

---

## Architecture des comportements cristallins

---

### Résumé :

Ce document décrit la structure de l'intégration des comportements cristallins (confer 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`.

## Table des Matières

1 Description générale des comportements cristallins.....	3
2 Architecture de DEFI_COMPOR.....	3
2.1 Illustration sur un exemple : test SSNV194.....	3
2.1.1 Modélisation A : un petit agrégat comportant 10 grains :.....	3
2.1.2 Modélisations B et C : polycristal comportant 10 grains :.....	4
2.2 Description de DEFI_COMPOR.....	5
2.2.1 Structure :.....	5
2.2.2 Ajout d'un comportement cristallin au catalogue de DEFI_MATERIAU / DEFI_COMPOR....	5
3 Architecture de l'intégration.....	7
3.1 Catalogues.....	7
3.2 Routines d'intégration des lois.....	8
3.3 Récupération des coefficients matériau.....	8
3.4 Intégration explicite – schéma de RUNGE - KUTTA.....	10
3.4.1 Cas du monocristal :.....	10
3.4.2 Cas du polycristal :.....	12
3.5 Intégration implicite du monocristal par Newton dans PLASTI.....	14
3.5.1 Architecture générale de PLASTI :.....	15
3.5.2 Calcul du résidu.....	16
3.5.3 Calcul de la matrice jacobienne.....	17
3.5.4 Critère de convergence et post-traitements.....	18
4 Objets internes pour le stockage des informations.....	18
4.1 Dans le cas du monocristal :.....	18
4.2 Dans le cas du polycristal :.....	19

## 1 Description générale des comportements cristallins

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

Le nom du comportement dans `COMPORTEMENT` 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 (environnement `PLASTI` cf. [R5.03.14])
- les objets internes utiles à la résolution.

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,
                    COMPORTEMENT=_F(RELATION='MONOCRISTAL',
                                     COMPOR=MONO1,
                                     ),)
```

## 2.1.2 Modélisations B et 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=COMPOR,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., MU_LOCA=145200./2.6, );
```

### Modélisation B: (maillage comportant un seul élément)

...

```
SOLNL=STAT_NON_LINE(MODELE=TROISD,  
CHAM_MATER=MAT2,  
EXCIT=( _F(CHARGE=TRAC,  
FONC_MULT=COEF,  
TYPE_CHARGE='FIXE_CSTE',),),  
INCREMENT=( _F(LIST_INST=LINST,),),  
COMPORTEMENT=( _F(RELATION='POLYCRISTAL',  
COMPOR=COMPORP2,  
DEFORMATION='PETIT',  
ALGO_INTE='RUNGE_KUTTA',  
TOUT='OUI',  
RESI_INTE_RELA=1.E-6  
),),  
NEWTON=( _F(PREDICTION='EXTRAPOLE',  
MATRICE='ELASTIQUE',  
REAC_ITER=0,),),  
CONVERGENCE=( _F(ITER_GLOB_MAXI=50,  
RESI_GLOB_RELA=1.E-4  
),),),);
```

### Modélisation C: (point matériel)

```
SOLNL=SIMU_POINT_MAT(COMPORTEMENT=_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'écoulement isotrope, à l'écoulement 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: catalogue de DEFI\_MATERIAU

```
MONO_DD_CFC =FACT(statut='f',
regles=( UN_PARMIS('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_G LIS=SIMP(statut='f',typ='TXM',
    into=('OCTAEDRIQUE','BCC24','CUBIQUE1','CUBIQUE2',
        'ZIRCONIUM','UNIAXIAL','UTILISATEUR')),
b_util =BLOC(condition="FAMI_SYST_G LIS=='UTILISATEUR' ",
    TABL_SYST_G LIS =SIMP(statut='f', typ=table_sdaster,,),),

# cas d'un comportement de type DD
b_dd_cc=BLOC(condition="ECOULEMENT=='MONO_DD_CC'",
    FAMI_SYST_G LIS=SIMP(statut='f',typ='TXM',into=('CUBIQUE1',
    UTILISATEUR',),
    b_util=BLOC(condition="FAMI_SYST_G LIS=='UTILISATEUR'",
    TABL_SYST_G LIS=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'),
    default='NON',fr="rotation de reseau : NON, POST, CALC"),

POLYCRISTAL =FACT(statut='f', max='**',
    regles=(UN_PAMI('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"),

b_poly =BLOC( condition = "POLYCRISTAL!=None",
    MU_LOCA =SIMP(statut='o',typ='R',max=1),
    LOCALISATION=SIMP(statut='f', typ='TXM', max=1, into=('BZ', 'BETA',),
    fr=tr("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 `LCMMMSG`, 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 `COMPORTEMENT`.
- 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 `COMPORTEMENT`.

