Architecture des comportements cristallins

Résumé :

Ce document décrit la structure de l'intégration des comportements cristallins (cf. R5.03.11) et les actions à entreprendre pour ajouter un nouveau comportement cristallin, dans le but d'effectuer des calculs d'agrégats, ou des calculs homogénéisés à l'aide de STAT_NON_LINE et SIMU_POINT_MAT.

Remarque : il est également possible d'utiliser les lois de comportement cristallines disponibles dans le module *Zmat* (cf. [U2.10.01]).

Table des Matières

1 Description générale des comportements cristallins

Un comportement cristallin utilise, en plus de DEFI_MATERIAU, DEFI_COMPOR.

Le nom du comportement dans COMP_INCR est générique : MONOCRISTAL ou POLYCRISTAL.

L'algorithme de résolution est au choix :
- pour le MONOCRISTAL : NEWTON, NEWTON_RELI, NEWTON_PERT et RUNGE_KUTTA
- pour le POLYCRISTAL : RUNGE_KUTTA

On décrit dans ce document :
- l'architecture de DEFI_COMPOR
- l'architecture de la résolution explicite à erreur contrôlée par Runge_Kutta (cf. [R5.03.14])
- l'architecture de la résolution implicite par Newton et ses variantes (environnement PLASTI cf. [R5.03.14])

Pour appréhender ce document, la lecture de R5.03.11 est vivement conseillée.

2 Architecture de DEFI_COMPOR

2.1 Illustration sur un exemple : test SSNV194

Partons d 'un exemple (test ssnv194).

2.1.1 Modélisation A : un petit agrégat comportant 10 grains :

```
ACIER=DEFI_MATERIAU(ELAS=_F(E=145200.0,NU=0.3,),
                    MONO_VISC1=_F(N=10.0, K=40.0,C=1.0,),
                    MONO_ISOT1=_F(R_0=75.5, Q=9.77, B=19.34),
                    MONO_CINE1=_F(D=36.68,),);

MONO1 =DEFI_COMPOR(MONOCRISTAL=(_F(  MATER=ACIER,ELAS='ELAS',
                                     ECOULEMENT='MONO_VISC1',
                                     ECRO_ISOT='MONO_ISOT1',
                                     ECRO_CINE='MONO_CINE1',
                                     FAMI_SYST_GLIS='BCC24',),),);

ORIEN=AFFE_CARA_ELEM(MODELE=TROISD, MASSIF=(
_F(GROUP_MA='GM1',ANGL_EULER=(-150.646,33.864,55.646,),),
_F(GROUP_MA='GM2',ANGL_EULER=(-137.138,41.5917,142.138,),),
_F(GROUP_MA='GM3',ANGL_EULER=(-166.271,35.46958,171.271,),),
_F(GROUP_MA='GM4',ANGL_EULER=(-77.676,15.61819,154.676,),),
_F(GROUP_MA='GM5',ANGL_EULER=(-78.6463,33.864,155.646,),),
_F(GROUP_MA='GM6',ANGL_EULER=(-65.1378,41.5917,142.138,),),
_F(GROUP_MA='GM7',ANGL_EULER=(-94.2711,35.46958,71.271,),),
_F(GROUP_MA='GM8',ANGL_EULER=(-5.67599,15.61819,154.676,),),
_F(GROUP_MA='GM9',ANGL_EULER=(-6.64634,33.864,155.646,),),
_F(GROUP_MA='GM10',ANGL_EULER=(6.86224,41.5917,142.138,),),
                                  ),);

SOLNL=STAT_NON_LINE(MODELE=...,CHAM_MATER=..., EXCIT=_...,
                    CARA_ELEM=ORIEN,
                    COMP_INCR=_F(RELATION='MONOCRISTAL',
                                 COMPOR=MONO1,
                                 ),)
```

2.1.2 Modélisation C : polycristal comportant 10 grains :

Un point matériel, 10 grains de fractions volumiques identiques (0.1), et d'orientations similaires à celles de la modélisation A (ce qui permet de retrouver la même solution moyenne) :

```
MONO1=DEFI_COMPOR(MONOCRISTAL=(... identique à la modélisation A)

COMPORP=DEFI_COMPOR(POLYCRISTAL=(
_F(MONOCRISTAL=COMPORT,FRAC_VOL=0.1,ANGL_EULER=(-150.646,33.864,55.646,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-137.138,41.5917,142.138,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-166.271,35.46958,171.271,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-77.676,15.61819,154.676,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-78.6463,33.864,155.646,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-65.1378,41.5917,142.138,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-94.2711,35.46958,71.271,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-5.67599,15.61819,154.676,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(-6.64634,33.864,155.646,),),
_F(MONOCRISTAL=MONO1,FRAC_VOL=0.1,ANGL_EULER=(6.86224,41.5917,142.138,),),
),
LOCALISATION  ='BETA',DL=0.,DA=0.,);

SOLNL=SIMU_POINT_MAT(COMP_INCR=_F(RELATION='POLYCRISTAL',
                                  COMPOR=COMPORP,
                                  ALGO_INTE='RUNGE_KUTTA',),
                     NEWTON=_F(MATRICE='ELASTIQUE',REAC_ITER=0),
                     MATER =..., NB_VARI_TABLE=6, INCREMENT=...,EPSI_IMPOSE=...
                           );
```

2.2 Description de DEFI_COMPOR

2.2.1 Structure :

La routine OP0050 a pour but de produire la structure de données décrite dans [D4.06.24] : les objets composant cette SD sont différents suivant que l'on traite un monocristal (routine OP5901) ou un polycristal (routine OP5902).

