

Implantation de lois de comportement mécanique à l'aide de MFronT : simplicité, efficacité, robustesse et portabilité

T. Helfer¹, J.M. Proix², O. Fandeur^{3,4}

¹ CEA, DEN/DEC, Département d'Études des Combustibles, thomas.helfer@cea.fr

² EDF R&D, Département Analyses Mécaniques et Acoustique, jean-michel.proix@edf.fr

³ CEA, DEN/DM2S, Département de Modélisation des Systèmes et des Structures, olivier.fandeur@cea.fr

⁴ IMSIA, UMR 8193, CNRS-EDF-CEA-ENSTA

Résumé — Ce papier est dédié au générateur de code open-source nommé MFronT [7]. Plusieurs langages dédiés (domain specific languages), basés sur le C++ [27], permettent d'écrire de manière simple, compacte, efficace et robuste, l'implantation de lois de comportement mécanique. Après une vue d'ensemble, nous décrivons certaines des techniques de programmation utilisées. Nous présentons ensuite divers algorithmes permettant d'accroître la robustesse des schémas d'intégration implicites. Enfin, nous discutons certaines questions liées à la portabilité des implantations entre différents codes.

1 Introduction

Le projet PLEIADES vise à fournir une plate-forme unifiée de développement des applications de simulation des éléments combustible nucléaires du CEA et d'EDF [23]. Au sein de ce projet, le générateur de code MFronT permet de traiter les aspects liés à l'écriture et la capitalisation des connaissances matériau dans un cadre d'assurance qualité strict. MFronT s'est ensuite révélé utile hors du domaine combustible, notamment pour les utilisateurs des codes Cast3M [6], Code_Aster [10] et AMITEX_FTT [5]. Une interface pour le code ZeBuLoN [2] est également disponible. MFronT a été mis en open-source en octobre 2014 dans l'espoir d'être utile aux ingénieurs et chercheurs de la communauté des matériaux et/ou du calcul de structure.

L'écriture de lois de comportement mécanique, par nature complexe, a un impact fort sur les temps de calcul et fait donc l'objet d'une attention particulière. MFronT propose plusieurs langages dédiés aux lois de comportement mécanique (appelés domain specific languages) sur la base du C++ : certains sont dédiés à des formalismes spécifiques (lois de viscoplasticité ou plasticité isotropes), d'autres aux schémas d'intégration en vitesse et les derniers aux schémas d'intégration implicites.

La première section de ce document propose une vue d'ensemble de MFronT.

La seconde met en avant une partie des techniques utilisées pour atteindre des performances numériques élevées. Nous concluons cette section par donner quelques éléments issus d'un benchmark comparant les lois générées via MFronT aux lois natives du Code_Aster.

La troisième section de ce document est dédiée à l'augmentation de la robustesse des algorithmes implicites. Nous y décrivons différents algorithmes introduits dans MFronT.

La quatrième section de ce document décrit certaines questions liées à la portabilité de l'implantation d'une loi de comportement. Nous considérerons les codes Cast3M, Code_Aster et ZeBuLoN [22]. Nous traiterons différents points liés aux stratégies de convergence utilisées par ces codes (choix de l'opérateur tangent) ou la gestion des contraintes planes.

2 Vue d'ensemble de MFronT

Cette section donne une vue d'ensemble de MFronT. Nous y décrivons un exemple très simple, puis nous présentons les différents algorithmes numériques disponibles avant d'aborder la notion d'interface.

```

@Parser IsotropicPlasticMisesFlow; //< domain specific language
@Behaviour Plasticity;           //< name of the behaviour
@Parameter H = 22e9;             //< hardening slope
@Parameter s0 = 200e6;           //< elasticity limit
@FlowRule{                       //< flow rule
    f = seq-H*p-s0;
    df_dseq = 1;
    df_dp = -H;
}

```

FIGURE 1 – Implantation d’une loi de comportement plastique avec un écrouissage isotrope.

2.1 Un premier exemple

La figure 1 décrit un fichier d’entrée utilisé par `MFront` pour implanter une loi de comportement plastique avec un écrouissage isotrope. Il s’agit d’un exemple d’un langage dédié à un type de loi particulier qui souligne qu’un des buts de `MFront` est de permettre une implantation de loi de comportement sous la forme la plus simple et la plus compacte possible.

