Introduire un nouveau comportement

Résumé :

Comment introduire un nouveau comportement ?

On décrit ici l'ajout d'un nouveau comportement pour résoudre un problème non linéaire posé sur une structure, avec STAT_NON_LINE ou DYNA_NON_LINE, pour tous les éléments 2D / 3D (et coques, tuyaux, poutres multi-fibres, ...)

Étapes essentielles :

- Écriture de la documentation de référence R (équations de la loi de comportement)
- Modification du catalogue de DEFI_MATERIAU (paramètres matériau de la loi de comportement)
- Ajout du catalogue python de la relation de comportement
- Choix de la méthode d'intégration parmi les possibilités suivantes :
  - écriture d'une routine autonome lc0nn intégrant le comportement en un point d'intégration
  - intégration explicite (ALGO_INTE='RUNGE_KUTTA') et écriture des routines associées
  - intégration implicite dans l'environnement PLASTI, implantation « complète » (ALGO_INTE='NEWTON')
  - intégration implicite dans l'environnement PLASTI, implantation «facile » (ALGO_INTE='NEWTON_PERT')
- Produire des tests !

Remarque : il est également possible de programmer des lois de comportement en dehors de l'environnement Aster, soit à l'aide du module *Zmat*, soit dans une routine de type *Umat* (cf. [U2.10.01]).

Table des Matières

1 Schéma général de résolution dans STAT_NON_LINE

Itération de Newton sur le système complet

- on part de la valeur obtenue à l'issue de la phase de prédiction :

$\Delta \mathbf{u}_i^0$

- on calcule la suite d'incréments $(\delta \mathbf{u}_i^{n+1})$

$\mathbf{K}_i^n . \delta \mathbf{u}_i^{n+1} = \mathbf{L}_i^{\text{méca}} - \mathbf{L}^{\text{int}}(\Delta \mathbf{u}_i^n, \boldsymbol{\sigma}_{i-1}, \boldsymbol{\alpha}_{i-1})$

- on actualise le déplacement $\mathbf{u}$:

$\Delta \mathbf{u}_i^{n+1} = \Delta \mathbf{u}_i^n + \delta \mathbf{u}_i^{n+1}$

- on teste la convergence et on réitère si besoin

Newton
i : n° de pas de temps
n : n° d'itération de Newton

Référence : Principe de résolution (algorithme de Newton) [R5.03.01] et module de cours Aster Non linéaire

au niveau global : l'itération *n* de Newton fournit $\Delta \mathbf{u}_i^n = \Delta \mathbf{u}_i^{n-1} + \delta \mathbf{u}_i^n$

au niveau de l'élément : calcul de $\boldsymbol{\varepsilon}(\Delta \mathbf{u}_i^n)$ **en chaque point de Gauss (PG)**

au niveau du PG : intégration de la loi de comportement

calcul des contraintes et variables internes $\begin{cases} \boldsymbol{\sigma}_{i-1}, \boldsymbol{\alpha}_{i-1} \\ \boldsymbol{\varepsilon}(\Delta \mathbf{u}_i^n) \end{cases} \rightarrow \boldsymbol{\sigma}_i^n, \boldsymbol{\alpha}_i^n$

calcul éventuel de la dérivée de $\boldsymbol{\sigma}$ par rapport à $\boldsymbol{\varepsilon}$ $\begin{cases} \boldsymbol{\sigma}_{i-1}, \boldsymbol{\alpha}_{i-1} \\ \boldsymbol{\varepsilon}(\Delta \mathbf{u}_i^n) \end{cases} \rightarrow \left(\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\right)_i^n$

seul endroit où intervenir

calcul des forces nodales élémentaires ${}^E(\mathbf{L}^{\text{int}})_i^n = \int_{\Omega_E} \mathbf{Q}^T \boldsymbol{\sigma}_i^n \, \mathrm{d}\Omega$

calcul éventuel de la matrice tangente élémentaire ${}^E\mathbf{K}_i^n = \int_{\Omega_E} \mathbf{Q}^T \left(\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\right)_i^n \mathbf{Q} \, \mathrm{d}\Omega$

assemblage des forces nodales $\mathbf{L}^{\text{int}}(\Delta \mathbf{u}_i^n, \boldsymbol{\sigma}_{i-1}, \boldsymbol{\alpha}_{i-1}) = \sum_E {}^E(\mathbf{L}^{\text{int}})_i^n$

calcul du résidu $\mathbf{L}_i^{\text{méca}} - \mathbf{L}^{\text{int}}(\Delta \mathbf{u}_i^n, \boldsymbol{\sigma}_{i-1}, \boldsymbol{\alpha}_{i-1})$

assemblage éventuel de la matrice tangente $\mathbf{K}_i^n = \sum_E {}^E\mathbf{K}_i^n$

1.1 Options de calcul concernées

1.1.1 Option FULL_MECA (Full Newton)

Au pas de temps i et à l'itération de Newton n , à partir des contraintes et variables internes à l'équilibre précédent $(\sigma_{i-1}, \alpha_{i-1})$ et de l'incrément de déformation $\varepsilon(\Delta \mathbf{u}_i^n)$ (et éventuellement avec les paramètres : température, hydratation, …), calcul en chaque point de Gauss de chaque élément fini :

•des contraintes et variables internes (SIEF_ELGA,VARI_ELGA) :

$$\begin{cases} \sigma_{i-1}, \alpha_{i-1} \\ \varepsilon(\Delta \mathbf{u}_i^n) \end{cases} \rightarrow \sigma_i^n, \alpha_i^n$$

•de l'opérateur tangent :

$$\begin{cases} \sigma_{i-1}, \alpha_{i-1} \\ \varepsilon(\Delta \mathbf{u}_i^n) \end{cases} \rightarrow \left(\frac{\partial \sigma}{\partial \varepsilon}\right)_i^n$$

Cette option est calculée si REAC_ITER=m dans le fichier de commandes, et que le numéro d'itération n est multiple de m (réactualisation de la matrice tangente cohérente).

1.1.2 Option RAPH_MECA (Newton-Raphson)

Au pas de temps i et à l'itération de Newton n , à partir des contraintes et variables internes à l'équilibre précédent $(\sigma_{i-1}, \alpha_{i-1})$ et de l'incrément de déformation $\varepsilon(\Delta \mathbf{u}_i^n)$ (et éventuellement avec les paramètres : température, hydratation, …), calcul en chaque point de Gauss de chaque élément fini :

•des contraintes et variables internes (SIEF_ELGA,VARI_ELGA) :

$$\begin{cases} \sigma_{i-1}, \alpha_{i-1} \\ \varepsilon(\Delta \mathbf{u}_i^n) \end{cases} \rightarrow \sigma_i^n, \alpha_i^n$$

Cette option est calculée si REAC_ITER=0 ou REAC_ITER=m dans le fichier de commandes, et que le numéro d'itération n n'est pas multiple de m .

