Neste anexo encontra-se a subrotina *Usermat* do modelo constitutivo EPVP.

```
1
2   *deck,usermat3d    USERDISTRIB  parallel                                  gal
3         subroutine usermat3d_EPVP (
4        &                   matId, elemId,kDomIntPt, kLayer, kSectPt,
5        &                   ldstep,isubst,keycut,
6        &                   nDirect,nShear,ncomp,nStatev,nProp,
7        &                   Time,dTime,Temp,dTemp,
8        &                   stress,ustatev,dsdePl,sedEl,sedPl,epseq,
9        &                   Strain,dStrain, epsPl, prop, coords,
10       &                   var0, defGrad_t, defGrad,
11       &                   tsstif, epsZZ, cutFactor,
12       &                   var1, var2, var3, var4, var5,
13       &                   var6, var7)
14  #include "impcom.inc"
15        !**************************************************************************!
16        !** SUBROTINA: USERMAT3D_EPVP                                            **!
17        !**                                                                      **!
18        !** Objetivo: atualiza as tensões, variáveis de estado e matriz          **!
19        !**           constitutiva para o modelo constitutivo EPVP               **!
20        !**                                                                      **!
21        !** Situação: OK (10/11/2021)                                            **!
22        !**                                                                      **!
23        !**************************************************************************!
24        !
25        !**************************************************************************!
26        ! Declaração variáveis de entrada e saída da subrotina                    !
27        !**************************************************************************!
28        INTEGER
29       &                  matId, elemId,
30       &                  kDomIntPt, kLayer, kSectPt,
31       &                  ldstep,isubst,keycut,
32       &                  nDirect,nShear,ncomp,nStatev,nProp
33        DOUBLE PRECISION
34       &                  Time,    dTime,   Temp,    dTemp,
35       &                  sedEl,   sedPl,   epseq,   epsZZ,   cutFactor
36        DOUBLE PRECISION
37       &                  stress  (ncomp  ), ustatev (nStatev),
38       &                  dsdePl  (ncomp,ncomp),
39       &                  Strain  (ncomp  ), dStrain (ncomp  ),
40       &                  epsPl   (ncomp  ), prop    (nProp  ),
41       &                  coords  (3),
42       &                  defGrad (3,3),     defGrad_t(3,3),
43       &                  tsstif  (2)
44        DOUBLE PRECISION var0, var1, var2, var3, var4, var5,
45       &                  var6, var7
46        !
47        !**************************************************************************!
48        ! Informações sobre as variáveis locais (precisão, tipo, escopo)          !
49        !**************************************************************************!
50  c     ! aux1          (dp,sc,l)                denominador no dlambdaVP
```

```
c     ! c              (dp,sc,l)               coesão
c     ! c1,c2,c3       (dp,sc,l)               cs do vetor de fluxo
c     ! cp,ci,cr       (dp,sc,l)               coesão inicial, pico e residual
c     ! czao           (dp,sc,l)               constante para def. plast. equiv.
c     ! czaoVP         (dp,sc,l)               constante para def. viscop. equiv.
c     ! cVP            (dp,sc,l)               coesão do modelo viscoplástio
c     ! ddlam          (dp,sc,l)               ddlam pelo NR do corretor plástico
c     ! dfds           (dp,ar(ncomp),l)        df/dsigma
c     ! dfds_m         (dp,ar(1,ncomp),l)      df/dsigma^T para aplicar MATMUL
c     ! dgds           (dp,ar(ncomp),l)        dg/dsigma
c     ! dgdsVP         (dp,ar(ncomp),l)        dg/dsigma viscoplástico
c     ! dgds_m         (dp,ar(ncomp,1),l)      dg/dsigma para aplicar MATMUL
c     ! dgdsVP_m       (dp,ar(ncomp,1),l)      dg/dsigmaVP para aplicar MATMUL
c     ! dhdq,dfdq,dfdc,dcde (dp,sc,l)           dh/dq, df/dq, df/dc, dc/de
c     ! dgPHItD        (dp,ar(ncomp,ncomp),l)  produto dg/dsigma*PHI^T*D
c     ! dlam           (dp,sc,l)               delta lambda elastoplastico
c     ! dlambdaVP      (dp,sc,l)               delta lambda viscoplastico
c     ! depsEP         (dp,ar(ncomp),l)        incremento deformações plásticas
c     ! depsVP         (dp,ar(ncomp),l)        incremento deformações viscoplásticas
c     ! depsEPeq       (dp,sc,l)               incremen. defor. plástica equivalente
c     ! DgftD          (dp,ar(ncomp,ncomp),l)  produto D*dg/dsigma*df/dsigma^T*D
c     ! dsdeEl         (dp,ar(ncomp,ncomp),l)  matriz constitutiva elastica
c     ! dsdeElinv      (dp,ar(ncomp,ncomp),l)  inversa de dsdeEl
c     ! dt1,dt2        (dp,sc,l)               tempos limites do modelo viscoplástico
c     ! eps1,eps2,eps3 (dp,sc,l)               deformações plast. equiva. amol/endur.