Ce fichier est converti en sources C++ formant un total de plus de 1500 lignes qui gèrent :

- l’algorithme d’intégration, tel que décrit par SIMO et HUGUES (réduction à une équation scalaire) [26] ;
- le calcul de l’opérateur tangent cohérent [25] ;
- dans le cadre d’une adhérence au code `Cast3M` (voir la notion d’interface ci-dessous), le traitement des contraintes planes et/ou le passage en grandes transformations suivant différents formalismes dont les déformations logarithmiques [19]. En effet, ces points doivent actuellement être gérés au niveau de la loi de comportement dans `Cast3M` alors que les codes `Code_Aster` et `ZeBuLoN` proposent des solutions génériques en amont et/ou en aval de la loi de comportement. Ce point est discuté en section 5.

Les sources générées sont essentiellement des classes `template` paramétrées par l’hypothèse de modélisation et le type numérique utilisé (simple, double ou quadruple précision). Les techniques d’optimisation présentées plus loin (section 3) permettent d’obtenir, pour chaque hypothèse de modélisation, un code optimisable par le compilateur : il n’y a, par exemple, aucune allocation dynamique ni aucune boucle dans les opérations tensorielles.

2.2 Algorithmes numériques disponibles

`MFront` dispose de nombreux algorithmes d’intégration. On distingue deux classes d’algorithmes [3] :

- ceux basés sur un schéma d’intégration explicite de type Runge-Kutta ;
- ceux basés sur un schéma d’intégration implicite.

La question de la robustesse des algorithmes implicites est traitée en section 4.

2.3 Interfaces

La notion d’interface dans `MFront` permet de générer un code spécifique au code cible. Il existe aujourd’hui des interfaces pour les codes aux éléments finis `Cast3M` [6], `Code_Aster` [10] et `ZeBuLoN` [22] et l’application combustible `Cyrano3` [28]. L’interface au code aux éléments finis `Cast3M` a été reprise par le solveur FFT massivement parallèle `AMITEX_FTT` [5].

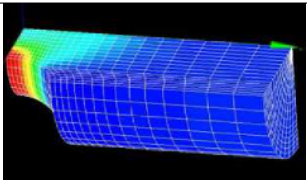
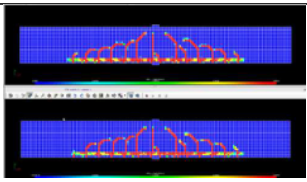
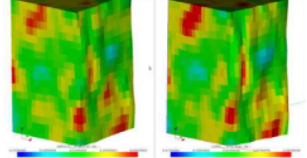
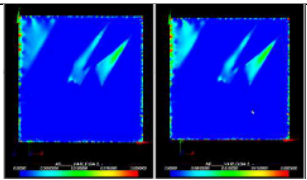
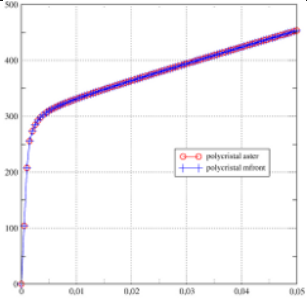
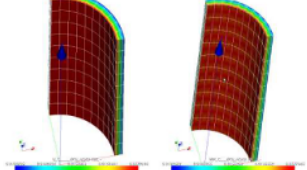
Si la notion d’interface permet de câbler les lois de comportement sur ces différents codes, ceux-ci présentent des spécificités qu’il faut prendre en compte pour les exploiter de façon efficace. Nous traitons ce point en section 5.

Le travail nécessaire à la création d’une nouvelle interface dépend fortement du code cible et des fonctionnalités prises en charge par celui-ci. L’interface la plus courte (environ 1 500 lignes de code) est aujourd’hui celle dédiée à `ZeBuLoN`, ce qui est dû au fait que de nombreuses fonctionnalités, laissées dans d’autres codes à la charge de la loi de comportement, sont traitées en amont ou en aval de l’intégration de la loi de comportement (contraintes planes, orthotropie, etc...).

