Dimensionnement des pièces chaudes à la fatigue thermomécanique

Intégration de Mfront dans le processus de dimensionnement

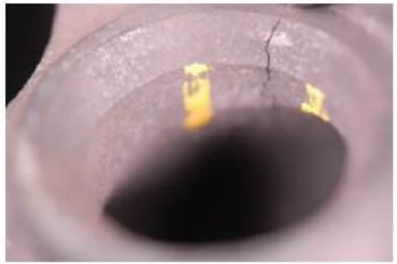
1.Contexte

MÉTHODOLOGIE DE DIMENSIONNEMENT DU GROUPE PSA

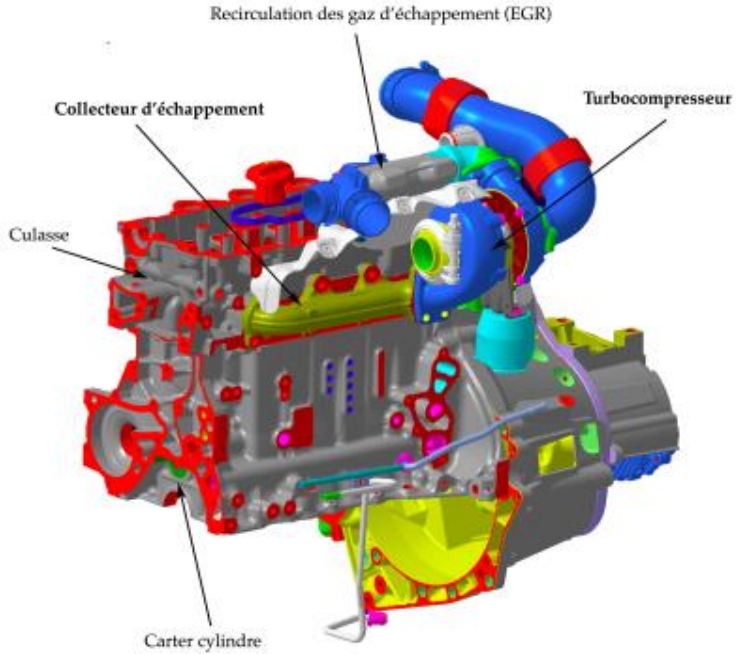
Contexte

- Optimisation des pièces du groupe motopropulseur
  - Chargements thermiques sévères
  - Pièces contraintes mécaniquement

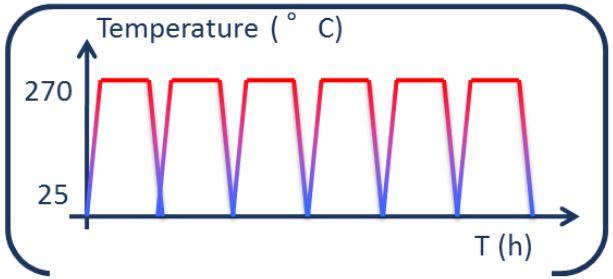
- Apparition de fissures dans des zones très sollicitées



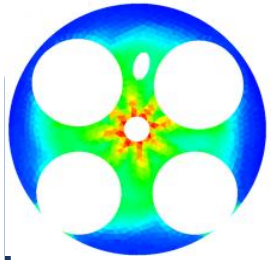
- Dimensionnement à la fatigue thermomécanique
  [Charkaluk 1999, Verger 2002, Szmytka 2007, …]



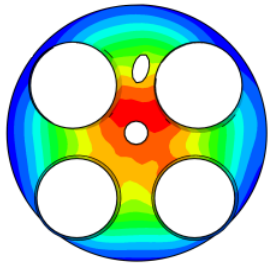
Dimensionnement actuel PSA – Calcul culasse



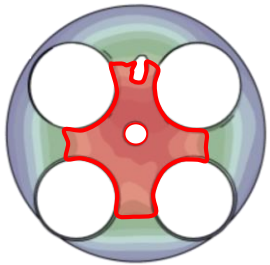
Chargement cyclique sévère



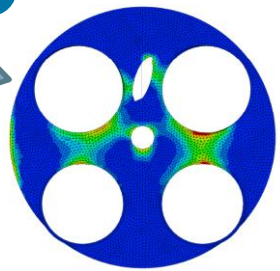
Analyse CFD



Analyse thermique



Analyse du vieillissement



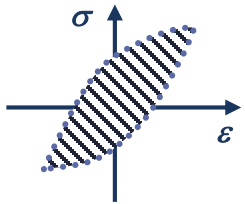
Analyse mécanique

Modèles de comportement
Type Chaboche pour Aluminiums - Szmytka 1 & 2 pour aciers et fontes

Non intégré dans ABAQUS Standard → UMAT fortran

Densité d'énergie dissipée locale comme variable d'endommagement
Nombre de cycles à amorçage de fissure

$$\Delta W = A_0 . N^{\beta}$$



Dimensionnement actuel PSA – Modèle de comportement

Pour les alliages d'aluminiums, modèle élasto-viscoplastique :

Décomposition de la déformation mécanique

$$\underline{\underline{\varepsilon}} = \underline{\underline{\varepsilon}}^{e} + \underline{\underline{\varepsilon}}^{vp}$$

Loi d'élasticité linéaire classique ⇨ ***Hooke***

$$\underline{\underline{\varepsilon}}^{e} = \frac{1+\nu}{E}\underline{\underline{\sigma}} - \frac{\nu}{E}tr(\underline{\underline{\sigma}})\underline{\underline{I}}$$

Critère de plasticité $f = (\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq} - \sigma_y$ avec $(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq} = \sqrt{\frac{3}{2}(\underline{\underline{\sigma}} - \underline{\underline{X}})^{dev} : (\underline{\underline{\sigma}} - \underline{\underline{X}})^{dev}}$

Loi d'évolution de la vitesse de déformation viscoplastique

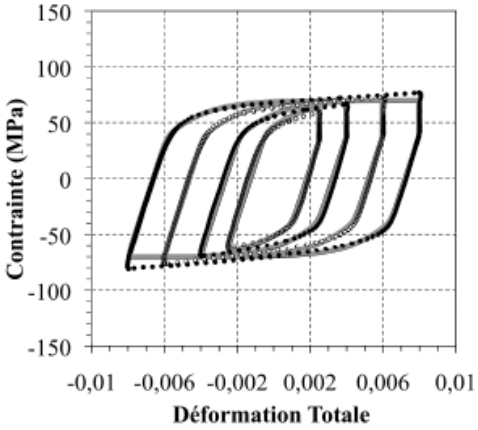
$$\dot{\underline{\underline{\varepsilon}}}^{vp} = \frac{3}{2}\dot{p}\frac{(\underline{\underline{\sigma}} - \underline{\underline{X}})^{dev}}{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq}}$$

$$\dot{p} = \left\langle \frac{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq} - \sigma_y}{\eta} \right\rangle^{m}$$

Loi d'évolution de la variable d'écrouissage cinématique

$$\underline{\underline{\alpha}} = \frac{3}{2C}\underline{\underline{X}}$$

$$\dot{\underline{\underline{\alpha}}} = \dot{\underline{\underline{\varepsilon}}}^{vp} - \frac{3D}{2C}\dot{p}\underline{\underline{X}}$$



[Osmond 2010]

Dimensionnement actuel – UMAT Fortran

Rôle de la User Material (UMAT)

Décrire un comportement complexe, non linéaire et **indisponible dans ABAQUS**

Interpréter la **carte matériau**

Fournir à Abaqus une cartographie des contraintes **à l'instant du calcul**

Permettre une **convergence** adéquate du calcul par éléments finis

Qu'est ce que c'est ?

