ARCLEN
     1(   DFACT      ,DLAMD      ,DLENG      ,DLENM      ,DLENP      ,
     2    IFNEG      ,IINCS      ,IITER      ,INCCUT     ,TFACT      )
C
            IF(INCCUT)THEN
C No real roots for arc-length constraint equation: break equilibrium
C iteration loop and activate increment cutting
              GOTO 30
            ENDIF
          ENDIF
C
C Update incremental and total displacements. Also update nodal
C coordinates for large deformation analyses
C --------------------------------------------------------------
          CALL UPCONF
C
C Re-set converged load factors and print out increment information
C ------------------------------------------------------------------
          IF(IITER.EQ.1)THEN
            IF(IINCS.EQ.1)THEN
C Re-set previous converged load factors/arc-length
              IF(NALGO.LT.0)DLENGO=DLENG
              DFACTO=DFACT
              TFACTO=R0
            ENDIF
            IF(NALGO.GT.0)THEN
C Fixed increments option: print current total load factor
              WRITE(*,1050) IINCS,TFACT
              WRITE(16,1050)IINCS,TFACT
            ELSE
C Arc-length: print current arc-length
              WRITE(*,1055) IINCS,DLENG
              WRITE(16,1055)IINCS,DLENG
            ENDIF
          ENDIF
C
C Re-set relevant problem variables to last converged solution
C ------------------------------------------------------------
          CALL SWITCH( 2 )
C
C Update problem variables (stress and other state variables) and
C evaluate internal force vectors of all elements
C ----------------------------------------------------------------
          CALL INTFOR( INCCUT )
C
          IF(INCCUT)THEN
C Internal force calculation failed: break equilibrium iteration loop
C and activate load increment cutting
            GOTO 30
          ENDIF
C
C Assemble internal and external global force vectors, reactions,
C compute residual and check for convergence
C ----------------------------------------------------------------
          CALL CONVER(CONVRG,DIVERG,IITER,TOLER,TFACT)
C
          ITACT=IITER
C
          IF(CONVRG)THEN
C Iterations have converged: break equilibrium iteration loop and go to
C next load increment
            WRITE(*,1060)
            WRITE(16,1060)
            IF(NALGO.GT.0)THEN
              WRITE(*,1063) DFACT
              WRITE(16,1063)DFACT
            ELSE
              WRITE(*,1065) TFACT
              WRITE(*,1067) DFACT
              WRITE(16,1065)TFACT
              WRITE(16,1067)DFACT
            ENDIF
            WRITE(*,1060)
            WRITE(16,1060)
            GOTO 40
          ELSEIF(DIVERG)THEN
C Iterations are diverging: break equilibrium iteration loop and
C activate load increment cutting
            WRITE(16,1100)
            WRITE(*,1100)
            GOTO 30
          ENDIF
C
   20   CONTINUE
C_____________________________________________________________________
C                                                                     |
```

```
C                 End loop over equilibrium iterations                 |
C______________________________________________________________________|
C
C Newton-Raphson procedure did not converge within the prescribed
C maximum number of iterations !!
C Print corresponding message and proceed to increment cutting
C
        WRITE(16,1040)
        WRITE(*,1040)
C
C
C
   30   CONTINUE
C
C
C Activate increment cutting
C
C REFERENCE: Section 5.4.3
C --------------------------
C
        IF(NALGO.GT.0)THEN
C For fixed increments option: split current load increment into two
C equally sized sub-increments
          WRITE(16,1110)
          WRITE(*,1110)
          IF(IPSUB.EQ.MSUBIN)THEN
C abort program if maximum permissible number of consecutive increment
C cuts has been exceeded (i.e. sub-increment stack array DFSUB is full)
            CALL ERRPRT('EE0002')
          ENDIF
          DFSUB(IPSUB)  =DFSUB(IPSUB)*RP5
          DFSUB(IPSUB+1)=DFSUB(IPSUB)
          IPSUB=IPSUB+1
        ELSE
C For arc-length method: reduce the arc-length
          WRITE(16,1120)
          WRITE(*,1120)
          IF(IINCS.EQ.1)THEN
            DFACT=DFACTO*RP7
            DFACTO=DFACT
          ELSE
            DLENG=DLENGO*RP7
            DLENGO=DLENG
          ENDIF
        ENDIF
C Switch relevant variables to last converged values (in load increment
C cutting mode) before re-trying with reduced load increment/arc-length
        TFACT=TFACTO
        CALL SWITCH( 3 )
        IINCS=IINCS-1
        GOTO 10
C
C
C
   40   CONTINUE
C
C Newton-Raphson iterations converged for the current load increment
C -------------------------------------------------------------------
C Reset some converged parameters
        DLENGO=DLENG
        TFACTO=TFACT
        IF(NALGO.GT.0)THEN
C Fixed increments option: update pointer to sub-increments stack array
          IPSUB=IPSUB-1
        ELSE
C Arc-length method: update arc-length according to the desired number
C of iterations for convergence and the actual number of iterations
C needed for convergence in the previous load step
          CALL LENGTH(DLENG ,DLENM ,ITACT ,ITDES )
        ENDIF
C
C Output results if required
C REFERENCE: Section 5.4.7
C
        CALL OUTPUT(TFACT,IINCS,IITER,NOUTP)
        IF((NALGO.GT.0.AND.IPSUB.EQ.0).OR.(NALGO.LT.0))THEN
          CALL RSTART
     1(  DFOLD      ,DLENG      ,DLENGO     ,DLENM      ,DLAMD      ,
     2   IFNEG      ,IINCS      ,MXFRON     ,NOUTP      ,TFACT      ,
     3   TFACTO     ,UNSYM      ,RSTINP     ,RSTOUT     ,1          ,
     4   INCRST     )
        ELSEIF(NALGO.GT.0.AND.IPSUB.NE.0)THEN
          CALL SWITCH( 1 )
          GOTO 10
        ENDIF
        IF(NALGO.LT.0.AND.FSTOP.NE.R0.AND.TFACT.GT.FSTOP)THEN
C Arc-length only: Break loop over increments and stop if maximum
C prescribed load factor has been exceeded
          GOTO 60
        ENDIF
C
C
   50 CONTINUE
C
C=======================================================================
C                                                                      |
C                    End loop over load increments                     |
C                                                                      |
```

```
C=========================================================================
C
   60 CONTINUE
C
C
C
C Exit HYPLAS
C ***********
C
C Close files before exit
      CALL FCLOSE
C Echo file names back to standard output and stop
      WRITE(*,1070)
      I=INDEX(RESFIL,' ')-1
      WRITE(*,1080)DATFIL(1:I)
      WRITE(*,1090)RESFIL(1:I)
      IF(INCRST.NE.0)THEN
        WRITE(*,1095)RSTOUT(1:I),INCRST
      ENDIF
      STOP ' '
      END
```

```
      SUBROUTINE IFFBA2
     1(   IELEM      ,INCCUT     ,MDIME      ,MELEM      ,MPOIN      ,
     2    MSTRE      ,MTOTV      ,NAXIS      ,NTYPE      ,
     3    COORD1     ,DINCR      ,ELOAD      ,IELPRP     ,IPROPS     ,
     4    LALGVA     ,LNODS      ,RALGVA     ,RELPRP     ,RPROPS     ,
     5    RSTAVA     ,STRSG      ,THKGP      ,TDISP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../MATERIAL.INC'
C
      PARAMETER( MGDIM=5 ,MBDIM=4 ,NDIME=2 ,NDOFN=2 )
C Arguments
      LOGICAL  INCCUT ,LALGVA
      DIMENSION
     1    COORD1(MDIME,MPOIN),DINCR(MTOTV)       ,ELOAD(MEVAB)       ,
     2    IELPRP(*)          ,IPROPS(*)         ,LALGVA(MLALGV,MTOTG),
     3    LNODS(MELEM,MEVAB),RALGVA(MRALGV,MTOTG),RELPRP(*)          ,
     4    RPROPS(*)          ,RSTAVA(MRSTAV,MTOTG),STRSG(MSTRE,MTOTG),
     5    THKGP(MTOTG)       ,TDISP(MTOTV)
C Local variables and arrays
      LOGICAL  SUFAIL
      DIMENSION
     1    BMATX(MBDIM,MEVAB) ,CARTD(NDIME,MNODE) ,DELDIS(MDOFN,MNODE),
     2    DERIV(NDIME,MNODE) ,EINCR(MBDIM)       ,ELCOD(NDIME,MNODE) ,
     3    FINCIN(3,3)        ,FINCR(3,3)         ,FINV(3,3)          ,
     4    GMATX(MGDIM,MEVAB) ,GPCOD(NDIME)       ,SHAPE(MNODE)       ,
     5    TELDIS(MDOFN,MNODE)
C Local numerical constants
      DATA
     1    R0   ,RP5  ,R1   ,R3   ,R8   /
     2    0.0D0,0.5D0,1.0D0,3.0D0,8.0D0/
C***********************************************************************
C COMPUTE INTERNAL FORCE VECTOR OF ALL ELEMENTS OF CLASS 'FBAR'
C (F-Bar ELEMENTS) IN 2-D: PLANE STRAIN AND AXISYMMETRIC
C
C REFERENCE: Box 15.1
C***********************************************************************
      R1D3=R1/R3
      IF(NTYPE.EQ.2)THEN
        NBDIM=3
      ELSEIF(NTYPE.EQ.3)THEN
        TWOPI=R8*ATAN(R1)
        NBDIM=4
      ELSE
C F-bar implementation valid only for plane strain and axisymmetric
        CALL ERRPRT('EI0033')
      ENDIF
C Identify element type
      IELTYP=IELPRP(1)
C retrieve some element integer properties
      NNODE =IELPRP(3)
      NGAUSP=IELPRP(4)
      NEVAB =IELPRP(5)
C
C Set element arrays of current nodal coordinates, total and incremental
C displacements
      DO 20 INODE =1,NNODE
        LNODE=IABS(LNODS(IELEM,INODE))
        NPOSN=(LNODE-1)*NDOFN
        DO 10 IDOFN=1,NDOFN
          NPOSN=NPOSN+1
          ELCOD(IDOFN,INODE)=COORD1(IDOFN,LNODE)
          TELDIS(IDOFN,INODE)=-TDISP(NPOSN)
          DELDIS(IDOFN,INODE)=-DINCR(NPOSN)
   10   CONTINUE
   20 CONTINUE
C
C Initialize element force vector
C
      CALL RVZERO(ELOAD,NEVAB)
C
C-----------------------------------------------------------------------
C Calculation of the F-bar deformation gradient determinat
C-----------------------------------------------------------------------
C Evaluate inverse of the incremental deformation gradient at the
C centroid of the F-bar element
      NGAUSB=IELPRP(8)
      IPOS  =NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+2*NGAUSB+1
      EXISC =RELPRP(IPOS)
      ETASC =RELPRP(IPOS+1)
C
      CALL SHPFUN
     1(   DERIV      ,ETASC      ,EXISC      ,0          ,IELTYP     ,
     2    NDIME      ,SHAPE      )
      CALL JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    NDIME      ,NDIME      ,NNODE      )
      IF(DETJAC.LE.R0)THEN
C cut increment if element jacobian is not positive definite
        CALL ERRPRT('WE0021')
        INCCUT=.TRUE.
        GOTO 999
      ENDIF
      IF(NTYPE.EQ.3)CALL GETGCO
     1(   GPCOD      ,ELCOD      ,NDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C
```

```
      CALL GETGMX
     1(   GPCOD      ,CARTD      ,GMATX      ,NDIME      ,MGDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C Determinant of the incremental deformation gradient at the centroid
      CALL DEFGRA
     1(   DELDIS     ,FINCIN     ,GMATX      ,MDOFN      ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
      IF(NTYPE.EQ.2)THEN
        AFACT=RP5
        DET=FINCIN(1,1)*FINCIN(2,2)-FINCIN(1,2)*FINCIN(2,1)
      ELSEIF(NTYPE.EQ.3)THEN
C... cut increment if incr. def. gradient is not positive definite
        IF(FINCIN(3,3).LE.R0)THEN
          CALL ERRPRT('WE0022')
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
        AFACT=R1D3
        DET=(FINCIN(1,1)*FINCIN(2,2)-FINCIN(1,2)*FINCIN(2,1))*
     1      FINCIN(3,3)
      ENDIF
      DET0=R1/DET
C Determinant of the total deformation gradient at the centroid
      CALL DEFGRA
     1(   TELDIS     ,FINV       ,GMATX      ,MDOFN      ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
      IF(NTYPE.EQ.2)THEN
        DET=FINV(1,1)*FINV(2,2)-FINV(1,2)*FINV(2,1)
      ELSEIF(NTYPE.EQ.3)THEN
C... cut increment if deformation gradient is not positive definite
        IF(FINV(3,3).LE.R0)THEN
          CALL ERRPRT('WE0023')
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
        DET=(FINV(1,1)*FINV(2,2)-FINV(1,2)*FINV(2,1))*FINV(3,3)
      ENDIF
      DETF0=R1/DET
C-----------------------------------------------------------------------
C
C=======================================================================
C                 Begin loop over Gauss points                         |
C=======================================================================
C
      IPPOS=1
      IPWEI=NGAUSP*NDIME+1
      DO 40 IGAUSP=1,NGAUSP
C Set Gauss points positions and weights
        EXISP=RELPRP(IPPOS-1+IGAUSP*2-1)
        ETASP=RELPRP(IPPOS-1+IGAUSP*2)
        WEIGP=RELPRP(IPWEI-1+IGAUSP)
C Evaluate shape functions and their derivatives (use current
C configuration for large strains)
        CALL SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
     2    NDIME      ,SHAPE      )
        CALL JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    NDIME      ,NDIME      ,NNODE      )
        IF(DETJAC.LE.R0)THEN
C ...cut increment if current jacobian is not positive definite
          CALL ERRPRT('WE0024')
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
        IF(NTYPE.EQ.3)CALL GETGCO
     1(   GPCOD      ,ELCOD      ,NDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C Evaluate symmetric gradient operator B
        CALL GETBMX
     1(   BMATX      ,GPCOD      ,CARTD      ,NDIME      ,NBDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C
C Compute basic kinematic variables needed for state update
C =========================================================
C
C Large strains: compute incremental deformation gradient
C -------------------------------------------------------
C gradient operator G in current configuration
        CALL GETGMX
     1(   GPCOD      ,CARTD      ,GMATX      ,NDIME      ,MGDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C incremental deformation gradient
        CALL DEFGRA
     1(   DELDIS     ,FINCIN     ,GMATX      ,MDOFN      ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
        IF(NTYPE.EQ.3.AND.FINCIN(3,3).LE.R0)THEN
C ...cut increment if determinant of incr. def. gradient non positive
          CALL ERRPRT('WE0025')
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
        CALL INVF2
     1(   FINCIN     ,FINCR      ,NTYPE      )
C modified incremental deformation gradient for F-bar element
        IF(NTYPE.EQ.2)THEN
          DET=FINCR(1,1)*FINCR(2,2)-FINCR(1,2)*FINCR(2,1)
        ELSEIF(NTYPE.EQ.3)THEN
```

```
          DET=(FINCR(1,1)*FINCR(2,2)-FINCR(1,2)*FINCR(2,1))*
     1         FINCR(3,3)
        ENDIF
        FACTOR=(DET0/DET)**AFACT
        FINCR(1,1)=FACTOR*FINCR(1,1)
        FINCR(1,2)=FACTOR*FINCR(1,2)
        FINCR(2,1)=FACTOR*FINCR(2,1)
        FINCR(2,2)=FACTOR*FINCR(2,2)
        IF(NTYPE.EQ.3)FINCR(3,3)=FACTOR*FINCR(3,3)
C... determinant of total deformation gradient
        DETF=DETF0
C
C Call material interface routine for state update calls: Update stress
C and other state variables
C ====================================================================
C
        NLARGE=1
        CALL MATISU
     1(   DETF        ,NLARGE     ,NTYPE      ,SUFAIL     ,
     2    THKGP(IGAUSP)           ,EINCR      ,FINCR      ,IPROPS     ,
     3    LALGVA(1,IGAUSP)        ,RALGVA(1,IGAUSP)       ,RPROPS     ,
     4    RSTAVA(1,IGAUSP)        ,STRSG(1,IGAUSP)        )
        IF(SUFAIL)THEN
C State updating failed for current Gauss point: break loop over Gauss
C points and exit with increment cutting flag activated
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
C
C Add current Gauss point contribution to the element internal force
C vector
C =================================================================
C
C evaluate elemental volume
        DVOLU=DETJAC*WEIGP
        IF(NTYPE.EQ.3)THEN
          DVOLU=DVOLU*TWOPI*GPCOD(NAXIS)
        ENDIF
C                           T
C add current gauss point  B [sigma]
        CALL RTV
     1(   0           ,NBDIM      ,NEVAB      ,NBDIM      ,ELOAD      ,
     2    BMATX       ,STRSG(1,IGAUSP)        ,DVOLU      )
C
   40 CONTINUE
C
C=====================================================================
C                 End of loop over Gauss points                      |
C=====================================================================
C
  999 CONTINUE
      RETURN
      END
```

```fortran
      SUBROUTINE IFSTD2
     1(   IELEM      ,INCCUT     ,MDIME      ,MELEM      ,MPOIN      ,
     2    MSTRE      ,MTOTV      ,NAXIS      ,NLARGE     ,NTYPE      ,
     3    COORD1     ,DINCR      ,ELOAD      ,IELPRP     ,IPROPS     ,
     4    LALGVA     ,LNODS      ,RALGVA     ,RELPRP     ,RPROPS     ,
     5    RSTAVA     ,STRSG      ,THKGP      ,TDISP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../MATERIAL.INC'
C
      PARAMETER( MGDIM=5 ,MBDIM=4 ,NDIME=2 ,NDOFN=2 )
C Arguments
      LOGICAL  INCCUT ,LALGVA
      DIMENSION
     1    COORD1(MDIME,MPOIN),DINCR(MTOTV)       ,ELOAD(MEVAB)       ,
     2    IELPRP(*)          ,IPROPS(*)          ,LALGVA(MLALGV,MTOTG),
     3    LNODS(MELEM,MEVAB),RALGVA(MRALGV,MTOTG),RELPRP(*)          ,
     4    RPROPS(*)          ,RSTAVA(MRSTAV,MTOTG),STRSG(MSTRE,MTOTG),
     5    THKGP(MTOTG)       ,TDISP(MTOTV)
C Local variables and arrays
      LOGICAL  SUFAIL
      DIMENSION
     1    BMATX(MBDIM,MEVAB) ,CARTD(NDIME,MNODE) ,DELDIS(MDOFN,MNODE),
     2    DERIV(NDIME,MNODE) ,EINCR(MBDIM)       ,ELCOD(NDIME,MNODE) ,
     3    FINCIN(3,3)        ,FINCR(3,3)         ,FINV(3,3)          ,
     4    GMATX(MGDIM,MEVAB) ,GPCOD(NDIME)       ,SHAPE(MNODE)       ,
     5    TELDIS(MDOFN,MNODE)
C Local numerical constants
      DATA
     1    R0   ,R1   ,R8   /
     2    0.0D0,1.0D0,8.0D0/
C***********************************************************************
C COMPUTE INTERNAL FORCE VECTOR OF ALL ELEMENTS OF CLASS 'STDARD'
C (STANDARD ISOPARAMETRIC ELEMENTS) IN 2-D: PLANE STRAIN, PLANE STRESS
C AND AXISYMMETRIC
C
C REFERENCE: Section 4.1.2
C            Box 4.2, item (viii)
C***********************************************************************
      IF(NTYPE.EQ.1)THEN
        NBDIM=3
      ELSEIF(NTYPE.EQ.2)THEN
        NBDIM=3
      ELSEIF(NTYPE.EQ.3)THEN
        TWOPI=R8*ATAN(R1)
        NBDIM=4
      ELSE
        CALL ERRPRT('EI0012')
      ENDIF
C Identify element type
      IELTYP=IELPRP(1)
C retrieve some element integer properties
      NNODE =IELPRP(3)
      NGAUSP=IELPRP(4)
      NEVAB =IELPRP(5)
C
C Set element arrays of current nodal coordinates, total and incremental
C displacements
      DO 20 INODE =1,NNODE
        LNODE=IABS(LNODS(IELEM,INODE))
        NPOSN=(LNODE-1)*NDOFN
        DO 10 IDOFN=1,NDOFN
          NPOSN=NPOSN+1
          IF(NLARGE.EQ.1)THEN
            ELCOD(IDOFN,INODE)=COORD1(IDOFN,LNODE)
            TELDIS(IDOFN,INODE)=-TDISP(NPOSN)
            DELDIS(IDOFN,INODE)=-DINCR(NPOSN)
          ELSE
            ELCOD(IDOFN,INODE)=COORD1(IDOFN,LNODE)
            DELDIS(IDOFN,INODE)=DINCR(NPOSN)
          ENDIF
   10   CONTINUE
   20 CONTINUE
C
C Initialize element force vector
C
      CALL RVZERO(ELOAD,NEVAB)
C
C=======================================================================
C                 Begin loop over Gauss points                          |
C=======================================================================
C
      IPPOS=1
      IPWEI=NGAUSP*NDIME+1
      DO 40 IGAUSP=1,NGAUSP
C Set Gauss points positions and weights
        EXISP=RELPRP(IPPOS-1+IGAUSP*2-1)
        ETASP=RELPRP(IPPOS-1+IGAUSP*2)
        WEIGP=RELPRP(IPWEI-1+IGAUSP)
C Evaluate shape functions and their derivatives (use current
C configuration for large strains)
        CALL SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
     2    NDIME      ,SHAPE      )
        CALL JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    NDIME      ,NDIME      ,NNODE      )
```

```
        IF(DETJAC.LE.R0)THEN
C ...cut increment if current jacobian is not positive definite
          CALL ERRPRT('WE0009')
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
        IF(NTYPE.EQ.3)CALL GETGCO
     1(   GPCOD      ,ELCOD      ,NDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C Evaluate symmetric gradient operator B
        CALL GETBMX
     1(   BMATX      ,GPCOD      ,CARTD      ,NDIME      ,NBDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C
C Compute basic kinematic variables needed for state update
C ==========================================================
C
        IF(NLARGE.EQ.1)THEN
C Large strains: compute incremental deformation gradient
C -------------------------------------------------------
C gradient operator G in current configuration
          CALL GETGMX
     1(   GPCOD      ,CARTD      ,GMATX      ,NDIME      ,MGDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C incremental deformation gradient
          CALL DEFGRA
     1(   DELDIS     ,FINCIN     ,GMATX      ,MDOFN      ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
          IF(NTYPE.EQ.3.AND.FINCIN(3,3).LE.R0)THEN
C ...cut increment if determinant of incr. def. gradient non positive
            CALL ERRPRT('WE0010')
            INCCUT=.TRUE.
            GOTO 999
          ENDIF
          CALL INVF2
     1(   FINCIN     ,FINCR      ,NTYPE      )
C... compute determinant of total deformation gradient
          CALL DEFGRA
     1(   TELDIS     ,FINV       ,GMATX      ,MDOFN      ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
          DETFIN=FINV(1,1)*FINV(2,2)-FINV(1,2)*FINV(2,1)
          IF(NTYPE.EQ.3)THEN
            IF(FINV(3,3).LE.R0)THEN
C... cut increment if deformation gradient is not positive definite
              CALL ERRPRT('WE0010')
              INCCUT=.TRUE.
              GOTO 999
            ENDIF
            DETFIN=DETFIN*FINV(3,3)
          ENDIF
          DETF=R1/DETFIN
        ELSE
C Small strains: compute incremental infinitesimal strain
C -------------------------------------------------------
          CALL LISTRA
     1(   BMATX      ,DELDIS     ,MDOFN      ,NBDIM      ,NDOFN      ,
     2    NNODE      ,NTYPE      ,EINCR      )
        ENDIF
C
C Call material interface routine for state update calls: Update stress
C and other state variables
C ======================================================================
C
        CALL MATISU
     1(   DETF       ,NLARGE     ,NTYPE      ,SUFAIL     ,
     2    THKGP(IGAUSP)          ,EINCR      ,FINCR      ,IPROPS     ,
     3    LALGVA(1,IGAUSP)       ,RALGVA(1,IGAUSP)       ,RPROPS     ,
     4    RSTAVA(1,IGAUSP)       ,STRSG(1,IGAUSP)        )
        IF(SUFAIL)THEN
C State updating failed for current Gauss point: break loop over Gauss
C points and exit with increment cutting flag activated
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
C
C Add current Gauss point contribution to the element internal force
C vector
C ===================================================================
C
C evaluate elemental volume
        DVOLU=DETJAC*WEIGP
        IF(NTYPE.EQ.1)THEN
          DVOLU=DVOLU*THKGP(IGAUSP)
        ELSEIF(NTYPE.EQ.3)THEN
          DVOLU=DVOLU*TWOPI*GPCOD(NAXIS)
        ENDIF
C                              T
C add current gauss point  B [sigma]
        CALL RTV
     1(   0          ,NBDIM      ,NEVAB      ,NBDIM      ,ELOAD      ,
     2    BMATX      ,STRSG(1,IGAUSP)        ,DVOLU      )
C
   40 CONTINUE
C
C=======================================================================
C                 End of loop over Gauss points                        |
C=======================================================================
C
```

```
999 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE INCREM
     1(   IINCS      ,TFACT      ,TOLER      ,MITER      ,
     2    NOUTP      ,DFACT      ,DFOLD      ,KUNLD      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas global database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Arguments
      DIMENSION NOUTP(5)
C Numerical constants
      DATA R0   ,RP01  /
     1     0.0D0,0.01D0/
C***********************************************************************
C INCREMENTS THE APPLIED EXTERNAL LOAD, SETS GLOBAL UNLOADING FLAG AND
C OUTPUTS CURRENT INCREMENT PARAMETERS TO RESULTS FILE (FOR FIXED LOAD
C INCREMENTS OPTION)
C
C REFERENCE: Figure 5.2
C***********************************************************************
 1000 FORMAT(////' INCREMENT NUMBER',I5/
     1           ' ====================')
 1010 FORMAT(//' Total load factor ............... = ',G15.6/
     1         ' Incremental load factor ......... = ',G15.6/
     2         ' Convergence tolerence ........... = ',G15.6/
     3         ' Max. No. of iterations .......... = ',I5)
 1020 FORMAT(/ ' Output control flags for results'/
     1         '        ( 0 - No, 1 - Yes )',/,
     2         ' Displacements..................... = ',I3/
     3         ' Reactions......................... = ',I3/
     4         ' State variables at gauss points.... = ',I3/
     5         ' State variables at nodes........... = ',I3/
     6         ' Output to re-start file............ = ',I3)
C
C Output current increment parameters to results file
C ---------------------------------------------------
C
      WRITE(16,1000)IINCS
C
      IF(TOLER.EQ.R0)TOLER=RP01
      TFACT=TFACT+DFACT
C
      WRITE(16,1010)TFACT,DFACT,TOLER,MITER
      WRITE(16,1020)(NOUTP(I),I=1,5)
C
C Increment forces
C ----------------
C New out-of-balance force := out-of-balance force at the end of the
C previous (converged) load increment + current external load increment
      DO 20 IELEM=1,NELEM
        IGRUP =IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NEVAB =IELPRP(5,IELIDN)
        DO 10 IEVAB=1,NEVAB
          ELOAD(IEVAB,IELEM)=ELOAD(IEVAB,IELEM)+RLOAD(IEVAB,IELEM)*DFACT
   10   CONTINUE
   20 CONTINUE
C
C Increment prescribed displacements
C ----------------------------------
C
      DO 40 ITOTV=1,NTOTV
        DO 30 IRHS=1,2
          FIXED(ITOTV,IRHS)=R0
   30   CONTINUE
   40 CONTINUE
      DO 60 IVFIX=1,NVFIX
        NLOCA=(NOFIX(IVFIX)-1)*NDOFN
        DO 50 IDOFN=1,NDOFN
          NGASH=NLOCA+IDOFN
          FIXED(NGASH,1)=PRESC(IVFIX,IDOFN)*DFACT
   50   CONTINUE
   60 CONTINUE
C
C Set unloading (load reversal) global flag
C -----------------------------------------
C
      KUNLD=0
      IF(IINCS.GT.1)THEN
        IF((DFOLD*DFACT).LT.R0)KUNLD=1
      ENDIF
      DFOLD=DFACT
C
      RETURN
      END
```

```
      SUBROUTINE INDATA(MXFRON ,UNSYM)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C Hyplas database: Global parameters and common blocks
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Local array and variables
      LOGICAL CARTES ,CYLIND ,FOUND ,UNSYM ,UNSAUX
      CHARACTER*80 ELTNAM ,MATNAM
      CHARACTER*72 TITLE
      CHARACTER*80 SUBKEY ,INLINE
      DIMENSION
     1    AUXCOR(MDIME)      ,DERIV(MDIME,MNODE) ,ELTHK(MNODE)       ,
     2    IELCHK(MELEM)      ,IETCHK(MGRUP)      ,IGRCHK(MGRUP)      ,
     3    INDCHK(MPOIN)      ,IWBEG(40)          ,IWEND(40)          ,
     4    KSLAV(48)          ,MATCHK(MGRUP)      ,SHAPE(MNODE)       ,
     5    THKNOD(MPOIN)
C Numerical constants
      PARAMETER
     1(   R0=0.0D0   ,R1=1.0D0   ,R45=45.0D0 )
C***********************************************************************
C READS MOST OF THE INPUT DATA
C
C REFERENCE: Section 5.3.2
C***********************************************************************
 1000 FORMAT(/' Title:'/' ======'/)
 1010 FORMAT(A)
 1015 FORMAT(1X,A)
 1020 FORMAT(A80)
 1030 FORMAT(//' Analysis description:'/' ====================='/)
 1040 FORMAT(
     1 ' Analysis type ....................................... =',I5/
     2 '        1 = Plane stress'/
     3 '        2 = Plane strain'/
     4 '        3 = Axisymmetric')
 1050 FORMAT(/
     1 ' Large deformation flag .............................. =',A5)
 1060 FORMAT(/
     1 ' Nonlinear solution algorithm ........................ =',I5/
     2 '        Negative for the arc length method'/
     3 '        1 = Initial stiffness method'/
     4 '        2 = Newton-Raphson tangential stiffness method'/
     5 '        3 = Modified Newton KT1'/
     6 '        4 = Modified Newton KT2'/
     7 '        5 = Secant Newton - Initial stiffness'/
     8 '        6 = Secant Newton - KT1'/
     9 '        7 = Secant Newton - KT2')
 1065 FORMAT(/
     1 ' Arc-length option ................................... =',I5/
     2 '        1 = Follow stiffness determinant sign'/
     3 '        2 = Follow current path')
 1070 FORMAT(//
     1 ' Element connectivities:      Number of elements = ',I5,/
     3 ' ======================='//
     4 ' Elem.  Group              Node numbers'/)
 1080 FORMAT(I4,I8,10X,9I5)
 1090 FORMAT(//
     1 ' Nodal point co-ordinates:       Number of nodes = ',I5,/
     2 ' ========================='//
     3 ' Node    X-Coord        Y-Coord'/)
 1100 FORMAT(//
     1 ' Nodal point co-ordinates:       Number of nodes = ',I5,/
     2 ' ========================='//
     3 ' Node    R-Coord        Z-Coord'/)
 1110 FORMAT(I5,3G15.6)
 1120 FORMAT(//
     1 ' Prescribed displacements:    Number of nodes with',
     2 ' prescribed displacement = ',I5/
     3 ' ========================='//
     4 ' Node   Code             Prescribed values           ',
     5 '      Angle'/)
 1130 FORMAT(1X,I4,1X,I6,3X,7G15.6)
 1140 FORMAT(//
     1 ' Element Groups:        Number of element groups = ',I5/
     2 ' ==============='//
     3 ' Group     Element type    Material type'/)
 1150 FORMAT(1X,I5,6X,I5,13X,I5)
 1160 FORMAT(//
     1 ' Element types:          Number of element types = ',I5/
     2 ' ==============')
 1170 FORMAT(/' Element type number  ',I2/,' -------------------')
 1180 FORMAT(//
     1 ' Material properties:        Number of materials = ',I5/
     2 ' ====================')
 1190 FORMAT(/' Material type number ',I2/,' --------------------')
 1200 FORMAT(
     1 ' Axis of symmetry .................................... =    ',A)
 1210 FORMAT(///' No Master/Slave nodal constraints specified'/
     1       ' ==========================================='/)
 1220 FORMAT(//
     1 ' Master/slave constraint sets:      Number of sets = ',I5/
     2 ' =============================')
 1230 FORMAT(/,' Master node =',I4,' Number of slave nodes =',I3,
     1 ' Fixity condition =',I3/)
 1240 FORMAT(' Slaves =',12I5:/(9X,12I5))
 1250 FORMAT(//
     1 ' Thickness distribution (initial thickness for large strains)'/
     2 ' ============================================================'/)
 1260 FORMAT(' Uniform thickness = ',G15.6/)
 1270 FORMAT(' Elem.   thickness'/)
 1280 FORMAT(' Node    thickness'/)
 1290 FORMAT(I5,1X,G15.6)
C
C Read basic analysis information
C ===============================
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'TITLE',
     2    INLINE    ,15       ,NWRD     )
      IF(.NOT.FOUND)CALL ERRPRT('ED0080')
```

```
      READ(15,1010)TITLE
      WRITE(16,1000)
      WRITE(16,1015)TITLE
C
      WRITE(16,1030)
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'ANALYSIS_TYPE',
     2    INLINE    ,15       ,NWRD     )
      IF(.NOT.FOUND)CALL ERRPRT('ED0081')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0018')
      NTYPE=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      WRITE(16,1040)NTYPE
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NDOFN=2
        NDIME=2
      ELSE
        CALL ERRPRT('ED0009')
      ENDIF
      IF(NTYPE.EQ.3)THEN
        CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'AXIS_OF_SYMMETRY',
     2    INLINE    ,15       ,NWRD     )
        IF(.NOT.FOUND)CALL ERRPRT('ED0082')
        IF(NWRD.EQ.1)CALL ERRPRT('ED0016')
        WRITE(16,1200)INLINE(IWBEG(2):IWEND(2))
        IF(INLINE(IWBEG(2):IWEND(2)).EQ.'Y')THEN
          NAXIS=1
        ELSEIF(INLINE(IWBEG(2):IWEND(2)).EQ.'X')THEN
          NAXIS=2
        ELSE
          CALL ERRPRT('ED0017')
        ENDIF
      ENDIF
C
      CALL FNDKEY
     1(   FOUND   ,IWBEG  ,IWEND  ,'LARGE_STRAIN_FORMULATION',
     2    INLINE  ,15     ,NWRD   )
      IF(.NOT.FOUND)CALL ERRPRT('ED0083')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0022')
      WRITE(16,1050)INLINE(IWBEG(2):IWEND(2))
      IF(INLINE(IWBEG(2):IWEND(2)).EQ.'ON')THEN
        NLARGE=1
      ELSEIF(INLINE(IWBEG(2):IWEND(2)).EQ.'OFF')THEN
        NLARGE=0
      ELSE
        CALL ERRPRT('ED0014')
      ENDIF
C
C Read non-linear equilibrium solution algorithm information
C ==========================================================
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'SOLUTION_ALGORITHM',
     2    INLINE    ,15       ,NWRD     )
      IF(.NOT.FOUND)CALL ERRPRT('ED0084')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0021')
      NALGO=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      WRITE(16,1060)NALGO
      IF(IABS(NALGO).NE.1.AND.IABS(NALGO).NE.2.AND.IABS(NALGO).NE.3.AND.
     1   IABS(NALGO).NE.4.AND.IABS(NALGO).NE.5.AND.IABS(NALGO).NE.6.AND.
     2   IABS(NALGO).NE.7)CALL ERRPRT('ED0010')
      NARCL=0
      IF(NALGO.LT.0)THEN
        CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'ARC_LENGTH_PREDICTOR_OPTION',
     2    INLINE    ,15       ,NWRD     )
        IF(NWRD.EQ.1)CALL ERRPRT('ED0140')
        IF(FOUND)THEN
          IF(INLINE(IWBEG(2):IWEND(2)).EQ.'STIFFNESS_SIGN')THEN
            NARCL=1
          ELSEIF(INLINE(IWBEG(2):IWEND(2)).EQ.'SECANT_PATH')THEN
            NARCL=2
          ELSE
            CALL ERRPRT('ED0141')
          ENDIF
          WRITE(16,1065)NARCL
        ELSE
          CALL ERRPRT('ED0139')
        ENDIF
      ENDIF
C
C Read all information concerning element groups
C ==============================================
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'ELEMENT_GROUPS',
     2    INLINE    ,15       ,NWRD     )
      IF(.NOT.FOUND)CALL ERRPRT('ED0085')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0052')
      NGRUP=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      WRITE(16,1140)NGRUP
      IF(NGRUP.LT.1)CALL ERRPRT('ED0053')
      IF(NGRUP.GT.MGRUP)CALL ERRPRT('ED0054')
C
C assign element & material type identification numbers to each group
C
      DO 101 IGRUP=1,NGRUP
        IGRCHK(IGRUP)=0
  101 CONTINUE
      DO 102 IGRUP=1,NGRUP
        READ(15,*)IGRP,IELIDN,MATIDN
        WRITE(16,1150)IGRP,IELIDN,MATIDN
        IF(IGRP.GT.NGRUP.OR.IELIDN.GT.NGRUP.OR.MATIDN.GT.NGRUP.OR.
     1     IGRP.LT.1.OR.IELIDN.LT.1.OR.MATIDN.LT.1)CALL ERRPRT('ED0062')
        IF(IGRCHK(IGRP).EQ.1)CALL ERRPRT('ED0063')
        IGRCHK(IGRP)=1
        IELTID(IGRP)=IELIDN
        MATTID(IGRP)=MATIDN
  102 CONTINUE
C
C Read type of element associated with each element type identification
C number and call the appropriate routines to read the element
```

```
C properties and set vectors IELPRP and RELPRP of integer and real
C element properties
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'ELEMENT_TYPES',
     2    INLINE    ,15       ,NWRD     )
      IF(.NOT.FOUND)CALL ERRPRT('ED0086')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0055')
      NELTS=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      WRITE(16,1160)NELTS
      IF(NELTS.LT.1)CALL ERRPRT('ED0056')
      IF(NELTS.GT.NGRUP)CALL ERRPRT('ED0057')
      DO 84 IELTS=1,NGRUP
        IETCHK(IELTS)=0
   84 CONTINUE
      UNSYM=.FALSE.
      DO 82 IELTS=1,NELTS
        READ(15,1020)SUBKEY
        NSKWRD=NWORD(SUBKEY,IWBEG,IWEND)
        IF(NSKWRD.EQ.0)CALL ERRPRT('ED0058')
        IF(NSKWRD.EQ.1)CALL ERRPRT('ED0059')
        IELIDN=INTNUM(SUBKEY(IWBEG(1):IWEND(1)))
        ELTNAM=SUBKEY(IWBEG(2):IWEND(2))
        WRITE(16,1170)IELIDN
        IF(IELIDN.LE.0.OR.IELIDN.GT.NELTS)CALL ERRPRT('ED0060')
        IF(IETCHK(IELIDN).EQ.1)CALL ERRPRT('ED0061')
        IETCHK(IELIDN)=1
C Set type, class and and read and set other properties
        IF(ELTNAM.EQ.'TRI_3')THEN
          IELTYP=TRI3
          IELCLS=STDARD
          CALL RST3
     1(   IELPRP(1,IELIDN)  ,16  ,RELPRP(1,IELIDN)  ,UNSAUX)
        ELSEIF(ELTNAM.EQ.'QUAD_4')THEN
          IELTYP=QUAD4
          IELCLS=STDARD
          CALL RSQ4
     1(   IELPRP(1,IELIDN)  ,15  ,16  ,RELPRP(1,IELIDN)  ,UNSAUX)
        ELSEIF(ELTNAM.EQ.'QUAD_8')THEN
          IELTYP=QUAD8
          IELCLS=STDARD
          CALL RSQ8
     1(   IELPRP(1,IELIDN)  ,15  ,16  ,RELPRP(1,IELIDN)  ,UNSAUX)
        ELSEIF(ELTNAM.EQ.'QUAD_4_FBAR')THEN
          IELTYP=QUA4FB
          IELCLS=FBAR
          IF(NLARGE.NE.1)THEN
            CALL ERRPRT('ED0180')
          ENDIF
          CALL RSQ4FB
     1(   IELPRP(1,IELIDN)  ,15  ,16  ,NTYPE ,RELPRP(1,IELIDN)  ,
     2    UNSAUX  )
        ELSE
          CALL ERRPRT('ED0064')
        ENDIF
        IELPRP(1,IELIDN)=IELTYP
        IELPRP(2,IELIDN)=IELCLS
        IF(UNSAUX)UNSYM=.TRUE.
   82 CONTINUE
C Check that the properties associated with all element type
C identification numbers have been read
      DO 83 IGRUP=1,NGRUP
        IELIDN=IELTID(IGRUP)
        IF(IETCHK(IELIDN).NE.1)CALL ERRPRT('ED0065')
   83 CONTINUE
C
C Read type of material associated with each material type
C identification number and call the appropriate routines to read the
C material properties and set vectors IPROPS and RPROPS of integer and
C real material properties
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'MATERIALS',
     2    INLINE    ,15       ,NWRD     )
      IF(.NOT.FOUND)CALL ERRPRT('ED0087')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0027')
      NMATS=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      WRITE(16,1180)NMATS
      IF(NMATS.LE.0.OR.NMATS.GT.NGRUP)CALL ERRPRT('ED0007')
      DO 99 IMATS=1,NMATS
        MATCHK(IMATS)=0
   99 CONTINUE
      DO 100 IMATS=1,NMATS
        READ(15,1020)SUBKEY
        NSKWRD=NWORD(SUBKEY,IWBEG,IWEND)
        IF(NSKWRD.EQ.0)CALL ERRPRT('ED0028')
        IF(NSKWRD.EQ.1)CALL ERRPRT('ED0029')
        MATIDN=INTNUM(SUBKEY(IWBEG(1):IWEND(1)))
        MATNAM=SUBKEY(IWBEG(2):IWEND(2))
        WRITE(16,1190)MATIDN
        IF(MATIDN.LE.0.OR.MATIDN.GT.NMATS)CALL ERRPRT('ED0044')
        IF(MATCHK(MATIDN).EQ.1)CALL ERRPRT('ED0045')
        MATCHK(MATIDN)=1
C
C Call material interface for reading material-specific data
C
        CALL MATIRD
     1(   MATNAM     ,NLARGE     ,NTYPE      ,UNSAUX     ,
     2    IPROPS(1,MATIDN)       ,RPROPS(1,MATIDN)       )
        IF(UNSAUX)UNSYM=.TRUE.
C
  100 CONTINUE
      DO 103 IGRUP=1,NGRUP
        MATIDN=MATTID(IGRUP)
        IF(MATCHK(MATIDN).NE.1)CALL ERRPRT('ED0066')
  103 CONTINUE
C
C Read elements nodal connectivities and group
C ============================================
C
      CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'ELEMENTS',
     2    INLINE    ,15       ,NWRD     )
```

```
      IF(.NOT.FOUND)CALL ERRPRT('ED0088')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0023')
      NELEM=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      WRITE(16,1070)NELEM
      IF(NELEM.LE.0)    CALL ERRPRT('ED0001')
      IF(NELEM.GT.MELEM)CALL ERRPRT('ED0002')
      CALL IVZERO(IELCHK,NELEM)
      MPOSPO=0
      DO 10 IELEM=1,NELEM
C Read element connectivity list
        READ(15,*)NUMEL,IGRPID(NUMEL),(LNODS(NUMEL,INODE),
     1            INODE=1,IELPRP(3,IELTID(IGRPID(NUMEL))))
        NNODE=IELPRP(3,IELTID(IGRPID(NUMEL)))
        WRITE(16,1080)NUMEL,IGRPID(NUMEL),(LNODS(NUMEL,INODE),
     1                INODE=1,NNODE)
        MPOSPO=MPOSPO+NNODE
C Check for repeated element connectivity specification
        IF(IELCHK(NUMEL).EQ.1)CALL ERRPRT('ED0047')
        IELCHK(NUMEL)=1
C Check for invalid element and group numbers
        IF(NUMEL.LE.0.OR.NUMEL.GT.NELEM)CALL ERRPRT('ED0046')
        IF(IGRPID(NUMEL).EQ.0.OR.IGRPID(NUMEL).GT.NGRUP)
     1  CALL ERRPRT('ED0037')
   10 CONTINUE
C
C Read nodal coordinates and thickness
C ====================================
C
      CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,'NODE_COORDINATES',
     2    INLINE     ,15         ,NWRD       )
      IF(.NOT.FOUND)CALL ERRPRT('ED0089')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0024')
      IF(NDIME.EQ.2.AND.NWRD.LT.3)CALL ERRPRT('ED0144')
      CYLIND=.FALSE.
      CARTES=.TRUE.
      IF(NDIME.EQ.2)THEN
C accepts data either in polar or cartesian system
        IF(INLINE(IWBEG(3):IWEND(3)).EQ.'CYLINDRICAL')THEN
          CYLIND=.TRUE.
          CARTES=.FALSE.
        ELSEIF(INLINE(IWBEG(3):IWEND(3)).EQ.'CARTESIAN')THEN
          CYLIND=.FALSE.
          CARTES=.TRUE.
        ELSE
          CALL ERRPRT('ED0145')
        ENDIF
      ENDIF
      NPOIN=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2)THEN
        WRITE(16,1090)NPOIN
      ELSEIF(NTYPE.EQ.3)THEN
        WRITE(16,1100)NPOIN
      ENDIF
      IF(NPOIN.LE.0)     CALL ERRPRT('ED0003')
      IF(NPOIN.GT.MPOIN) CALL ERRPRT('ED0004')
      IF(NPOIN.GT.MPOSPO)CALL ERRPRT('ED0049')
C
C Set global variable NTOTV (total number of degrees of freedom)
C
      NTOTV=NPOIN*NDOFN
C
C Read coordinates
      CALL IVZERO(INDCHK,NPOIN)
      DO 20 ICOUNT=1,NPOIN
        IF(CYLIND)THEN
C...in polar system
          READ(15,*)IPOIN,(AUXCOR(IDIME),IDIME=1,NDIME),RAD,THET
        ELSEIF(CARTES)THEN
C...in cartesian system
          READ(15,*)IPOIN,(AUXCOR(IDIME),IDIME=1,NDIME)
        ENDIF
        IF(IPOIN.GT.NPOIN.OR.IPOIN.LE.0)CALL ERRPRT('ED0127')
        DO 15 IDIME=1,NDIME
          COORD(IDIME,IPOIN,1)=AUXCOR(IDIME)
   15   CONTINUE
        IF(INDCHK(IPOIN).NE.0)CALL ERRPRT('ED0126')
        INDCHK(IPOIN)=1
        IF(CYLIND)THEN
          IF(RAD.NE.R0)THEN
C Input data in polar coordinates - transform into cartesian coordinates
            THET=THET*ATAN(R1)/R45
            COORD(1,IPOIN,1)=COORD(1,IPOIN,1)+RAD*COS(THET)
            COORD(2,IPOIN,1)=COORD(2,IPOIN,1)+RAD*SIN(THET)
          ENDIF
        ENDIF
C Echo coordinates
        WRITE(16,1110)IPOIN,(COORD(IDIME,IPOIN,1),IDIME=1,NDIME)
   20 CONTINUE
      DO 30 IPOIN=1,NPOIN
C check that the coordinates of all nodes have been defined
        IF(INDCHK(IPOIN).NE.1)CALL ERRPRT('ED0125')
C initialise initial and last converged coordinates sub-arrays
        DO 25 IDIME=1,NDIME
          COORD(IDIME,IPOIN,0)=COORD(IDIME,IPOIN,1)
          COORD(IDIME,IPOIN,2)=COORD(IDIME,IPOIN,1)
   25   CONTINUE
   30 CONTINUE
C
C Read initial thickness (for plane stress only)
C ==============================================
C
      IF(NTYPE.EQ.1)THEN
        CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,'THICKNESS',
     2    INLINE     ,15         ,NWRD       )
        IF(.NOT.FOUND)CALL ERRPRT('ED0090')
        IF(NWRD.EQ.1)CALL ERRPRT('ED0072')
        WRITE(16,1250)
C
C Uniform initial thickness in the whole mesh
C -------------------------------------------
        IF(INLINE(IWBEG(2):IWEND(2)).EQ.'UNIFORM')THEN
```