1.1.3 Option RIGI_MECA_TANG (calcul de la matrice tangente en prédiction)

A l'itération 0 du pas de temps i (initialisation de l'algorithme de Newton), on choisit comme matrice tangente de prédiction la matrice tangente à l'équilibre précédent ($i-1$), soit $\mathbf{K}_i^0 = \mathbf{K}_{i-1}$. A partir des contraintes et variables internes à l'équilibre précédent $(\sigma_{i-1}, \alpha_{i-1})$, calcul en chaque points de Gauss de chaque élément fini :

•de l'opérateur tangent en prédiction :

$$\sigma_{i-1}, \alpha_{i-1} \rightarrow \left(\frac{\partial \sigma}{\partial \varepsilon}\right)_i^0$$

Cette option est calculée si REAC_INCR=m dans le fichier de commandes, et que le numéro de pas de temps i est multiple de m (réactualisation de l'opérateur tangent en prédiction).

1.2 Remarques concernant le calcul du résidu et de la matrice tangente

1.2.1 Calcul du résidu

Le calcul exact du résidu $\mathbf{L}_i^{\text{meca}} - \mathbf{L}^{\text{int}}(\Delta \mathbf{u}_i^n, \boldsymbol{\sigma}_{i-1}, \boldsymbol{\alpha}_{i-1})$ (et donc des contraintes et des variables internes) est fondamental : il garantit que l'on convergera vers la solution du problème. U ne petite erreur dans l'évaluation du résidu peut avoir des conséquences graves

1.2.2 Calcul de la matrice tangente

Matrice tangente dite cohérente ou consistante (Option FULL_MECA) :

Réactualisée à chaque itération, elle assure la meilleure vitesse de convergence (quadratique) à l'algorithme de Newton (figure 1.2.2-1). Son calcul reste cependant coûteux, et dans le cas où l'on utilise un solveur direct, il faut ajouter au coût de chaque réactualisation celui d'une factorisation. Enfin, pour de grands incréments de chargement, la matrice tangente cohérente peut conduire à des divergences de l'algorithme.

Autres matrices « tangentes » :

On peut faire des erreurs ou des approximations dans le calcul de la matrice "tangente" : ceci conduit à dégrader la vitesse de convergence par rapport à celle qui est obtenue avec la matrice tangente cohérente réactualisée à chaque itération, mais la solution obtenue reste juste tant que le résidu est calculé de manière exacte. Il existe plusieurs variantes (méthodes de quasi-Newton) possibles autorisées par STAT_NON_LINE (pour plus de détails voir [R5.03.01]) :

•Matrice élastique (figure 1.2.2-2)

- calculée une seule fois (économique) à partir des paramètres d'élasticité
- recommandée en cas de décharge
- convergence lente mais assurée

•Matrice tangente réactualisée tous les i_0 incréments de charge (figure 1.2.2-3) ou toutes les n_0 itérations de Newton (figure 1.2.2-4)

- coût moindre
- direction moins bien évaluée
- diverge parfois dans les zones de forte non linéarité

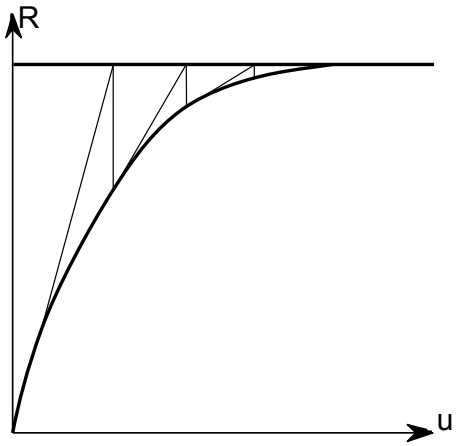


Figure 1.2.2-1: matrice tangente réactualisée à chaque itération

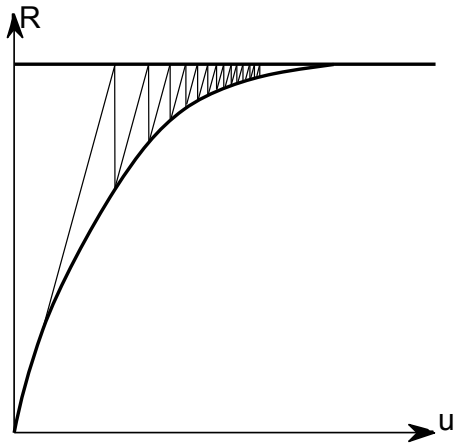


Figure 1.2.2-2: matrice élastique

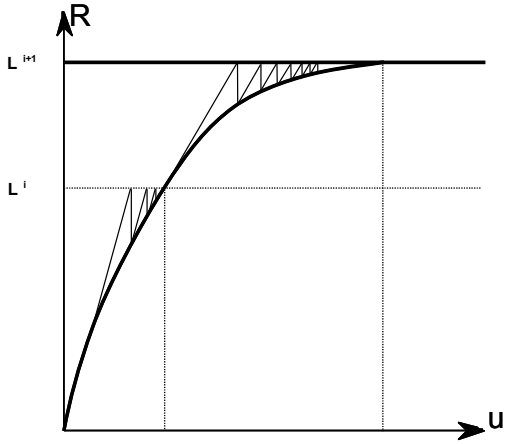


Figure 1.2.2-3: matrice tangente réactualisée à chaque incrément de chargement

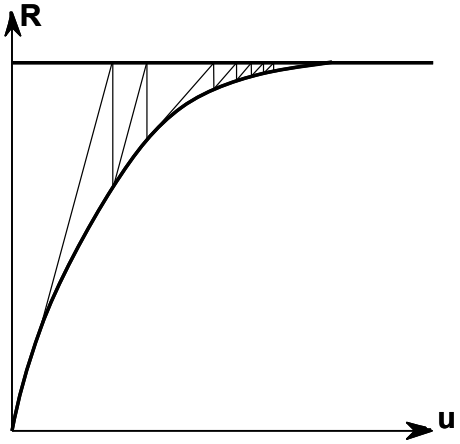


Figure 1.2.2-4: matrice tangente réactualisée toute les deux itérations de Newton

2 Documentation de Référence et choix de la méthode d'intégration

L'écriture de la Documentation de Référence est préalable à la phase de développement. Le document (*cf.* par exemple [R5.03.02]) doit spécifier selon le type de modélisation concernée (milieux continus 2D D_PLAN / AXIS et 3D , ou modèles à plasticité locale tels que les coques, les plaques et les tuyaux et C_PLAN ...) :

- le choix de la méthode d'intégration (voir les différentes possibilités détaillées au § 8) ;
- les équations permettant de calculer les contraintes et variables internes ;
- les équations permettant de calculer les deux matrices tangentes (options RIGI_MECA_TANG et FULL_MECA).