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.

On peut imprimer la structure de données produite à l'aide de IMPR\_CO.

## 3 Architecture de l'intégration

### 3.1 Catalogues

Les lois de comportement cristallines sont utilisables dans C\_COMPORTEMENT.capy via RELATION = 'MONOCRISTAL' ou RELATION='POLYCRISTAL'. Les données relatives aux lois cristallines spécifiques sont quant à elle définies dans sd\_compor, issue de DEFI\_COMPOR et fournie sous le mot-clé COMPOR dans c\_comportement.capy.

#### catapy.commun/c\_comportement.capy

```
b_monox      = BLOC(condition = "RELATION == 'MONOCRISTAL' ",
                    fr=tr("SD issue de DEFI_COMPOR"),
                    COMPOR =SIMP(statut='o',typ=compor_sdaster,max=1)),

b_polyx      = BLOC(condition = "RELATION == 'POLYCRISTAL' ",
                    fr=tr("SD issue de DEFI_COMPOR"),
                    COMPOR =SIMP(statut='o',typ=compor_sdaster,max=1)),
```

Les catalogues des lois de comportement sont :

#### bibpyt/Comportement/monocristal.py

```
from cata_comportement import LoiComportement
loi = LoiComportement(
    nom          = 'MONOCRISTAL',
    doc = """Ce modèle permet de décrire le comportement d'un monocristal dont
les relations de comportement sont fournies via le concept compor, issu de
DEFI_COMPOR. Le nombre de variables internes est fonction des choix
effectués dans DEFI_COMPOR ; pour plus de précisions voir [R5.03.11].""",
    num_lc      = 32,
    nb_vari     = 0,
    nom_vari    = None,
    mc_mater    = None,
    modelisation = ('3D', 'AXIS', 'D_PLAN'),
    deformation = ('PETIT', 'PETIT_REAC', 'SIMO_MIEHE'),
    nom_varc    = ('TEMP'),
    algo_inte   = ('NEWTON', 'NEWTON_RELI', 'RUNGE_KUTTA', 'NEWTON_PERT'),
    type_matr_tang = ('PERTURBATION', 'VERIFICATION'),
    proprietes  = None, )
```

#### bibpyt/Comportement/polycristal.py

```
from cata_comportement import LoiComportement
loi = LoiComportement(
    nom          = 'POLYCRISTAL',
    doc = """Comportement poly-cristallin homogénéisé, défini par DEFI_COMPOR""",
    num_lc      = 37,
    nb_vari     = 0,
    nom_vari    = None,
    mc_mater    = None,
    modelisation = ('3D', 'AXIS', 'D_PLAN'),
    deformation = ('PETIT', 'PETIT_REAC', 'GROT_GDEP'),
    nom_varc    = ('TEMP'),
```

```
algo_inte          = ('RUNGE_KUTTA'),  
type_matr_tang    = ('PERTURBATION', 'VERIFICATION'),  
proprietes        = None, )
```

## 3.2 Routines d'intégration des lois

L'intégration des lois de comportement cristallines se fait comme pour toutes les lois de comportement au niveau de chaque point d'intégration des éléments finis pour les options non linéaires (FULL\_MECA, RAPH\_MECA, RIGI\_MECA\_TANG).

La routine LC0032 (MONOCRISTAL) est appelée pour chaque point l'intégration par les routines suivantes :

- si DEFORMATION='PETIT' ou ='PETIT\_REAC' :
  - en 3D : TE0139/NMPL3D/NMCOMP/REDECE/LC0000/ LC0032
  - en 2D : TE0100/NMPL2D/NMCOMP/REDECE/LC0000/ LC0032
- si DEFORMATION='SIMO\_MIEHE' :
  - en 2D ou en 3D, NMPL2D ou NMPL3D sont remplacées par NMGPMI

La routine LC0032 fait appel suivant l'algorithme choisi, soit à PLASTI (intégration implicite par la méthode de Newton ou une variante) soit à NMVPRK (intégration par Runge-Kutta).

La routine LC0037 (POLYCRISTAL) est appelée pour chaque point l'intégration par les routines suivantes :

- si DEFORMATION='PETIT' ou ='PETIT\_REAC' :
  - en 3D : TE0139/NMPL3D/NMCOMP/REDECE/LC0000/ LC0032
  - en 2D : TE0100/NMPL2D/NMCOMP/REDECE/LC0000/ LC0032

La routine LC0037 fait appel à NMVPRK (intégration par Runge-Kutta).