3 Efficacité du code généré grâce aux techniques de programmation utilisées

Cette section s'intéresse à la question de l'efficacité du code généré. Un benchmark permet de montrer que malgré le caractère générique et multi-code, les performances obtenues sont satisfaisantes. La seconde partie décrit une partie des techniques utilisées pour obtenir ce résultat.

TABLE 1: Quelques exemples issus du benchmark comparant, dans Code_Aster, les performances du code généré par MFront aux lois natives de ce code.

Description	Algorithme	Temps CPU total (Aster vs MFront)	Illustration
Visco-plastic and damaging for steel [12, 21]	Implicit	17mn43s vs 7mn58s	
Damaging for concrete ([14, 17])	Default	45mn vs 63mn	
Generic Single crystal viscoplasticity ([11, 18])	Implicit	28mn vs 24mn	
FCC single crystal viscoplasticity ([11, 20])	Implicit	33m54s vs 29m30s	
FCC homogenized polycrystals 30 grains ([1, 11])	Runge-Kutta 4/5	9s67 vs 8s22	
Anisotropic creep with phase transformation ([13])	Implicit	180s vs 171s	

3.1 Benchmark

Le tableau 1 présente un extrait d'un benchmark réalisé par l'équipe de Code_Aster comparant les performances du code généré par MFront aux lois natives de ce code.

On remarque que les lois générées par MFront sont en général compétitives avec les implantations

natives¹. L'un des intérêts forts de MFront est le temps de développement de ces lois de l'ordre de la journée.

3.2 Template metaprogramming, expression templates et concepts

L'utilisation d'un générateur de code permet de masquer l'emploi de techniques de programmation avancées proposées par le langage C++, qui sont d'un accès difficile même pour des développeurs agueris. Elles sont cependant une des clés pour obtenir des implantations numériquement efficaces.

MFront repose essentiellement sur trois techniques :

- la programmation à base de template (Template metaprogramming) ;
- les expressions templates ;
- la notion de concept, implantée sur la base du standard C++ de 1998.

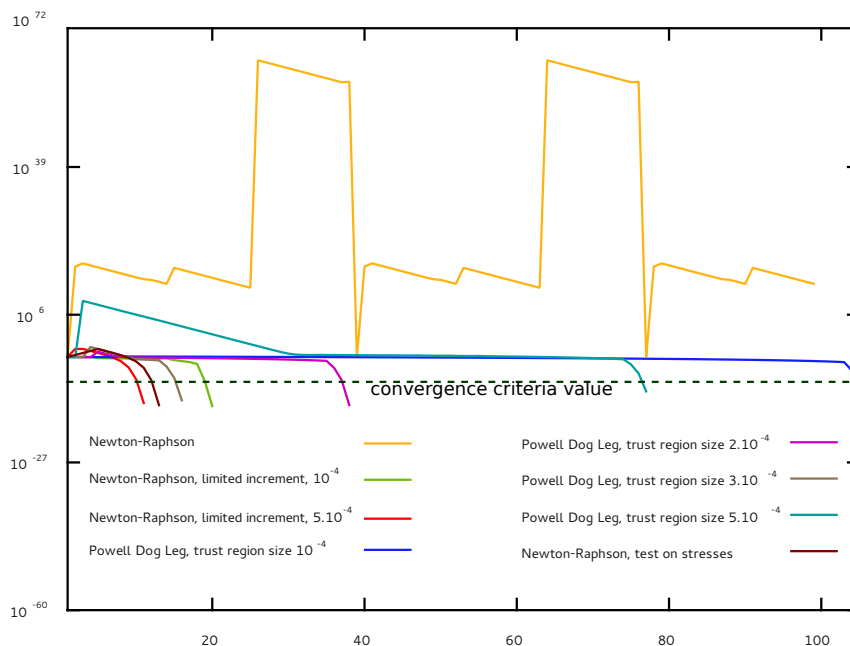


FIGURE 2 – Évolution du résidu en fonction du nombre d'itérations : échec de l'algorithme de NEWTON et emploi de stratégies alternatives.