2.2.2 Ajout d'un comportement cristallin au catalogue de DEFI_MATERIAU / DEFI_COMPOR

Si le nouveau comportement cristallin utilise des paramètres matériau différents de ceux qui sont déjà disponibles dans les mots-clés MONO_* de DEFI_MATERIAU il suffit d'introduire ces nouveaux paramètres de comportement, soit sous un seul mot-clé (cas des comportements MONO_DD_*, soit en séparant les coefficients relatifs à l'écrouissage isotrope, à l'écrouissage cinématique, et à l'écoulement (cf. MONO_ISOT*, MONO_CINE*, MONO_VISC*,) . Ces paramètres seront exploités dans l'intégration (routines LCMMAT, LCMMAP), et les mot-clés facteurs correspondants seront utilisés dans DEFI_COMPOR.

Exemple:

```
MONO_DD_CFC =FACT(statut='f',
regles=(  UN_PARMI('H','H1'),
          PRESENT_PRESENT('H1','H2','H3','H4','H5'),
          PRESENT_ABSENT('H','H1','H2','H3','H4','H5'),),

GAMMA0 =SIMP(statut='f',typ='R',defaut=0.001, unités : s**-1"),
TAU_F =SIMP(statut='o',typ='R',fr="en unite de contraintes ex 20 MPa"),
```

```
A =SIMP(statut='f',typ='R',defaut=0.13,fr="paramètre A, sans unité"),
B =SIMP(statut='f',typ='R',defaut=0.005,fr="paramètre B, sans unité"),
N =SIMP(statut='f',typ='R',defaut=200.,fr="paramètre n, sans unité"),
Y =SIMP(statut='o',typ='R',fr="en unité de longueur ex 2.5 A"),
ALPHA=SIMP(statut='f',typ='R',defaut=0.35,fr="paramètre alpha"),
BETA =SIMP(statut='o',typ='R',fr="paramètre b, en unite de longueur"),
...),
```

Ceci permet de décrire chaque coefficient, son caractère facultatif (avec une éventuelle valeur par défaut) ou obligatoire (pour plus de précisions, se reporter à [D5,01,01]) .

Le catalogue de DEFI_COMPOR est, suivant les cas :

```
MONOCRISTAL =FACT(statut='f', max=5,
   MATER =SIMP(statut='o', typ=mater_sdaster, max=1),
   ECOULEMENT=SIMP(statut='o',typ='TXM',into=('MONO_VISC1', 'MONO_VISC2',
                                              'MONO_DD_CFC', 'MONO_DD_CC',...),
                                    fr="type d'écoulement viscoplastique"),
   ELAS =SIMP(statut='f', typ='TXM',),

   # cas d'un comportement de type MONO_VISC*
   b_non_dd=BLOC(condition="ECOULEMENT=='MONO_VISC1'
                            or ECOULEMENT=='MONO_VISC2',
                 ECRO_ISOT=SIMP(statut='f', typ='TXM', max=1,
                    fr="Donner le type d'écrouissage isotrope"),
                 ECRO_CINE=SIMP(statut='f', typ='TXM', max=1,
                    fr="Donner type d'écrouissage cinématique"),
                 FAMI_SYST_GLIS=SIMP(statut='f',typ='TXM',
                    into=('OCTAEDRIQUE','BCC24','CUBIQUE1','CUBIQUE2',
                          'ZIRCONIUM','UNIAXIAL','UTILISATEUR'),),
                 b_util =BLOC(condition="FAMI_SYST_GLIS=='UTILISATEUR' ",
                  TABL_SYST_GLIS =SIMP(statut='f', typ=table_sdaster,),),),

   # cas d'un comportement de type DD
   b_dd_cc=BLOC(condition="ECOULEMENT=='MONO_DD_CC'",
                 FAMI_SYST_GLIS=SIMP(statut='f',typ='TXM',into=('CUBIQUE1','
                 UTILISATEUR',),
                 b_util=BLOC(condition="FAMI_SYST_GLIS=='UTILISATEUR'",
                 TABL_SYST_GLIS=SIMP(statut='f', typ=table_sdaster),),),),

MATR_INTER =SIMP(statut='f', typ=table_sdaster, max=1,),

ROTA_RESEAU=SIMP(statut='f',typ='TXM',max=1,into=('NON','POST','CALC'),
                 defaut='NON',fr="rotation de reseau : NON, POST, CALC"),

POLYCRISTAL =FACT(statut='f', max='**',
              regles=(UN_PARMI('ANGL_REP','ANGL_EULER'),),

   MONOCRISTAL=SIMP(statut='o', typ=compor_sdaster, max=1),
   FRAC_VOL  =SIMP(statut='o', typ='R', fr="fraction volumique »),
   ANGL_REP=SIMP(statut='f',typ='R',max=3,fr="angles nautiques en degrés"),
   ANGL_EULER=SIMP(statut='f',typ='R',max=3,fr="angles d'Euler en degrés"),

LOCALISATION=SIMP(statut='f', typ='TXM',max=1,into=('BZ', 'BETA'),
   fr="Donner le nom de la règle de localisation"),
   b_beta      =BLOC( condition = "LOCALISATION=='BETA'",
                      DL=SIMP(statut='o',typ='R',max=1),
                      DA=SIMP(statut='o',typ='R',max=1),)
```