## 3.3 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 LCMMAF, 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\_compom (issue de DEFI\_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_compom issue de DEFI_COMPOR  
pour chaque famille de systèmes :  
  LCMMMSG 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,  
définis dans le repère global, 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 chaînes 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_CFC	5
MONO_DD_CFC_IRRA	8
MONO_DD_FAT	6
MONO_DD_CC	7
MONO_DD_CC_IRRA	7

Tableau 3.3-1

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_CFC_IRRA	8
MONO_DD_FAT	4
MONO_DD_CC	7
MONO_DD_CC_IRRA	7

Tableau 3.3-2

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 3.3-3

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

LCMMMSG fournit le nombre de systèmes de glissement

LCMAFL récupère les coefficients matériau relatifs à l'écoulement

LCMAEC coefficients matériau relatifs à l'érouissage cinématique,

LCMAEI coefficients matériau relatifs à l'érouissage isotrope, et 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 (`nbcomm`, `coef/materf`, `cpmono`, décrits en fin de ce document).

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, définis en repère global, pour optimiser les performances.

## 3.4 Intégration explicite – schéma de RUNGE - KUTTA

### 3.4.1 Cas du monocristal :

C'est la façon la plus rapide (mais moins optimale que l'intégration implicite, pour un système de quelques dizaines d'équations) 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  $6+3 \times n_s$  équations différentielles, du type (cf. [R5.03.11]) :  $\frac{dY}{dt} = F(Y, t)$ , où  $Y$

représente l'ensemble des variables internes :  $Y = \begin{pmatrix} \alpha_s \\ \gamma_s \\ p_s \\ E^{vp} \end{pmatrix}$

Le système d'équations différentielles à résoudre est :

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

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

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 =  $\sigma$

Puis elle calcule les  $6+3 \times n_s$  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 LCMMMSG(NOMFAM,NBSYS,0,PGL,MS,NG,LG,0,Q)
```

```
Boucle sur les systèmes de glissement de la famille IFA :
```

```
DO IS=1,NBSYS
```

```
CALL LCMMMSG(..) relecture du tenseur d'orientation MuS
```

```
CALCUL DE LA CISSION REDUITE
```

```
TAUS= SIG(I)*MuS(I) pour i=1,6  $\tau_s = \mu_s : \sigma$ 
```

```
CALL LCMMFI ( => RP) routine de calcul de l'érouissage isotrope  $R_s(p)$ 
```

```
CALL LCMMFE (=> DGAMMA, DP) routine de calcul de l'écoulement  $\dot{\gamma}_s, \dot{p}_s$ 
```

```
CALL LCMMEC ( => DALPHA) routine de calcul de l'érouissage cinématique  $\dot{\alpha}_s$ 
```

```
Calcul de la déformation viscoplastique globale
```

```
DO ITENS=1,6
```

```
DEVI (ITENS) =DEVI (ITENS) +MuS (ITENS) *DGAMMA  $\epsilon^{vp} = \sum_s \mu_s \dot{\gamma}_s$ 
```

```
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\_REL) [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  
C     IF (NUEISO.EQ.1) THEN  
C       .....  
C       RP=...  
C   ELSEIF (NECRIS.EQ.'MONO_ISOT2') THEN  
C     ELSEIF (NUEISO.EQ.2) THEN  
C       .....  
C       RP=...  
C   ELSEIF (NECRIS.EQ.'MONO_DD_CFC') THEN  
C     ELSEIF (NUEISO.EQ.3) THEN  
C       .....  
C       RP=MU*SQRT(RP)*CEFF  
C     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 (NECRIC.EQ.'MONO_CINE1') THEN  
C     IF (NUECIN.EQ.1) THEN  
C       DALPHA=...  
C   ELSEIF (NECRIC.EQ.'MONO_CINE2') THEN  
C     ELSEIF (NUECIN.EQ.2) THEN  
C     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  
C     IF (NUECOU.EQ.1) THEN  
C       DP=...  
C       DGAMMA= ...  
C     ELSEIF (NUECOU.EQ.2) THEN  
C     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.