4 Robustesse de l'intégration implicite

Le schéma d'intégration implicite est privilégié. Cependant, l'algorithme de NEWTON généralement utilisé pour résoudre le système non-linéaire qui en résulte peut manquer de robustesse, ce que nous avons illustré sur la figure 2 dans le cas d'une loi de comportement cristalline en grandes déformations.

L'utilisation de diverses modifications et stratégies alternatives à l'algorithme de NEWTON, notamment en réduisant la taille des corrections faites à chaque itération de l'algorithme, ou en considérant les algorithmes « en patte de chien » de POWELL ou l'algorithme de LEVENBERG-MARQUARDT [8, 24], est illustrée.

5 Sur l'écriture portable et efficace d'une loi comportement mécanique

Bien que la notion d'interface évoquée plus haut permette de se brancher à différents codes, ceux-ci présentent des particularités notamment liées aux algorithmes d'équilibre disponibles.

1. Parmi les cas présentés, nous voyons que l'utilisation de la loi native de Mazars est nettement plus efficace. Ceci est en partie dû au fait qu'il s'agit d'une loi explicite : le temps passé dans la loi est inférieur au temps nécessaire au branchement de la loi (l'appel à la loi native est plus « direct »). Un travail est en cours pour réduire ce temps de branchement.

Dans le cadre de ce papier, nous proposons un survol à (très) grosses mailles de certaines de ces particularités dans le cas des codes `Cast3M`, `Code_Aster` et `ZeBuLoN`.

`MFront` permet de tenir compte de ces particularités pour assurer une implantation à la fois portable et efficace d'une loi comportement mécanique de deux manières :

- soit en déléguant leurs implantations dans l'interface spécifique au code. C'est le cas par exemple des déformations logarithmiques ou d'une gestion générique (mais peu efficace) des contraintes planes dans l'interface `Cast3M`.
- soit en proposant d'écrire des blocs de code supplémentaires dans l'implantation de la loi de comportement. Ceci permettront de supporter des hypothèses de modélisation particulières (contraintes planes) ou de rajouter de nouvelles fonctionnalités (calcul de l'opérateur tangent ou d'un opérateur de prédiction).

Contraintes planes Dans `Cast3M`, les contraintes doivent être prises en charge par la loi de comportement. Si la loi est disponible en déformations planes généralisées, l'interface `Cast3M` mettra en place par défaut un algorithme générique traitant les contraintes planes. Cet algorithme est peu efficace et il vaut beaucoup mieux supporter explicitement cette hypothèse dans l'implantation de la loi [16].

Dans `Code_Aster`, l'approche la plus efficace est de prendre en charge les contraintes planes dans l'intégration de la loi de comportement. Si tel n'est pas le cas, un algorithme de convergence global spécifique, dû à DE BORST, peut être utilisé [15] : la loi est alors intégrée en 2D déformations planes généralisées.

Dans `ZeBuLoN`, les éléments finis possèdent un degré de liberté supplémentaire gérant la contrainte plane. La loi est alors intégrée en 2D déformations planes généralisées [4].

Opérateur de recherche L'un des aspects les plus intéressants de cette comparaison est le type d'opérateur de recherche utilisé par les différents codes pour trouver un incrément de déplacement satisfaisant l'équilibre mécanique en fin de pas.

`Code_Aster` et `ZeBuLoN` se basent préférentiellement sur l'algorithme de NEWTON [2, 9] et requiert que la loi fournisse un opérateur tangent cohérent [25]. Pour cette raison, l'intégration par une méthode implicite est préférée. L'opérateur tangent retourné par la loi doit être d'une grande qualité pour assurer la convergence quadratique de l'algorithme : toute erreur ou imprécision peut avoir des conséquences numériques importantes, voir désastreuses. En plus de l'opérateur tangent cohérent, `Code_Aster` peut également choisir d'utiliser d'autres d'opérateurs de recherche [9] : matrice élastique, opérateur sécant, opérateur tangent, etc.. *Dans certains cas, l'implantation du calcul de l'opérateur tangent peut représenter une part significative du temps d'implantation total.*

`Cast3M` utilise *par défaut* un algorithme original qui se base sur l'opérateur élastique² combiné à un algorithme d'accélération qui compense en (grande) partie la perte de la convergence quadratique [29]. Cette stratégie permet de n'effectuer qu'une unique factorisation de matrice (pour chaque pas de temps ou pour l'ensemble du calcul), évite le calcul et l'assemblage de la matrice tangente cohérente. On peut également remarquer que l'opérateur élastique est toujours défini positif, ce qui permet d'éviter certains écueils de l'algorithme de NEWTON en cas de comportement adoucissant³.