c     ! epsEP          (dp,ar(ncomp),l)        deformações plásticas
c     ! epsVP          (dp,ar(ncomp),l)        deformações viscoplasticas
c     ! epsEPeq        (dp,sc,l)               deformação plástica equivalente
c     ! epsVPeq        (dp,sc,l)               deformação viscoplástica equivalente
c     ! f              (dp,sc,l)               função de escoamento elastoplástico
c     ! fVP            (dp,sc,l)               função de escoamento viscoplástico
c     ! fi             (dp,sc,l)               ângulo de atrito elastoplástico
c     ! fiVP           (dp,sc,l)               ângulo de atrito viscoplástico
c     ! ftDg           (dp,sc,l)               produto df/dsigma^T*D*dg/dsigma
c     ! g1,g2,g3       (dp,ar(ncomp),l)        componentes diretoras do vetor de fluxo
c     ! i,j,k          (int,sc,l)              contadores
c     ! I1             (dp,sc,l)               primeiro invariante das tensões
c     ! J2             (dp,sc,l)               segundo invariante do desviador
c     ! J3             (dp,sc,l)               terceiro invariante do desviador
c     ! n,eta,f0       (dp,sc,l)               constantes do modelo de Perzyna
c     ! ncompgt        (int,sc,l)              componentes de tensões iniciais
c     ! nrmax          (int,sc,l)              quantidade máxima de interações de NR
c     ! PHI            (dp,sc,l)               função de sobretensão
c     ! PHItDdeps      (dp,sc,l)               produto PHI^T*D*deps
c     ! PHItDg         (dp,sc,l)               produto dPHI/ds^T*D*dg/ds
c     ! psi            (dp,sc,l)               angulo de dilatância
c     ! psiVP          (dp,sc,l)               angulo de dilatância viscoplástico
c     ! s              (dp,ar(ncomp),l)        tensor desviador
c     ! sigi           (dp,ar(ncomp),l)        tensão inicial
c     ! sigmap         (dp,ar(ncomp),l)        tensão inicial a descontar
c     ! stressn        (dp,ar(ncomp),l)        tensão no inicio do passo
c     ! stresstrial    (dp,ar(ncomp),l)        tensão tentativa
c     ! theta          (dp,sc,l)               ângulo de Lode
c     ! thetaVP        (dp,sc,l)               ângulo de Lode do modelo viscoplástico
c     ! young          (dp,sc,l)               módulo de Young
c     ! posn           (dp,sc,l)               coeficiente de Poisson
c     ! q1,dq          (dp,sc,l)               variaveis auxiliares
c     ! superficief    (int,sc,l)              f   : 1-DPI, 2-DPII, 3-DPIII
c     ! superficiefVP  (int,sc,l)              fVP : 1-DPI, 2-DPII, 3-DPIII
c     ! superficiegVP  (int,sc,l)              g   : 1-DPI, 2-DPII, 3-DPIII
```

```
c     ! superficieg   (int,sc,l)               gVP : 1-DPI, 2-DPII, 3-DPIII
c     ! vPi           (dp,sc,l)               Pi
c     !
c     !*************************************************************************!
c     ! INFORMAÇÕES DAS VARIÁVEIS DE ESTADO                                     !
c     !*************************************************************************!
c     !
c     !   nstatev    = 20
c     !   ustatev(1) = epsEPeq
c     !   ustatev(2) = dlam
c     !   ustatev(3) = c
c     !   ustatev(4) = q
c     !   ustatev(5) = f
c     !   ustatev(6) = epsVPeq
c     !   ustatev(7) = dt1
c     !   ustatev(8) = dt2
c     !   ustatev(9:ncomp) = espEP(1:ncomp)
c     !   ustatev(15:ncomp) = espVP(1:ncomp)
c     !
c     ! OBS: as variáveis de estados podem ser plotadas na GUI do ANSYS
c     ! utilizando o comando PLESOL,SVAR,[componente] ou PLNESOL,SVAR,[componente]
c     !
c     !*************************************************************************!
c     ! Declaração das variáveis locais                                         !
c     !*************************************************************************!
c     !
c     ! Variáveis comuns a ambos modelos
      INTEGER
     &                i, j, k, ncompgt
      DOUBLE PRECISION
     &                I1,J2,J3,theta,
     &                c1,c2,c3,
     &                vPi, young,posn
     &                dstress(ncomp),sigi(ncomp),
     &                dsdeEl(ncomp,ncomp),
     &                s(ncomp), g1(ncomp),g2(ncomp),g3(ncomp)
      PARAMETER           (
     &                    vPi = 3.14159265358979323846d0
     &                   )
      EXTERNAL
     &                vzero, vmove, get_ElmData, get_ElmInfo,
     &                matrizD,invars,normatensor,calcula_czao,
     &                matinv,yield,c1c2c3,g1g2g3
c     !
c     ! variáveis do modelo viscoplástico
      DOUBLE PRECISION
     &                fVP,PHI,
     &                dlambdaVP,PHItDdeps,PHItDg,aux1,
     &                epsVPeq,depsVPeq,czaoVP
      DOUBLE PRECISION
     &                dgdsVP(ncomp),dPHIds(ncomp),
     &                dPHIds_m(1,ncomp),dgdsVP_m(ncomp,1),
     &                epsVP(ncomp),depsVP(ncomp),
     &                dgPHItD(ncomp,ncomp),
     &                sigmap(ncomp),dt1,dt2
      INTEGER
     &                superficiefVP,superficiegVP
      DOUBLE PRECISION
     &                cVP,fiVP,psiVP,n,eta,f0,thetaVP
c     !
c     ! variáveis do modelo elastoplástico
```

```
      INTEGER
     &                 nrmax, dalg
      DOUBLE PRECISION
     &                 tolEP,
     &                 f,
     &                 dlam,ddlam,ftDg,
     &                 dhdq,dfdq,dqdc,dcde,
     &                 epsEPeq,depsEPeq,czao,
     &                 q,dq,Xi
      DOUBLE PRECISION
     &                 dsdeElinv(ncomp,ncomp),
     &                 stresstrial(ncomp),
     &                 dgds(ncomp),dfds(ncomp),
     &                 dfds_m(1,ncomp),dgds_m(ncomp,1),
     &                 epsEP(ncomp),depsEP(ncomp),
     &                 DgftD(ncomp,ncomp),
     &                 stressn(ncomp)
      INTEGER
     &                 superficief,superficieg
      DOUBLE PRECISION
     &                 c,fi,psi,ci,cp,cr,eps1,eps2,eps3
      PARAMETER           (nrmax = 100000,
     &                     tolEP = 0.0000000000001d0
     &                    )
      EXTERNAL
     &                 calcula_Xi,calcula_dcde
c     !