## 3.4.2 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  $6 + n_g(6 + 3 \times n_s(g))$  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 :

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

- pour chacun des  $n_s(g)$  systèmes de glissement de chaque grain  $g$ , 3 relations :
  - $\dot{\gamma}_s = \dot{p}_s(\tau_s(\sigma), \alpha_s, \gamma_s, R_s(p)) \eta(\tau_s, \alpha_s)$  où  $\eta(\tau_s, \alpha_s) = \pm 1$
  - $\dot{\alpha}_s = h(\tau_s, \alpha_s, \gamma_s, p_s)$
  - $R_s(p)$
- à l'échelle du grain, l'calcul de la déformation plastique :  $\varepsilon_g^{ip} = \sum_s \mu_s \dot{\gamma}_s$
- calcul de la déformation plastique macroscopique :  $E^{ip} = \sum_g f_g \varepsilon_g^{ip}$

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` ( $\Sigma$ )

Puis elle calcule les  $6 + n_g(6 + 3 \times n_s(g))$  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 ( $\sigma_g$ )
```

- Boucle sur les familles de systèmes de glissement :

```
DO IFA=1, NBFSYS
  ...
  Récupération du nombre de systèmes de glissement
  CALL LCMMMSG(...)
```

Boucle sur les systèmes de glissement de la famille `IFA` :

```
DO IS=1, NBSYS
```

```
CALL LCMMMSG(..) relecture du tenseur d'orientation MuS(IGRAIN, IS)
```

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

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

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

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

Calcul de la déformation viscoplastique globale

```
DO ITENS=1, 6
```

```
c      DEVG(ITENS)=DEVG(ITENS)+MuS(ITENS)*DGAMMA (DEVG= $\varepsilon_g^{vp}$ )
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. Enfin, pour certains post-traitements spécifiques au niveau du point d'intégration, il faut intervenir dans la routine `LCDPEQ`.

## 3.5 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'équations à résoudre sous forme purement implicite, de la façon suivante :  $R(Y) = 0$

$$R(Y) = \begin{pmatrix} \Lambda^{-1} \Sigma - (\Lambda^{-1}) \Sigma - (\Delta E - \Delta E^{th} - \Delta E^{vp}) \\ \Delta E^{vp} - \sum_s \mu_s \Delta \gamma_s \\ n_s \begin{pmatrix} \Delta \alpha_s - h(\tau_s^+, \alpha_s^+, \gamma_s^+, p_s^+) \\ \Delta \gamma_s - g(\tau_s^+, \alpha_s^+, \gamma_s^+, p_s^+) \\ \Delta p_s - f(\tau_s^+, \alpha_s^+, \gamma_s^+, p_s^+) \end{pmatrix} \end{pmatrix} = 0$$

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). Mais ce n'est pas le système effectivement résolu : 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,  $\Delta E^{vp}$  peut être exprimée en fonction de  $\sum_s \mu_s \Delta \gamma_s$  donc 6 équations peuvent être éliminées du système global à résoudre.
- Comme  $\Delta p_s = |\Delta \gamma_s|$ , l'équation en  $\Delta p_s$  peut être éliminée
- 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'érouissage 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*`. Il y a donc une seule inconnue par système de glissement :  $\Delta \beta_s$ , qui représente soit  $\Delta \gamma_s$ , soit  $\Delta \alpha_s$

On obtient donc le système suivant :

- **Petites déformations:**

$$\begin{aligned} \mathbf{R}_1(\boldsymbol{\sigma}, \Delta \beta) &= \boldsymbol{\Lambda}^{-1} \cdot \Delta \boldsymbol{\sigma} - \Delta \boldsymbol{\varepsilon} + \Delta \boldsymbol{\varepsilon}^{th} + \sum_s \Delta \gamma_s \boldsymbol{\mu}_s = 0 \\ \mathbf{R}_2(\boldsymbol{\sigma}, \Delta \beta) &= \Delta \beta_s - k_s(\tau_s(\boldsymbol{\sigma}), \Delta \beta) = 0 \end{aligned} \quad \text{où l'inconnue est : } \mathbf{Y} = \begin{bmatrix} \boldsymbol{\sigma} \\ \Delta \beta \end{bmatrix}$$

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

- **Grandes déformations**

$$\begin{aligned} \mathbf{R}_1(\mathbf{S}, \Delta \beta) &= \boldsymbol{\Lambda}^{-1} \cdot \mathbf{S} - \frac{1}{2} (\mathbf{F}^{eT} \mathbf{F}^e - \mathbf{I}_d) = 0 \\ \mathbf{R}_2(\mathbf{S}, \Delta \beta) &= \Delta \beta_s - k_s(\tau_s(\mathbf{S}), \Delta \beta) = 0 \end{aligned} \quad \text{où l'inconnue est : } \mathbf{Y} = \begin{bmatrix} \mathbf{S} \\ \Delta \beta \end{bmatrix}$$

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

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

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

Il faut donc définir les valeurs initiales  $\Delta Y_0$  (0 par défaut, dans la routine LCMMIN), et calculer le résidu  $Y_k$ , ainsi que la matrice jacobienne du système :  $\frac{dR}{dY_k}$

### 3.5.1 Architecture générale de PLASTI :

L'algorithme de Newton utilisé dans PLASTI est décrit en [R5.03.14].