```
          READ(15,*)THICK
          WRITE(16,1260)THICK
          DO 35 IELEM=1,NELEM
            IGRUP=IGRPID(IELEM)
            IELIDN=IELTID(IGRUP)
            NGAUSP=IELPRP(4,IELIDN)
            DO 34 IGAUSP=1,NGAUSP
              THKGP(IGAUSP,IELEM,0)=THICK
              THKGP(IGAUSP,IELEM,1)=THICK
   34       CONTINUE
   35     CONTINUE
C
C Non-uniform initial thickness (constant within each element)
C -----------------------------------------------------------
        ELSEIF(INLINE(IWBEG(2):IWEND(2)).EQ.'DEFINED_BY_ELEMENT')THEN
          WRITE(16,1270)
          CALL IVZERO(IELCHK,NELEM)
C Read element thicknesses and set corresponding gauss point thicknesses
          IF(NWRD.EQ.2)THEN
            DO 41 I=1,NELEM
              READ(15,*)IELEM,THICK
              WRITE(16,1290)IELEM,THICK
              IF(IELEM.LE.0.OR.IELEM.GT.NELEM)CALL ERRPRT('ED0068')
              IF(IELCHK(IELEM).EQ.1)CALL ERRPRT('ED0069')
              IELCHK(IELEM)=1
              IGRUP=IGRPID(IELEM)
              IELIDN=IELTID(IGRUP)
              NGAUSP=IELPRP(4,IELIDN)
              DO 40 IGAUSP=1,NGAUSP
                THKGP(IGAUSP,IELEM,0)=THICK
                THKGP(IGAUSP,IELEM,1)=THICK
   40         CONTINUE
   41       CONTINUE
          ELSE
            NSET=INTNUM(INLINE(IWBEG(3):IWEND(3)))
            DO 45 ISET=1,NSET
              READ(15,*)IFIRST,ILAST,INC,THICK
              DO 44 IELEM=IFIRST,ILAST,INC
                WRITE(16,1290)IELEM,THICK
                IF(IELEM.LE.0.OR.IELEM.GT.NELEM)CALL ERRPRT('ED0068')
                IF(IELCHK(IELEM).EQ.1)CALL ERRPRT('ED0069')
                IELCHK(IELEM)=1
                IGRUP=IGRPID(IELEM)
                IELIDN=IELTID(IGRUP)
                NGAUSP=IELPRP(4,IELIDN)
                DO 43 IGAUSP=1,NGAUSP
                  THKGP(IGAUSP,IELEM,0)=THICK
                  THKGP(IGAUSP,IELEM,1)=THICK
   43           CONTINUE
   44         CONTINUE
   45       CONTINUE
          ENDIF
C Check that the thickness has been defined for all elements
          DO 46 IELEM=1,NELEM
            IF(IELCHK(IELEM).NE.1)CALL ERRPRT('ED0075')
   46     CONTINUE
C
C Non-uniform initial thickness (continuous across elements)
C ---------------------------------------------------------
        ELSEIF(INLINE(IWBEG(2):IWEND(2)).EQ.'DEFINED_BY_NODE')THEN
          WRITE(16,1280)
          CALL IVZERO(INDCHK,NPOIN)
C Read nodal thicknesses
          IF(NWRD.EQ.2)THEN
            DO 50 I=1,NPOIN
              READ(15,*)IPOIN,THICK
              WRITE(16,1290)IPOIN,THICK
              IF(IPOIN.LE.0.OR.IPOIN.GT.NPOIN)CALL ERRPRT('ED0070')
              IF(INDCHK(IPOIN).EQ.1)CALL ERRPRT('ED0071')
              INDCHK(IPOIN)=1
              THKNOD(IPOIN)=THICK
   50       CONTINUE
          ELSE
            NSET=INTNUM(INLINE(IWBEG(3):IWEND(3)))
            DO 55 ISET=1,NSET
              READ(15,*)IFIRST,ILAST,INC,THICK
              DO 54 IPOIN=IFIRST,ILAST,INC
                WRITE(16,1290)IPOIN,THICK
                IF(IPOIN.LE.0.OR.IPOIN.GT.NPOIN)CALL ERRPRT('ED0070')
                IF(INDCHK(IPOIN).EQ.1)CALL ERRPRT('ED0071')
                INDCHK(IPOIN)=1
                THKNOD(IPOIN)=THICK
   54         CONTINUE
   55       CONTINUE
          ENDIF
C Check that the thickness has been defined for all nodes
          DO 56 IPOIN=1,NPOIN
            IF(INDCHK(IPOIN).NE.1)CALL ERRPRT('ED0076')
   56     CONTINUE
          DO 60 IELEM=1,NELEM
C Set element nodal thicknesses array
            IGRUP=IGRPID(IELEM)
            IELIDN=IELTID(IGRUP)
            IELTYP=IELPRP(1,IELIDN)
            NNODE =IELPRP(3,IELIDN)
            NGAUSP=IELPRP(4,IELIDN)
            DO 58 INODE=1,NNODE
              LNODE=IABS(LNODS(IELEM,INODE))
              ELTHK(INODE)=THKNOD(LNODE)
   58       CONTINUE
C Interpolate nodal thicknesses to gauss points
            IPPOS=1
            DO 59 IGAUSP=1,NGAUSP
              EXISP=RELPRP(IPPOS-1+IGAUSP*2-1,IELIDN)
              ETASP=RELPRP(IPPOS-1+IGAUSP*2  ,IELIDN)
              CALL SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
     2    MDIME      ,SHAPE      )
              THKGP(IGAUSP,IELEM,0)=SCAPRD(ELTHK,SHAPE,NNODE)
              THKGP(IGAUSP,IELEM,1)=SCAPRD(ELTHK,SHAPE,NNODE)
   59       CONTINUE
   60     CONTINUE
        ELSE
```

```
          CALL ERRPRT('ED0073')
        ENDIF
      ENDIF
C
C
C Read prescribed displacements
C =============================
C
      CALL FNDKEY
     1(   FOUND  ,IWBEG  ,IWEND  ,'NODES_WITH_PRESCRIBED_DISPLACEMENTS',
     2    INLINE ,15     ,NWRD   )
      IF(.NOT.FOUND)CALL ERRPRT('ED0091')
      IF(NWRD.EQ.1)CALL ERRPRT('ED0025')
      NVFIX=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      IF(NDIME.EQ.2)THEN
        WRITE(16,1120)NVFIX
      ENDIF
      IF(NVFIX.LT.1)    CALL ERRPRT('ED0005')
      IF(NVFIX.GT.MVFIX)CALL ERRPRT('ED0006')
      IF(NVFIX.GT.NPOIN)CALL ERRPRT('ED0048')
      DO 70 ITOTV=1,NTOTV
        IFFIX(ITOTV)=0
        DO 65 IRHS=1,2
          FIXED(ITOTV,IRHS)=R0
   65   CONTINUE
   70 CONTINUE
      DO 91 IVFIX=1,NVFIX
        IF(NDIME.EQ.2)THEN
          READ(15,*)NOFIX(IVFIX),IFPRE,(PRESC(IVFIX,IDOFN),
     1               IDOFN=1,NDOFN),THETA
          WRITE(16,1130)NOFIX(IVFIX),IFPRE,(PRESC(IVFIX,IDOFN),
     1                   IDOFN=1,NDOFN),THETA
          ANGLE(IVFIX)=THETA*ATAN(R1)/R45
        ENDIF
        NLOCA=(NOFIX(IVFIX)-1)*NDOFN
        IFDOF=10**(NDOFN-1)
        DO 90 IDOFN=1,NDOFN
          NGASH=NLOCA+IDOFN
          IF(IFPRE.LT.IFDOF)GOTO 80
          IFFIX(NGASH)=1
          FIXED(NGASH,1)=PRESC(IVFIX,IDOFN)
          FIXED(NGASH,2)=R0
          IFPRE=IFPRE-IFDOF
   80     CONTINUE
          IFDOF=IFDOF/10
   90   CONTINUE
   91 CONTINUE
C Initialize master array
      DO 93 ITOTV=1,NTOTV
        MASTER(ITOTV)=ITOTV
   93 CONTINUE
C
C Master/slave nodal constraints
C ==============================
C
      CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,'MASTER_SLAVE_SETS',
     2    INLINE     ,15         ,NWRD       )
      IF(.NOT.FOUND)THEN
        NMAST=0
      ELSE
        IF(NWRD.EQ.1)CALL ERRPRT('ED0026')
        NMAST=INTNUM(INLINE(IWBEG(2):IWEND(2)))
      ENDIF
C Master/slave nodes specified
      IF(NMAST.NE.0)THEN
        WRITE(16,1220)NMAST
        IF(NMAST.LT.0)CALL ERRPRT('ED0036')
C Loop over sets of slave nodes
        DO 97 IMAST=1,NMAST
          READ(15,*)NODEM,ISLAV,ISFIX
          WRITE(16,1230)NODEM,ISLAV,ISFIX
          READ(15,*)(KSLAV(I),I=1,ISLAV)
          WRITE(16,1240)(KSLAV(I),I=1,ISLAV)
C Place these degrees of freedom in the master array
          IDOFN=(NODEM-1)*NDOFN+1
          IDOF1=IDOFN+1
          DO 96 IS=1,ISLAV
            KDOFN=(KSLAV(IS)-1)*NDOFN+1
            IF(NDOFN.EQ.2)THEN
              IF(ISFIX.EQ.10.OR.ISFIX.EQ.11)THEN
                IF(MASTER(KDOFN).NE.KDOFN)CALL ERRPRT('ED0050')
                MASTER(KDOFN)=IDOFN
              ENDIF
              KDOFN=KDOFN+1
              IF(ISFIX.EQ.1.OR.ISFIX.EQ.11)THEN
                IF(MASTER(KDOFN).NE.KDOFN)CALL ERRPRT('ED0051')
                MASTER(KDOFN)=IDOF1
              ENDIF
            ENDIF
   96     CONTINUE
   97   CONTINUE
      ELSE
C No master/slave nodes specified
        WRITE(16,1210)
      ENDIF
C
C Does some further checks on input data
C ======================================
      CALL CHECK2(MXFRON)
C
C Set up nodal valencies for nodal averaging
C ==========================================
C
      DO 110 IPOIN=1,NPOIN
        DO 105 IGRUP=1,NGRUP
          NVALEN(IPOIN,IGRUP)=0
  105   CONTINUE
  110 CONTINUE
C Evaluate nodal valencies
      DO 130 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
```

```
      NNODE=IELPRP(3,IELIDN)
      DO 120 INODE=1,NNODE
        LNODE=IABS(LNODS(IELEM,INODE))
        NVALEN(LNODE,IGRUP)=NVALEN(LNODE,IGRUP)+1
120   CONTINUE
130 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE ININCR
     1(   DFACT      ,DLENP      ,FSTOP      ,ITDES      ,MINCS      ,
     2    MITER      ,NALGO      ,NINCS      ,TOLER      ,
     3    DFACTV     ,MITERV     ,NOUTP      ,NOUTPV     ,TOLERV     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    DFACTV(MINCS)      ,MITERV(MINCS)      ,NOUTP(5)           ,
     2    NOUTPV(5,MINCS)    ,TOLERV(MINCS)
C
      LOGICAL FOUND
      CHARACTER*80 INLINE
      DIMENSION
     1    IWBEG(40), IWEND(40)
      PARAMETER
     1(   R0=0.0D0    )
C***********************************************************************
C READS INPUT DATA FOR LOAD INCREMENTATION
C
C REFERENCE: Figure 5.1
C***********************************************************************
 1000 FORMAT(///' Increment control with fixed load increments selected'
     1        /' ====================================================='
     2       //'       Number of proportional load increments =',I5/)
C
 1010 FORMAT(///' Increment control by the Arc-Length method selected'/
     1         ' ==================================================='//
     2         ' Arc length data'/' ---------------'//
     3' Maximum allowed number of increments ........= ',I5)
 1020 FORMAT(
     1' Initial load increment factor ............. = ',G15.6/
     2' Convergence tolerence ..................... = ',G15.6/
     3' Max. No. of iterations .................... = ',I5)
 1030 FORMAT(/' Output control parameter for results'/,
     1       '       ( Output frequencies )'/
     2    ' Displacements......................... = ',I3/
     3    ' Reactions............................. = ',I3/
     4    ' State variables at gauss points....... = ',I3/
     5    ' State variables at nodes.............. = ',I3/
     6    ' Output to re-start file............... = ',I3)
 1040 FORMAT(/
     1' Desired number of iterations per increment .. =',I5/
     2' Maximum load factor ......................... =',G15.6/
     3' Maximum arc length parameter ................ =',G15.6)
C
      CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,'INCREMENTS',
     2    INLINE     ,15         ,NWRD       )
      IF(.NOT.FOUND)CALL ERRPRT('ED0079')
C
      IF(NALGO.GT.0)THEN
C Fixed increments option
C -----------------------
        IF(NWRD.EQ.1)CALL ERRPRT('ED0034')
        NINCS=INTNUM(INLINE(IWBEG(2):IWEND(2)))
        WRITE(16,1000)NINCS
        IF(NINCS.LE.0)    CALL ERRPRT('ED0013')
        IF(NINCS.GT.MINCS)CALL ERRPRT('ED0035')
        DO 10 IINCS=1,NINCS
          READ(15,*,ERR=997,END=997)DFACTV(IINCS),TOLERV(IINCS),
     1                             MITERV(IINCS),(NOUTPV(I,IINCS),I=1,5)
          IF(TOLERV(IINCS).LE.R0)CALL ERRPRT('ED0142')
          IF(MITERV(IINCS).LT.1)CALL ERRPRT('ED0146')
   10   CONTINUE
      ELSE
C Arc-length control
C ------------------
        IF(NWRD.EQ.1) CALL ERRPRT('ED0097')
        NINCS=INTNUM(INLINE(IWBEG(2):IWEND(2)))
        WRITE(16,1010)NINCS
        IF(NINCS.LE.0)CALL ERRPRT('ED0098')
C
        READ(15,*,ERR=998,END=998)DFACT,TOLER,MITER,(NOUTP(I),I=1,5),
     1                            ITDES,FSTOP,DLENP
        IF(TOLER.LT.R0)CALL ERRPRT('ED0142')
        IF(ITDES.LT.0)CALL ERRPRT('ED0143')
        WRITE(16,1020)DFACT,TOLER,MITER
        WRITE(16,1030)(NOUTP(I),I=1,5)
        WRITE(16,1040)ITDES,FSTOP,DLENP
      ENDIF
C Send error messages in case of I/O error while reading increment data
      GOTO 999
  997 CALL ERRPRT('ED0178')
      GOTO 999
  998 CALL ERRPRT('ED0179')
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INITIA
     1(   DLAMD      ,IFNEG      ,KUNLD      ,TFACT      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas global database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Local variables and numerical constants
      LOGICAL  LDUMMY
      DATA R0   /
     1     0.0D0/
C***********************************************************************
C INITIALISES SOME ARRAYS AND VARIABLES
C
C REFERENCE: Section 5.3.4
C***********************************************************************
      KUNLD=0
      TFACT=R0
      DLAMD=R0
      IFNEG=1
      DO 10 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NEVAB=IELPRP(5,IELIDN)
        CALL RVZERO(ELOAD(1,IELEM),NEVAB)
        CALL RVZERO(ELOADO(1,IELEM),NEVAB)
   10 CONTINUE
      CALL RVZERO(DTANG,NTOTV)
      CALL RVZERO(TDISP,NTOTV)
      CALL RVZERO(TDISPO,NTOTV)
      CALL RVZERO(DINCR,NTOTV)
      CALL RVZERO(DINCRO,NTOTV)
      CALL RVZERO(DITER,NTOTV)
C Arrays from common block STATE
      DO 30 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NGAUSP=IELPRP(4,IELIDN)
        DO 20 IGAUSP=1,NGAUSP
C Call material interface routine to initialise material-specific Gauss
C point data
          MODE=0
          CALL MATISW
     1(   MODE       ,NLARGE     ,NTYPE      ,
     2    IPROPS(1,MATTID(IGRUP)),LALGVA(1,IGAUSP,IELEM,1)          ,
     3    LDUMMY     ,RALGVA(1,IGAUSP,IELEM,1)          ,DUMMY      ,
     4    RPROPS(1,MATTID(IGRUP))            ,
     5    RSTAVA(1,IGAUSP,IELEM,1)           ,DUMMY      ,
     6    STRSG(1,IGAUSP,IELEM,1)            ,DUMMY      )
   20   CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INLOAD
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas global database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      LOGICAL FOUND
      CHARACTER*80 INLINE
      DIMENSION
     1    SHAPE(MNODE)       ,DERIV(MDIME,MNODE) ,CARTD(MDIME,MNODE) ,
     2    GPCOD(MDIME)
      DIMENSION
     1    DGASH(MDOFN)       ,ELCOD(MDIME,MNODE) ,NOPRS(MNODE)       ,
     2    PGASH(MDOFN)       ,POINT(MDOFN)       ,PRESS(MNODE,MDOFN)
      DIMENSION
     1    IWBEG(40)          ,IWEND(40)          ,NODCHK(MNODE)      ,
     2    NODAUX(MPOIN)      ,THKN(MNODE)
      DATA R0,R1,R8,R45/0.0D0,1.0D0,8.0D0,45.0D0/
C***********************************************************************
C READS EXTERNAL LOADINGS (BODY FORCE AND SURFACE TRACTIONS) FROM INPUT
C DATA FILE AND ASSEMBLES THE GLOBAL EXTERNAL FORCE VECTOR
C
C REFERENCE: Figure 5.1
C            Section 5.3.3
C***********************************************************************
 1040 FORMAT(///
     1' Loading specification (other than prescribed displacements)'/
     1' ============================================================'//
     2'    If any of the flags below is set to 1,  then'/
     3'    the corresponding type of loading is applied'/
     4'    to the structure.'//
     5' Point loading flag ..........................=',I3/
     6' Gravity loading flag ........................=',I3/
     7' Distributed edge loading flag ...............=',I3)
 1050 FORMAT(//' Point load applied in',I6,' nodes'/
     1         ' ---------------------------------'//
     2         ' Node        X-Component    Y-Component')
 1055 FORMAT(//' Point load applied in',I6,' nodes'/
     1         ' ---------------------------------'//
     2         ' Node        R-Component    Z-Component')
 1070 FORMAT(I5,5X,3G15.6)
 1080 FORMAT(//' Gravity load'/
     1         ' ------------'//
     2         ' Gravity angle (degrees) =',G15.6/
     3         ' Gravity constant .......=',G15.6)
 1110 FORMAT(//' Edge load applied in ',I6,' edges'/
     1         ' ---------------------------------')
 1130 FORMAT(/' Element Number =',I5/' Node      X-Coord.   ',
     1'      Y-Coord.       Norm. Load        Tang. Load')
 1135 FORMAT(/' Element Number =',I5/' Node      R-Coord.   ',
     1'      Z-Coord.       Norm. Load        Tang. Load')
 1140 FORMAT(I5,2X,2G15.6,3X,2G15.6)
C
C
      IF(NTYPE.EQ.3)TWOPI=R8*ATAN(R1)
C
C Initialize load vector of all elements
C ======================================
C
      DO 10 IELEM=1,NELEM
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NEVAB=IELPRP(5,IELIDN)
        CALL RVZERO(RLOAD(1,IELEM),NEVAB)
   10 CONTINUE
C
C Read data controlling loading types to be inputted
C ==================================================
C
      IPLOD=0
      IGRAV=0
      IEDGE=0
C
      CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,'LOADINGS',
     2    INLINE     ,15         ,NWRD       )
      IF(.NOT.FOUND)CALL ERRPRT('ED0092')
      NLOADS=NWRD-1
      IF(NLOADS.NE.0)THEN
        DO 12 I=1,NLOADS
          IF(INLINE(IWBEG(1+I):IWEND(1+I)).EQ.'POINT')THEN
            IPLOD=1
          ELSEIF(INLINE(IWBEG(1+I):IWEND(1+I)).EQ.'EDGE')THEN
            IEDGE=1
          ELSEIF(INLINE(IWBEG(1+I):IWEND(1+I)).EQ.'GRAVITY')THEN
            IGRAV=1
          ELSEIF((INLINE(IWBEG(1+I):IWEND(1+I)).EQ.'0').OR.
     1           (INLINE(IWBEG(1+I):IWEND(1+I)).EQ.'NONE'))THEN
            CONTINUE
          ELSE
            CALL ERRPRT('ED0030')
          ENDIF
   12   CONTINUE
      ENDIF
C
      WRITE(16,1040)IPLOD,IGRAV,IEDGE
C
C Read nodal point loads
C ======================
C
      IF(IPLOD.NE.0)THEN
        CALL FNDKEY
     1(   FOUND      ,IWBEG      ,IWEND      ,'POINT_LOADS',
     2    INLINE     ,15         ,NWRD       )
        IF(.NOT.FOUND)CALL ERRPRT('ED0093')
        IF(NWRD.EQ.1)CALL ERRPRT('ED0031')
        NPLOAD=INTNUM(INLINE(IWBEG(2):IWEND(2)))
        IF(NTYPE.EQ.1.OR.NTYPE.EQ.2)THEN
          WRITE(16,1050)NPLOAD
        ELSEIF(NTYPE.EQ.3)THEN
```

```
         WRITE(16,1055)NPLOAD
       ENDIF
       DO 55 IPLOAD=1,NPLOAD
         READ(15,*)LODPT,(POINT(IDOFN),IDOFN=1,NDOFN)
         WRITE(16,1070)LODPT,(POINT(IDOFN),IDOFN=1,NDOFN)
         IF(LODPT.LE.0.OR.LODPT.GT.NPOIN)CALL ERRPRT('ED0134')
C
C Associate the nodal point loads with an element
C
         DO 35 IELEM=1,NELEM
           IGRUP=IGRPID(IELEM)
           IELIDN=IELTID(IGRUP)
           NNODE=IELPRP(3,IELIDN)
           DO 30 INODE=1,NNODE
             NLOCA=IABS(LNODS(IELEM,INODE))
             IF(LODPT.EQ.NLOCA)GOTO 40
   30      CONTINUE
   35    CONTINUE
   40    CONTINUE
         DO 50 IDOFN=1,NDOFN
           NGASH=(INODE-1)*NDOFN+IDOFN
           RLOAD(NGASH,IELEM)=RLOAD(NGASH,IELEM)+POINT(IDOFN)
   50    CONTINUE
   55  CONTINUE
     ENDIF
C
C Gravity loading
C ===============
C
     IF(IGRAV.NE.0)THEN
C
C Read gravity angle and gravitational constant
C
       CALL FNDKEY
    1(   FOUND      ,IWBEG      ,IWEND      ,'GRAVITY_LOAD',
    2    INLINE     ,15         ,NWRD       )
       IF(.NOT.FOUND)CALL ERRPRT('ED0094')
       READ(15,*)THETA,GRAVY
       WRITE(16,1080)THETA,GRAVY
       THETA=THETA*ATAN(R1)/R45
C
C Loop over elements
C
       DO 90 IELEM=1,NELEM
         IGRUP =IGRPID(IELEM)
         IELIDN=IELTID(IGRUP)
         IELTYP=IELPRP(1,IELIDN)
         NNODE =IELPRP(3,IELIDN)
         NGAUSP=IELPRP(4,IELIDN)
C Set up preliminary constants
         MATIDN=MATTID(IGRPID(IELEM))
         DENSE=RPROPS(1,MATIDN)
         IF(DENSE.EQ.R0) GOTO 90
         GXCOM=DENSE*GRAVY*SIN(THETA)
         GYCOM=-DENSE*GRAVY*COS(THETA)
C Compute coordinates of the element nodal points
         DO 65 INODE=1,NNODE
           LNODE=IABS(LNODS(IELEM,INODE))
           DO 60 IDIME=1,NDIME
             ELCOD(IDIME,INODE)=COORD(IDIME,LNODE,1)
   60      CONTINUE
   65    CONTINUE
C
C Loop for numerical integration over element domain
C
         IPPOS=1
         IPWEI=NGAUSP*NDIME+1
         DO 85 IGAUSP=1,NGAUSP
           EXISP=RELPRP(IPPOS-1+IGAUSP*2-1,IELIDN)
           ETASP=RELPRP(IPPOS-1+IGAUSP*2  ,IELIDN)
           WEIGP=RELPRP(IPWEI-1+IGAUSP    ,IELIDN)
C Compute the shape functions at the sampling points and elemental
C volume
           CALL SHPFUN
    1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
    2    MDIME      ,SHAPE      )
           CALL JACOB2
    1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
    2    MDIME      ,NDIME      ,NNODE      )
           CALL GETGCO
    1(   GPCOD      ,ELCOD      ,MDIME      ,NDIME      ,NNODE      ,
    2    SHAPE      )
C
           DVOLU=DETJAC*WEIGP
           IF(NTYPE.EQ.1)THEN
             DVOLU=DVOLU*THKGP(IGAUSP,IELEM,1)
           ELSEIF(NTYPE.EQ.3)THEN
             DVOLU=DVOLU*TWOPI*GPCOD(NAXIS)
           ENDIF
C Calculate equivalent nodal loads and add them to element force vector
           DO 70 INODE=1,NNODE
             NGASH=(INODE-1)*NDOFN+1
             MGASH=(INODE-1)*NDOFN+2
             RLOAD(NGASH,IELEM)=RLOAD(NGASH,IELEM)+
    1                           GXCOM*SHAPE(INODE)*DVOLU
             RLOAD(MGASH,IELEM)=RLOAD(MGASH,IELEM)+
    1                           GYCOM*SHAPE(INODE)*DVOLU
   70      CONTINUE
   85    CONTINUE
   90  CONTINUE
     ENDIF
C
C Distributed edge loads (pressure)
C =================================
C
     IF(IEDGE.NE.0)THEN
       CALL FNDKEY
    1(   FOUND      ,IWBEG      ,IWEND      ,'EDGE_LOADS',
    2    INLINE     ,15         ,NWRD       )
       IF(.NOT.FOUND)CALL ERRPRT('ED0095')
       IF(NWRD.EQ.1)CALL ERRPRT('ED0032')
       NLOADE=INTNUM(INLINE(IWBEG(2):IWEND(2)))
```

```
      WRITE(16,1110)NLOADE
C
C Loop over loaded edges
C
      DO 160 ILOADE=1,NLOADE
C Read and echo the element number and corresponding global node numbers
C with prescribed pressure
        READ(15,*)IELEM,NNODEG,(NOPRS(INODEG),INODEG=1,NNODEG)
        IF(NTYPE.NE.3)THEN
          WRITE(16,1130)IELEM
        ELSE
          WRITE(16,1135)IELEM
        ENDIF
        IF(IELEM.LE.0.OR.IELEM.GT.NELEM)CALL ERRPRT('ED0019')
        DO 80 INODEG=1,NNODEG
          IPOIN=NOPRS(INODEG)
          IF(IPOIN.LE.0.OR.IPOIN.GT.NPOIN)CALL ERRPRT('ED0020')
   80   CONTINUE
C Set properties of the current element
        IGRUP=IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        IELTYP=IELPRP(1,IELIDN)
        NNODE =IELPRP(3,IELIDN)
        NGAUSP=IELPRP(4,IELIDN)
        NEDGEL=IELPRP(6,IELIDN)
        MNODEG=IELPRP(7,IELIDN)
        NGAUSB=IELPRP(8,IELIDN)
        IPOS=9
C Read and echo pressures
        READ(15,*)((PRESS(INODEG,IDOFN),INODEG=1,NNODEG),
     1                                                    IDOFN=1,NDOFN)
        DO 95 INODEG=1,NNODEG
          IPOIN=NOPRS(INODEG)
          WRITE(16,1140)IPOIN,(COORD(I,IPOIN,1),I=1,NDIME),
     1                                     (PRESS(INODEG,I),I=1,NDIME)
   95   CONTINUE
        IF(NNODEG.GT.MNODEG)CALL ERRPRT('ED0011')
C Check that global node numbers supplied correspond exactly to an edge
C of the current element
        DO 96 INODE=1,NNODE
          NODCHK(INODE)=0
   96   CONTINUE
        DO 98 INODEG=1,NNODEG
          DO 97 INODE=1,NNODE
            IPOIN=IABS(LNODS(IELEM,INODE))
            IF(IPOIN.EQ.NOPRS(INODEG))NODCHK(INODE)=1
   97     CONTINUE
   98   CONTINUE
        CALL CHKNDB
     1(   FOUND    ,NNODE    ,NEDGEL   ,NODCHK   ,IELPRP(IPOS,IELIDN))
        IF(.NOT.FOUND)CALL ERRPRT('ED0012')
C
C Get the global coordinates of the nodes of the loaded edge
        DO 104 INODEG=1,NNODEG
          IPOIN=NOPRS(INODEG)
          DO 100 IDIME=1,NDIME
            ELCOD(IDIME,INODEG)=COORD(IDIME,IPOIN,1)
  100     CONTINUE
C
          DO 102 II=1,NNODEG
            IF(IABS(LNODS(IELEM,NODCHK(II))).EQ.IPOIN)NODAUX(IPOIN)=II
  102     CONTINUE
  104   CONTINUE
C Extrapolate thickness to nodes (for plane stress only)
        IF(NTYPE.EQ.1)THEN
          IPOS=NGAUSP*NDIME+NGAUSP+1
          CALL EXTNOD
     1(   RELPRP(IPOS,IELIDN),
     2    THKGP(1,IELEM,1)   ,THKN     ,1         ,NGAUSP     ,NNODE   )
        ENDIF
C
C Loop for (boundary) numerical integration over loaded edge
C
        DO 150 IGAUSB=1,NGAUSB
C Evaluate the shape functions at the boundary sampling points
          IPPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+1
          IPWEI=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+NGAUSB+1
          EXISPB=RELPRP(IPPOS-1+IGAUSB,IELIDN)
          WEIGPB=RELPRP(IPWEI-1+IGAUSB,IELIDN)
          CALL SHPFUN
     1(   DERIV      ,DUMMY      ,EXISPB     ,1          ,IELTYP     ,
     2    MDIME      ,SHAPE      )
C Calculate components of the equivalent nodal loads
          DO 114 IDOFN=1,NDOFN
            PGASH(IDOFN)=R0
            DGASH(IDOFN)=R0
            DO 110 INODEG=1,NNODEG
              II=NODAUX(NOPRS(INODEG))
              PGASH(IDOFN)=PGASH(IDOFN)+
     1                     PRESS(INODEG,IDOFN)*SHAPE(II)
              DGASH(IDOFN)=DGASH(IDOFN)+
     1                     ELCOD(IDOFN,INODEG)*DERIV(1,II)
  110       CONTINUE
  114     CONTINUE
          PXCOMP=DGASH(1)*PGASH(2)-DGASH(2)*PGASH(1)
          PYCOMP=DGASH(1)*PGASH(1)+DGASH(2)*PGASH(2)
C
          DVOLU=WEIGPB
          IF(NTYPE.EQ.1)THEN
C interpolate to find thickness at boundary gauss point (plane stress)
            THICK=R0
            DO 115 INODEG=1,NNODEG
              II=NODAUX(NOPRS(INODEG))
              INODE=NODCHK(II)
              THICK=THICK+THKN(INODE)*SHAPE(II)
  115       CONTINUE
            DVOLU=DVOLU*THICK
          ELSEIF(NTYPE.EQ.3)THEN
C interpolate to find radius at boundary gauss point (axisymmetric case)
            RADUS=R0
            DO 117 INODEG=1,NNODEG
              II=NODAUX(NOPRS(INODEG))
```

```
                RADUS=RADUS+SHAPE(II)*ELCOD(NAXIS,INODEG)
  117         CONTINUE
              DVOLU=DVOLU*TWOPI*RADUS
            ENDIF
C
C Add the equivalent nodal loads to the element force vector
            DO 130 INODEG=1,NNODEG
              INODE=NODCHK(INODEG)
              NGASH=(INODE-1)*NDOFN+1
              MGASH=(INODE-1)*NDOFN+2
              RLOAD(NGASH,IELEM)=RLOAD(NGASH,IELEM)+
     1                           SHAPE(INODEG)*PXCOMP*DVOLU
              RLOAD(MGASH,IELEM)=RLOAD(MGASH,IELEM)+
     1                           SHAPE(INODEG)*PYCOMP*DVOLU
  130       CONTINUE
  150     CONTINUE
  160   CONTINUE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE INTFOR( INCCUT )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C Arguments
      LOGICAL  INCCUT
C Local variables
      LOGICAL  IFFAIL
C***********************************************************************
C LOOPS OVER ALL ELEMENTS OF THE STRUCTURE TO COMPUTE ELEMENT INTERNAL
C FORCE VECTORS
C
C REFERENCE: Figures 5.2-3
C***********************************************************************
C Initialise increment cutting flag
      INCCUT=.FALSE.
C
C Begin loop over elements
C ========================
      DO 50 IELEM=1,NELEM
C
C Call element interface for internal force vector computation
C ------------------------------------------------------------
        CALL ELEIIF
     1(   IELEM      ,IFFAIL     )
C
        IF(IFFAIL)THEN
C Internal force calculation failed for current element: Break loop
C over elements and return to main program with increment cutting
C flag activated
          INCCUT=.TRUE.
          GOTO 999
        ENDIF
C
   50 CONTINUE
C Emergency exit
  999 CONTINUE
      RETURN
      END
```

```
      INTEGER FUNCTION INTNUM(CHRSTR)
      IMPLICIT NONE
      CHARACTER*(*) CHRSTR
      INTEGER I, IASCII, IEND, IPOWER, LEN, LENGTH, NUMBER
C***********************************************************************
C CONVERTS A NUMBER CONTAINED IN A CHARACTER STRING INTO AN INTEGER
C***********************************************************************
 1000 FORMAT(/15X,'ERROR: String of blank characters passed'/
     1        22X,'into integer conversion function INTMUN')
 1100 FORMAT(/15X,'ERROR: Invalid character in string ''',A,''' passed'/
     1        22X,'into integer conversion function INTMUN')
C
      LENGTH=LEN(CHRSTR)
      DO 10 I=LENGTH,1,-1
        IF(CHRSTR(I:I).NE.' ')THEN
          IEND=I
          GOTO 20
        ENDIF
   10 CONTINUE
      WRITE(*,1000)
      WRITE(16,1000)
      CALL PEXIT
   20 CONTINUE
      INTNUM=0
      IPOWER=0
      DO 30 I=IEND,1,-1
        IASCII=ICHAR(CHRSTR(I:I))
        IF(IASCII.GE.48.AND.IASCII.LE.57)THEN
          NUMBER=IASCII-48
          INTNUM=INTNUM+NUMBER*(10**IPOWER)
          IPOWER=IPOWER+1
        ELSEIF(CHRSTR(I:I).EQ.' ')THEN
          GOTO 40
        ELSEIF(CHRSTR(I:I).EQ.'-'.OR.CHRSTR(I:I).EQ.'+')THEN
          IF(I.NE.IEND)THEN
            IF(CHRSTR(I:I).EQ.'-')INTNUM=-INTNUM
            GOTO 40
          ELSE
            WRITE(*,1100)CHRSTR(1:IEND)
            WRITE(16,1100)CHRSTR(1:IEND)
            CALL PEXIT
          ENDIF
        ELSE
          WRITE(*,1100)CHRSTR(1:IEND)
          WRITE(16,1100)CHRSTR(1:IEND)
          CALL PEXIT
        ENDIF
   30 CONTINUE
   40 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INVF2
     1(   F          ,FINV       ,NTYPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    F(3,3)             ,FINV(3,3)
      DATA
     1   R0    ,R1   /
     2   0.0D0,1.0D0/
C***********************************************************************
C INVERTS DEFORMATION GRADIENT TENSORS FOR PLANE STRESS/STRAIN AND
C AXISYMMETRIC PROBLEMS
C***********************************************************************
      DETFPL=F(1,1)*F(2,2)-F(1,2)*F(2,1)
      IF(DETFPL.EQ.R0)CALL ERRPRT('EE0001')
      IF(NTYPE.EQ.3.AND.F(3,3).EQ.R0)CALL ERRPRT('EE0001')
C
      DETFIN=R1/DETFPL
      FINV(1,1)=F(2,2)*DETFIN
      FINV(2,2)=F(1,1)*DETFIN
      FINV(1,2)=-F(1,2)*DETFIN
      FINV(2,1)=-F(2,1)*DETFIN
      IF(NTYPE.EQ.2)THEN
        FINV(3,3)=R1
      ELSEIF(NTYPE.EQ.3)THEN
        FINV(3,3)=R1/F(3,3)
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE INVMT3
     1(   S          ,SINV       ,DETS       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    S(3,3)             ,SINV(3,3)
      DATA
     1   R0   ,R1   /
     2   0.0D0,1.0D0/
C***********************************************************************
C INVERT A REAL 3x3 MATRIX
C***********************************************************************
      DETS=S(1,1)*S(2,2)*S(3,3)+S(1,2)*S(2,3)*S(3,1)+
     1     S(1,3)*S(2,1)*S(3,2)-S(1,2)*S(2,1)*S(3,3)-
     2     S(1,1)*S(2,3)*S(3,2)-S(1,3)*S(2,2)*S(3,1)
      IF(DETS.EQ.R0)CALL ERRPRT('EE0011')
C
      DETSIN=R1/DETS
      SINV(1,1)=+DETSIN*(S(2,2)*S(3,3)-S(2,3)*S(3,2))
      SINV(2,1)=-DETSIN*(S(2,1)*S(3,3)-S(2,3)*S(3,1))
      SINV(3,1)=+DETSIN*(S(2,1)*S(3,2)-S(2,2)*S(3,1))
      SINV(1,2)=-DETSIN*(S(1,2)*S(3,3)-S(1,3)*S(3,2))
      SINV(2,2)=+DETSIN*(S(1,1)*S(3,3)-S(1,3)*S(3,1))
      SINV(3,2)=-DETSIN*(S(1,1)*S(3,2)-S(1,2)*S(3,1))
      SINV(1,3)=+DETSIN*(S(1,2)*S(2,3)-S(1,3)*S(2,2))
      SINV(2,3)=-DETSIN*(S(1,1)*S(2,3)-S(1,3)*S(2,1))
      SINV(3,3)=+DETSIN*(S(1,1)*S(2,2)-S(1,2)*S(2,1))
C
      RETURN
      END
```

```
      SUBROUTINE ISO2
     1(   FUNC       ,OUTOFP     ,X          ,Y          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MCOMP=4    ,NDIM=2     )
      LOGICAL OUTOFP ,REPEAT
      DIMENSION
     1    X(*)                   ,Y(*)
      DIMENSION
     1    EIGPRJ(MCOMP,NDIM)        ,EIGX(NDIM)                ,
     1    EIGY(NDIM)
C***********************************************************************
C COMPUTE THE TENSOR Y (STORED IN VECTOR FORM) AS AN ISOTROPIC
C FUNCTION OF THE TYPE:
C
C                      Y(X) = sum{ y(x_i) E_i }
C
C WHERE Y AND X ARE SYMMETRIC TENSORS, x_i AND E_i ARE, RESPECTIVELY
C THE EIGENVALUES AND EIGENPROJECTIONS OF X, AND y(.) IS A SCALAR
C FUNCTION. THIS ROUTINE IS RESTRICTED TO 2-D TENSORS WITH ONE
C POSSIBLE (TRANSVERSAL) OUT-OF-PLANE COMPONENT.
C
C REFERENCE: Section A.5
C***********************************************************************
C Performs the spectral decomposition of X
      CALL SPDEC2
     1(   EIGPRJ     ,EIGX       ,REPEAT     ,X          )
C Computes the in-plane eigenvalues of Y
      DO 10 IDIR=1,2
        EIGY(IDIR)=FUNC(EIGX(IDIR))
   10 CONTINUE
C Assembles in-plane component of Y (in vector form)
      CALL RVZERO(Y,3)
      DO 30 ICOMP=1,3
        DO 20 IDIR=1,2
          Y(ICOMP)=Y(ICOMP)+EIGY(IDIR)*EIGPRJ(ICOMP,IDIR)
   20   CONTINUE
   30 CONTINUE
C Out-of-plane component required
      IF(OUTOFP)Y(4)=FUNC(X(4))
C
      RETURN
      END
```

```
      SUBROUTINE IVZERO
     1(   IV         ,N          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION IV(N)
C***********************************************************************
C INITIALISES TO ZERO AN INTEGER ARRAY OF DIMENSION N
C***********************************************************************
      DO 10 I=1,N
        IV(I)=0
   10 CONTINUE
      RETURN
      END
```

```fortran
      SUBROUTINE JACOB(A,D,V,N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MJITER=50,NMAX=100)
      DIMENSION
     1    A(N,N)     ,D(N)      ,V(N,N)
      DIMENSION
     1    B(NMAX)    ,Z(NMAX)
      DATA R0   ,RP2  ,RP5  ,R1   ,R100   /
     1     0.0D0,0.2D0,0.5D0,1.0D0,100.0D0/
      DATA TOLER  /
     1     1.0D-12/
C***********************************************************************
C JACOBI ITERATIVE PROCEDURE FOR SPECTRAL DECOMPOSITION OF A
C N-DIMENSIONAL SYMMETRIC MATRIX
C
C REFERENCE: WH Press, SA Teukolsky, WT Vetting & BP Flannery. Numerical
C            recipes in FORTRAN: The art of scientific computing. 2nd
C            Edn., Cambridge University Press, 1992.
C***********************************************************************
      IF(N.GT.NMAX)THEN
        CALL ERRPRT('EI0025')
      ENDIF
      DO 20 IP=1,N
        DO 10 IQ=1,N
          V(IP,IQ)=R0
   10   CONTINUE
        V(IP,IP)=R1
   20 CONTINUE
      DO 30 IP=1,N
        B(IP)=A(IP,IP)
        D(IP)=B(IP)
        Z(IP)=R0
   30 CONTINUE
      DO 130 I=1,MJITER
        SM=R0
        DO 50 IP=1,N-1
          DO 40 IQ=IP+1,N
            SM=SM+ABS(A(IP,IQ))
   40     CONTINUE
   50   CONTINUE
        IF(SM.LT.TOLER)GOTO 999
        IF(I.LT.4)THEN
          TRESH=RP2*SM/DBLE(N**2)
        ELSE
          TRESH=R0
        ENDIF
        DO 110 IP=1,N-1
          DO 100 IQ=IP+1,N
            G=R100*ABS(A(IP,IQ))
            IF((I.GT.4).AND.(ABS(D(IP))+G.EQ.ABS(D(IP)))
     1         .AND.(ABS(D(IQ))+G.EQ.ABS(D(IQ))))THEN
              A(IP,IQ)=R0
            ELSE IF(ABS(A(IP,IQ)).GT.TRESH)THEN
              H=D(IQ)-D(IP)
              IF(ABS(H)+G.EQ.ABS(H))THEN
                T=A(IP,IQ)/H
              ELSE
                THETA=RP5*H/A(IP,IQ)
                T=R1/(ABS(THETA)+SQRT(R1+THETA**2))
                IF(THETA.LT.R0)T=-T
              ENDIF
              C=R1/SQRT(R1+T**2)
              S=T*C
              TAU=S/(R1+C)
              H=T*A(IP,IQ)
              Z(IP)=Z(IP)-H
              Z(IQ)=Z(IQ)+H
              D(IP)=D(IP)-H
              D(IQ)=D(IQ)+H
              A(IP,IQ)=R0
              DO 60 J=1,IP-1
                G=A(J,IP)
                H=A(J,IQ)
                A(J,IP)=G-S*(H+G*TAU)
                A(J,IQ)=H+S*(G-H*TAU)
   60         CONTINUE
              DO 70 J=IP+1,IQ-1
                G=A(IP,J)
                H=A(J,IQ)
                A(IP,J)=G-S*(H+G*TAU)
                A(J,IQ)=H+S*(G-H*TAU)
   70         CONTINUE
              DO 80 J=IQ+1,N
                G=A(IP,J)
                H=A(IQ,J)
                A(IP,J)=G-S*(H+G*TAU)
                A(IQ,J)=H+S*(G-H*TAU)
   80         CONTINUE
              DO 90 J=1,N
                G=V(J,IP)
                H=V(J,IQ)
                V(J,IP)=G-S*(H+G*TAU)
                V(J,IQ)=H+S*(G-H*TAU)
   90         CONTINUE
            ENDIF
  100     CONTINUE
  110   CONTINUE
        DO 120 IP=1,N
```