c     !**********************************************************************!
c     ! Entrada de dados                                                     !
c     !**********************************************************************!
c     !
c     ! Variavel que controla o método da bisseção caso não haja convergência
      keycut      = 0
c     !
c     ! Propriedades elásticas
      young       = prop(2)
      posn        = prop(3)
c     !
c     ! Propriedades do modelo EP
      superficief = prop(4)
      superficieg = prop(5)
      fi          = prop(6)*vPi/180
      psi         = prop(7)*vPi/180
      ci          = prop(8)
      cp          = prop(9)
      cr          = prop(10)
      eps1        = prop(11)
      eps2        = prop(12)
      eps3        = prop(13)
      dalg        = prop(14)
c     !
c     ! Propriedades do modelo VP
      superficiefVP = prop(15)
      superficiegVP = prop(16)
      fiVP          = prop(17)*vPi/180
      psiVP         = prop(18)*vPi/180
      cVP           = prop(19)
      n             = prop(20)
      eta           = prop(21)
      f0            = prop(22)
```

```
235         thetaVP       = prop(23)
236   c     !
237   c     ! Limpando variáveis do modelo EP
238         depsEP      = 0.0d0
239         depsEPeq    = 0.0d0
240         dq          = 0.0d0
241         f           = 0.0d0
242         stressn     = 0.0d0
243         dcde        = 0.0d0
244         k           = 0
245   c     !
246   c     ! Limpando variáveis do modelo VP
247         depsVP      = 0.0d0
248         fVP         = 0.0d0
249         dt1         = 0.0d0
250         dt2         = 0.0d0
251   c     !
252   c     !*************************************************************************!
253   c     ! Coletando variáveis de estado e deformações plásticas do passo convergido!
254   c     !*************************************************************************!
255   c     ! Modelo EP
256         epsEPeq = ustatev(1)
257         dlam    = ustatev(2)
258         c       = ustatev(3)
259         q       = ustatev(5)
260         CALL vmove(ustatev(9), epsEP(1), ncomp)
261   c     !
262   c     ! Inicializa o valor da coesão
263         IF(c.EQ.0.0d0)c=ci
264         IF(q.EQ.0.0d0)THEN
265             CALL calcula_Xi(superficief,fi,Xi)
266             q=Xi*ci
267         ENDIF
268   c     !
269   c     ! Modelo VP
270         epsVPeq = ustatev(6)
271         CALL vmove(ustatev(15), epsVP(1), ncomp)
272   c     !
273   c     !*************************************************************************!
274   c     ! Calculo da matriz constitutiva                                          !
275   c     !*************************************************************************!
276         dsdeEl      = 0.0d0
277         CALL MatrizD(young,posn,ncomp,dsdeEl)
278         dsdePl = dsdeEl
279   c     !
280   c     !*************************************************************************!
281   c     ! Calculo módulo de rigidez transversal para hourglass                    !
282   c     !*************************************************************************!
283         tsstif(1) = 0.5d0*(young /(1.0d0+posn))
284   c     !
285   c     !*************************************************************************!
286   c     ! Coletando tensões iniciais                                              !
287   c     !*************************************************************************!
288         call get_ElmInfo('NCOMP', ncompgt)
289         call vzero(sigi(1),ncompgt)
290         call get_ElmData ('ISIG', elemId,kDomIntPt, ncompgt, sigi)
291   c     !
292   c     !*************************************************************************!
293   c     ! Calculo da tensão no passo n                                            !
294   c     !*************************************************************************!
```

```
295         stress = MATMUL(dsdeEl(1:ncomp,1:ncomp),
296        &         strain(1:ncomp)
297        &         -epsVP(1:ncomp)
298        &         -epsEP(1:ncomp))+sigi
299   c     !
300   c     !*********************************************************************!
301   c     ! Calculo da função de sobretensão                                    !
302   c     !*********************************************************************!
303         PHI = 0.0d0
304         CALL invars(stress,ncomp,I1,J2,J3,theta,s)
305         CALL yield(superficiefVP,I1,J2,theta,cVP,fiVP,fVP)
306         PHI = (fVP/f0)**n
307         IF(PHI.LE.0.0d0)PHI = 0.0d0
308   c     !
309   c     !*********************************************************************!
310   c     ! Calculo dgdsvp                                                      !
311   c     !*********************************************************************!
312         CALL g1g2g3(s,ncomp,J2,g1,g2,g3)
313         CALL c1c2c3(J2,theta,superficiegVP,psiVP,c1,c2,c3)
314         dgdsVP = c1*g1 + c2*g2 + c3*g3
315   c     !
316   c     !*********************************************************************!
317   c     ! Calculo dPHIds,PHItDg,PHItDdeps,aux1                                !
318   c     !*********************************************************************!
319         CALL c1c2c3(J2,theta,superficiefVP,psiVP,c1,c2,c3)
320         dPHIds = c1*g1 + c2*g2 + c3*g3
321         IF(PHI.LE.0.0d0)dPHIds = 0.0d0
322   c     !
323   c     !*********************************************************************!
324   c     ! Verificação do incremento de tempo                                  !
325   c     !*********************************************************************!
326         IF(PHI.GT.0.0d0)THEN
327         dt1 = 4.0d0/3.0d0*eta/PHI*(1.0d0+posn)/young*DSQRT(3.0d0*J2)
328         dt2 = eta*f0/(n*(fVP/f0)**(n-1))*
329        &         (1.0d0+posn)*(1.0d0-2.0d0*posn)/young*
330        &         ((3.0d0-DSIN(fiVP))**2)/
331        &         (3.0d0/4.0d0*(1.0d0-2.0d0*posn)*(3.0d0-DSIN(fiVP))**2+
332        &         6.0d0*(1.0d0+posn)*DSIN(fiVP)**2)
333         IF(dtime.GT.dt1.OR.dtime.GT.dt2)THEN
334             keycut = 1
335             RETURN
336         ENDIF
337         ENDIF
338   c     !