On peut citer d'autres exemples de Documentations de Référence :

- [R5.03.04] : *Relations de comportement élasto-visco-plastique de Chaboche.*
- [R5.03.16] : *Relation de comportement élasto-plastique à écrouissage cinématique linéaire et isotrope non linéaire*.
- [R5.03.20] : *Relation de comportement élastique non linéaire en grands déplacements.*
- [R5.03.21] : *Modélisation élasto-plastique avec écrouissage isotrope en grandes déformations.*

3 Mode opératoire : les catalogues

3.1 Modification du catalogue de DEFI_MATERIAU

Le but de l'opérateur DEFI_MATERIAU [U4.43.01] est d'introduire des paramètres de comportement. Ces paramètres peuvent être communs à plusieurs relations de comportement (voir l'exemple ci-dessous pour les relations de comportement VMIS_ISOT_LINE et VMIS_CINE_LINE).

Il faut donc ajouter dans le catalogue defi_materiau.capy (répertoire catapy/commande) un mot clé facteur correspondant au type de comportement que l'on souhaite introduire, et sous ce mot clé facteur, ajouter les mots clés simples représentant les paramètres de ce type de comportement.

Exemple :

```
ECRO_LINE = FACT(statut='f',
       D_SIGM_EPSI = SIMP(statut='o',typ='R',),
```

```
        SY          = SIMP(statut='o',typ='R',),),...
```

signifie que les deux mots-clés SY et D_SIGM_EPSI sont obligatoires pour ECRO_LINE (pour plus de précisions, se reporter à [U1.03.01] : *Superviseur et langage de commandes*)

3.2 Modification du catalogue C_RELATION

Il faut ajouter à la liste renvoyée par C_RELATION() le nom choisi pour la relation de comportement que l'on souhaite introduire ('MA_RELATION' dans l'exemple ci-dessous). Le catalogue à modifier est c_relation.capy (répertoire catapy/commun).

Exemple :

```
def C_RELATION() : return (             "ELAS",             #COMMUN#
                                        ...
                                        "LAIGLE",
                                        "LEMAITRE",
                                        "LEMAITRE_IRRA",
                                        "LEMA_SEUIL",
                                        "LETK",
                                        "LMARC_IRRA",
                                        "MA_RELATION",
                                        "MAZARS",
                                        "MAZARS_1D",
                                        ...
                                    )
```

3.3 Ajouter le catalogue de la loi de comportement

Ce catalogue est à ajouter dans le répertoire bibpyt/Comportement.

Exemple : vmis_cine_line.py

```
loi = LoiComportement(
   nom            = 'VMIS_CINE_LINE',
   doc            = """Loi de Von Mises... [R5.03.02]""",
   num_lc         = 3,
   nb_vari        = 7,
   nom_vari       = ('XCINXX', 'XCINYY', 'XCINZZ', 'XCINXY', 'XCINXZ',
                     'XCINYZ', 'INDIPLAS',),
   mc_mater       = ('ELAS', 'ECRO_LINE'),
   modelisation   = ('3D', 'AXIS', 'D_PLAN', '1D'),
   deformation    = ('PETIT', 'PETIT_REAC',
                     'GROT_GDEP','GDEF_LOG','GDEF_HYPO_ELAS'),
   nom_varc       = ('TEMP',),
   algo_inte      = ('ANALYTIQUE',),
   type_matr_tang = ('PERTURBATION', 'VERIFICATION'),
   proprietes     = None,
)
```

On fournit donc dans ce catalogue une grande partie des informations relatives au comportement :

- nom : nom de la loi, identique à celui fourni pour COMPR_INCR / RELATION
- num_lc : numéro de routine lc00nn
- nb_vari / nom_vari : nombre de variables internes, et leurs noms (K8)
- mc_mater : mots-clés utilisés dans DEFI_MATERIAU
- modelisation : types de modélisations possibles, pour les comportements de milieux continus : 3D, D_PLAN, AXIS, C_PLAN, COMP1D, INCO, GRADEPSI, GRADVARI, …

•deformation : type de déformations possibles :'PETIT', 'PETIT_REAC', 'GROT_GDEP', 'GDEF_LOG', 'GDEF_HYPO_ELAS'.
•nom_varc : nom des variables de commandes prises en compte
•algo_inte: schémas d'intégration possibles: implicite ('ANALYTIQUE','NEWTON_PERT'...), explicite ('RUNGE_KUTTA')
•type_matr_tang : types de matrices tangentes disponibles. Outre la matrice par perturbation, on peut aussi utiliser les matrices sécantes, et la combinaison TANGENTE_SECANTE.

Remarque :
Les noms des variables internes de l'ensemble des comportements sont définis dans le catalogue python cata_vari.py, afin de nommer de façon identique les variables internes de même signification. Ce catalogue est disponible dans le répertoire bibpyt/Comportement. Pour un nouveau comportement, il est souhaitable de réutiliser des noms déjà existants. Si on ajoute de nouveaux noms, une erreur se produit à l'exécution ; il faut donc modifier cata_vari.py, en justifiant son choix lors de la restitution.

