MFRON02 – Test of the Code_Aster-MFront *interface* with Summarized

damage models:

This test validates certain behaviors with damage defined using *MFront* by comparison with behavior similar of *Code_Aster* .

Modelization a: this modelization makes it possible to the model validate élasto-visco-plastic with damage of Hayhurst and implicit integration, by comparison with the model HAYHURST of test SSNV225C on a material point.

Modelization b: this modelization makes it possible to the model validate élasto-visco-plastic with damage of Hayhurst and explicit integration, by comparison with the model HAYHURST of test SSNV225A on a material point.

Modelization C: this modelization makes it possible to validate the model with damage of Mazars, by comparison with the model MAZARS of test SSNP113A on an element 3D.

Modelization D: this modelization makes it possible to validate the model with damage of Mazars, by comparison with the model MAZARS of test SSLA103C on an axisymmetric element, with taking into account of drying and the hydration.

Modelization E: this modelization makes it possible to validate the model with damage of Mazars, by comparison with the model MAZARS of test SSNP161A on an element in plane stresses.

1 Problem of reference

1.1 Geometry

the geometry of the modelizations A and B is a material point. That of the modelization C is identical to that of test SSNP113A.

1.2 Properties of the materials

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

C1	145000	Young
C2	0.3	Fish
C5	9,691	K
C6	5,82516E-11	eps0
C7	27,9317	sig0
C8	3, E4	h1
C9	-280	H2
C10	0,33	H1*
C11	1	H2*
C12	9.707593E-08	A0
C13	0,5	AlphaD
C14	1.	DELTA1
C14	0	DELTA2