Un **fichier FORTRAN** permettant de réaliser l'ensemble des opérations souhaitées

Une **carte matériau** associé à la sous routine

Un **lien étroit** avec le solveur ABAQUS

La possibilité de **traiter toutes les données que l'on souhaite via les variables d'état**

Avantages et Inconvénients de la UMAT Fortran

☺ Écriture de modèles de comportements complexes

☺ Post-traitement des variables souhaitées

☺ Maîtrise totale du processus d'intégration numérique

☺ Ecriture en Fortran77 : relative facilité d'utilisation et de programmation

☹ Complexité d'implémentation du processus algorithmique demandant un certain apprentissage

- **6 mois** pour obtenir un nouveau modèle (souvent travail de doctorant) + maintenance

☹ Obligation d'écrire le schéma d'intégration pour chaque type d 'élément utilisé

☹ Très peu de compétences en interne

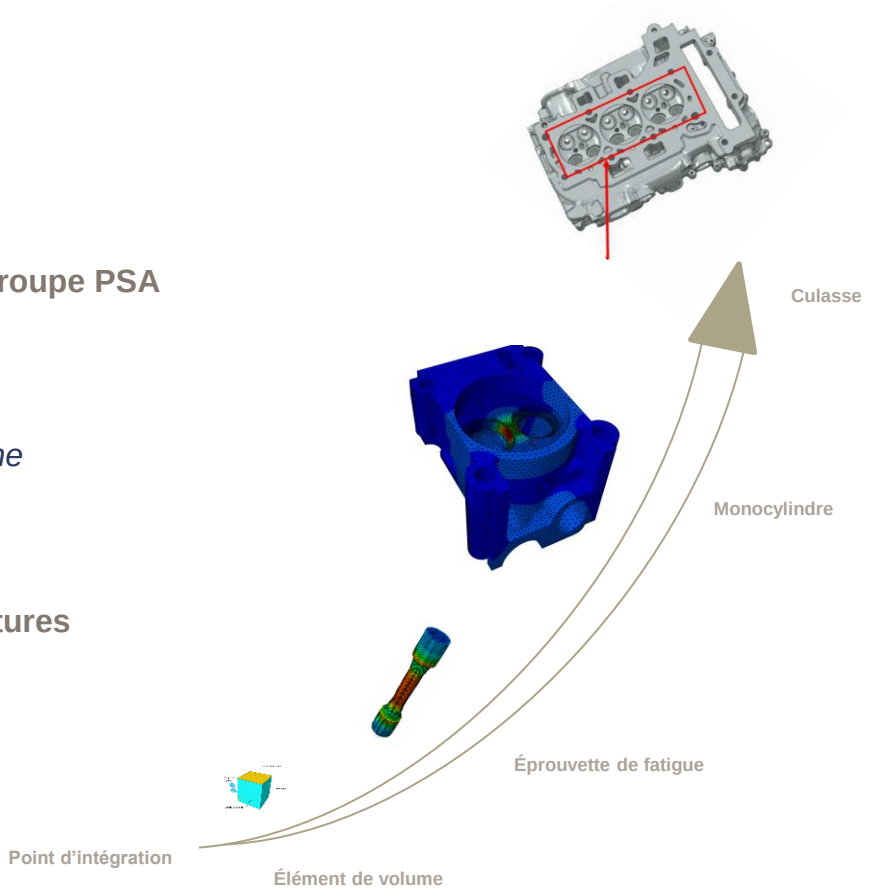
Nouveaux modèles plus complexes développés récemment : Osmond 2007 (aluminiums), Rémy 2011 (aciers), Forré 2015 (tôles) , etc…

→ problèmes de convergence

Démarche mise en place

- **1er stage 2017 (2 mois)** – *Charlie Prétot*
  - Identifier besoins métiers
  - Mettre en place Mfront sur la plateforme du groupe PSA
  - Lancement premiers calculs

- **2ème stage 2018 (6 mois)** - *Khouloud Derouiche*
  - Intégration du modèle des Aluminiums
  - Test de différents algorithmes
  - Comparaison au fortran sur différentes structures



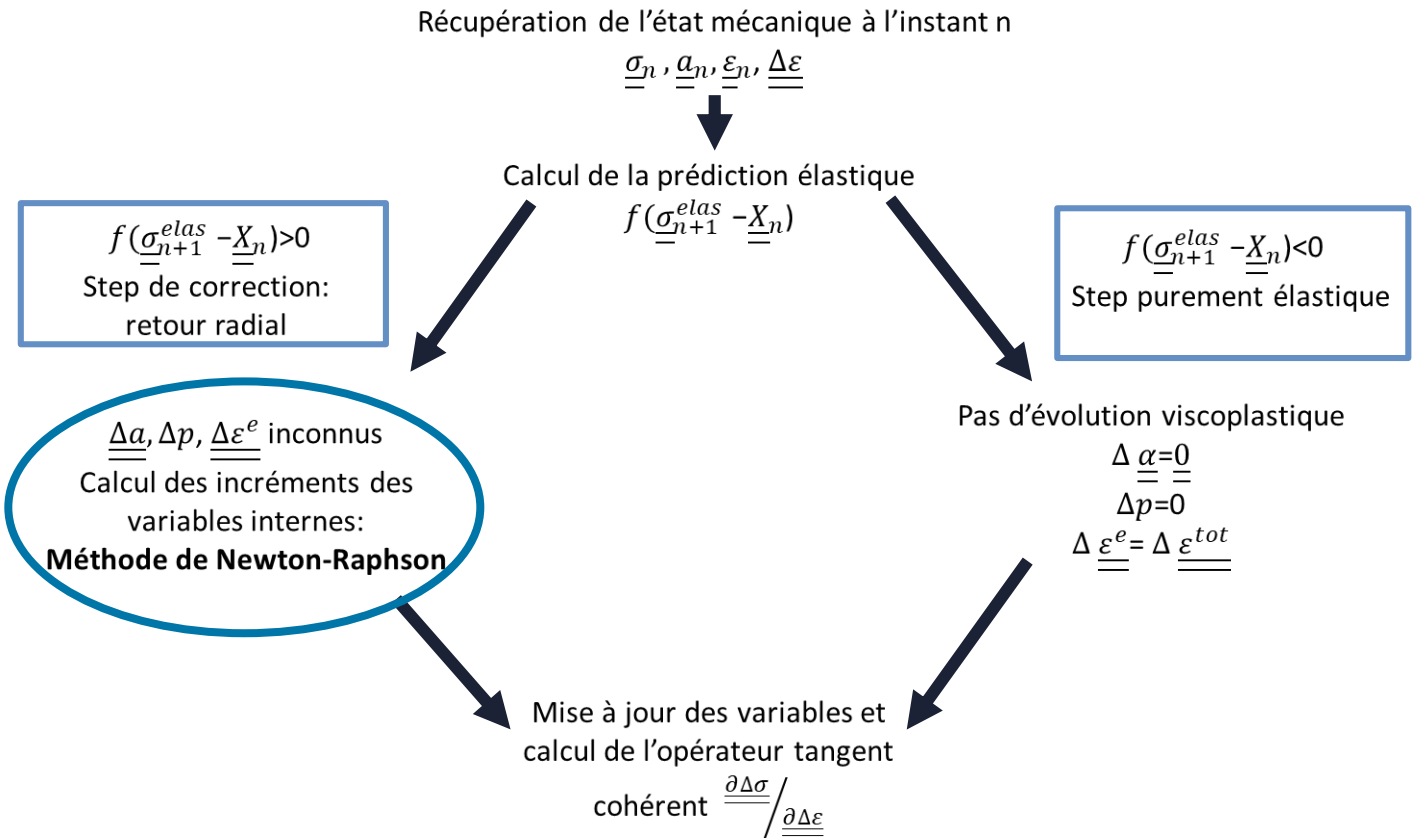
Plan de la présentation

- 1. Contexte
- 2. Intégration numérique
  - Principe d'intégration locale
  - Algorithme avec Jacobienne analytique
  - Algorithme avec Jacobienne numérique
  - Algorithme avec système réduit
- 3. Résultats
  - Principe de validation
  - MTEST
  - ABAQUS – Elément de volume
  - ABAQUS – Eprouvette de fatigue thermique
  - ABAQUS – Monocylindre
  - ABAQUS – Culasse
- Bilan