Attention :
Lorsque que l'on ajoute un nouveau catalogue, bien vérifier la présence de la carte d'ajout en tête de fichier. Elle précise dans quelle bibliothèque python placer le fichier (ici Comportement) :
#@ AJOUT maloidecomportement Comportement

4 Mode opératoire : les routines à écrire

C'est à ce niveau que doit être fait le choix du type d'intégration. Il existe quatre possibilités :

1.Utiliser l'architecture de l'environnement d'intégration explicite par un schéma de Runge-Kutta d'ordre 2 [R5.03.14] (ALGO_INTE='RUNGE_KUTA') :

•il s'agit de la méthode la plus simple. Outre la récupération des données matériaux, il suffit d'écrire une routine calculant les dérivées des variables internes
•le calcul de l'opérateur tangent n'est pas disponible sous cet environnement, c'est l'opérateur de rigidité élastique qui est utilisé

1.Implantation « complète » du nouveau comportement dans l'environnement d'intégration implicite PLASTI [R5.03.14] (ALGO_INTE='NEWTON') :

•résolution du système non linéaire local par la méthode de Newton. Outre la récupération des données matériaux, il faut écrire plusieurs routines appelées dans l'algorithme de Newton local (évaluation du seuil, calcul du résidu, calcul analytique de la matrice jacobienne...)
•L'opérateur cohérent est obtenu directement à partir de la jacobienne du système local, et l'opérateur tangent en prédiction est par défaut l'opérateur de rigidité élastique
•PLASTI ne permet pas d'obtenir des modèles optimisés en temps calcul

1.Implantation «facile » du nouveau comportement dans l'environnement d'intégration implicite PLASTI avec [R5.03.14] (ALGO_INTE='NEWTON_PERT') :

•le système non-linéaire local peut être réécrit de telle sorte que l'évaluation du résidu ne nécessite de la part du développeur que de spécifier l'expression des dérivées des variables internes écrites dans le cadre de la méthode 2 (intégration explicite par RK2). On peut donc également réaliser une intégration implicite dans PLASTI avec les deux seules routines nécessaires à l'intégration explicite.
•La jacobienne du système local est calculée par perturbation, le calcul est donc encore plus coûteux qu'avec la méthode 2. De la même manière, l'opérateur tangent cohérent est obtenu directement à partir de la jacobienne du système local

1.Créer une routine autonome d'intégration complète du comportement :

- permet souvent d'obtenir les modèles les plus performants (par exemple, en réduisant le système à résoudre à une seule équation scalaire, non linéaire, voir par exemple [R5.03.04], [R5.03.16], [R5.03.21], ...)
- nécessite plus de travail « sur le papier » pour optimiser les équations

Attention :

Dans le cas 4, on choisira un numéro de routine nn et on écrira la routine lc00nn. Dans les autres cas on choisira comme point d'entrée le numéro 32 : LC0032 appelle PLASTI ou NMVPRK (Runge-Kutta) suivant la valeur de ALGO_INTE choisie par l'utilisateur.

4.1 Première possibilité : introduire un nouveau comportement explicite – schéma de RUNGE - KUTTA

Ce type d'intégration correspond à ALGO_INTE='RUNGE_KUTTA', c'est la façon la plus rapide d'introduire un nouveau comportement.

Il faut écrire dans un premier temps une routine XXXMAT appelée par la routine d'aiguillage LCMATE afin de récupérer les paramètres matériaux et la taille du système différentiel non-linéaire à intégrer.

```
SUBROUTINE XXXMAT(FAMI,KPG,KSP,MOD,IMAT,NMAT,MATERD,MATERF,
                  MATCST,NDT,NDI,NR,NVI,VIND)
```

Arguments en entrée :

```
FAMI,KPG,KSP : famille et numéro de point de gauss / sous-point
IMAT         : adresse du matériau
MOD          : type de modelisation
NMAT         : dimension de MATERD / MATERF
```

Arguments en sortie :

```
MATERD       :  coefficients matériau a t
MATERF       :  coefficients matériau a t+dt
                MATERx(*,1) = caractéristiques élastiques
                MATERx(*,2) = caractéristiques plastiques
MATCST       :  'oui' si  matériau a t = matériau a t+dt
                'non' sinon
NDT          :  nb total de composantes des tenseurs
NDI          :  nb de composantes directes des tenseurs
NR           :  nb de composantes du système non-linéaire
NVI          :  nb de variables internes
```

On donne ci-dessous un exemple pour chacune des deux fonctions principales que doit remplir cette routine

• AFFECTATION DES DIMENSIONS DU PROBLEME LOCAL (NDT,NDI,NR,NVI)

```
NVI=7
IF ( MOD .EQ.'3D' ) THEN
   NDT = 6
   NDI = 3
   NR  = NDT+2
ELSE IF ( MOD .EQ. 'D_PLAN' .OR. MOD .EQ. 'AXIS') THEN
   NDT = 4
   NDI = 3
   NR  = NDT+2
ELSE
```

```
      CALL U2MESS('F',...)
    ENDIF
```

•RECUPERATION DU MATERIAU

```
 NOMC(1) = 'E'
 NOMC(2) = 'NU'
 NOMC(3) = 'ALPHA'
 NOMC(4) = 'SY'
 NOMC(5) = 'D_SIGM_EPSI'
 CALL RCVALB(FAMI,KPG,KSP,'-',IMAT,' ','ELAS',0,' ',
&            0.D0,3,NOMC(1),MATERD(1,1),ICODRE,1)
 CALL RCVALB(FAMI,KPG,KSP,'-',IMAT,' ','ECRO_LINE',0,' ',
&            0.D0,2,NOMC(4),MATERD(1,2),ICODRE,1)
```

Il faut ensuite écrire une routine RKDXXX appelée par la routine d'aiguillage LCDVIN et donnant les dérivées temporelles des variables internes.

Exemples de routine RKDXXX : RKDCHA, RKDVEC, RKDHAY.

4.2 Deuxième possibilité : introduction « complète » d'un nouveau comportement dans PLASTI (implicite)

Cet type d'intégration correspond à ALGO_INTE='NEWTON'. L'environnement PLASTI permet d'intégrer de manière systématique des relations de comportement non-linéaires par une méthode de Newton locale (au niveau du point de Gauss). Connaissant les contraintes et les variables internes à l'instant $i-1$ ainsi que l'incrément de déformation totale $\Delta\boldsymbol{\varepsilon}_i^n$ donné par l'algorithme de Newton global, le système local d'équations à résoudre sous forme purement implicite est écrit de la façon suivante :

$$R(\Delta\boldsymbol{y})=\begin{pmatrix} g(\Delta\boldsymbol{y}) \\ l(\Delta\boldsymbol{y}) \\ f(\Delta\boldsymbol{y}) \end{pmatrix}=0 \text{ avec } \Delta\boldsymbol{y}=\begin{pmatrix} \Delta\boldsymbol{\sigma} \\ \Delta \text{ vari} \\ \Delta p \end{pmatrix}$$