the Mfront file defining the elastoplastic behavior of Hayhurst into implicit is:

```
@Parser Implicit;
@Behavior Hayhurst;
@Theta 0.5;
@MaterialProperty stress Young;
@MaterialProperty real nu;
@MaterialProperty real rho;
@MaterialProperty real alpha;
@MaterialProperty real K;
@MaterialProperty real epsi0;
@MaterialProperty real sigma0;
@MaterialProperty real h1;
@MaterialProperty real H2;
@MaterialProperty real H1star;
@MaterialProperty real H2star;
@MaterialProperty real A0;
@MaterialProperty real alphaD;
@MaterialProperty real delta1;
@MaterialProperty real delta2;
@Includes {
#include " TFEL/Material/Lame.hxx”}
@StateVariable
real p;    @StateVariable
real H1    ; @StateVariable
```

```
real H2    ; @StateVariable
real D;    @LocalVariable
average real    ; @LocalVariable
real driven    ; /*
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; Stensor
  4 I; Stensor
  Jd; computeElasticStiffness
  <N, Type>:: exe (Hooke, lambda, driven); getPartialJacobianInvert
  (I, Jd); Dt
  = (1-d-dd) *Hooke*Je+- ((Jd) ^ (Hooke* (eel+deel))); }
@ComputeStress
{yew
  (D > 1. - 1.e-8) {sig
    = Stensor (0.); }
  else {sig
    = (1. - D) * (lambda*trace (eel) *Stensor:: Id () +2*mu*eel); }
  }
@Integrator
{Stensor
  N = Stensor (0.); const
  real seq = sigmaeq (sig); Stensor
  sig  0 = (lambda*trace (eel) *Stensor:: Id () +2*mu*eel); const
  real seq0 = sigmaeq (sig0); yew
  (seq > 1.e-8*young) {const
    real H1_ =  H1+theta*dH1; const
    real H2_ =  H2+theta*dH2; const
    real d_ =   d+theta*dd; const
    real H_ =   H1_+H2_; const
    real shp =  sinh (seq* (1-H_) /K/ (1 (d_))); const
    real chp =  sqrt (1.+shp*shp); const
    real trsig=max (trace (sig), 0.); const
    real trsig0=max (trace (sig0), 0.); const
    real shd =  sinh ((alphaD*trsig + (1-alphaD) *seq) /sigma0); const
    real chd =  sqrt (1.+shd*shd); const
    real inv_seq = 1/seq; const
    real dtrsde= (3.*lambda+2.*mu) *theta* (1. - d_) *trsig/trace (sig); N
    =       1.5*deviator (sig) *inv_seq; Stensor
    dsvmde=2.*mu*theta* (1. - d_) *n; Stensor
    dsvm0de=2.*mu*theta*n; //
    system has to solve feel
    += dp*n-deto; FP
    =   dp-epsi0*dt*shp; fH
    1 =  dH1-h1*dp* (H1star-delta1*H1_) *inv_seq; fH
    2 =  dH2-h2*dp* (H2star-delta2*H2_) *inv_seq; fd
    =   dd - A0 * shd*dt; //
    jacobienne dfeel
     _ddeel += 2.*mu*theta*dp*inv_seq* (1-d_) * (Stensor4:: M () - (n^n));
dfeel
    _ddp = N; dfp
    _ddeel =    - (dt*epsi0*chp* (1-H_) /K) *dsvm0de; dfp
    _ddH1 = dt*epsi0*chp*theta*seq0/K; dfp
```

```
    _ddH2 = dfp_ddH1; dfH
    1_ddeel = h1*dp* (H1star - delta1*H1_) *inv_seq*inv_seq*dsvmde; dfH
    2_ddeel = h2*dp* (H2star - delta2*H2_) *inv_seq*inv_seq*dsvmde; dfH
    1_ddp = - h1*inv_seq* (H1star - delta1*H1_); dfH
    2_ddp = - h2*inv_seq* (H2star - delta2*H2_); dfH
    1_ddH1+= h1  *delta1*dp*theta*inv_seq; dfH
    2_ddH2+= H2  *delta2*dp*theta*inv_seq; dfH
    1_ddd= h1*dp* (H1star - delta1*H1_) *theta*seq0*inv_seq*inv_seq; dfH
    2_ddd= h2*dp* (H2star - delta2*H2_) *theta*seq0*inv_seq*inv_seq; dfd
    _ddd +=  A0*dt*chd/sigma0 * (1-alphaD) *theta*seq0; yew
    (trsig > 1.e-8*young) {dfd
_ddeel=-A0*dt*chd/sigma0*  (alphaD*dtrsde*Stensor::  Id  ()  +  (1alphaD)
*dsvmde); dfd
      _ddd +=  A0*dt*chd/sigma0 *alphaD*theta*trsig0; }
    else {dfd
      _ddeel = - A0*dt*chd/sigma0 * (1-alphaD) *dsvmde; }
    }
  else {feel
    - = deto; }      }
```

the Mfront file defining the elastoplastic behavior of Hayhurst into explicit differs from the precedent only by integration: @Parser

```
RungeKutta; @Behavior
Hayhurst; @Algorithm
rk54; ….
@Derivative
{seq
  = sigmaeq (sig); Stensor
  sig  0 = (lambda*trace (eel) *Stensor:: Id () +2*mu*eel); const
  real seq0 = sigmaeq (sig0); yew
  (seq > 0.01*young) {return
    false; }
  yew
  (endo > 1. - 1.e-8) {sig
     = Stensor (0.); dp
     =0.; dendo
     =0.; dH
     1=0.; dH
     2=0.; }
  real
  inv_seq (0); N
  = Stensor (0.); yew
  (seq > 1.e-8*young) {H=
     H1+H2; dp
     =  epsi0*sinh (seq0* (1-H) /K); dH
     1=h1/seq * (H1star - delta1*H1) *dp; dH
     2=h2/seq * (H2star - delta2*H2) *dp; trsigplus
     =0; yew
     (sequid==0) {real
        v1; real
        v2; real
        v3; sig.computeEigenValues
        (v1, v2, v3); trsigplus
        =max (0. , v1); } yew
        (sequid
     ==1) {trsig
        =trace (sig); trsigplus
        =0.; yew (trsig
```

```
          >0) {trsigplus
             =trsig; }} dendo


      =A0 * sinh ((alphaD*trsigplus + (1-alphaD) *seq) /sigma0); inv_seq
      = 1/seq; N =
      1.5       *deviator (sig) *inv_seq; } devp

  = dp*n; deel
  = deto - devp; }  The coefficients
```