L'architecture des routines OP5901 et OP5902 est simple, et consiste à remplir la structure de données sd_compor, destinée à préparer les calculs. Pour cela, plusieurs informations sont déduites des données de l'utilisateur :

- Pour le monocristal :
  - le nombre de systèmes de glissement, soit en faisant appel à la routine **LCMMSG,** qui définit les familles de systèmes de glissement pré-établies, soit en lisant la table fournie pour chaque famille (qui est elle-même stockée dans la sd_compor)
  - le nombre de variables internes qui se déduit du nombre de systèmes de glissements total, qui sera associé au comportement MONOCRISTAL dans COMP_INCR.
- Pour le polycristal, :
  - les différents monocristaux relatifs à chaque grain, avec la fraction volumique et l'orientation
  - le règle de localisation et ses paramètres.
  - Le nombre de variables internes total, déduit des monocristaux et du nombre de grains, qui sera associé au comportement POLYCRISTAL dans COMP_INCR.

L'ajout d'un comportement cristallin se réduit donc, dans DEFI_COMPOR, à la modification du catalogue pour la partie MONOCRISTAL (pour la vérification syntaxique). L'ajout d'une famille de systèmes de glissement se traduit également pas une modification du catalogue de DEFI_COMPOR, avec d'éventuels blocs pour gérer les possibilités d'association entre lois d'écoulement et familles de systèmes de glissement.
Pour la partie POLYCRISTAL, l'ajout d'un comportement cristallin ne modifie pas le catalogue de DEFI_COMPOR, Une des seules modifications consisterait en l'ajout d'une règle de localisation.

3 Architecture de l'intégration

3.1 Récupération des coefficients matériau

Quelque soit le type d'intégration choisi (implicite ou explicite) la récupération des caractéristiques matériau et comportement, issues de DEFI_MATERIAU et DEFI_COMPOR, se fait par l'intermédiaire de la routine LCMMAT, appelée par LCMATE, pour le MONOCRISTAL , et par la routine LCMMAP, appelée également par LCMATE, pour le POLYCRISTAL.

Ces routines ont plusieurs fonctions :

- Récupération des valeurs des mot-clés définissant les paramètres, principalement à l'aide de la routine générale RCVALB, et stockage dans deux tableaux (différents seulement si les coefficients dépendent de la température) : MATERD définissant les paramètres à l'instant précédent, c'est à dire au début du pas de temps, et MATERF à l'instant actuel, donc à la fin du pas de temps). Ces tableaux permettent de passer les paramètres matériau aux routines de résolution ;
- Lecture de la sd_compor et stockage des informations (familles de systèmes de glissement, matrice d'interaction, …) dans des tableaux utilisés lors de la résolution.

Architecture de LCMMAT :

LCMMJV : lecture de la sd_compor issue de DEFI_COMPOR
pour chaque famille de systèmes :
  LCMMSG fournit le nombre de systèmes de glissement
    LCMMJS (si il s'agit d'une famille « utilisateur »)
  LCMAFL récupère les coefficients matériau relatifs à l'écoulement
    LCMHSR+LCMHDD : appel spécifique dans cette routine pour MONO_DD_KR
  LCMAEC coefficients matériau relatifs à l'écrouissage cinématique,
  LCMAEI coefficients matériau relatifs à l'écrouissage isotrope,
    LCMHSR : calcul ou lecture de la matrice d'interaction
DMAT3D, D1MA3D : opérateur d'élasticité et son inverse
CALCMM LCMMSG. : calcul et stockage des tenseurs d'orientation de tous les systèmes, pour optimiser les performances.

Dans le cas d'un nouveau comportement monocristallin, il suffit a priori d'intervenir dans les routines LCMAFL, LCMAEI et éventuellement LCMAEC, en ajoutant dans chacune de ces routines le bloc d'instructions nécessaires à la récupération des coefficients matériau de ce comportement.

Il convient également de lui attribuer un numéro : en effet, pour optimiser les performances, il est préférable de lire et comparer des entiers plutôt que ces chaines de caractères ; dans les routines d'intégration, plutôt que de tester :

```
IF (NECOUL.EQ.'MONO_VISC1') THEN..
```

on testera :

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

La nomenclature des numéros de lois d'écoulement est définie dans LCMAFL :

Nom de la loi d'écoulement	Numéro associé
MONO_VISC1	1
MONO_VISC2	2
MONO_DD_KR	4
MONO_DD_CC (athermique)	5
MONO_DD_CFC	5
MONO_DD_FAT	6

Tableau 3.1-1

Les numéros de lois d'écrouissage isotrope sont définis dans LCMAEI :

Nom de la loi d'écoulement	Numéro associé
MONO_ISOT1	1
MONO_ISOT2	2
MONO_DD_CFC	3
MONO_DD_CC (athermique)	3
MONO_DD_FAT	4

Tableau 3.1-2

Les numéros de lois d'écrouissage isotrope sont définis dans LCMAEC :

Nom de la loi d'écoulement	Numéro associé
MONO_CINE1	1
MONO_CINE2	2

Tableau 3.1-3

Architecture de LCMMAP :