2. Intégration numérique

COMPARAISON DE DIFFÉRENTES METHODOLOGIES D'INTÉGRATION

Principe d'intégration locale



Identification des algorithmes à tester

Méthode d'intégration avec Jacobienne analytique	- Réalisable peu importe le modèle - Fastidieux avec des codes comme le Fortran - Calcul analytique des termes de la Jacobienne
Méthode d'intégration avec Jacobienne numérique	- Très facile à intégrer - Test des modèles de comportement sur structures quasi instantanément - Calcul les termes de la matrice Jacobienne par la méthode des différences finies - Temps de résolution "local" a priori plus important mais temps limité pour le développement.
Méthode d'intégration avec réduction du système d'équation à une inconnue	- Une seule équation scalaire à considérer dans l'algorithme de Newton-Raphson - Même algorithme que les UMAT fortran PSA - Gain de temps sur l'inversion de la matrice Jacobienne - Long travail d'intégration difficile à mettre en œuvre - Calcul de l'opérateur tangent plus complexe

Algorithme avec Jacobienne Analytique – Principe d'intégration locale

$\underline{\underline{\Delta a}}, \Delta p, \underline{\underline{\Delta \varepsilon^e}}$ inconnus
Calcul des incréments des variables internes:
Méthode de Newton-Raphson

$$[J] = \begin{bmatrix} \frac{\partial r_{\Delta \underline{\underline{\varepsilon}}^{el}}}{\partial \Delta \underline{\underline{\varepsilon}}^{el}} & \frac{\partial r_{\Delta \underline{\underline{\alpha}}}}{\partial \Delta \underline{\underline{\varepsilon}}^{el}} & \frac{\partial r_{\Delta p}}{\partial \Delta \underline{\underline{\varepsilon}}^{el}} \\ \frac{\partial r_{\Delta \underline{\underline{\varepsilon}}^{el}}}{\partial \Delta \underline{\underline{\alpha}}} & \frac{\partial r_{\Delta \underline{\underline{\alpha}}}}{\partial \Delta \underline{\underline{\alpha}}} & \frac{\partial r_{\Delta p}}{\partial \Delta \underline{\underline{\alpha}}} \\ \frac{\partial r_{\Delta \underline{\underline{\varepsilon}}^{el}}}{\partial \Delta p} & \frac{\partial r_{\Delta \underline{\underline{\alpha}}}}{\partial \Delta p} & \frac{\partial r_{\Delta p}}{\partial \Delta p} \end{bmatrix}$$

Initialisation $\Delta[A]$ avec
$[A] = \left(\underline{\underline{\varepsilon^e}}, \underline{\underline{a}}, p\right)$ $(13\ inconnus)$

↓

Calcul des fonctions résidus
$[R] = (r_{\Delta \underline{\underline{\varepsilon}}^e}, r_{\underline{\underline{\Delta a}}}, r_{\Delta p})$

↓

Calcul de la matrice jacobienne $[J]$

↓

Calcul $\Delta[A]_{n+1} = \Delta[A]_n - [J]^{-1}[R](\Delta[A]_n)$

↓

Recalcul de $[R]$ en fonction de $\Delta[A]$ et vérification du critère de convergence

Sinon → retour au calcul de $\Delta[A]_{n+1}$

Algorithme avec Jacobienne Analytique – Equations des résidus et des dérivées

$$\underline{\underline{r_{\Delta\varepsilon^e}}} = \underline{\underline{\Delta\varepsilon^e}} + \underline{\underline{\Delta\varepsilon^{vp}}} - \underline{\underline{\Delta\varepsilon^{tot}}} = \underline{\underline{\Delta\varepsilon^e}} + \Delta p\, \underline{\underline{n}} - \underline{\underline{\Delta\varepsilon^{tot}}}$$

- $\dfrac{\partial \underline{\underline{r_{\Delta\varepsilon^e}}}}{\partial \underline{\underline{\Delta\varepsilon^e}}} = \underline{\underline{\underline{\underline{I}}}} + \Delta p \dfrac{\partial \underline{\underline{n}}}{\partial \underline{\underline{\Delta\varepsilon^e}}} = \underline{\underline{\underline{\underline{I}}}} + \Delta p \dfrac{1}{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq}} (\underline{\underline{\underline{\underline{M}}}} - \underline{\underline{n}} \otimes \underline{\underline{n}})\, \underline{\underline{\underline{\underline{E}}}}$$
- $\dfrac{\partial \underline{\underline{r_{\Delta\varepsilon^e}}}}{\partial \underline{\underline{\Delta\alpha}}} = \Delta p \dfrac{\partial \underline{\underline{n}}}{\partial \underline{\underline{\Delta\alpha}}} = - \dfrac{2\,\Delta p}{3(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq}} (\underline{\underline{\underline{\underline{M}}}} - \underline{\underline{n}} \otimes \underline{\underline{n}})$
- $\dfrac{\partial \underline{\underline{r_{\Delta\varepsilon^e}}}}{\partial \Delta p} = \underline{\underline{n}}$

$$[J] = \begin{bmatrix} \dfrac{\partial \underline{\underline{r_{\Delta\varepsilon^{el}}}}}{\partial \underline{\underline{\Delta\varepsilon^{el}}}} & \dfrac{\partial \underline{\underline{r_{\Delta\alpha}}}}{\partial \underline{\underline{\Delta\varepsilon^{el}}}} & \dfrac{\partial r_{\Delta p}}{\partial \underline{\underline{\Delta\varepsilon^{el}}}} \\ \dfrac{\partial \underline{\underline{r_{\Delta\varepsilon^{el}}}}}{\partial \underline{\underline{\Delta\alpha}}} & \dfrac{\partial \underline{\underline{r_{\Delta\alpha}}}}{\partial \underline{\underline{\Delta\alpha}}} & \dfrac{\partial r_{\Delta p}}{\partial \underline{\underline{\Delta\alpha}}} \\ \dfrac{\partial \underline{\underline{r_{\Delta\varepsilon^{el}}}}}{\partial \Delta p} & \dfrac{\partial \underline{\underline{r_{\Delta\alpha}}}}{\partial \Delta p} & \dfrac{\partial r_{\Delta p}}{\partial \Delta p} \end{bmatrix}$$

avec $\underline{\underline{\underline{\underline{M}}}} = \dfrac{3}{2}\, \underline{\underline{\underline{\underline{I}}}} - \underline{\underline{1}} \otimes \underline{\underline{1}}$

$\underline{\underline{\sigma}} = \underline{\underline{\underline{\underline{E}}}} : \underline{\underline{\varepsilon^e}}$

$$\underline{\underline{r_{\Delta\alpha}}} = \underline{\underline{\Delta\alpha}} - \Delta p\, \underline{\underline{n}} + \Delta p\, D\, \underline{\underline{\alpha}}$$

- $\dfrac{\partial \underline{\underline{r_{\Delta\alpha}}}}{\partial \underline{\underline{\Delta\varepsilon^e}}} = - \Delta p \dfrac{\partial \underline{\underline{n}}}{\partial \underline{\underline{\Delta\varepsilon^e}}} = - \dfrac{\Delta p}{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq}} (\underline{\underline{\underline{\underline{M}}}} - \underline{\underline{n}} \otimes \underline{\underline{n}})\, \underline{\underline{\underline{\underline{E}}}}$$
- $\dfrac{\partial \underline{\underline{r_{\Delta\alpha}}}}{\partial \underline{\underline{\Delta\alpha}}} = \underline{\underline{\underline{\underline{I}}}} - \Delta p \dfrac{\partial \underline{\underline{n}}}{\partial \underline{\underline{\Delta\alpha}}} = (1 + \Delta p D)\, \underline{\underline{\underline{\underline{I}}}} + \dfrac{2\, C\, \Delta p}{3(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq}} (\underline{\underline{\underline{\underline{M}}}} - \underline{\underline{n}} \otimes \underline{\underline{n}})$
- $\dfrac{\partial \underline{\underline{r_{\Delta\alpha}}}}{\partial \Delta p} = - \underline{\underline{n}} + D\, \underline{\underline{a}}$

$$r_{\Delta p} = \Delta p - \left\langle \frac{f}{K} \right\rangle^{m} \Delta t$$

