MFRON03 – Test of the Code_Aster-MFront *interface* for models with orthotropy

Summarized:

This test validates definite behaviors orthotropic using *MFront* by comparison with behavior similar of *Code_Aster* .

Modelization a: this modelization makes it possible to validate the model elastic orthotropic by comparison with test SSNV225C on a material point.

Modelization b: this modelization makes it possible to the model validate single-crystal élasto-visco-plastic with implicit integration, by comparison with the model MONOCRISTAL of the test zmat007a on an aggregate with 10 grains.

Modelization C: this modelization makes it possible to the model validate single-crystal élasto-visco-plastic with implicit integration, and complete definition of the family of sliding systems and the matrix of interaction, by comparison with the model MONOCRISTAL of the test zmat007a on an aggregate with 10 grains.

1 Problem of reference

1.1 Geometry

the geometry of the modelization A is identical to that of test SSLV113A.

The geometry of the modelization B is identical to that of test ZMAT007A (aggregate with 10 grains).

1.2 Properties of the materials

the coefficients of the Mfront behavior are, for modelization a:

C1	11000	E_L
C2	5000	E_T
C5	8000	E_N
C6	0,396	NU_LT
C7	0,11	NU_TN
C8	0,15	NU_LN
C9	10500	G_LT
C10	13000	G_TN
C11	7000	G_LN

the Mfront file defining the orthotropic elastic behavior is:

```
@Parser    Implicit;

@Behavior elasorth;

@Algorithm NewtonRaphson_NumericalJacobian;


@OrthotropicBehaviour;
@RequireStiffnessTensor;

@Theta 1. ;
@AuxiliaryStateVariable real seq;

@TangentOperator {
  Dt = D;
}

@ComputeStress {
  sig = D*eel;
}
@Integrator {
  feel += - deto;
}

@UpdateAuxiliaryStateVars {
  seq = sqrt (sig|sig);
}
```

The Mfront file defining the single-crystal behavior of CFC type with isotropic elasticity (modelization B) is:

```
@Parser    Implicit;
@Behavior monocrystal;
@Algorithm NewtonRaphson_NumericalJacobian;
@OrthotropicBehaviour;
@IsotropicElasticBehaviour;
@RequireStiffnessTensor;
@Theta 1. ;
@Epsilon 1.e-8;
@IterMax 100;
@MaterialProperty stress Young;
@MaterialProperty real nu;
@MaterialProperty real m;
@MaterialProperty real K;
@MaterialProperty real C;
@MaterialProperty real R0;
@MaterialProperty real Q;
@MaterialProperty real B;
@MaterialProperty real d1;
@Includes {
#include " TFEL/Material/Lame.hxx” #include
“TFEL/Material/MetallicCFCSlidingSystems.hxx” #include
“TFEL/Material/MetallicCFCGenericSlidingSystemsInteractionMatrix.hxx”} @Link {
“_
lTFELMaterialSingleCristals”}; @StateVariable
strain G [12]; @    AuxiliaryStateVariable
real p [12]; @AuxiliaryStateVariable
real has [12]; @LocalVariable
average real;    @LocalVariable
real driven; @Parameter
h1, H2, h3, h4, h5, h6; h1.setDefaultValue
(1.); h2.setDefaultValue
(0.); h3.setDefaultValue
(0.); h4.setDefaultValue
(0.); h5.setDefaultValue
(0.); h6.setDefaultValue
(0.); /* Initialize
Blade coefficients * @InitLocalVars
{using namespace
  tfel:: material:: blade; lambda =
  computeLambda (Young, nu); driven = computeMu
  (Young, nu); } @TangentOperator

{using namespace
  tfel:: material:: blade; StiffnessTensor
  Hooke; Stensor4
  I; computeElasticStiffness
  <N, Type>:: exe (Hooke, lambda, driven); getPartialJacobianInvert
  (I); Dt = Hooke
  *Je; } @ComputeStress

{sig = (average
    *trace (eel) *Stensor:: Id () +2*mu*eel); } @Integrator

{typedef
  MetallicCFCSlidingSystems<N> CFCSlidingSystems; typedef
```

```
     MetallicCFCGenericSlidingSystemsInteractionMatrix  InteractionMatrix;  const
CFCSlidingSystems
  & ss = CFCSlidingSystems:: getMetallicCFCSlidingSystems (); const tmatrix
<12,12, double>& H = InteractionMatrix
 :: getInteractionMatrix (h1, H2, h3, h4, h5, h6); StrainStensor
  vepsp (real (0)); real tau
  [12]; real vp [
  12]; real goes [
  12]; real Ag [
  12]; real tma
  [12]; real tmR
  [12]; real RP [
  12]; real EP [
  12]; for (unsigned
  shorts i=0; I! =12; ++i) {Ag [I] =
     ab (dg  [I]); EP [I] =
     Q* (1. - exp (- b* (p [I] +ag [I]))) ; } for (unsigned

  shorts i=0; I! =12; ++i) {const stensor
     <N>& driven = ss.mus [I]; RP [I] = R0
     ; for (unsigned
     shorts j=0; J! =12; ++j) {RP [I] +=h (I
        , J) *pe [I]; } tau [I] =
     driven
     | sig; goes [I] = (dg
     [I] - d1  *a [I] *ag [I])/(1.+d1*ag [I]); tma [I] = tau
     [I] - C* (goes [I] +a [I]); ; tmR [I] = ab
     (tma [I]) - RP [I]; yew (tmR [I] >
     0.) {real sgn=tma
         [I] /abs (tma [I]); vp [I] = dt
         *sgn*pow  ((tmR [I] /K), m); } else {vp [
         I]
     =0.; }
         vepsp+=vp
     [I
     ] *mus; } feel += vepsp

  - deto; for (unsigned
  shorts i=0; I! =12; ++i) {fg [I] - = vp
    [I]; }  } @UpdateAuxiliaryStateVars


{for (unsigned
  shorts i=0; I! =12; ++i) {p [I] +=abs (dg
     [I]); [I] += (dg [I]
     has - d1*a [I] *p [I])/(1.+d1*p [I]); }} the Mfront
```