Lecture de la sd_compor de type polycristal

Pour chaque comportement monocristal (5 au maximum) utilisé par l'ensemble des grains
pour chaque famille de systèmes : lecture des caractéristiques comme LCMMAT
LCMMSG fournit le nombre de systèmes de glissement
LCMAFL récupère les coefficients matériau relatifs à l'écoulement
LCMAEC coefficients matériau relatifs à l'écrouissage cinématique,
LCMAEI coefficients matériau relatifs à l'écrouissage isotrope, matrice d'interaction

DMAT3D, D1MA3D : opérateur d'élasticité et son inverse

Stockage des informations relatives aux monocristaux utilisés pour chaque phase dans des tableaux spécifiques. Juste avant l'appel à la résolution explicite par Runge_Kutta (routine GERPAS), appel à la routine spécifique CALCMS permettant de stocker dans un unique tableau les systèmes de glissement relatifs à tous les grains, pour optimiser les performances.

3.2 Intégration explicite – schéma de RUNGE - KUTTA

3.2.1 Cas du monocristal :

C'est la façon la plus rapide (mais la moins optimale) d'introduire un nouveau comportement monocristallin en petites déformations : il suffit de d'écrire les dérivées des variables internes dans la routine LCMMON, appelée par RDIF01.

La routine LCMMON a pour but de calculer les dérivées des variables internes. On doit résoudre un système de $6+3*n_s$ équations différentielles, du type :

- $\dot{\varepsilon}^{vp}=\sum_s \mu_s \dot{\gamma}_s$ 6 équations
- pour chaque système de glissement (sur l'ensemble des familles de systèmes) 3 relations :
  - $\dot{\gamma}_s=\dot{p}_s\left(\tau_s(\sigma),\alpha_s,\gamma_s,R_s(p)\right)\eta(\tau_s,\alpha_s)$ où $\eta(\tau_s,\alpha_s)=\pm 1$
  - $\dot{\alpha}_s=h\left(\tau_s,\alpha_s,\gamma_s,p_s\right)$
  - $R_s(p)$

où $\tau_s=\mu_s:\sigma$ où σ est déduite de la relation : $\sigma=\Lambda\left(\varepsilon-\varepsilon^{vp}\right)$

En pratique, la résolution s'effectue de la façon suivante :

La routine LCMMON calcule les contraintes par la relation d'élasticité (isotrope ou orthotrope)

CALL CALSIG(...) ce qui fournit le tenseur SIG = σ

Puis elle calcule les $6+3*n_s$ dérivées issues des équations différentielles ci-dessus, et les stocke dans un tableau DVIN :

- Boucle sur les familles de systèmes de glissement :

```
DO IFA=1,NBFSYS
   …
   Récupération du nombre de systèmes de glissement
   CALL LCMMSG(NOMFAM,NBSYS,0,PGL,MS,NG,LG,0,Q)

   Boucle sur les systèmes de glissement de la famille IFA :
   DO IS=1,NBSYS

      CALL LCMMSG(..)  calcul du tenseur d'orientation MuS

      CALCUL DE LA CISSION REDUITE
      TAUS= SIG(I)*MuS(I) pour i=1,6

       CALL LCMMFI( => RP)          routine de calcul de l'écrouissage isotrope

       CALL LCMMFE(=> DGAMMA, DP)          routine de calcul de l'écoulement

       CALL LCMMEC( => DALPHA )  routine de calcul de l'écrouissage cinématique
```

```
            Calcul de la déformation viscoplastique globale
            DO ITENS=1,6
               DEVI(ITENS)=DEVI(ITENS)+MuS(ITENS)*DGAMMA
            ENDDO

            stockage des dérivées des variables internes pour le système de glissement IS
            DVIN(NUVI-2)=DALPHA
            DVIN(NUVI-1)=DGAMMA
            DVIN(NUVI  )=DP

         ENDDO

      ENDDO

      stockage du tenseur dérivée de la déformation viscoplastique.
      DO ITENS=1,6
         DVIN(ITENS)= DEVI(ITENS)
      ENDDO
```

L'algorithme de Runge-Kutta gère alors l'intégration de ces équations différentielles, en contrôlant l'erreur, et en raffinant le pas de temps jusqu'à obtenir une erreur inférieure à la précision demandée (RESI_INTE_RELA) [R5.03.14].

A priori, la structure de la routine LCMMON, ne doit pas évoluer lors de l'ajout d'un nouveau comportement monocristallin ; seules les routines LCMMFI, LCMMFE et LCMMEC sont à modifier.

Elles sont construites de la façon suivante :

```
LCMMFI
C-----------------------------------------------------------------------
C     POUR UN NOUVEAU TYPE D'ECROUISSAGE ISOTROPE, AJOUTER UN BLOC IF
C-----------------------------------------------------------------------
C     IF (NECRIS.EQ.'MONO_ISOT1') THEN
      IF (NUEISO.EQ.1) THEN
       ….....
            RP=...
C     ELSEIF (NECRIS.EQ.'MONO_ISOT2') THEN
      ELSEIF (NUEISO.EQ.2) THEN
      ….........
         RP=...

C     ELSEIF (NECRIS.EQ.'MONO_DD_CFC') THEN
      ELSEIF (NUEISO.EQ.3) THEN
         ….......
         RP=MU*SQRT(RP)*CEFF
      ENDIF
```

Remarque : on utilise ici les numéros associés à chaque type de comportement plutôt que les noms, pour optimiser les performances.