Lecture des coefficients matériau et stockage dans les objets NBCOMM, MATERF/MATERD, CPMONO

CALL LCMATE => LCMMAT, identique à RUNGE\_KUTTA

Prédiction élastique

CALL LCELAS

Calcul du SEUIL

CALL LCCNVX => routine LCMMVX évaluation du seuil pour MONOCRISTAL.

Calcul de la solution élasto-visco-plastique par la méthode de Newton :

```
IF ( SEUIL .GE. 0.DO ) 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. Elle appelle dans le cas du monocristal la routine LCMNVX . Sa structure est la suivante :

```
SEUIL=0.DO
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.DO) SEUIL=1.DO
  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 (générique à l'ensemble des lois de comportement sous PLASTI) est la suivante :

```
LCPLNL
  • LCAFYD (extraction des variables internes utiles au système réduit)
  • LCINIT => LCMMIN : initialisation de  $\Delta Y_0$  , 0 par défaut
  • LCRESI => LCMRE : 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 => LCMCV critère de convergence
  • LCPLNF => LCDPEC calcul des toutes les variables internes.
```

## 3.5.2 Calcul du résidu

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

```
DO IFA=1,NBFSYS
  DO IS=1,NBSYS
    CALTAU calcul de  $\tau_s$  (petites ou grandes déformations)
    LCMMLC calcule des quantités relatives au système de glissement :
      CALL LCMMFI( => RP) routine de calcul de l'érouissage isotrope
      CALL LCMMFE( => DGAMMA, DP) routine de calcul de l'écoulement
      CALL LCMMEC( => DALPHA ) routine de calcul de l'érouissage cinématique
    calcul de  $k_s$  et stockage dans  $R_2(\sigma, \Delta \beta) = \Delta \beta_s - k_s(\tau_s(\sigma), \Delta \beta)$ 
```



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  $\Delta F^p(\Delta \gamma_s)$

ENDDO

ENDDO

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

- en grandes déformations

CALCFE calcul de  $F_{n+1}^e = \Delta F F_n^e (\Delta F^p(\Delta \gamma_s))^{-1}$

LCGRLA calcul de  $E_{GL}^e = \frac{1}{2} (F^{eT} F^e - I_d)$   $S = \Lambda : E_{GL}^e$

et  $R_1(S, \Delta \beta) = \Lambda^{-1} \cdot S - \frac{1}{2} (F^{eT} F^e - I_d)$

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.

### 3.5.3 Calcul de la matrice jacobienne

La routine LCMMJA calcule la matrice jacobienne (sauf si ALGO\_INTE= 'NEWTON\_PERT').

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 (plus\_IRRA)

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

Tableau 3.5.3-1

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.

### 3.5.4 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 LCMLLC.

## 4 Objets internes pour le stockage des informations

La lecture des coefficients matériau est effectuée une seule fois par point d'intégration dans les routines LCMMAT / LCMMAP. Les quantités lues (ou calculées, dans la cas des tenseurs d'orientation) doivent être transmises aux routines d'intégration. Pour cela on utilise des tableaux spécifiques : nbcomm, coeft/materf, cpmono, toutms...)

### 4.1 Dans le cas du monocristal :

Stockage des informations relatives aux monocristal dans des tableaux spécifiques (nbcomm, coeft/materf, materd, cpmono,).

MATERF : coefficients matériau a t+dt  
mater(\*,1) = caractéristiques élastiques  
mater(\*,2) = caractéristiques plastiques

MATERD : coefficients matériau à t

NBCOMM(\*,3) : POSITION DES COEF POUR CHAQUE SYSTEME  
tableau d'entiers

	Colonne 1	Colonne 2	Colonne3
--	-----------	-----------	----------

Ligne 1	1	Nb var.int.	Nb monocristaux
---------	---	-------------	-----------------

pour chaque famille de systèmes de glissement :



```
pour chaque phase
pour la localisation indice coef nb param 0
phase g nb fam g 0 NVIg
... et pour chaque famille de systèmes de glissement :
famille 1 ind coef ind coef ind coef
          ecoulement ecr iso ecr cin
.....
(ind signifie l'indice dans COEFT(*))
```

TOUTMS : TOUS LES TENSEURS D'ORIENTATION POUR TOUS LES  
SYSTEMES DE GLISSEMENT DE TOUS LES MONOCRISTAUX.