La première équation représente par exemple la relation contrainte-déformation élastique (6 équations à 6 inconnues), avec $\mathbf{A}$ l'opérateur d'élasticité (éventuellement modifié pour les lois avec endommagement), $\Delta\boldsymbol{\varepsilon}^p$ la variation de déformation plastique et $\Delta\boldsymbol{\varepsilon}^{th}$ la variation de déformation thermique :

$$g(\Delta\boldsymbol{y})=\Delta\boldsymbol{\sigma}-\mathbf{A}(\Delta\boldsymbol{\varepsilon}_i^n-\Delta\boldsymbol{\varepsilon}^{th}-\Delta\boldsymbol{\varepsilon}^p)=0$$

la seconde représente l'ensemble des lois d'évolution des différentes variables internes scalaires et/ou vectorielles (n_v équations scalaires à n_v inconnues)

$$l(\Delta\boldsymbol{y})=0$$

la dernière représente le critère éventuel de plasticité (1 équation)

$$f(\Delta\boldsymbol{y})=0$$

Ce système de $6+n_v(+1)$ équations à $6+n_v(+1)$ inconnues est résolu par une méthode de Newton :

$$\begin{cases} \dfrac{\partial R}{\partial \Delta y}(\Delta y_k).d[\Delta y_k] = -R(\Delta y_k) \\ \Delta y_{k+1} = \Delta y_k + d(\Delta y_k) \end{cases}$$

A convergence,on obtient donc les incréments de contraintes et de variables internes. L'opérateur tangent cohérent est quant à lui calculé de manière systématique à partir de la jacobienne du système local par la routine LCOPTG (voir [R5.03.14] pour le détail des équations). Il faut donc programmer a minima une routine définissant le résidu (formé des équations ci-dessus) ainsi qu'une routine construisant la matrice jacobienne. On décrit brièvement ci-dessous l'architecture générale de PLASTI, en indiquant au fur et à mesure la liste des routines à écrire.

Architecture générale de PLASTI :

```
CALL LCMATE(...)
→ écriture nécessaire d'une routine spécifique XXXMAT de récupération du
  matériau identique à RUNGE_KUTTA

IF ( OPT .EQ. 'RAPH_MECA' .OR. OPT .EQ. 'FULL_MECA' ) THEN
  INTEGRATION ELASTIQUE SUR DT
  CALL LCELAS(...)
  → écriture éventuelle d'une routine spécifique XXXELA (par défaut
    LCELIN : élasticité linéaire)

  PREDICTION ETAT ELASTIQUE A T+DT
  CALL LCCNVX(...,SEUIL)
  → écriture nécessaire d'une routine spécifique XXXCVX d' évaluation du
    seuil

  IF ( SEUIL .GE. 0.D0 ) THEN
    CALL LCPLAS(...)
  ENDIF
ENDIF
```

La routine LCPLAS appelle LCPLNL, qui réalise **la boucle de Newton** dont la structure est la suivante :

Notations :

YD=(SIGD,VIND) : vecteur des inconnues (de dimension $6+n_v$) à l'instant T
YF=(SIGF,VINF) : vecteur des inconnues à l'instant T+DT
DY : incrément du vecteur des inconnues entre les instants T et T+DT
DDY : incrément de vecteur des inconnues entre deux itérations de Newton successives
R : résidu
DRDY :jacobienne

On résout donc :	R(DY) = 0
Par une méthode de Newton	DRDY(DYK) DDYK = - R(DYK)
	DYK+1 = DYK + DDYK (DY0 DEBUT)
et on réactualise	YF = YD + DY

```
CALCUL DE LA SOLUTION D ESSAI INITIALE DU SYSTEME NL EN DY
   CALL LCINIT(...DY,...)
   → écriture éventuelle d'une routine spécifique XXXINI (par défaut DY est
     initialisé à 0)

ITERATIONS DE NEWTON
   ITER = 0
   1 CONTINUE
     ITER = ITER + 1
```

```
INCREMENTATION DE  YF = YD + DY
      CALL LCSOVN(NR,YD,DY,YF)

CALCUL DES TERMES DU SYSTEME A T+DT = -R(DY)
      CALL LCRESI(...,DY,R,IRET)
    → écriture nécessaire d'une routine spécifique XXXRES de calcul du
      résidu

CALCUL DU JACOBIEN DU SYSTEME A T+DT = DRDY(DY)
      si ALGO_INTE='NEWTON' calcul exact de la jacobienne
        CALL LCJACB(...DY,...DRDY,IRET)
        → écriture nécessaire d'une routine spécifique XXXJAC
      sinon si ALGO_INTE='NEWTON_PERT', calcul par perturbation (cf § 12)
        CALL LCJACP(...DRDY,...)

RESOLUTION DU SYSTEME LINEAIRE DRDY(DY).DDY = -R(DY)
      CALL LCEQMN(NR,DRDY,DRDY1)
      CALL LCEQVN(NR,  R ,  DDY)
      CALL MGAUSS('NCWP',DRDY1,DDY,NR,NR,1,RBID,IRET )

REACTUALISATION DE DY = DY + DDY
      CALL LCSOVN ( NR , DDY , DY , DY )

RECHERCHE LINEAIRE dans le cas ALGO_INTE='NEWTON_RELI'
      CALL LCRELI ( …)

ESTIMATION DE LA CONVERGENCE
      CALL LCCONV(DY,DDY,NR,ITMAX,TOLER,...,R,...,IRTET)
      → écriture éventuelle d'une routine spécifique XXXCVG du critère de
        convergence (critère relatif par défaut dans LCCONG)
      IF ( IRTET.GT.0 ) GOTO 1

CONVERGENCE -> INCREMENTATION DE  YF = YD + DY
    CALL LCSOVN ( NDT+NVI , YD , DY , YF )

MISE A JOUR DE SIGF , VINF
    CALL LCEQVN ( NDT ,   YF(1)     , SIGF )
    CALL LCEQVN ( NVI-1 , YF(NDT+1) , VINF )
```

En résumé, pour une introduction « complète » d'un nouveau comportement dans PLASTI, il faut nécessairement écrire les routines spécifiques suivantes :

- XXXMAT appelée par LCMATE : récupération du matériau et de la taille du problème local
- XXXCVX appelée par LCCNVX : évaluation du seuil
- XXXRES appelée par LCRESI : calcul du résidu
- XXXJAC appelée par LCJACB : calcul de la jacobienne

Il peut aussi être utile, selon le besoin, d'écrire les routines spécifiques suivantes :

- XXXELA appelée par LCELAS : intégration élastique (si élasticité non-linéaire)
- XXXINI appelée par LCINIT : initialisation (pour une initialisation autre que DY0=0)
- XXXCVG appelée par LCCONV : pour modifier le critère de convergence