files defining the single-crystal behavior with elasticity orthotropic (modelization C) and definition of the family of sliding systems and the matrix of interaction are: @Parser Implicit

```
; @    Behavior
monocrystal; @Algorithm
NewtonRaphson_NumericalJacobian; @OrthotropicBehaviour
; @RequireStiffnessTensor
; @Theta 1. ;
@Epsilon 1.
e-8; @IterMax 100
; @MaterialProperty
real ind; @MaterialProperty
```

```
real m; @MaterialProperty
real K; @MaterialProperty
real C; @MaterialProperty
real R0; @MaterialProperty
real Q; @MaterialProperty
real B; @MaterialProperty
real H; @MaterialProperty
real d1; @StateVariable
strain G [12]; @AuxiliaryStateVariable
real p [12]; @AuxiliaryStateVariable
real has [12]; @Includes {
#include " TFEL
/FSAlgorithm/copy.hxx”} @TangentOperator

{Stensor4 I; getPartialJacobianInvert

  (I); Dt = D*Je; } @
  Importation “CFCGlidingSystems.mfront

”; @Import “CFCInteractionMatrix.mfront
”; @ComputeStress {sig
= D*eel; } @Integrator
  {StrainStensor

vepsp (
  real (0)); real tau [12]; real vp
  [12]; real goes
  [12]; real Ag
  [12]; real tma
  [12]; real tmR
  [12]; real RP
  [12]; real EP
  [12]; //sliding systems

  for (unsigned shorts I
  =0; I! =12; ++i) {Ag [I] = ab (dg [I]);
    EP [I]  = Q* (1. - exp (- B
    * (p [I] +ag [I]))) ; } for (unsigned shorts
  I
  =0; I! =12; ++i) {RP [I] = R0; for (unsigned
    shorts J
     =0; J! =12; ++j) {RP [I] +=mh (I, J) *pe [I
        ]; } tau [I] = driven [I] |
     sig
    ; goes [I] = (dg [I] - d1*a
    [I] *ag [  I])/(1.+d1*ag [I]); tma [I] = tau [I] - C* (goes
    [I] +a [I]); tmR [I] = ab (tma [I])
    - RP [I]; yew (tmR [I] >0.) {real
    sgn=tma [I] /abs (
      tma [I]); vp [I] = dt*sgn*pow (
      (tmR [I]  /K), m); } else {vp [I] =0.; }
    vepsp
    +=vp
      [I] *mus [I
    ]
    ; } feel += vepsp-deto
  ;
  for (unsigned shorts I
  =0; I! =12; ++i) {fg [I] - = vp [I]; }
```

```
    } @UpdateAuxiliaryStateVars


{for (unsigned shorts I
  =0; I! =12; ++i) {p [I] +=abs (dg [I]); [I
    ] += (dg [I] has - d1*a [I]
    *p [I])/(1.+d1*p [I]); }} the family of sliding systems
```