339   c     !*********************************************************************!
340   c     ! Atualização do módulo constitutivo                                  !
341   c     !*********************************************************************!
342         PHItDg = DOT_PRODUCT(dPHIds,MATMUL(dsdeEl,dgdsVP))
343         aux1 = (eta/dtime + thetaVP*PHItDg)
344         dPHIds_m(1,:) = dPHIds
345         dgdsVP_m(:,1) = dgdsVP
346         DgPHItD = MATMUL(MATMUL
347        &               (MATMUL(dsdeEl,dgdsVP_m),dPHIds_m),dsdeEl)
348         dsdePl = dsdePl - thetaVP*DgPHItD/aux1
349   c     !
350   c     !*********************************************************************!
351   c     ! Calculo do p                                                        !
352   c     !*********************************************************************!
353         sigmap = PHI*MATMUL(dsdeEl(1:ncomp,1:ncomp),
354        &         dgdsVP(1:ncomp))/aux1
355   c     !
```

```
c     !**********************************************************************!
c     ! Calculo da deformação viscoplastica                                  !
c     !**********************************************************************!
      PHItDdeps = DOT_PRODUCT(dPHIds,MATMUL(dsdeEl,dstrain))
      dlambdaVP = (PHI + thetaVP*PHItDdeps)/aux1
      depsVP = dlambdaVP*dgdsVP
c     !
c     ! Calcula as deformações viscoplasticas totais
      epsVP = epsVP + depsVP
      CALL normatensor(epsVP,ncomp,epsVPeq)
      CALL calcula_Czao(superficiefVP,fiVP,CzaoVP)
      epsVPeq = CzaoVP*epsVPeq
c     !
c     !
c     !**********************************************************************!
c     ! Calculo preditor elástico                                            !
c     !**********************************************************************!
      stresstrial = 0.0d0
      stresstrial = MATMUL(dsdeEl(1:ncomp,1:ncomp),
     &              strain(1:ncomp)+dstrain(1:ncomp)
     &              -epsEP(1:ncomp)-epsVP(1:ncomp)) + sigi
      stress = stresstrial
c     !
c     !**********************************************************************!
c     ! Calcula função de escoamento                                         !
c     !**********************************************************************!
      CALL invars(stresstrial,ncomp,I1,J2,J3,theta,s)     ! Invariantes
      CALL yield(superficief,I1,J2,theta,c,fi,f)     ! função de escoamento
c     !
c     !**********************************************************************!
c     ! Verifica o critério de escoamento                                    !
c     !**********************************************************************!
      IF(f.GT.0.0d0)THEN
c         !
c         ! Aplica o corretor plástico
c         !
c         !**********************************************************************!
c         ! Calcula dgds                                                       !
c         !**********************************************************************!
          dgds = 0.0d0
          stressn = stresstrial
          CALL invars(stressn,ncomp,I1,J2,J3,theta,s)
          CALL g1g2g3(s,ncomp,J2,g1,g2,g3)
          CALL c1c2c3(J2,theta,superficieg,psi,c1,c2,c3)
          dgds = c1*g1 + c2*g2 + c3*g3
c         !
c         !**********************************************************************!
c         ! Calcula dhdq referente ao endurecimento e amolecimento               !
c         !**********************************************************************!
          dhdq = 0.0d0
          dfdq = -1.0d0
          CALL calcula_Xi(superficief,fi,Xi)
          dqdc = Xi
          CALL calcula_dcde(ci,cp,cr,eps1,eps2,eps3,epsEPeq,dcde)
          dhdq  = -dfdq*dqdc*dcde
c         !
c         !**********************************************************************!
c         ! Interações de NR local do modelo constitutivo                        !
c         !**********************************************************************!
          k = 0
          DO
```

```
c               !*********************************************************************!
c               ! Calculo de ddlamb                                                   !
c               !*********************************************************************!
c               !
c               ! Calcula dfds
                dfds = 0.0d0
                CALL invars(stress,ncomp,I1,J2,J3,theta,s)
                CALL g1g2g3(s,ncomp,J2,g1,g2,g3)
                CALL c1c2c3(J2,theta,superficief,fi,c1,c2,c3)
                dfds    = c1*g1 + c2*g2 + c3*g3
c               !
c               ! Calcula dfdq
                dfdq = -1.0d0
                !
c               ! Calcula denominador de ddlamb
                ftDg    = DOT_PRODUCT(dfds,MATMUL(dsdeEl,dgds))
c               !
c               ! Calcula ddlamb
                ddlam   = f/(ftDg - dfdq*dhdq)
c               !
c               !*********************************************************************!
c               ! Calculo do corretor plástico                                        !
c               !*********************************************************************!
                dstress = -ddlam*MATMUL(dsdeEl,dgds)
                dq      = -ddlam*(-dhdq)
c               !
c               !*********************************************************************!
c               ! Incremento das tensões e do dlam                                    !
c               !*********************************************************************!
                stress  = stress + dstress
                c       = c + dq/Xi
                dlam    = dlam + ddlam
                q       = q + dq
                k = k + 1
c               !
c               !*********************************************************************!
c               ! Calcula deformação plástica equivalente                             !
c               !*********************************************************************!
                call matinv(ncomp,dsdeEl,dsdeElinv)
                depsEP = depsEP-MATMUL(dsdeElinv,dstress)
c               !
c               !*********************************************************************!
c               ! Verifica critério de escoamento                                     !
c               !*********************************************************************!