4.3 Troisième possibilité : utilisation des routines de l'intégration explicite dans une intégration implicite avec PLASTI

4.3.1 Principe

Ce dernier cas correspond à ALGO_INTE_='NEWTON_PERT', il s'agit de la méthode d'implantation «facile » du nouveau comportement dans l'environnement d'intégration implicite PLASTI. Il est possible d'utiliser directement les deux routines XXXMAT et RKDXXX (récupération des données matériau, et dérivées des variables internes) utilisées avec ALGO_INTE_='RUNGE_KUTTA' pour réaliser un intégration implicite. En effet, le système d'équations différentielles résolu par RUNGE_KUTTA peut s'écrire :

$$\begin{cases} \Delta\boldsymbol{\sigma} = \mathbf{A}\left(\Delta\boldsymbol{\varepsilon}_i^n - \Delta\boldsymbol{\varepsilon}^{th} - \Delta\boldsymbol{\varepsilon}^p(\boldsymbol{Y})\right) \\ \dfrac{d\boldsymbol{Y}}{dt} = F(\boldsymbol{Y}, t; \boldsymbol{\sigma}) \end{cases}$$

où $\boldsymbol{Y}$ représente l'ensemble des variables internes du modèle. La relation entre le tenseur des contraintes et la partie élastique du tenseur des déformations est généralement linéaire, mais peut être évaluée de façon non linéaire par une expression spécifique.

Une fois programmée la routine RKDXXX permettant de calculer $\dfrac{d\boldsymbol{Y}}{dt} = F(\boldsymbol{Y}, t; \boldsymbol{\sigma})$, il est possible de l'utiliser pour une intégration implicite, ce qui consiste à résoudre (*cf.* [R5.03.14]) :

$$R(\Delta\boldsymbol{Z}) = 0 = \begin{bmatrix} R_1(\Delta\boldsymbol{Z}) \\ R_2(\Delta\boldsymbol{Z}) \end{bmatrix}, \text{ avec } \Delta\boldsymbol{Z} = \begin{pmatrix} \Delta\boldsymbol{\sigma} \\ \Delta\boldsymbol{Y} \end{pmatrix} = \boldsymbol{Z}(t+\Delta t) - \boldsymbol{Z}(t)$$

- Le premier système d'équations représente la relation contrainte - déformation élastique
  $$R_1(\Delta Z) = \boldsymbol{A}^{-1}\boldsymbol{\sigma} - (\Delta\boldsymbol{\varepsilon}_i^n - \Delta\boldsymbol{\varepsilon}^{th} - \Delta\boldsymbol{\varepsilon}^p(\boldsymbol{Y})) = \boldsymbol{A}^{-1}\boldsymbol{\sigma} - G(\boldsymbol{Y}) = 0$$
  Par convention, les premières valeurs de $\boldsymbol{Y}$ représentent la variation de déformation plastique, pour faciliter le calcul de $G(\boldsymbol{Y})$ (voir la routine LCRESA pour plus de détails)
- Le deuxième exprime les lois d'évolution des différentes variables internes, soit après discrétisation temporelle par un schéma d'Euler implicite :
  $$R_2(\Delta Z) = \Delta Y - \Delta t \,.\, F(Y, \boldsymbol{\sigma}) = 0$$

Ce système est résolu par la méthode de Newton proposée dans l'environnement PLASTI et décrite au paragraphe précédent:

$$\begin{cases} \dfrac{\partial R}{\partial \Delta Z}\, d(\Delta Z_k) = -R(\Delta Z_k) \\ \Delta Z_{k+1} = \Delta Z_k + d(\Delta Z_k) \end{cases}$$

Les quantités G et F intervenant dans le résidu sont calculées par la routine « explicite » RKDXXX à écrire, et le résidu est construit automatiquement par la routine LCRESA. Le matrice jacobienne est calculée automatiquement par perturbation (routine LCJACP). L'opérateur tangent cohérent est quant à lui calculé de manière systématique à partir de la jacobienne (routine LCOPTG, voir [R5.03.14] pour le détail des équations).

En résumé, ce procédé permet, avec les deux seules routines nécessaires à l'intégration explicite, (coefficients matériau et calcul des dérivées des variables internes) d'utiliser une intégration implicite, et de bénéficier d'une matrice tangente. Ce procédé est économique en termes de temps de développement, mais *a priori* moins efficace en temps CPU qu'une matrice jacobienne programmée explicitement.

On détaille ci-dessous le calcul de l'opérateur tangent par perturbation, ainsi que le critère de convergence de l'algorithme de Newton local.

Calcul par perturbation (LCJACP) : différences finies d'ordre 2

- Initialisation de la perturbation : $\eta = 10^{-7} \|\Delta Z\|$
- boucle sur les colonnes j de la matrice à remplir :
  - calcul de $R(\Delta Z + \eta I_j)$ avec $I_j = [0 \ 0 \ \ldots \ 1 \ \ldots \ 0 \ 0]^T$ vecteur nul sauf à la ligne j
  - calcul de $R(\Delta Z - \eta I_j)$
  - calcul de la colonne j : $\left[\frac{\partial R}{\partial \Delta Z}\right]_{\ldots,j} \simeq \frac{R(\Delta Z + \eta I_j) - R(\Delta Z - \eta I_j)}{2\eta}$

Critère de convergence (LCCONG) :

On sépare les deux blocs R_1 et R_2 du résidu pour éviter les problèmes dus à des ordres de grandeur différents. A chaque itération k de l'algorithme de Newton local, on calcule :

$err_1 = \frac{\left\|R_1(\Delta Z_k)\right\|_\infty}{\left\|R_1(\Delta Z_0)\right\|_\infty}$, avec $R_1(\Delta Z_0) = \Delta \varepsilon_i^n$ car on a $\Delta Z_0 = 0$ à l'initialisation. (cf. § 10)

$err_2 = \frac{\left\|R_2(\Delta Z_k)\right\|_\infty}{\left\|Y(t) + \Delta Y_k\right\|_\infty}$

Le critère d'arrêt est alors le suivant :

$max(err_1, err_2) < \xi$, où ξ est donné par RESI_INTE_RELA.

4.3.2 Exemple

Un exemple : la loi visco-élasto-plastique de Hayhurst.

