MFRON01 – Test of the Code_Aster-MFront *interface* with the models of Summarized

Chaboche:

This test validates behaviors elastoplastic and viscoplastic of Chaboche defined using *MFront* by comparison with behavior similar of *Code_Aster* .

Modelization a: this modelization makes it possible to validate the model elastoplastic with 2 kinematical variables of Chaboche, by comparison with the model `VMIS_CIN2_CHAB` of test SSNV101 in $3D$.

Modelization b: this modelization makes it possible to the model validate élasto-visco-plastic with 2 kinematical variables of Chaboche, by comparison with the model `VISCOCHAB` of test HSNV125D in $3D$.

1 Problem of reference

1.1 Geometry

the geometry is identical to that of tests SSNV101A and HSNV125D

1.2 Properties of the materials

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

C1	145200	Young
C2	0.3	Fish
C5	151.	`R_I`
C6	87.	`R_0`
C7	2.3	`B`
C8	0.43	`K`
C9	6.09	`W`
C10	187.×341.	`C1_I`
C11	29.×17184.	`C2_I`
C12	341	`G1_0`
C13	17184	`G2_0`
C14	1.	`A_inf`

the Mfront file defining the elastoplastic behavior of Chaboche (similar to `VMIS_CIN2_CHAB`), is:

```
@Parser Implicit;
@Behavior Chaboche;
@Algorithm NewtonRaphson_NumericalJacobian;

@MaterialProperty stress Young;
@MaterialProperty real nu;
@MaterialProperty real rho;
@MaterialProperty thermalexpansion alpha;
@MaterialProperty R_inf stress;
@MaterialProperty stress R_0;
@MaterialProperty real B;
@MaterialProperty real K;
@MaterialProperty real W;
@MaterialProperty stress C_inf [2];
@MaterialProperty real   g_0 [2];
@MaterialProperty real   a_inf;

@Includes {
#include " TFEL/Material/Lame.hxx”}
@StateVariable

strain p;    @StateVariable
StrainStensor has [2]; @LocalVariable
average stress        ; @LocalVariable
driven stress        ; @LocalVariable
```

```
StressStensor s0; /*

Initialize Blade coefficients * @InitLocalVars
{using
  namespace tfel:: material:: blade; lambda
  = computeLambda (Young, nu); driven
  =     computeMu (Young, nu); s0
  =     lambda*trace (eto+deto) *Stensor:: Id () +2*mu* (eto+deto); }
@TangentOperator
{using
  namespace tfel:: material:: blade; StiffnessTensor
  Of; Stensor
  4 I; computeElasticStiffness
  <N, Type>:: exe (Of, lambda, driven); getPartialJacobianInvert
  (I); Dt
  = De*Je;} @ComputeStress

{sig
  = lambda*trace (eel) *Stensor:: Id () +2*mu*eel; }
@Integrator
{const
  real eps  =           1.e-12; const
  real M_  2_3 =         real (2) /real (3); const
  strain p_ =           p +theta*dp; const
  Rp_ stress =          R_inf + (R_0-R_inf) *exp (- b*p_); const
  real tmpC0 =          (1.+ (k-1.) *exp (- w*p)); const
  real tmpC =           (1.+ (k-1.) *exp (- w*p_)); const
  real tmpG =           (a_inf+ (1-a_inf) *exp (- b*p_)); StressStensor
  sr_ =         deviator (sig); StressStensor
  sigel =       s0; StrainStensor
  a_ [2]; real
  g_ [2]; for
  (unsigned shorts i=0; I! =2; ++i) {const
     C_ stress =         C_inf [I] *tmpC; const
     stress Concealment =        C_inf [I] *tmpC0; g_
     [I] =                   g_0 [I] *tmpG; a_
     [I] =                   has [I] +theta*da [I]; const
     StressStensor X_ =  M_2_3*C_*a_ [I]; Sr
     _- =                    X_; sigel
     - =                  Cel*a [I] *M_2_3; }
  //
    test on elastic predictor const
  real seqel = sigmaeq (sigel); const
  real Rpel =  R_inf + (R_0-R_inf) *exp (- b*p); const
  real Fel =   seqel - Rpel; yew
  (Fel > 0) {Stensor
    n_ (real (0)); const
    stress seq  _= sigmaeq (sr_); yew
    (seq_>eps*young) {n_
      = 1.5*sr_/seq_; }
    feel
    += dp*n_-deto; FP
    =    (seq_-Rp_) /young; for
    (unsigned shorts i=0; I! =2; ++i) {F
      [I] - =  dp* (n_-g_ [I] *a_ [I]); }
    }
  else {feel
    - = deto; }
  //cout
```

```
<< “J: ” << jacobian << endl; }
```

The coefficients material for the modelization B are defined by functions of the temperature (confer HSNV125D) YOUN