- $\dfrac{\partial r_{\Delta p}}{\partial \underline{\underline{\Delta\varepsilon^e}}} = - \dfrac{\Delta t\, m}{K} \left\langle \dfrac{f}{K} \right\rangle^{m-1} \underline{\underline{\underline{\underline{E}}}} : \underline{\underline{n}}$
- $\dfrac{\partial r_{\Delta p}}{\partial \underline{\underline{\Delta\alpha}}} = \dfrac{3}{2} \dfrac{C\, \Delta t\, m}{K} \left\langle \dfrac{f}{K} \right\rangle^{m-1} \underline{\underline{n}}$
- $\dfrac{\partial r_{\Delta p}}{\partial \Delta p} = 1$

Algorithme avec Jacobienne Analytique – Opérateur tangent cohérent

$$\delta[R] = [J].\delta\Delta[A] - \begin{pmatrix} \delta\Delta\underline{\underline{\varepsilon}}^{to} \\ 0 \\ \underline{\underline{0}} \end{pmatrix} = \begin{pmatrix} \underline{\underline{0}} \\ 0 \\ \underline{\underline{0}} \end{pmatrix}$$

$$[A] = (\Delta\underline{\underline{\varepsilon}}^{e}, \Delta p, \Delta\underline{\underline{a}})$$

$$[R] = (r_{\Delta\underline{\underline{\varepsilon}}^{e}}, r_{\Delta p}, r_{\Delta\underline{\underline{a}}})$$

$$\delta\Delta[A] = [J]^{-1}. \begin{pmatrix} \Delta\underline{\underline{\varepsilon}}^{to} \\ 0 \\ \underline{\underline{0}} \end{pmatrix} \quad \text{avec} \quad [J]^{-1} = [J^*] = \begin{pmatrix} \left[\frac{\partial r_{\Delta\underline{\underline{\varepsilon}}^{e}}}{\partial\Delta\underline{\underline{\varepsilon}}^{e}}\right]^* & \left[\frac{\partial r_{\Delta\underline{\underline{\varepsilon}}^{e}}}{\partial\Delta\underline{\underline{a}}}\right]^* & \left[\frac{\partial r_{\Delta\underline{\underline{\varepsilon}}^{e}}}{\partial\Delta p}\right]^* \\ \left[\frac{\partial r_{\Delta\underline{\underline{a}}}}{\partial\Delta\underline{\underline{\varepsilon}}^{e}}\right]^* & \left[\frac{\partial r_{\Delta\underline{\underline{a}}}}{\partial\Delta\underline{\underline{a}}}\right]^* & \left[\frac{\partial r_{\Delta\underline{\underline{a}}}}{\partial\Delta p}\right]^* \\ \left[\frac{\partial r_{\Delta p}}{\partial\Delta\underline{\underline{\varepsilon}}^{e}}\right]^* & \left[\frac{\partial r_{\Delta p}}{\partial\Delta\underline{\underline{a}}}\right]^* & \left[\frac{\partial r_{\Delta p}}{\partial\Delta p}\right]^* \end{pmatrix}$$

$$\delta\Delta\underline{\underline{\sigma}} = \underline{\underline{\underline{\underline{E}}}} : \delta\Delta\underline{\underline{\varepsilon}}^{e}$$

$$\boxed{\underline{\underline{\underline{\underline{L_c}}}} = \frac{\partial\Delta\underline{\underline{\sigma}}}{\partial\Delta\underline{\underline{\varepsilon}}} = \underline{\underline{\underline{\underline{E}}}} \quad \left[\frac{\partial r_{\Delta\underline{\underline{\varepsilon}}^{e}}}{\partial\Delta\underline{\underline{\varepsilon}}^{e}}\right]^*}$$

Code MFront - Méthode d'intégration avec Jacobienne analytique

```
@Parser Implicit;
@Behaviour EVPunifieeECNLNortonVieillissement;
@Epsilon 1.e-14;
@ModellingHypotheses{".+"};
@Brick StandardElasticity;
@ElasticMaterialProperties{"YoungModulus.mfront", "PoissonRatio.mfront"};
@Theta 1.;
/*déclaration des variables */
@StateVariable strain p;
p.setGlossaryName("EquivalentPlasticStrain");
@StateVariable StrainStensor a;
a.setEntryName("KinematicHardening");
@AuxiliaryStateVariable real Evis;
Evis.setEntryName("Dissipated_Energy");
@AuxiliaryStateVariable real nb;
nb.setEntryName("NumberofNewtonIteration");
@LocalVariable Stensor sig_n;
@LocalVariable real  Fel;
@Import "PSAMaterialProperties.mfront"; //Gestion des propriétés matériaux PSA
/* Initialisation des variables locales */
@InitLocalVariables<Append> {
sig_n = sig;
StressStensor sigel = computeElasticPrediction();
sigel  -= 2*C*a/3;
Fel = sigmaeq(sigel) - R;}
/* Initialisation de l'incrément de la déformation élastique pour l'itération de NR */
@Predictor{
if (Fel < 0)
      deel=deto;
Else
      deel=Stensor(0.);}
/* Itération de Newton Raphson pour le calcul des variables d'états */
@Integrator {
if (Fel > 0) {
      constexpr const real eps = 1.e-12;
      const auto Ce = 2 * C / 3;
      const StrainStensor a_ = a + da;
      const StressStensor se = sig - Ce * a_;
      const auto seq = sigmaeq(se);
      const auto iseq = 1 / max(seq, eps * young);
      const Stensor n = 3 * deviator(se) * (iseq / 2);
      const auto F = max((seq - R) / K, stress(0));
      const auto iF = 1 / max(F, eps);
      const auto vp = pow(F, H);
/* Equations des fonctions résidus  */
      feel += dp * n;
      fp -= vp * dt;
      fa -= dp * (n - D1 * a_);
/* Calcul de la jacobienne */
      Stensor4 dn_dse = (Stensor4::M() - (n ^ n)) * iseq;
      const auto dvp_dseq = H * vp * iF / K;
      dfeel_ddeel += 2 * mu * dp * dn_dse;
      dfeel_ddp = n;
      dfp_ddeel = -2 * mu * dt * dvp_dseq * n;
      dfp_ddp = 1;
      dfeel_dda = -Ce * dp * dn_dse;
      dfp_dda = dvp_dseq * dt * Ce * n;
      dfa_ddeel = -2 * mu * dp * dn_dse;
      dfa_ddp = -n + D1 * a_;
      dfa_dda += (dp * D1) * Stensor4::Id() + Ce * dp * dn_dse;
} // fin de boucle if
} // fin de @Integrator
/* Actualisation des variables auxiliaires */
@UpdateAuxiliaryStateVariables {
nb = this->iter;
Evis += ((sig + sig_n) / 2) | (deto-deel);
}
```