                CALL invars(stress,ncomp,I1,J2,J3,theta,s)
                CALL yield(superficief,I1,J2,theta,c,fi,f)
                IF(f.LE.tolEP)EXIT
c               !
c               ! Caso atinja o número de iterações limites, faça a bisseção
                IF(k.EQ.nrmax)THEN
                    keycut = 1
                    RETURN
                ENDIF
            ENDDO
c           !
c           !*********************************************************************!
c           ! Atualizando o módulo constitutivo                                   !
c           !*********************************************************************!
            IF(dalg.EQ.1)THEN
                dfds_m(1,:) = dfds
                dgds_m(:,1) = dgds
```

```
                DgftD= MATMUL(MATMUL(MATMUL(dsdeEl,dgds_m),dfds_m),dsdeEl)
                dsdePl = dsdePl - DgftD/(ftDg - dfdq*dhdq)
            ENDIF
c           !
        ELSE
            depsEp = 0.0d0
        ENDIF
c       !
c       !*********************************************************************!
c       ! Guardando deformações plásticas totais                              !
c       !*********************************************************************!
c       !
c       ! Calcula as deformações plásticas totais
        epsEP = epsEP + depsEP
        CALL normatensor(epsEP,ncomp,epsEPeq)
        CALL calcula_Czao(superficief,fi,Czao)
        epsEPeq = Czao*epsEPeq
c       !
c       ! Retorna as deformações inelasticas totais e equivalentes (só tem plásticas)
        epsPl = epsEP+epsVP
        epseq = epsEPeq+epsVPeq
c       !
c       ! Calcula o trabalho elástico
        sedEl    = 0.0d0
        sedEl    = 1.0d0/2.0d0*
       &           DOT_PRODUCT(stress,strain+dstrain-epsPl-epsVP)
c       !
c       !
c       ! Calcula o trabalho inelástico (só tem plástico)
        CALL normatensor(depsEP,ncomp,depsEPeq)
        CALL calcula_Czao(superficief,fi,Czao)
        depsEPeq = Czao*depsEPeq
        CALL normatensor(depsVP,ncomp,depsVPeq)
        CALL calcula_Czao(superficief,fi,Czao)
        depsVPeq = Czao*depsVPeq
        sedPl = sedPl + (q)*depsEPeq+depsVPeq
c       !
c       ! Guarda valores nas variáveis de estado
        ustatev(1) = epsEPeq
        ustatev(2) = dlam
        ustatev(3) = c
        ustatev(4) = q
        ustatev(5) = f
        ustatev(6) = epsVPeq
        ustatev(7) = dt1
        ustatev(8) = dt2
        CALL vmove(epsEP(1), ustatev(9), ncomp)
        CALL vmove(epsVP(1), ustatev(15), ncomp)
c       !
        RETURN
        END
        !
```

```
      SUBROUTINE matrizD(E,Poisson,ncomp,D)
c     !**********************************************************************!
c     !** Função: matrizD                                                  **!
c     !**                                                                  **!
c     !** Objetivo: calcula a matriz consitutiva do material isotrópico    **!
c     !**           adaptado de Smith, Griffiths e Margetts (2014, p.42-44)**!
c     !**                                                                  **!
c     !** Situação: (28-09-2016) OK                                        **!
c     !**                                                                  **!
c     !**********************************************************************!
      IMPLICIT NONE
      DOUBLE PRECISION E              ! módulo de elasticidade
      DOUBLE PRECISION Poisson        ! coeficiente de Poisson
      INTEGER ncomp                   ! numero de componentes
      DOUBLE PRECISION D(ncomp,ncomp) ! matriz constitutiva elástica isotrópica
c     !
        D=0.1d0
        D(1,1)=(E*(1.0d0-Poisson))/((1.0d0+Poisson)*(1.0d0-2.0d0*Poisson))
        D(1,2)=(E*Poisson)/((1.0d0+Poisson)*(1.0d0-2.0d0*Poisson))
        D(1,3)=D(1,2)
        D(2,1)=D(1,2)
        D(2,2)=D(1,1)
        D(2,3)=D(1,2)
        D(3,1)=D(1,3)
        D(3,2)=D(2,3)
        D(3,3)=D(1,1)
        D(4,4)=(E)/((1.0d0+Poisson)*2.0d0)
c     !
      IF(ncomp.EQ.6)THEN
          D(ncomp-1,ncomp-1)=D(4,4)
            D(ncomp,ncomp)=D(4,4)
      ENDIF
c     !
      END SUBROUTINE MatrizD
c     !
      SUBROUTINE invars(stress,ncomp,I1,J2,J3,theta,s)
c     !**********************************************************************!
c     !** Subrotina: invars                                                **!
c     !**                                                                  **!
c     !** Objetivo: calcula os invariantes do tensor de tensões e o tensor **!
c     !**           desviador. Adaptado de Chen e Han (1998, p.57-72)      **!
c     !**                                                                  **!
c     !** Situação: (10-11-2021) OK                                        **!
c     !**                                                                  **!
c     !**********************************************************************!
      IMPLICIT NONE
      DOUBLE PRECISION stress(ncomp)  ! tensões
      INTEGER          ncomp          ! numero de componentes
      DOUBLE PRECISION I1             ! primeiro invariante do tensor de tensões
      DOUBLE PRECISION J2,J3          ! segundo e terceiro invariante do desviador
      DOUBLE PRECISION theta          ! angulo de Lode
      DOUBLE PRECISION s(6)           ! desviador
      DOUBLE PRECISION p,q            ! pressão hidrostática e tensão eq. de vm
      DOUBLE PRECISION sine           ! variavel auxiliar
c     !
c     ! Inicializando variaveis
      I1      = 0.0d0
      J2      = 0.0d0
      p       = 0.0d0
      q       = 0.0d0
      s       = 0.0d0
```

```
      J3      = 0.0d0
      theta   = 0.0d0
      sine    = 0.0d0
c     !
c     ! Calculo do I1 (p. 53)
      I1 = stress(1) + stress(2) + stress(3)
c     !
c     ! Calculo do J2 (p. 58)
      J2 = 1/6.0d0*((stress(1)-stress(2))**2+(stress(2)-stress(3))**2+
     &     (stress(3)-stress(1))**2)+
     &     stress(4)**2
c     !