and the matrix of interaction are defined by @LocalVariable tfel:

```
: maths:: tvector<12, StrainStensor> driven; @InitLocalVariables {
const real coefm=1.0
  /sqrt (2.); const real coefn=1.0
  /sqrt (3.); const real nx [12] = {
  1.0,1.0, 1.0, 1.0,1.0,1.0, - 1.0, - 1.0, - 1.0, - 1.0, - 1.0, - 1.0}; const real
ny [12] = {
  1.0,1.0, 1.0, - 1.0, - 1.0, - 1.0, 1.0,1.0,1.0, - 1.0, - 1.0, - 1.0}; const real
nz [12] = {
  1.0,1.0, 1.0, 1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0}; const real MX [12] = {-
   1.0,0.0, - 1.0, - 1.0, 0.0, 1.0, 0.0,1.0, 1.0, - 1.0, 1.0, 0.0}; const real my
[12] = {
  0.0, - 1.0, 1.0,0.0,1.0, 1.0, - 1.0, 1.0, 0.0, 1.0,0.0, 1.0}; const real mz [12]
= {
  1.0,1.0,0.0,1.0,1.0, 0.0, 1.0,0.0, 1.0, 0.0,1.0, 1.0}; for (unsigned shorts I
  =0; I! =12; ++i) {tvector<3, real> NS (real
    (0)); tvector<3, real> ms (real
    (0)); stensor<3, real> driven;
    NS [0] =nx [I] *coefn; NS
    [1] =ny [I] *coefn; NS
    [2] =nz [I] *coefn; ms
    [0] =mx [I] *coefm; ms
    [1] =my [I] *coefm; ms
    [2] =mz [I] *coefm; driven
    [0] =ns [0] *ms [0]; driven
    [1] =ns [1] *ms [1]; driven
    [2] =ns [2] *ms [2]; driven
    [3] = (NS [0] *ms [1] +ns
    [1] *ms [0]) *0.5*sqrt (2); driven [4] = (NS [0] *ms [2] +ns
    [2] *ms [0]) *0.5*sqrt (2); driven [5] = (NS [1] *ms [2] +ns
    [2] *ms [1]) *0.5*sqrt (2); tfel:: fsalgo:: copy<StensorSize
    >:: exe (mu.begin (), driven [I] .begin ()); }} //
                ! interaction matrix


@LocalVariable tfel:: maths
:: tmatrix<12,12, real> mh; @InitLocalVariables {const
real h1 = 1. ; const
  real H2 = 0. ; const
  real h3 = 0. ; const
  real h4 = 0. ; const
  real h5 = 0. ; const
  real h6 = 0. ; for
  (unsigned shorts i=0;
  I! =12; ++i) {mh (I, I) =h1; } mh (1,0) =mh
    (0,1) =h2; mh
  (
  2,0) =mh (0,2) =h2; mh (
  2,1) =mh (1,2) =h2; mh (
  3,0) =mh (0,3) =h4; mh (
```

```
  3,1) =mh (1,3) =h5; mh (
  3,2) =mh (2,3) =h5; mh (
  4,0) =mh (0,4) =h5; mh (
  4,1) =mh (1,4) =h3; mh (
  4,2) =mh (2,4) =h6; mh (
  4,3) =mh (3,4) =h2; mh (
  5,0) =mh (0,5) =h5; mh (
  5,1) =mh (1,5) =h6; mh (
  5,2) =mh (2,5) =h3; mh (
  5,3) =mh (3,5) =h2; mh (
  5,4) =mh (4,5) =h2; mh (
  6,0) =mh (0,6) =h5; mh (
  6,1) =mh (1,6) =h4; mh (
  6,2) =mh (2,6) =h5; mh (
  6,3) =mh (3,6) =h6; mh (
  6,4) =mh (4,6) =h3; mh (
  6,5) =mh (5,6) =h5; mh (
  7,0) =mh (0,7) =h6; mh (
  7,1) =mh (1,7) =h5; mh (
  7,2) =mh (2,7) =h3; mh (
  7,3) =mh (3,7) =h5; mh (
  7,4) =mh (4,7) =h5; mh (
  7,5) =mh (5,7) =h4; mh (
  7,6) =mh (6,7) =h2; mh (
  8,0) =mh (0,8) =h3; mh (
  8,1) =mh (1,8) =h5; mh (
  8,2) =mh (2,8) =h6; mh (
  8,3) =mh (3,8) =h3; mh (
  8,4) =mh (4,8) =h6; mh (
  8,5) =mh (5,8) =h5; mh (
  8,6) =mh (6,8) =h2; mh (
  8,7) =mh (7,8) =h2; mh (
  9,0) =mh (0,9) =h5; mh (
  9,1) =mh (1,9) =h5; mh (
  9,2) =mh (2,9) =h4; mh (
  9,3) =mh (3,9) =h6; mh (
  9,4) =mh (4,9) =h5; mh (
  9,5) =mh (5,9) =h3; mh (
  9,6) =mh (6,9) =h6; mh (
  9,7) =mh (7,9) =h3; mh (
  9,8) =mh (8,9) =h5; mh (
 10,0) =mh (0,10) =h3; mh
 (10,1) =mh (1,10) =h6; mh
 (10,2) =mh (2,10) =h5; mh
 (10,3) =mh (3,10) =h3; mh
 (10,4) =mh (4,10) =h5; mh
 (10,5) =mh (5,10) =h6; mh
 (10,6) =mh (6,10) =h5; mh
 (10,7) =mh (7,10) =h5; mh
 (10,8) =mh (8,10) =h4; mh
 (10,9) =mh (9,10) =h2; mh
 (11,0) =mh (0,11) =h6; mh
 (11,1) =mh (1,11) =h3; mh
 (11,2) =mh (2,11) =h5; mh
 (11,3) =mh (3,11) =h5; mh
 (11,4) =mh (4,11) =h4; mh
 (11,5) =mh (5,11) =h5; mh
 (11,6) =mh (6,11) =h3; mh
 (11,7) =mh (7,11) =h6; mh
```

```
(11,8) =mh (8,11) =h5; mh
(11,9) =mh (9,11) =h2; mh
(11,10) =mh (10,11) =h2;
} Boundary conditions
```