Routine de lecture des coefficients matériau (appelée par la routine d'aiguillage LCMATT) :
- HAYMAT

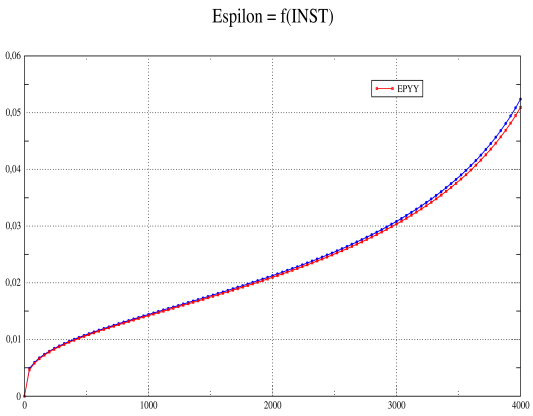
Routine de calcul des dérivées des variables internes (appelée par la routine d'aiguillage LCDVIN) :
- RKDHAY

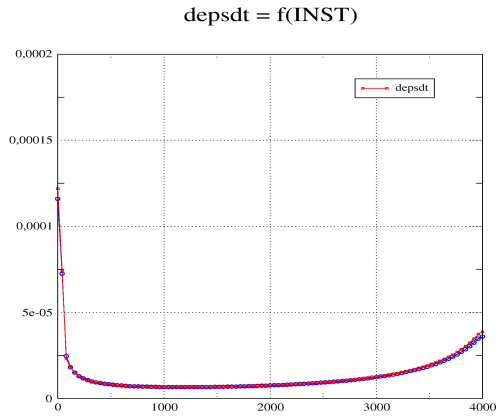
Les développements explicite / implicite de cette relation de comportement sont testés et comparés dans le cas-test ssnv225 [V6.04.225]. Il s'agit d'un test au point matériel de fluage en grandes déformations permettant de valider les capacités du modèle de HAYHURST à représenter le fluage primaire, secondaire et tertiaire. Voici les caractéristiques d'exécution pour ce test en version 11.2 :

Modélisation A : ALGO_INTE='RUNGE_KUTTA'
- temps CPU : 90.59 s
- Nombre de pas de temps : 1660
- Nombre d'itérations de Newton : 7615

Modélisation B : ALGO_INTE='NEWTON_PERT' :
- temps CPU : 27.38 s
- Nombre de pas de temps : 520
- Nombre d'itérations de Newton : 1404

Les résultats sont quasi identiques :





4.4 Quatrième possibilité : écrire une routine lc00nn autonome

4.4.1 lc00nn : routine relative à un point d'intégration d'un élément, spécifique à une loi de comportement

Chercher dans le répertoire bibfor/algorith un numéro de routine lc00nn non utilisé $(50< nn< 100)$) et partir de cette routine vide.

Remarque :

L'appel à lc00nn par lc0000 (qui est la routine appelant toute les routines d'intégration des comportement disponibles dans Code_Aster) est déjà écrit. Cependant, il faut s'assurer que les arguments (ainsi que l'ordre de ces arguments) choisis à la déclaration de lc00nn par le développeur soient les mêmes qu'à l'appel dans lc0000.

Les arguments d'entrée d'une routine lc00nn sont *a minima* :

FAMI	famille de points de gauss (RIGI,MASS,...)
KPG,KSP	numéro du point de gauss et du sous-point
NDIM	dimension de l espace (3d=3 , 2d=2 , 1d=1)
IMATE	adresse du matériau
COMPOR	infos sur le comportement
	compor(1) = relation de comportement (vmis_cine_...)
	compor(2) = nombre de variables internes
	compor(3) = type de déformation (petit,green...)
CRIT	critères locaux
	crit(1) = nombre d itérations maxi a convergence (ITER_INTE_MAXI)
	crit(3) = valeur de la tolérance de convergence (RESI_INTE_RELA)
INSTAM	instant t-
INSTAP	instant t = t- + dt
EPSM	déformation totale a t- (ou éventuellement le gradient de transformation suivant le type de déformation : les arguments de la routine appelante, lc0000, contiennent la dimension de EPSM, et DEPS),
DEPS	incrément de déformation totale (même remarque)
SIGM	contrainte a t-
VIM	variables internes a t-
OPTION	option de calcul
	RIGI_MECA_TANG -> dsidep(t)
	FULL_MECA -> dsidep(t+dt) , sig(t+dt)
	RAPH_MECA -> sig(t+dt)

ANGMAS	les trois angles (nautiques ou d'Euler) du mot_clef massif
TYPMOD	type de modélisation : 3D, AXIS, D_PLAN, C_PLAN,...
ICOMP	compteur pour le redécoupage local du pas de temps
NVI	nombre de variables internes du comportement

les arguments de sortie sont, selon l'option de calcul :

VIP	variables internes a l'instant actuel (options RAPH_MECA et FULL_MECA)
SIGP	contraintes a l'instant actuel (options RAPH_MECA et FULL_MECA)
DSIDEP	opérateur tangent cohérent ou en vitesse (option FULL_MECA ou RIGI_MECA_TANG).
CODRET	code retour permettant d'indiquer (s'il est non nul) une problème d'intégration locale, donc d'effectuer un redécoupage du pas de temps

Remarques :

- *le cas échéant, il est possible d'utiliser également un tableau de travail en entrée (WKIN dans LC0000, de dimension NWKIN). Ce tableau contient des arguments supplémentaires, par exemple une longueur caractéristique dans le cas des modèles non locaux...*
- *De même, il est possible de transférer des valeurs en sortie de la routine lc00nn (tableau WKOUT). Mais dans ce cas il est nécessaire de définir la dimension et l'usage de ce tableau dans les routines élémentaires appelant le comportement – en plasticité 3D HPP, par exemple, TE0139 / NMPL3D / NMCOMP)*
- *Attention, dans le cas RIGI_MECA_TANG, les tableaux relatifs aux contraintes et variables internes en fin de pas de temps ne sont pas alloués. Il ne faut donc pas les utiliser pour calculer la matrice tangente de prédiction.*

4.4.2 Organisation de la routine à écrire

On prend comme au § 7 l'exemple de la loi de Von Mises à écrouissage cinématique linéaire VMIS_CINE_LINE (num_lc=3 dans vmis_cine_line.py) :

```
 SUBROUTINE LC0003(FAMI,KPG,KSP,NDIM,IMATE,COMPOR,CRIT,INSTAM,
&                  INSTAP,EPSM,DEPS,SIGM,VIM,OPTION,ANGMAS,SIGP,VIP,
&                  TAMPON,TYPMOD,ICOMP,NVI,DSIDEP,CODRET)
```

Cette routine est en fait une routine d'aiguillage pour les comportements VMIS_CINE_LINE et VMIS_ECMI_*. L'intégration de VMIS_CINE_LINE est réalisée dans la routine NMCINE, appelée par lc0003 et dont le contenu est décrit ici brièvement :