```
        B(IP)=B(IP)+Z(IP)
        D(IP)=B(IP)
        Z(IP)=R0
120   CONTINUE
130 CONTINUE
    CALL ERRPRT('EE0005')
999 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    MDIME      ,NDIME      ,NNODE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    CARTD(MDIME,*)     ,DERIV(MDIME,*)     ,ELCOD(MDIME,*)
      DIMENSION
     1    XJACI(2,2)         ,XJACM(2,2)
      DATA R0   /
     1     0.0D0/
C***********************************************************************
C EVALUATES THE JACOBIAN MATRIX, ITS DETERMINANT AND THE CARTESIAN
C DERIVATIVES OF THE SHAPE FUNCTIONS OF 2-D ISOPARAMETRIC ELEMENTS
C
C REFERENCE: Section 4.1.2
C            Expression (4.33)
C***********************************************************************
 1000 FORMAT(//'  Warning from subroutine JACOB2:'//
     1 10X,'Negative jacobian determinant ',G12.4,' Element number ',I5)
 1010 FORMAT(//'  Warning from subroutine JACOB2:'//
     1 10X,'Zero jacobian determinant ',12X,' Element number ',I5/
     2 10X,'Jacobian matrix not inverted and cartesian derivatives ',/,
     3 10X,'of shape functions not computed')
C
C Evaluate jacobian matrix XJACM
C -----------------------------
C
      DO 30 IDIME=1,NDIME
        DO 20 JDIME=1,NDIME
          XJACM(IDIME,JDIME)=R0
          DO 10 INODE=1,NNODE
            XJACM(IDIME,JDIME)=XJACM(IDIME,JDIME)+DERIV(IDIME,INODE)*
     1                         ELCOD(JDIME,INODE)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C Determinant of jacobian matrix
      DETJAC=XJACM(1,1)*XJACM(2,2)-XJACM(1,2)*XJACM(2,1)
      IF(DETJAC.LT.R0)THEN
        WRITE(*,1000)DETJAC,IELEM
        WRITE(16,1000)DETJAC,IELEM
      ELSEIF(DETJAC.EQ.R0)THEN
        WRITE(*,1010)IELEM
        WRITE(16,1010)IELEM
        GOTO 999
      ENDIF
C Inverse of jacobian matrix
      XJACI(1,1)=XJACM(2,2)/DETJAC
      XJACI(2,2)=XJACM(1,1)/DETJAC
      XJACI(1,2)=-XJACM(1,2)/DETJAC
      XJACI(2,1)=-XJACM(2,1)/DETJAC
C
C Evaluate cartesian derivatives of shape functions
C -------------------------------------------------
C
      DO 60 IDIME=1,NDIME
        DO 50 INODE=1,NNODE
          CARTD(IDIME,INODE)=R0
          DO 40 JDIME=1,NDIME
            CARTD(IDIME,INODE)=CARTD(IDIME,INODE)+XJACI(IDIME,JDIME)*
     1                         DERIV(JDIME,INODE)
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LEFTCG
     1(   BN         ,BNP1       ,FINCR      ,NTYPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    BN(*)              ,BNP1(*)            ,FINCR(3,3)
      DIMENSION
     1    AUXM(2,2)          ,BNMTX(2,2)         ,BNP1M(2,2)
C***********************************************************************
C COMPUTES THE LEFT CAUCHY-GREEN STRAIN TENSOR ACCORDING TO THE
C FORMULA:
C
C                                                T
C                     B        :=   F     B   F
C                      n+1           incr  n   incr
C
C REFERENCE: Box 13.1. The formula used here is equivalent to that of
C                       item (i) of Box 13.1.
C***********************************************************************
C Convert previously converged left Cauchy-Green strain tensor from
C vector form to matrix form
      BNMTX(1,1)=BN(1)
      BNMTX(2,1)=BN(3)
      BNMTX(1,2)=BN(3)
      BNMTX(2,2)=BN(2)
C
C In-plane components of the left Cauchy-Green tensor
C
      CALL RVZERO(AUXM,4)
      DO 30 I=1,2
        DO 20 J=1,2
          DO 10 K=1,2
            AUXM(I,J)=AUXM(I,J)+FINCR(I,K)*BNMTX(K,J)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      CALL RVZERO(BNP1M,4)
      DO 60 I=1,2
        DO 50 J=1,2
          DO 40 K=1,2
            BNP1M(I,J)=BNP1M(I,J)+AUXM(I,K)*FINCR(J,K)
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
C
C Store B    in vector form
C        n+1
C
      BNP1(1)=BNP1M(1,1)
      BNP1(2)=BNP1M(2,2)
      BNP1(3)=BNP1M(1,2)
C out-of-plane component
      IF(NTYPE.EQ.2)THEN
        BNP1(4)=BN(4)
      ELSEIF(NTYPE.EQ.3)THEN
        BNP1(4)=BN(4)*FINCR(3,3)*FINCR(3,3)
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE LENGTH(DLENG ,DLENM ,ITACT ,ITDES )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C***********************************************************************
C ADJUSTS STEP LENGTH ACCORDING TO THE DESIRED NUMBER OF ITERATIONS AND
C THE NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE IN THE PREVIOUS
C LOAD STEP (USED FOR ARC-LENGTH METHOD ONLY)
C
C REFERENCE: Expression (5.3)
C***********************************************************************
      DLENG=DLENG*DBLE(ITDES)/DBLE(ITACT)
      DLENG=MIN(DLENG,DLENM)
      RETURN
      END
```

```
      SUBROUTINE LISTRA
     1(   BMATX      ,ELDISP     ,MDOFN      ,MBDIM      ,NDOFN      ,
     2    NNODE      ,NTYPE      ,STRAN      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    BMATX(MBDIM,*)     ,ELDISP(MDOFN,*)    ,STRAN(*)
C***********************************************************************
C COMPUTES THE SYMMETRIC GRADIENT (LINEAR STRAIN MEASURE) ASSOCIATED
C WITH THE ELEMENT DISPLACEMENT 'ELDISP' IN 2-D: PLANE STRAIN, PLANE
C STRESS AND AXISYMMETRIC PROBLEMS
C
C REFERENCE: Expression (4.53)
C***********************************************************************
      IF(NTYPE.EQ.1)THEN
        NSTRE=3
        NBDIM=3
      ELSEIF(NTYPE.EQ.2)THEN
        NSTRE=4
        NBDIM=3
      ELSEIF(NTYPE.EQ.3)THEN
        NSTRE=4
        NBDIM=4
      ELSE
        CALL ERRPRT('EI0023')
      ENDIF
C
      CALL RVZERO(STRAN,NSTRE)
      DO 30 ISTRE=1,NBDIM
        IEVAB=0
        DO 20 INODE=1,NNODE
          DO 10 IDOFN=1,NDOFN
            IEVAB=IEVAB+1
            STRAN(ISTRE)=STRAN(ISTRE)+
     1                   BMATX(ISTRE,IEVAB)*ELDISP(IDOFN,INODE)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LOGSTR
     1(   B          ,E          ,NTYPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL  DLGD2
      LOGICAL   OUTOFP
      DIMENSION
     1    B(*)                ,E(*)
      DATA  R2   /2.0D0/
C***********************************************************************
C COMPUTES THE LOGARITHMIC STRAIN TENSOR:
C
C                       E :=  1/2 ln[ B ]
C
C REFERENCE: Box 14.3, item (ii)
C***********************************************************************
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        OUTOFP=.TRUE.
      ELSEIF(NTYPE.EQ.1)THEN
        OUTOFP=.FALSE.
      ELSE
        CALL ERRPRT('EI0022')
      ENDIF
C
C Use isotropic tensor function to compute the logarithmic (physical)
C strain components
C
      CALL ISO2
     1(   DLGD2      ,OUTOFP     ,B          ,E          )
C
C Convert physical components into engineering strain components
C
      E(3)=R2*E(3)
C
      RETURN
      END
```

```
      SUBROUTINE MATICT
     1(   DETF       ,KUNLD      ,MBDIM      ,MGDIM      ,
     2    NLARGE     ,NTYPE      ,
     3    AMATX      ,DMATX      ,EINCR      ,FINCR      ,IPROPS     ,
     4    LALGVA     ,RALGVA     ,RPROPS     ,RSTAVA     ,RSTAV2     ,
     5    STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../MATERIAL.INC'
C
      PARAMETER( MSTRA=4 )
C Arguments
      LOGICAL  LALGVA
      DIMENSION
     1    AMATX(MGDIM,MGDIM) ,DMATX(MBDIM,MBDIM) ,EINCR(MBDIM)       ,
     2    FINCR(3,3)         ,IPROPS(*)          ,LALGVA(*)          ,
     3    RALGVA(*)          ,RPROPS(*)          ,RSTAVA(*)          ,
     4    RSTAV2(*)          ,STRES(*)
C Local arrays and variables
      LOGICAL EPFLAG ,IFPLAS
      DIMENSION
     1    BETRL(MSTRA)       ,STRAT(MSTRA)       ,STRESK(4)
C***********************************************************************
C MATERIAL INTERFACE FOR CONSISTENT TANGENT COMPUTATION ROUTINE CALLS:
C ACCORDING TO THE MATERIAL TYPE, CALLS MATERIAL-SPECIFIC TANGENT
C COMPUTATION ROUTINE
C
C REFERENCE: Figure 5.5
C            Sections 5.7.4, 5.7.6
C***********************************************************************
C Start by identifying the material type and class
      MATTYP=IPROPS(1)
      MATCLS=IPROPS(2)
C
C Then call material class/type-specific routines
C
      IF(MATCLS.EQ.HYPEPL)THEN
C
C Elastic/elasto-plastic materials with logarithmic finite strain
C extension
C ================================================================
C
C Retrieve current stress
        DO 50 ISTRE=1,4
          STRESK(ISTRE)=STRES(ISTRE)
   50   CONTINUE
        IF(NLARGE.EQ.1)THEN
C Large strains: compute last elastic trial LOGARITHMIC strain
C -------------------------------------------------------------
C elastic trial left Cauchy-Green tensor
          CALL BETRIA
     1(   BETRL      ,RSTAV2     ,FINCR      ,NTYPE      )
C elastic trial eulerian logarithmic strain
          CALL LOGSTR
     1(   BETRL      ,STRAT      ,NTYPE      )
          DETFT=DETF
          IF(NTYPE.EQ.1)THEN
C compute total deformation gradient (including thickness strain
C contribution) for plane stress, according to material model
            IF(MATTYP.EQ.ELASTC)THEN
C... Elastic (Hencky material in large strain)
              CALL TUEL
     1(   DETFT      ,RSTAVA     ,DUMMY      ,0          )
            ELSEIF(MATTYP.EQ.VMISES)THEN
C... von Mises elasto-plastic
              CALL TUVM
     1(   DETFT      ,RSTAVA     ,DUMMY      ,0          )
            ELSE
C... Error: Material type not recognised or not implemented for finite
C    strains under plane stress
              CALL ERRPRT('EI0059')
            ENDIF
          ENDIF
C retrieve current KIRCHHOFF stress in large strains
          CALL RVSCAL(STRESK,4,DETFT)
        ELSE
C Small strains: compute last elastic trial INFINITESIMAL strain
C ---------------------------------------------------------------
          DO 60 ISTRE=1,4
            STRAT(ISTRE)=RSTAV2(ISTRE)+EINCR(ISTRE)
   60     CONTINUE
        ENDIF
C Set plastic elasto-plastic tangent flag
C ---------------------------------------
        IF(MATTYP.NE.ELASTC)THEN
          IFPLAS=LALGVA(1)
          IF((.NOT.IFPLAS).OR.KUNLD.EQ.1)THEN
            EPFLAG=.FALSE.
          ELSE
            EPFLAG=.TRUE.
          ENDIF
        ENDIF
C Call material type-specific routines
C ------------------------------------
        IF(MATTYP.EQ.ELASTC)THEN
C Elastic
          CALL CTEL
     1(   DMATX      ,NTYPE      ,RPROPS     )
```

```
        ELSEIF(MATTYP.EQ.TRESCA)THEN
C Tresca
          IF(NTYPE.EQ.1)THEN
            CALL CTTRPN
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRESK     )
          ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
            CALL CTTR
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRAT      ,STRESK     )
          ENDIF
        ELSEIF(MATTYP.EQ.VMISES)THEN
C von Mises
          IF(NTYPE.EQ.1)THEN
            CALL CTVMPS
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE     ,
     2    RPROPS     ,RSTAVA     ,STRESK     )
          ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
            CALL CTVM
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE     ,
     2    RPROPS     ,RSTAVA     ,STRESK     )
          ENDIF
        ELSEIF(MATTYP.EQ.MOHCOU)THEN
C Mohr-Coulomb
          CALL CTMC
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRAT      ,STRESK     )
        ELSEIF(MATTYP.EQ.DRUPRA)THEN
C Drucker-Prager
          IF(NTYPE.EQ.1)THEN
            CALL CTDPPN
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,
     2    NTYPE      ,RPROPS     ,RSTAVA     ,STRAT      )
          ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
            CALL CTDP
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,
     2    NTYPE      ,RPROPS     ,RSTAVA     ,STRAT      )
          ENDIF
        ELSEIF(MATTYP.EQ.LEMDAM)THEN
C Lemaitre's ductile damage model
          CALL CTDAMA
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRESK     )
        ELSEIF(MATTYP.EQ.DAMELA)THEN
C Isotropically damaged isotropic elastic material with crack closure
C effects
          CALL CTDMEL
     1(   DMATX      ,NTYPE      ,RPROPS     ,STRAT      ,STRESK     )
        ELSE
C Error: Material type not recognised
          CALL ERRPRT('EI0044')
        ENDIF
C Perform extra kinematical operations required by materials of this
C class at large strains for computation of the spatial modulus 'a'
C ---------------------------------------------------------------------
        IF(NLARGE.EQ.1)THEN
          CALL CSTEP2
     1(   AMATX      ,BETRL      ,DMATX      ,STRES     ,DETFT      ,
     2    NTYPE      )
        ENDIF
      ELSEIF(MATCLS.EQ.SINCRY)THEN
C
C Single crystal anisotropic elasto-plastic models
C ================================================
C
C Set plastic loading/unloading flag
        IFPLAS=LALGVA(1)
        IF((.NOT.IFPLAS).OR.KUNLD.EQ.1)THEN
          EPFLAG=.FALSE.
        ELSE
          EPFLAG=.TRUE.
        ENDIF
C Call material type-specific routines
C ------------------------------------
        IF(MATTYP.EQ.PDSCRY)THEN
C Planar double slip single crystal
          CALL CSTPDS
     1(   AMATX      ,RALGVA     ,EPFLAG     ,FINCR      ,IPROPS     ,
     2    LALGVA     ,NTYPE      ,RPROPS     ,RSTAVA     ,RSTAV2     ,
     3    STRES      )
        ELSE
C... Error: Material type not recognised
          CALL ERRPRT('EI0044')
        ENDIF
      ELSEIF(MATCLS.EQ.HYPER)THEN
C
C Generic isotropic finite hyperelasticity models
C ===============================================
C
C Call material type-specific routines
C ------------------------------------
        IF(MATTYP.EQ.OGDEN)THEN
C Ogden model
          CALL CSTOGD
     1(   AMATX      ,RSTAVA     ,IPROPS     ,NTYPE      ,RPROPS     ,
     2    STRES      )
        ELSE
C Error: Material type not recognised
```

```
          CALL ERRPRT('EI0044')
        ENDIF
      ELSEIF(MATCLS.EQ.PLASTC)THEN
C
C Elasto-plastic materials with small strain implementation only
C ==============================================================
C
        IFPLAS=LALGVA(1)
        IF((.NOT.IFPLAS).OR.KUNLD.EQ.1)THEN
          EPFLAG=.FALSE.
        ELSE
          EPFLAG=.TRUE.
        ENDIF
        IF(MATTYP.EQ.VMMIXD)THEN
C von Mises with mixed isotropic/kinematic hardening
          CALL CTVMMX
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,RSTAV2     ,STRES      )
        ELSE
C Error: Material type not recognised
          CALL ERRPRT('EI0044')
        ENDIF
      ELSE
C Error: Material class not recognised
        CALL ERRPRT('EI0043')
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE MATIOR
     1(   NTYPE      ,IPROPS     ,RALGVA     ,RPROPS     ,RSTAVA     ,
     2    STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE '../MATERIAL.INC'
C Arguments
      DIMENSION
     1    IPROPS(*)          ,RALGVA(*)          ,RPROPS(*)          ,
     2    RSTAVA(*)          ,STRES(*)
C***********************************************************************
C MATERIAL INTERFACE FOR OUTPUT RESULT ROUTINE CALLS:
C ACCORDING TO THE MATERIAL TYPE, CALLS MATERIAL-SPECIFIC OUTPUT
C ROUTINE
C
C REFERENCE: Sections 5.7.5-6
C***********************************************************************
C First identify material type
C ---------------------------
      MATTYP=IPROPS(1)
C Then call corresponding routine to output material-specific results
C -------------------------------------------------------------------
      IF(MATTYP.EQ.ELASTC)THEN
C Elastic
        CALL OREL(16      ,NTYPE   ,STRES   )
      ELSEIF(MATTYP.EQ.TRESCA)THEN
C Tresca
        CALL ORTR(RALGVA  ,16      ,NTYPE   ,RSTAVA  ,STRES   )
      ELSEIF(MATTYP.EQ.VMISES)THEN
C von Mises
        CALL ORVM(RALGVA  ,16      ,NTYPE   ,RSTAVA  ,STRES   )
      ELSEIF(MATTYP.EQ.MOHCOU)THEN
C Mohr-Coulomb
        CALL ORMC(RALGVA  ,16      ,NTYPE   ,RSTAVA  ,STRES   )
      ELSEIF(MATTYP.EQ.DRUPRA)THEN
C Drucker-Prager
        CALL ORDP(RALGVA  ,16      ,NTYPE   ,RSTAVA  ,STRES   )
      ELSEIF(MATTYP.EQ.LEMDAM)THEN
C Lemaitre's ductile damage model
        CALL ORDAMA(RALGVA  ,16    ,NTYPE   ,RSTAVA  ,STRES   )
      ELSEIF(MATTYP.EQ.DAMELA)THEN
C Isotropically damaged isotropic elastic material with crack closure
C effects
        CALL ORDMEL(16    ,NTYPE   ,STRES   )
      ELSEIF(MATTYP.EQ.OGDEN)THEN
C Ogden hyperelastic
        CALL OROGD(16     ,NTYPE   ,STRES   )
      ELSEIF(MATTYP.EQ.PDSCRY)THEN
C Planar double-slip single crystal plasticity model
        CALL ORPDSC(RALGVA,16      ,NTYPE   ,RPROPS  ,RSTAVA  ,STRES   )
      ELSEIF(MATTYP.EQ.VMMIXD)THEN
C von Mises with mixed isotropic/kinematic hardening (infinitesimal
C only)
        CALL ORVMMX(RALGVA,16      ,NTYPE   ,RSTAVA  ,STRES   )
      ELSE
C Error: Material type not recognised
        CALL ERRPRT('EI0045')
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE MATIRD
     1(   MATNAM      ,NLARGE     ,NTYPE      ,UNSAUX     ,IPROPS     ,
     2    RPROPS      )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MATERIAL.INC'
C Arguments
      LOGICAL      UNSAUX
      CHARACTER*80 MATNAM
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
C***********************************************************************
C MATERIAL INTERFACE FOR READING MATERIAL-SPECIFIC INPUT DATA
C
C REFERENCE: Sections 5.7.1, 5.7.6
C***********************************************************************
C According to MATNAM, call the appropriate routine to read the
C the material-specific data from the input file and store it
C ---------------------------------------------------------------
      IF(MATNAM.EQ.'ELASTIC')THEN
C Elastic
        MATTYP=ELASTC
        MATCLS=HYPEPL
        CALL RDEL
     1(   MRPROP ,MRSTAV ,RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'TRESCA')THEN
C Tresca elasto-plastic
        MATTYP=TRESCA
        MATCLS=HYPEPL
        CALL RDTR
     1(   IPROPS ,MIPROP ,MLALGV ,MRALGV ,MRPROP ,MRSTAV ,
     2    NLARGE ,NTYPE  ,RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'VON_MISES')THEN
C von Mises elasto-plastic
        MATTYP=VMISES
        MATCLS=HYPEPL
        CALL RDVM
     1(   IPROPS ,MIPROP ,MLALGV ,MRPROP ,MRSTAV ,
     2    RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'MOHR_COULOMB')THEN
C Mohr-Coulomb elasto-plastic
        MATTYP=MOHCOU
        MATCLS=HYPEPL
        CALL RDMC
     1(   IPROPS ,MIPROP ,MLALGV ,MRALGV ,MRPROP ,MRSTAV ,
     2    RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'DRUCKER_PRAGER')THEN
C Drucker-Prager elasto-plastic
        MATTYP=DRUPRA
        MATCLS=HYPEPL
        CALL RDDP
     1(   IPROPS  ,MIPROP ,MLALGV ,MRALGV ,MRPROP ,MRSTAV ,
     2    RPROPS  ,UNSAUX)
      ELSEIF(MATNAM.EQ.'LEMAITRE_DAMAGE')THEN
C Lemaitre's ductile damage model
        MATTYP=LEMDAM
        MATCLS=HYPEPL
        CALL RDDAMA
     1(   IPROPS ,MIPROP ,MLALGV ,MRPROP ,MRSTAV ,NTYPE  ,
     2    RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'DAMAGED_ELASTIC')THEN
C Isotropically damaged isotropic elastic material with crack closure
C effects
        MATTYP=DAMELA
        MATCLS=HYPEPL
        CALL RDDMEL
     1(   NTYPE ,MRPROP ,MRSTAV ,RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'OGDEN')THEN
C Ogden hyperelastic
        MATTYP=OGDEN
        MATCLS=HYPER
        CALL RDOGD
     1(   IPROPS ,MIPROP ,MRPROP ,MRSTAV ,RPROPS ,UNSAUX)
      ELSEIF(MATNAM.EQ.'PLANAR_DOUBLE_SLIP_SINGLE_CRYSTAL')THEN
C Planar double-slip single crystal elasto-plastic model
        MATTYP=PDSCRY
        MATCLS=SINCRY
        CALL RDPDSC
     1(   IPROPS ,MIPROP ,MLALGV ,MRALGV ,MRPROP , MRSTAV ,
     2    NLARGE ,NTYPE  ,RPROPS ,UNSAUX )
      ELSEIF(MATNAM.EQ.'VON_MISES_MIXED')THEN
C von Mises elasto-plastic with mixed hardening (small strains only)
        MATTYP=VMMIXD
        MATCLS=PLASTC
        CALL RDVMMX
     1(   IPROPS      ,MIPROP     ,MLALGV     ,MRPROP     ,MRSTAV     ,
     2    NLARGE      ,NTYPE      ,RPROPS     ,UNSAUX     )
      ELSE
        CALL ERRPRT('ED0015')
      ENDIF
C Store material type and class flags in IPROPS
C -----------------------------------------------
      IPROPS(1)=MATTYP
      IPROPS(2)=MATCLS
C
      RETURN
      END
```

```
      SUBROUTINE MATISU
     1(   DETF       ,NLARGE     ,NTYPE      ,SUFAIL     ,THKGP      ,
     3    EINCR      ,FINCR      ,IPROPS     ,LALGVA     ,RALGVA     ,
     4    RPROPS     ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../MATERIAL.INC'
C
      PARAMETER( MSTRA=4 )
C Arguments
      LOGICAL
     1    SUFAIL             ,LALGVA
      DIMENSION
     1    EINCR(*)           ,FINCR(3,3)         ,IPROPS(*)          ,
     2    LALGVA(*)          ,RALGVA(*)          ,RPROPS(*)          ,
     3    RSTAVA(*)          ,STRES(*)
C Local arrays
      DIMENSION
     1    B(MSTRA)           ,BETRL(MSTRA)       ,STRAT(MSTRA)
C Local numerical constants
      DATA
     1    R1   /
     2    1.0D0/
C***********************************************************************
C MATERIAL INTERFACE FOR STATE UPDATE ROUTINE CALLS:
C ACCORDING TO THE MATERIAL TYPE, CALLS MATERIAL-SPECIFIC STATE UPDATE
C ROUTINE TO UPDATE STRESS AND OTHER STATE VARIABLES
C
C REFERENCE: Figure 5.4
C            Sections 5.7.2, 5.7.6
C***********************************************************************
C Set up number of stress components
      IF(NTYPE.EQ.1)THEN
        NSTRE=3
      ELSEIF(NTYPE.EQ.2)THEN
        NSTRE=4
      ELSEIF(NTYPE.EQ.3)THEN
        NSTRE=4
      ELSE
        CALL ERRPRT('EI0040')
      ENDIF
C Identify material type and class
      MATTYP=IPROPS(1)
      MATCLS=IPROPS(2)
C
C Then call material class/type-specific routines
C
      IF(MATCLS.EQ.HYPEPL)THEN
C
C Isotropic elastic/elasto-plastic materials with logarithmic finite
C strain extension
C ===================================================================
C
C Compute elastic trial strains. Note that for the purely elastic models
C the elastic trial strain equals the total strain
C -----------------------------------------------------------------------
        IF(NLARGE.EQ.0)THEN
C Small strains: compute elastic trial INFINITESIMAL strain
          DO 10 ISTRE=1,NSTRE
            STRAT(ISTRE)=RSTAVA(ISTRE)+EINCR(ISTRE)
   10     CONTINUE
        ELSEIF(NLARGE.EQ.1)THEN
C Large strains: compute elastic trial LOGARITHMIC strain
C... elastic trial left Cauchy-Green tensor
          CALL BETRIA
     1(   BETRL      ,RSTAVA     ,FINCR       ,NTYPE      )
C... elastic trial eulerian logarithmic strain
          CALL LOGSTR
     1(   BETRL      ,STRAT      ,NTYPE       )
        ENDIF
C Apply small strain material type-specific state updating procedure
C -------------------------------------------------------------------
        IF(MATTYP.EQ.ELASTC)THEN
C Linear elastic (Hencky material in large strains)
          CALL SUEL
     1(   NTYPE      ,RPROPS     ,RSTAVA     ,STRAT      ,STRES      )
C...set elasto-plastic flag and state update failure flag
          LALGVA(1)=.FALSE.
          LALGVA(2)=.FALSE.
        ELSEIF(MATTYP.EQ.TRESCA)THEN
C Tresca elasto-plastic
          IF(NTYPE.EQ.1)THEN
            CALL SUTRPN
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
          ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
            CALL SUTR
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
          ENDIF
        ELSEIF(MATTYP.EQ.VMISES)THEN
C von Mises elasto-plastic
          IF(NTYPE.EQ.1)THEN
            CALL SUVMPS
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
          ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
            CALL SUVM
```

```
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
          ENDIF
        ELSEIF(MATTYP.EQ.MOHCOU)THEN
C Mohr-Coulomb elasto-plastic
          CALL SUMC
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
        ELSEIF(MATTYP.EQ.DRUPRA)THEN
C Drucker-Prager elasto-plastic
          IF(NTYPE.EQ.1)THEN
            CALL SUDPPN
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT       ,STRES     )
          ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
            CALL SUDP
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
          ENDIF
        ELSEIF(MATTYP.EQ.LEMDAM)THEN
C Lemaitre's ductile damage elasto-plastic model
          CALL SUDAMA
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
        ELSEIF(MATTYP.EQ.DAMELA)THEN
C Isotropically damaged isotropic elastic material with crack closure
C effects
          CALL SUDMEL
     1(   NTYPE      ,RPROPS     ,RSTAVA     ,STRAT      ,STRES      ,
     2    LALGVA(2)  )
C...set elasto-plastic flag to false
          LALGVA(1)=.FALSE.
        ELSE
C Error: Material type not recognised
          CALL ERRPRT('EI0042')
        ENDIF
C Exit routine in case of failure of the state update procedure
        SUFAIL=LALGVA(2)
        IF(SUFAIL)GOTO 999
        IF(NLARGE.EQ.1)THEN
C Perform extra updating operations required by this class of
C elasto-pastic materials at large strains only
C -------------------------------------------------------------
          DETFT=DETF
          IF(NTYPE.EQ.1)THEN
C Plane stress: update Gauss point thickness according to material
C model. Also update the total deformation gradient (taking the
C thickness strain into account)
            IF(MATTYP.EQ.ELASTC)THEN
C... Elastic (Hencky material in large strains)
              CALL TUEL
     1(   DETFT      ,RSTAVA     ,THKGP      ,1          )
            ELSEIF(MATTYP.EQ.VMISES)THEN
C... von Mises elasto-plastic
              CALL TUVM
     1(   DETFT      ,RSTAVA     ,THKGP      ,1          )
            ELSE
C... Error: Material type not recognised or not implemented for finite
C    strains under plane stress
              CALL ERRPRT('EI0058')
            ENDIF
          ENDIF
C Transform Kirchhoff into Cauchy stress
          DETFIN=R1/DETFT
          CALL RVSCAL(STRES,NSTRE,DETFIN)
        ENDIF
C
      ELSEIF(MATCLS.EQ.SINCRY)THEN
C
C Single crystal anisotropic finite elasto-plastic models
C =======================================================
C
        IF(MATTYP.EQ.PDSCRY)THEN
C Planar double slip single crystal
          CALL SUPDSC
     1(   RALGVA     ,FINCR      ,IPROPS     ,LALGVA     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRES      )
        ELSE
C Error: Material type not recognised
          CALL ERRPRT('EI0042')
        ENDIF
        SUFAIL=LALGVA(2)
        IF(SUFAIL)GOTO 999
      ELSEIF(MATCLS.EQ.HYPER)THEN
C
C Generic isotropic finite hyperelasticity models
C ===============================================
C
C First compute current Left Cauchy-Green strain tensor, B
        CALL LEFTCG
     1(   RSTAVA     ,B          ,FINCR      ,NTYPE      )
C Then call the material type-specific state update procedure
        IF(MATTYP.EQ.OGDEN)THEN
C Ogden model
          CALL SUOGD
     1(   B          ,IPROPS     ,NTYPE      ,RPROPS     ,RSTAVA     ,
     2    STRES      ,THKGP      )
          SUFAIL=.FALSE.
```

```
        ELSE
C Error: Material type not recognised
          CALL ERRPRT('EI0042')
        ENDIF
      ELSEIF(MATCLS.EQ.PLASTC)THEN
C
C Elasto-plastic materials with small strain implementation only
C ===============================================================
C
C compute elastic trial INFINITESIMAL strain
        DO 20 ISTRE=1,NSTRE
          STRAT(ISTRE)=RSTAVA(ISTRE)+EINCR(ISTRE)
   20   CONTINUE
        IF(MATTYP.EQ.VMMIXD)THEN
C von Mises with mixed isotropic/kinematic hardening
          CALL SUVMMX
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
        ELSE
C Error: Material type not recognised
          CALL ERRPRT('EI0042')
        ENDIF
        SUFAIL=LALGVA(2)
        IF(SUFAIL)GOTO 999
      ELSE
C Error: Material class not recognised
        CALL ERRPRT('EI0041')
      ENDIF
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MATISW
     1(   MODE       ,NLARGE     ,NTYPE      ,IPROPS     ,LALGVC     ,
     2    LALGVL     ,RALGVC     ,RALGVL     ,RPROPS     ,RSTAVC     ,
     3    RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE '../MATERIAL.INC'
C Arguments
      LOGICAL
     1     LALGVC           ,LALGVL
      DIMENSION
     1    IPROPS(*)          ,LALGVC(*)          ,LALGVL(*)          ,
     2    RALGVC(*)          ,RALGVL(*)          ,RPROPS(*)          ,
     3    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     4    STRESL(*)
C***********************************************************************
C MATERIAL INTERFACE FOR INITIALISATION/SWITCHING ROUTINE CALLS:
C ACCORDING TO THE MATERIAL TYPE, CALLS MATERIAL-SPECIFIC ROUTINE TO
C INITIALISE/SWITCH GAUSS POINT STATE AND ALGORITHMIC VARIABLES
C
C REFERENCE: Sections 5.7.3, 5.7.6
C***********************************************************************
C First identify material type and class
C --------------------------------------
      MATTYP=IPROPS(1)
C
C Then call material type-specific routines
C -----------------------------------------
      IF(MATTYP.EQ.ELASTC)THEN
C Elastic (Hencky material in large strains)
        CALL SWEL
     1(   MODE       ,NTYPE      ,RSTAVC     ,RSTAVL     ,STRESC     ,
     2    STRESL     )
      ELSEIF(MATTYP.EQ.TRESCA)THEN
C Tresca elasto-plastic
        CALL SWTR
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      ELSEIF(MATTYP.EQ.VMISES)THEN
C von Mises elasto-plastic
        CALL SWVM
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      ELSEIF(MATTYP.EQ.MOHCOU)THEN
C Mohr-Coulomb elasto-plastic
        CALL SWMC
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      ELSEIF(MATTYP.EQ.DRUPRA)THEN
C Drucker-Prager elasto-plastic
        CALL SWDP
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      ELSEIF(MATTYP.EQ.LEMDAM)THEN
C Lemaitre's ductile damage elasto-plastic model
        CALL SWDAMA
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      ELSEIF(MATTYP.EQ.DAMELA)THEN
C Isotropically damaged isotropic elastic material with crack closure
C effects
        CALL SWDMEL
     1(   MODE       ,NTYPE      ,RSTAVC     ,RSTAVL     ,STRESC     ,
     2    STRESL     )
      ELSEIF(MATTYP.EQ.PDSCRY)THEN
C Planar double-slip single crystal
        CALL SWPDSC
     1(   MODE       ,LALGVC     ,LALGVL     ,RALGVC     ,RSTAVC     ,
     2    RSTAVL     ,STRESC     ,STRESL     )
      ELSEIF(MATTYP.EQ.OGDEN)THEN
C Ogden hyperelasticity model
        CALL SWOGD
     1(   MODE       ,NTYPE      ,RSTAVC     ,RSTAVL     ,STRESC     ,
     2    STRESL     )
      ELSEIF(MATTYP.EQ.VMMIXD)THEN
C von Mises with mixed isotropic/kinematic hardening (infinitesimal
C only)
        CALL SWVMMX
     1(   MODE       ,NLARGE     ,NTYPE      ,LALGVC     ,LALGVL     ,
     2    RALGVC     ,RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      ELSE
C Error: Material type not recognised
        CALL ERRPRT('EI0046')
      ENDIF
      RETURN
      END
```

```
INTEGER FUNCTION NFUNC(N1,N2)
I = N1
J = N2
NF = (J*J-J)/2+I
NFUNC=NF
RETURN
END
```

```
      SUBROUTINE NODAVE
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      DIMENSION
     1    RALGN(MRALGV,MNODE),RSTAN(MRSTAV,MNODE),STRSN(MSTRE,MNODE) ,
     2    THKN(MNODE)
      DIMENSION
     1    STRSA(MSTRE,MGRUP,MPOIN)     ,PSTRA(3)                       ,
     2    RALGA(MRALGV,MGRUP,MPOIN)    ,RSTAA(MRSTAV,MGRUP,MPOIN)    ,
     3    THKA(MGRUP,MPOIN)
      DATA R0   ,R1   /0.0D0,1.0D0/
C***********************************************************************
C PRINTS AVERAGED (SMOOTHED) NODAL STRESSES AND OTHER STATE AND
C ALGORITHMIC VARIABLES. THE SMOOTHED VARIABLE AT EACH NODE IS OBTAINED
C BY EXTRAPOLATING THE VALUE OF THE VARIABLE FROM THE GAUSS POINTS TO
C THE NODE AND THEN AVERAGING THE VALUES OBTAINED FROM ALL ELEMENTS
C SHARING THAT NODE.
C
C REFERENCE: E Hinton & JS Campbel. Local and global smoothing of
C            discontinuous finite element functions using a least
C            squares method. Int. J. Num. Meth. Engng., 8:461-480, 1974.
C            E Hinton & DRJ Owen. An introduction to finite element
C            computations. Pineridge Press, Swansea, 1979.
C***********************************************************************
 1000 FORMAT(//' Averaged nodal stresses and other state',
     1         ' variables for group number ',I2/
     1         ' ========================================',
     3         '===============================')
 1010 FORMAT(/' Node number ',I5,3X,' X-Coord= ',G11.4,
     1        ' Y-Coord= ',G11.4)
 1020 FORMAT(/' Node number ',I5,3X,' R-Coord= ',G11.4,
     1        ' Z-Coord= ',G11.4)
 1030 FORMAT(/' Node number ',I5,3X,' X-Coord= ',G11.4,
     1        ' R-Coord= ',G11.4)
 1040 FORMAT(' S-xx  = ',G12.4,' S-yy  = ',G12.4,' S-xy  = ',G12.4)
 1050 FORMAT(' S-xx  = ',G12.4,' S-yy  = ',G12.4,' S-xy  = ',G12.4,
     1       ' S-zz  = ',G12.4)
 1060 FORMAT(' S-rr  = ',G12.4,' S-zz  = ',G12.4,' S-rz  = ',G12.4,
     1       ' S-h   = ',G12.4)
 1070 FORMAT(' S-xx  = ',G12.4,' S-rr  = ',G12.4,' S-xr  = ',G12.4,
     1       ' S-h   = ',G12.4)
 1080 FORMAT(' S-max = ',G12.4,' S-min = ',G12.4,' Angle = ',G12.4)
 1090 FORMAT(' Thick = ',G12.4)
C
C Loop over element groups
C ==========================
      DO 90 IGRUP=1,NGRUP
        WRITE(16,1000)IGRUP
        DO 10 IPOIN=1,NPOIN
          CALL RVZERO(STRSA(1,IGRUP,IPOIN),MSTRE)
          CALL RVZERO(PSTRA,3)
          CALL RVZERO(RSTAA(1,IGRUP,IPOIN),MRSTAV)
          CALL RVZERO(RALGA(1,IGRUP,IPOIN),MRALGV)
          THKA(IGRUP,IPOIN)=R0
   10   CONTINUE
C Loop over elements
C ------------------
        DO 70 IELEM=1,NELEM
          LGRUP=IGRPID(IELEM)
          IF(LGRUP.NE.IGRUP)GOTO 70
          IELIDN=IELTID(IGRUP)
          NNODE =IELPRP(3,IELIDN)
          NGAUSP=IELPRP(4,IELIDN)
C Extrapolate stresses and other state and algorithmic variables from
C gauss points to nodes
          IPOS=NGAUSP*NDIME+NGAUSP+1
          CALL EXTNOD
     1(   RELPRP(IPOS,IELIDN),
     2    STRSG(1,1,IELEM,1) ,STRSN    ,MSTRE     ,NGAUSP     ,NNODE   )
          CALL EXTNOD
     1(   RELPRP(IPOS,IELIDN),
     2    RSTAVA(1,1,IELEM,1),RSTAN    ,MRSTAV    ,NGAUSP     ,NNODE   )
          CALL EXTNOD
     1(   RELPRP(IPOS,IELIDN),
     2    RALGVA(1,1,IELEM,1),RALGN    ,MRALGV    ,NGAUSP     ,NNODE   )
C thickness (for large strains in plane stress only)
          IF(NLARGE.EQ.1.AND.NTYPE.EQ.1)CALL EXTNOD
     1(   RELPRP(IPOS,IELIDN),
     2    THKGP(1,IELEM,1)   ,THKN     ,1         ,NGAUSP     ,NNODE   )
C Nodal averaging
          DO 60 INODE=1,NNODE
            IPOIN=IABS(LNODS(IELEM,INODE))
            R1DVAL=R1/DBLE(NVALEN(IPOIN,IGRUP))
            DO 30 ISTRE=1,MSTRE
              STRSA(ISTRE,IGRUP,IPOIN)=STRSA(ISTRE,IGRUP,IPOIN)+
     1                                 STRSN(ISTRE,INODE)*R1DVAL
   30       CONTINUE
            DO 40 INTV=1,MRSTAV
              RSTAA(INTV,IGRUP,IPOIN)=RSTAA(INTV,IGRUP,IPOIN)+
     1                                RSTAN(INTV,INODE)*R1DVAL
   40       CONTINUE
            DO 50 IALGV=1,MRALGV
              RALGA(IALGV,IGRUP,IPOIN)=RALGA(IALGV,IGRUP,IPOIN)+
```

```
     1                                 RALGN(IALGV,INODE)*R1DVAL
   50       CONTINUE
            IF(NLARGE.EQ.1.AND.NTYPE.EQ.1)
     1               THKA(IGRUP,IPOIN)=THKA(IGRUP,IPOIN)+
     2                                 THKN(INODE)*R1DVAL
   60     CONTINUE
   70   CONTINUE
C
C Output average stresses to results file (common to all materials)
C ------------------------------------------------------------------
C
        DO 80 IPOIN=1,NPOIN
          IF(NVALEN(IPOIN,IGRUP).EQ.0)GOTO 80
          IF(NTYPE.EQ.1)THEN
            LSTRE=3
            WRITE(16,1010)IPOIN,(COORD(I,IPOIN,1),I=1,NDIME)
            WRITE(16,1040)(STRSA(I,IGRUP,IPOIN),I=1,LSTRE)
          ELSEIF(NTYPE.EQ.2)THEN
            LSTRE=4
            WRITE(16,1010)IPOIN,(COORD(I,IPOIN,1),I=1,NDIME)
            WRITE(16,1050)(STRSA(I,IGRUP,IPOIN),I=1,LSTRE)
          ELSEIF(NTYPE.EQ.3)THEN
            LSTRE=4
            IF(NAXIS.EQ.1)THEN
              WRITE(16,1020)IPOIN,(COORD(I,IPOIN,1),I=1,NDIME)
              WRITE(16,1060)(STRSA(I,IGRUP,IPOIN),I=1,LSTRE)
            ELSE
              WRITE(16,1030)IPOIN,(COORD(I,IPOIN,1),I=1,NDIME)
              WRITE(16,1070)(STRSA(I,IGRUP,IPOIN),I=1,LSTRE)
            ENDIF
          ENDIF
C compute and output principal stresses and angle
          CALL PRINC2(PSTRA,STRSA(1,IGRUP,IPOIN))
          WRITE(16,1080)(PSTRA(I),I=1,3)
C current thickness (for large strains in plane stress only)
          IF(NLARGE.EQ.1.AND.NTYPE.EQ.1)WRITE(16,1090)THKA(IGRUP,IPOIN)
C
C and other (material-specific) state and algorithmic variables
C --------------------------------------------------------------
C
          CALL MATIOR
     1(   NTYPE      ,IPROPS(1,MATTID(IGRUP))  ,RALGA(1,IGRUP,IPOIN)  ,
     2    RPROPS(1,MATTID(IGRUP))             ,RSTAA(1,IGRUP,IPOIN)  ,
     3    STRSA(1,IGRUP,IPOIN)                )
C
   80   CONTINUE
   90 CONTINUE
      RETURN
      END
```

```
      INTEGER FUNCTION NWORD(CHRSTR,IWBEG,IWEND)
      IMPLICIT NONE
      CHARACTER*(*) CHRSTR
      INTEGER IWBEG(*), IWEND(*)
      LOGICAL OUT
      INTEGER I, LEN, LENGTH
C************************************************************************
C FIND NUMBER OF WORDS CONTAINED IN A CHARACTER STRING AND SET POINTERS
C TO BEGINNING AND END OF EACH WORD
C************************************************************************
      LENGTH=LEN(CHRSTR)
      NWORD=0
      OUT=.TRUE.
      DO 10 I=1,LENGTH
        IF(OUT)THEN
          IF(CHRSTR(I:I).NE.' ')THEN
            OUT=.FALSE.
            NWORD=NWORD+1
            IWBEG(NWORD)=I
          ENDIF
        ELSE
          IF(CHRSTR(I:I).EQ.' ')THEN
            OUT=.TRUE.
            IWEND(NWORD)=I-1
          ELSEIF(I.EQ.LENGTH)THEN
            IWEND(NWORD)=I
          ENDIF
        ENDIF
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ORDAMA
     1(   DGAMA      ,NOUTF      ,NTYPE      ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=6  ,MSTRE=4)
      DIMENSION  RSTAVA(MSTRE+2), STRES(*)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR LEMAITRE'S
C DUCTILE DAMAGE ELASTO-PLASTIC MODEL WITH NON-LINEAR ISOTROPIC
C HARDENING
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' R     = ',G12.4,' dgama = ',G12.4)
 2000 FORMAT(' Damage= ',G12.4)
C
C Retrieve current values of hardening variable and damage
      HVAR=RSTAVA(MSTRE+1)
      DAMAGE=RSTAVA(MSTRE+2)
      IF(NTYPE.EQ.1)THEN
C Plane stress
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
C Plane strain and axisymmetric
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,HVAR,DGAMA
      WRITE(NOUTF,2000)DAMAGE
      RETURN
      END
```

```
      SUBROUTINE ORDMEL
     1(   NOUTF      ,NTYPE      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(MSTRE=4)
      DIMENSION  STRES(*)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS FOR ISOTROPICALLY DAMAGED ISOTROPIC ELASTIC MODEL
C ACCOUNTING FOR PARTIAL MICROCRACK/VOID CLOSURE EFFECTS
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Press.= ',G12.4)
C
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+(STRES(4)-P)**2))
      ELSE
        CALL ERRPRT('EI0055')
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,P
      RETURN
      END
```

```fortran
      SUBROUTINE ORDP
     1(   DGAM       ,NOUTF      ,NTYPE      ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MSTRE=4)
      DIMENSION  DGAM(1), RSTAVA(MSTRE+1), STRES(*)
      DATA  R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR DRUCKER-PRAGER
C TYPE ELASTO-PLASTIC MATERIAL WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW
C RULE AND NON-LINEAR ISOTROPIC HARDENING
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Press.= ',G12.4,' Eps.  = ',G12.4,
     1       ' dgama = ',G12.4)
C
      EPBAR=RSTAVA(MSTRE+1)
      IF(NTYPE.EQ.1)THEN
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                     R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                     R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,P,EPBAR,DGAM(1)
      RETURN
      END
```

```
      SUBROUTINE OREL
     1(   NOUTF      ,NTYPE      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(MSTRE=4)
      DIMENSION  STRES(*)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS FOR LINEAR ELASTIC MATERIAL MODEL
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Press.= ',G12.4)
C
      IF(NTYPE.EQ.1)THEN
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                    R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                    R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,P
      RETURN
      END
```

```fortran
      SUBROUTINE ORMC
     1(   DGAM       ,NOUTF      ,NTYPE      ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MSTRE=4)
      DIMENSION  DGAM(2), RSTAVA(MSTRE+1), STRES(*)
      DATA  R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR THE
C MOHR-COULOMB TYPE ELASTO-PLASTIC MATERIAL WITH ASSOCIATIVE/NON-
C ASSOCIATIVE FLOW RULE AND NON-LINEAR ISOTROPIC HARDENING
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Press.= ',G12.4,' Eps.  = ',G12.4,
     1       ' dgama = ',G12.4,' dgamb = ',G12.4)
C
      EPBAR=RSTAVA(MSTRE+1)
      IF(NTYPE.EQ.1)THEN
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                    R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                    R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,P,EPBAR,DGAM(1),DGAM(2)
      RETURN
      END
```

```
      SUBROUTINE OROGD
     1(   NOUTF       ,NTYPE      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  STRES(*)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS FOR OGDEN TYPE HYPERELATIC MATERIAL MODEL
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Press.= ',G12.4)
C
      IF(NTYPE.EQ.1)THEN
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,P
      RETURN
      END
```