Code MFront - Méthode d'intégration avec Jacobienne numérique

```
@Parser Implicit;
@Behaviour EVPunifieeECNLNortonVieillissementNumerique;
@Algorithm NewtonRaphson_NumericalJacobian ;
@PerturbationValueForNumericalJacobianComputation 1.e-9;
@Epsilon 1.e-14;
@ModellingHypotheses{".+"};
@Brick StandardElasticity;
@ElasticMaterialProperties{"YoungModulus.mfront", "PoissonRatio.mfront"};
@Theta 1.;
/*déclaration des variables */
@StateVariable strain p;
p.setGlossaryName("EquivalentPlasticStrain");
@StateVariable StrainStensor a;
a.setEntryName("KinematicHardening");
@AuxiliaryStateVariable real Evis;
Evis.setEntryName("Dissipated_Energy");
@AuxiliaryStateVariable real nb;
nb.setEntryName("NumberofNewtonIteration");
@LocalVariable Stensor sig_n;
@LocalVariable real  Fel;
@Import "PSAMaterialProperties.mfront"; //Gestion des propriétés matériaux PSA
/* Initialisation des variables locales*/
@InitLocalVariables<Append> {
sig_n = sig;
StressStensor sigel = computeElasticPrediction();
sigel  -= 2*C*a/3;
Fel = sigmaeq(sigel) - R;}
/* Initialisation de l'incrément de la déformation élastique pour l'itération de NR*/
@Predictor{
if (Fel < 0)
      deel=deto;
Else
      deel=Stensor(0.);}
/* Itération de Newton Raphson pour le calcul des variables d'états*/
@Integrator {
if (Fel > 0) {
      constexpr const real eps = 1.e-12;
      const auto Ce = 2 * C / 3;
      const StrainStensor a_ = a + da;
      const StressStensor se = sig - Ce * a_;
      const auto seq = sigmaeq(se);
      const auto iseq = 1 / max(seq, eps * young);
      const Stensor n = 3 * deviator(se) * (iseq / 2);
      const auto F = max((seq - R) / K, stress(0));
      const auto vp = pow(F, H);
/* Equations des fonctions résidus  */
      feel += dp * n;
      fp -= vp * dt;
      fa -= dp * (n - D1 * a_);
}  // fin de boucle if
}  // fin de @Integrator
/* Actualisation des variables auxiliaires*/
@UpdateAuxiliaryStateVariables {
nb = this->iter;
Evis += ((sig + sig_n) / 2) | (deto-deel);
}
```

Pas d'écriture de la matrice jacobienne

Algorithme avec système réduit – Principe d'intégration locale

Incrément d'écrouissage cinématique

$$\Delta \underline{\underline{a}} = \frac{\Delta p}{1 + D\Delta p} \underline{\underline{n}}_{t+\Delta t} - \frac{D\Delta p}{1 + D\Delta p} \underline{\underline{a}}_{t}$$

Incrément du multiplicateur viscoplastique

$$\Delta p = \langle \frac{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq|t+\Delta t} - R}{K} \rangle^{H} \Delta t$$

$$r_{\Delta p} = \Delta p - \langle \frac{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq|t+\Delta t} - R}{K} \rangle^{H} \Delta t = 0$$

$$\frac{\partial r_{\Delta p}}{\partial \Delta p} = 1 - \frac{H\Delta t}{K} \langle \frac{(\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq|t+\Delta t} - R}{K} \rangle^{H-1} \langle \frac{\partial (\underline{\underline{\sigma}} - \underline{\underline{X}})_{eq|t+\Delta t}}{\partial \Delta p} \rangle$$

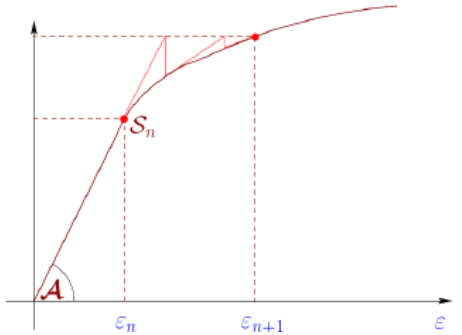
Incrément de déformation élastique

$$\Delta \underline{\underline{\varepsilon}}^{e} = \Delta \underline{\underline{\varepsilon}}^{to} - \Delta \underline{\underline{\varepsilon}}^{vp} = \Delta \underline{\underline{\varepsilon}} - \Delta p \underline{\underline{n}}_{t+\Delta t}$$

Opérateur tangent cohérent

$$\underline{\underline{\underline{\underline{L_c}}}} = \frac{\partial \Delta \underline{\underline{\sigma}}}{\partial \Delta \underline{\underline{\varepsilon}}}$$

$$\underline{\underline{\underline{\underline{L_c}}}} = \underline{\underline{\underline{\underline{E}}}} - \underline{\underline{\underline{\underline{E}}}} \ (\Delta p \frac{\partial \underline{\underline{n}}_{t+\Delta t}}{\partial \Delta \underline{\underline{\varepsilon}}} + \frac{\partial \Delta p}{\partial \Delta \underline{\underline{\varepsilon}}} \otimes \underline{\underline{n}}_{t+\Delta t})$$



Code MFront - Méthode d'intégration avec système réduit

```
@DSL ImplicitII;
@Behaviour EvpUnifieeEcnlNortonVieillissementReduit3d;
@Epsilon 1.e-14;
@Theta 1.;
@ElasticMaterialProperties{"YoungModulus.mfront", "PoissonRatio.mfront"};
/*déclaration des variables */
@AuxiliaryStateVariable StrainStensor eel;
eel.setGlossaryName("ElasticStrain");
@StateVariable real p ;
p.setEntryName("DeformationPlastiqueCumule");
@AuxiliaryStateVariable StrainStensor a;
a.setEntryName("KinematicHardening");
@AuxiliaryStateVariable real Evis;Evis.setEntryName("Dissipated_Energy");
@AuxiliaryStateVariable real
nb;nb.setEntryName("NumberofNewtonIteration");
@LocalVariable real Ce, C1dp, C2dp, Fel, F, vp;
@LocalVariable Stensor  sig_0, a_n, n_, B;
@Import "PSAMaterialProperties.mfront";
/* Initialisation des variables locales*/
@InitLocalVars<Append>{
sig_0 = sig;
a_n=a;
Ce = 2 * C / 3;
StressStensor sel =2*mu*(eel+deto);
sel  -= Ce*a_n;
Fel = sigmaeq(sel) - R;
}
/*Itération de Newton Raphson pour la détermination de p*/
@Integrator{
if(Fel>0){
        C1dp=dp/(1.+(D1*dp));
        C2dp=D1*C1dp;
        constexpr const real eps = 1.e-12;
        B = 2.*mu*deviator(eel+deto)-Ce*a*(1-C2dp);
        const auto C3dp=2.*mu*dp+Ce*C1dp;
        const auto seq =sqrt(3.*(B|B)/2.) - 3.*C3dp/2.;
        const auto iseq = 1. / max(2*seq/3.+C3dp, eps * young);
        n_= B*iseq;
        F = max((seq - R) / K, stress(0));
        vp = pow(F, H-1);
/* Equations du résidu */
        fp -= vp* F * dt;
/* dérivé de la fonction résidu */
        const auto dC3_dp = 2 * mu +       Ce/((1.+D1*dp)*(1.+D1*dp));
        const auto dB_ddp=Ce*D1*a/((1+D1*dp)*(1+D1*dp));
        const auto dNB_dB=B/sqrt(B|B);
        const auto dNB_dp=dNB_dB|dB_ddp;
        const auto dseq_ddp=sqrt(3./2.)*dNB_dp-3.*dC3_dp/2.;
        dfp_ddp  =1-(vp*dseq_ddp*dt*H/K);
}
}
/* Actualisation des variables auxiliaires*/
@UpdateAuxiliaryStateVariables{
nb = this->iter;
if(Fel>0){
        const auto deel=deto-dp*n_;
        const auto da = C1dp * n_ - C2dp * a;
        a+=da;
        eel+=deel;
        sig = lambda*trace(eel)*Stensor::Id()+2*mu*eel;
        Evis+=((sig+sig_0)/2.)|(deto-deel);  }
else {
        eel+=deto;
        sig = lambda*trace(eel)*Stensor::Id()+2*mu*eel;
}  }
/*Calcul de l'opérateur tangent*/
@TangentOperator{
if(Fel>0){
        Stensor4 De = lambda*(Stensor::Id()^Stensor::Id())+2.*mu*Stensor4::Id();
        real dC2_dp=D1/((1+D1*dp)*(1+D1*dp));
        real ddp_dseq=H*dt*vp/K;
        real dseq_dp=sqrt(3./2.)*((B/sqrt(B|B))|(Ce*a_n*dC2_dp))-
(3./2.*(2*mu+Ce/((1+D1*dp)*(1+D1*dp))));
        Stensor dseq_de=sqrt(3./2.)*2*mu*(Stensor4::Id()-
((1./3.)*(Stensor::Id()^Stensor::Id())))*(B/sqrt(B|B));
        Stensor dpp_de= ddp_dseq*dseq_de/(1-ddp_dseq*dseq_dp);
        Stensor4 dB_de=2*mu*(Stensor4::Id()-
(1./3.)*(Stensor::Id()^Stensor::Id()))+Ce*((dC2_dp*dpp_de)^a_n);
        Stensor4 dn_dB=sqrt(3./(2.*(B|B)))*(Stensor4::Id()-2*(n_^n_)/3.);
        Stensor4 dn_de=dn_dB*dB_de;
        Dt=De-De*((n_^dpp_de)+(dp*dn_de));    }//
else {
        computeAlteredElasticStiffness<hypothesis,Type>::exe(Dt,lambda,mu);
}
}
```