De même la structure de la routine LCMMFC est :

```
C-----------------------------------------------------------------------
C   POUR UN NOUVEAU TYPE D'ECROUISSAGE CINEMATIQUE, AJOUTER UN BLOC IF
C-----------------------------------------------------------------------
C     IF (NECRCI.EQ.'MONO_CINE1') THEN
      IF (NUECIN.EQ.1) THEN
          DALPHA=...
C     ELSEIF (NECRCI.EQ.'MONO_CINE2') THEN
```

```
      ELSEIF (NUECIN.EQ.2) THEN
      ENDIF
```

Et de façon similaire, la structure de la routine LCMMFC est :

```
C-----------------------------------------------------------
C     POUR UN NOUVEAU TYPE D'ECOULEMENT, CREER UN BLOC IF
C-----------------------------------------------------------
C     IF (NECOUL.EQ.'MONO_VISC1') THEN
      IF (NUECOU.EQ.1) THEN
          DP=...
          DGAMMA= ...
      ELSEIF (NUECOU.EQ.2) THEN
      ENDIF
```

Remarque : Les routines LCMMFE, LCMMFI, LCMMFC sont également utilisées par l'intégration explicite du polycristal et par l'intégration implicite. Ceci simplifie la mise en œuvre d'un nouveau comportement cristallin.

3.2.2 Cas du polycristal :

L'intégration du polycristal s'appuie largement sur celle du monocristal : on calcule là encore les dérivées des variables internes pour chaque monocristal de chaque grain g , dans la routine LCMMOP, appelée par RDIF01.

On doit résoudre un système de $6+n_g\left(6+3*n_s(g)\right)$ équations différentielles, du type :

•pour chaque grain défini par une orientation et une proportion f_g , une relation de localisation des contraintes, de la forme générale :

$\boldsymbol{\sigma_g}=L\left(\boldsymbol{\Sigma},\boldsymbol{E^{vp}},\varepsilon_g^{vp},\beta_g\right)$ avec, $\boldsymbol{\Sigma}=\boldsymbol{\Lambda}(\boldsymbol{\Lambda}_-^{-1})\boldsymbol{\Sigma}^-+\boldsymbol{\Lambda}(\Delta\boldsymbol{E}-\Delta\boldsymbol{E^{th}}-\Delta\boldsymbol{E^{vp}})$

•pour chacun des $n_s(g)$ systèmes de glissement de chaque grain g , 3 relations :

- $\dot{\gamma}_s=\dot{p}_s\left(\tau_s(\boldsymbol{\sigma}),\alpha_s,\gamma_s,R_s(p)\right)\eta(\tau_s,\alpha_s)$ où $\eta(\tau_s,\alpha_s)=\pm1$
- $\dot{\alpha}_s=h\left(\tau_s,\alpha_s,\gamma_s,p_s\right)$
- $R_s(p)$

•a l'échelle du grain, les équations d'homogénéisation : $\boldsymbol{\varepsilon_g^{vp}}=\sum_s \boldsymbol{\mu}_s\dot{\gamma}_s$

En pratique, la résolution s'effectue de la façon suivante :

La routine LCMMOP calcule les contraintes par la relation d'élasticité (isotrope ou orthotrope)
CALL CALSIG(...) ce qui fournit le tenseur SIG ($\boldsymbol{\Sigma}$)

Puis elle calcule les $6+n_g\left(6+3*n_s(g)\right)$ dérivées issues des équations différentielles ci-dessus, et les stocke dans un tableau DVIN :

•Boucle sur les grains :

DO IGRAIN=1,NGRAIN
CALL LCLOCA() relation de localisation permettant de calculer SIGG $(\boldsymbol{\sigma_g})$

•Boucle sur les familles de systèmes de glissement :

```
DO IFA=1,NBFSYS
...
```

Récupération du nombre de systèmes de glissement
CALL LCMMSG(...)

Boucle sur les systèmes de glissement de la famille IFA :
```
DO IS=1,NBSYS

   CALL LCMMSG(..)  calcul du tenseur d'orientation MuS

   CALCUL DE LA CISSION REDUITE
   TAUS= SIGG(I)*MuS(I) pour i=1,6
```

CALL LCMMFI(=> RP) **routine de calcul de l'écrouissage isotrope**

CALL LCMMFE(=> DGAMMA, DP) **routine de calcul de l'écoulement**

CALL LCMMEC(=> DALPHA) **routine de calcul de l'écrouissage cinématique**

Calcul de la déformation viscoplastique globale
```
DO ITENS=1,6
c      DEVG(ITENS)=DEVG(ITENS)+MuS(ITENS)*DGAMMA   (DEVG = ε_g^vp)
ENDDO
```

stockage des dérivées des variables internes pour le système de glissement IS
```
DVIN(NUVI-2)=DALPHA
DVIN(NUVI-1)=DGAMMA
DVIN(NUVI  )=DP

ENDDO

ENDDO
homogénéisation des déformations viscoplastiques
DO I=1,6
   DEVI(I)=DEVI(I)+FV*DEVG(I)
ENDDO
```
stockage du tenseur dérivée de la déformation viscoplastique.
```
DO ITENS=1,6
   DVIN(ITENS)= DEVI(ITENS)
ENDDO
```

La septième variable interne contient la déformation viscoplastique équivalente cumulée
```
DVIN(7)= DVINEQ
```

L'algorithme de Runge-Kutta gère alors l'intégration de ces équations différentielles, en contrôlant l'erreur, et en raffinant le pas de temps jusqu'à obtenir une erreur inférieure à la précision demandée (RESI_INTE_RELA) [R5.03.14].