      IF(ncomp.EQ.6)THEN
          J2 = J2 + stress(ncomp-1)**2+stress(ncomp)**2
      ENDIF
c     !
c     ! Calculo do p (p. 57)
      p = 1.0d0/3.0d0*I1
c     !
c     ! Calculo do desviador s (p. 57)
      s(1) = stress(1) - p
      s(2) = stress(2) - p
      s(3) = stress(3) - p
      s(4) = stress(4)
      IF(ncomp.EQ.6)THEN
          s(ncomp-1) = stress(ncomp-1)
          s(ncomp) = stress(ncomp)
      ENDIF
c     !
c     ! Calculo do J3 (p. 58)
      J3 = s(1)*s(2)*s(3)-s(3)*s(4)*s(4)
      IF(ncomp.EQ.6)THEN
          J3 = J3 -s(1)*s(ncomp-1)*s(ncomp-1)-s(2)*s(ncomp)*s(ncomp)+
     &         2.0d0*s(4)*s(ncomp-1)*s(ncomp)
      ENDIF
c     !
c     ! Calculo do ângulo de Lode (p. 70) e Owen e Hinton (1980, p.229)
      q = DSQRT(3.0d0*J2)
      IF(q < 1.E-7)THEN
          theta = 0.0d0
      ELSE
          sine = -3.0d0*DSQRT(3.0d0)*J3/(2.0d0*DSQRT(J2)**3)
          IF(sine>1.0d0)sine=1.0d0
          IF(sine<-1.0d0)sine=-1.0d0
          theta=DASIN(sine)/3.0d0
      END IF
c     !
      END SUBROUTINE
c     !
      SUBROUTINE g1g2g3(s,ncomp,J2,g1,g2,g3)
c     !*********************************************************************!
c     !** Subrotina: g1g2g3                                                **!
c     !**                                                                  **!
c     !** Objetivo: calcula as direções do vetor de fluxo. Adaptado de Owen e  **!
c     !**           Hinton (1980, p.231 e 233)                             **!
c     !**                                                                  **!
c     !** Situação: (10-11-2021) OK                                        **!
c     !**                                                                  **!
c     !*********************************************************************!
      IMPLICIT NONE
      DOUBLE PRECISION s(ncomp)             ! desviador
```

```
652         INTEGER          ncomp                ! numero de componentes
653         DOUBLE PRECISION J2                   ! segundo invariante do desviador
654         DOUBLE PRECISION g1(ncomp), g2(ncomp), g3(ncomp)
655   c     !
656   c     ! Inicializando variaveis
657         g1 = 0.0d0
658         g2 = 0.0d0
659         g3 = 0.0d0
660   c     !
661   c     ! Calculo do g1
662         g1(1) = 1.0d0
663         g1(2) = 1.0d0
664         g1(3) = 1.0d0
665   c     !
666   c     ! Calculo do g2
667         g2(1) = s(1)
668         g2(2) = s(2)
669         g2(3) = s(3)
670         g2(4) = 2.0d0*s(4)
671         IF(ncomp.EQ.6)THEN
672             g2(ncomp-1) = 2.0d0*s(ncomp-1)
673             g2(ncomp) = 2.0d0*s(ncomp)
674         ENDIF
675         IF(J2.EQ.0.0d0)THEN
676             g2 = 0.0d0
677         ELSE
678             g2 = 1.0d0/(2.0d0*DSQRT(J2))*g2
679         ENDIF
680   c     !
681   c     ! Calculo do g3
682         g3(1) = s(2)*s(3) + J2/3.0d0
683         g3(2) = s(1)*s(3) + J2/3.0d0
684         g3(3) = s(1)*s(2) - s(4)**2 + J2/3.0d0
685         g3(4) = 2.0d0*(-s(3)*s(4))
686         IF(ncomp.EQ.6)THEN
687             g3(1) = g3(1) - s(ncomp-1)**2
688             g3(2) = g3(2) - s(ncomp)**2
689             g3(4) = g3(4) + 2.0d0*s(ncomp-1)*s(ncomp)
690             g3(ncomp-1) = 2.0d0*(s(ncomp)*s(4)-s(1)*s(ncomp-1))
691             g3(ncomp) = 2.0d0*(s(4)*s(ncomp-1)-s(2)*s(ncomp))
692         ENDIF
693   c     !
694         END SUBROUTINE
695   c     !
696   c     !
697         SUBROUTINE c1c2c3(J2,theta,superficie,fi,c1,c2,c3)
698   c     !**********************************************************************!
699   c     !** Subrotina: c1c2c3                                                 **!
700   c     !**                                                                  **!
701   c     !** Objetivo: calcula a magnitude das componentes do vetor de fluxo. **!
702   c     !**           Adaptado de Bernaud (1991, p.90, 91)                   **!
703   c     !**           Adaptado de Owen e Hinton (1980, p.231)                **!
704   c     !**                                                                  **!
705   c     !** Situação: (10-11-2021) OK                                        **!
706   c     !**                                                                  **!
707   c     !**********************************************************************!
708         IMPLICIT NONE
709         DOUBLE PRECISION J2                ! segundo invariante do desviador
710         DOUBLE PRECISION theta             ! ângulo de Lode
711         DOUBLE PRECISION fi                ! Ângulo de atrito
712         INTEGER          superficie        ! 1-DPI, 2-DPII, 3-DPIII
```

```
      DOUBLE PRECISION c1,c2,c3           ! magnitude das componentes do vetor
      DOUBLE PRECISION beta1,beta2,beta3  ! parâmetros do DP
      DOUBLE PRECISION k                  ! coeficiente de empuxo
c     !
c     ! Seleciona o modelo
      SELECT CASE (superficie)
          CASE(1)
c             !
c             ! DPI
              k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
              c1 = (k-1.0d0)/3.0d0
              c2 = (k+2.0d0)/DSQRT(3.0d0)
              c3 = 0.0d0
c             !