Uniquement pour éléments volumiques

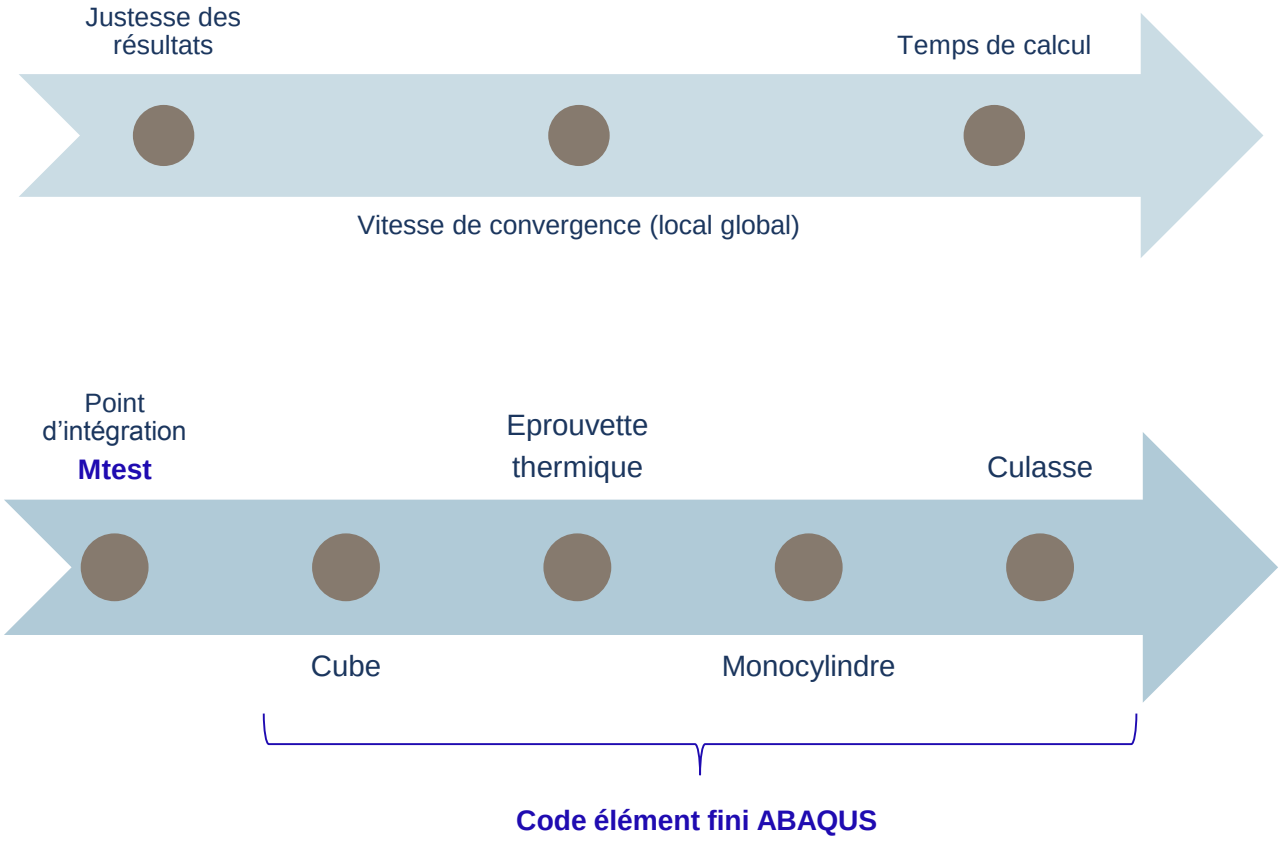
Comparaison des différentes méthodes (avant calculs)

	Méthode d'intégration avec Jacobienne analytique	Méthode d'intégration avec Jacobienne numérique	Méthode d'intégration avec réduction du système d'équation à une inconnue	**Umat Fortran**
	Equations inspirées des méthodes proposées dans Mfront sur des modèles similaires Calcul des termes de la Jacobienne Intégration dans Mfront facilitée par blocs prédéfinis • Opérateur tangent • Calcul des contraintes • Prediction élastique • Contraintes planes	Pas besoin de calculer analytiquement les termes de la Jacobienne Intégration dans Mfront facilitée : • Pas besoin d'écrire la Jacobienne • Opérateur tangent • Calcul des contraintes • Prediction élastique • Contraintes planes	Méthode d'intégration inspirée de [Szmytka 2007], mais améliorée dans un soucis de simplification Intégration dans Mfront complexe, pas de blocs prédéfinis car $\Delta\varepsilon_{ela}$ n'est pas une variable du problème Création d'un deuxième modèle spécifique aux contraintes planes	Méthode d'intégration inspirée de [Szmytka 2007],
Écriture des équations du problème	-	+++	--	--
Facilité d'intégration (code)	++	+++	-	--
Temps de développement d'un modèle	++	+++	+	--
Temps de calcul a priori	+	-	++	++

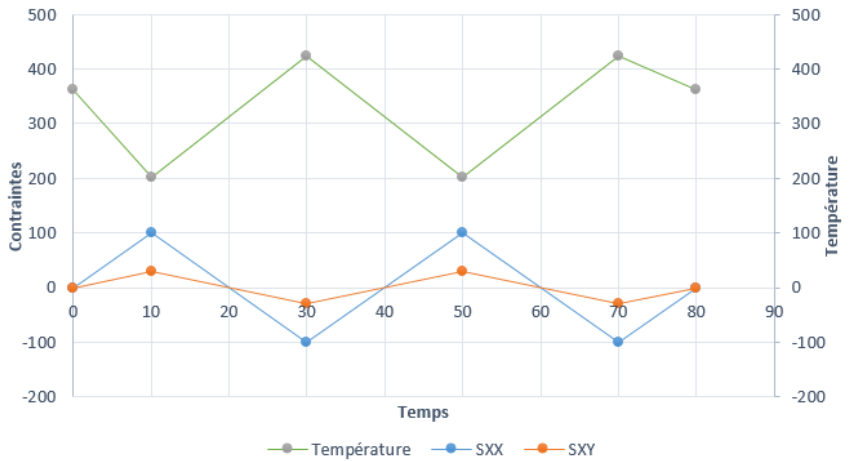
3. Résultats

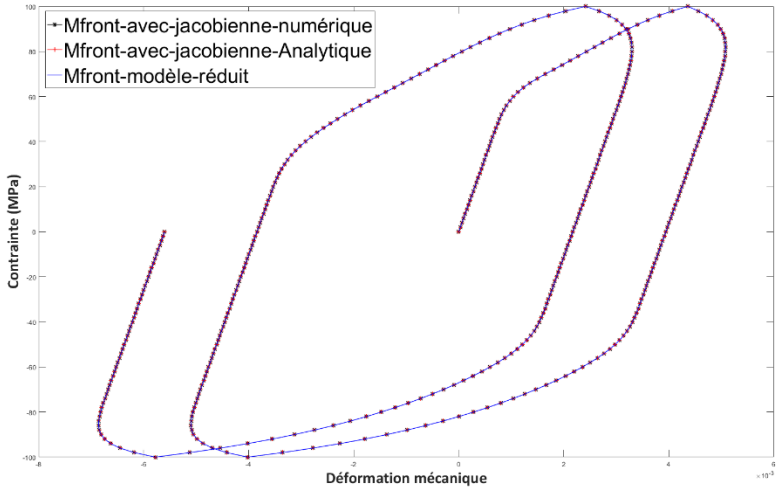
VALIDATION DU MODÈLE À DIFFÉRENTES ECHELLES