La structure de la routine LCMMOP, ne doit pas évoluer lors de l'ajout d'un nouveau comportement monocristallin ; seules les routines LCMMFI, LCMMFE et LCMMEC sont à modifier (ce qui est déjà fait en principe pour l'intégration du monocristal). Une modification supplémentaire est à effectuer dans la routine LCLOCA lors de l'ajout d'une nouvelle règle de localisation.

3.3 Intégration implicite du monocristal par Newton dans PLASTI

On intègre cette fois le comportement monocristallin par une méthode de Newton. Cette méthode est programmée dans PLASTI [R5.03.14]. Il faut donc fournir à cet algorithme écrire le système d'équation à résoudre sous forme purement implicite, de la façon suivante : $\boldsymbol{R}(\boldsymbol{Y})=0$

$$R(Y)=\left\{\begin{array}{l}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Sigma}-(\boldsymbol{\Lambda}_{-}^{-1})\boldsymbol{\Sigma}^{-}-(\Delta\boldsymbol{E}-\Delta\boldsymbol{E}^{\mathrm{th}}-\Delta\boldsymbol{E}^{\mathrm{vp}})\\ \Delta\boldsymbol{E}^{\mathrm{vp}}-\sum_s \boldsymbol{\mu}_s\Delta\gamma_s\\ n_s\left\{\begin{array}{l}\Delta\alpha_s-h(\tau_s^+,\alpha_s^+,\gamma_s^+,p_s^+)\\ \Delta\gamma_s-g(\tau_s^+,\alpha_s^+,\gamma_s^+,p_s^+)\\ \Delta p_s-f(\tau_s^+,\alpha_s^+,\gamma_s^+,p_s^+)\end{array}\right.\end{array}\right.=\mathbf{0}$$

C'est un système de $6+6+3n_s$ équations non linéaires (de même taille que celui qui est intégré par la méthode de Runge-Kutta en explicite).

Afin d'optimiser les performances, on résout en fait un système d'équations réduit, de taille $6+n_s$, construit de la façon suivante [R5.03.11] :

- Dans l'expression des 6 composantes du tenseur des contraintes, $\Delta\boldsymbol{E}^{vp}$ peut être exprimée en fonction de $\sum_s \boldsymbol{\mu}_s\Delta\gamma_s$ donc 6 équations peuvent être éliminées du système global à résoudre.
- Comme $\Delta p_s=|\Delta\gamma_s|$, l'équation en Δp_s peut être éliminée, et pour tous les comportements cristallins considérés actuellement, soit $\Delta\alpha_s$ s'exprime directement en fonction de $\Delta\gamma_s$, dans le cas de l'écrouissage cinématique, soit $\Delta\gamma_s$ s'exprime en fonction de $\Delta\alpha_s$ qui représente alors la densité de dislocation (à un facteur près) pour les comportements de type MONO_DD_*.

On obtient donc le système suivant :

- **Petites déformations:**

$$\begin{array}{l}\boldsymbol{R}_1(\boldsymbol{\sigma},\Delta\beta)=\boldsymbol{\Lambda}^{-1}.\Delta\boldsymbol{\sigma}-\Delta\boldsymbol{\varepsilon}+\Delta\boldsymbol{\varepsilon}^{th}+\sum_s \Delta\gamma_s\boldsymbol{\mu}_s=0\\ \boldsymbol{R}_2(\boldsymbol{\sigma},\Delta\beta)=\Delta\beta_s-k_s(\tau_s(\boldsymbol{\sigma}),\Delta\beta)=0\end{array}$$ où l'inconnue est : $\boldsymbol{Y}=\left[\begin{array}{c}\boldsymbol{\sigma}\\ \Delta\beta\end{array}\right]$

avec $\tau_s(\boldsymbol{\sigma})=\boldsymbol{\sigma}:\boldsymbol{\mu}_s$

- **Grandes déformations**

$$\begin{array}{l}\boldsymbol{R}_1(\boldsymbol{S},\Delta\beta)=\boldsymbol{\Lambda}^{-1}.\boldsymbol{S}-\frac{1}{2}\left(\boldsymbol{F}^{\mathrm{eT}}\boldsymbol{F}^{\mathrm{e}}-\boldsymbol{I}_d\right)=0\\ \boldsymbol{R}_2(\boldsymbol{S},\Delta\beta)=\Delta\beta_s-k_s(\tau_s(\boldsymbol{S}),\Delta\beta)=0\end{array}$$ où l'inconnue est : $\boldsymbol{Y}=\left[\begin{array}{c}\boldsymbol{S}\\ \Delta\beta\end{array}\right]$

avec $\tau_s(\boldsymbol{S})=\left[\left(2\boldsymbol{\Lambda}^{-1}\boldsymbol{S}+\boldsymbol{I}_d\right)\boldsymbol{S}\right]:\boldsymbol{m}_s\otimes\boldsymbol{n}_s$ et $\boldsymbol{F}_{n+1}^e=\Delta\boldsymbol{F}\,\boldsymbol{F}_n^e\left(\Delta\boldsymbol{F}^p(\Delta\gamma_s)\right)^{-1}$