          CASE(2)
c             !
c             ! DPII
              k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
              c1 = (k-1.0d0)/3.0d0
              c2 = (2.0d0*k+1.0d0)/DSQRT(3.0d0)
              c3 = 0.0d0
c             !
          CASE(3)
c             !
c             ! DPIII - Owen e Hinton (1980, p.231)
              c1 = 2.0d0*DSIN(fi)/(DSQRT(3.0d0)*(3.0d0+DSIN(fi)))
              c2 = 1.0d0
              c3 = 0.0d0
c             !
      ENDSELECT
      END SUBROUTINE
c     !
      SUBROUTINE yield(superficie,I1,J2,theta,c,fi,f)
c     !**********************************************************************!
c     !** Subrotina: yield                                                 **!
c     !**                                                                  **!
c     !** Objetivo: calcula o critério de escoamento.                      **!
c     !**           Adaptado de Bernaud (1991, p.90, 91)                   **!
c     !**           Adaptado de Souza Neto,  Peri, Owen (2008, p. 162-167) **!
c     !**                                                                  **!
c     !** Situação: (10-11-2021) OK                                        **!
c     !**                                                                  **!
c     !**********************************************************************!
      IMPLICIT NONE
      INTEGER          superficie         ! 1-DPI, 2-DPII, 3-DPIII
      DOUBLE PRECISION I1                 ! primeiro invariante do tensor de tensões
      DOUBLE PRECISION J2                 ! segundo invariante do desviador
      DOUBLE PRECISION theta              ! ângulo de Lode
      DOUBLE PRECISION c                  ! coesão
      DOUBLE PRECISION fi                 ! Ângulo de atrito
      DOUBLE PRECISION f                  ! função de escoamento
      DOUBLE PRECISION beta1,beta2,beta3  ! Parametros para DP
      DOUBLE PRECISION k                  ! coeficiente de empuxo
c     !
c     ! Seleciona o modelo
      SELECT CASE (superficie)
      CASE(1)
c             !
c             ! DPI
              k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
              beta1 = (k-1.0d0)/3.0d0
```

```
774                 beta2 = (k+2.0d0)/DSQRT(3.0d0)
775                 beta3 = 2.0d0*DSQRT(k)*c
776                 f = beta1*I1 + beta2*DSQRT(J2)-beta3
777   c             !
778               CASE(2)
779   c             !
780   c             ! DPII
781                 k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
782                 beta1 = (k-1.0d0)/3.0d0
783                 beta2 = (2.0d0*k+1.0d0)/DSQRT(3.0d0)
784                 beta3 = 2.0d0*DSQRT(k)*c
785                 f = beta1*I1 + beta2*DSQRT(J2)-beta3
786   c             !
787               CASE(3)
788   c             !
789   c             ! DPIII Souza Neto,  Peri, Owen (2008, p. 162-167)
790                 beta1 = 2.0d0*DSIN(fi)/(DSQRT(3.0d0)*(3.0d0+DSIN(fi)))
791                 beta2 = 1.0d0
792                 beta3 = 6.0d0*DCOS(fi)/(DSQRT(3.0d0)*(3.0d0+DSIN(fi)))*c
793                 f = beta1*I1 + beta2*DSQRT(J2)-beta3
794   c             !
795           ENDSELECT
796           END SUBROUTINE
797   c       !
798           subroutine matinv(n,a,ainv)
799   c       !**********************************************************************!
800   c       !** Subrotina: matinv                                                 **!
801   c       !**                                                                  **!
802   c       !** Objetivo: inverte uma matriz pela técnica de pivotamento         **!
803   c       !**                                                                  **!
804   c       !** Situação: (26-10-2016) OK                                        **!
805   c       !**                                                                  **!
806   c       !**********************************************************************!
807           INTEGER          n          ! dimensão do sistema
808           DOUBLE PRECISION a(n,n)     ! matriz dos coeficientes
809           DOUBLE PRECISION ainv(n,n)  ! matriz inversa
810           DOUBLE PRECISION b(n,2*n)   ! matriz aumentada
811           DOUBLE PRECISION pivot      ! pivô
812           DOUBLE PRECISION xnum       ! auxiliar
813           INTEGER          i,j,k      ! contador
814   c       !
815   c       ! Fazer matriz aumentada
816           do i=1,n
817               do j=1,n
818                   b(i,j) = 0.0d0
819                   b(i,j+n) = 0.0d0
820                   b(i,j)=a(i,j)
821                   if(i.eq.j)then
822                       b(i,j+n)=1.0d0
823                   endif
824               enddo
825           enddo
826   c       !
827           do i=1,n
828   c           ! Escolher o elemento não nulo mais a esquerda como pivot
829               do j=1,n
830                   if (dabs(b(i,j)).gt.0.0d0)then
831                       pivot=b(i,j)
832                       exit
833                   endif
834               enddo
```

```
c          !
c          ! Passo 1: alterar o pivo escolhido
           do j=1,2*n
               b(i,j)=b(i,j)/pivot
           enddo
           pivot=b(i,i)
c          !
c          ! Passo 2: mudando o restante da coluno do pivo para 0,
c              adicionando a cada linha um multiplo adequado do pivot
           do k=1,n
               if(k.ne.i)then
                   xnum=b(k,i)/pivot
                   do j=1,2*n
                       b(k,j)=b(k,j)-xnum*b(i,j)
                   enddo
               endif
           enddo
       enddo
c      !
c      ! Prepara a matriz inversa final
       do i=1,n
           do j=1,n
               ainv(i,j)=b(i,j+n)
           enddo
       enddo
       return
       end
c      !