Principe de validation



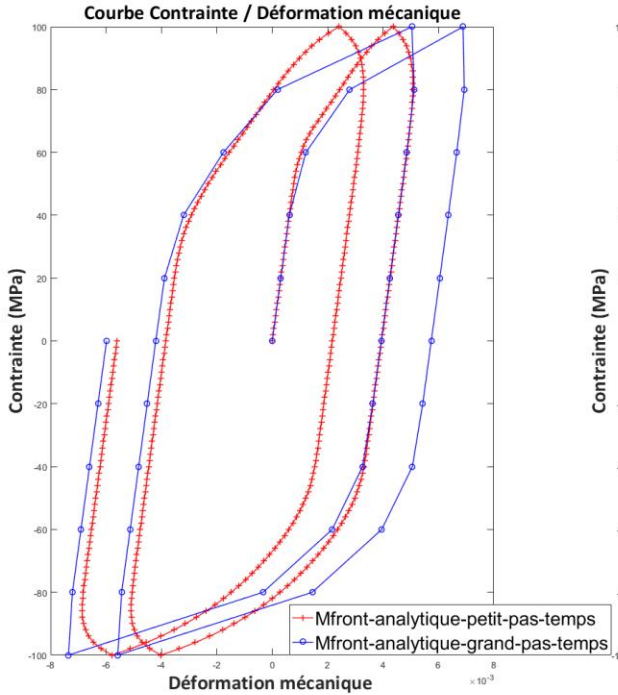
Définition d’un chargement cyclique anisotherme selon 2 directions



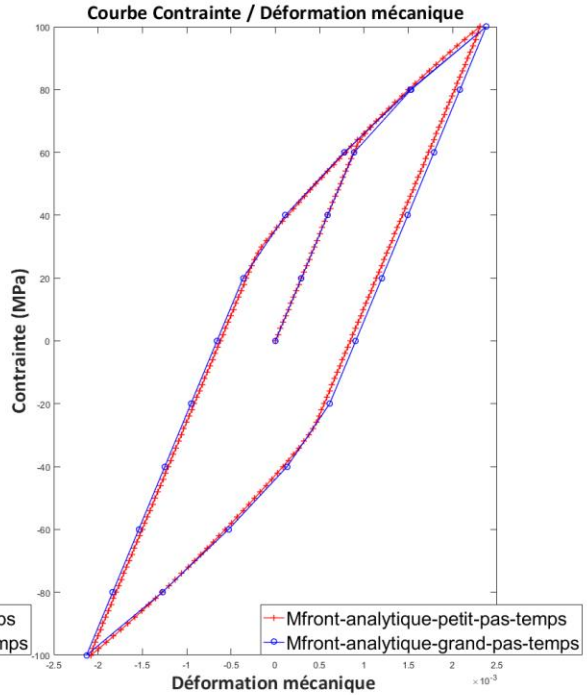


➡ Validation des méthodes intégrées dans MFront sur un point d’intégration

Mtest – Influence du pas de temps



Chargement anisotherme



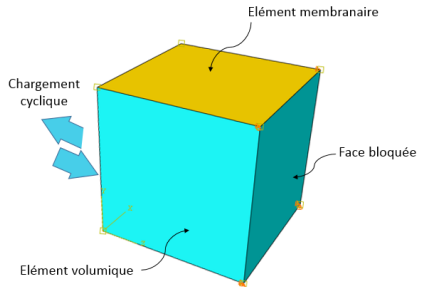
Chargement isotherme

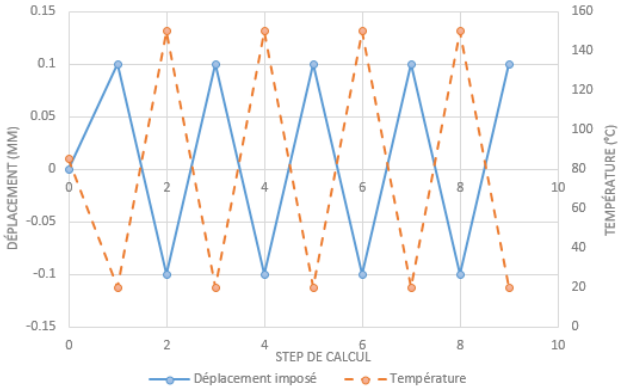
Schéma d'integration inconditionnellement stable, mais approximation du chargement

Conséquence :
Un découpage temporel différent implique un résultat différent

ABAQUS – Elément de volume + Elément membranaire

Calcul élasto-viscoplastique anisotherme

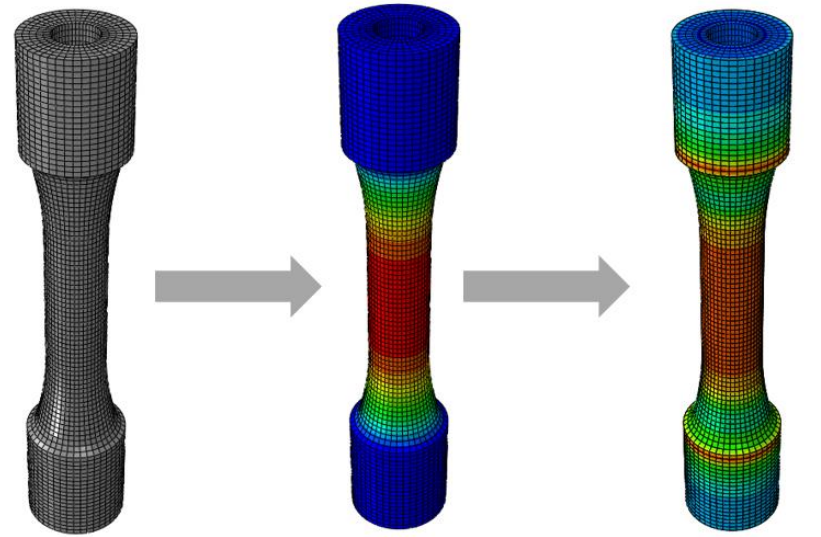




	Modèle avec Jacobienne analytique	Modèle avec Jacobienne numérique	Modèle réduit MFRONT	Modèle réduit Fortran
Pas de temps bridé				
Contrainte de Von Mises	183 MPa	183 MPa	183 MPa	183 MPa
Energie dissipée sur un cycle	9.248	9.248	9.248	9.248
Nombre d'incréments du calcul global	510			
Nombre d'itérations du calcul global	518	518	518	518
Pas de temps libre				
Contrainte de Von Mises	184 MPa	184 MPa	183 MPa	186 MPa
Energie dissipée sur un cycle	9.15	9.15	9.12	9.20
Nombre d'incréments du calcul global	88	88	102	88
Nombre d'itérations du calcul global	96	96	111	96

ABAQUS – Eprouvette de fatigue thermique

- Couplage faible
- Un cycle de chargement chauffage-refroidissement.