```
      SUBROUTINE ORPDSC
     1(   DGAM       ,NOUTF      ,NTYPE      ,RPROPS     ,RSTAVA     ,
     2    STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   IPHARD=6   ,NDIM=2     ,NRSTAV=5   ,NSYST=4    )
C Arguments
      DIMENSION
     1    DGAM(NSYST)        ,RPROPS(*)          ,RSTAVA(NRSTAV)     ,
     2    STRES(*)
C Local arrays and variables
      DIMENSION
     1    AUX1(NDIM,NDIM)    ,DKIRCH(NDIM,NDIM)  ,FE(NDIM,NDIM)      ,
     2    RE(NDIM,NDIM)      ,ROTSTR(NDIM,NDIM)  ,SCHMID(NSYST)      ,
     3    UE(NDIM,NDIM)      ,VECM0(NDIM,NSYST)  ,VECS0(NDIM,NSYST)
      DATA R0   ,R1   ,R3   ,R180   /
     1     0.0D0,1.0D0,3.0D0,180.0D0/
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR THE PLANAR
C DOUBLE-SLIP SINGLE CRYSTAL ELASTO-PLASTIC MODEL
C***********************************************************************
 1000 FORMAT(' Lattice rotation (degrees) = ',G12.4/
     1       ' Accumulated plastic slip   = ',G12.4/
     2       ' tau_1 = ',G12.4,' tau_2 = ',G12.4,' tau_3 = ',G12.4,
     3       ' tau_4 = ',G12.4/
     4       ' dlmd1 = ',G12.4,' dlmd2 = ',G12.4,' dlmd3 = ',G12.4,
     5       ' dlmd4 = ',G12.4)
C Stops program if not plane strain
      IF(NTYPE.NE.2)CALL ERRPRT('EI0037')
C Retrieve stored state variables
C ------------------------------
C... current elastic deformation gradient
      FE(1,1)=RSTAVA(1)
      FE(2,1)=RSTAVA(2)
      FE(1,2)=RSTAVA(3)
      FE(2,2)=RSTAVA(4)
C... current Taylor hardening variable (acummulated slip)
      HRVAR=RSTAVA(NRSTAV)
C Compute lattice rotation
C ------------------------
C Perform polar decomposition of the elastic deformation gradient
      CALL PODEC2
     1(   FE         ,RE         ,UE         )
C From the elastic rotation tensor, compute crystal lattice rotation
      SINE=RE(2,1)
      IF(SINE.GT.R1)SINE=R1
      IF(SINE.LT.-R1)SINE=-R1
      COSINE=RE(1,1)
      IF(COSINE.GT.R1)COSINE=R1
      IF(COSINE.LT.-R1)COSINE=-R1
      DEGRAD=R180/ACOS(-R1)
      SANGLE=DEGRAD*ASIN(SINE)
      CANGLE=DEGRAD*ACOS(COSINE)
      IF(SINE.GE.R0)THEN
        CLROT=CANGLE
      ELSEIF(SINE.LT.R0.AND.COSINE.LT.R0)THEN
        CLROT=-CANGLE
      ELSE
        CLROT=SANGLE
      ENDIF
C Evaluate resolved Schmid stresses (these are not stored in memory)
C ------------------------------------------------------------------
C Set up initial slip systems vectors
C... retrieve initial system orientation
      THETA=RPROPS(4)
C... retrieve relative angle between systems
      BETA=RPROPS(5)
C... system 1:
      VECS0(1,1)=COS(THETA)
      VECS0(2,1)=SIN(THETA)
      VECM0(1,1)=-SIN(THETA)
      VECM0(2,1)=COS(THETA)
C... system 2:
      VECS0(1,2)=COS(THETA+BETA)
      VECS0(2,2)=SIN(THETA+BETA)
      VECM0(1,2)=-SIN(THETA+BETA)
      VECM0(2,2)=COS(THETA+BETA)
C... system 3:
      VECS0(1,3)=-VECS0(1,1)
      VECS0(2,3)=-VECS0(2,1)
      VECM0(1,3)=VECM0(1,1)
      VECM0(2,3)=VECM0(2,1)
C... system 4:
      VECS0(1,4)=-VECS0(1,2)
      VECS0(2,4)=-VECS0(2,2)
      VECM0(1,4)=VECM0(1,2)
      VECM0(2,4)=VECM0(2,2)
C compute rotated stress tensor
      DETF=FE(1,1)*FE(2,2)-FE(1,2)*FE(2,1)
C... deviatoric Kirchhoff stress
      PKIRCH=R1/R3*DETF*(STRES(1)+STRES(2)+STRES(4))
      DKIRCH(1,1)=DETF*STRES(1)-PKIRCH
      DKIRCH(2,2)=DETF*STRES(2)-PKIRCH
      DKIRCH(1,2)=DETF*STRES(3)
      DKIRCH(2,1)=DKIRCH(1,2)
C... rotated deviatoric Kirchhoff stress
      CALL RVZERO(AUX1,NDIM*NDIM)
      DO 30 I=1,NDIM
```

```
        DO 20 J=1,NDIM
          DO 10 K=1,NDIM
            AUX1(I,J)=AUX1(I,J)+RE(K,I)*DKIRCH(K,J)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      CALL RVZERO(ROTSTR,NDIM*NDIM)
      DO 60 I=1,NDIM
        DO 50 J=1,NDIM
          DO 40 K=1,NDIM
            ROTSTR(I,J)=ROTSTR(I,J)+AUX1(I,K)*RE(K,J)
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
C Current Schmid resolved stresses
      CALL RVZERO(SCHMID,NSYST)
      DO 90 ISYST=1,NSYST
        DO 80 I=1,NDIM
          DO 70 J=1,NDIM
            SCHMID(ISYST)=SCHMID(ISYST)+
     1                    ROTSTR(I,J)*VECS0(I,ISYST)*VECM0(J,ISYST)
   70     CONTINUE
   80   CONTINUE
   90 CONTINUE
C
C Write results to output file
C ----------------------------
      WRITE(NOUTF,1000)CLROT,HRVAR,SCHMID(1),SCHMID(2),SCHMID(3),
     1                 SCHMID(4),DGAM(1),DGAM(2),DGAM(3),DGAM(4)
C
      RETURN
      END
```

```
      SUBROUTINE ORTR
     1(   DGAM       ,NOUTF      ,NTYPE      ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
      DIMENSION  DGAM(2), RSTAVA(MSTRE+1), STRES(*)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR TRESCA
C TYPE ELASTO-PLASTIC MATERIAL WITH NON-LINEAR ISOTROPIC HARDENING
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Eps.  = ',G12.4,' dgama = ',G12.4,
     1       ' dgamb = ',G12.4)
C
      EPBAR=RSTAVA(MSTRE+1)
      IF(NTYPE.EQ.1)THEN
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,EPBAR,DGAM(1),DGAM(2)
      RETURN
      END
```

```
      SUBROUTINE ORVM
     1(   DGAMA      ,NOUTF      ,NTYPE      ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
      DIMENSION  RSTAVA(MSTRE+1), STRES(*)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR VON MISES
C TYPE ELASTO-PLASTIC MATERIAL WITH NON-LINEAR ISOTROPIC HARDENING
C***********************************************************************
 1000 FORMAT(' S-eff = ',G12.4,' Eps.  = ',G12.4,' dgama = ',G12.4)
C
      EPBAR=RSTAVA(MSTRE+1)
      IF(NTYPE.EQ.1)THEN
C Plane stress
        P=(STRES(1)+STRES(2))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+P**2))
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
C Plane strain and axisymmetric
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
C Write to output file
      WRITE(NOUTF,1000)EFFST,EPBAR,DGAMA
      RETURN
      END
```

```
      SUBROUTINE ORVMMX
     1(   DGAMA      ,NOUTF      ,NTYPE      ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
C Arguments
      DIMENSION  RSTAVA(2*MSTRE+1), STRES(*)
C Local arrays and variables
      DIMENSION  BACSTR(MSTRE)
      DATA   R2   ,R3    / 2.0D0,3.0D0 /
C***********************************************************************
C OUTPUT RESULTS (INTERNAL AND ALGORITHMIC VARIABLES) FOR VON MISES
C ELASTO-PLASTIC MATERIAL WITH NON-LINEAR MIXED HARDENING
C***********************************************************************
 1000 FORMAT(' b-xx  = ',G12.4,' b-yy  = ',G12.4,' b-xy  = ',G12.4,
     1       ' b-zz  = ',G12.4)
 1100 FORMAT(' S-eff = ',G12.4,' Eps.  = ',G12.4,' dgama = ',G12.4)
C
C Plane strain and axisymmetric only
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0050')
C Print backstress tensor
      BACSTR(1)=RSTAVA(6)
      BACSTR(2)=RSTAVA(7)
      BACSTR(3)=RSTAVA(8)
      BACSTR(4)=RSTAVA(9)
      WRITE(NOUTF,1000)BACSTR(1),BACSTR(2),BACSTR(3),BACSTR(4)
C Compute effective stress
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        P=(STRES(1)+STRES(2)+STRES(4))/R3
        EFFST=SQRT(R3/R2*((STRES(1)-P)**2+(STRES(2)-P)**2+
     1                   R2*STRES(3)**2+(STRES(4)-P)**2))
      ENDIF
      EPBAR=RSTAVA(MSTRE+1)
      WRITE(NOUTF,1100)EFFST,EPBAR,DGAMA
      RETURN
      END
```

```fortran
      SUBROUTINE OUTPUT
     1(   TFACT      ,IINCS      ,IITER      ,NOUTP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      DIMENSION NOUTP(5)
C
      DIMENSION
     1    DERIV(MDIME,MNODE) ,ELCOD(MDIME,MNODE) ,GPCOD(MDIME)       ,
     2    PSTRS(3)           ,SHAPE(MNODE)
      DATA R0/0.0D0/
C***********************************************************************
C OUTPUTS DISPLACEMENTS, REACTIONS, STRESSES AND OTHER STATE AND
C ALGORITHMIC VARIABLES TO RESULTS FILE
C
C REFERENCE: Section 5.4.7
C***********************************************************************
 1020 FORMAT(//' Results for load increment ',I3,' Load factor =',G15.6/
     1 1X,59('=')//' Converged solution at iteration number =',2I5/)
 1030 FORMAT(//' Displacement of structure from initial configuration'/,
     1          ' =====================================================')
 1040 FORMAT(/' Node       R-Disp         Z-Disp')
 1045 FORMAT(/' Node       X-Disp         R-Disp')
 1050 FORMAT(/' Node       X-Disp         Y-Disp')
 1060 FORMAT(I5,2X,6G15.6)
 1070 FORMAT(//' Reactions'/1X,9('='))
 1080 FORMAT(/' Node      R-Force          Z-Force')
 1081 FORMAT(/' Node      X-Force          R-Force')
 1085 FORMAT(/' Node      R-Force          Z-Force          X-Local',
     1'          Y-Local')
 1086 FORMAT(/' Node      X-Force          R-Force          X-Local',
     1'          Y-Local')
 1090 FORMAT(/' Node      X-Force          Y-Force')
 1095 FORMAT(/' Node      X-Force          Y-Force          X-Local',
     1'          Y-Local')
 1100 FORMAT(I5,6(2X,G15.6))
 1105 FORMAT('        ---------------  ---------------'/' Totals',
     1G15.6,2X,G15.6)
 1110 FORMAT(//' Gauss point stresses and and other state variables'/
     1          ' ===================================================')
 1115 FORMAT(//' Element number',I5)
 1150 FORMAT(/' Gauss point ',I2,6X,' X-Coord= ',G11.4,
     1         ' Y-Coord= ',G11.4)
 1152 FORMAT(/' Gauss point ',I2,6X,' R-Coord= ',G11.4,
     1         ' Z-Coord= ',G11.4)
 1153 FORMAT(/' Gauss point ',I2,6X,' X-Coord= ',G11.4,
     1         ' R-Coord= ',G11.4)
 1160 FORMAT(' S-xx  = ',G12.4,' S-yy  = ',G12.4,' S-xy  = ',G12.4)
 1161 FORMAT(' S-xx  = ',G12.4,' S-yy  = ',G12.4,' S-xy  = ',G12.4,
     1        ' S-zz  = ',G12.4)
 1162 FORMAT(' S-rr  = ',G12.4,' S-zz  = ',G12.4,' S-rz  = ',G12.4,
     1        ' S-h   = ',G12.4)
 1163 FORMAT(' S-xx  = ',G12.4,' S-rr  = ',G12.4,' S-xr  = ',G12.4,
     1        ' S-h   = ',G12.4)
 1164 FORMAT(' S-max = ',G12.4,' S-min = ',G12.4,' Angle = ',G12.4)
 1165 FORMAT(' Thick = ',G12.4)
C
C
C Set output flags
C ================
C
      N1=NOUTP(1)
      N2=NOUTP(2)
      N3=NOUTP(3)
      N4=NOUTP(4)
      IF(NALGO.LT.0)THEN
        IF(N1.NE.0)THEN
          IF(MOD(IINCS,N1).EQ.0)THEN
            N1=1
          ELSE
            N1=0
          ENDIF
        ENDIF
        IF(N2.NE.0)THEN
          IF(MOD(IINCS,N2).EQ.0)THEN
            N2=1
          ELSE
            N2=0
          ENDIF
        ENDIF
        IF(N3.NE.0)THEN
          IF(MOD(IINCS,N3).EQ.0)THEN
            N3=1
          ELSE
            N3=0
          ENDIF
        ENDIF
        IF(N4.NE.0)THEN
          IF(MOD(IINCS,N4).EQ.0)THEN
            N4=1
          ELSE
            N4=0
          ENDIF
        ENDIF
```

```
      ENDIF
      IF(N1.EQ.0.AND.N2.EQ.0.AND.N3.EQ.0.AND.N4.EQ.0)RETURN
C
C
      WRITE(16,1020)IINCS,TFACT,IITER
C
C Output displacements
C ====================
C
      IF(N1.NE.0)THEN
        WRITE(16,1030)
        IF(NTYPE.EQ.3)THEN
          IF(NAXIS.EQ.1)THEN
            WRITE(16,1040)
          ELSE
            WRITE(16,1045)
          ENDIF
        ELSE
          WRITE(16,1050)
        ENDIF
        DO 10 IPOIN=1,NPOIN
          NGASH=IPOIN*NDOFN
          NGISH=NGASH-NDOFN+1
          WRITE(16,1060)IPOIN,(TDISP(IGASH),IGASH=NGISH,NGASH)
   10   CONTINUE
      ENDIF
C
C Output reactions
C ================
C
      IF(N2.NE.0)THEN
        WRITE(16,1070)
        DO 45 IVFIX=1,NVFIX
          IF(ANGLE(IVFIX).NE.R0)THEN
            IF(NTYPE.EQ.3)THEN
              IF(NAXIS.EQ.1)THEN
                WRITE(16,1085)
              ELSE
                WRITE(16,1086)
              ENDIF
            ELSE
              WRITE(16,1095)
            ENDIF
            GOTO 47
          ENDIF
   45   CONTINUE
        IF(NTYPE.EQ.3)THEN
          IF(NAXIS.EQ.1)THEN
            WRITE(16,1080)
          ELSE
            WRITE(16,1081)
          ENDIF
        ELSE
          WRITE(16,1090)
        ENDIF
   47   CONTINUE
        TRX=R0
        TRY=R0
        DO 70 IPOIN=1,NPOIN
          ISVAB=(IPOIN-1)*NDOFN
          DO 50 IVFIX=1,NVFIX
            IF(NOFIX(IVFIX).EQ.IPOIN)GOTO 60
   50     CONTINUE
          GOTO 70
   60     CONTINUE
          IF(ANGLE(IVFIX).NE.R0)THEN
            C=COS(ANGLE(IVFIX))
            S=SIN(ANGLE(IVFIX))
            GASHI= C*TREAC(IVFIX,1)+S*TREAC(IVFIX,2)
            GASHJ=-S*TREAC(IVFIX,1)+C*TREAC(IVFIX,2)
            IF(IFFIX(ISVAB+1).EQ.0)GASHI=R0
            IF(IFFIX(ISVAB+2).EQ.0)GASHJ=R0
            WRITE(16,1100)IPOIN,(TREAC(IVFIX,IDOFN),IDOFN=1,NDOFN),
     1                       GASHI,GASHJ
          ELSE
            WRITE(16,1100)IPOIN,(TREAC(IVFIX,IDOFN),IDOFN=1,NDOFN)
          ENDIF
          TRX=TRX+TREAC(IVFIX,1)
          TRY=TRY+TREAC(IVFIX,2)
   70   CONTINUE
        WRITE(16,1105)TRX,TRY
      ENDIF
C
C Stresses and other state and algorithmic variables at gauss points
C ==================================================================
C
      IF(N3.NE.0)THEN
        WRITE(16,1110)
        DO 120 IELEM=1,NELEM
          IGRUP=IGRPID(IELEM)
          IELIDN=IELTID(IGRUP)
          IELTYP=IELPRP(1,IELIDN)
          NNODE =IELPRP(3,IELIDN)
          NGAUSP=IELPRP(4,IELIDN)
C
          WRITE(16,1115)IELEM
C Evaluate Gauss point coordinates
          DO 91 INODE=1,NNODE
```

```
          LNODE=IABS(LNODS(IELEM,INODE))
          DO 90 IDIME=1,NDIME
            ELCOD(IDIME,INODE)=COORD(IDIME,LNODE,1)
   90     CONTINUE
   91   CONTINUE
        IF(NTYPE.EQ.1)THEN
          LSTRE=3
        ELSEIF(NTYPE.EQ.2)THEN
          LSTRE=4
        ELSEIF(NTYPE.EQ.3)THEN
          LSTRE=4
        ENDIF
        IPPOS=1
        DO 110 IGAUSP=1,NGAUSP
          EXISP=RELPRP(IPPOS-1+IGAUSP*2-1,IELIDN)
          ETASP=RELPRP(IPPOS-1+IGAUSP*2  ,IELIDN)
          CALL SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
     2    MDIME      ,SHAPE      )
          CALL GETGCO
     1(   GPCOD      ,ELCOD      ,MDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C Output gauss points stresses (common to all materials)
C ------------------------------------------------------
          IF(NTYPE.EQ.1)THEN
            WRITE(16,1150)IGAUSP,(GPCOD(I),I=1,NDIME)
            WRITE(16,1160)(STRSG(I,IGAUSP,IELEM,1),I=1,LSTRE)
          ELSEIF(NTYPE.EQ.2)THEN
            WRITE(16,1150)IGAUSP,(GPCOD(I),I=1,NDIME)
            WRITE(16,1161)(STRSG(I,IGAUSP,IELEM,1),I=1,LSTRE)
          ELSEIF(NTYPE.EQ.3)THEN
            IF(NAXIS.EQ.1)THEN
              WRITE(16,1152)IGAUSP,(GPCOD(I),I=1,NDIME)
              WRITE(16,1162)(STRSG(I,IGAUSP,IELEM,1),I=1,LSTRE)
            ELSE
              WRITE(16,1153)IGAUSP,(GPCOD(I),I=1,NDIME)
              WRITE(16,1163)(STRSG(I,IGAUSP,IELEM,1),I=1,LSTRE)
            ENDIF
          ENDIF
C and principal stresses
          CALL PRINC2(PSTRS,STRSG(1,IGAUSP,IELEM,1))
          WRITE(16,1164)(PSTRS(I),I=1,3)
C output current thickness (for large strains in plane stress only)
          IF(NLARGE.EQ.1.AND.NTYPE.EQ.1)THEN
            WRITE(16,1165)THKGP(IGAUSP,IELEM,1)
          ENDIF
C Output other (material-specific) state and algorithmic variables
C ----------------------------------------------------------------
          CALL MATIOR
     1(   NTYPE                   ,IPROPS(1,MATTID(IGRPID(IELEM)))  ,
     2    RALGVA(1,IGAUSP,IELEM,1) ,RPROPS(1,MATTID(IGRPID(IELEM)))  ,
     3    RSTAVA(1,IGAUSP,IELEM,1) ,STRSG(1,IGAUSP,IELEM,1)          )
C
  110   CONTINUE
  120 CONTINUE
      ENDIF
C
C Stresses and other state and algorithmic variables at nodes
C ===========================================================
C
      IF(N4.NE.0)THEN
        CALL NODAVE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE PEXIT
C Print message
      WRITE(*,'(///15X,A,///)')'Program HYPLAS aborted.'
      WRITE(16,'(///15X,A,///)')'Program HYPLAS aborted.'
C Close files
      CALL FCLOSE
C and exit program
      STOP ' '
      END
```

```
      DOUBLE PRECISION FUNCTION PLFUN(X, NPOINT, XFX)
C
      INTEGER NPOINT, I
      DOUBLE PRECISION X, XFX(2,*)
C*************************************************************************
C PIECEWISE LINEAR FUNCTION DEFINED BY A SET OF NPOINT PAIRS
C {X,F(X)} STORED IN THE MATRIX XFX (DIM. 2*NPOINT).
C*************************************************************************
      DO 100 I=1,NPOINT
        IF (X.GE.XFX(1,I)) THEN
          GOTO 100
        ELSE
          IF (I.EQ.1) THEN
C           -- x < x1 --> f(x)=f(x1) ---
            PLFUN=XFX(2,1)
            GOTO 999
          ELSE
C           -- x(i-1) <= x < x(i) ---
            PLFUN=XFX(2,I-1)+(X-XFX(1,I-1))*
     1                 (XFX(2,I)-XFX(2,I-1))/
     2                 (XFX(1,I)-XFX(1,I-1))
            GOTO 999
          ENDIF
        ENDIF
 100  CONTINUE
C     ----  x >= x(npoint) --> f(x) = f(x(npoint))  ---
      PLFUN=XFX(2,NPOINT)
 999  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PODEC2
     1(   F          ,R          ,U          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   NDIM=2     )
C Arguments
      DIMENSION
     1    F(NDIM,NDIM)       ,R(NDIM,NDIM)       ,U(NDIM,NDIM)
C Local variables and arrays
      LOGICAL DUMMY
      DIMENSION
     1    C(NDIM,NDIM)       ,CVEC(4)            ,EIGPRJ(4,NDIM)     ,
     2    EIGC(NDIM)         ,UM1(NDIM,NDIM)     ,UM1VEC(3)          ,
     3    UVEC(3)
      DATA
     1    R1   /
     2    1.0D0/
C***********************************************************************
C PERFORMS THE RIGHT POLAR DECOMPOSITION OF A 2-D TENSOR
C
C REFERENCE: Section 2.2.9
C***********************************************************************
C                 T
C Compute  C := F  F
C -------------------
      CALL RVZERO(C,NDIM*NDIM)
      DO 30 I=1,NDIM
        DO 20 J=1,NDIM
          DO 10 K=1,NDIM
            C(I,J)=C(I,J)+F(K,I)*F(K,J)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C Perform spectral decomposition of C
C -----------------------------------
      CVEC(1)=C(1,1)
      CVEC(2)=C(2,2)
      CVEC(3)=C(1,2)
      CALL SPDEC2(EIGPRJ     ,EIGC       ,DUMMY      ,CVEC       )
C
C                  1/2          -1
C Compute  U := (C)    and   U
C -----------------------------
C assemble in vector form
      CALL RVZERO(UVEC,3)
      CALL RVZERO(UM1VEC,3)
      DO 50 IDIM=1,NDIM
        UEIG=SQRT(EIGC(IDIM))
        UM1EIG=R1/UEIG
        DO 40 ICOMP=1,3
          UVEC(ICOMP)=UVEC(ICOMP)+UEIG*EIGPRJ(ICOMP,IDIM)
          UM1VEC(ICOMP)=UM1VEC(ICOMP)+UM1EIG*EIGPRJ(ICOMP,IDIM)
   40   CONTINUE
   50 CONTINUE
C and matrix form
      U(1,1)=UVEC(1)
      U(2,2)=UVEC(2)
      U(1,2)=UVEC(3)
      U(2,1)=UVEC(3)
      UM1(1,1)=UM1VEC(1)
      UM1(2,2)=UM1VEC(2)
      UM1(1,2)=UM1VEC(3)
      UM1(2,1)=UM1VEC(3)
C                            -1
C Compute rotation  R := F U
C ---------------------------
      CALL RVZERO(R,NDIM*NDIM)
      DO 80 I=1,NDIM
        DO 70 J=1,NDIM
          DO 60 K=1,NDIM
            R(I,J)=R(I,J)+F(I,K)*UM1(K,J)
   60     CONTINUE
   70   CONTINUE
   80 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PRINC2(PSTRS ,STRSG )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION PSTRS(3),STRSG(3)
      DATA R0   ,RP01  ,RP5  ,R1   ,R4   ,R90   ,SMALL/
     1     0.0D0,0.01D0,0.5D0,1.0D0,4.0D0,90.0D0,1.D-6/
C***********************************************************************
C COMPUTES THE PRINCIPAL STRESSES FOR TWO-DIMENSIONAL STRESSES
C***********************************************************************
      PI=R4*ATAN(R1)
      XGASH=(STRSG(1)+STRSG(2))*RP5
      XGISH=(STRSG(1)-STRSG(2))*RP5
      XGESH=STRSG(3)
      XGOSH=SQRT(XGISH*XGISH+XGESH*XGESH)
      PSTRS(1)=XGASH+XGOSH
      PSTRS(2)=XGASH-XGOSH
      AUX=SQRT(STRSG(1)**2+STRSG(2)**2+STRSG(3)**2)
      IF(AUX.EQ.R0)AUX=R1
      IF(ABS(XGESH/AUX).LT.SMALL.AND.ABS(XGISH/AUX).LT.SMALL)THEN
        PSTRS(3)=R0
      ELSE
        PSTRS(3)=(ATAN2(XGESH,XGISH))*R90/PI
        IF(PSTRS(3).LT.(-R90+RP01))PSTRS(3)=R90
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE RDDAMA
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRPROP     ,MRSTAV     ,
     2    NTYPE      ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPHARD=6  ,NLALGV=2  ,NRSTAV=6 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
      DATA R0   /0.0D0/
C***********************************************************************
C READS AND ECHOES MATERIAL PROPERTIES FOR LEMAITRE'S DUCTILE DAMAGE
C ELASTO-PLASTIC MATERIAL MODEL WITH NON-LINEAR (PIECEWISE LINEAR)
C ISOTROPIC HARDENING
C
C REFERENCE Box: 12.3
C***********************************************************************
 1000 FORMAT(' LEMAITRE''S DUCTILE DAMAGE elasto-plastic model'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus .................................. =',G15.6/
     3' Poisson''s ratio .................................. =',G15.6/
     4' Damage evolution law exponent ..................... =',G15.6/
     5' Damage evolution law denominator .................. =',G15.6)
 1200 FORMAT(/
     1' Number of points on hardening curve ............... =',I3//
     2'           R           uniaxial yield stress '/)
 1300 FORMAT(2(5X,G15.6))
C
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('ED0172')
C Set unsymmetric tangent stiffness flag
      UNSYM=.TRUE.
C
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS,DAMEXP,DAMDEN
      WRITE(16,1100)DENSE,YOUNG,POISS,DAMEXP,DAMDEN
      IF(YOUNG.LE.R0)CALL ERRPRT('ED0164')
      IF(DAMEXP.LE.R0)CALL ERRPRT('ED0170')
      IF(DAMDEN.LE.R0)CALL ERRPRT('ED0171')
C number of points on hardening curve
      READ(15,*)NHARD
      WRITE(16,1200)NHARD
      IF(NHARD.LT.2) CALL ERRPRT('ED0165')
C check dimensions of IPROPS
      IF(MIPROP.LT.3)CALL ERRPRT('ED0166')
      IPROPS(3)=NHARD
C check dimensions of RPROPS
      NRPROP=IPHARD+NHARD*2-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0167')
C
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
      RPROPS(4)=DAMEXP
      RPROPS(5)=DAMDEN
C Read and set hardening curve
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPHARD+IHARD*2-2),
     1            RPROPS(IPHARD+IHARD*2-1)
        WRITE(16,1300)RPROPS(IPHARD+IHARD*2-2),
     1                RPROPS(IPHARD+IHARD*2-1)
   10 CONTINUE
C Check dimension of RSTAVA and LALGVA
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0168')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0169')
C
      RETURN
      END
```

```
      SUBROUTINE RDDMEL
     1(   NTYPE      ,MRPROP     ,MRSTAV     ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( NRPROP=7  ,NRSTAV=4 )
      DIMENSION
     1    RPROPS(*)
      DATA R0   ,RP5  ,R1   ,R2   ,R3   /
     1     0.0D0,0.5D0,1.0D0,2.0D0,3.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR ISOTROPICALLY DAMAGED ISOTROPIC
C ELASTIC MATERIAL MODEL WITH PARTIAL MICROCRACK/VOID CLOSURE EFFECTS
C
C REFERENCE: Section 12.6.1
C***********************************************************************
 1000 FORMAT(' DAMAGED ELASTIC material (damaged HENCKY material in',
     1       ' large strains)'/
     2       ' with partial microcrack closure effect'/)
 1010 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus .................................. =',G15.6/
     3' Poisson''s ratio .................................. =',G15.6/
     4' Damage constant (D) .............................. =',G15.6/
     5' Crack closure parameter (h) ....................... =',G15.6)
C
C Check that required stress state type is implemented
C
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('ED0173')
C
C Set unsymmetric tangent stiffness flag
C
      UNSYM=.FALSE.
C
C Read and echo material properties
C
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS
      READ(15,*)DAMAGE,HFACT
      WRITE(16,1010)DENSE,YOUNG,POISS,DAMAGE,HFACT
C Perform checks
      IF(YOUNG.LT.R0)CALL ERRPRT('ED0174')
      IF(POISS.LE.-R1.AND.POISS.GE.RP5)CALL ERRPRT('ED0175')
      IF(DAMAGE.LT.R0.OR.DAMAGE.GE.R1)CALL ERRPRT('ED0176')
      IF(HFACT.LT.R0.OR.HFACT.GT.R1)CALL ERRPRT('ED0177')
C
C Check dimensioning
C
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0193')
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0194')
C
C Set vector of real material properties
C
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
      RPROPS(4)=DAMAGE
      RPROPS(5)=HFACT
      RPROPS(6)=GMODU
      RPROPS(7)=BULK
C
      RETURN
      END
```

```fortran
      SUBROUTINE RDDP
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRALGV     ,MRPROP     ,
     2    MRSTAV     ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPHARD=7  ,NIPROP=3  ,NLALGV=3  ,NRALGV=3  ,NRSTAV=5 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
      DATA R0   ,R1   ,R2   ,R3   ,R6   ,R9   ,R12   ,R90   ,R180   /
     1     0.0D0,1.0D0,2.0D0,3.0D0,6.0D0,9.0D0,12.0D0,90.0D0,180.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR DRUCKER-PRAGER TYPE
C ELASTO-PLASTIC MATERIAL WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW
C RULE AND NON-LINEAR ISOTROPIC HARDENING
C
C REFERENCE:  Section 8.3
C***********************************************************************
 1000 FORMAT(' Elasto-plastic with DRUCKER-PRAGER yield criterion'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus .................................. =',G15.6/
     3' Poisson''s ratio .................................. =',G15.6/
     4' Friction angle  (degrees) ......................... =',G15.6/
     5' Dilatancy angle (degrees) ......................... =',G15.6)
 1200 FORMAT(/
     1'  Friction and dilatancy angles coincide ->',
     2 ' ASSOCIATIVE flow')
 1300 FORMAT(/
     1'  Friction and dilatancy angles are distinct ->',
     2 ' NON-ASSOCIATIVE flow')
 1350 FORMAT(/
     1'  OUTER EDGES match with Mohr-Coulomb criterion selected')
 1360 FORMAT(/
     1'  INNER EDGES match with Mohr-Coulomb criterion selected')
 1370 FORMAT(/
     1'  PLANE STRAIN match with Mohr-Coulomb criterion selected')
 1380 FORMAT(/
     1'  UNIAXIAL COMPRESSION and UNIAXIAL TENSION match with',/,
     2'  Mohr-Coulomb criterion selected')
 1390 FORMAT(/
     1'  BIAXIAL COMPRESSION and BIAXIAL TENSION match with',/,
     2'  Mohr-Coulomb criterion selected')
 1400 FORMAT(/
     1' Number of points on hardening curve ............... =',I3//
     2'           Epstn            cohesion'/)
 1500 FORMAT(2(5X,G15.6))
C
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS,PHI,PSI,IFLAG
      WRITE(16,1100)DENSE,YOUNG,POISS,PHI,PSI
C Check validity if some material properties
      IF(YOUNG.LT.R0)THEN
        CALL ERRPRT('ED0114')
      ENDIF
      IF(PHI.LT.R0.OR.PHI.GE.R90.OR.PSI.LT.R0.OR.PSI.GE.R90)THEN
        CALL ERRPRT('ED0115')
      ENDIF
C Check friction and dilatancy angles
      IF(PHI.EQ.PSI)THEN
        WRITE(16,1200)
      ELSE
        WRITE(16,1300)
      ENDIF
C Echo selected approximation of Mohr-Coulomb criterion
C and set related material constants
      ROOT3=SQRT(R3)
      RADEG=ACOS(-R1)/R180
      PHIRAD=PHI*RADEG
      SINPHI=SIN(PHIRAD)
      COSPHI=COS(PHIRAD)
      TANPHI=TAN(PHIRAD)
      PSIRAD=PSI*RADEG
      SINPSI=SIN(PSIRAD)
      TANPSI=TAN(PSIRAD)
      IF(IFLAG.EQ.0)THEN
C Outer edge match with Mohr-Coulomb criterion
        WRITE(16,1350)
        DENOMA=ROOT3*(R3-SINPHI)
        DENOMB=ROOT3*(R3-SINPSI)
        ETA=R6*SINPHI/DENOMA
        XI=R6*COSPHI/DENOMA
        ETABAR=R6*SINPSI/DENOMB
      ELSEIF(IFLAG.EQ.1)THEN
C Inner edge match with Mohr-Coulomb criterion
        WRITE(16,1360)
        DENOMA=ROOT3*(R3+SINPHI)
        DENOMB=ROOT3*(R3+SINPSI)
        ETA=R6*SINPHI/DENOMA
        XI=R6*COSPHI/DENOMA
        ETABAR=R6*SINPSI/DENOMB
      ELSEIF(IFLAG.EQ.2)THEN
C Plane strain match with Mohr-Coulomb criterion
        WRITE(16,1370)
        DENOMA=SQRT(R9+R12*TANPHI**2)
        DENOMB=SQRT(R9+R12*TANPSI**2)
        ETA=R3*TANPHI/DENOMA
```

```
        XI=R3/DENOMA
        ETABAR=R3*TANPSI/DENOMB
      ELSEIF(IFLAG.EQ.3)THEN
C Match Mohr-Coulomb criterion in uniaxial compression and uniaxial
C tension
        WRITE(16,1380)
        ETA=R3*SINPHI/ROOT3
        XI=R2*COSPHI/ROOT3
        ETABAR=R3*SINPSI/ROOT3
      ELSEIF(IFLAG.EQ.4)THEN
C Match Mohr-Coulomb criterion in biaxial compression and biaxial
C tension
        WRITE(16,1390)
        ETA=R3*SINPHI/(R2*ROOT3)
        XI=R2*COSPHI/ROOT3
        ETABAR=R3*SINPSI/(R2*ROOT3)
      ELSE
        CALL ERRPRT('ED0116')
      ENDIF
C Hardening curve
      READ(15,*)NHARD
      WRITE(16,1400)NHARD
      IF(NHARD.LT.2)THEN
        CALL ERRPRT('ED0117')
      ENDIF
C Check dimensions of IPROPS
      IF(MIPROP.LT.NIPROP)CALL ERRPRT('ED0189')
      IPROPS(3)=NHARD
C Check dimensions of RPROPS
      NRPROP=IPHARD+NHARD*2-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0188')
C Store real properties in RPROPS
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
      RPROPS(4)=ETA
      RPROPS(5)=XI
      RPROPS(6)=ETABAR
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPHARD+IHARD*2-2),
     1            RPROPS(IPHARD+IHARD*2-1)
        WRITE(16,1500)RPROPS(IPHARD+IHARD*2-2),
     1                RPROPS(IPHARD+IHARD*2-1)
   10 CONTINUE
C
C Check dimension of RSTAVA, LALGVA and RALGVA
C
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0190')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0191')
      IF(NRALGV.GT.MRALGV)CALL ERRPRT('ED0192')
C
C Set unsymmetric tangent stiffness flag
      IF(PHI.EQ.PSI)THEN
        UNSYM=.FALSE.
      ELSE
        UNSYM=.TRUE.
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE RDEL
     1(   MRPROP     ,MRSTAV     ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( NRSTAV=4 )
      DIMENSION
     1    RPROPS(*)
      DATA R0   ,RP5  ,R1   ,R2   ,R3   /
     1     0.0D0,0.5D0,1.0D0,2.0D0,3.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR LINEAR ELASTIC MATERIAL MODEL
C***********************************************************************
 1000 FORMAT(' LINEAR ELASTIC material (HENCKY material in large',
     1       ' strains)'/)
 1010 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus ................................... =',G15.6/
     3' Poisson''s ratio ................................... =',G15.6)
C
C Set unsymmetric tangent stiffness flag
C
      UNSYM=.FALSE.
C
C Read and echo material properties
C
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS
      WRITE(16,1010)DENSE,YOUNG,POISS
      IF(YOUNG.LT.R0)CALL ERRPRT('ED0077')
      IF(POISS.LE.-R1.AND.POISS.GE.RP5)CALL ERRPRT('ED0078')
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
C
C Set vector of real material properties
C
      NRPROP=4
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0181')
      RPROPS(1)=DENSE
      RPROPS(2)=GMODU
      RPROPS(3)=BULK
      RPROPS(4)=YOUNG
C
C Check dimensioning of RSTAVA
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0182')
C
      RETURN
      END
```

```
      SUBROUTINE RDMC
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRALGV     ,MRPROP     ,
     2    MRSTAV     ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPHARD=7  ,NIPROP=3  ,NLALGV=5  ,NRALGV=2  ,NRSTAV=5 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
      DATA R0   ,R1   ,R90   ,R180   /
     1     0.0D0,1.0D0,90.0D0,180.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR MOHR-COULOMB TYPE
C ELASTO-PLASTIC MATERIAL WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW
C RULE AND NON-LINEAR ISOTROPIC HARDENING
C
C REFERENCE: Section 8.2
C***********************************************************************
 1000 FORMAT(' Elasto-plastic with MOHR-COULOMB yield criterion'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus ................................... =',G15.6/
     3' Poisson''s ratio ................................... =',G15.6/
     4' Friction angle  (degrees) ......................... =',G15.6/
     5' Dilatancy angle (degrees) ......................... =',G15.6)
 1200 FORMAT(/
     1'  Friction and dilatancy angles coincide ->',
     2 ' ASSOCIATIVE flow')
 1300 FORMAT(/
     1'  Friction and dilatancy angles are distinct ->',
     2 ' NON-ASSOCIATIVE flow')
 1400 FORMAT(/
     1' Number of points on hardening curve ............... =',I3//
     2'           Epstn            cohesion'/)
 1500 FORMAT(2(5X,G15.6))
C
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS,PHI,PSI
      WRITE(16,1100)DENSE,YOUNG,POISS,PHI,PSI
C Check validity if some material properties
      IF(YOUNG.LT.R0)CALL ERRPRT('ED0136')
      IF(PHI.LT.R0.OR.PHI.GE.R90)CALL ERRPRT('ED0137')
C Check and echo if flow rule is associative or non-associative
      IF(PHI.EQ.PSI)THEN
        WRITE(16,1200)
      ELSE
        WRITE(16,1300)
      ENDIF
C Set material constants array
      RADEG=ACOS(-R1)/R180
      PHIRAD=PHI*RADEG
      SINPHI=SIN(PHIRAD)
      COSPHI=COS(PHIRAD)
      PSIRAD=PSI*RADEG
      SINPSI=SIN(PSIRAD)
C Hardening curve
      READ(15,*)NHARD
      WRITE(16,1400)NHARD
      IF(NHARD.LT.2)CALL ERRPRT('ED0138')
C Check dimension of IPROPS
      IF(MIPROP.LT.NIPROP)CALL ERRPRT('ED0187')
      IPROPS(3)=NHARD
C Check dimension of RPROPS
      NRPROP=IPHARD+NHARD*2-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0183')
C Store real properties in RPROPS
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
      RPROPS(4)=SINPHI
      RPROPS(5)=COSPHI
      RPROPS(6)=SINPSI
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPHARD+IHARD*2-2),
     1            RPROPS(IPHARD+IHARD*2-1)
        WRITE(16,1500)RPROPS(IPHARD+IHARD*2-2),
     1                RPROPS(IPHARD+IHARD*2-1)
   10 CONTINUE
C
C Check dimension of RSTAVA, LALGVA and RALGVA
C
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0184')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0185')
      IF(NRALGV.GT.MRALGV)CALL ERRPRT('ED0186')
C
C Set unsymmetric tangent stiffness flag for non-associative flow
      IF(PHI.NE.PSI)THEN
        UNSYM=.TRUE.
      ELSE
        UNSYM=.FALSE.
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE RDOGD
     1(   IPROPS     ,MIPROP     ,MRPROP     ,MRSTAV     ,RPROPS     ,
     2    UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPOGDC=2  ,NIPROP=3  ,NRSTAV=4 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
C***********************************************************************
C READS AND ECHOES MATERIAL PROPERTIES FOR OGDEN TYPE HYPERELASTIC
C MATERIAL MODEL
C
C REFERENCE: Section 13.2.2
C***********************************************************************
 1000 FORMAT(' Ogden type hyperelastic material'/)
 1010 FORMAT(
     1' Mass density ...................................... =',G15.6)
 1020 FORMAT(/
     1' Number of terms in Ogden''s strain-energy function.. =',I3//
     2'            mu                  alpha'/)
 1030 FORMAT(2(5X,G15.6))
 1040 FORMAT(/
     1' Bulk modulus ...................................... =',G15.6/)
C
C Set unsymmetric tangent stiffness flag
      UNSYM=.FALSE.
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      WRITE(16,1010)DENSE
C Ogden's constants
      READ(15,*)NOGTRM
      WRITE(16,1020)NOGTRM
      IF(NOGTRM.LT.1)CALL ERRPRT('ED0067')
C Check dimension of IPROPS
      IF(MIPROP.LT.NIPROP)CALL ERRPRT('ED0195')
      IPROPS(3)=NOGTRM
C Check dimension of RPROPS
      NRPROP=IPOGDC+NOGTRM*2
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0196')
C Store real properties in RPROPS
      RPROPS(1)=DENSE
      DO 10 I=1,NOGTRM
        READ(15,*)RPROPS(IPOGDC+I*2-2),RPROPS(IPOGDC+I*2-1)
        WRITE(16,1030)RPROPS(IPOGDC+I*2-2),RPROPS(IPOGDC+I*2-1)
   10 CONTINUE
C Bulk modulus
      READ(15,*)BULK
      RPROPS(IPOGDC+NOGTRM*2)=BULK
      WRITE(16,1040)BULK
C Check dimension of RSTAVA
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0197')
C
      RETURN
      END
```

```
      SUBROUTINE RDPDSC
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRALGV     ,MRPROP     ,
     2    MRSTAV     ,NLARGE     ,NTYPE      ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   IPHARD=6   ,NIPROP=3   ,NLALGV=6   ,NRALGV=4   ,NRSTAV=5   )
C Arguments
      LOGICAL UNSYM
      DIMENSION
     1    IPROPS(MIPROP)     ,RPROPS(MRPROP)
C Local data
      DATA R1   ,R180   /
     1     1.0D0,180.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR THE LARGE STRAIN PLANAR
C DOUBLE-SLIP SINGLE CRYSTAL ELASTO-PLASTIC MODEL WITH NON-LINEAR
C (PIECEWISE LINEAR) ISOTROPIC TAYLOR HARDENING
C***********************************************************************
 1000 FORMAT(' Large strain planar double-slip SINGLE CRYSTAL'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Shear modulus ..................................... =',G15.6/
     3' Bulk modulus ...................................... =',G15.6/
     4' Initial orientation of FIRST SLIP SYSTEM relative'           /
     5' to X-axis (degrees, counterclockwise-positive) .... =',G15.6/
     6' Initial orientation of SECOND SLIP SYSTEM relative'          /
     7' to first syst. (degrees, counterclockwise-positive) =',G15.6)
 1200 FORMAT(/
     1' Number of points on Taylor hardening curve ........ =',I3//
     2'       Accum. slip    Resolved Schmid yield stress '/)
 1300 FORMAT(2(5X,G15.6))
C
C Set unsymmetric tangent stiffness flag
      UNSYM=.TRUE.
C Check that stress state type and large strain flags are compatible
C with the present model
      IF(NTYPE.NE.2)CALL ERRPRT('ED0154')
      IF(NLARGE.NE.1)CALL ERRPRT('ED0155')
C
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)GMODU,BULK,THETA,BETA
      WRITE(16,1100)DENSE,GMODU,BULK,THETA,BETA
C number of points on hardening curve
      READ(15,*)NHARD
      WRITE(16,1200)NHARD
      IF(NHARD.LT.2) CALL ERRPRT('ED0148')
C check dimensions of IPROPS
      IF(NIPROP.GT.MIPROP)CALL ERRPRT('ED0149')
      IPROPS(3)=NHARD
C check dimensions of RPROPS
      NRPROP=IPHARD+NHARD*2-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0150')
C convert angles into radians
      RADEG=ACOS(-R1)/R180
      THETA=RADEG*THETA
      BETA=RADEG*BETA
      RPROPS(1)=DENSE
      RPROPS(2)=GMODU
      RPROPS(3)=BULK
      RPROPS(4)=THETA
      RPROPS(5)=BETA
C Read and set hardening curve
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPHARD+IHARD*2-2),
     1            RPROPS(IPHARD+IHARD*2-1)
        WRITE(16,1300)RPROPS(IPHARD+IHARD*2-2),
     1                RPROPS(IPHARD+IHARD*2-1)
   10 CONTINUE
C Check dimension of RSTAVA, RALGVA and LALGVA
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0151')
      IF(NRALGV.GT.MRALGV)CALL ERRPRT('ED0152')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0153')
C
      RETURN
      END
```

```
      SUBROUTINE RDTR
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRALGV     ,MRPROP     ,
     2    MRSTAV     ,NLARGE     ,NTYPE      ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPHARD=4  ,NIPROP=3  ,NLALGV=4  ,NRALGV=2  ,NRSTAV=5 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
      DATA R0 / 0.0D0 /
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR TRESCA TYPE ELASTO-PLASTIC
C MATERIAL WITH NON-LINEAR ISOTROPIC HARDENING
C
C REFERENCE: Section 8.1
C***********************************************************************
 1000 FORMAT(' Elasto-plastic with TRESCA yield criterion'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus ................................... =',G15.6/
     3' Poisson''s ratio ................................... =',G15.6)
 1200 FORMAT(/
     1' Number of points on hardening curve ............... =',I3//
     2'           Epstn        uniaxial yield stress '/)
 1300 FORMAT(2(5X,G15.6))
C
C Model not yet implemented for large strains with plane stress
      IF(NLARGE.EQ.1.AND.NTYPE.EQ.1)CALL ERRPRT('ED0198')
C Set unsymmetric tangent stiffness flag
      UNSYM=.FALSE.
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS
      WRITE(16,1100)DENSE,YOUNG,POISS
      IF(YOUNG.LT.R0)CALL ERRPRT('ED0107')
C Hardening curve
      READ(15,*)NHARD
      WRITE(16,1200)NHARD
      IF(NHARD.LT.2) CALL ERRPRT('ED0108')
C Check dimensions of IPROPS
      IF(MIPROP.LT.NIPROP)CALL ERRPRT('ED0109')
      IPROPS(3)=NHARD
C Check dimensions of RPROPS
      NRPROP=IPHARD+NHARD*2-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0110')
C Store real properties in RPROPS
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
C
C Read and set hardening curve
C
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPHARD+IHARD*2-2),
     1            RPROPS(IPHARD+IHARD*2-1)
        WRITE(16,1300)RPROPS(IPHARD+IHARD*2-2),
     1                RPROPS(IPHARD+IHARD*2-1)
   10 CONTINUE
C
C Check dimension of RSTAVA, LALGVA and RALGVA
C
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0111')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0112')
      IF(NRALGV.GT.MRALGV)CALL ERRPRT('ED0113')
C
      RETURN
      END
```