       SUBROUTINE normatensor(tensor,ncomp,norma)
c      !********************************************************************!
c      !** Subrotina: normatensor                                         **!
c      !**                                                               **!
c      !** Objetivo: calcula a norma de um tensor escrito em notação de Voigt  **!
c      !**                                                               **!
c      !** Situação: (10-11-2021) OK                                      **!
c      !**                                                               **!
c      !********************************************************************!
       IMPLICIT NONE
       INTEGER ncomp
       DOUBLE PRECISION tensor(ncomp)
       DOUBLE PRECISION norma
c      !
       IF(ncomp.EQ.6)THEN
       norma = DSQRT(tensor(1)**2 + tensor(2)**2 + tensor(3)**2 +
      &            2*((tensor(4))**2 +
      &            (tensor(5))**2 + (tensor(6))**2))
       ELSEIF(ncomp.EQ.4)THEN
       norma = DSQRT(tensor(1)**2 + tensor(2)**2 + tensor(3)**2 +
      &            2*(tensor(4)**2))
       ELSE
       ENDIF
c      !
       END SUBROUTINE
```

```
c     !
      SUBROUTINE calcula_Czao(superficie,fi,Czao)
c     !**********************************************************************!
c     !** Subrotina: Czao                                                  **!
c     !**                                                                  **!
c     !** Objetivo: calcula o C utilizado no calculo da deformação plástica **!
c     !**           efetiva. Adaptado de Chen e Han (1988, p. 257-259).    **!
c     !**                                                                  **!
c     !** Situação: (10-11-2021) OK                                        **!
c     !**                                                                  **!
c     !**********************************************************************!
      IMPLICIT NONE
      INTEGER             superficie  ! 1-DPI, 2-DPII, 3-DPIII
      DOUBLE PRECISION    fi          ! angulo de atrito
      DOUBLE PRECISION    Czao        ! constante da deformação plástica efetiva
      DOUBLE PRECISION    beta        ! constante referente a pressão hidrostática
c     !
c     ! Seleciona o modelo
      SELECT CASE (superficie)
          CASE(1)
c             !
c             ! DPI
              beta = 2.0d0*DSIN(fi)/(DSQRT(3.0d0)*(3.0d0-DSIN(fi)))
c             !
          CASE(2)
c             !
c             ! DPII
              beta = 2.0d0*DSIN(fi)/(DSQRT(3.0d0)*(3.0d0+DSIN(fi)))
c             !
          CASE(3)
c             !
c             ! DPIII
              beta = 2.0d0*DSIN(fi)/(DSQRT(3.0d0)*(3.0d0+DSIN(fi)))
c             !
      ENDSELECT
      Czao = (beta+1.0d0/DSQRT(3.0d0))/
     &       (DSQRT(3.0d0*beta**2+1.0d0/2.0d0))
      END SUBROUTINE
c     !
      SUBROUTINE calcula_Xi(superficie,fi,Xi)
c     !**********************************************************************!
c     !** Subrotina: calcula_Xi                                            **!
c     !**                                                                  **!
c     !** Objetivo: calcula Xi                                             **!
c     !**           Adaptado de Potts e Zdravkovic (1999, p. 158)          **!
c     !**                                                                  **!
c     !**                                                                  **!
c     !** Situação: (10-11-2021) OK                                        **!
c     !**                                                                  **!
c     !**********************************************************************!
      IMPLICIT NONE
      INTEGER             superficie          ! 1-DPI, 2-DPII, 3-DPIII
      DOUBLE PRECISION    fi                  ! angulo de atrito
      DOUBLE PRECISION    Xi                  ! derivada de f em relação a c
      DOUBLE PRECISION    k                   ! coeficiente de empuxo
      !
      SELECT CASE(superficie)
      CASE(1)
          k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
          Xi = 2.0d0*DSQRT(k)
      CASE(2)
```

```
            k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
            Xi = 2.0d0*DSQRT(k)
         CASE(3)
            k = (1.0d0+DSIN(fi))/(1.0d0-DSIN(fi))
            Xi = 6.0d0*DCOS(fi)/(DSQRT(3.0d0)*(3.0d0+DSIN(fi)))
         END SELECT
         END SUBROUTINE
c        !
         SUBROUTINE calcula_dcde(ci,cp,cr,eps1,eps2,eps3,epsPleq,
        & dcde)
c        !**********************************************************************!
c        !** Subrotina: calcula_dcde                                           **!
c        !**                                                                   **!
c        !** Objetivo: calcula dc/de                                           **!
c        !**           Adaptado de Potts e Zdravkovic (1999, p. 158)           **!
c        !**                                                                   **!
c        !** Situação: (10-11-2021) OK                                         **!
c        !**                                                                   **!
c        !**********************************************************************!
         IMPLICIT NONE
         DOUBLE PRECISION    ci                  ! coesão inicial
         DOUBLE PRECISION    cp                  ! coesão no pico
         DOUBLE PRECISION    cr                  ! coesão residual
         DOUBLE PRECISION    eps1                ! deformação plastica equivalente 1
         DOUBLE PRECISION    eps2                ! deformação plastica equivalente 2
         DOUBLE PRECISION    eps3                ! deformação plastica equivalente 3
         DOUBLE PRECISION    epsPleq             ! deformação plástica equivalente
         DOUBLE PRECISION    dcde                ! dc/depsPleq
c        !
         IF(epsPleq.LT.eps1)THEN
            dcde = (cp-ci)/(eps1)
         ELSEIF((epsPleq.GE.eps1).AND.(epsPleq.LE.eps2))THEN
            dcde = 0.0d0
         ELSEIF((epsPleq.GT.eps2).AND.(epsPleq.LT.eps3))THEN
            dcde = (cr-cp)/(eps3-eps2)
         ELSEIF(epsPleq.GE.eps3)THEN
            dcde = 0.0d0
         ENDIF
c        !
         END SUBROUTINE
```