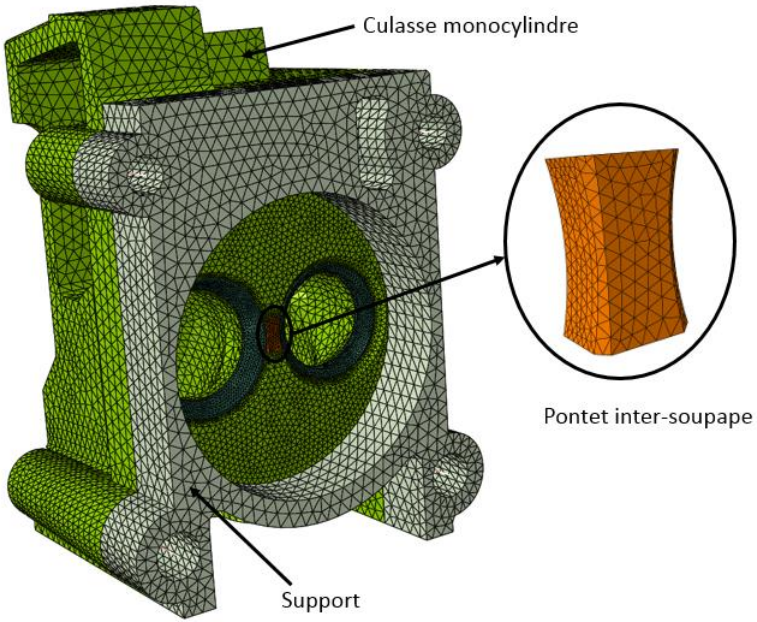


	Modèle avec Jacobienne analytique	Modèle avec Jacobienne numérique	Modèle réduit MFRONT	Modèle réduit Fortran
Pas de temps bridé				
Contrainte de Von Mises	33.46 MPa	33.46 MPa	33.46 MPa	33.48 MPa
Energie dissipée sur un cycle	1.288 10-2	1.288 10-2	1.288 10-2	1.296 10-2
Temps de calcul	1h 10mn	1h 16mn	1h 11mn	1h 29 min
Nombre d'incréments du calcul global	221			
Nombre d'itérations du calcul global	237	237	291	408
Pas de temps libre				
Contrainte de Von Mises	33.4 MPa	33.4 MPa	33.4 MPa	33.5 MPa
Energie dissipée sur un cycle	3.8 10-3	3.8 10-3	3.7 10-3	3.4 10-3
Temps de calcul	12min	13 min	13min	25mn
Nombre d'incréments du calcul global	26	26	27	31
Nombre d'itérations du calcul global	43	43	56	126

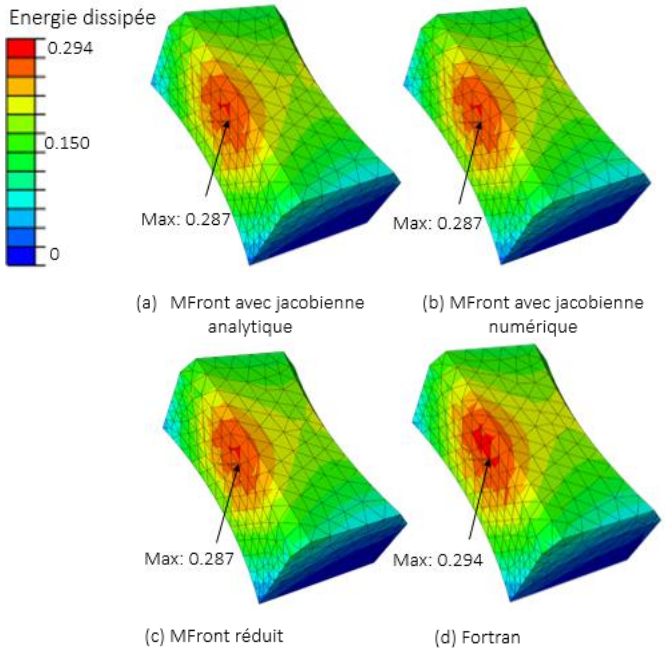
ABAQUS – Culasse monocylindre

Essai banc monocylindre :

- Comparaison des nuances matériaux
- Mise au point des méthodologie de dimensionnement



Energie dissipée par cycle



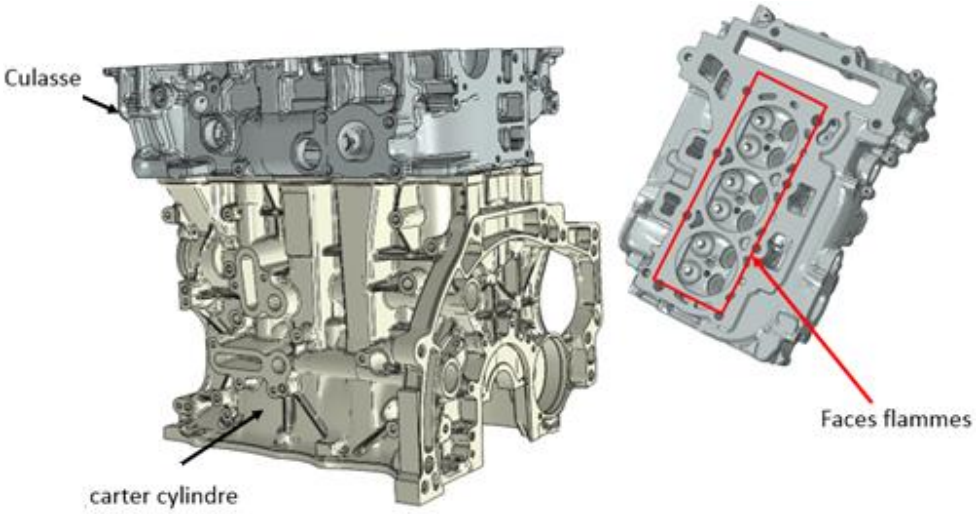
ABAQUS – Culasse monocylindre

	Modèle avec Jacobienne analytique	Modèle avec Jacobienne numérique	Modèle réduit MFRONT	Modèle réduit Fortran
Nombre d'incréments du calcul global	50	50	50	45
Nombre d'itérations du calcul global	297	297	293	270
Temps calcul	3:26	3:26	3:23	3:07
Frettage des sièges	0:09	0:09	0:09	0:09
Serrage des vis	0:31	0:31	0:27	0:06
Chauffage 1	0:24	0:24	0:24	0:26
Refroidissement 1	0:46	0:46	0:46	0:49
Chauffage 2	0:49	0:49	0:49	0:51
Refroidissement 2	0:44	0:44	0:45	0:44

- Résultats similaires : écarts liés au pas de temps
- Mfront plus sensible aux fortes non linéarités : réduction du pas de temps plus fréquente
- En fonction du serveur de calcul utilisé : variation des temps de calculs >15' (fortran)

ABAQUS – Culasse 3 cylindres

Culasse essence du groupe PSA
Peu de viscoplasticité



	Modèle avec Jacobienne analytique	Modèle avec Jacobienne numérique	Modèle réduit MFRONT	Modèle réduit Fortran
Contrainte de Von Mises	321 MPa	321 MPa	321 MPa	323 MPa
Energie dissipée sur un cycle	3.42.10-3	3.42.10-3	3.34.10-3	3.39.10-3
Temps de calcul	1j 6h 3mn	1j 6h 2mn	1j 5h 6mn	1j 6h 47mn
Nombre d'incréments du calcul global	150	150	146	145
Nombre d'itérations du calcul global	572	572	554	573

Bilan

- Conclusions de l'étude
  - Jacobienne numérique : très bons résultats → test de nombreux modèles à venir
  - Jacobienne analytique : très bons résultats → méthode stable, robuste sous Mfront et simple à intégrer (pour nos modèles)
  - Système réduit à une équation : complexe, pas de gain significatif…
  - Tous ces résultats sont sur le rapport de stage de Khouloud Derouiche disponible sur Researchgate
- Gains côté PSA
  - Plusieurs mois de travail en moins pour les futurs doctorants
  - Amélioration continue des algorithmes de résolution
  - Extrapolation aux modèles utilisés pour composites, plastiques…
- Côté Mfront, nos besoins :
  - Documentation plus complètes sur les différentes directives
  - Autres interfaces, en explicite notamment
  - Identification des paramètres
  - Gestion du pas de temps

Merci pour votre attention,

ET UN GRAND MERCI À THOMAS POUR SON AIDE !