• LECTURE DES CARACTERISTIQUES ELASTIQUES DU MATERIAU (TEMPS T = +)

```
 NOMRES(1)='E'
 NOMRES(2)='NU'
 CALL RCVALB(FAMI,KPG,KSP,'+',IMATE,' ','ELAS',0,' ',
&            0.D0,2,NOMRES,VALRES,ICODRE,2)
 E  = VALRES(1)
 NU = VALRES(2)
```

Remarque :

RCVALB est une routine générale permettant d'interpoler les valeurs des coefficients matériaux par rapport aux variables de commandes dont ils dépendent (voir les utilitaires).
RCVARC est une routine qui permet de récupérer la valeur d'une variables de commandes (température, séchage , irradiation, ...) à l'instant considéré, et au point de Gauss considéré (voir les utilitaires). Exemple :

```
CALL RCVARC(' ','TEMP','-',FAMI,KPG,KSP,TM,IRET)
```

```
CALL RCVARC(' ','TEMP','+',FAMI,KPG,KSP,TP,IRET)
```

•LECTURE DES CARACTERISTIQUES D'ECROUISSAGE

```
 NOMRES(1)='D_SIGM_EPSI'
 NOMRES(2)='SY'
 CALL RCVALB(FAMI,KPG,KSP,'+',IMATE,' ','ECRO_LINE',0,' ',
&            0.D0,2,NOMRES,VALRES,ICODRE, 2)
 DSDE=VALRES(1)
 SIGY=VALRES(2)
 C  = 2.D0/3.D0*DSDE/(1.D0-DSDE/E)
```

• CALCUL DES CONTRAINTES ELASTIQUES ET DU CRITERE DE VON MISES

```
 DO 110 K=1,3
   DEPSTH(K)   = DEPS(K) -EPSTHE
   DEPSTH(K+3) = DEPS(K+3)
110  CONTINUE
 EPSMO = (DEPSTH(1)+DEPSTH(2)+DEPSTH(3))/3.D0
 DO 115 K=1,NDIMSI
   DEPSDV(K)   = DEPSTH(K) - EPSMO * KRON(K)
115  CONTINUE
 SIGMO = (SIGM(1)+SIGM(2)+SIGM(3))/3.D0
 SIELEQ = 0.D0
 DO 114 K=1,NDIMSI
   SIGDV(K) = SIGM(K) - SIGMO*KRON(K)
   SIGDV(K) = DEUXMU/DEUMUM*SIGDV(K)
   SIGEL(K) = SIGDV(K) + DEUXMU * DEPSDV(K)
   SIELEQ   = SIELEQ   + (SIGEL(K)-C/CM*VIM(K))**2
114  CONTINUE
 SIGMO = TROISK/TROIKM * SIGMO
 SIELEQ = SQRT(1.5D0*SIELEQ)
 SEUIL  = SIELEQ - SIGY
 DP = 0.D0
 PLASTI=VIM(7)
```

•CALCUL DES CONTRAINTES ET DES VARIABLES INTERNES

Les expressions des contraintes et des variables internes (RAPH_MECA et FULL_MECA) sont données dans [R5.03.02]

```
 IF ( OPTION(1:9) .EQ. 'RAPH_MECA' .OR.
&     OPTION(1:9) .EQ. 'FULL_MECA' ) THEN
   IF (SEUIL.LT.0.D0) THEN
     VIP(7) = 0.D0
     DP = 0.D0
     SIELEQ = 1.D0
     A1 = 0.D0
     A2 = 0.D0
   ELSE
     VIP(7) = 1.D0
     DP = SEUIL/(1.5D0*(DEUXMU+C))
     A1 = (DEUXMU/(DEUXMU+C))*(SEUIL/SIELEQ)
     A2 = (C     /(DEUXMU+C))*(SEUIL/SIELEQ)
   ENDIF
   PLASTI=VIP(7)
   DO 160 K = 1,NDIMSI
     SIGDV(K) = SIGEL(K) - A1*(SIGEL(K)-VIM(K)*C/CM)
     SIGP(K)  = SIGDV(K) + (SIGMO + TROISK*EPSMO)*KRON(K)
     VIP(K)   = VIM(K)*C/CM   + A2*(SIGEL(K)-VIM(K)*C/CM)
```

```
      160    CONTINUE
           ENDIF
```

•CALCUL DE L'OPERATEUR TANGENT 'DSIPSEP' : RIGI_MECA_TANG (en vitesse) ou FULL_MECA (cohérent)

```
           IF ( OPTION(1:14) .EQ. 'RIGI_MECA_TANG' .OR.
          &     OPTION(1:9)    .EQ. 'FULL_MECA' ) THEN
             CALL MATINI(6,6,0.D0,DSIDEP)
             DO 120 K=1,6
               DSIDEP(K,K) = DEUXMU
      120    CONTINUE
             IF ( OPTION(1:14) .EQ. 'RIGI_MECA_TANG') THEN
               DO 174 K = 1,NDIMSI
                 SIGDV(K) = SIGDV(K) - VIM(K)*C/CM
      174      CONTINUE
             ELSE
               DO 175 K = 1,NDIMSI
                 SIGDV(K) = SIGDV(K) - VIP(K)
      175      CONTINUE
             ENDIF
             SIGEPS = 0.D0
             DO 170 K = 1,NDIMSI
               SIGEPS = SIGEPS + SIGDV(K)*DEPSDV(K)
      170    CONTINUE
             A1 = 1.D0/(1.D0+1.5D0*(DEUXMU+C)*DP/SIGY)
             A2 = (1.D0+1.5D0*C*DP/SIGY)*A1
             IF(PLASTI.GE.0.5D0.AND.SIGEPS.GE.0.D0) THEN
               COEF = -1.5D0*(DEUXMU/SIGY)**2 / (DEUXMU+C) * A1
               DO 135 K=1,NDIMSI
                 DO 135 L=1,NDIMSI
                   DSIDEP(K,L) = A2 * DSIDEP(K,L) + COEF*SIGDV(K)*SIGDV(L)
      135      CONTINUE
               LAMBDA = LAMBDA + DEUXMU**2*A1*DP/SIGY/2.D0
             ENDIF
             DO 130 K=1,3
               DO 131 L=1,3
                 DSIDEP(K,L) = DSIDEP(K,L) + LAMBDA
      131      CONTINUE
      130    CONTINUE
           ENDIF
```