Dans tous les cas, la robustesse des algorithmes peut éventuellement être augmentée par différents techniques de type « line-search ».

Opérateur tangent en grandes transformations Le calcul de l'opérateur tangent en grandes transformations posent deux difficultés. La première est que chaque code cible requiert un opérateur différent. Par exemple, `Code_Aster` utilise la dérivée de la contrainte de Kirchhoff par rapport à l'incrément (spatial)

2. Notons que les propriétés élastiques doivent être données indépendamment de la loi de comportement. En pratique, on observe que la convergence de l'algorithme d'équilibre nécessite que la matrice de raideur élastique soit plus raide que la réponse effective de la loi de comportement.

3. Notons que cela n'est cependant pas suffisant en cas de réponse brutale de la structure, par exemple en cas de propagation instable d'une fissure. Dans ces cas, différentes stratégies, qui peuvent impacter la loi de comportement, sont également disponibles dans chaque code.

du gradient de la déformation \underline{F} :

$$\underline{\underline{D}}^{\text{tang}} = \frac{\partial \underline{\tau}|_{t+\Delta t}}{\partial \Delta \underline{F}} \quad \text{avec} \quad \Delta \underline{F} = \underline{F}|_{t+\Delta t} \cdot \underline{F}|_t^{-1}$$

où $\underline{F}|_t$ et $\underline{F}|_{t+\Delta t}$ désignent les gradients de la déformation respectivement en début et en fin de pas de temps ($\Delta \underline{F}$ est la transformation permettant de passer de la configuration en début de pas à la configuration en fin de pas).

La seconde difficulté est que chaque loi de comportement utilisera naturellement un tenseur de déformation et un tenseur de contrainte. Par exemple, la loi de Saint-Venant Kirchhoff relie le tenseur de Green-Lagrange $\underline{\varepsilon}_{GL}$ au second tenseur de Piola-Kirchhoff \underline{S} par une extension naturelle de la loi de Hooke :

$$\underline{S} = \lambda \text{tr} \underline{\varepsilon}_{GL} \underline{I} + 2\mu \underline{\varepsilon}_{GL} = \underline{\underline{D}} : \underline{\varepsilon}_{GL}$$

où λ et μ sont les premier et second coefficients de LAMÉ.

L'opérateur tangent « naturel », défini par la dérivée du tenseur de contrainte par rapport à l'incrément du tenseur de déformation, pour cette loi est probablement différent de l'opérateur attendu par le code. Dans le cas de la loi de Saint-Venant Kirchhoff, l'opérateur « naturel » est le tenseur d'élasticité $\underline{\underline{D}}$.

Nous avons introduit dans MFront la possibilité de fournir un ou plusieurs opérateurs tangents. On s'attend néanmoins à ce que l'utilisateur ne fournisse que l'opérateur tangent « naturel ». MFront essaiera alors de trouver « une chaîne de transformation », la plus « courte » possible⁴, pour arriver à l'opérateur attendu par le code. Dans le cas où le code est Code_Aster et où la loi relie $\underline{\varepsilon}_{GL}$ à \underline{S} , cette chaîne sera :

$$\frac{\partial \underline{S}|_{t+\Delta t}}{\partial \Delta \underline{\varepsilon}_{GL}} \Rightarrow \frac{\partial \underline{S}|_{t+\Delta t}}{\partial \underline{C}|_{t+\Delta t}} \Rightarrow \frac{\partial \underline{\tau}|_{t+\Delta t}}{\partial \underline{F}|_{t+\Delta t}} \Rightarrow \frac{\partial \underline{\tau}|_{t+\Delta t}}{\partial \Delta \underline{F}|_{t+\Delta t}}$$

où : – $\Delta \underline{\varepsilon}_{GL} = \underline{\varepsilon}_{GL}|_{t+\Delta t} - \underline{\varepsilon}_{GL}|_t$;
– \underline{C} est le tenseur de Cauchy droit.