```
      SUBROUTINE RDVM
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRPROP     ,MRSTAV     ,
     2    RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPHARD=4  ,NLALGV=2  ,NRSTAV=5 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
      DATA R0   /0.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR VON MISES TYPE ELASTO-PLASTIC
C MATERIAL WITH NON-LINEAR (PIECEWISE LINEAR) ISOTROPIC HARDENING
C
C REFERENCE: Section 7.3.5
C***********************************************************************
 1000 FORMAT(' Elasto-plastic with VON MISES yield criterion'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus .................................. =',G15.6/
     3' Poisson''s ratio .................................. =',G15.6)
 1200 FORMAT(/
     1' Number of points on hardening curve ............... =',I3//
     2'           Epstn        uniaxial yield stress '/)
 1300 FORMAT(2(5X,G15.6))
C
C Set unsymmetric tangent stiffness flag
      UNSYM=.FALSE.
C
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS
      WRITE(16,1100)DENSE,YOUNG,POISS
      IF(YOUNG.LE.R0)CALL ERRPRT('ED0100')
C number of points on hardening curve
      READ(15,*)NHARD
      WRITE(16,1200)NHARD
      IF(NHARD.LT.2) CALL ERRPRT('ED0101')
C check dimensions of IPROPS
      IF(MIPROP.LT.3)CALL ERRPRT('ED0102')
      IPROPS(3)=NHARD
C check dimensions of RPROPS
      NRPROP=IPHARD+NHARD*2-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0103')
C
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
C Read and set hardening curve
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPHARD+IHARD*2-2),
     1            RPROPS(IPHARD+IHARD*2-1)
        WRITE(16,1300)RPROPS(IPHARD+IHARD*2-2),
     1                RPROPS(IPHARD+IHARD*2-1)
   10 CONTINUE
C Check dimension of RSTAVA and LALGVA
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0104')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0105')
C
      RETURN
      END
```

```fortran
      SUBROUTINE RDVMMX
     1(   IPROPS     ,MIPROP     ,MLALGV     ,MRPROP     ,MRSTAV     ,
     2    NLARGE     ,NTYPE      ,RPROPS     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      PARAMETER( IPHARD=4  ,NLALGV=2  ,NRSTAV=9 )
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)
      DATA R0   /0.0D0/
C***********************************************************************
C READ AND ECHO MATERIAL PROPERTIES FOR VON MISES TYPE ELASTO-PLASTIC
C MATERIAL WITH NON-LINEAR (PIECEWISE LINEAR) MIXED HARDENING
C
C REFERENCE: Section 7.6.1
C***********************************************************************
 1000 FORMAT(' VON MISES elasto-plastic model with mixed hardening'/)
 1100 FORMAT(
     1' Mass density ...................................... =',G15.6/
     2' Young''s modulus .................................. =',G15.6/
     3' Poisson''s ratio .................................. =',G15.6)
 1200 FORMAT(/
     1' Number of points on hardening curves ............. =',I3//
     2'          Epbar         isotr.hard. stress ',
     3'   kin.hard. stress'/)
 1300 FORMAT(3(5X,G15.6))
C
C Model currently implemented for plane strain and axisymmetric states
C only
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('ED0156')
C Only small strain implementation is currently available
      IF(NLARGE.EQ.1)CALL ERRPRT('ED0157')
C
C Set unsymmetric tangent stiffness flag
      UNSYM=.FALSE.
C Read and echo some of the real properties
      WRITE(16,1000)
      READ(15,*)DENSE
      READ(15,*)YOUNG,POISS
      WRITE(16,1100)DENSE,YOUNG,POISS
      IF(YOUNG.LT.R0)CALL ERRPRT('ED0158')
C number of points on hardening curve
      READ(15,*)NHARD
      WRITE(16,1200)NHARD
      IF(NHARD.LT.2) CALL ERRPRT('ED0159')
C check dimensions of IPROPS
      IF(MIPROP.LT.3)CALL ERRPRT('ED0160')
      IPROPS(3)=NHARD
C check dimensions of RPROPS
      NRPROP=IPHARD+NHARD*4-1
      IF(NRPROP.GT.MRPROP)CALL ERRPRT('ED0161')
C
      RPROPS(1)=DENSE
      RPROPS(2)=YOUNG
      RPROPS(3)=POISS
C Read and set isotropic and kinematic hardening curves
      IPIHAR=IPHARD
      IPKHAR=IPHARD+2*NHARD
      DO 10 IHARD=1,NHARD
        READ(15,*)RPROPS(IPIHAR+IHARD*2-2),RPROPS(IPIHAR+IHARD*2-1),
     1                                     RPROPS(IPKHAR+IHARD*2-1)
        RPROPS(IPKHAR+IHARD*2-2)=RPROPS(IPIHAR+IHARD*2-2)
        WRITE(16,1300)RPROPS(IPHARD+IHARD*2-2),RPROPS(IPHARD+IHARD*2-1),
     1                                         RPROPS(IPKHAR+IHARD*2-1)
   10 CONTINUE
C Check dimension of RSTAVA and LALGVA
      IF(NRSTAV.GT.MRSTAV)CALL ERRPRT('ED0162')
      IF(NLALGV.GT.MLALGV)CALL ERRPRT('ED0163')
C
      RETURN
      END
```

```fortran
      SUBROUTINE RSQ4
     1(   IELPRP      ,NDATF       ,NRESF       ,RELPRP      ,UNSYM       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MGAUSP=4   ,MNODEG=2   ,NDIME=2    ,NDOFEL=8   ,NEDGEL=4   ,
     2    NGAUSB=1   ,NNODE=4    )
      LOGICAL UNSYM
      DIMENSION
     1    IELPRP(*)          ,RELPRP(*)
      DIMENSION
     1    NORDEB(NNODE,NEDGEL),POSGP(2,MGAUSP)   ,POSGPB(NGAUSB)     ,
     2    WEIGP(MGAUSP)      ,WEIGPB(NGAUSB)
C***********************************************************************
C READ INPUT DATA AND SET PROPERTIES FOR ELEMENT TYPE 'QUAD_4'
C (STANDARD ISOPARAMETRIC 4-NODED BI-LINEAR QUADRILATERAL)
C
C REFERENCE: Figure 4.5
C***********************************************************************
 1000 FORMAT(' QUAD_4 (standard 4-noded quadrilateral)'/
     1       ' Integration rule: ',I2,' gauss points')
C
C Read number of gauss points for domain integration
C ---------------------------------------------------
      READ(NDATF,*)NGAUSP
      WRITE(NRESF,1000)NGAUSP
      IF(NGAUSP.NE.1.AND.NGAUSP.NE.4)CALL ERRPRT('ED0033')
C Set element integer properties (stored in vector IELPRP)
C ---------------------------------------------------------
C total number of nodes and gauss points for domain integration
      IELPRP(3)=NNODE
      IELPRP(4)=NGAUSP
C number of degrees of freedom of the element
      IELPRP(5)=NDOFEL
C number of edges of the element
      IELPRP(6)=NEDGEL
C maximum number of nodes per edge
      IELPRP(7)=MNODEG
C number of gauss points for boundary integration
      IELPRP(8)=NGAUSB
C node numbering order on boundaries (set correspondance between local
C element node numbers and "edge" node numbering for boundary
C integration)
      NORDEB(1,1)=1
      NORDEB(2,1)=2
      NORDEB(3,1)=0
      NORDEB(4,1)=0
      NORDEB(1,2)=0
      NORDEB(2,2)=1
      NORDEB(3,2)=2
      NORDEB(4,2)=0
      NORDEB(1,3)=0
      NORDEB(2,3)=0
      NORDEB(3,3)=1
      NORDEB(4,3)=2
      NORDEB(1,4)=2
      NORDEB(2,4)=0
      NORDEB(3,4)=0
      NORDEB(4,4)=1
      IPOS=9
      DO 20 IEDGEL=1,NEDGEL
        DO 10 INODE=1,NNODE
          IELPRP(IPOS)=NORDEB(INODE,IEDGEL)
          IPOS=IPOS+1
   10   CONTINUE
   20 CONTINUE
C Set element real properties (stored in vector RELPRP)
C ------------------------------------------------------
C gaussian constants for domain integration
      CALL GAUS2D
     1(  'QUA'        ,NGAUSP     ,POSGP      ,WEIGP      )
      IPOS=1
      DO 30 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=POSGP(1,IGAUSP)
        RELPRP(IPOS+1)=POSGP(2,IGAUSP)
        IPOS=IPOS+NDIME
   30 CONTINUE
      IPOS=NGAUSP*NDIME+1
      DO 40 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=WEIGP(IGAUSP)
        IPOS=IPOS+1
   40 CONTINUE
C set matrix of coefficients for extrapolation from gauss points to
C nodes
      IPOS=NGAUSP*NDIME+NGAUSP+1
      CALL EXQ4
     1(   NGAUSP      ,RELPRP(IPOS))
C gaussian constants for boundary integration (intergration over edges)
      CALL GAUS1D
     1(   NGAUSB      ,POSGPB     ,WEIGPB     )
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+1
      DO 50 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=POSGPB(IGAUSB)
        IPOS=IPOS+1
   50 CONTINUE
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+NGAUSB+1
      DO 60 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=WEIGPB(IGAUSB)
        IPOS=IPOS+1
```

```
   60 CONTINUE
C Set unsymmetric solver flag
C ---------------------------
      UNSYM=.FALSE.
C
      RETURN
      END
```

```
      SUBROUTINE RSQ4FB
     1(   IELPRP     ,NDATF      ,NRESF      ,NTYPE      ,RELPRP     ,
     2    UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MGAUSP=4   ,MNODEG=2   ,NDIME=2    ,NDOFEL=8   ,NEDGEL=4   ,
     2    NGAUSB=1   ,NNODE=4    )
      LOGICAL UNSYM
      DIMENSION
     1    IELPRP(*)          ,RELPRP(*)
      DIMENSION
     1    NORDEB(NNODE,NEDGEL),POSGP(2,MGAUSP)   ,POSGPB(NGAUSB)     ,
     2    WEIGP(MGAUSP)      ,WEIGPB(NGAUSB)
      DATA R0   /
     1     0.0D0/
C***********************************************************************
C READ INPUT DATA AND SET PROPERTIES FOR ELEMENT TYPE 'QUAD_4_FBAR'
C (F-BAR 4-NODED BI-LINEAR QUADRILATERAL FOR PLANE STRAIN AND
C AXISYMMETRIC ANALYSIS)
C***********************************************************************
 1000 FORMAT(' QUAD_4_FBAR (F-Bar 4-noded quadrilateral)'/
     1       ' Integration rule: ',I2,' gauss points')
C
C Stops program if not plane strain neither axisymmetric
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)THEN
        CALL ERRPRT('ED0147')
      ENDIF
C
C Read number of gauss points for domain integration
C ---------------------------------------------------
      READ(NDATF,*)NGAUSP
      WRITE(NRESF,1000)NGAUSP
      IF(NGAUSP.NE.4)CALL ERRPRT('ED0099')
C Set element integer properties (stored in vector IELPRP)
C ---------------------------------------------------------
C total number of nodes and gauss points for domain integration
      IELPRP(3)=NNODE
      IELPRP(4)=NGAUSP
C number of degrees of freedom of the element
      IELPRP(5)=NDOFEL
C number of edges of the element
      IELPRP(6)=NEDGEL
C maximum number of nodes per edge
      IELPRP(7)=MNODEG
C number of gauss points for boundary integration
      IELPRP(8)=NGAUSB
C node numbering order on boundaries (set correspondance between local
C element node numbers and "edge" node numbering for boundary
C integration)
      NORDEB(1,1)=1
      NORDEB(2,1)=2
      NORDEB(3,1)=0
      NORDEB(4,1)=0
      NORDEB(1,2)=0
      NORDEB(2,2)=1
      NORDEB(3,2)=2
      NORDEB(4,2)=0
      NORDEB(1,3)=0
      NORDEB(2,3)=0
      NORDEB(3,3)=1
      NORDEB(4,3)=2
      NORDEB(1,4)=2
      NORDEB(2,4)=0
      NORDEB(3,4)=0
      NORDEB(4,4)=1
      IPOS=9
      DO 20 IEDGEL=1,NEDGEL
        DO 10 INODE=1,NNODE
          IELPRP(IPOS)=NORDEB(INODE,IEDGEL)
          IPOS=IPOS+1
   10   CONTINUE
   20 CONTINUE
C Set element real properties (stored in vector RELPRP)
C ------------------------------------------------------
C gaussian constants for domain integration
      CALL GAUS2D
     1(  'QUA'       ,NGAUSP     ,POSGP      ,WEIGP      )
      IPOS=1
      DO 30 IGAUSP=1,NGAUSP
        RELPRP(IPOS   )=POSGP(1,IGAUSP)
        RELPRP(IPOS+1)=POSGP(2,IGAUSP)
        IPOS=IPOS+NDIME
   30 CONTINUE
      IPOS=NGAUSP*NDIME+1
      DO 40 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=WEIGP(IGAUSP)
        IPOS=IPOS+1
   40 CONTINUE
C set matrix of coefficients for extrapolation from gauss points to
C nodes
      IPOS=NGAUSP*NDIME+NGAUSP+1
      CALL EXQ4FB
     1(   RELPRP(IPOS)   )
C gaussian constants for boundary integration (intergration over edges)
      CALL GAUS1D
     1(   NGAUSB     ,POSGPB     ,WEIGPB     )
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+1
      DO 50 IGAUSB=1,NGAUSB
```

```
        RELPRP(IPOS)=POSGPB(IGAUSB)
        IPOS=IPOS+1
   50 CONTINUE
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+NGAUSB+1
      DO 60 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=WEIGPB(IGAUSB)
        IPOS=IPOS+1
   60 CONTINUE
C set coordinates of the element centroid
      EXISC=R0
      ETASC=R0
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+2*NGAUSB+1
      RELPRP(IPOS  )=EXISC
      RELPRP(IPOS+1)=ETASC
C Set unsymmetric solver flag
C --------------------------
      UNSYM=.TRUE.
C
      RETURN
      END
```

```
      SUBROUTINE RSQ8
     1(   IELPRP     ,NDATF      ,NRESF      ,RELPRP     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MGAUSP=9   ,MNODEG=3   ,NDIME=2    ,NDOFEL=16  ,NEDGEL=4   ,
     2    NGAUSB=2   ,NNODE=8    )
      LOGICAL UNSYM
      DIMENSION
     1    IELPRP(*)          ,RELPRP(*)
      DIMENSION
     1    NORDEB(NNODE,NEDGEL),POSGP(2,MGAUSP)   ,POSGPB(NGAUSB)     ,
     2    WEIGP(MGAUSP)      ,WEIGPB(NGAUSB)
C***********************************************************************
C READ INPUT DATA AND SET PROPERTIES FOR ELEMENT TYPE 'QUAD_8'
C (STANDARD ISOPARAMETRIC 8-NODED QUADRILATERAL)
C
C REFERENCE: Section 4.1.3
C***********************************************************************
 1000 FORMAT(' QUAD_8 (standard 8-noded quadrilateral)'/
     1       ' Integration rule: ',I2,' gauss points')
C
C Read number of gauss points for domain integration
C --------------------------------------------------
      READ(NDATF,*)NGAUSP
      WRITE(NRESF,1000)NGAUSP
      IF(NGAUSP.NE.1.AND.NGAUSP.NE.4.AND.
     1   NGAUSP.NE.5.AND.NGAUSP.NE.9)THEN
        CALL ERRPRT('ED0008')
      ENDIF
C Set element integer properties (stored in vector IELPRP)
C --------------------------------------------------------
C total number of nodes and gauss points for domain integration
      IELPRP(3)=NNODE
      IELPRP(4)=NGAUSP
C number of degrees of freedom of the element
      IELPRP(5)=NDOFEL
C number of edges of the element
      IELPRP(6)=NEDGEL
C maximum number of nodes per edge
      IELPRP(7)=MNODEG
C number of gauss points for boundary integration
      IELPRP(8)=NGAUSB
C node numbering order on boundaries (set correspondance between local
C element node numbers and "edge" node numbering for boundary
C integration)
      NORDEB(1,1)=1
      NORDEB(2,1)=2
      NORDEB(3,1)=3
      NORDEB(4,1)=0
      NORDEB(5,1)=0
      NORDEB(6,1)=0
      NORDEB(7,1)=0
      NORDEB(8,1)=0
      NORDEB(1,2)=0
      NORDEB(2,2)=0
      NORDEB(3,2)=1
      NORDEB(4,2)=2
      NORDEB(5,2)=3
      NORDEB(6,2)=0
      NORDEB(7,2)=0
      NORDEB(8,2)=0
      NORDEB(1,3)=0
      NORDEB(2,3)=0
      NORDEB(3,3)=0
      NORDEB(4,3)=0
      NORDEB(5,3)=1
      NORDEB(6,3)=2
      NORDEB(7,3)=3
      NORDEB(8,3)=0
      NORDEB(1,4)=3
      NORDEB(2,4)=0
      NORDEB(3,4)=0
      NORDEB(4,4)=0
      NORDEB(5,4)=0
      NORDEB(6,4)=0
      NORDEB(7,4)=1
      NORDEB(8,4)=2
      IPOS=9
      DO 20 IEDGEL=1,NEDGEL
        DO 10 INODE=1,NNODE
          IELPRP(IPOS)=NORDEB(INODE,IEDGEL)
          IPOS=IPOS+1
   10   CONTINUE
   20 CONTINUE
C Set element real properties (stored in vector RELPRP)
C -----------------------------------------------------
C gaussian constants for domain integration
      CALL GAUS2D
     1(  'QUA'       ,NGAUSP     ,POSGP      ,WEIGP      )
      IPOS=1
      DO 30 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=POSGP(1,IGAUSP)
        RELPRP(IPOS+1)=POSGP(2,IGAUSP)
        IPOS=IPOS+NDIME
   30 CONTINUE
      IPOS=NGAUSP*NDIME+1
      DO 40 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=WEIGP(IGAUSP)
```

```
        IPOS=IPOS+1
   40 CONTINUE
C set matrix of coefficients for extrapolation from gauss points to
C nodes
      IPOS=NGAUSP*NDIME+NGAUSP+1
      CALL EXQ8
     1(   NGAUSP     ,RELPRP(IPOS))
C gaussian constants for boundary integration (intergration over edges)
      CALL GAUS1D
     1(   NGAUSB     ,POSGPB     ,WEIGPB     )
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+1
      DO 50 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=POSGPB(IGAUSB)
        IPOS=IPOS+1
   50 CONTINUE
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+NGAUSB+1
      DO 60 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=WEIGPB(IGAUSB)
        IPOS=IPOS+1
   60 CONTINUE
C Set unsymmetric solver flag
C ---------------------------
      UNSYM=.FALSE.
C
      RETURN
      END
```

```fortran
      SUBROUTINE RST3
     1(   IELPRP     ,NRESF      ,RELPRP     ,UNSYM      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MGAUSP=1   ,MNODEG=2   ,NDIME=2    ,NDOFEL=6   ,NEDGEL=3   ,
     2    NGAUSB=1   ,NNODE=3    )
      LOGICAL UNSYM
      DIMENSION
     1    IELPRP(*)          ,RELPRP(*)
      DIMENSION
     1    NORDEB(NNODE,NEDGEL),POSGP(2,MGAUSP)   ,POSGPB(NGAUSB)     ,
     2    WEIGP(MGAUSP)       ,WEIGPB(NGAUSB)
C***********************************************************************
C READ INPUT DATA AND SET PROPERTIES FOR ELEMENT TYPE 'TRI_3'
C (STANDARD ISOPARAMETRIC 3-NODED LINEAR TRIANGLE)
C
C REFERENCE: Figure 4.4
C***********************************************************************
 1000 FORMAT(' TRI_3 (standard 3-noded quadrilateral)'/
     1       ' with 1 gauss point')
      WRITE(NRESF,1000)
C
C Set number of gauss points for domain integration
C --------------------------------------------------
      NGAUSP=1
C Set element integer properties (stored in vector IELPRP)
C ---------------------------------------------------------
C total number of nodes and gauss points for domain integration
      IELPRP(3)=NNODE
      IELPRP(4)=NGAUSP
C number of degrees of freedom of the element
      IELPRP(5)=NDOFEL
C number of edges of the element
      IELPRP(6)=NEDGEL
C maximum number of nodes per edge
      IELPRP(7)=MNODEG
C number of gauss points for boundary integration
      IELPRP(8)=NGAUSB
C node numbering order on boundaries (set correspondance between local
C element node numbers and "edge" node numbering for boundary
C integration)
      NORDEB(1,1)=1
      NORDEB(2,1)=2
      NORDEB(3,1)=0
      NORDEB(1,2)=0
      NORDEB(2,2)=1
      NORDEB(3,2)=2
      NORDEB(1,3)=2
      NORDEB(2,3)=0
      NORDEB(3,3)=1
      IPOS=9
      DO 20 IEDGEL=1,NEDGEL
        DO 10 INODE=1,NNODE
          IELPRP(IPOS)=NORDEB(INODE,IEDGEL)
          IPOS=IPOS+1
   10   CONTINUE
   20 CONTINUE
C Set element real properties (stored in vector RELPRP)
C -------------------------------------------------------
C gaussian constants for domain integration
      CALL GAUS2D
     1(  'TRI'       ,NGAUSP     ,POSGP      ,WEIGP      )
      IPOS=1
      DO 30 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=POSGP(1,IGAUSP)
        RELPRP(IPOS+1)=POSGP(2,IGAUSP)
        IPOS=IPOS+NDIME
   30 CONTINUE
      IPOS=NGAUSP*NDIME+1
      DO 40 IGAUSP=1,NGAUSP
        RELPRP(IPOS)=WEIGP(IGAUSP)
        IPOS=IPOS+1
   40 CONTINUE
C set matrix of coefficients for extrapolation from gauss points to
C nodes
      IPOS=NGAUSP*NDIME+NGAUSP+1
      CALL EXT3
     1(   RELPRP(IPOS)    )
C gaussian constants for boundary integration (intergration over edges)
      CALL GAUS1D
     1(   NGAUSB     ,POSGPB     ,WEIGPB     )
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+1
      DO 50 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=POSGPB(IGAUSB)
        IPOS=IPOS+1
   50 CONTINUE
      IPOS=NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+NGAUSB+1
      DO 60 IGAUSB=1,NGAUSB
        RELPRP(IPOS)=WEIGPB(IGAUSB)
        IPOS=IPOS+1
   60 CONTINUE
C Set unsymmetric solver flag
C ---------------------------
      UNSYM=.FALSE.
C
      RETURN
      END
```

```fortran
      SUBROUTINE RSTART
     1(  DFOLD      ,DLENG      ,DLENGO     ,DLENM      ,DLAMD      ,
     2   IFNEG      ,IINCS      ,MXFRON     ,NOUTP      ,TFACT      ,
     3   TFACTO     ,UNSYM      ,RSTINP     ,RSTOUT     ,MODE       ,
     4   INCRST     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      DIMENSION NOUTP(5)
      LOGICAL UNSYM
      CHARACTER*256 RSTINP,RSTOUT
C***********************************************************************
C WRITE DATA TO OUTPUT RE-START FILE AND READ DATA FROM INPUT RE-START
C FILE
C
C REFERENCE: Figures 5.1-3
C***********************************************************************
 1000 FORMAT(//15X,
     1       'Writing converged results to re-start file ...'//)
 1010 FORMAT(////15X,
     1       'Reading data from input re-start file ...')
C
C
      IF(MODE.EQ.1)THEN
C
C Writing (output) mode
C =====================
C
C Check re-start output flag
C
        IF(NOUTP(5).EQ.0)GOTO 999
        IF(NOUTP(5).NE.0.AND.NALGO.LT.0)THEN
          IF(MOD(IINCS,NOUTP(5)).NE.0)GOTO 999
        ENDIF
C
        INCRST=IINCS
        WRITE(*,1000)
        WRITE(16,1000)
        OPEN(UNIT=17,FILE=RSTOUT,STATUS='UNKNOWN',FORM='UNFORMATTED')
C
C Write some global variables first
C ---------------------------------
        WRITE(17)DFOLD,DLENG,DLENGO,DLENM,DLAMD,IFNEG,IINCS,MXFRON,
     1           TFACT,TFACTO,UNSYM
C
C Then write all common blocks
C ----------------------------
C COMMON/CONTRL/
        WRITE(17)
     1    NDOFN      ,NELEM      ,NGRUP      ,NPOIN      ,NTOTV      ,
     2    NVFIX      ,NTYPE      ,NALGO      ,NARCL      ,NDIME      ,
     3    NLARGE     ,NAXIS
C COMMON/CORE  /
        WRITE(17)
     1    FIXED                       ,STFOR                       ,
     2    TOFOR                       ,ELOAD                       ,
     3    ELOADO                      ,RLOAD
C COMMON/MATERL/
        WRITE(17)
     1    RPROPS              ,IPROPS
C COMMON/MESH  /
        WRITE(17)
     1    ANGLE              ,COORD              ,PRESC              ,
     2    IELTID             ,IFFIX              ,IGRPID             ,
     3    LNODS              ,MASTER             ,MATTID             ,
     4    NOFIX              ,NVALEN
C COMMON/ELEMEN/
        WRITE(17)
     1    RELPRP              ,IELPRP
C COMMON/RESULT/
        WRITE(17)
     1    DITER              ,DINCR              ,DINCRO             ,
     2    DTANG              ,TDISP              ,TDISPO             ,
     3    TREAC
C COMMON/STATE /
        WRITE(17)
     1    RALGVA                      ,RSTAVA                      ,
     2    STRSG                       ,THKGP                       ,
     3    LALGVA
C
C
      ELSEIF(MODE.EQ.0)THEN
C
C Reading (input) mode
C ====================
C
        WRITE(*,1010)
        OPEN(UNIT=17,FILE=RSTINP,STATUS='OLD',FORM='UNFORMATTED')
C
C Read some global variables first
C --------------------------------
        READ(17)DFOLD,DLENG,DLENGO,DLENM,DLAMD,IFNEG,IINCS,MXFRON,
     1          TFACT,TFACTO,UNSYM
C
```

```
C Then read all common blocks
C ---------------------------
C COMMON/CONTRL/
        READ(17)
     1    NDOFN      ,NELEM      ,NGRUP      ,NPOIN      ,NTOTV      ,
     2    NVFIX      ,NTYPE      ,NALGO      ,NARCL      ,NDIME      ,
     3    NLARGE     ,NAXIS
C COMMON/CORE  /
        READ(17)
     1    FIXED                       ,STFOR                       ,
     2    TOFOR                       ,ELOAD                       ,
     3    ELOADO                      ,RLOAD
C COMMON/MATERL/
        READ(17)
     1    RPROPS              ,IPROPS
C COMMON/MESH  /
        READ(17)
     1    ANGLE              ,COORD              ,PRESC              ,
     2    IELTID             ,IFFIX              ,IGRPID             ,
     3    LNODS              ,MASTER             ,MATTID             ,
     4    NOFIX              ,NVALEN
C COMMON/ELEMEN/
        READ(17)
     1    RELPRP              ,IELPRP
C COMMON/RESULT/
        READ(17)
     1    DITER              ,DINCR              ,DINCRO             ,
     2    DTANG              ,TDISP              ,TDISPO             ,
     3    TREAC
C COMMON/STATE /
        READ(17)
     1    RALGVA                      ,RSTAVA                      ,
     2    STRSG                       ,THKGP                       ,
     3    LALGVA
C
      ENDIF
C
      CLOSE(UNIT=17,STATUS='KEEP')
C
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RSTCHK(   RSTINP     ,RSTRT      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL RSTRT
      CHARACTER*256 RSTINP
C
      LOGICAL AVAIL,FOUND
      CHARACTER*80 INLINE
      DIMENSION IWBEG(40),IWEND(40)
C***********************************************************************
C CHECKS WETHER MAIN DATA IS TO BE READ FROM INPUT RE-START FILE
C AND SET INPUT RE-START FILE NAME IF REQUIRED
C***********************************************************************
 1000 FORMAT(////,
     1' Main input data read from re-start file'/
     2' ========================================'///
     3'           Input re-start file name ------> ',A)
C
C Checks whether the input data file contains the keyword RESTART
C
        CALL FNDKEY
     1(   FOUND     ,IWBEG    ,IWEND    ,'RESTART',
     2    INLINE    ,15       ,NWRD     )
        IF(FOUND)THEN
C sets re-start flag and name of input re-start file
          RSTRT=.TRUE.
          RSTINP=INLINE(IWBEG(2):IWEND(2))//'.rst'
          WRITE(16,1000)INLINE(IWBEG(2):IWEND(2))//'.rst'
C checks existence of the input re-start file
          INQUIRE(FILE=RSTINP,EXIST=AVAIL)
          IF(.NOT.AVAIL)CALL ERRPRT('ED0096')
        ELSE
          RSTRT=.FALSE.
        ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE RTSR
     1(   AUXM        ,MODE        ,MROWQ       ,MROWR       ,NCOLR       ,
     2    NROWR       ,Q           ,R           ,S           ,SCAL        ,
     3    UNSYM       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL UNSYM
      DIMENSION
     1    AUXM(NCOLR,NROWR)  ,Q(MROWQ,MROWQ)     ,R(MROWR,NCOLR)     ,
     2    S(MROWR,MROWR)
      DATA  R0   /0.0D0/
C***********************************************************************
C PERFORMS THE MATRIX PRODUCTS
C                                  T
C                     Q := SCAL * R  S R        (IF MODE=1)
C OR
C                                      T
C                     Q := Q + SCAL * R  S R    (OTHERWISE)
C
C WHERE 'R' IS A REAL RECTANGULAR MATRIX, 'S' A REAL SQUARE MATRIX
C AND 'SCAL' A SCALAR.
C***********************************************************************
      CALL RVZERO(AUXM,NCOLR*NROWR)
      DO 30 I=1,NCOLR
        DO 20 K=1,NROWR
          IF(R(K,I).NE.R0)THEN
            DO 10 J=1,NROWR
              AUXM(I,J)=AUXM(I,J)+SCAL*R(K,I)*S(K,J)
   10       CONTINUE
          ENDIF
   20   CONTINUE
   30 CONTINUE
C
      IF(MODE.EQ.1)THEN
        DO 50 I=1,NCOLR
          DO 40 J=1,NCOLR
            Q(I,J)=R0
   40     CONTINUE
   50   CONTINUE
      ENDIF
C
      IF(UNSYM)THEN
C Construct the whole matrix Q at once
        DO 80 J=1,NCOLR
          DO 70 K=1,NROWR
            IF(R(K,J).NE.R0)THEN
              DO 60 I=1,NCOLR
                Q(I,J)=Q(I,J)+AUXM(I,K)*R(K,J)
   60         CONTINUE
            ENDIF
   70     CONTINUE
   80   CONTINUE
      ELSE
C Construct the lower triangle of Q first
        DO 110 J=1,NCOLR
          DO 100 K=1,NROWR
            IF(R(K,J).NE.R0)THEN
              DO 90 I=J,NCOLR
                Q(I,J)=Q(I,J)+AUXM(I,K)*R(K,J)
   90         CONTINUE
            ENDIF
  100     CONTINUE
  110   CONTINUE
C and then assemble the upper triangle
        DO 130 I=1,NCOLR
          DO 120 J=I+1,NCOLR
            Q(I,J)=Q(J,I)
  120     CONTINUE
  130   CONTINUE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE RTSX
     1(   AUXM       ,MODE       ,MROWQ      ,MROWR      ,NCOLR      ,
     2    NROWR      ,Q          ,R          ,S          ,X          ,
     3    SCAL       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    AUXM(NCOLR,NROWR)  ,Q(MROWQ,MROWQ)     ,R(MROWR,NCOLR)     ,
     2    S(MROWR,MROWR)     ,X(MROWR,NCOLR)
      DATA  R0   /0.0D0/
C***********************************************************************
C PERFORMS THE MATRIX PRODUCTS
C                                   T
C                     Q := SCAL * R  S X        (IF MODE=1)
C OR
C                                       T
C                     Q := Q + SCAL * R  S X    (OTHERWISE)
C
C WHERE 'R' AND 'X' ARE REAL RECTANGULAR MATRICES OF IDENTICAL
C DIMENSIONS, 'S' A REAL SQUARE MATRIX AND 'SCAL' A SCALAR.
C***********************************************************************
      CALL RVZERO(AUXM,NCOLR*NROWR)
      DO 30 I=1,NCOLR
        DO 20 K=1,NROWR
          IF(R(K,I).NE.R0)THEN
            DO 10 J=1,NROWR
              AUXM(I,J)=AUXM(I,J)+SCAL*R(K,I)*S(K,J)
   10       CONTINUE
          ENDIF
   20   CONTINUE
   30 CONTINUE
C
      IF(MODE.EQ.1)THEN
        DO 50 I=1,NCOLR
          DO 40 J=1,NCOLR
            Q(I,J)=R0
   40     CONTINUE
   50   CONTINUE
      ENDIF
C
C Construct the matrix Q
      DO 80 J=1,NCOLR
        DO 70 K=1,NROWR
          IF(X(K,J).NE.R0)THEN
            DO 60 I=1,NCOLR
              Q(I,J)=Q(I,J)+AUXM(I,K)*X(K,J)
   60       CONTINUE
          ENDIF
   70   CONTINUE
   80 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE RTV
     1(   MODE        ,MROWR      ,NCOLR      ,NROWR      ,P          ,
     2    R           ,V          ,SCAL       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    P(NCOLR)           ,R(MROWR,NCOLR)     ,V(NROWR)
      DATA  R0   /0.0D0/
C***********************************************************************
C PERFORMS THE PRODUCT
C                                    T
C                     P := SCAL * R  V          (IF MODE=1)
C OR
C                                        T
C                     P := P + SCAL * R  V      (OTHERWISE)
C
C WHERE 'R' IS A REAL RECTANGULAR MATRIX, 'V' A REAL VECTOR AND
C 'SCAL' A SCALAR.
C***********************************************************************
      IF(MODE.EQ.1)CALL RVZERO(P,NCOLR)
      DO 30 I=1,NCOLR
        DO 20 J=1,NROWR
          IF(R(J,I).NE.R0)THEN
            P(I)=P(I)+SCAL*R(J,I)*V(J)
          ENDIF
   20   CONTINUE
   30 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE RVSCAL
     1(   V          ,N          ,SCAL       )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION V(N)
C***********************************************************************
C MULTIPLIES THE DOUBLE PRECISION VECTOR 'V', OF DIMENSION 'N',
C BY THE SCALAR 'SCAL'
C***********************************************************************
      DO 10 I=1,N
        V(I)=SCAL*V(I)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RVSUB
     1(   U          ,V           ,W           ,N           )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    U(N)                ,V(N)                ,W(N)
C***********************************************************************
C SUBTRACTS THE VECTOR 'W' FROM THE VECTOR 'V' AND STORE THE RESULT
C IN 'U'. U ,V AND W ARE DOUBLE PRECISION VECTORS OF DIMENSION N.
C***********************************************************************
      DO 10 I=1,N
        U(I)=V(I)-W(I)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RVZERO
     1(   V          ,N          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION V(N)
      DATA R0/0.0D0/
C***********************************************************************
C INITIALISES TO ZERO A DOUBLE PRECISION ARRAY OF DIMENSION N
C***********************************************************************
      DO 10 I=1,N
        V(I)=R0
   10 CONTINUE
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION SCAPRD(U  ,V  ,N  )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION U(N), V(N)
      DATA  R0  / 0.0D0 /
C***********************************************************************
C SCALAR PRODUCT OF DOUBLE PRECISION VECTORS U AND V OF DIMENSION N
C***********************************************************************
      SCAPRD=R0
      DO 10 I=1,N
        SCAPRD=SCAPRD+U(I)*V(I)
  10  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SETBE
     1(   BE         ,NTYPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      EXTERNAL  EXP2X
      LOGICAL   OUTOFP
      DIMENSION BE(*)
      DATA  RP5  /0.5D0/
C***********************************************************************
C COMPUTES THE ELASTIC CAUCHY-GREEN TENSOR AS A FUNCTION OF
C THE ELASTIC LOGARITHMIC STRAIN TENSOR:
C
C                        Be   :=  exp[ 2 Ee  ]
C
C REFERENCE: Box 14.3, item (ii)
C***********************************************************************
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        OUTOFP=.TRUE.
      ELSEIF(NTYPE.EQ.1)THEN
        OUTOFP=.TRUE.
      ELSE
        CALL ERRPRT('EI0024')
      ENDIF
C Convert engineering elastic strain components into physical components
      BE(3)=RP5*BE(3)
C Use isotropic tensor function to compute elastic Cauchy-Green tensor
      CALL ISO2
     1(    EXP2X      ,OUTOFP     ,BE         ,BE         )
C
      RETURN
      END
```

```
      SUBROUTINE SFQ4
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    DERIV(MDIME,*)     ,SHAPE(*)
      DATA RP25 ,RP5  ,R1   /0.25D0,0.5D0,1.0D0/
C***********************************************************************
C COMPUTES SHAPE FUNCTIONS AND SHAPE FUNCTION DERIVATIVES FOR
C ELEMENT 'QUAD_4':
C                          4         3
C                           o-------o
C                           |       |     STANDARD ISOPARAMETRIC
C                           |       |     BI-LINEAR 4-NODE QUADRILATERAL
C                           |       |
C                           o-------o
C                          1         2
C
C REFERENCE: Expression (4.42)
C***********************************************************************
      IF(IBOUND.EQ.0)THEN
C Shape functions and derivatives on element DOMAIN
C -------------------------------------------------
        S=EXISP
        T=ETASP
        ST=S*T
C Shape functions
        SHAPE(1)=(R1-T-S+ST)*RP25
        SHAPE(2)=(R1-T+S-ST)*RP25
        SHAPE(3)=(R1+T+S+ST)*RP25
        SHAPE(4)=(R1+T-S-ST)*RP25
C Shape function derivatives
        DERIV(1,1)=(-R1+T)*RP25
        DERIV(1,2)=(+R1-T)*RP25
        DERIV(1,3)=(+R1+T)*RP25
        DERIV(1,4)=(-R1-T)*RP25
        DERIV(2,1)=(-R1+S)*RP25
        DERIV(2,2)=(-R1-S)*RP25
        DERIV(2,3)=(+R1+S)*RP25
        DERIV(2,4)=(+R1-S)*RP25
      ELSE
C Shape function and derivatives on element BOUNDARY (1-D)
C --------------------------------------------------------
        S=EXISP
C Shape functions
        SHAPE(1)=(-S+R1)*RP5
        SHAPE(2)=(+S+R1)*RP5
C Shape functions derivatives
        DERIV(1,1)=-RP5
        DERIV(1,2)=RP5
C
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SFQ4FB
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    DERIV(MDIME,*)     ,SHAPE(*)
      DATA RP25 ,RP5  ,R1   /0.25D0,0.5D0,1.0D0/
C***********************************************************************
C COMPUTES SHAPE FUNCTIONS AND SHAPE FUNCTION DERIVATIVES FOR
C ELEMENT 'QUAD_4_FBAR':
C                         4         3
C                          o-------o
C                          |    Fo |     F-BAR BI-LINEAR
C                          |   x   |     4 NODE QUADRILATERAL
C                          |       |
C                          o-------o
C                         1         2
C
C
C REFERENCE: Expression (4.42)
C***********************************************************************
      IF(IBOUND.EQ.0)THEN
C Shape functions and derivatives on element DOMAIN
C -------------------------------------------------
        S=EXISP
        T=ETASP
        ST=S*T
C Shape functions
        SHAPE(1)=(R1-T-S+ST)*RP25
        SHAPE(2)=(R1-T+S-ST)*RP25
        SHAPE(3)=(R1+T+S+ST)*RP25
        SHAPE(4)=(R1+T-S-ST)*RP25
C Shape function derivatives
        DERIV(1,1)=(-R1+T)*RP25
        DERIV(1,2)=(+R1-T)*RP25
        DERIV(1,3)=(+R1+T)*RP25
        DERIV(1,4)=(-R1-T)*RP25
        DERIV(2,1)=(-R1+S)*RP25
        DERIV(2,2)=(-R1-S)*RP25
        DERIV(2,3)=(+R1+S)*RP25
        DERIV(2,4)=(+R1-S)*RP25
      ELSE
C Shape function and derivatives on element BOUNDARY (1-D)
C --------------------------------------------------------
        S=EXISP
C Shape functions
        SHAPE(1)=(-S+R1)*RP5
        SHAPE(2)=(+S+R1)*RP5
C Shape functions derivatives
        DERIV(1,1)=-RP5
        DERIV(1,2)=RP5
C
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SFQ8
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    DERIV(MDIME,*)     ,SHAPE(*)
      DATA RP25  ,RP5  ,R1   ,R2   /
     1     0.25D0,0.5D0,1.0D0,2.0D0/
C***********************************************************************
C COMPUTES SHAPE FUNCTIONS AND SHAPE FUNCTION DERIVATIVES FOR
C ELEMENT 'QUAD_8' (STANDARD ISOPARAMETRIC 8-NODED QUADRILATERAL)
C
C REFERENCE: Section 4.1.3
C***********************************************************************
      IF(IBOUND.EQ.0)THEN
C Shape functions and derivatives on element DOMAIN
C ---------------------------------------------------
        S=EXISP
        T=ETASP
        S2=S*R2
        T2=T*R2
        SS=S*S
        TT=T*T
        ST=S*T
        SST=S*S*T
        STT=S*T*T
        ST2=S*T*R2
C Shape functions
        SHAPE(1)=(-R1+ST+SS+TT-SST-STT)*RP25
        SHAPE(2)=(R1-T-SS+SST)*RP5
        SHAPE(3)=(-R1-ST+SS+TT-SST+STT)*RP25
        SHAPE(4)=(R1+S-TT-STT)*RP5
        SHAPE(5)=(-R1+ST+SS+TT+SST+STT)*RP25
        SHAPE(6)=(R1+T-SS-SST)*RP5
        SHAPE(7)=(-R1-ST+SS+TT+SST-STT)*RP25
        SHAPE(8)=(R1-S-TT+STT)*RP5
C Shape functions derivatives
        DERIV(1,1)=(T+S2-ST2-TT)*RP25
        DERIV(1,2)=-S+ST
        DERIV(1,3)=(-T+S2-ST2+TT)*RP25
        DERIV(1,4)=(R1-TT)*RP5
        DERIV(1,5)=(T+S2+ST2+TT)*RP25
        DERIV(1,6)=-S-ST
        DERIV(1,7)=(-T+S2+ST2-TT)*RP25
        DERIV(1,8)=(-R1+TT)*RP5
        DERIV(2,1)=(S+T2-SS-ST2)*RP25
        DERIV(2,2)=(-R1+SS)*RP5
        DERIV(2,3)=(-S+T2-SS+ST2)*RP25
        DERIV(2,4)=-T-ST
        DERIV(2,5)=(S+T2+SS+ST2)*RP25
        DERIV(2,6)=(R1-SS)*RP5
        DERIV(2,7)=(-S+T2+SS-ST2)*RP25
        DERIV(2,8)=-T+ST
      ELSE
C Shape function and derivatives on element BOUNDARY (1-D)
C ----------------------------------------------------------
        S=EXISP
        SS=S*S
        S2=S*R2
C Shape functions
        SHAPE(1)=(-S+SS)*RP5
        SHAPE(2)=R1-SS
        SHAPE(3)=(S+SS)*RP5
C Shape functions derivatives
        DERIV(1,1)=(-R1+S2)*RP5
        DERIV(1,2)=-S2
        DERIV(1,3)=(R1+S2)*RP5
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SFT3
     1(   DERIV     ,ETASP     ,EXISP     ,IBOUND    ,MDIME     ,
     2    SHAPE     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION
     1    DERIV(MDIME,*)     ,SHAPE(*)
      DATA R0   ,RP5  ,R1   /0.0D0,0.5D0,1.0D0/
C***********************************************************************
C COMPUTES SHAPE FUNCTIONS AND SHAPE FUNCTION DERIVATIVES FOR
C ELEMENT 'TRI_3':
C                          3 o
C                            |\
C                            | \
C                            |  \
C                            |   \           STANDARD LINEAR
C                            |    \          3 NODE TRIANGLE
C                            |     \
C                            o------o
C                           1       2
C
C REFERENCE: Expression (4.38)
C***********************************************************************
      IF(IBOUND.EQ.0)THEN
C Shape functions and derivatives on element DOMAIN
C -------------------------------------------------
        S=EXISP
        T=ETASP
C Shape functions
        SHAPE(1)=R1-S-T
        SHAPE(2)=S
        SHAPE(3)=T
C Shape function derivatives
        DERIV(1,1)=-R1
        DERIV(1,2)=+R1
        DERIV(1,3)=+R0
        DERIV(2,1)=-R1
        DERIV(2,2)=+R0
        DERIV(2,3)=+R1
      ELSE
C Shape function and derivatives on element BOUNDARY (1-D)
C --------------------------------------------------------
        S=EXISP
C Shape functions
        SHAPE(1)=(-S+R1)*RP5
        SHAPE(2)=(+S+R1)*RP5
C Shape functions derivatives
        DERIV(1,1)=-RP5
        DERIV(1,2)=RP5
C
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,IELTYP     ,
     2    MDIME      ,SHAPE      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INCLUDE '../ELEMENTS.INC'
      DIMENSION
     1    DERIV(MDIME,*)     ,SHAPE(*)
C***********************************************************************
C CALL SPECIFIC ROUTINES FOR COMPUTATION OF SHAPE FUNCTIONS AND
C SHAPE FUNCTION DERIVATIVES FOR EACH TYPE OF ELEMENT
C
C REFERENCE: Section 5.6.3
C***********************************************************************
      IF(IELTYP.EQ.TRI3)THEN
        CALL SFT3
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      ELSEIF(IELTYP.EQ.QUAD4)THEN
        CALL SFQ4
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      ELSEIF(IELTYP.EQ.QUAD8)THEN
        CALL SFQ8
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      ELSEIF(IELTYP.EQ.QUA4FB)THEN
        CALL SFQ4FB
     1(   DERIV      ,ETASP      ,EXISP      ,IBOUND     ,MDIME      ,
     2    SHAPE      )
      ELSE
        CALL ERRPRT('EI0005')
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SOLQUA
     1(   A          ,B          ,C          ,ONEROO     ,TWOROO     ,
     2    ROOT1      ,ROOT2      )
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL  ONEROO ,TWOROO
      DATA
     1    R0   ,R1   ,R2   ,R4   ,SMALL /
     2    0.D0 ,1.0D0,2.0D0,4.0D0,1.D-12/
C***********************************************************************
C FINDS THE REAL ROOTS OF A QUADRATIC EQUATION:  A X**2 + B X + C = 0.
C
C REFERENCE:
C W.H.Press, S.A.Teukolsky, W.T.Vetterling and B.P.Flannery. Numerical
C recipes in FORTRAN. The art of scientific computing. 2nd Ed.,
C Cambridge Univ. Press, 1992. (Section 5.6)
C***********************************************************************
C Initialises logical flags
      ONEROO=.FALSE.
      TWOROO=.FALSE.
      IF(A.NE.R0)THEN
C The equation is non-linear in fact
C ----------------------------------
        IF(B.NE.R0)THEN
          SIGNB=B/ABS(B)
        ELSE
          SIGNB=R1
        ENDIF
        B2=B*B
        R4AC=R4*A*C
        SQUAR=B2-R4AC
        IF(SQUAR.GT.R0)THEN
C there are two distinct real roots: uses formula which minimises
C round-off errors when the coefficients A and/or C are small
          TWOROO=.TRUE.
          SQUAR=SQRT(SQUAR)
          Q=-(B+SIGNB*SQUAR)/R2
          ROOT1=Q/A
          ROOT2=C/Q
        ELSEIF(SQUAR.EQ.R0.OR.
     1         (SQUAR/DMAX1(B2,ABS(R4AC))+SMALL).GE.R0)THEN
C there is only one root
          ONEROO=.TRUE.
          ROOT1=-B/(R2*A)
        ENDIF
      ELSE
C The equation is linear
C ---------------------
        IF(B.NE.R0)THEN
C and well defined -> (only) one root exists
          ONEROO=.TRUE.
          ROOT1=-C/B
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SPDEC2
     1(   EIGPRJ     ,EIGX       ,REPEAT     ,X          )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   MCOMP=4    ,NDIM=2     )
      LOGICAL REPEAT
      DIMENSION
     1    EIGPRJ(MCOMP,NDIM)        ,EIGX(NDIM)                ,
     2    X(MCOMP)
      DIMENSION
     1    AUXMTX(NDIM,NDIM)         ,EIGVEC(NDIM,NDIM)
      DATA
     1    R0   ,RP5  ,R1   ,R4   ,SMALL  /
     2    0.0D0,0.5D0,1.0D0,4.0D0,1.D-5  /
C***********************************************************************
C PERFORMS THE CLOSED FORM SPECTRAL DECOMPOSITION OF A
C SYMMETRIC 2-D TENSOR STORED IN VECTOR FORM
C
C REFERENCE: Box A.2
C***********************************************************************
      REPEAT=.FALSE.
C Compute eigenvalues of X
C ------------------------
      TRX=X(1)+X(2)
      B=SQRT((X(1)-X(2))**2+R4*X(3)*X(3))
      EIGX(1)=RP5*(TRX+B)
      EIGX(2)=RP5*(TRX-B)
C Compute eigenprojection tensors
C -------------------------------
      DIFFER=ABS(EIGX(1)-EIGX(2))
      AMXEIG=DMAX1(ABS(EIGX(1)),ABS(EIGX(2)))
      IF(AMXEIG.NE.R0)DIFFER=DIFFER/AMXEIG
      IF(DIFFER.LT.SMALL)THEN
        REPEAT=.TRUE.
C for repeated (or nearly repeated) eigenvalues, re-compute eigenvalues
C and compute eigenvectors using the iterative procedure. In such cases,
C the closed formula for the eigenvectors is singular (or dominated by
C round-off errors)
        AUXMTX(1,1)=X(1)
        AUXMTX(2,2)=X(2)
        AUXMTX(1,2)=X(3)
        AUXMTX(2,1)=AUXMTX(1,2)
        CALL JACOB(AUXMTX,EIGX,EIGVEC,2)
        DO 10 IDIR=1,2
          EIGPRJ(1,IDIR)=EIGVEC(1,IDIR)*EIGVEC(1,IDIR)
          EIGPRJ(2,IDIR)=EIGVEC(2,IDIR)*EIGVEC(2,IDIR)
          EIGPRJ(3,IDIR)=EIGVEC(1,IDIR)*EIGVEC(2,IDIR)
          EIGPRJ(4,IDIR)=R0
 10     CONTINUE
      ELSE
C Use closed formula to compute eigenprojection tensors
        DO 20 IDIR=1,2
          B=EIGX(IDIR)-TRX
          C=R1/(EIGX(IDIR)+B)
          EIGPRJ(1,IDIR)=C*(X(1)+B)
          EIGPRJ(2,IDIR)=C*(X(2)+B)
          EIGPRJ(3,IDIR)=C*X(3)
          EIGPRJ(4,IDIR)=R0
 20     CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE STFBA2
     1(   IELEM      ,KUNLD      ,MDIME      ,MELEM      ,
     2    MPOIN      ,MSTRE      ,MTOTV      ,NAXIS      ,
     3    NTYPE      ,UNSYM      ,
     4    COORD1     ,DINCR      ,ESTIF      ,IELPRP     ,IPROPS     ,
     5    LALGVA     ,LNODS      ,RALGVA     ,RELPRP     ,RPROPS     ,
     6    RSTAVA     ,RSTAV2     ,STRSG      ,TDISP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
C
      PARAMETER( MGDIM=5 ,MBDIM=4 ,NDIME=2 ,NDOFN=2 )
C Arguments
      LOGICAL  LALGVA ,UNSYM
      DIMENSION
     1    COORD1(MDIME,MPOIN),DINCR(MTOTV)       ,ESTIF(MEVAB,MEVAB) ,
     2    IELPRP(*)          ,IPROPS(*)         ,LALGVA(MLALGV,MTOTG),
     3    LNODS(MELEM,MEVAB) ,RALGVA(MRALGV,MTOTG),RELPRP(*)          ,
     4    RPROPS(*)        ,RSTAVA(MRSTAV,MTOTG),RSTAV2(MRSTAV,MTOTG),
     5    STRSG(MSTRE,MTOTG) ,TDISP(MTOTV)
C Local arrays and variables
      DIMENSION
     1    AUXM(MEVAB,MGDIM)  ,AMATX(MGDIM,MGDIM) ,
     2    CARTD(NDIME,MNODE) ,DELDIS(2,MNODE)    ,DERIV(NDIME,MNODE) ,
     3    DMATX(MBDIM,MBDIM) ,EINCR(MBDIM)       ,ELCOD(NDIME,MNODE) ,
     4    FINCIN(3,3)        ,FINCR(3,3)         ,FINV(3,3)          ,
     5    GMATX(MGDIM,MEVAB) ,G0MATX(MGDIM,MEVAB),G0MGMX(MGDIM,MEVAB),
     6    GPCOD(NDIME)       ,QMATX(MGDIM,MGDIM) ,SHAPE(MNODE)       ,
     7    TELDIS(2,MNODE)
      DATA R0   ,RP5  ,R1   ,R3   ,R8   /
     1     0.0D0,0.5D0,1.0D0,3.0D0,8.0D0/
C***********************************************************************
C EVALUATES THE ELEMENT TANGENT STIFFNESS MATRIX FOR ELEMENTS OF CLASS
C 'FBAR' (F-bar ELEMENTS) IN 2-D (PLANE STRAIN AND AXISYMMETRIC
C PROBLEMS)
C
C REFERENCE: Box 15.2
C            Section 15.1.3
C***********************************************************************
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)THEN
C F-bar implementation valid only for plane strain and axisymmetric
        CALL ERRPRT('EI0034')
      ENDIF
      IF(NTYPE.EQ.3)TWOPI=R8*ATAN(R1)
      R1D3=R1/R3
C Identify element type
      IELTYP=IELPRP(1)
C Recover element information
      NNODE =IELPRP(3)
      NGAUSP=IELPRP(4)
      NEVAB =IELPRP(5)
C
C Set element nodal coordinates, total and incremental displacements
C vectors
C
      DO 20 INODE =1,NNODE
        LNODE=IABS(LNODS(IELEM,INODE))
        NPOSN=(LNODE-1)*NDOFN
        DO 10 IDOFN=1,NDOFN
          NPOSN=NPOSN+1
          ELCOD(IDOFN,INODE)=COORD1(IDOFN,LNODE)
          TELDIS(IDOFN,INODE)=-TDISP(NPOSN)
          DELDIS(IDOFN,INODE)=-DINCR(NPOSN)
   10   CONTINUE
   20 CONTINUE
C
C Initialize the element stiffness matrix
C
      DO 40 IEVAB=1,NEVAB
        DO 30 JEVAB=1,NEVAB
          ESTIF(IEVAB,JEVAB)=R0
   30   CONTINUE
   40 CONTINUE
C
C
C Evaluate inverse of the incremental deformation gradient at the
C element centroid for F-bar element
      NGAUSB=IELPRP(8)
      IPOS  =NGAUSP*NDIME+NGAUSP+NGAUSP*NNODE+2*NGAUSB+1
      EXISC =RELPRP(IPOS)
      ETASC =RELPRP(IPOS+1)
C
      CALL SHPFUN
     1(   DERIV      ,ETASC      ,EXISC      ,0          ,IELTYP     ,
     2    NDIME      ,SHAPE      )
      CALL JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    NDIME      ,NDIME      ,NNODE      )
      IF(DETJAC.LE.R0)THEN
C stops program if element jacobian is not positive definite
        CALL ERRPRT('EE0007')
      ENDIF
      IF(NTYPE.EQ.3)CALL GETGCO
     1(   GPCOD      ,ELCOD      ,NDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C
      CALL GETGMX
```