1.3 and loadings the loadings for

the modelizations A and B are identical to tests SSLV131A and ZMAT007A. Reference solution

2 Values of the stresses

, strains and local variables, by intercomparison with tests SSLV131A and ZMAT007A. Modelization A Characteristic

3 of

3.1 the modelization Modelization 3D, identical

to SSLV131A. Quantities tested and Comparison

3.2 results with SSLV131

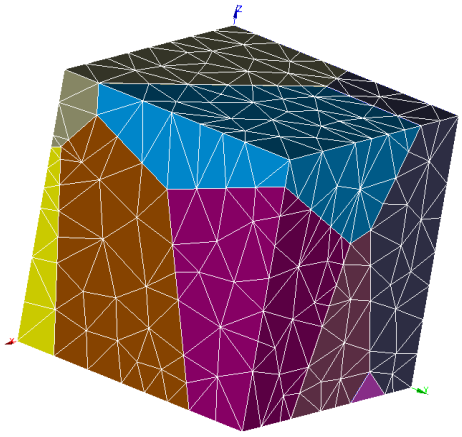
A Identification Reference

Tolerance	% field of	displacement 21
1.E-5 field EPSI_ELGA		
$dy(c)$		3 1.E-
5 4 1.E - 5 6 1.E-5		
EPXY	field	
EPXZ	SIEF_ELGA	
EPYZ		601.8754
1.E- 5 80053.665		
SIXX	1.E-5 78596.607	
SIYY	1.E-5 83948.263	
SIZZ	1.E-5 17339.093	
SIXY	1.E-5 126571.71	
SIXZ	1.E-5 Modelizatio n	
SIYZ	B Characterist ic	

4 of

4.1 the modelization the geometry is that

of an aggregate with 10 grains generated by a procedure python based on cells of Voronoï. One to impose defines planes of cut in edges the boundary conditions. Quantities tested and Comparison



4.2 results with ZMAT007

A. Comparaison of the results
got with Code_Aster with formula *of total* 0.3% . Identification reference

Tolerance	*% formulates*	SIEF_ELGA -16.112
σ_{xx} formula of	EPSI_ELGA	-1,0250
ε_{xx} 03 0,01 formula	of EPSI_ELGA	-9.3714
ε_{yy} 04 0,01 formula	of EPSP_ELGA	-2
ε_{yy} E-04 0,01 Modelization	C Characteristic	

5 of

5.1 the modelization It is identical to that

of the modelization B. Quantities tested and Comparison

5.2 results with ZMAT007

A. Comparaison of the results

got with Code_Aster with formula *of total* 0.3% . Identification reference

Tolerance	*% formulates*	SIEF_ELGA -16.112
σ_{xx} formula of	EPSI_ELGA	-1,0250
ε_{xx} 03 0,01 formula	of EPSI_ELGA	-9.3714
ε_{yy} 04 0,01 formula	of EPSP_ELGA	-2
ε_{yy} E-04 0,01 Summary	of the results	

6 the results is satisfactory

and validates the interface between Code_Aster and MFRONT in *3D, for* behaviors with orthotropy.