Remerciements

Le développement de MFront a été soutenu par le projet PLEIADES, co-développé par le CEA, EDF et AREVA, et le projet Simu-Meca2015 d'EDF R&D. Les auteurs remercient Jacques BESSON et Stéphane QUILICI pour le temps qu'ils nous ont consacré lors de l'écriture de l'interface zmat dédiée au code aux éléments finis ZeBuLoN.

Références

- [1] M. BERVEILLER et A. ZAOUI. « An extension of the self-consistent scheme to plastically-flowing polycrystals ». Dans : *Journal of the Mechanics and Physics of Solids* 26.5 (oct. 1978), p. 325–344. ISSN : 0022-5096. DOI : 10.1016/0022-5096(78)90003-0. URL : <http://www.sciencedirect.com/science/article/pii/0022509678900030> (visité le 07/05/2014).
- [2] Jacques BESSON, Georges CAILLETAUD et Jean-Louis CHABOCHE. *Mécanique non linéaire des matériaux*. Paris : Hermès, 2001. ISBN : 2746202689 9782746202689.
- [3] J. BESSON et D. DESMORAT. « Numerical implementation of constitutive models ». Dans : *Local approach to fracture*. Sous la dir. de J. BESSON. École des Mines de Paris - les presses, 2004.
- [4] J. BESSON et R. FOERCH. « Large scale object-oriented finite element code design ». Dans : *Computer Methods in Applied Mechanics and Engineering* 142.1 (15 mar. 1997), p. 165–187. ISSN : 0045-7825. DOI : 10.1016/S0045-7825(96)01124-3. URL : <http://www.sciencedirect.com/science/article/pii/S0045782596011243> (visité le 29/12/2013).

4. Si plusieurs chaînes sont possibles, alors MFront choisira celle contenant le moins d'étapes, ce qui n'est pas nécessairement la plus efficace en termes de coût numérique.