material for the modelization C are (cf SSNP113A) 3,2E10

C1	YOUN	0,2 PEAS
C2		1,15
C5	AC 0,8	AT
C6	1391,3	
C7	BC 10000	
C8	, BT 0,7	K
C9	9,37	E
C10	- 5 ed0	the Mfront

file defining the behavior of Mazars by a direct resolution (noniterative) (modelization C) is: @Parser

```
DefaultParser; @Behavior
mazars; @ProvidesTangentOperator
; @MaterialProperty

stress Young; @MaterialProperty
real nu; @MaterialProperty
real rho; @MaterialProperty
real alpha; @MaterialProperty
real ac; @MaterialProperty
real At; @MaterialProperty
real Bc; @MaterialProperty
real BT; @MaterialProperty
real K; @MaterialProperty
real ed0; @Includes

{#include
"TFEL/Material/Lame.hxx"} @AuxiliaryStateVariable


real D; @AuxiliaryStateVariable
real Y; @AuxiliaryStateVariable
real Tmax; @AuxiliaryStateVariable
real eeqcor; @LocalVariable

average real;    @LocalVariable
real driven; @LocalVariable
real A; @LocalVariable
real B; @LocalVariable
```

```
real eeqc; /*    Initialize

Blade coefficients * @InitLocalVars
{using namespace
  tfel:: material:: blade; lambda =
  computeLambda (Young, nu); driven = computeMu
  (Young, nu); } @Integrator


 {const Stensor
  etop=eto+deto; const real
  dmax=0.99999; const real
  normeps= etop | etop; yew (normeps
  <1.e-30) {sig=Stensor
      (real (0)); } else {
  real
  e1;
      real e2;
      real e3;
      real EPP
      1; real EPP
      2; real EPP
      3; real PNS
      1; real PNS
      2; real PNS
      3; real PS
      1; real PS
      2; real PS
      3; etop.computeEigenValues
      (e1, e2, e3); Stensor sig
       0 = (average  *trace (etop) *Stensor:: Id () +2*mu*etop); real s1;
real
      s2; real
      s3; sig
      0.computeEigenValues
      (s1, s2, s3); real gam=1.
      ; yew (min (s1,
      min (s2, s3))<0) {pns1=min (0.
         , s1); pns2=min (0.
         , s2); pns3=min (0.
         , s3); gam=-sqrt (PNS
         1*pns1+pns2*pns2+pns3*pns3)/(pns1+pns2+pns3); } ppe1=max (
      0.
      , e1); ppe2=max (0.
      , e2); ppe3=max (0.
      , e3); pps1=max (0.
      , s1); pps2=max (0.
      , s2); pps3=max (0.
      , s3); eeqc= gam*sqrt
      (ppe1*ppe1+ppe2*ppe2+ppe3*ppe3); eeqcor=max (
      eeqc, eeqcor); real r= (PS
      1+pps2+pps3)/(ab (s1) +abs (s2) +abs (s3)); A=At* (2*r*r
      * (1. - 2*k) - r* (1-4*k))+Ac* (2*r*r-3*r+1); B=r*r*Bt+ (1
      - r*r) *Bc; real Y1=max
      (ed0, eeqcor); Y=max (Y1, Y)
      ; d=1- (1-A) *ed
      0/Y-A*exp (- B* (Y-ed0)); d=min (dmax,
      d); sig = (1. - D
```

```
          ) * (lambda*trace (etop) *Stensor:: Id () +2*mu*etop); } yew
(computeTangentOperator

  _) {using namespace
     tfel:: material:: blade; StiffnessTensor
     Hooke; computeElasticStiffness
     <N, Type>:: exe (Hooke, lambda, driven); Dt = (1. - D)
     *Hooke; }} @UpdateAuxiliaryStateVars


{Tmax=max (Tmax
  , T); } The Mfront
```