```
     1(   GPCOD      ,CARTD      ,G0MATX     ,NDIME      ,MGDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C Determinant of the incremental deformation gradient at the centroid
      CALL DEFGRA
     1(   DELDIS     ,FINCIN     ,G0MATX     ,2          ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
      IF(NTYPE.EQ.2)THEN
        AFACT=RP5
        DET=FINCIN(1,1)*FINCIN(2,2)-FINCIN(1,2)*FINCIN(2,1)
      ELSEIF(NTYPE.EQ.3)THEN
C stops program if deformation gradient determinant is non-positive
        IF(FINCIN(3,3).LE.R0)CALL ERRPRT('EE0008')
        AFACT=R1D3
        DET=(FINCIN(1,1)*FINCIN(2,2)-FINCIN(1,2)*FINCIN(2,1))*
     1       FINCIN(3,3)
      ENDIF
      DET0=R1/DET
C Determinant of the total deformation gradient at the centroid
      CALL DEFGRA
     1(   TELDIS     ,FINV       ,G0MATX     ,2          ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
      IF(NTYPE.EQ.2)THEN
        DET=FINV(1,1)*FINV(2,2)-FINV(1,2)*FINV(2,1)
      ELSEIF(NTYPE.EQ.3)THEN
C stops program if deformation gradient determinant is non-positive
        IF(FINV(3,3).LE.R0)CALL ERRPRT('EE0008')
        DET=(FINV(1,1)*FINV(2,2)-FINV(1,2)*FINV(2,1))*FINV(3,3)
      ENDIF
      DETF0=R1/DET
C
C=========================================================================
C                  Begin loop over Gauss points                          |
C=========================================================================
      IPPOS=1
      IPWEI=NGAUSP*NDIME+1
      DO 70 IGAUSP=1,NGAUSP
        EXISP=RELPRP(IPPOS-1+IGAUSP*2-1)
        ETASP=RELPRP(IPPOS-1+IGAUSP*2)
        WEIGP=RELPRP(IPWEI-1+IGAUSP)
C Evaluate the shape functions and derivatives
        CALL SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
     2    NDIME      ,SHAPE      )
        CALL JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    NDIME      ,NDIME      ,NNODE      )
        IF(DETJAC.LE.R0)THEN
C... stops program if element jacobian is not positive definite
          CALL ERRPRT('EE0007')
        ENDIF
        IF(NTYPE.EQ.3)CALL GETGCO
     1(   GPCOD      ,ELCOD      ,NDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C
C Large strains: compute incremental deformation gradient
C -------------------------------------------------------
C gradient operator G in the current configuration
        CALL GETGMX
     1(   GPCOD      ,CARTD      ,GMATX      ,NDIME      ,MGDIM      ,
     2    NAXIS      ,NNODE      ,NTYPE      ,SHAPE      )
C inverse of incremental deformation gradient
        CALL DEFGRA
     1(   DELDIS     ,FINCIN     ,GMATX      ,2          ,MGDIM      ,
     2    NDOFN      ,NTYPE      ,NNODE      )
C stops program if deformation gradient determinant is non-positive
        IF(NTYPE.EQ.3.AND.FINCIN(3,3).LE.R0)CALL ERRPRT('EE0008')
C incremental deformation gradient
        CALL INVF2
     1(   FINCIN     ,FINCR      ,NTYPE      )
C Modified incremental deformation gradient for F-bar element
        IF(NTYPE.EQ.2)THEN
          DET=FINCR(1,1)*FINCR(2,2)-FINCR(1,2)*FINCR(2,1)
        ELSEIF(NTYPE.EQ.3)THEN
          DET=(FINCR(1,1)*FINCR(2,2)-FINCR(1,2)*FINCR(2,1))*
     1         FINCR(3,3)
        ENDIF
        FACTOR=(DET0/DET)**AFACT
        FINCR(1,1)=FACTOR*FINCR(1,1)
        FINCR(1,2)=FACTOR*FINCR(1,2)
        FINCR(2,1)=FACTOR*FINCR(2,1)
        FINCR(2,2)=FACTOR*FINCR(2,2)
        IF(NTYPE.EQ.3)FINCR(3,3)=FACTOR*FINCR(3,3)
C... and the determinant of the total deformation gradient
        DETF=DETF0
C
C Call material interface routine for consistent tangent computation
C calls: Compute the spatial tangent modulus AMATX (large strains)
C ======================================================================
C
        NLARGE=1
        CALL MATICT
     1(   DETF       ,KUNLD      ,MBDIM      ,MGDIM      ,
     2    NLARGE     ,NTYPE      ,
     3    AMATX      ,DMATX      ,EINCR      ,FINCR      ,IPROPS     ,
     4    LALGVA(1,IGAUSP)       ,RALGVA(1,IGAUSP)       ,RPROPS     ,
     5    RSTAVA(1,IGAUSP)       ,RSTAV2(1,IGAUSP)       ,
     5    STRSG(1,IGAUSP)        )
C
```

```
C Add current Gauss point contribution to element stiffness
C ===========================================================
C
C Compute elemental volume
        DVOLU=DETJAC*WEIGP
        IF(NTYPE.EQ.3)THEN
          DVOLU=DVOLU*TWOPI*GPCOD(NAXIS)
        ENDIF
C                                                           T
C Large strains:  assemble the element stiffness as  K:= G [a] G
C ----------------------------------------------------------------
        IF(NTYPE.EQ.3)THEN
          NGDIM=5
        ELSE
          NGDIM=4
        ENDIF
        CALL RTSR
     1(   AUXM       ,0          ,MEVAB      ,MGDIM      ,NEVAB      ,
     2    NGDIM      ,ESTIF      ,GMATX      ,AMATX      ,DVOLU      ,
     3    UNSYM      )
C Add extra matrix required by F-bar type element
        CALL ATMDFB
     1(   AMATX      ,NTYPE      ,QMATX      ,STRSG(1,IGAUSP)        )
CC NGDIM*NEVAB ??? Fix this!!!
        CALL RVSUB(G0MGMX,G0MATX,GMATX,MGDIM*MEVAB)
        CALL RTSX
     1(   AUXM       ,0          ,MEVAB      ,MGDIM      ,NEVAB      ,
     2    NGDIM      ,ESTIF      ,GMATX      ,QMATX      ,G0MGMX     ,
     3    DVOLU      )
C
   70 CONTINUE
C========================================================================
C                 End of loop over Gauss points                        |
C========================================================================
      RETURN
      END
```

```fortran
      SUBROUTINE STSTD2
     1(   IELEM      ,KUNLD      ,MDIME      ,MELEM      ,
     2    MPOIN      ,MSTRE      ,MTOTV      ,NAXIS      ,NLARGE     ,
     3    NTYPE      ,UNSYM      ,
     4    COORD1     ,DINCR      ,ESTIF      ,IELPRP     ,IPROPS     ,
     5    LALGVA     ,LNODS      ,RALGVA     ,RELPRP     ,RPROPS     ,
     6    RSTAVA     ,RSTAV2     ,STRSG      ,THKGP      ,TDISP      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
C
      PARAMETER( MGDIM=5 ,MBDIM=4 ,NDIME=2 ,NDOFN=2 )
C Arguments
      LOGICAL  LALGVA ,UNSYM
      DIMENSION
     1    COORD1(MDIME,MPOIN),DINCR(MTOTV)       ,ESTIF(MEVAB,MEVAB) ,
     2    IELPRP(*)         ,IPROPS(*)         ,LALGVA(MLALGV,MTOTG),
     3    LNODS(MELEM,MEVAB) ,RALGVA(MRALGV,MTOTG),RELPRP(*)          ,
     4    RPROPS(*)         ,RSTAVA(MRSTAV,MTOTG),RSTAV2(MRSTAV,MTOTG),
     5    STRSG(MSTRE,MTOTG) ,THKGP(MTOTG)       ,TDISP(MTOTV)
C Local arrays and variables
      DIMENSION
     1    AUXM(MEVAB,MGDIM)  ,AMATX(MGDIM,MGDIM) ,BMATX(4,MEVAB)      ,
     2    CARTD(NDIME,MNODE) ,DELDIS(2,MNODE)    ,DERIV(NDIME,MNODE) ,
     3    DMATX(MBDIM,MBDIM) ,EINCR(MBDIM)       ,ELCOD(NDIME,MNODE) ,
     4    FINCIN(3,3)        ,FINCR(3,3)         ,FINV(3,3)          ,
     5    GMATX(MGDIM,MEVAB) ,GPCOD(NDIME)       ,SHAPE(MNODE)       ,
     6    TELDIS(2,MNODE)
      DATA R0   ,R1   ,R8   /
     1     0.0D0,1.0D0,8.0D0/
C***********************************************************************
C EVALUATES THE ELEMENT TANGENT STIFFNESS MATRIX FOR ELEMENTS OF CLASS
C 'STDARD' (STANDARD ISOPARAMETRIC ELEMENTS) IN 2-D (PLANE STRAIN,
C PLANE STRESS AND AXISYMMETRIC PROBLEMS
C
C REFERENCE: Sections 4.2.4, 4.3.4
C            Box 4.2, item (iii)
C            Box 4.3
C***********************************************************************
      IF(NTYPE.EQ.3)TWOPI=R8*ATAN(R1)
C Identify element type
      IELTYP=IELPRP(1)
C Recover element information
      NNODE =IELPRP(3)
      NGAUSP=IELPRP(4)
      NEVAB =IELPRP(5)
C
C Set element nodal coordinates, total and incremental displacements
C vectors
C
      DO 20 INODE =1,NNODE
        LNODE=IABS(LNODS(IELEM,INODE))
        NPOSN=(LNODE-1)*NDOFN
        DO 10 IDOFN=1,NDOFN
          NPOSN=NPOSN+1
          ELCOD(IDOFN,INODE)=COORD1(IDOFN,LNODE)
          IF(NLARGE.EQ.1)THEN
            TELDIS(IDOFN,INODE)=-TDISP(NPOSN)
            DELDIS(IDOFN,INODE)=-DINCR(NPOSN)
          ELSE
            DELDIS(IDOFN,INODE)=DINCR(NPOSN)
          ENDIF
   10   CONTINUE
   20 CONTINUE
C
C Initialize the element stiffness matrix
C
      DO 40 IEVAB=1,NEVAB
        DO 30 JEVAB=1,NEVAB
          ESTIF(IEVAB,JEVAB)=R0
   30   CONTINUE
   40 CONTINUE
C
C=======================================================================
C                  Begin loop over Gauss points                        |
C=======================================================================
      IPPOS=1
      IPWEI=NGAUSP*NDIME+1
      DO 70 IGAUSP=1,NGAUSP
        EXISP=RELPRP(IPPOS-1+IGAUSP*2-1)
        ETASP=RELPRP(IPPOS-1+IGAUSP*2)
        WEIGP=RELPRP(IPWEI-1+IGAUSP)
C Evaluate the shape functions and derivatives
        CALL SHPFUN
     1(   DERIV      ,ETASP      ,EXISP      ,0          ,IELTYP     ,
     2    NDIME      ,SHAPE      )
        CALL JACOB2
     1(   CARTD      ,DERIV      ,DETJAC     ,ELCOD      ,IELEM      ,
     2    NDIME      ,NDIME      ,NNODE      )
        IF(DETJAC.LE.R0)THEN
C... stops program if element jacobian is not positive definite
          CALL ERRPRT('EE0003')
        ENDIF
        IF(NTYPE.EQ.3)CALL GETGCO
     1(   GPCOD      ,ELCOD      ,NDIME      ,NDIME      ,NNODE      ,
     2    SHAPE      )
C
```

```
        IF(NLARGE.EQ.1)THEN
C Large strains: compute incremental deformation gradient
C -------------------------------------------------------
C gradient operator G in the current configuration
          CALL GETGMX
     1(   GPCOD     ,CARTD     ,GMATX     ,NDIME     ,MGDIM     ,
     2    NAXIS     ,NNODE     ,NTYPE     ,SHAPE     )
C inverse of incremental deformation gradient
          CALL DEFGRA
     1(   DELDIS    ,FINCIN    ,GMATX     ,2         ,MGDIM     ,
     2    NDOFN     ,NTYPE     ,NNODE     )
C stops program if deformation gradient determinant is non-positive
          IF(NTYPE.EQ.3.AND.FINCIN(3,3).LE.R0)CALL ERRPRT('EE0004')
C incremental deformation gradient
          CALL INVF2
     1(   FINCIN    ,FINCR     ,NTYPE     )
C... and the determinant of the total deformation gradient
          CALL DEFGRA
     1(   TELDIS    ,FINV      ,GMATX     ,2         ,MGDIM     ,
     2    NDOFN     ,NTYPE     ,NNODE     )
          DETFIN=FINV(1,1)*FINV(2,2)-FINV(1,2)*FINV(2,1)
          IF(NTYPE.EQ.3)THEN
C stops program if deformation gradient is not positive definite
            IF(FINV(3,3).LE.R0)CALL ERRPRT('EE0004')
            DETFIN=DETFIN*FINV(3,3)
          ENDIF
          DETF=R1/DETFIN
        ELSE
C Small strains: compute incremental infinitesimal strain
C -------------------------------------------------------
C compute the symmetric gradient operator B
          CALL GETBMX
     1(   BMATX     ,GPCOD     ,CARTD     ,NDIME     ,4         ,
     2    NAXIS     ,NNODE     ,NTYPE     ,SHAPE     )
C and the incremental infinitesimal strain
          CALL LISTRA
     1(   BMATX     ,DELDIS    ,2         ,4         ,NDOFN     ,
     2    NNODE     ,NTYPE     ,EINCR     )
        ENDIF
C
C Call material interface routine for consistent tangent computation
C calls: Compute either the standard consistent tangent matrix DMATX
C (small strains) or the spatial tangent modulus AMATX (large strains)
C ====================================================================
C
        CALL MATICT
     1(   DETF      ,KUNLD     ,MBDIM     ,MGDIM     ,
     2    NLARGE    ,NTYPE     ,
     3    AMATX     ,DMATX     ,EINCR     ,FINCR     ,IPROPS    ,
     4    LALGVA(1,IGAUSP)     ,RALGVA(1,IGAUSP)     ,RPROPS    ,
     5    RSTAVA(1,IGAUSP)     ,RSTAV2(1,IGAUSP)     ,
     5    STRSG(1,IGAUSP)      )
C
C Add current Gauss point contribution to element stiffness
C =========================================================
C
C Compute elemental volume
        DVOLU=DETJAC*WEIGP
        IF(NTYPE.EQ.1)THEN
          DVOLU=DVOLU*THKGP(IGAUSP)
        ELSEIF(NTYPE.EQ.3)THEN
          DVOLU=DVOLU*TWOPI*GPCOD(NAXIS)
        ENDIF
        IF(NLARGE.EQ.1)THEN
C                                                          T
C Large strains:  assemble the element stiffness as  K:= G [a] G
C --------------------------------------------------------------
          IF(NTYPE.EQ.3)THEN
            NGDIM=5
          ELSE
            NGDIM=4
          ENDIF
          CALL RTSR
     1(   AUXM      ,0         ,MEVAB     ,MGDIM     ,NEVAB     ,
     2    NGDIM     ,ESTIF     ,GMATX     ,AMATX     ,DVOLU     ,
     3    UNSYM     )
        ELSE
C                                                         T
C Small strains: assemble the element stiffness as  K:= B D B
C -----------------------------------------------------------
          IF(NTYPE.EQ.1.OR.NTYPE.EQ.2)THEN
            NBDIM=3
          ELSEIF(NTYPE.EQ.3)THEN
            NBDIM=4
          ENDIF
          CALL RTSR
     1(   AUXM      ,0         ,MEVAB     ,4         ,NEVAB     ,
     2    NBDIM     ,ESTIF     ,BMATX     ,DMATX     ,DVOLU     ,
     3    UNSYM     )
        ENDIF
C
   70 CONTINUE
C=====================================================================
C                 End of loop over Gauss points                      |
C=====================================================================
      RETURN
      END
```

```
      SUBROUTINE SUDAMA
     1(   DGAMA      ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=6  ,MSTRE=4)
      LOGICAL IFPLAS, LALGVA(2), SUFAIL
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)          ,RSTAVA(MSTRE+2)    ,
     2    STRAT(MSTRE)       ,STRES(MSTRE)
      DIMENSION
     1    EET(MSTRE)
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,R3   ,R6   ,SMALL ,TOL   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,3.0D0,6.0D0,1.D-20,1.D-08/
      DATA MXITER / 25 /
C***********************************************************************
C STATE UPDATE PROCEDURE FOR LEMAITRE'S DUCTILE DAMAGE MATERIAL MODEL
C WITH NON-LINEAR (PIECEWISE LINEAR) ISOTROPIC HARDENING:
C IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM.
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Box 12.4
C***********************************************************************
C Stop program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0051')
C Initialise some algorithmic and internal variables
      DGAMA=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
C Retrieve hardening and damage internal variables
      HVARN=RSTAVA(MSTRE+1)
      DAMAGN=RSTAVA(MSTRE+2)
C... integrity
      OMEGAN=R1-DAMAGN
C Retrieve some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      DAMEXP=RPROPS(4)
      DAMDEN=RPROPS(5)
      NHARD=IPROPS(3)
C Shear and bulk moduli and other necessary constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2BULK=R2*BULK
      R2G=R2*GMODU
      R3G=R3*GMODU
      R6G=R6*GMODU
C Elastic predictor: Compute elastic trial state
C ==============================================
C Volumetric strain and (undamaged) pressure stress
      EEV=STRAT(1)+STRAT(2)+STRAT(4)
      PTILDE=BULK*EEV
C Elastic trial deviatoric strain
      EEVD3=EEV/R3
      EET(1)=STRAT(1)-EEVD3
      EET(2)=STRAT(2)-EEVD3
      EET(4)=STRAT(4)-EEVD3
C Convert engineering shear component into physical component
      EET(3)=STRAT(3)/R2
C Compute trial (undamaged) von Mises effective stress and uniaxial
C yield stress
      VARJ2T=R2G*R2G*(EET(3)*EET(3)+RP5*(EET(1)*EET(1)+
     1                     EET(2)*EET(2)+EET(4)*EET(4)))
      QTILTR=SQRT(R3*VARJ2T)
      SIGMAY=PLFUN(HVARN,NHARD,RPROPS(IPHARD))
C Check for plastic admissibility
C ==============================
      PHI=QTILTR-SIGMAY
      IF(PHI/SIGMAY.GT.TOL)THEN
C Plastic step: Apply return mapping - use Newton-Raphson algorithm
C               to solve the return mapping equation for DGAMA
C =================================================================
C Reset plastic flag
        IFPLAS=.TRUE.
C Initial guess for DGAMA: Use perfectly plastic solution with frozen
C yield surface at the beginning of the load increment
        DGAMA=OMEGAN*PHI/R3G
C Initialise hardening variable
        HVAR=HVARN+DGAMA
C Start N-R iterations
C --------------------
        PTILD2=PTILDE**2
        DO 10 NRITER=1,MXITER
C yield stress
          SIGMAY=PLFUN(HVAR,NHARD,RPROPS(IPHARD))
C integrity
          AUX2=R3G/(QTILTR-SIGMAY)
          OMEGA=AUX2*DGAMA
C stress triaxiality and damage energy release rate
          Y=-SIGMAY**2/R6G-PTILD2/R2BULK
C Compute residual function
          RES=OMEGA-OMEGAN+(-Y/DAMDEN)**DAMEXP/AUX2
C Check for convergence
          IF(ABS(RES).LE.TOL)THEN
C... update hardening and damage variables
            IF(OMEGA.LT.SMALL)THEN
C... check if converged damage variable is acceptable
              SUFAIL=.TRUE.
              CALL ERRPRT('WE0019')
              GOTO 999
            ENDIF
            DAMAGE=R1-OMEGA
            RSTAVA(MSTRE+1)=HVAR
            RSTAVA(MSTRE+2)=DAMAGE
C... update stress components
            P=OMEGA*PTILDE
            Q=SIGMAY*OMEGA
            FACTOR=R2G*Q/QTILTR
            STRES(1)=FACTOR*EET(1)+P
            STRES(2)=FACTOR*EET(2)+P
            STRES(3)=FACTOR*EET(3)
            STRES(4)=FACTOR*EET(4)+P
C... compute and store converged elastic (engineering) strain components
```

```
            FACTOR=R1-R3G*DGAMA/(OMEGA*QTILTR)
            RSTAVA(1)=FACTOR*EET(1)+EEVD3
            RSTAVA(2)=FACTOR*EET(2)+EEVD3
            RSTAVA(3)=FACTOR*EET(3)*R2
            RSTAVA(4)=FACTOR*EET(4)+EEVD3
            GOTO 999
          ENDIF
C Compute derivative of residual function
C... slope of hardening function
          HSLOPE=DPLFUN(HVAR,NHARD,RPROPS(IPHARD))
C... derivative of -Y
          DY=-HSLOPE*SIGMAY/R3G
C... residual derivative
          AUX=-Y/DAMDEN
          DRES=AUX2+AUX2*DGAMA*HSLOPE/(QTILTR-SIGMAY)-
     1         HSLOPE/R3G*AUX**DAMEXP-
     2         DAMEXP*DY/(AUX2*DAMDEN)*AUX**(DAMEXP-R1)
C Apply N-R correction to DGAMA
          DDGAMA=-RES/DRES
          DGAMA=DGAMA+DDGAMA
C... update hardening variable
          HVAR=HVARN+DGAMA
   10   CONTINUE
C N-R loop failed to converge: Reset failure flag and issue warning
C                              message before exiting
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0018')
      ELSE
C Elastic step: Update stress using damaged elastic law
C =====================================================
        FACTOR=R2G*OMEGAN
        P=OMEGAN*PTILDE
        STRES(1)=FACTOR*EET(1)+P
        STRES(2)=FACTOR*EET(2)+P
        STRES(3)=FACTOR*EET(3)
        STRES(4)=FACTOR*EET(4)+P
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=STRAT(4)
      ENDIF
  999 CONTINUE
C Update some algorithmic variables before exit
      LALGVA(1)=IFPLAS
      LALGVA(2)=SUFAIL
      RETURN
      END
```

```
      SUBROUTINE SUDMEL
     1(   NTYPE      ,RPROPS     ,RSTAVA     ,STRAN      ,STRES      ,
     2    SUFAIL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(MSTRE=4)
C Arguments
      LOGICAL  SUFAIL
      DIMENSION
     1    RPROPS(*)          ,RSTAVA(MSTRE)      ,STRAN(MSTRE)       ,
     2    STRES(MSTRE)
C Local variables and arrays
      LOGICAL  DUMMY
      DIMENSION
     1    DPSTRA(3,3)        ,DPSTRE(3,3)        ,EED(MSTRE)         ,
     2    EIGPRJ(MSTRE,2)    ,PDMINU(3,3)        ,PDPLUS(3,3)        ,
     4    PSTRA(3)           ,PSTRES(3)          ,RESVEC(3)          ,
     5    SIGMA(MSTRE)       ,STRAIN(MSTRE)      ,VI(3)              ,
     6    VIDMIN(3)          ,VIDPLU(3)
      DATA
     1    VI(1),VI(2),VI(3)/
     2    1.D0 ,1.D0 ,1.D0 /
      DATA
     1    R0   ,R1   ,R2    ,R3    ,TOL   /
     2    0.D0 ,1.D0 ,2.0D0 ,3.0D0 ,1.D-10/
      DATA
     1    MXITER/    10/
C***********************************************************************
C STATE UPDATE PROCEDURE FOR ISOTROPICALLY DAMAGED ISOTROPIC ELASTIC
C MODEL ACCOUNTING FOR PARTIAL MICROCRACK/VOID CLOSURE EFFECTS
C
C REFERENCE: Box 12.7
C***********************************************************************
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0057')
C Initialise logical flag
      SUFAIL=.FALSE.
C Set material constants
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      DAMAGE=RPROPS(4)
      HFACT=RPROPS(5)
      GMODU=RPROPS(6)
      BULK=RPROPS(7)
C Transform engineering into physical strains
      STRAIN(1)=STRAN(1)
      STRAIN(2)=STRAN(2)
      STRAIN(3)=STRAN(3)/R2
      STRAIN(4)=STRAN(4)
C Perform spectral decomposition of the strain tensor
      CALL SPDEC2(EIGPRJ,PSTRA,DUMMY,STRAIN)
      PSTRA(3)=STRAN(4)
C
C Newton-Raphson iterations to solve the piece-wise linear elastic
C constitutive equation
C ------------------------------------------------------------------
C
C Set initial guess for principal stresses as the stresses obtained for
C the damaged elastic material without partial crack closure effects
C... compute corresponding stress tensor components first
      EEV=STRAN(1)+STRAN(2)+STRAN(4)
      EEVD3=EEV/R3
      EED(1)=STRAN(1)-EEVD3
      EED(2)=STRAN(2)-EEVD3
      EED(4)=STRAN(4)-EEVD3
      P=(R1-DAMAGE)*BULK*EEV
      R1DR2G=(R1-DAMAGE)*R2*GMODU
      SIGMA(1)=R1DR2G*EED(1)+P
      SIGMA(2)=R1DR2G*EED(2)+P
      SIGMA(3)=R1DR2G*EED(3)
      SIGMA(4)=R1DR2G*EED(4)+P
C... and then the principal stresses (internal product between stress
C    tensor and individual eigenprojection tensors)
      PSTRES(1)=SIGMA(1)*EIGPRJ(1,1)+SIGMA(2)*EIGPRJ(2,1)+
     1          R2*SIGMA(3)*EIGPRJ(3,1)
      PSTRES(2)=SIGMA(1)*EIGPRJ(1,2)+SIGMA(2)*EIGPRJ(2,2)+
     1          R2*SIGMA(3)*EIGPRJ(3,2)
      PSTRES(3)=SIGMA(4)
C Zero relevant arrays
      CALL RVZERO(PDPLUS,9)
      CALL RVZERO(PDMINU,9)
C
C Begin N-R iterations
C
      DO 80 ITER=1,MXITER
C Construct current projection matrices
C... positive and negative principal stress projection matrices
        IF(PSTRES(1).GE.R0)THEN
          PDPLUS(1,1)=R1/(R1-DAMAGE)
          PDMINU(1,1)=R0
        ELSE
          PDPLUS(1,1)=R0
          PDMINU(1,1)=R1/(R1-HFACT*DAMAGE)
        ENDIF
        IF(PSTRES(2).GE.R0)THEN
          PDPLUS(2,2)=R1/(R1-DAMAGE)
          PDMINU(2,2)=R0
        ELSE
          PDPLUS(2,2)=R0
          PDMINU(2,2)=R1/(R1-HFACT*DAMAGE)
```

```
        ENDIF
        IF(PSTRES(3).GE.R0)THEN
          PDPLUS(3,3)=R1/(R1-DAMAGE)
          PDMINU(3,3)=R0
        ELSE
          PDPLUS(3,3)=R0
          PDMINU(3,3)=R1/(R1-HFACT*DAMAGE)
        ENDIF
C... positive and negative trace operators
        TRACE=PSTRES(1)+PSTRES(2)+PSTRES(3)
        IF(TRACE.GE.R0)THEN
          VIDPLU(1)=R1/(R1-DAMAGE)
          VIDPLU(2)=R1/(R1-DAMAGE)
          VIDPLU(3)=R1/(R1-DAMAGE)
          VIDMIN(1)=R0
          VIDMIN(2)=R0
          VIDMIN(3)=R0
        ELSE
          VIDPLU(1)=R0
          VIDPLU(2)=R0
          VIDPLU(3)=R0
          VIDMIN(1)=R1/(R1-HFACT*DAMAGE)
          VIDMIN(2)=R1/(R1-HFACT*DAMAGE)
          VIDMIN(3)=R1/(R1-HFACT*DAMAGE)
        ENDIF
C Inverse elasticity operator that transforms principal stresses into
C principal strains (matrix of derivatives of principal strains with
C respect to principal stresses)
        DO 20 I=1,3
          DO 10 J=1,3
            DPSTRA(I,J)=(R1+POISS)/YOUNG*(PDPLUS(I,J)+PDMINU(I,J))-
     1                  POISS/YOUNG*(VI(I)*(VIDPLU(J)+VIDMIN(J)))
   10     CONTINUE
   20   CONTINUE
C Compute residual of constitutive equation
        DO 40 I=1,3
          RESVEC(I)=R0
          DO 30 J=1,3
            RESVEC(I)=RESVEC(I)-DPSTRA(I,J)*PSTRES(J)
   30     CONTINUE
          RESVEC(I)=PSTRA(I)+RESVEC(I)
   40   CONTINUE
C... residual norm
        RESNOR=SQRT(RESVEC(1)**2+RESVEC(2)**2+RESVEC(3)**2)
        IF(RESNOR.LT.TOL)THEN
C Iterations converged: update stress tensor components, store
C                       engineering strain in RSTAVA, break N-R loop
C                       and exit this routine
          STRES(1)=PSTRES(1)*EIGPRJ(1,1)+PSTRES(2)*EIGPRJ(1,2)
          STRES(2)=PSTRES(1)*EIGPRJ(2,1)+PSTRES(2)*EIGPRJ(2,2)
          STRES(3)=PSTRES(1)*EIGPRJ(3,1)+PSTRES(2)*EIGPRJ(3,2)
          STRES(4)=PSTRES(3)
          RSTAVA(1)=STRAN(1)
          RSTAVA(2)=STRAN(2)
          RSTAVA(3)=STRAN(3)
          RSTAVA(4)=STRAN(4)
          GOTO 999
        ENDIF
C Compute elasticity operator that transforms principal strains into
C principal stresses (matrix of derivatives of principal stresses with
C respect to principal strains). This is the inverse of the residual
C vector derivative
        CALL INVMT3(DPSTRA,DPSTRE,DET)
C Apply N-R correction to principal stresses
        DO 70 I=1,3
          DO 60 J=1,3
            PSTRES(I)=PSTRES(I)+DPSTRE(I,J)*RESVEC(J)
   60     CONTINUE
   70   CONTINUE
   80 CONTINUE
C N-R loop not converged: Failure of stress update procedure.
C                         Stresses are not updated. Send warning message
      SUFAIL=.TRUE.
      CALL ERRPRT('WE0020')
C
  999 RETURN
      END
```

```
      SUBROUTINE SUDP
     1(   DGAM       ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MSTRE=4)
      LOGICAL APEX, IFPLAS, LALGVA(3), SUFAIL
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)          ,RSTAVA(MSTRE+1)    ,
     2    STRAT(MSTRE)       ,STRES(MSTRE)
      DIMENSION
     1    STRIAL(MSTRE)
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,R3   ,TOL   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,3.0D0,1.D-08/
      DATA MAXRT / 50 /
C***********************************************************************
C STRESS UPDATE PROCEDURE FOR DRUCKER PRAGER TYPE ELASTO-PLASTIC
C MATERIAL WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW RULE AND PIECE_WISE
C LINEAR ISOTROPIC HARDENING:
C IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM (Boxes 8.8-10)
C
C REFERENCE: Boxes 8.8-10
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0016')
C Initialize some algorithmic and internal variables
      DGAMA=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
      EPBARN=RSTAVA(MSTRE+1)
      EPBAR=EPBARN
C Set some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      ETA=RPROPS(4)
      XI=RPROPS(5)
      ETABAR=RPROPS(6)
      NHARD=IPROPS(3)
C and some constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R1D3=R1/R3
C Compute elastic trial state
C ----------------------------
C Elastic trial volumetric strain and pressure stress
      EETV=STRAT(1)+STRAT(2)+STRAT(4)
      PT=BULK*EETV
C Elastic trial deviatoric stress
      EEVD3=EETV*R1D3
      STRIAL(1)=R2G*(STRAT(1)-EEVD3)
      STRIAL(2)=R2G*(STRAT(2)-EEVD3)
      STRIAL(4)=R2G*(STRAT(4)-EEVD3)
C shear component
      STRIAL(3)=R2G*(STRAT(3)*RP5)
C Compute elastic trial stress J2 invariant and cohesion
      VARJ2T=STRIAL(3)*STRIAL(3)+RP5*(STRIAL(1)*STRIAL(1)+
     1       STRIAL(2)*STRIAL(2)+STRIAL(4)*STRIAL(4))
      COHE=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
C Check for plastic consistency
C ------------------------------
      SQRJ2T=SQRT(VARJ2T)
      PHI=SQRJ2T+ETA*PT-XI*COHE
      RES=PHI
      IF(COHE.NE.R0)RES=RES/ABS(COHE)
      IF(RES.GT.TOL)THEN
C Plastic step: Use return mapping
C ================================
        IFPLAS=.TRUE.
        APEX=.FALSE.
C Apply return mapping to smooth portion of cone - REFERENCE: Box 8.9
C -------------------------------------------------------------------
        DO 20 IPTER1=1,MAXRT
C Compute residual derivative
          DENOM=-GMODU-BULK*ETABAR*ETA-
     1           XI*XI*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
C Compute Newton-Raphson increment and update variable DGAMA
          DDGAMA=-PHI/DENOM
          DGAMA=DGAMA+DDGAMA
C Compute new residual
          EPBAR=EPBARN+XI*DGAMA
          COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          SQRJ2=SQRJ2T-GMODU*DGAMA
          P=PT-BULK*ETABAR*DGAMA
          PHI=SQRJ2+ETA*P-XI*COHE
C Check convergence
          RESNOR=ABS(PHI)
          IF(COHE.NE.R0)RESNOR=RESNOR/ABS(COHE)
          IF(RESNOR.LE.TOL)THEN
C Check validity of return to smooth portion
            IF(SQRJ2.GE.R0)THEN
C results are valid, update stress components and other variables
              IF(SQRJ2T.EQ.R0)THEN
                FACTOR=R0
              ELSE
                FACTOR=R1-GMODU*DGAMA/SQRJ2T
              ENDIF
              GOTO 50
            ELSE
C smooth wall return not valid - go to apex return procedure
              GOTO 30
            ENDIF
          ENDIF
   20   CONTINUE
C failure of stress update procedure
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0002')
        GOTO 999
   30   CONTINUE
C Apply return mapping to APEX - REFERENCE: Box 8.10
C --------------------------------------------------
C perform checks and set some variables
        APEX=.TRUE.
```

```
        IF(ETA.EQ.R0)CALL ERRPRT('EE0011')
        IF(ETABAR.EQ.R0)CALL ERRPRT('EE0012')
        ALPHA=XI/ETABAR
        BETA=XI/ETA
C Set initial guess for unknown DEPV and start iterations
        DEPV=R0
        EPBAR=EPBARN
        COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
        RES=BETA*COHE-PT
        DO 40 IPTER2=1,MAXRT
          DENOM=ALPHA*BETA*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))+BULK
C Compute Newton-Raphson increment and update variable DEPV
          DDEPV=-RES/DENOM
          DEPV=DEPV+DDEPV
C Compute new residual
          EPBAR=EPBARN+ALPHA*DEPV
          COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          P=PT-BULK*DEPV
          RES=BETA*COHE-P
C Check convergence
          RESNOR=ABS(RES)
          IF(COHE.NE.R0)RESNOR=RESNOR/ABS(COHE)
          IF(RESNOR.LE.TOL)THEN
C update stress components and other variables
            DGAMA=DEPV/ETABAR
            FACTOR=R0
            GOTO 50
          ENDIF
   40   CONTINUE
C failure of stress update procedure
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0002')
        GOTO 999
C Store converged stress components and other state variables
C -------------------------------------------------------------
   50   CONTINUE
        STRES(1)=FACTOR*STRIAL(1)+P
        STRES(2)=FACTOR*STRIAL(2)+P
        STRES(3)=FACTOR*STRIAL(3)
        STRES(4)=FACTOR*STRIAL(4)+P
C update EPBAR
        RSTAVA(MSTRE+1)=EPBAR
C compute converged elastic (engineering) strain components
        FACTOR=FACTOR/R2G
        EEVD3=P/(BULK*R3)
        RSTAVA(1)=FACTOR*STRIAL(1)+EEVD3
        RSTAVA(2)=FACTOR*STRIAL(2)+EEVD3
        RSTAVA(3)=FACTOR*STRIAL(3)*R2
        RSTAVA(4)=FACTOR*STRIAL(4)+EEVD3
      ELSE
C Elastic step: update stress using linear elastic law
C ====================================================
        STRES(1)=STRIAL(1)+PT
        STRES(2)=STRIAL(2)+PT
        STRES(3)=STRIAL(3)
        STRES(4)=STRIAL(4)+PT
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=STRAT(4)
      ENDIF
  999 CONTINUE
C Update some algorithmic variables before exit
C ===============================================
      LALGVA(1)=IFPLAS
      LALGVA(2)=SUFAIL
      LALGVA(3)=APEX
      DGAM=DGAMA
      RETURN
      END
```