Et, suivant le comportement considéré,

- $\Delta\gamma_s=\Delta p_s(\tau_s,\Delta\beta_s)\xi_s$ et $\xi_s=\frac{\tau_s}{|\tau_s|}$ *ou* $\frac{\tau_s-f(\alpha)}{|\tau_s-f(\alpha)|}$ correspond au signe de l'écoulement
- $\Delta\beta_s$ représente soit l'incrément de glissement plastique $\Delta\gamma_s$, pour les lois MONO_VISC*, soit la variation de densité de dislocations $\Delta\omega_s$ pour les lois MONO_DD_*

Remarque : l'extraction des inconnues du système à partir des variables internes (qui restent au nombre de 3 par système de glissement) se fait dans la routine LCAFYD. Inversement, après la résolution par NEWTON, le calcul des 3 variables internes par système de glissement en fonction de la variable principale utilisée dans la résolution se fait dans la routine LCPLNF.

La forme générale de l'algorithme résolu par Newton est

$$Y_{k+1}=Y_k-(\frac{dR}{dY_k})^{-1}R(Y_k)$$

Il faut donc définir les valeurs initiales ΔY_0 (0 par défaut, dans la routine LCMMIN) , et calculer le résidu Y_k , ainsi que la matrice jacobienne du système : $\frac{dR}{dY_k}$

3.3.1 Architecture générale de PLASTI :

```
CALL LCMATE => LCMMAT, identique à RUNGE_KUTTA
```

Prédiction élastique
```
   CALL LCELAS
```
Calcul du SEUIL
```
   CALL LCCNVX => routine LCMMVX évaluation du seuil pour MONOCRISTAL.
```

Calcul le solution élasto-visco-plastique par la méthode de Newton :
```
IF ( SEUIL .GE. 0.D0 ) THEN
   CALL LCPLAS / CALL LCPLNL
ENDIF
```

Calcul de l'opérateur tangent :
```
IF ( OPT .EQ. 'RIGI_MECA_TANG' .OR. OPT .EQ. 'FULL_MECA' ) THEN
   CALL LCJPLC => CALL LCMMJP
ENDIF
```

Remarque : L'opérateur tangent est calculé automatiquement en fonction de la matrice jacobienne du système d'équations local [R5.03.11]. Ceci est réalisé en petites et en grandes déformations dans la routine LCMMJP. Il n'y a donc a priori rien à modifier pour ce calcul lors de l'ajout d'un nouveau comportement cristallin.

La routine LCCNVX permet de détecter si le seuil est franchi pour au moins un système de glissement. Sa structure est la suivante :
```
SEUIL=0.D0
DO IFA=1,NBFSYS
   DO IS=1,NBSYS
      CALL LCMMFI
C     ECOULEMENT VISCOPLASTIQUE
      CALL LCMMFE => DP  calculé à partir de la prédiction élastique
      IF (DP.GT.0.D0) SEUIL=1.D0
   ENDDO
ENDDO
```

On utilise donc les mêmes routines LCMMFE et LCMMFI que pour l'intégration explicite et implicite.

La routine LCPLNL réalise **la boucle de Newton**. Sa structure est la suivante :

LCPLNL
- LCAFYD (extraction des variables internes utiles au système réduit)
- LCINIT => LCMMIN : initialisation de ΔY_0 , 0 par défaut
- LCRESI => LCMMRE : calcul du résidu

- Soit LCJACB => LCMMJA : calcul de la matrice jacobienne
- Soit (si ALGO_INTE=NEWTON_PERT) LCJACP calcul de la matrice jacobienne par perturbation, qui appelle LCRESI

- MGAUSS résolution
- LCRELI recherche linéaire (si ALGO_INTE=NEWTON_RELI), qui appelle LCRESI...
- LCCONV => LCMMCV critère de convergence
- LCPLNF => LCDPEC calcul des toutes les variables internes.

3.3.2 Calcul du résidu

La routine LCMMRE calcule le résidu. Sa structure est la suivante :

```
DO IFA=1,NBFSYS
   DO IS=1,NBSYS
      CALTAU calcul de  τ_s
      LCMMLC calcule des quantités relatives au système de glissement :
         CALL LCMMFI( => RP)        routine de calcul de l'écrouissage isotrope
         CALL LCMMFE( => DGAMMA, DP)      routine de calcul de l'écoulement
         CALL LCMMEC( => DALPHA )routine de calcul de l'écrouissage cinématique
      calcul de  k_s  et stockage dans  R_2(σ, Δβ)=Δβ_s−k_s(τ_s(σ), Δβ)

      En petites déformations :déformation viscoplastique globale
      DO ITENS=1,6
         DEVI(ITENS)=DEVI(ITENS)+MuS(ITENS)*DGAMMA
      ENDDO
      En grandes déformations, calcul des termes nécessaires à
 ΔF^p(Δγ_s)
   ENDDO
ENDDO
```

- en petites déformations, calcul de $\boldsymbol{R_1}(\boldsymbol{\sigma}, \Delta\beta)=\boldsymbol{\Lambda}^{-1}.\Delta\boldsymbol{\sigma}-\Delta\boldsymbol{\varepsilon}+\Delta\boldsymbol{\varepsilon}^{th}+\sum_s \Delta\gamma_s \boldsymbol{\mu_s}=0$
- en grandes déformations