file defining the behavior of Mazars into implicit (modelizations D and E) is: @Parser Implicit

```
; @Behavior
mazars; @Algorithm
NewtonRaphson_NumericalJacobian; @Theta 1. ;

@Epsilon 1.
e-12; @IterMax 100
; @MaterialProperty

stress Young; @MaterialProperty
real nu; @MaterialProperty
real rho; @MaterialProperty
real alpha; @MaterialProperty
real ac; @MaterialProperty
real At; @MaterialProperty
real Bc; @MaterialProperty
real BT; @MaterialProperty
real K; @MaterialProperty
real ed0; @Includes {

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


real D; @AuxiliaryStateVariable
real Y; @AuxiliaryStateVariable
real Tmax; @AuxiliaryStateVariable
real eeqcor; @LocalVariable

average real; @LocalVariable
real driven; @LocalVariable
real A; @LocalVariable
real B; @LocalVariable
real eeqc; /* 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; Stensor
  Jd; computeElasticStiffness

  <N, Type>:: exe (Hooke, lambda, driven); getPartialJacobianInvert
  (I, Jd); Dt = (1-d-dd) *
  Hooke*Je+- ((Jd) ^ (Hooke* (eel+deel))); //cost << "Dt
:  " << Dt << endl; } @ComputeStress


{sig = (1. - D) * (
    lambda*trace (eel) *Stensor:: Id () +2*mu*eel); } @Integrator
{const
Stensor
  etop=eto+deto; const real dmax
  =0.99999; real e1; real
  e2; real
  e3; real
  ppe1; real
  ppe2; real
  ppe3; real
  pns1; real
  pns2; real
  pns3; real
  pps1; real
  pps2; real
  pps3; etop.computeEigenValues

  (e1, e2, e3); Stensor sig0 = (
  lambda*trace  (etop) *Stensor:: Id () +2*mu*etop); real s1; real s2;
  real s3;
  sig0.computeEigenValues

  (s1, s2, s3); real gam=1. ; yew
  (min (s1, min (s2
  , s3))<0) {pns1=min (0. , s1);
     pns2=min (0. , s2);
     pns3=min (0. , s3);
     gam=-sqrt (pns1*pns
     1+pns2*pns2+pns3*pns3)/(pns1+pns2+pns3); } ppe1=max (0. , e1)
  ;
  ppe2=max (0. , e2);
  ppe3=max (0. , e3);
  pps1=max (0. , s1);
  pps2=max (0. , s2);
  pps3=max (0. , s3);
  eeqc= gam*sqrt (EPP
  1*ppe1+ppe2*ppe2+ppe3*ppe3); eeqcor=max (eeqc, eeqcor
  ); real r= (pps1+pps2
  +pps3)/(ab (s1) +abs (s2) +abs (s3)); A=At* (2*r*r* (1. - 2
  *k) - r* (1-4*k))+Ac* (2*r*r-3*r+1); B=r*r*Bt+ (1-r*r) *
  Bc; real Y1=max (ed0, eeqcor
  ); Y=max (Y1, Y); feel
  - = deto; real
  dnew=1- (1-A)
  *ed0/Y-A*exp (- B* (Y-ed0)); dnew=min (dmax, dnew
  ); fd = dd-dnew+d;
  } @UpdateAuxiliaryStateVars

{Tmax=max (Tmax, T);
```

```
} Boundary conditions
```

1.3 and loadings For each modelization

, the loadings and boundary conditions are identical to those of the tests of reference. Reference solution

2 Values of the stresses

, strains and local variables, by intercomparison with the tests of reference. Modelization A Characteristic

3

3.1 of the modelization Modelization material point

in large deformations with implicit integration, identical to SSNV225C. Quantities tested