```
      SUBROUTINE SUDPPN
     1(   RALGVA     ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MSTRE=4)
C Arguments
      LOGICAL LALGVA(3)
      DIMENSION
     1    RALGVA(3)          ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(MSTRE)       ,STRES(MSTRE)
C Local arrays and variables
      LOGICAL EPFLAG ,IFPLAS ,SUFAIL
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,RSTAUX(MSTRE+1)
      DATA
     1    R0    ,TOL   /
     2    0.0D0 ,1.D-08/
      DATA MXITER / 20 /
C***********************************************************************
C STATE UPDATE PROCEDURE FOR THE DRUCKER-PRAGER ELASTO-PLASTIC MODEL
C WITH NON-LINEAR (PIECEWISE LINEAR) ISOTROPIC HARDENING IN PLANE
C STRESS. NESTED ITERATION APPROACH.
C
C REFERENCE: Section 9.2.2
C***********************************************************************
C Stop program if not plane stress
      IF(NTYPE.NE.1)CALL ERRPRT('EI0038')
C Initialise the state update failure flag
      SUFAIL=.FALSE.
C Set some material properties
      NHARD=IPROPS(3)
C
C Begin Newton-Raphson iteration loop for plane stress enforcement
C -----------------------------------------------------------------
C
C Set initial guess for elastic trial thickness strain. Use previously
C converged elastic thickness strain.
      E33TRL=RSTAVA(4)
C Start N-R loop
      DO 20 ITER=1,MXITER
C Set state variables to values at beginning of increment
        DO 10 I=1,MSTRE+1
          RSTAUX(I)=RSTAVA(I)
   10   CONTINUE
C Use axisymmetric integration algorithm to compute stresses, etc.
        STRAT(4)=E33TRL
        CALL SUDP
     1(   RALGVA     ,IPROPS     ,LALGVA     ,3          ,RPROPS     ,
     2    RSTAUX     ,STRAT      ,STRES      )
        SUFAIL=LALGVA(2)
        IF(SUFAIL)THEN
C... emergency exit in case of failure of the state apdate procedure
          GOTO 999
        ENDIF
        IFPLAS=LALGVA(1)
C Check plane stress convergence
        EPBAR=RSTAVA(MSTRE+1)
        COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
        RES=ABS(STRES(4))
C...use normalised out-of-plane stress
        IF(COHE.NE.R0)RES=RES/ABS(COHE)
        IF(RES.LE.TOL)THEN
C...and break N-R loop in case of convergence
          GOTO 30
        ENDIF
C Compute axisymmetric consistent tangent components
        EPFLAG=IFPLAS
        CALL CTDP
     1(   RALGVA     ,DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,
     2    3          ,RPROPS     ,RSTAUX     ,STRAT      )
C Apply Newton-Raphson correction to normal elastic trial strain
        D22=DMATX(4,4)
        E33TRL=E33TRL-STRES(4)/D22
   20 CONTINUE
C Emergency exit in case of failure of the plane stress enforcement loop
      SUFAIL=.TRUE.
      LALGVA(2)=SUFAIL
      CALL ERRPRT('WE0016')
      GOTO 999
   30 CONTINUE
C Set state variables to current updated values
      DO 40 I=1,MSTRE+1
        RSTAVA(I)=RSTAUX(I)
   40 CONTINUE
C Store the converged elastic trial thickness strain in the array of
C real algorithmic variables
      RALGVA(3)=E33TRL
C
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SUEL
     1(   NTYPE      ,RPROPS     ,RSTAVA     ,STRAN      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(MSTRE=4)
      DIMENSION
     1    RPROPS(*)          ,RSTAVA(MSTRE)      ,STRAN(*)           ,
     2    STRES(*)
      DIMENSION
     1    EED(MSTRE)
      DATA
     1    RP5  ,R2   ,R3   ,R4   /
     2    0.5D0,2.0D0,3.0D0,4.0D0/
C***********************************************************************
C STATE UPDATE PROCEDURE FOR LINEAR ELASTIC MATERIAL MODEL
C
C REFERENCE: Expression (4.43)
C***********************************************************************
C
C Set shear and bulk modulus
C
      GMODU=RPROPS(2)
      BULK=RPROPS(3)
C
C Decompose strain into deviatoric and volumetric components
C ----------------------------------------------------------
C
      R2G=R2*GMODU
      IF(NTYPE.EQ.1)THEN
C for plane stress
        R4G=R4*GMODU
        R4GD3=R4G/R3
        FACTOR=R2G/(BULK+R4GD3)
        EEV=(STRAN(1)+STRAN(2))*FACTOR
        EEVD3=EEV/R3
        EED(1)=STRAN(1)-EEVD3
        EED(2)=STRAN(2)-EEVD3
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
C for plane strain and axisymmetric cases
        EEV=STRAN(1)+STRAN(2)+STRAN(4)
        EEVD3=EEV/R3
        EED(1)=STRAN(1)-EEVD3
        EED(2)=STRAN(2)-EEVD3
        EED(4)=STRAN(4)-EEVD3
      ELSE
        CALL ERRPRT('EI0018')
      ENDIF
C Convert engineering shear component into physical component
      EED(3)=STRAN(3)*RP5
C
C Update stress using linear elastic law
C ---------------------------------------
C
C hydrostatic stress
      P=BULK*EEV
C stress tensor components
      STRES(1)=R2G*EED(1)+P
      STRES(2)=R2G*EED(2)+P
      STRES(3)=R2G*EED(3)
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)STRES(4)=R2G*EED(4)+P
C
C Store elastic engineering strain in RSTAVA
C ------------------------------------------
C
      RSTAVA(1)=STRAN(1)
      RSTAVA(2)=STRAN(2)
      RSTAVA(3)=STRAN(3)
      IF(NTYPE.EQ.1)THEN
        R3BULK=R3*BULK
        RSTAVA(4)=-(STRAN(1)+STRAN(2))*(R3BULK-R2G)/(R3BULK+R4G)
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        RSTAVA(4)=STRAN(4)
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SUMC
     1(   DGAM       ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=7  ,MSTRE=4)
C Arguments
      LOGICAL
     1    LALGVA(5)
      DIMENSION
     1    DGAM(2)            ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(MSTRE)       ,STRES(MSTRE)
C Local variables and arrays
      LOGICAL
     1    APEX, DUMMY, EDGE, IFPLAS, RIGHT, SUFAIL
      DIMENSION
     1    EIGPRJ(MSTRE,2)    ,PSTRS(3)           ,STREST(3)
      DATA
     1    R0   ,R1   ,R2   ,R3   ,R4   ,SMALL ,TOL   /
     2    0.0D0,1.0D0,2.0D0,3.0D0,4.0D0,1.D-06,1.D-10/
      DATA MXITER / 50 /
C***********************************************************************
C STATE UPDATE PROCEDURE FOR MOHR-COULOMB TYPE ELASTO-PLASTIC MATERIAL
C WITH ASSOCIATIVE/NON-ASSOCIATIVE FLOW RULE AND PIECE-WISE LINEAR
C ISOTROPIC HARDENING:
C IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM (BOXES 8.4-7).
C PLANE STRAIN AND AXISYMMETRIC IMPLMENTATIONS.
C
C REFERENCE: Boxes 8.4-7
C            Section 8.2.2
C***********************************************************************
C Stops program if neither plane strain nor plane stress state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0027')
C Initialize some algorithmic and internal variables
      DGAMA=R0
      DGAMB=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
      EDGE=.FALSE.
      APEX=.FALSE.
      EPBARN=RSTAVA(MSTRE+1)
      EPBAR=EPBARN
C Set some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      SINPHI=RPROPS(4)
      COSPHI=RPROPS(5)
      SINPSI=RPROPS(6)
      NHARD=IPROPS(3)
C Set some constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R4G=R4*GMODU
      R2BULK=R2*BULK
      R2CPHI=R2*COSPHI
      R1D3=R1/R3
C Compute elastic trial state
C ---------------------------
C Elastic trial volumetric strain and pressure stress
      EETV=STRAT(1)+STRAT(2)+STRAT(4)
      PT=BULK*EETV
C Spectral decomposition of the elastic trial stress
      EETVD3=EETV*R1D3
      STREST(1)=R2G*(STRAT(1)-EETVD3)+PT
      STREST(2)=R2G*(STRAT(2)-EETVD3)+PT
      STREST(3)=GMODU*STRAT(3)
      CALL SPDEC2(EIGPRJ,PSTRS,DUMMY,STREST)
      PSTRS(3)=R2G*(STRAT(4)-EETVD3)+PT
C Identify maximum (PSTRS1) and minimum (PSTRS3) principal stresses
      II=1
      JJ=1
      PSTRS1=PSTRS(II)
      PSTRS3=PSTRS(JJ)
      DO 10 I=2,3
        IF(PSTRS(I).GE.PSTRS1)THEN
          II=I
          PSTRS1=PSTRS(II)
        ENDIF
        IF(PSTRS(I).LT.PSTRS3)THEN
          JJ=I
          PSTRS3=PSTRS(JJ)
        ENDIF
   10 CONTINUE
      IF(II.NE.1.AND.JJ.NE.1)MM=1
      IF(II.NE.2.AND.JJ.NE.2)MM=2
      IF(II.NE.3.AND.JJ.NE.3)MM=3
      PSTRS2=PSTRS(MM)
C Compute trial yield function and check for plastic consistency
C --------------------------------------------------------------
      COHE=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
      SMCT=PSTRS1-PSTRS3+(PSTRS1+PSTRS3)*SINPHI
      PHIA=SMCT-R2CPHI*COHE
      RES=PHIA
      IF(COHE.NE.R0)RES=RES/ABS(COHE)
      IF(RES.GT.TOL)THEN
C Plastic step: Apply return mapping
C ==================================
        IFPLAS=.TRUE.
C identify possible edge return: either right or left of main plane
        SCAPRD=PSTRS1*(R1-SINPSI)+PSTRS2*(-R2)+PSTRS3*(R1+SINPSI)
        IF(SCAPRD.GE.R0)THEN
          RIGHT=.TRUE.
        ELSE
          RIGHT=.FALSE.
        ENDIF
C Apply one-vector return mapping first (return to MAIN PLANE)
C ------------------------------------------------------------
        SPHSPS=SINPHI*SINPSI
        CONSTA=R4G*(R1+R1D3*SPHSPS)+R4*BULK*SPHSPS
        R4C2PH=R2CPHI*R2CPHI
C Start Newton-Raphson iterations for DGAMA
        DO 20 NRITER=1,MXITER
C Compute residual derivative
```

```
          DENOM=-CONSTA-R4C2PH*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
C Compute Newton-Raphson increment and update variable DGAMA
          DDGAMA=-PHIA/DENOM
          DGAMA=DGAMA+DDGAMA
C Compute new residual
          EPBAR=EPBARN+R2CPHI*DGAMA
          COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          PHIA=SMCT-CONSTA*DGAMA-R2CPHI*COHE
C Check convergence
          RESNOR=ABS(PHIA)
          IF(SMCT.NE.R0)RESNOR=RESNOR/ABS(SMCT)
          IF(RESNOR.LE.TOL)THEN
C Check validity of 1-vector return (check sextant of converged stress)
            S1=PSTRS1-(R2G*(R1+R1D3*SINPSI)+R2BULK*SINPSI)*DGAMA
            S2=PSTRS2+(R4G*R1D3-R2BULK)*SINPSI*DGAMA
            S3=PSTRS3+(R2G*(R1-R1D3*SINPSI)-R2BULK*SINPSI)*DGAMA
            DELTA=DMAX1(ABS(S1),ABS(S2),ABS(S3))*SMALL
            IF(S1+DELTA.GE.S2.AND.S2+DELTA.GE.S3)THEN
C converged stress is in the same sextant as trial stress -> 1-vector
C return is valid.
              P=(S1+S2+S3)*R1D3
              GOTO 70
            ELSE
C converged stress is not in the same sextant -> 1-vector result is
C not valid. Go to two-vector return map to edge
              GOTO 30
            ENDIF
          ENDIF
   20   CONTINUE
C failure of stress update procedure
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0003')
        GOTO 999
   30   CONTINUE
C Apply two-vector return mapping to appropriate EDGE
C --------------------------------------------------
        DGAMA=R0
        EPBAR=EPBARN
        COHE=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
        SMCTA=PSTRS1-PSTRS3+(PSTRS1+PSTRS3)*SINPHI
        IF(RIGHT)THEN
          SMCTB=PSTRS1-PSTRS2+(PSTRS1+PSTRS2)*SINPHI
        ELSE
          SMCTB=PSTRS2-PSTRS3+(PSTRS2+PSTRS3)*SINPHI
        ENDIF
        PHIA=SMCTA-R2CPHI*COHE
        PHIB=SMCTB-R2CPHI*COHE
        IF(RIGHT)THEN
          CONSTB=R2G*(R1+SINPHI+SINPSI-R1D3*SPHSPS)+R4*BULK*SPHSPS
        ELSE
          CONSTB=R2G*(R1-SINPHI-SINPSI-R1D3*SPHSPS)+R4*BULK*SPHSPS
        ENDIF
C Start Newton-Raphson iterations for DGAMA and DGAMB
        DO 40 NRITER=1,MXITER
C Compute residual derivative matrix
          FACTA=R4C2PH*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          DRVAA=-CONSTA-FACTA
          DRVAB=-CONSTB-FACTA
          DRVBA=-CONSTB-FACTA
          DRVBB=-CONSTA-FACTA
C Compute Newton-Raphson increment and update variables DGAMA and DGAMB
          R1DDET=R1/(DRVAA*DRVBB-DRVAB*DRVBA)
          DDGAMA=(-DRVBB*PHIA+DRVAB*PHIB)*R1DDET
          DDGAMB=(DRVBA*PHIA-DRVAA*PHIB)*R1DDET
          DGAMA=DGAMA+DDGAMA
          DGAMB=DGAMB+DDGAMB
C Compute new residual
          EPBAR=EPBARN+R2CPHI*(DGAMA+DGAMB)
          COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          PHIA=SMCTA-CONSTA*DGAMA-CONSTB*DGAMB-R2CPHI*COHE
          PHIB=SMCTB-CONSTB*DGAMA-CONSTA*DGAMB-R2CPHI*COHE
C Check convergence
          RESNOR=(ABS(PHIA)+ABS(PHIB))
          FACTOR=(ABS(SMCTA)+ABS(SMCTB))
          IF(FACTOR.NE.R0)RESNOR=RESNOR/FACTOR
          IF(RESNOR.LE.TOL)THEN
C Check validity of 2-vector return to edge
            AUX1=R2G*(R1+R1D3*SINPSI)+R2BULK*SINPSI
            AUX2=(R4G*R1D3-R2BULK)*SINPSI
            AUX3=R2G*(R1-R1D3*SINPSI)-R2BULK*SINPSI
            IF(RIGHT)THEN
              S1=PSTRS1-AUX1*(DGAMA+DGAMB)
              S2=PSTRS2+AUX2*DGAMA+AUX3*DGAMB
              S3=PSTRS3+AUX3*DGAMA+AUX2*DGAMB
            ELSE
              S1=PSTRS1-AUX1*DGAMA+AUX2*DGAMB
              S2=PSTRS2+AUX2*DGAMA-AUX1*DGAMB
              S3=PSTRS3+AUX3*(DGAMA+DGAMB)
            ENDIF
            DELTA=DMAX1(ABS(S1),ABS(S2),ABS(S3))*SMALL
            IF(S1+DELTA.GE.S2.AND.S2+DELTA.GE.S3)THEN
C converged stress is in the same sextant as trial stress -> 2-vector
C return to edge is valid.
              EDGE=.TRUE.
              P=(S1+S2+S3)*R1D3
              GOTO 70
            ELSE
C converged stress is not in the same sextant -> 2-vector return to edge
C is not valid. Go to two-vector return map to APEX
              GOTO 50
            ENDIF
          ENDIF
   40   CONTINUE
C failure of stress update procedure
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0003')
        GOTO 999
   50   CONTINUE
C Apply multi-vector return mapping to APEX
C ---------------------------------------
C Check conditions for which return to apex does not make sense
        IF(SINPHI.EQ.R0)CALL ERRPRT('EE0009')
        IF(SINPSI.EQ.R0)CALL ERRPRT('EE0010')
```

```
C Set initial guess for volumetric plastic strain increment DEPV
        DEPV=R0
        EPBAR=EPBARN
        COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
        COTPHI=COSPHI/SINPHI
        RES=COTPHI*COHE-PT
C Newton-Raphson iterations for DEPV
        DO 60 NRITER=1,MXITER
          DENOM=COSPHI*COTPHI/SINPSI*DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))+
     1          BULK
          DDEPV=-RES/DENOM
          DEPV=DEPV+DDEPV
          EPBAR=EPBARN+COSPHI/SINPSI*DEPV
          COHE=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          P=PT-BULK*DEPV
          RES=COTPHI*COHE-P
C check for convergence
          RESNOR=ABS(RES)
          IF(PT.NE.R0)RESNOR=RESNOR/ABS(PT)
          IF(RESNOR.LE.TOL)THEN
            APEX=.TRUE.
            DGAMA=DEPV
            DGAMB=R0
C update principal stresses
            S1=P
            S2=P
            S3=P
            GOTO 70
          ENDIF
   60   CONTINUE
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0003')
        GOTO 999
   70   CONTINUE
C update internal variable EPBAR  and stress components
C -----------------------------------------------------
        RSTAVA(MSTRE+1)=EPBAR
        PSTRS(II)=S1
        PSTRS(JJ)=S3
        PSTRS(MM)=S2
        STRES(1)=PSTRS(1)*EIGPRJ(1,1)+PSTRS(2)*EIGPRJ(1,2)
        STRES(2)=PSTRS(1)*EIGPRJ(2,1)+PSTRS(2)*EIGPRJ(2,2)
        STRES(3)=PSTRS(1)*EIGPRJ(3,1)+PSTRS(2)*EIGPRJ(3,2)
        STRES(4)=PSTRS(3)
C and elastic engineering strain
        EEVD3=P/BULK*R1D3
        RSTAVA(1)=(STRES(1)-P)/R2G+EEVD3
        RSTAVA(2)=(STRES(2)-P)/R2G+EEVD3
        RSTAVA(3)=STRES(3)/GMODU
        RSTAVA(4)=(STRES(4)-P)/R2G+EEVD3
      ELSE
C Elastic step: update stress using linear elastic law
C ====================================================
        STRES(1)=STREST(1)
        STRES(2)=STREST(2)
        STRES(3)=STREST(3)
        STRES(4)=PSTRS(3)
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=STRAT(4)
      ENDIF
  999 CONTINUE
C Update algorithmic variables before exit
C ========================================
      DGAM(1)=DGAMA
      DGAM(2)=DGAMB
      LALGVA(1)=IFPLAS
      LALGVA(2)=SUFAIL
      LALGVA(3)=EDGE
      LALGVA(4)=RIGHT
      LALGVA(5)=APEX
      RETURN
      END
```

```
      SUBROUTINE SUOGD
     1(   B          ,IPROPS     ,NTYPE      ,RPROPS     ,RSTAVA    ,
     2    STRES      ,THICK      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPOGDC=2)
      LOGICAL DUMMY
      PARAMETER
     1(   MCOMP=4    ,MSTRE=4    ,NDIM=2     )
      DIMENSION
     1    B(MCOMP)           ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE)      ,STRES(MSTRE)
      DIMENSION
     1    EIGPRJ(MCOMP,NDIM) ,EIGB(NDIM)         ,PSTRES(3)          ,
     2    PSTRTC(3)
      DATA  R1   ,R3   /
     1      1.0D0,3.0D0/
C***********************************************************************
C STRESS UPDATE PROCEDURE FOR OGDEN TYPE HYPERELASTIC MATERIAL MODEL.
C PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 13.5.1
C            Box 13.1
C***********************************************************************
C Retrieve Ogden material constants
C ===================================
C Number of terms in Ogden's strain-energy function
      NOGTRM=IPROPS(3)
C Bulk modulus
      BULK=RPROPS(IPOGDC+NOGTRM*2)
C Compute principal stretches
C ===========================
C Perform spectral decomposition of the left Cauchy-Green tensor B
      CALL SPDEC2
     1(   EIGPRJ     ,EIGB       ,DUMMY      ,B          )
C Compute in-plane principal stretches
      PSTRTC(1)=SQRT(EIGB(1))
      PSTRTC(2)=SQRT(EIGB(2))
C...and out-of-plane stretches
      IF(NTYPE.EQ.1)THEN
        PSTRTC(3)=R1/(PSTRTC(1)*PSTRTC(2))
      ELSEIF(NTYPE.EQ.2)THEN
        PSTRTC(3)=R1
      ELSEIF(NTYPE.EQ.3)THEN
        PSTRTC(3)=SQRT(B(4))
      ENDIF
C Compute principal Kirchhoff stresses
C ====================================
      CALL RVZERO(PSTRES,3)
      IF(NTYPE.EQ.1) THEN
C Plane stress: Exact incompressibility assumed
C ---------------------------------------------
        DO 10 I=1,NOGTRM
          CMU=RPROPS(IPOGDC-1+I*2-1)
          ALPHA=RPROPS(IPOGDC-1+I*2)
          PSTRES(1)=PSTRES(1)+CMU*(PSTRTC(1)**ALPHA-
     1              (PSTRTC(1)*PSTRTC(2))**(-ALPHA))
          PSTRES(2)=PSTRES(2)+CMU*(PSTRTC(2)**ALPHA-
     1              (PSTRTC(1)*PSTRTC(2))**(-ALPHA))
   10   CONTINUE
        DETF=R1
      ELSE IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
C Plane strain and axisymmetric: Regularised Ogden constitutive law
C -----------------------------------------------------------------
C Compute principal deviatoric Kirchhoff stresses
        R1D3=R1/R3
        DETF=PSTRTC(1)*PSTRTC(2)
        IF(NTYPE.EQ.3)DETF=DETF*PSTRTC(3)
        DO 20 I=1,NOGTRM
          CMU=RPROPS(IPOGDC-1+I*2-1)
          ALPHA=RPROPS(IPOGDC-1+I*2)
          FACTOR=R1D3*(PSTRTC(1)**ALPHA+PSTRTC(2)**ALPHA+
     1         PSTRTC(3)**ALPHA)
          FACVOL=DETF**(-ALPHA*R1D3)
          PSTRES(1)=PSTRES(1)+CMU*FACVOL*(PSTRTC(1)**ALPHA-FACTOR)
          PSTRES(2)=PSTRES(2)+CMU*FACVOL*(PSTRTC(2)**ALPHA-FACTOR)
          PSTRES(3)=PSTRES(3)+CMU*FACVOL*(PSTRTC(3)**ALPHA-FACTOR)
   20   CONTINUE
C Add hydrostatic Kirchhoff pressure (incompressibility penalty term)
        PRESS=BULK*LOG(DETF)
        DO 30 I=1,3
          PSTRES(I)=PSTRES(I)+PRESS
  30    CONTINUE
      ENDIF
C Assemble array of Cauchy stress tensor components
C =================================================
      CALL RVZERO(STRES,3)
      R1DDET=R1/DETF
      PSTRES(1)=PSTRES(1)*R1DDET
      PSTRES(2)=PSTRES(2)*R1DDET
      DO 50 ICOMP=1,3
        DO 40 IDIR=1,2
          STRES(ICOMP)=STRES(ICOMP)+PSTRES(IDIR)*EIGPRJ(ICOMP,IDIR)
   40   CONTINUE
   50 CONTINUE
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)STRES(4)=PSTRES(3)*R1DDET
C Update thickness (plane stress only) and store left Cauchy-Green
C tensor components in state variables vector RSTAVA
C ======================================================================
```

```
      RSTAVA(1)=B(1)
      RSTAVA(2)=B(2)
      RSTAVA(3)=B(3)
      IF(NTYPE.EQ.1)THEN
        THICK=THICK*PSTRTC(3)
        RSTAVA(4)=PSTRTC(3)*PSTRTC(3)
      ELSEIF(NTYPE.EQ.2)THEN
        RSTAVA(4)=R1
      ELSEIF(NTYPE.EQ.3)THEN
        RSTAVA(4)=B(4)
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SUPDSC
     1(   DGAM       ,FINCR      ,IPROPS     ,LALGVA     ,NTYPE      ,
     2    RPROPS     ,RSTAVA     ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER
     1(   IPHARD=6   ,IPHVAR=5   ,MTRIAL=5   ,NDIM=2     ,NIPROP=3   ,
     2    NLALGV=6   ,NRALGV=4   ,NRSTAV=5   ,NSTRE=4    ,NSYST=4    )
C Arguments
      LOGICAL
     1    LALGVA(NLALGV)
      DIMENSION
     1    DGAM(NRALGV)       ,FINCR(3,3)         ,IPROPS(NIPROP)     ,
     2    RPROPS(*)          ,RSTAVA(NRSTAV)     ,STRES(NSTRE)
C Local arrays and variables
      LOGICAL
     1    IFPLAS ,NOCONV ,SUFAIL ,S1ACT  ,S2ACT  ,S3ACT  ,S4ACT
      DIMENSION
     1    BEDEV(4)           ,BEISO(3,3)      ,BMATX(NDIM,NDIM,NSYST),
     2    DDGAM(NSYST)       ,DEREXP(3,3,3,3)    ,DS0M0(NDIM,NDIM)   ,
     3    FEISO(2,2)         ,FEN(2,2)           ,FETISO(2,2)        ,
     4    FETRL(2,2)         ,FPILOG(3,3)        ,FPINCI(3,3)        ,
     5    GINV(NSYST,NSYST)  ,GMATX(NSYST,NSYST),IACSET(NSYST,MTRIAL),
     6    IPACT(0:5)         ,NACSYS(MTRIAL)     ,PHI(NSYST)         ,
     7    SCHMID(NSYST)      ,SM0MS0(NDIM,NDIM,NSYST)                ,
     8    S0M0(NDIM,NDIM,NSYST),VECM(NDIM,NSYST) ,VECM0(NDIM,NSYST)  ,
     9    VECS(NDIM,NSYST)   ,VECS0(NDIM,NSYST)
      DATA
     1    IPACT(0) ,IPACT(1) ,IPACT(2) ,IPACT(3) ,IPACT(4) ,IPACT(5) /
     2    4        ,1        ,2        ,3        ,4        ,1        /
      DATA
     1    R0   ,R1   ,R3   ,SMALL ,TOL   /
     2    0.0D0,1.0D0,3.0D0,1.D-10,1.D-08/
      DATA MXITER / 50 /
C***********************************************************************
C STRESS UPDATE PROCEDURE FOR THE ANISOTROPIC PLANAR DOUBLE-SLIP SINGLE
C CRYSTAL ELASTO-PLASTIC MODEL WITH PIECE-WISE LINEAR TAYLOR ISOTROPIC
C HARDENING:
C MULTI-SURFACE TYPE IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM
C BASED ON THE EXPONENTIAL MAP APPROXIMATION OF THE PLASTIC FLOW RULE.
C
C REFERENCE: Section 16.6.2
C            Boxes 16.2-3
C***********************************************************************
C Stops program if not plane strain
      IF(NTYPE.NE.2)CALL ERRPRT('EI0036')
C Initialize some algorithmic and internal variables
      CALL RVZERO(DGAM,NSYST)
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
      S1ACT=.FALSE.
      S2ACT=.FALSE.
      S3ACT=.FALSE.
      S4ACT=.FALSE.
C... hardening internal variable
      HRVARN=RSTAVA(IPHVAR)
      HRVAR=HRVARN
C... elastic deformation gradient
      FEN(1,1)=RSTAVA(1)
      FEN(2,1)=RSTAVA(2)
      FEN(1,2)=RSTAVA(3)
      FEN(2,2)=RSTAVA(4)
C Retrieve material properties
C... neo-Hookean constants
      GMODU=RPROPS(2)
      BULK=RPROPS(3)
C... initial system orientation
      THETA=RPROPS(4)
C... relative angle between systems
      BETA=RPROPS(5)
C... number of sampling points on hardening curve
      NHARD=IPROPS(3)
C Set up initial slip systems vectors
C... system 1:
      VECS0(1,1)=COS(THETA)
      VECS0(2,1)=SIN(THETA)
      VECM0(1,1)=-SIN(THETA)
      VECM0(2,1)=COS(THETA)
C... system 2:
      VECS0(1,2)=COS(THETA+BETA)
      VECS0(2,2)=SIN(THETA+BETA)
      VECM0(1,2)=-SIN(THETA+BETA)
      VECM0(2,2)=COS(THETA+BETA)
C... system 3:
      VECS0(1,3)=-VECS0(1,1)
      VECS0(2,3)=-VECS0(2,1)
      VECM0(1,3)=VECM0(1,1)
      VECM0(2,3)=VECM0(2,1)
C... system 4:
      VECS0(1,4)=-VECS0(1,2)
      VECS0(2,4)=-VECS0(2,2)
      VECM0(1,4)=VECM0(1,2)
      VECM0(2,4)=VECM0(2,2)
C Set some constants
      R1D3=R1/R3
C Compute elastic trial state
C ----------------------------
C Elastic trial deformation gradient
      CALL RVZERO(FETRL,NDIM*NDIM)
      DO 30 I=1,NDIM
        DO 20 J=1,NDIM
          DO 10 K=1,NDIM
            FETRL(I,J)=FETRL(I,J)+FINCR(I,K)*FEN(K,J)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
C Perform isochoric/volumetric split of elastic trial def. grad.
      DETFET=FETRL(1,1)*FETRL(2,2)-FETRL(1,2)*FETRL(2,1)
      VOLFAC=DETFET**(-R1D3)
      FETISO(1,1)=VOLFAC*FETRL(1,1)
      FETISO(2,1)=VOLFAC*FETRL(2,1)
      FETISO(1,2)=VOLFAC*FETRL(1,2)
      FETISO(2,2)=VOLFAC*FETRL(2,2)
C Check plastic consistency
```

```
C -------------------------
C Compute yield functions values
C... elastic push forward of slip-systems vectors
      CALL RVZERO(VECS,NDIM*NSYST)
      CALL RVZERO(VECM,NDIM*NSYST)
      DO 60 I=1,NDIM
        DO 50 J=1,NDIM
          DO 40 ISYST=1,NSYST
            VECS(I,ISYST)=VECS(I,ISYST)+FETISO(I,J)*VECS0(J,ISYST)
            VECM(I,ISYST)=VECM(I,ISYST)+FETISO(I,J)*VECM0(J,ISYST)
   40     CONTINUE
   50   CONTINUE
   60 CONTINUE
C... current resolved yield stress
      RYIELD=PLFUN(HRVAR,NHARD,RPROPS(IPHARD))
C... elastic trial Schmid resolved shear stresses
      DO 70 ISYST=1,NSYST
        SCHMID(ISYST)=GMODU*SCAPRD(VECS(1,ISYST),VECM(1,ISYST),2)
   70 CONTINUE
C Check consistency
C... compute yield functions values and determine set of active systems
C    at elastic trial state
      IACSYS=0
      DO 80 ISYST=1,NSYST
        PHI(ISYST)=SCHMID(ISYST)-RYIELD
        IF(PHI(ISYST)/RYIELD.GT.TOL)THEN
          IFPLAS=.TRUE.
          IACSYS=IACSYS+1
          IACSET(IACSYS,1)=ISYST
        ENDIF
   80 CONTINUE
      NACSYS(1)=IACSYS
C... define the other possible tentative sets of active systems
      IF(NACSYS(1).EQ.1)THEN
        NTENT=3
        NACSYS(2)=2
        IACSET(1,2)=IACSET(1,1)
        IACSET(2,2)=IPACT(IACSET(1,1)-1)
        NACSYS(3)=2
        IACSET(1,3)=IACSET(1,1)
        IACSET(2,3)=IPACT(IACSET(1,1)+1)
      ELSEIF(NACSYS(1).EQ.2)THEN
        NTENT=5
        NACSYS(2)=1
        IACSET(1,2)=IACSET(1,1)
        NACSYS(3)=1
        IACSET(1,3)=IACSET(2,1)
        IF(IACSET(1,1).EQ.1.AND.IACSET(2,1).EQ.4)THEN
          NACSYS(4)=2
          IACSET(1,4)=1
          IACSET(2,4)=2
          NACSYS(5)=2
          IACSET(1,5)=3
          IACSET(2,5)=4
        ELSE
          NACSYS(4)=2
          IACSET(1,4)=IACSET(1,1)
          IACSET(2,4)=IPACT(IACSET(1,1)-1)
          NACSYS(5)=2
          IACSET(1,5)=IACSET(2,1)
          IACSET(2,5)=IPACT(IACSET(2,1)+1)
        ENDIF
      ENDIF
      IF(IFPLAS)THEN
C Plastic step: Apply return mapping
C ==================================
C Loop over the tentative sets of active systems
C ----------------------------------------------
        DO 420 ITENT=1,NTENT
C re-set elastic push-forward of slip systems vectors
          CALL RVZERO(VECS,NDIM*NSYST)
          CALL RVZERO(VECM,NDIM*NSYST)
          DO 110 I=1,NDIM
            DO 100 J=1,NDIM
              DO 90 ISYST=1,NSYST
                VECS(I,ISYST)=VECS(I,ISYST)+FETISO(I,J)*VECS0(J,ISYST)
                VECM(I,ISYST)=VECM(I,ISYST)+FETISO(I,J)*VECM0(J,ISYST)
   90         CONTINUE
  100       CONTINUE
  110     CONTINUE
C re-set hardening variable
          HRVAR=HRVARN
C re-set yield function values at trial state
          RYIELD=PLFUN(HRVAR,NHARD,RPROPS(IPHARD))
C... elastic trial Schmid resolved shear stresses
          DO 120 ISYST=1,NSYST
            SCHMID(ISYST)=GMODU*SCAPRD(VECS(1,ISYST),VECM(1,ISYST),NDIM)
            PHI(ISYST)=SCHMID(ISYST)-RYIELD
  120     CONTINUE
C Start Newton-Raphson iterations for plastic multipliers
          CALL RVZERO(DGAM,NSYST)
          DO 410 NRITER=1,MXITER
            HSLOPE=DPLFUN(HRVAR,NHARD,RPROPS(IPHARD))
            IF(NRITER.EQ.1)CALL RVZERO(FPILOG,9)
            CALL DEXPMP(    DEREXP      ,NOCONV     ,FPILOG     )
            CALL RVZERO(BMATX,NDIM*NDIM*NSYST)
            DO 220 II=1,NACSYS(ITENT)
              ISYST=IACSET(II,ITENT)
              DO 140 I=1,NDIM
                DO 130 J=1,NDIM
                  S0M0(I,J,ISYST)=VECS0(I,ISYST)*VECM0(J,ISYST)
                  SM0MS0(I,J,ISYST)=VECS(I,ISYST)*VECM0(J,ISYST)+
     1                              VECM(I,ISYST)*VECS0(J,ISYST)
  130           CONTINUE
  140         CONTINUE
              CALL RVZERO(DS0M0,NDIM*NDIM)
              DO 180 I=1,NDIM
                DO 170 J=1,NDIM
                  DO 160 K=1,NDIM
                    DO 150 L=1,NDIM
                      DS0M0(I,J)=DS0M0(I,J)+
     1                           DEREXP(I,J,K,L)*S0M0(K,L,ISYST)
  150               CONTINUE
```

```
  160             CONTINUE
  170           CONTINUE
  180         CONTINUE
              DO 210 I=1,NDIM
                DO 200 J=1,NDIM
                  DO 190 K=1,NDIM
                    BMATX(I,J,ISYST)=BMATX(I,J,ISYST)+
     1                               FETISO(I,K)*DS0M0(K,J)
  190             CONTINUE
  200           CONTINUE
  210         CONTINUE
  220       CONTINUE
            DO 240 II=1,NACSYS(ITENT)
              ISYST=IACSET(II,ITENT)
C Compute exact jacobian of non-linear system of equations
              DO 230 JJ=1,NACSYS(ITENT)
                JSYST=IACSET(JJ,ITENT)

                GMATX(II,JJ)=GMODU*
     1             SCAPRD(SM0MS0(1,1,ISYST),BMATX(1,1,JSYST),NDIM*NDIM)+
     2             HSLOPE

  230         CONTINUE
  240       CONTINUE
C Invert jacobian: Note that for the double slip model only one or two
C systems may be active
            IF(NACSYS(ITENT).EQ.1)THEN
              IF(GMATX(1,1).LT.SMALL)THEN
C... jacobian is singular: Try another active set or exit if
C                         all possible sets have already been tried
                GOTO 420
              ENDIF
              GINV(1,1)=R1/GMATX(1,1)
            ELSEIF(NACSYS(ITENT).EQ.2)THEN
              DETG=GMATX(1,1)*GMATX(2,2)-GMATX(1,2)*GMATX(2,1)
              IF(DETG.LT.SMALL)THEN
C... jacobian is singular: Try another active set or exit if
C                         all possible sets have already been tried
                GOTO 420
              ENDIF
              DETGIN=R1/DETG
              GINV(1,1)=GMATX(2,2)*DETGIN
              GINV(2,2)=GMATX(1,1)*DETGIN
              GINV(1,2)=-GMATX(1,2)*DETGIN
              GINV(2,1)=-GMATX(2,1)*DETGIN
            ENDIF
C Apply Newton-Raphson correction to plastic multipliers
            CALL RVZERO(DDGAM,NSYST)
            DO 260 II=1,NACSYS(ITENT)
              ISYST=IACSET(II,ITENT)
              DO 250 JJ=1,NACSYS(ITENT)
                JSYST=IACSET(JJ,ITENT)
                DDGAM(ISYST)=DDGAM(ISYST)+GINV(II,JJ)*PHI(JSYST)
  250         CONTINUE
              DGAM(ISYST)=DGAM(ISYST)+DDGAM(ISYST)
  260       CONTINUE
C Compute inverse of incremental plastic deformation gradient
C... sum up contributions from each active slip system
            CALL RVZERO(FPILOG,9)
            DO 290 II=1,NACSYS(ITENT)
              ISYST=IACSET(II,ITENT)
              DO 280 I=1,NDIM
                DO 270 J=1,NDIM
                  FPILOG(I,J)=FPILOG(I,J)-
     1                        DGAM(ISYST)*VECS0(I,ISYST)*VECM0(J,ISYST)
  270           CONTINUE
  280         CONTINUE
  290       CONTINUE
C... use exponential map to update inverse of incremental Fp
            CALL EXPMAP
     1(   FPINCI     ,NOCONV     ,FPILOG     )
            IF(NOCONV)THEN
C... exponential map algorithm failed: Break loop and exit
              SUFAIL=.TRUE.
              CALL ERRPRT('WE0014')
              GOTO 999
            ENDIF
C Update isochoric component of elastic deformation gradient
            CALL RVZERO(FEISO,4)
            DO 320 I=1,NDIM
              DO 310 J=1,NDIM
                DO 300 K=1,NDIM
                  FEISO(I,J)=FEISO(I,J)+FETISO(I,K)*FPINCI(K,J)
  300           CONTINUE
  310         CONTINUE
  320       CONTINUE
C Update hardening internal variable and yield resolved shear stress
            HRVAR=HRVARN
            DO 330 II=1,NACSYS(ITENT)
              ISYST=IACSET(II,ITENT)
              HRVAR=HRVAR+DGAM(ISYST)
  330       CONTINUE
C Compute yield functions values and check for convergence
C... elastic push forward of all slip-systems vectors
            CALL RVZERO(VECS,NDIM*NSYST)
            CALL RVZERO(VECM,NDIM*NSYST)
            DO 360 I=1,NDIM
              DO 350 J=1,NDIM
                DO 340 ISYST=1,NSYST
                  VECS(I,ISYST)=VECS(I,ISYST)+FEISO(I,J)*VECS0(J,ISYST)
                  VECM(I,ISYST)=VECM(I,ISYST)+FEISO(I,J)*VECM0(J,ISYST)
  340           CONTINUE
  350         CONTINUE
  360       CONTINUE
C... update resolved yield stress
            RYIELD=PLFUN(HRVAR,NHARD,RPROPS(IPHARD))
C... Schmid resolved shear stresses for all systems and corresponding
C    yield function values
            DO 370 ISYST=1,NSYST
              SCHMID(ISYST)=GMODU*
     1                      SCAPRD(VECS(1,ISYST),VECM(1,ISYST),NDIM)
              PHI(ISYST)=SCHMID(ISYST)-RYIELD
  370       CONTINUE
```

```
C... check for convergence
            RESNOR=R0
            DO 380 II=1,NACSYS(ITENT)
              ISYST=IACSET(II,ITENT)
              RESNOR=RESNOR+ABS(PHI(ISYST))
  380       CONTINUE
            RESNOR=RESNOR/RYIELD
            IF(RESNOR.LE.TOL)THEN
C... N-R loop converged: check validity of current solution
              DO 390 ISYST=1,NSYST
                IF(DGAM(ISYST).LT.R0.OR.
     1                             PHI(ISYST)/RYIELD-TOL.GT.R0)THEN
C... current solution is not valid: Try another active set or exit if
C                                   all possible sets have already been
C                                   tried
                  GOTO 420
                ENDIF
  390         CONTINUE
C... Stress updated converged: Break loop to update necessary variables
C                              and exit
              DO 400 II=1,NACSYS(ITENT)
                ISYST=IACSET(II,ITENT)
                IF(ISYST.EQ.1)S1ACT=.TRUE.
                IF(ISYST.EQ.2)S2ACT=.TRUE.
                IF(ISYST.EQ.3)S3ACT=.TRUE.
                IF(ISYST.EQ.4)S4ACT=.TRUE.
  400         CONTINUE
              GOTO 450
            ENDIF
  410     CONTINUE
  420   CONTINUE
C Stress update procedure failed to converge: Break loop and exit
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0015')
        GOTO 999
      ELSE
C Elastic step: Trial state is the actual one
C ===========================================
        DO 440 I=1,NDIM
          DO 430 J=1,NDIM
            FEISO(I,J)=FETISO(I,J)
  430     CONTINUE
  440   CONTINUE
      ENDIF
C Use neo-Hookean law to update stresses
C ======================================
  450 CONTINUE
C Compute elastic left Cauchy-Green tensor
      CALL RVZERO(BEISO,9)
      DO 480 I=1,NDIM
        DO 470 J=1,NDIM
          DO 460 K=1,NDIM
            BEISO(I,J)=BEISO(I,J)+FEISO(I,K)*FEISO(J,K)
  460     CONTINUE
  470   CONTINUE
  480 CONTINUE
      BEISO(3,3)=VOLFAC*VOLFAC
C Hydrostatic pressure
      P=BULK*LOG(DETFET)
C Deviatoric component of isochoric elastic left Cauchy-Green tensor
      TRACE=BEISO(1,1)+BEISO(2,2)+BEISO(3,3)
      BEDEV(1)=BEISO(1,1)-R1D3*TRACE
      BEDEV(2)=BEISO(2,2)-R1D3*TRACE
      BEDEV(3)=BEISO(1,2)
      BEDEV(4)=BEISO(3,3)-R1D3*TRACE
C Update Cauchy stress components
      DETINV=R1/DETFET
      STRES(1)=(GMODU*BEDEV(1)+P)*DETINV
      STRES(2)=(GMODU*BEDEV(2)+P)*DETINV
      STRES(3)=(GMODU*BEDEV(3))*DETINV
      STRES(4)=(GMODU*BEDEV(4)+P)*DETINV
C Update elastic deformation gradient components
      RSTAVA(1)=FEISO(1,1)/VOLFAC
      RSTAVA(2)=FEISO(2,1)/VOLFAC
      RSTAVA(3)=FEISO(1,2)/VOLFAC
      RSTAVA(4)=FEISO(2,2)/VOLFAC
C Store updated hardening variable
      RSTAVA(5)=HRVAR
  999 CONTINUE
C Update some algorithmic variables before exit
C =============================================
      LALGVA(1)=IFPLAS
      LALGVA(2)=SUFAIL
      IF(.NOT.SUFAIL)THEN
C Update active system flags if state update was successful
        LALGVA(3)=S1ACT
        LALGVA(4)=S2ACT
        LALGVA(5)=S3ACT
        LALGVA(6)=S4ACT
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SUTR
     1(   DGAM       ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
C Arguments
      LOGICAL
     1    LALGVA(4)
      DIMENSION
     1    DGAM(2)            ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(MSTRE)       ,STRES(MSTRE)
C Local arrays and variables
      LOGICAL
     1    DUMMY, IFPLAS, RIGHT, SUFAIL, TWOVEC
      DIMENSION
     1    EIGPRJ(MSTRE,2)    ,PSTRS(3)           ,STREST(3)

      DATA
     1    R0   ,R1   ,R2   ,R3   ,R4   ,SMALL ,TOL   /
     2    0.0D0,1.0D0,2.0D0,3.0D0,4.0D0,1.D-10,1.D-10/
      DATA MXITER / 50 /
C***********************************************************************
C STRESS UPDATE PROCEDURE FOR TRESCA TYPE ELASTO-PLASTIC MATERIAL WITH
C PIECE-WISE LINEAR ISOTROPIC HARDENING:
C IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM (Boxes 8.1-3).
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Boxes 8.1-3
C            Section 8.1.2
C***********************************************************************
C Stops program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0029')
C Initialize some algorithmic and internal variables
      DGAMA=R0
      DGAMB=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
      EPBARN=RSTAVA(MSTRE+1)
      EPBAR=EPBARN
C Set some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Set some constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R4G=R4*GMODU
      R1D3=R1/R3
C Compute elastic trial state
C ----------------------------
C Volumetric strain and pressure stress
      EEV=STRAT(1)+STRAT(2)+STRAT(4)
      P=BULK*EEV
C Spectral decomposition of the elastic trial deviatoric stress
      EEVD3=EEV*R1D3
      STREST(1)=R2G*(STRAT(1)-EEVD3)
      STREST(2)=R2G*(STRAT(2)-EEVD3)
      STREST(3)=GMODU*STRAT(3)
      CALL SPDEC2(EIGPRJ,PSTRS,DUMMY,STREST)
      PSTRS(3)=R2G*(STRAT(4)-EEVD3)
C Identify maximum (PSTRS1) and minimum (PSTRS3) principal stresses
      II=1
      JJ=1
      PSTRS1=PSTRS(II)
      PSTRS3=PSTRS(JJ)
      DO 10 I=2,3
        IF(PSTRS(I).GE.PSTRS1)THEN
          II=I
          PSTRS1=PSTRS(II)
        ENDIF
        IF(PSTRS(I).LT.PSTRS3)THEN
          JJ=I
          PSTRS3=PSTRS(JJ)
        ENDIF
   10 CONTINUE
      IF(II.NE.1.AND.JJ.NE.1)MM=1
      IF(II.NE.2.AND.JJ.NE.2)MM=2
      IF(II.NE.3.AND.JJ.NE.3)MM=3
      PSTRS2=PSTRS(MM)
C Compute trial yield function and check for plastic consistency
C ---------------------------------------------------------------
      SHMAXT=PSTRS1-PSTRS3
      SIGMAY=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
      PHIA=SHMAXT-SIGMAY
      IF(PHIA/SIGMAY.GT.TOL)THEN
C Plastic step: Apply return mapping
C ==================================
        IFPLAS=.TRUE.
C identify possible two-vector return: right or left of main plane
        SCAPRD=PSTRS1+PSTRS3-PSTRS2*R2
        IF(SCAPRD.GE.R0)THEN
          RIGHT=.TRUE.
        ELSE
          RIGHT=.FALSE.
        ENDIF
C Apply one-vector return mapping first (return to main plane)
C ------------------------------------------------------------
        TWOVEC=.FALSE.
C Start Newton-Raphson iterations
        DO 20 NRITER=1,MXITER
C Compute residual derivative
          DENOM=-R4G-DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
C Compute Newton-Raphson increment and update variable DGAMA
          DDGAMA=-PHIA/DENOM
          DGAMA=DGAMA+DDGAMA
C Compute new residual
          EPBAR=EPBARN+DGAMA
          SIGMAY=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          SHMAX=SHMAXT-R4G*DGAMA
          PHIA=SHMAX-SIGMAY
C Check convergence
          RESNOR=ABS(PHIA/SIGMAY)
```