- [5] CEA. *AMITEX_FFT*. 2014. URL : <http://www.maisondelasimulation.fr/projects/amitex/html/>.
- [6] CEA. *Cast3M Web Site*. 2014. URL : <http://www-cast3m.cea.fr/>.
- [7] CEA et EDF. *MFront Web Site*. 2014. URL : <http://www.tfel.sourceforge.net/>.
- [8] Hern-Shann CHEN et Mark A. STADTHERR. « A modification of Powell's dogleg method for solving systems of nonlinear equations ». Dans : *Computers & Chemical Engineering* 5.3 (1981), p. 143–150. ISSN : 0098-1354. DOI : 10.1016/0098-1354(81)85003-X.
- [9] EDF. *Algorithme non linéaire quasi-statique STAT_NON_LINE*. Référence du Code Aster R5.03.01 révision : 10290. EDF, 2013. URL : <http://www.code-aster.org>.
- [10] EDF. *Code_Aster Web Site*. 2013. URL : <http://www.code-aster.org>.
- [11] EDF. *Comportements élastoviscoplastiques mono et polycristallins*. Référence du Code Aster R5.03.11 révision : 10623. EDF, 2013. URL : <http://www.code-aster.org>.
- [12] EDF. *Comportement viscoplastique avec endommagement de Hayhurst*. Référence du Code Aster R5.03.13 révision : 8886. EDF, 2012. URL : <http://www.code-aster.org>.
- [13] EDF. *Modèle de comportement élasto-visqueux META_LEMA_ANI avec prise en compte de la métallurgie pour les tubes de gaine du crayon combustible*. Documentation du Code-Aster R4.04.05. EDF, 25 sept. 2013. URL : <http://www.code-aster.org>.
- [14] EDF. *Modèle d'endommagement de Mazars*. Référence du Code Aster R7.01.08 révision : 10461. EDF, 2013. URL : <http://www.code-aster.org>.
- [15] EDF. *Prise en compte de l'hypothèse des contraintes planes dans les comportements non linéaires*. Référence du Code Aster R5.03.03 révision 10101. EDF, 2012. URL : <http://www.code-aster.org>.
- [16] Thomas HELFER. *Prise en compte des hypothèses de contraintes planes dans les lois générées par le générateur de code MFront : application à Cyrano3*. Note technique 14-013. Disponible sur le site MFront. CEA, 2014.
- [17] J. MAZARS et François HAMON. « A new strategy to formulate a 3D damage model for concrete under monotonic, cyclic and severe loadings ». Dans : *Engineering Structures* (2015). À paraître.
- [18] L. MÉRIC et Georges CAILLETAUD. « Single crystal modelling for structural calculations ». Dans : *Journal of Engineering Material and Technology* 113 (1991), p. 171–182.
- [19] C. MIEHE, N. APEL et M. LAMBRECHT. « Anisotropic additive plasticity in the logarithmic strain space : modular kinematic formulation and implementation based on incremental minimization principles for standard materials ». Dans : *Computer Methods in Applied Mechanics and Engineering* 191.47 (22 nov. 2002), p. 5383–5425. ISSN : 0045-7825. DOI : 10.1016/S0045-7825(02)00438-3. URL : <http://www.sciencedirect.com/science/article/pii/S0045782502004383>.
- [20] G. MONNET, S. NAAMANE et B. DEVINCRE. « Orowan strengthening at low temperatures in bcc materials studied by dislocation dynamics simulations ». Dans : *Acta Materialia* 59.2 (2011), p. 451–461. ISSN : 1359-6454. DOI : 10.1016/j.actamat.2010.09.039. URL : <http://www.sciencedirect.com/science/article/pii/S1359645410006166> (visité le 07/05/2014).
- [21] R. MUSTATA et D. R. HAYHURST. « Creep constitutive equations for a 0.5Cr 0.5 Mo 0.25V ferritic steel in the temperature range 565°C - 675°C ». Dans : *International Journal of Pressure Vessels and Piping* 82.5 (mai 2005), p. 363–372. ISSN : 0308-0161. DOI : 10.1016/j.ijpvp.2004.11.002. URL : <http://www.sciencedirect.com/science/article/pii/S0308016105000037> (visité le 13/05/2014).
- [22] Inc. NORTHWEST NUMERICS {AND} MODELING. *ZeBuLoN*. 2014. URL : <http://www.zset-software.com/products/zebulon/>.
- [23] David PLANCQ et al. « PLEIADES : a unified environment for multi-dimensional fuel performance modeling ». Dans : *International meeting on LWR fuel performance*. Florida, 2004.

- [24] A. SHTERENLIKHT et N. A. ALEXANDER. « Levenberg–Marquardt vs Powell’s dogleg method for Gurson–Tvergaard–Needleman plasticity model ». Dans : *Computer Methods in Applied Mechanics and Engineering* 237–240 (1^{er} sept. 2012), p. 1–9. ISSN : 0045-7825. DOI : 10.1016/j.cma.2012.04.018. URL : <http://www.sciencedirect.com/science/article/pii/S0045782512001454> (visité le 02/06/2014).
- [25] J. C. SIMO et R. L. TAYLOR. « Consistent tangent operators for rate-independent elastoplasticity ». Dans : *Computer Methods in Applied Mechanics and Engineering* 48.1 (fév. 1985), p. 101–118. ISSN : 0045-7825. DOI : 10.1016/0045-7825(85)90070-2. URL : <http://www.sciencedirect.com/science/article/pii/0045782585900702> (visité le 02/04/2014).
- [26] Juan C SIMO et Thomas J. R HUGHES. *Computational inelasticity*. New York : Springer, 1998. ISBN : 0387975209 9780387975207.
- [27] Bjarne STROUSTRUP et Christine EBERHARDT. *Le langage C++*. Paris : Pearson Education, 2004. ISBN : 2744070033 9782744070037.
- [28] Gilles THOUVENIN et al. « EDF CYRANO3 code, recent innovations ». Dans : *LWR Fuel Performance Meeting/TopFuel/WRFPM*. Orlando, Florida, USA, sept. 2010.
- [29] Pierre VERPEAUX. « Algorithmes et méthodes ». Support de cours. 2014. URL : <http://www-cast3m.cea.fr/index.php?xml=supportcours>.