3.2 and Comparison results with

SSNV225C (even reference solution: SSNV225A) Identification Urgent

() Reference	Tolerance h	formulate	2000 0,020968
$EPYY$	2000	4000	0,05093
$EPYY$	4000	2000	0,0323
$V11(endo)$	2000	4000	0,06808
$V11(endo)$	4000	1520	6,6539
$dEPYY/dt$	1520	% Modelization	B Characteristic

4

4.1 of the modelization Modelization material point

with explicit integration, comparable to SSNV225A, but in small strains. Quantities tested

4.2 and Comparison results with

SSNV225A (the results differ by the type of strain, and are provided as an indication) Identification Times

(H) Reference	Tolerance	formulates	2000 0,020968
$EPYY$	2000	2000	0,0323
$V11(endo)$	2000	1520	6,6539
$dEPYY/dt$	1520	Modelization	C Characteristic

5

5.1 of the modelization Modelization material point

with explicit integration, comparable to SSNP113A Quantities tested

5.2 and Comparison results with

SSNP113A One compared to the 3 time step

different (at the end of stage 1, during the phase of growth of the damage and at the end of the loading) strains, the stresses as well as the value of the damage. Identification Reference

tolerance	% N°	10 9.375 10 –

5.0,1	ε_{xx}	3.00 106.0,1	– 1.875
	σ_{xx}	10 –	5.0,1
	ε_{yy}	0. 10-8 0.	10 –
	σ_{yy}	8	0. 10 –
	ε_{xy}	8	0. 10 –
	σ_{xy}	8	Identification
	D		Reference

	tolerance	**% N°**	**25 1.64 10 –**
4.0,1	ε_{xx}	2.04 106.0,1	8.67
	σ_{xx}	10-5 0,1	1.35
	ε_{yy}	10 6.0,1	7.03
	σ_{yy}	10 –	5.0,1
	ε_{xy}	6.34 10 5.0,1	0.66211
	σ_{xy}	0,1 Identification	
	D		Reference

	tolerance	**% N°**	**310 1.50 10 –**
3.0,1	ε_{xx}	3.69 105.0,1	2.09
	σ_{xx}	10-3 0,1	4.59
	ε_{yy}	10 5.0,1	1.41
	σ_{yy}	10 –	3.0,1
	ε_{xy}	2.16 10 5.0,1	0.99423
	σ_{xy}	0,1 Modelization	
	D		D Characteristic

6

6.1 of the modelization Modelization similar

to SSLA103C. Quantities tested

6.2 and Variable results Standard

Time of Reference		Reference Tolerance	Object	OLE 3600 ANALYTIQUE
ε_{xx}	10	- 4 0.50% Object	OLE 3600	ANALYTIQUE
ε_{yy}	10	- 4 0.50% Object	OLE 3600	ANALYTIQUE
ε_{xx}^{p}	. 1,00	E-006 Object	OLE	1.00E-006
ε_{yy}^{p}	. 1,00	E-006 Object	OLE	1.00E-006
σ_{xx}	. 1,00	E-006 Object	OLE	1.00E-006
σ_{yy}	. 1,00	E-006 formulates		1.00E-006

D	. 1,00	E-006 Modelization	1.00E-006

7

7.1 of the modelization Modelization similar

to SSNP161A. The computation in plane stresses is realized by the method of DEBORST. Quantities tested

7.2 and Strains results and

stresses with the node formulates. Standard $N1$ identification

Increment of	reference	Value of reference	Forced Tolerance	σ yy
7 "ANALYTIQUE " formulates	7	5% Strain	$-10842857\,Pa$	ε
yy 7 "ANALYTIQUE " formulates	7	5% Stress	$-3{,}00\text{E-}004$	yy
69 "ANALYTIQUE " formulates	69	5% Summary	$-42428571\,Pa$	the results

8 the results is

satisfactory and validates the interface between Code_Aster and MFRONT *in 3D,* axisymmetric and plane stresses, in great or small strains, for behaviors with damage.