```
          IF(RESNOR.LE.TOL)THEN
C Check validity of one-vector return
            S1=PSTRS1-R2G*DGAMA
            S2=PSTRS2
            S3=PSTRS3+R2G*DGAMA
            DELTA=DMAX1(ABS(S1),ABS(S2),ABS(S3))*SMALL
            IF(S1+DELTA.GE.S2.AND.S2+DELTA.GE.S3)THEN
C converged stress is in the same sextant as trial stress -> 1-vector
C return is valid. Update EPBAR and principal deviatoric stresses
              RSTAVA(MSTRE+1)=EPBAR
              PSTRS1=S1
              PSTRS3=S3
              GOTO 50
            ELSE
C 1-vector return is not valid - go to two-vector procedure
              GOTO 30
            ENDIF
          ENDIF
   20   CONTINUE
C failure of stress update procedure
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0001')
        GOTO 999
   30   CONTINUE
C Apply two-vector return mapping (return to corner - right or left)
C -------------------------------------------------------------------
        TWOVEC=.TRUE.
        DGAMA=R0
        DGABAR=R1
        EPBAR=EPBARN
        SIGMAY=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
        SHMXTA=PSTRS1-PSTRS3
        IF(RIGHT)THEN
          SHMXTB=PSTRS1-PSTRS2
        ELSE
          SHMXTB=PSTRS2-PSTRS3
        ENDIF
        PHIA=SHMXTA-SIGMAY
        PHIB=SHMXTB-SIGMAY
C Start Newton-Raphson iterations
        DO 40 NRITER=1,MXITER
C Compute residual derivative
          HSLOPE=DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          DRVAA=-R4G-HSLOPE
          DRVAB=-R2G-HSLOPE
          DRVBA=-R2G-HSLOPE
          DRVBB=-R4G-HSLOPE
C Compute Newton-Raphson increment and update variables DGAMA and DGAMB
          R1DDET=R1/(DRVAA*DRVBB-DRVAB*DRVBA)
          DDGAMA=(-DRVBB*PHIA+DRVAB*PHIB)*R1DDET
          DDGAMB=(DRVBA*PHIA-DRVAA*PHIB)*R1DDET
          DGAMA=DGAMA+DDGAMA
          DGAMB=DGAMB+DDGAMB
C Compute new residual
          DGABAR=DGAMA+DGAMB
          EPBAR=EPBARN+DGABAR
          SIGMAY=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          PHIA=SHMXTA-R2G*(R2*DGAMA+DGAMB)-SIGMAY
          PHIB=SHMXTB-R2G*(DGAMA+R2*DGAMB)-SIGMAY
C Check convergence
          RESNOR=(ABS(PHIA)+ABS(PHIB))/SIGMAY
          IF(RESNOR.LE.TOL)THEN
C Update EPBAR and principal deviatoric stresses
            RSTAVA(MSTRE+1)=EPBAR
            IF(RIGHT)THEN
              PSTRS1=PSTRS1-R2G*(DGAMA+DGAMB)
              PSTRS3=PSTRS3+R2G*DGAMA
              PSTRS2=PSTRS2+R2G*DGAMB
            ELSE
              PSTRS1=PSTRS1-R2G*DGAMA
              PSTRS3=PSTRS3+R2G*(DGAMA+DGAMB)
              PSTRS2=PSTRS2-R2G*DGAMB
            ENDIF
            GOTO 50
          ENDIF
   40   CONTINUE
C failure of stress update procedure
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0001')
        GOTO 999
   50   CONTINUE
C update stress components
C ------------------------
        PSTRS(II)=PSTRS1
        PSTRS(JJ)=PSTRS3
        PSTRS(MM)=PSTRS2
        STRES(1)=PSTRS(1)*EIGPRJ(1,1)+PSTRS(2)*EIGPRJ(1,2)+P
        STRES(2)=PSTRS(1)*EIGPRJ(2,1)+PSTRS(2)*EIGPRJ(2,2)+P
        STRES(3)=PSTRS(1)*EIGPRJ(3,1)+PSTRS(2)*EIGPRJ(3,2)
        STRES(4)=PSTRS(3)+P
C and elastic engineering strain
        RSTAVA(1)=(STRES(1)-P)/R2G+EEVD3
        RSTAVA(2)=(STRES(2)-P)/R2G+EEVD3
        RSTAVA(3)=STRES(3)/GMODU
        RSTAVA(4)=PSTRS(3)/R2G+EEVD3
      ELSE
C Elastic step: update stress using linear elastic law
C ====================================================
        STRES(1)=STREST(1)+P
        STRES(2)=STREST(2)+P
        STRES(3)=STREST(3)
        STRES(4)=PSTRS(3)+P
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=STRAT(4)
      ENDIF
  999 CONTINUE
C Update algorithmic variables before exit
C ========================================
      DGAM(1)=DGAMA
      DGAM(2)=DGAMB
```

```
LALGVA(1)=IFPLAS
LALGVA(2)=SUFAIL
LALGVA(3)=TWOVEC
LALGVA(4)=RIGHT
RETURN
END
```

```
      SUBROUTINE SUTRPN
     1(   DGAM       ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
C Arguments
      LOGICAL
     1    LALGVA(4)
      DIMENSION
     1    DGAM(2)            ,IPROPS(*)          ,RPROPS(*)          ,
     2    RSTAVA(MSTRE+1)    ,STRAT(*)           ,STRES(*)
C Local arrays and variables
      LOGICAL EPFLAG, IFPLAS, SUFAIL
      DIMENSION
     1    DMATX(MSTRE,MSTRE) ,RSTAUX(MSTRE+1)
      DATA
     1    R0    ,TOL   /
     2    0.D0  ,1.D-08/
      DATA MXITER / 20 /
C***********************************************************************
C STRESS UPDATE PROCEDURE FOR TRESCA TYPE ELASTO-PLASTIC MATERIAL WITH
C PIECE-WISE LINEAR ISOTROPIC HARDENING IN PLANE STRESS. NESTED
C ITERATION APPROACH.
C
C REFERENCE: Section 9.2.2
C***********************************************************************
C Stop program if not plane stress
      IF(NTYPE.NE.1)CALL ERRPRT('EI0060')
C Initialise state update failure flag
      SUFAIL=.FALSE.
C Set some material properties
      NHARD=IPROPS(3)
C
C Newton-Raphson iteration loop for plane stress enforcement
C
C Set initial guess for elastic trial thickness strain. Use previously
C converged elastic trial thickness strain.
      E33TRL=RSTAVA(4)
C Set axisymmetric state flag
      NTYPAX=3
C Start N-R loop
      DO 20 ITER=1,MXITER
C Set state variables to values at beginning of increment
        DO 10 I=1,MSTRE+1
          RSTAUX(I)=RSTAVA(I)
   10   CONTINUE
C Use axisymmetric integration algorithm to compute stresses
        STRAT(4)=E33TRL
        CALL SUTR
     1(   DGAM       ,IPROPS     ,LALGVA     ,NTYPAX     ,RPROPS     ,
     2    RSTAUX     ,STRAT      ,STRES      )
        SUFAIL=LALGVA(2)
C ...emergency exit in case of failure of state update procedure
        IF(SUFAIL)THEN
          GOTO 999
        ENDIF
        IFPLAS=LALGVA(1)
C Check plane stress convergence
        EPBAR=RSTAUX(MSTRE+1)
        SIGMAY=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
        RES=ABS(STRES(4))
C ...use normalised out-of-plane stress
        IF(SIGMAY.NE.R0)RES=RES/SIGMAY
        IF(RES.LE.TOL)THEN
C ...and break N-R loop in case of convergence
          GOTO 30
        ENDIF
C Compute axisymmetric consistent tangent components
        EPFLAG=IFPLAS
        CALL CTTR
     1(   DMATX      ,EPFLAG     ,IPROPS     ,LALGVA     ,NTYPAX     ,
     2    RPROPS     ,RSTAUX     ,STRAT      ,STRES      )
C Apply Newton-Raphson correction to normal elastic trial strain
        D22=DMATX(4,4)
        E33TRL=E33TRL-STRES(4)/D22
   20 CONTINUE
C Emergency exit in case of failure of plane stress enforcement loop
      SUFAIL=.TRUE.
      CALL ERRPRT('WE0026')
      GOTO 999
   30 CONTINUE
C Set state variables to current updated values
      DO 40 I=1,MSTRE+1
        RSTAVA(I)=RSTAUX(I)
   40 CONTINUE
      RSTAVA(4)=E33TRL
C
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SUVM
     1(   DGAMA      ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
      LOGICAL IFPLAS, LALGVA(2), SUFAIL
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)          ,RSTAVA(MSTRE+1)    ,
     2    STRAT(MSTRE)       ,STRES(MSTRE)
      DIMENSION
     1    EET(MSTRE)
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,R3   ,TOL   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,3.0D0,1.D-06/
      DATA MXITER / 50 /
C***********************************************************************
C STATE UPDATE PROCEDURE FOR THE VON MISES ELASTO-PLASTIC MATERIAL MODEL
C WITH NON-LINEAR (PIECEWISE LINEAR) ISOTROPIC HARDENING:
C IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM (BOXES 7.3-4).
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS.
C
C REFERENCE: Section 7.3.5
C***********************************************************************
C Stop program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0013')
C Initialise some algorithmic and internal variables
      DGAMA=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
      EPBARN=RSTAVA(MSTRE+1)
C Set some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Shear and bulk moduli and other necessary constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R3G=R3*GMODU
C Elastic predictor: Compute elastic trial state
C ----------------------------------------------
C Volumetric strain and pressure stress
      EEV=STRAT(1)+STRAT(2)+STRAT(4)
      P=BULK*EEV
C Elastic trial deviatoric strain
      EEVD3=EEV/R3
      EET(1)=STRAT(1)-EEVD3
      EET(2)=STRAT(2)-EEVD3
      EET(4)=STRAT(4)-EEVD3
C Convert engineering shear component into physical component
      EET(3)=STRAT(3)/R2
C Compute trial effective stress and uniaxial yield stress
      VARJ2T=R2G*R2G*(EET(3)*EET(3)+RP5*(EET(1)*EET(1)+
     1                     EET(2)*EET(2)+EET(4)*EET(4)))
      QTRIAL=SQRT(R3*VARJ2T)
      SIGMAY=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
C Check for plastic admissibility
C ------------------------------
      PHI=QTRIAL-SIGMAY
      IF(PHI/SIGMAY.GT.TOL)THEN
C Plastic step: Apply return mapping - use Newton-Raphson algorithm
C               to solve the return mapping equation (Box 7.4)
C ---------------------------------------------------------------------
        IFPLAS=.TRUE.
        EPBAR=EPBARN
        DO 10 NRITER=1,MXITER
C Compute residual derivative
          DENOM=-R3G-DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
C Compute Newton-Raphson increment and update variable DGAMA
          DDGAMA=-PHI/DENOM
          DGAMA=DGAMA+DDGAMA
C Compute new residual
          EPBAR=EPBAR+DDGAMA
          SIGMAY=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          PHI=QTRIAL-R3G*DGAMA-SIGMAY
C Check convergence
          RESNOR=ABS(PHI/SIGMAY)
          IF(RESNOR.LE.TOL)THEN
C update accumulated plastic strain
            RSTAVA(MSTRE+1)=EPBAR
C update stress components
            FACTOR=R2G*(R1-R3G*DGAMA/QTRIAL)
            STRES(1)=FACTOR*EET(1)+P
            STRES(2)=FACTOR*EET(2)+P
            STRES(3)=FACTOR*EET(3)
            STRES(4)=FACTOR*EET(4)+P
C compute converged elastic (engineering) strain components
            FACTOR=FACTOR/R2G
            RSTAVA(1)=FACTOR*EET(1)+EEVD3
            RSTAVA(2)=FACTOR*EET(2)+EEVD3
            RSTAVA(3)=FACTOR*EET(3)*R2
            RSTAVA(4)=FACTOR*EET(4)+EEVD3
            GOTO 999
          ENDIF
   10   CONTINUE
C reset failure flag and print warning message if the algorithm fails
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0004')
      ELSE
C Elastic step: Update stress using linear elastic law
C ---------------------------------------------------
        STRES(1)=R2G*EET(1)+P
        STRES(2)=R2G*EET(2)+P
        STRES(3)=R2G*EET(3)
        STRES(4)=R2G*EET(4)+P
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=STRAT(4)
      ENDIF
  999 CONTINUE
C Update some algorithmic variables before exit
```

```
LALGVA(1)=IFPLAS
LALGVA(2)=SUFAIL
RETURN
END
```

```
      SUBROUTINE SUVMMX
     1(   DGAMA      ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(IPHARD=4  ,MSTRE=4)
C Arguments
      LOGICAL LALGVA(2)
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)          ,RSTAVA(2*MSTRE+1)  ,
     2    STRAT(MSTRE)       ,STRES(MSTRE)
C Local arrays and variables
      LOGICAL IFPLAS, SUFAIL
      DIMENSION
     1    BACSTN(MSTRE)      ,BACSTR(MSTRE)      ,EET(MSTRE)         ,
     2    ETATRL(MSTRE)      ,FLOVEC(MSTRE)      ,STRIAL(MSTRE)
      DATA
     1    R0   ,R1   ,R2   ,R3   ,TOL   /
     2    0.0D0,1.0D0,2.0D0,3.0D0,1.D-08/
      DATA MXITER / 50 /
C***********************************************************************
C STATE UPDATE PROCEDURE FOR THE VON MISES ELASTO-PLASTIC MATERIAL MODEL
C WITH NON-LINEAR (PIECEWISE LINEAR) MIXED ISOTROPIC/KINEMATIC
C HARDENING:
C IMPLICIT ELASTIC PREDICTOR/RETURN MAPPING ALGORITHM (Box 7.5).
C PLANE STRAIN AND AXISYMMETRIC IMPLEMENTATIONS ONLY.
C
C REFERENCE: Box 7.5
C***********************************************************************
C Stop program if neither plane strain nor axisymmetric state
      IF(NTYPE.NE.2.AND.NTYPE.NE.3)CALL ERRPRT('EI0047')
C Retrieve some previous converged state variables
C... accumulated plastic strain (hardening internal variable)
      EPBARN=RSTAVA(5)
C... backstress tensor components
      BACSTN(1)=RSTAVA(6)
      BACSTN(2)=RSTAVA(7)
      BACSTN(3)=RSTAVA(8)
      BACSTN(4)=RSTAVA(9)
C Initialise some algorithmic and internal variables
      DGAMA=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
C Retrieve some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Set pointers to isotropic and kinematic hardening curves
      IPIHAR=IPHARD
      IPKHAR=IPHARD+2*NHARD
C Shear and bulk moduli and other necessary constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R3G=R3*GMODU
C Elastic predictor: Compute elastic trial state [item(i) Box 7.5]
C ----------------------------------------------------------------
C Volumetric strain and pressure stress
      EEV=STRAT(1)+STRAT(2)+STRAT(4)
      P=BULK*EEV
C Elastic trial deviatoric strain
      EEVD3=EEV/R3
      EET(1)=STRAT(1)-EEVD3
      EET(2)=STRAT(2)-EEVD3
      EET(4)=STRAT(4)-EEVD3
C Convert engineering shear component into physical component
      EET(3)=STRAT(3)/R2
C Compute trial deviatoric stress and trial relative stress
      DO 10 ISTRE=1,MSTRE
        STRIAL(ISTRE)=R2G*EET(ISTRE)
        ETATRL(ISTRE)=STRIAL(ISTRE)-BACSTN(ISTRE)
   10 CONTINUE
C Compute trial effective relative stress
      QBARTR=SQRT(R3/R2*(ETATRL(1)**2+ETATRL(2)**2+R2*ETATRL(3)**2+
     1            ETATRL(4)**2))
C and radius of von Mises cylinder
      SIGMAY=PLFUN(EPBARN,NHARD,RPROPS(IPIHAR))
C Check for plastic admissibility [item (ii) Box 7.5]
C ----------------------------------------------------
      PHI=QBARTR-SIGMAY
      IF(PHI/SIGMAY.GT.TOL)THEN
C Plastic step: Apply return mapping - use Newton-Raphson algorithm to
C               solve the return mapping equation [item (iii) Box 7.5]
C ---------------------------------------------------------------------
        IFPLAS=.TRUE.
        EPBAR=EPBARN
        BETBAN=PLFUN(EPBARN,NHARD,RPROPS(IPKHAR))
        DO 40 NRITER=1,MXITER
C Compute residual derivative
          HISLOP=DPLFUN(EPBAR,NHARD,RPROPS(IPIHAR))
          HKSLOP=DPLFUN(EPBAR,NHARD,RPROPS(IPKHAR))
          DENOM=-R3G-HISLOP-HKSLOP
C Compute Newton-Raphson increment and update variable DGAMA
          DDGAMA=-PHI/DENOM
          DGAMA=DGAMA+DDGAMA
C Compute new residual
          EPBAR=EPBAR+DDGAMA
          SIGMAY=PLFUN(EPBAR,NHARD,RPROPS(IPIHAR))
          BETBAR=PLFUN(EPBAR,NHARD,RPROPS(IPKHAR))
          PHI=QBARTR-R3G*DGAMA-BETBAR+BETBAN-SIGMAY
C Check convergence
          RESNOR=ABS(PHI/SIGMAY)
          IF(RESNOR.LE.TOL)THEN
C update stress components
            ETANOR=SQRT(ETATRL(1)**2+ETATRL(2)**2+R2*ETATRL(3)**2+
     1                  ETATRL(4)**2)
            FACTOR=SQRT(R3/R2)/ETANOR
            DO 20 ISTRE=1,MSTRE
              FLOVEC(ISTRE)=FACTOR*ETATRL(ISTRE)
   20       CONTINUE
            FACTOR=R2G*DGAMA
            STRES(1)=STRIAL(1)-FACTOR*FLOVEC(1)+P
            STRES(2)=STRIAL(2)-FACTOR*FLOVEC(2)+P
            STRES(3)=STRIAL(3)-FACTOR*FLOVEC(3)
```

```
            STRES(4)=STRIAL(4)-FACTOR*FLOVEC(4)+P
C compute and store converged elastic (engineering) strain components
            RSTAVA(1)=EET(1)-DGAMA*FLOVEC(1)+EEVD3
            RSTAVA(2)=EET(2)-DGAMA*FLOVEC(2)+EEVD3
            RSTAVA(3)=R2*(EET(3)-DGAMA*FLOVEC(3))
            RSTAVA(4)=EET(4)-DGAMA*FLOVEC(4)+EEVD3
C store updated accumulated plastic strain
            RSTAVA(5)=EPBAR
C compute and store updated backstress tensor components
            FACTOR=R2/R3*(BETBAR-BETBAN)
            DO 30 ISTRE=1,MSTRE
              BACSTR(ISTRE)=BACSTN(ISTRE)+FACTOR*FLOVEC(ISTRE)
   30       CONTINUE
            RSTAVA(6)=BACSTR(1)
            RSTAVA(7)=BACSTR(2)
            RSTAVA(8)=BACSTR(3)
            RSTAVA(9)=BACSTR(4)
            GOTO 999
          ENDIF
   40   CONTINUE
C reset failure flag and print warning message if the algorithm fails
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0017')
      ELSE
C Elastic step: Update stress using linear elastic law
C ------------------------------------------------------
        STRES(1)=R2G*EET(1)+P
        STRES(2)=R2G*EET(2)+P
        STRES(3)=R2G*EET(3)
        STRES(4)=R2G*EET(4)+P
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=STRAT(4)
      ENDIF
  999 CONTINUE
C Update some algorithmic variables before exit
      LALGVA(1)=IFPLAS
      LALGVA(2)=SUFAIL
      RETURN
      END
```

```
      SUBROUTINE SUVMPS
     1(   DGAMA      ,IPROPS     ,LALGVA     ,NTYPE      ,RPROPS     ,
     2    RSTAVA     ,STRAT      ,STRES      )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER( IPHARD=4  ,MSTRE=4  ,NSTRE=3 )
      LOGICAL IFPLAS, LALGVA(2), SUFAIL
      DIMENSION
     1    IPROPS(*)          ,RPROPS(*)          ,RSTAVA(MSTRE+1)    ,
     2    STRAT(MSTRE)       ,STRES(MSTRE)
      DIMENSION
     1    EET(MSTRE)         ,STREST(NSTRE)
      DATA
     1    R0   ,RP5  ,R1   ,R2   ,R3   ,R4   ,R6   ,TOL   /
     2    0.0D0,0.5D0,1.0D0,2.0D0,3.0D0,4.0D0,6.0D0,1.D-08/
      DATA MXITER / 50 /
C***********************************************************************
C STATE UPDATE PROCEDURE FOR THE VON MISES ELASTO-PLASTIC MODEL WITH
C NON-LINEAR (PIECEWISE LINEAR) ISOTROPIC HARDENING IN PLANE STRESS:
C IMPLICIT PLANE STRESS-PROJECTED ELASTIC PREDICTOR/RETURN MAPPING
C ALGORITHM (BOXES 9.4-5).
C
C REFERENCE: Section 9.4.3
C            Boxes 9.4-5
C***********************************************************************
C Stop program if not plane stress
      IF(NTYPE.NE.1)CALL ERRPRT('EI0031')
C Initialise some algorithmic and internal variables
      DGAMA=R0
      IFPLAS=.FALSE.
      SUFAIL=.FALSE.
C...set previously (equilibrium) converged accumulated plastic strain
      EPBARN=RSTAVA(MSTRE+1)
C Set some material properties
      YOUNG=RPROPS(2)
      POISS=RPROPS(3)
      NHARD=IPROPS(3)
C Shear and bulk moduli and other necessary constants
      GMODU=YOUNG/(R2*(R1+POISS))
      BULK=YOUNG/(R3*(R1-R2*POISS))
      R2G=R2*GMODU
      R4G=R4*GMODU
      R1D3=R1/R3
      R1D6=R1/R6
      R2D3=R2*R1D3
      SQR2D3=SQRT(R2D3)
      R4GD3=R4G*R1D3
C Elastic predictor: Compute elastic trial state
C ----------------------------------------------
C Volumetric strain
      FACTOR=R2G/(BULK+R4GD3)
      EEV=(STRAT(1)+STRAT(2))*FACTOR
C Elastic trial deviatoric strain
      EEVD3=EEV/R3
      EET(1)=STRAT(1)-EEVD3
      EET(2)=STRAT(2)-EEVD3
C Convert engineering shear component into physical component
      EET(3)=STRAT(3)*RP5
C Elastic trial stress components
      PT=BULK*EEV
      STREST(1)=R2G*EET(1)+PT
      STREST(2)=R2G*EET(2)+PT
      STREST(3)=R2G*EET(3)
C Compute yield function value at trial state
      A1=(STREST(1)+STREST(2))*(STREST(1)+STREST(2))
      A2=(STREST(2)-STREST(1))*(STREST(2)-STREST(1))
      A3=STREST(3)*STREST(3)
      XI=R1D6*A1+RP5*A2+R2*A3
      SIGMAY=PLFUN(EPBARN,NHARD,RPROPS(IPHARD))
C...yield function
      PHI=RP5*XI-R1D3*SIGMAY*SIGMAY
C Check for plastic admissibility
C -------------------------------
      IF(PHI/SIGMAY.GT.TOL)THEN
C Plastic step: Apply return mapping - use Newton-Raphson algorithm
C               to solve the plane stress-projected return mapping
C               equation for the plastic multiplier (Box 9.5)
C -----------------------------------------------------------------
        IFPLAS=.TRUE.
        EPBAR=EPBARN
        SQRTXI=SQRT(XI)
        B1=R1
        B2=R1
        FMODU=YOUNG/(R3*(R1-POISS))
        DO 10 NRITER=1,MXITER
C Compute residual derivative
          HSLOPE=DPLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          DXI=-A1*FMODU/(R3*B1*B1*B1)-R2G*(A2+R4*A3)/(B2*B2*B2)
          HBAR=R2*SIGMAY*HSLOPE*SQR2D3*(SQRTXI+DGAMA*DXI/(R2*SQRTXI))
          DPHI=RP5*DXI-R1D3*HBAR
C Compute Newton-Raphson increment and update equation variable DGAMA
          DGAMA=DGAMA-PHI/DPHI
C Compute new residual (yield function value)
          B1=R1+FMODU*DGAMA
          B2=R1+R2G*DGAMA
          XI=R1D6*A1/(B1*B1)+(RP5*A2+R2*A3)/(B2*B2)
          SQRTXI=SQRT(XI)
          EPBAR=EPBARN+DGAMA*SQR2D3*SQRTXI
          SIGMAY=PLFUN(EPBAR,NHARD,RPROPS(IPHARD))
          PHI=RP5*XI-R1D3*SIGMAY*SIGMAY
C Check for convergence
          RESNOR=ABS(PHI/SIGMAY)
          IF(RESNOR.LE.TOL)THEN
C update accumulated plastic strain
            RSTAVA(MSTRE+1)=EPBAR
C update stress components:   sigma := A sigma^trial
            ASTAR1=R3*(R1-POISS)/(R3*(R1-POISS)+YOUNG*DGAMA)
            ASTAR2=R1/(R1+R2G*DGAMA)
            A11=RP5*(ASTAR1+ASTAR2)
            A22=A11
            A12=RP5*(ASTAR1-ASTAR2)
            A21=A12
            A33=ASTAR2
            STRES(1)=A11*STREST(1)+A12*STREST(2)
```

```
            STRES(2)=A21*STREST(1)+A22*STREST(2)
            STRES(3)=A33*STREST(3)
C compute corresponding elastic (engineering) strain components
            FACTG=R1/R2G
            P=R1D3*(STRES(1)+STRES(2))
            EEV=P/BULK
            EEVD3=R1D3*EEV
            RSTAVA(1)=FACTG*(R2D3*STRES(1)-R1D3*STRES(2))+EEVD3
            RSTAVA(2)=FACTG*(R2D3*STRES(2)-R1D3*STRES(1))+EEVD3
            RSTAVA(3)=FACTG*STRES(3)*R2
            RSTAVA(4)=-POISS/(R1-POISS)*(RSTAVA(1)+RSTAVA(2))
            GOTO 999
          ENDIF
   10   CONTINUE
C reset failure flag and print warning message if N-R algorithm fails
        SUFAIL=.TRUE.
        CALL ERRPRT('WE0013')
      ELSE
C Elastic step: Update stress using linear elastic law
C ---------------------------------------------------
        STRES(1)=STREST(1)
        STRES(2)=STREST(2)
        STRES(3)=STREST(3)
C elastic engineering strain
        RSTAVA(1)=STRAT(1)
        RSTAVA(2)=STRAT(2)
        RSTAVA(3)=STRAT(3)
        RSTAVA(4)=-POISS/(R1-POISS)*(STRAT(1)+STRAT(2))
      ENDIF
  999 CONTINUE
C Update some algorithmic variables before exit
      LALGVA(1)=IFPLAS
      LALGVA(2)=SUFAIL
      RETURN
      END
```

```
      SUBROUTINE SWDAMA
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC             ,LALGVL
      DIMENSION
     1    LALGVC(*)          ,LALGVL(*)          ,RALGVC(*)          ,
     2    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     3    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR LEMAITRE'S DUCTILE DAMAGE ELASTO-PLASTIC
C MODEL WITH ISOTROPIC HARDENING
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=6
      ENDIF
      NRALGV=1
      NLALGV=2
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the infinitesimal elastic egineering strain tensor
C components (logarithmic strains inlarge strains), the hardening
C variable and damage variable. All initialised to zero.
        CALL RVZERO(RSTAVC,NRSTAV)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multiplier before starting a new increment
          CALL RVZERO(RALGVC,NRALGV)
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multiplier before starting a new increment
            CALL RVZERO(RALGVC,NRALGV)
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWDMEL
     1(   MODE       ,NTYPE      ,RSTAVC     ,RSTAVL     ,STRESC     ,
     2    STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      DIMENSION
     1    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     2    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE DAMAGED ELASTIC MATERIAL MODEL WITH
C PARTIAL MICROCRACK/VOID CLOSURE EFFECTS
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
      ELSE
        CALL ERRPRT('EI0054')
      ENDIF
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
C RSTAVA stores the infinitesimal egineering strain tensor components
C (logarithmic strains in large strains)
        CALL RVZERO(RSTAVC,NSTRE)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 10 ISTRE=1,NSTRE
            STRESL(ISTRE)=STRESC(ISTRE)
            RSTAVL(ISTRE)=RSTAVC(ISTRE)
   10     CONTINUE
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 20 ISTRE=1,NSTRE
            STRESC(ISTRE)=STRESL(ISTRE)
            RSTAVC(ISTRE)=RSTAVL(ISTRE)
   20     CONTINUE
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWDP
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC             ,LALGVL
      DIMENSION
     1    LALGVC(*)          ,LALGVL(*)          ,RALGVC(*)          ,
     2    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     3    STRESL(*)
      DATA R0   /
     1     0.0D0/
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE DRUCKER-PRAGER MATERIAL MODEL
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.1)THEN
        NSTRE=4
        NRSTAV=5
        NRALGV=3
      ELSEIF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=5
        NRALGV=2
      ENDIF
      NLALGV=3
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the infinitesimal egineering elastic strain tensor
C (engineering logarithmic strains in large strains) and the effective
C plastic strain
        CALL RVZERO(RSTAVC,NRSTAV)
      ELSE
C Switching modes
C ===============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multipliers before starting a new increment
          RALGVC(1)=R0
          RALGVC(2)=R0
          IF(NTYPE.EQ.1)THEN
C Reset elastic trial thickness strain to the current (converged) value
C of the elastic thickness strain (used by the nested iteration plane
C stress algorithm only)
            RALGVC(3)=RSTAVC(4)
          ENDIF
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multipliers before starting a new increment
            RALGVC(1)=R0
            RALGVC(2)=R0
            IF(NTYPE.EQ.1)THEN
C Reset elastic trial thickness strain to the last converged value of
C the elastic thickness strain (used by the nested iteration plane
C stress algorithm only)
              RALGVC(3)=RSTAVL(4)
            ENDIF
          ENDIF
        ENDIF
      ENDIF
```

```
RETURN
END
```

```
      SUBROUTINE SWEL
     1(   MODE       ,NTYPE      ,RSTAVC     ,RSTAVL     ,STRESC     ,
     2    STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      DIMENSION
     1    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     2    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE ELASTIC MATERIAL MODEL
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)NSTRE=4
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
C RSTAVA stores the infinitesimal egineering strain tensor components
C in small strains and the logarithmic eng. strains in large strain
C analysis
        CALL RVZERO(RSTAVC,NSTRE)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 10 ISTRE=1,NSTRE
            STRESL(ISTRE)=STRESC(ISTRE)
            RSTAVL(ISTRE)=RSTAVC(ISTRE)
   10     CONTINUE
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 20 ISTRE=1,NSTRE
            STRESC(ISTRE)=STRESL(ISTRE)
            RSTAVC(ISTRE)=RSTAVL(ISTRE)
   20     CONTINUE
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWITCH( MODE )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      DATA R0 /0.0D0 /
C***********************************************************************
C SWITCHES STATE VARIABLES VALUES AND COORDINATES BETWEEN CURRENT
C AND LAST CONVERGED SOLUTION DURING GLOBAL EQUILIBRIUM ITERATIONS
C
C    MODE=1 -> Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C                             -----------
C
C    MODE=2 -> Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).  -------------
C
C
C    MODE=3 -> Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C                                       -----------------
C
C NOTE THAT FOR GAUSS POINT THICKNESS THE INITIAL VALUE IS NEEDED.
C FOR THIS VARIABLE, APPROPRIATE ASSIGNEMENTS ARE MADE WHICH DIFFER
C FROM THE CONVENTIONAL ONES
C
C REFERENCE: Section 5.4.6
C            Figure 5.4
C***********************************************************************
      IF(MODE.EQ.1)THEN
        NTI=1
        NTO=2
        NTITHK=1
        NTOTHK=2
      ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
        NTI=2
        NTO=1
        NTITHK=0
        NTOTHK=1
      ENDIF
C
C
C Set stress, other state variables, thickness and algorithmic variables
C =======================================================================
C
      DO 20 IELEM=1,NELEM
        IGRUP =IGRPID(IELEM)
        IELIDN=IELTID(IGRUP)
        NGAUSP=IELPRP(4,IELIDN)
        DO 10 IGAUSP=1,NGAUSP
C
C Call material interface to switch Gauss point data (state and
C algorithmic variables)
          CALL MATISW
     1(   MODE        ,NLARGE     ,NTYPE       ,IPROPS(1,MATTID(IGRUP)),
     2    LALGVA(1,IGAUSP,IELEM,1)      ,LALGVA(1,IGAUSP,IELEM,2)      ,
     3    RALGVA(1,IGAUSP,IELEM,1)      ,RALGVA(1,IGAUSP,IELEM,2)      ,
     4    RPROPS(1,MATTID(IGRUP))       ,RSTAVA(1,IGAUSP,IELEM,1)      ,
     5    RSTAVA(1,IGAUSP,IELEM,2)      ,STRSG(1,IGAUSP,IELEM,1)       ,
     6    STRSG(1,IGAUSP,IELEM,2)       )
C
C
C thickness (for large strain analysis in plane stress only)
C
          IF(NLARGE.EQ.1.AND.NTYPE.EQ.1)THEN
            THKGP(IGAUSP,IELEM,NTOTHK)=THKGP(IGAUSP,IELEM,NTITHK)
          ENDIF
C
   10   CONTINUE
   20 CONTINUE
C
C
C Set nodal coordinates (for large strain analysis only)
C ========================================================
C
      IF(NLARGE.EQ.1.AND.(MODE.EQ.1.OR.MODE.EQ.3))THEN
        DO 40 IPOIN=1,NPOIN
          DO 30 IDIME=1,NDIME
            COORD(IDIME,IPOIN,NTO)=COORD(IDIME,IPOIN,NTI)
   30     CONTINUE
   40   CONTINUE
      ENDIF
C
C Set converged ELOAD and displacements
C =====================================
C
      IF(MODE.EQ.1)THEN
        DO 50 ITOTV=1,NTOTV
          TDISPO(ITOTV)=TDISP(ITOTV)
          DINCRO(ITOTV)=DINCR(ITOTV)
          DINCR(ITOTV)=R0
          DITER(ITOTV)=R0
   50   CONTINUE
```

```
        DO 70 IELEM=1,NELEM
          IGRUP =IGRPID(IELEM)
          IELIDN=IELTID(IGRUP)
          NEVAB =IELPRP(5,IELIDN)
          DO 60 IEVAB=1,NEVAB
            ELOADO(IEVAB,IELEM)=ELOAD(IEVAB,IELEM)
   60     CONTINUE
   70   CONTINUE
      ENDIF
C
C Reset displacements, logical algorithmic variables and
C ELOAD to last converged values (for increment cutting mode only)
C ================================================================
C
      IF(MODE.EQ.3)THEN
        DO 80 ITOTV=1,NTOTV
          TDISP(ITOTV)=TDISPO(ITOTV)
          DINCR(ITOTV)=R0
          DITER(ITOTV)=R0
   80   CONTINUE
        DO 100 IELEM=1,NELEM
          IGRUP =IGRPID(IELEM)
          IELIDN=IELTID(IGRUP)
          NEVAB =IELPRP(5,IELIDN)
          DO 90 IEVAB=1,NEVAB
            ELOAD(IEVAB,IELEM)=ELOADO(IEVAB,IELEM)
   90     CONTINUE
  100   CONTINUE
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE SWMC
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC              ,LALGVL
      DIMENSION
     1    LALGVC(*)           ,LALGVL(*)           ,RALGVC(*)           ,
     2    RSTAVC(*)           ,RSTAVL(*)           ,STRESC(*)           ,
     3    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE MOHR-COULOMB MATERIAL MODEL
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=5
      ENDIF
      NRALGV=2
      NLALGV=5
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the infinitesimal elastic egineering strain tensor
C (engineering logarithmic strains in large strains) and the effective
C plastic strain
        CALL RVZERO(RSTAVC,NRSTAV)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multipliers before starting a new increment
          CALL RVZERO(RALGVC,NRALGV)
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multipliers before starting a new increment
            CALL RVZERO(RALGVC,NRALGV)
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWOGD
     1(   MODE       ,NTYPE      ,RSTAVC     ,RSTAVL     ,STRESC     ,
     2    STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      DIMENSION
     1    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     2    STRESL(*)
C Local numerical constants
      DATA R0   ,R1   /
     1     0.0D0,1.0D0/
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE OGDEN MATERIAL MODEL
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=4
      ENDIF
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
C Zero stress component array
        CALL RVZERO(STRESC,NSTRE)
C RSTAVA stores the left Cauchy-Green strain tensor components.
C Initialised as identity
        IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
          RSTAVC(1)=R1
          RSTAVC(2)=R1
          RSTAVC(3)=R0
          RSTAVC(4)=R1
        ENDIF
C Switching modes
C ===============
      ELSEIF(MODE.EQ.1)THEN
        DO 10 I=1,NSTRE
          STRESL(I)=STRESC(I)
   10   CONTINUE
        DO 20 I=1,NRSTAV
          RSTAVL(I)=RSTAVC(I)
   20   CONTINUE
      ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
        DO 30 I=1,NSTRE
          STRESC(I)=STRESL(I)
   30   CONTINUE
        DO 40 I=1,NRSTAV
          RSTAVC(I)=RSTAVL(I)
   40   CONTINUE
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWPDSC
     1(   MODE       ,LALGVC     ,LALGVL     ,RALGVC     ,RSTAVC     ,
     2    RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC             ,LALGVL
      DIMENSION
     1    LALGVC(*)          ,LALGVL(*)          ,RALGVC(*)          ,
     2    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     3    STRESL(*)
      DATA R0   ,R1   /
     1     0.0D0,1.0D0/
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE PLANAR DOUBLE-SLIP SINGLE CRYSTAL
C ELASTO-PLASTIC MATERIAL MODEL
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      NSTRE=4
      NRSTAV=5
      NRALGV=4
      NLALGV=6
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the elastic deformation gradient tensor (in-plane
C component only)
        RSTAVC(1)=R1
        RSTAVC(2)=R0
        RSTAVC(3)=R0
        RSTAVC(4)=R1
C and the accumulated plastic slip
        RSTAVC(5)=R0
      ELSE
C Switching modes
C ===============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multipliers before starting a new increment
          CALL RVZERO(RALGVC,NRALGV)
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multipliers before starting a new increment
            CALL RVZERO(RALGVC,NRALGV)
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWTR
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC             ,LALGVL
      DIMENSION
     1    LALGVC(*)          ,LALGVL(*)          ,RALGVC(*)          ,
     2    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     3    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE TRESCA MATERIAL MODEL
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=5
      ENDIF
      NRALGV=2
      NLALGV=4
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the infinitesimal elastic egineering strain tensor
C (engineering logarithmic strains in large strains) and the
C effective plastic strain
        CALL RVZERO(RSTAVC,NRSTAV)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multipliers before starting a new increment
          CALL RVZERO(RALGVC,NRALGV)
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multipliers before starting a new increment
            CALL RVZERO(RALGVC,NRALGV)
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWVM
     1(   MODE       ,NTYPE      ,LALGVC     ,LALGVL     ,RALGVC     ,
     2    RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC             ,LALGVL
      DIMENSION
     1    LALGVC(*)          ,LALGVL(*)          ,RALGVC(*)          ,
     2    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     3    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE VON MISES MODEL WITH ISOTROPIC
C HARDENING
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=5
      ENDIF
      NRALGV=1
      NLALGV=2
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the infinitesimal elastic egineering strain tensor
C (engineering logarithmic strains in large strains) and the effective
C plastic strain
        CALL RVZERO(RSTAVC,NRSTAV)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multipliers before starting a new increment
          CALL RVZERO(RALGVC,NRALGV)
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multipliers before starting a new increment
            CALL RVZERO(RALGVC,NRALGV)
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE SWVMMX
     1(   MODE       ,NLARGE     ,NTYPE      ,LALGVC     ,LALGVL     ,
     2    RALGVC     ,RSTAVC     ,RSTAVL     ,STRESC     ,STRESL     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Arguments
      LOGICAL
     1    LALGVC             ,LALGVL
      DIMENSION
     1    LALGVC(*)          ,LALGVL(*)          ,RALGVC(*)          ,
     2    RSTAVC(*)          ,RSTAVL(*)          ,STRESC(*)          ,
     3    STRESL(*)
C***********************************************************************
C INITIALISE/SWITCH DATA FOR THE VON MISES MODEL WITH MIXED HARDENING
C
C    MODE=0:   Initialises the relevant data.
C
C    MODE=1:   Assigns current values of the state variables to
C              converged solution (when the current iteration
C              satisfies the convergence criterion).
C
C    MODE=2:   Assigns the last converged solution to current state
C              variables values (when a new iteration is required by
C              the iterative process).
C
C    MODE=3:   Assigns the last converged solution to current state
C              variables values (when increment cutting is required).
C***********************************************************************
C
      IF(NLARGE.EQ.1)THEN
C Internal error: Model implemented only for small strain
        CALL ERRPRT('EI0049')
      ENDIF
      IF(NTYPE.EQ.1.OR.NTYPE.EQ.2.OR.NTYPE.EQ.3)THEN
        NSTRE=4
        NRSTAV=9
      ENDIF
      NRALGV=1
      NLALGV=2
C
      IF(MODE.EQ.0)THEN
C Initialisation mode
C ===================
        CALL RVZERO(STRESC,NSTRE)
        CALL RVZERO(RALGVC,NRALGV)
        DO 10 I=1,NLALGV
          LALGVC(I)=.FALSE.
   10   CONTINUE
C RSTAVA stores the infinitesimal elastic egineering strain tensor
C components, the effective plastic strain and the backstress tensor
C components
          CALL RVZERO(RSTAVC,NRSTAV)
      ELSE
C Switching mode
C ==============
        IF(MODE.EQ.1)THEN
          DO 20 I=1,NSTRE
            STRESL(I)=STRESC(I)
   20     CONTINUE
          DO 30 I=1,NRSTAV
            RSTAVL(I)=RSTAVC(I)
   30     CONTINUE
          DO 40 I=1,NLALGV
            LALGVL(I)=LALGVC(I)
   40     CONTINUE
C Zero plastic multipliers before starting a new increment
          CALL RVZERO(RALGVC,NRALGV)
        ELSEIF(MODE.EQ.2.OR.MODE.EQ.3)THEN
          DO 50 I=1,NSTRE
            STRESC(I)=STRESL(I)
   50     CONTINUE
          DO 60 I=1,NRSTAV
            RSTAVC(I)=RSTAVL(I)
   60     CONTINUE
          DO 70 I=1,NLALGV
            LALGVC(I)=LALGVL(I)
   70     CONTINUE
          IF(MODE.EQ.3)THEN
C Zero plastic multipliers before starting a new increment
            CALL RVZERO(RALGVC,NRALGV)
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE TANGEN
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C
      DATA R0/0.0D0/
C***********************************************************************
C SETS UP PRESCRIBED DISPLACEMENTS FOR THE TANGENTIAL SOLUTION FOR
C THE ARC-LENGTH METHOD
C
C REFERENCE: Item (iii), Box 4.4
C***********************************************************************
      DO 20 IVFIX=1,NVFIX
        NLOCA=(NOFIX(IVFIX)-1)*NDOFN
        DO 10 IDOFN=1,NDOFN
          NPOS=NLOCA+IDOFN
          FIXED(NPOS,1)=PRESC(IVFIX,IDOFN)
          FIXED(NPOS,2)=R0
   10   CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TUEL
     1(   DETF       ,RSTAVA     ,THICK      ,MODE    )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER( MSTRE=4 )
      DIMENSION
     1    RSTAVA(MSTRE)
C***********************************************************************
C THICKNESS UPDATE FOR THE HENCKY ELASTIC MODEL MODEL UNDER LARGE
C STRAINS AND PLANE STRESS
C
C REFERENCE: Section 13.3.2
C***********************************************************************
C Compute determinant of total deformation gradient (including
C out-of-plane contribution).
C ...start by retrieving the diagonal components of the logarithmic
C    strain tensor
      E11=RSTAVA(1)
      E22=RSTAVA(2)
      E33=RSTAVA(4)
C ...then compute determinant of total deformation gradient
      DETF=EXP(E11+E22+E33)
      IF(MODE.EQ.1)THEN
C Compute thickness stretch
        STRTC3=EXP(E33)
C Update thickness
        THICK=THICK*STRTC3
      ENDIF
C
      RETURN
      END
```

```
      SUBROUTINE TUVM
     1(   DETF       ,RSTAVA     ,THICK      ,MODE     )
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER( MSTRE=4 )
      DIMENSION
     1    RSTAVA(MSTRE+1)
C***********************************************************************
C THICKNESS UPDATE FOR THE VON MISES ELASTO-PLASTIC MODEL UNDER LARGE
C STRAINS AND PLANE STRESS
C
C REFERENCE: Expressions (14.113-115)
C***********************************************************************
C Compute determinant of total deformation gradient (including
C out-of-plane contribution). Note that, for this model, the determinant
C of the total and elastic deformation gradient coincide due to plastic
C incompressibility.
C... start by retrieving the diagonal components of the elastic
C    logarithmic strain tensor
      EE11=RSTAVA(1)
      EE22=RSTAVA(2)
      EE33=RSTAVA(4)
C... then compute determinant of total deformation gradient
      DETFT=EXP(EE11+EE22+EE33)
      IF(MODE.EQ.1)THEN
C Compute thickness stretch
        STRTC3=DETFT/DETF
C Update thickness
        THICK=THICK*STRTC3
      ENDIF
C return total deformation gradient determinant in DETF
      DETF=DETFT
C
      RETURN
      END
```

```
      SUBROUTINE UPCONF
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C Hyplas database
      INCLUDE '../MAXDIM.INC'
      INCLUDE '../MATERIAL.INC'
      INCLUDE '../ELEMENTS.INC'
      INCLUDE '../GLBDBASE.INC'
C***********************************************************************
C KINEMATIC/GEOMETRIC CONFIGUTATION UPDATE:
C GIVEN THE ITERATIVE DISPLACEMENTS, THIS ROUTINE UPDATES THE GLOBAL
C ARRAYS OF INCREMENTAL AND TOTAL DISPLACEMENTS.
C FOR GEOMETRICALLY NON-LINEAR ANALYSES (LARGE DEFORMATIONS) IT ALSO
C UPDATES THE CURRENT NODAL COORDINATES.
C
C REFERENCE: Figures 5.2-3
C***********************************************************************
C
C Update incremental and total displacements
C ==========================================
C
      DO 10 ITOTV=1,NTOTV
        DINCR(ITOTV)=DINCR(ITOTV)+DITER(ITOTV)
        TDISP(ITOTV)=TDISP(ITOTV)+DITER(ITOTV)
   10 CONTINUE
C
C Update current nodal coordinates for large deformation analyses
C ===============================================================
C
      IF(NLARGE.EQ.1)THEN
        DO 30 IPOIN=1,NPOIN
          NPOSN=(IPOIN-1)*NDOFN
          DO 20 IDOFN=1,NDIME
            NPOSN=NPOSN+1
            COORD(IDOFN,IPOIN,1)=COORD(IDOFN,IPOIN,2)+DINCR(NPOSN)
   20     CONTINUE
   30   CONTINUE
      ENDIF
C
      RETURN
      END
```