MFRON01 – Test de l'interface *Code_Aster-MFront* avec les lois de Chaboche

Résumé :

Ce test valide des comportements élasto-plastique et visco-plastique de Chaboche définis à l'aide de *MFront* par comparaison avec des comportement similaires de *Code_Aster* .

Modélisation A : cette modélisation permet de valider le modèle élasto-plastique à 2 variables cinématiques de Chaboche, par comparaison au modèle VMIS_CIN2_CHAB du test SSNV101 en 3D .

Modélisation B : cette modélisation permet de valider le modèle élasto-visco-plastique à 2 variables cinématiques de Chaboche, par comparaison au modèle VISCOCHAB du test HSNV125D en 3D .

1 Problème de référence

1.1 Géométrie

La géométrie est identique à celle des tests SSNV101A et HSNV125D

1.2 Propriétés des matériaux

Les coefficients du comportement Mfront sont, pour la modélisation A :

C1	145200	Young
C2	0.3	Poisson
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

Le fichier Mfront définissant le comportement élastoplastique de Chaboche (similaire à VMIS_CIN2_CHAB), est :

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

@MaterialProperty stress young;
@MaterialProperty real nu;
@MaterialProperty real rho;
@MaterialProperty thermalexpansion alpha;
@MaterialProperty stress R_inf;
@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 a[2];
@LocalVariable stress        lambda;
@LocalVariable stress        mu;
```

```
@LocalVariable StressStensor s0;

/* Initialize Lame coefficients */
@InitLocalVars{
  using namespace tfel::material::lame;
  lambda = computeLambda(young,nu);
  mu     = computeMu(young,nu);
  s0     = lambda*trace(eto+deto)*Stensor::Id()+2*mu*(eto+deto);
}
@TangentOperator{
  using namespace tfel::material::lame;
  StiffnessTensor De;
  Stensor4 Je;
  computeElasticStiffness<N,Type>::exe(De,lambda,mu);
  getPartialJacobianInvert(Je);
  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 stress Rp_          = 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 short i=0;i!=2;++i){
     const stress C_         = C_inf[i]*tmpC;
     const stress Cel        = C_inf[i]*tmpC0;
     g_[i]                   = g_0[i]*tmpG;
     a_[i]                   = a[i]+theta*da[i];
     const StressStensor X_  = M_2_3*C_*a_[i];
     sr_                    -= X_;
     sigel                  -= Cel*a[i]*M_2_3 ;
  }
    // test sur predicteur elastique
  const real seqel = sigmaeq(sigel);
  const real Rpel  = R_inf + (R_0-R_inf)*exp(-b*p) ;
  const real Fel   = seqel - Rpel ;
  if(Fel > 0){
    Stensor n_(real(0));
    const stress  seq_ = sigmaeq(sr_);
    if(seq_>eps*young){
      n_ = 1.5*sr_/seq_;
    }
    feel += dp*n_-deto;
    fp    = (seq_-Rp_)/young;
    for(unsigned short i=0;i!=2;++i){
      fa[i]  -= dp*(n_-g_[i]*a_[i]);
    }
  } else {
    feel -= deto;
  }
//cout << "J : " << jacobian << endl;
```

```
}
```

Les coefficients matériau pour la modélisation B sont définis par des fonctions de la température (confer HSNV125D)

C1	YOUN
C2	POISS
C5	Rinf
C6	SIGY
C7	B
C8	K
C9	W
C10	C1_T
C11	G1_T
C12	ZERO
C13	ZERO
C14	N_T
C15	K_T

Le fichier Mfront définissant le comportement élastoplastique de Chabochce (similaire à VISC_CIN2_CHAB ou VISCOCHAB), avec en plus l'intégration par une theta-méthode :

```
@Parser Implicit;
@Behaviour 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 real    lambda;
```

```
@LocalVariable real    mu;
/* Initialize Lame coefficients */
@InitLocalVars{
  using namespace tfel::material::lame;
  lambda = computeLambda(young,nu);
  mu = computeMu(young,nu);
}
// construction de l'opérateur tangent à partir de la jacobienne
@TangentOperator{
  using namespace tfel::material::lame;
  StiffnessTensor De;
  Stensor4 Je;
  computeElasticStiffness<N,Type>::exe(De,lambda,mu);
  getPartialJacobianInvert(Je);
  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.;
  if(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;
  }

}
```

1.3 Conditions aux limites et chargements

Les chargements et conditions aux limites pour la modélisation A sont identiques au test SSNV101.
Les chargements et conditions aux limites pour la modélisation B sont identiques au test HSNV125D.

2 Solution de référence

Valeurs des contraintes, déformations et variables internes, par inter-comparaison avec les tests SSNV101A et HSNV125D.

3 Modélisation A

3.1 Caractéristiques de la modélisation

Identique à SSNV101A

3.2 Grandeurs testées et résultats

La solution de référence est celle du test SSNV101A, et les résultats sont identiques

Identification	Référence	Tolérance
ε sur nœud $NO1$ pour NUME_ORDRE= 1 3	9,7090E-2	1,1 % (relatif)
γ sur nœud $NO1$ pour NUME_ORDRE= 1 3	1,4540E-1	1,1 % (relatif)
σ_{11} sur nœud $NO1$ pour NUME_ORDRE= 1 3	1,4350E+2	0,1 % (relatif)
p sur nœud $NO1$ pour NUME_ORDRE= 1 3	1.9220E-001	1,1 % (relatif)

4 Modélisation B

4.1 Caractéristiques de la modélisation

Identique à HSNV125D

4.2 Grandeurs testées et résultats

La solution de référence est celle du test HSNV125A, et les résultats sont identiques.

Contrainte $SIXX$ (MPa)	Instant(s)	Référence	Tolérance en %
SIXX NŒUD 1	481	-337.04	1
SIXX NŒUD 1	510	320.54	1
SIXX NŒUD 1	525	211.13	1
SIXX NŒUD 1	534	-31.97	10
SIXX NŒUD 1	579	-89.79	21

Déformation $EPXX$	Instant(s)	Référence	Tolérance en %
EPXXNŒUD 1	481	$8\ 10^{-4}$	0
EPXX NŒUD 1	579	$2.08\ 10^{-2}$	0

Déformation $EPXY$	Instant(s)	Référence	Tolérance en %
EPXY NŒUD 1	481	$1.4608\ 10^{-2}$	7
EPXY NŒUD 1	510	$1.5251\ 10^{-2}$	7
EPXY NŒUD 1	525	$1.5917\ 10^{-2}$	7
EPXY NŒUD 1	534	$1.6086\ 10^{-2}$	7
EPXY NŒUD 1	579	$1.9981\ 10^{-2}$	10

5 Synthèse des résultats

Les résultats sont satisfaisants et valident l'interface entre *Code_Aster* et MFRONT en 3D, petites déformations, pour des comporements élasto-visco-plastiques.