CALCFE calcul de $\boldsymbol{F}^e_{n+1}=\Delta\boldsymbol{F}\,\boldsymbol{F}^e_n\left(\Delta\boldsymbol{F}^p(\Delta\gamma_s)\right)^{-1}$

LCGRLA calcul de $\boldsymbol{F}^e_{n+1}=\Delta\boldsymbol{F}\,\boldsymbol{F}^e_n\left(\Delta\boldsymbol{F}^p(\Delta\gamma_s)\right)^{-1}$ $\boldsymbol{E}^e_{GL}=\frac{1}{2}(\boldsymbol{F}^{eT}\boldsymbol{F}^e-\boldsymbol{I}_d)$

$\boldsymbol{S}=\boldsymbol{\Lambda}:\boldsymbol{E}^e_{GL}$. et $\boldsymbol{R_1}(\boldsymbol{S}, \Delta\beta)=\boldsymbol{\Lambda}^{-1}.\boldsymbol{S}-\frac{1}{2}\left(\boldsymbol{F}^{eT}\boldsymbol{F}^e-\boldsymbol{I}_d\right)$

On constate donc que l'on utilise là encore les même routines que pour l'intégration explicite : LCMMFI, LCMMFE, LCMMEC sont a priori les seules routines à modifier lors de l'ajout d'un comportement, que ce soit en petites ou en grandes déformations, pour le calcul du résidu, lors de l'intégration implicite.

3.3.3 Calcul de la matrice jacobienne

La routine LCMMJA calcule la matrice jacobienne. Sa structure est la suivante :

```
DO IFA=1,NBFSYS
   DO IS=1,NBSYS
      LCMMJB : calcul des termes dérivés
         LCMMJ2 : calcul des termes dérivés pour MONO_DD_KR
         LCMMJD : calcul des termes dérivés pour MONO_DD_CFC, MONO_DD_CC
         LCMMJ1 : calcul des termes dérivés pour MONO_VISC1, MONO_VISC2
   ENDDO
ENDDO
```

Les dérivées de ces équations pour le calcul de la matrice jacobienne peuvent être écrites de façon générale :

HPP		GDEF	
$J_{11}=\frac{\partial \boldsymbol{R_1}(\boldsymbol{\sigma},\Delta\beta)_i}{\partial \boldsymbol{\sigma_j}}$	$J_{12}=\frac{\partial \boldsymbol{R_1}(\sigma,\Delta\beta)_i}{\partial \beta_s}$	$J_{11}=\frac{\partial \boldsymbol{R_1}(\boldsymbol{S},\Delta\beta)_i}{\partial \boldsymbol{S_j}}$	$J_{12}=\frac{\partial \boldsymbol{R_1}(\boldsymbol{S},\Delta\beta)_i}{\partial \beta_s}$
$J_{21}=\frac{\partial \boldsymbol{R_2}(\boldsymbol{\sigma},\Delta\beta)_s}{\partial \boldsymbol{\sigma_i}}$	$J_{22}=\frac{\partial \boldsymbol{R_2}(\boldsymbol{\sigma},\Delta\beta)_s}{\partial \beta_r}$	$J_{21}=\frac{\partial \boldsymbol{R_2}(\boldsymbol{S},\Delta\beta)_s}{\partial \boldsymbol{S_i}}$	$J_{22}=\frac{\partial \boldsymbol{R_2}(\boldsymbol{S},\Delta\beta)_s}{\partial \beta_r}$

Tableau 3.3.3-1

Les termes intervenant dans chaque sous-matrice ont en commun des termes spécifiques à chaque comportement, qui sont calculés dans les routines LCMMJ* [R5,03,11 annexe 5] :

- $\frac{\partial \Delta\gamma_s}{\partial \tau_s}=\frac{\partial \Delta p_s}{\partial \tau_s}\xi_s$,
- $\frac{\partial \Delta\gamma_r}{\partial \Delta\beta_s}=\frac{\partial \Delta p_r}{\partial \Delta\beta_s}\xi_r$,
- $\frac{\partial k_s}{\partial \tau_s}$
- $\frac{\partial k_r}{\partial \Delta\beta_s}$

Dans le cas d'un nouveau comportement, il faut donc soit ajouter le calcul de ces termes dérivés dans une routine LCMMJ* existante, soit en ajouter une nouvelle.

Remarque : pour un premier test, on peut se passer du calcul de la matrice jacobienne, en utilisant la construction automatique de la matrice jacobienne (par perturbation, `ALGO_INTE='NEWTON_PERT'`). De ce fait, il est très rapide d'introduire un nouveau comportement cristallin dans l'environnement `PLASTI`: il suffit de calculer le résidu dans la routine `LCMMRE`, appelée par `LCRESI`. Par contre, le temps calcul ne sera optimisé qu'avec une matrice jacobienne programmée.

3.3.4 Critère de convergence et post-traitements

Le critère de convergence est générique a priori, et ne dépend pas du comportement, mais peut être éventuellement modifié :

- `LCCONV => LCMMCV` critère de convergence

La dernière routine à modifier (éventuellement, si on veut calculer et ajouter aux variables internes en sortie des valeurs utiles au post-traitement) est :

- `LCPLNF => LCDPEC` qui recalcule toutes les variables internes à partir de la solution du système réduit. Elle fait appel encore une fois à la routine de comportement `LCMMLC`.