C1	PEA
C2	Rinf
C5	SIGY
C6	B
C7	K
C8	W.C.
C9	1
C10	_T G1
C11	_T ZERO
C12	ZERO
C13	N_
C14	T K_T
C15	

the Mfront file defining the elastoplastic behavior of Chabochce (similar to VISC_CIN2_CHAB `or` VISCOCHAB `)`, with in more integration by a theta-method: @Parser

```
Implicit; @Behavior
Viscochab; @Algorithm
NewtonRaphson_NumericalJacobian; @Theta
0.5; @Epsilon
1.e-8; @IterMax
20; @MaterialProperty
stress Young; @MaterialProperty
real nu; @MaterialProperty
real rho; @MaterialProperty
real alpha; @MaterialProperty
real Rinf; @MaterialProperty
real R0; @MaterialProperty
real B; @MaterialProperty
real K; @MaterialProperty
real W; @MaterialProperty
real C1inf; @MaterialProperty
real g1; @MaterialProperty
real C2inf; @MaterialProperty
real g2; @MaterialProperty
real E; @MaterialProperty
real UNsurK; @Includes
{#include
“TFEL/Material/Lame.hxx”} @StateVariable

real p; @StateVariable
Stensor a1; @StateVariable
```

```
Stensor a2; @LocalVariable
average real    ; @LocalVariable
real driven; /    * Initialize
Blade coefficients * @InitLocalVars
{using
  namespace tfel:: material:: blade; lambda
  = computeLambda (Young, nu); driven =
  computeMu (Young, nu); } //
construction
of the tangent operator from the jacobienne @TangentOperator
{using
  namespace tfel:: material:: blade; StiffnessTensor
  Of; Stensor
  4 I; computeElasticStiffness
  <N, Type>:: exe (Of, lambda, driven); getPartialJacobianInvert
  (I); Dt =
  De*Je; } @ComputeStress

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

{Stensor
  N = Stensor (0.); const
  Stensor a1_ = (a1+theta*da1); const
  Stensor a2_ = (a2+theta*da2); const
  Stensor X1_ = C1inf* (a1_)/1.5; const
  Stensor X2_ = C2inf* (a2_)/1.5; const
  real p_ = (p +theta*dp); const
  Stensor scin = sig - X1_ - X2_; const
  real seq = sigmaeq (scin); const
  real RP = Rinf + (R0-Rinf) *exp (- b*p_); const
  real F = seq - RP; real
  vp=0.; yew (F
  > 0) {vp =
     pow  (F*UNsurK, E); const
     real inv_seq = 1/seq; N =
     1.5      *deviator (scin) *inv_seq; feel
     += vp*dt*n-deto; FP -
     = vp   *dt; fa1
     = da1  - vp*dt*n + g1*vp*dt*a1_; fa2
     = da2  - vp*dt*n + g2*vp*dt*a2_; } else
  {feel
    - = deto; }}
  Boundary conditions
```

1.3 and loadings the loadings

and boundary conditions for the modelization A are identical to test SSNV101. The loadings and boundary conditions for the modelization B are identical to test HSNV125D. Reference solution

2 Values

of the stresses, strains and local variables, by intercomparison with tests SSNV101A and HSNV125D. Modelization

3 A Characteristic

3.1 of the modelization Identical

to SSNV101A Quantities

3.2 tested and results the reference solution

is that of test SSNV101A, and the results are identical Identification

Reference	Tolerance	on node
ε for $NO1$ NUME_ `ORDRE= 1 3 9,7090`	E-2 1,1%	(relative) on node
γ for $NO1$ NUME_ `ORDRE= 1 3 1,4540`	E-1 1,1%	(relative) on node
σ_{11} for $NO1$ NUME_ `ORDRE= 1 3 1,4350`	E+2 0,1%	(relative) on node
p for $NO1$ NUME_ `ORDRE= 1 3 1,9220`	1.9220E-001	(relative) Modelization

4 B Characteristic

4.1 of the modelization Identical

to HSNV125D Quantities

4.2 tested and results the reference solution

is that of test HSNV125A, and the results are identical. Stress

() Urgent $SIXX\ MPa$	**(S) Reference**	**Tolerance**	**in % SIXX**
NODE 1.481 -337.04		1 SIXX	
NODE 1.510 320.54		1 SIXX	
NODE 1.525 211.13		1 SIXX	
NODE 1.534 -31.97		10 SIXX	
NODE 1.579 -89.79		21 Strain	

Time $EPXX$	**(S) Reference**	**Tolerance**	**in % EPXXNŒUD**
1.481 8	10 –	4 0 ^{EPXX}	
NODE 1.579 2.08		10-2 0 ^{Strain}	

Time $EPXY$	**(S) Reference**	**Tolerance**	**in % EPXY**
NODE 1.481 1.4608		10-2 7 EPXY	
NODE 1.510 1.5251		10-2 7 EPXY	
NODE 1.525 1.5917		10-2 7 EPXY	
NODE 1.534 1.6086		10-2 7 EPXY	
NODE 1.579 1.9981		10-2 10 ^{Summary}	

5 of the results the results

are satisfactory and validate the interface between Code_Aster *and MFRONT* in 3D, small strains, for comporements élasto-visco-plastics.