MANTA : un code HPC généraliste pour la simulation de problèmes complexes en mécanique

Olivier Jamond, Nicolas Lelong, Axel Fourmont, Joffrey Bluthé, Matthieu Breuze, Pascal Bouda, Guillaume Brooking, Florence Drui, Alexandre Epalle, Olivier Fandeur, et al.

CSMA 2022
15ème Colloque National en Calcul des Structures
16-20 Mai 2022, Presqu'île de Giens (Var)

MANTA : un code HPC généraliste pour la simulation de problèmes complexes en mécanique.

O. Jamond[1], N. Lelong[1], A. Fourmont[1]
J. Bluthé[1], M. Breuze[1], P. Bouda[1], G. Brooking[1], F. Drui[1], A. Epalle[2], O. Fandeur[1], G. Folzan[1], T. Helfer[2], F. Kloss[1], G. Latu[2], A. Motte[1,3], C. Nahed[1], A. Picard[1], R. Prat[2], I. Ramière[2], M. Steins[1], B. Prabel[1]

[1] *Université Paris-Saclay, CEA, Service d'Études Mécaniques et Thermiques, 91191, Gif-sur-Yvette, France*
[2] *CEA, DES/IRESNE/DEC/SESC, Service d'Études et de Simulation des Combustibles, Cadarache, France*
[3] *Onera — The French Aerospace Lab, F-92322, Châtillon, France*

Résumé — Le code MANTA a l'ambition de permettre la réalisation de simulations complexes en mécanique sur des supercalculateurs actuels et futurs tout en préservant les fondamentaux des codes développés au CEA : adaptabilité au problème posé, robustesse des algorithmes, pérennité des modèles et du code. On expose les principes de développement de ce code de nouvelle génération, et quelques exemples représentatifs de ses capacités actuelles sont également décrits.
Mots clés — code de calcul, mécanique des structures et des fluides, HPC, implicite, explicite, *toolbox*, éléments finis, volumes finis.

1 Contexte et objectifs généraux

Dans la course à l'exascale, les simulations numériques HPC en mécanique se cantonnent souvent à des "problèmes modèles" : linéaires, avec des modèles mécaniques, des chargements et conditions aux limites simples, permettant un traitement uniforme de l'ensemble des éléments de la discrétisation.

Cependant, la réalité des besoins industriels et de recherche nécessite de prendre en compte toute la complexité de la physique modélisée des systèmes, de manière à disposer de "jumeaux numériques" au juste niveau de représentativité souhaitée, conférant aux modélisations leurs caractères prédictifs et adaptées.

Ainsi, tout code de mécanique doit impérativement être :

- Multi-physique : structure, fluide, thermique, ... et leur interaction.
- Multi-formulation : éléments finis volumiques, de type coque et poutre, volumes finis pour les fluides, Galerkin discontinu, ...
- Multi-échelle en temps et en espace : intégration temporelle implicite et explicite, approches spectrales, adaptation automatique de maillage, ...
- Capable des gérer des physiques complexes : non-linéarités matériaux, géométriques, conditions de contact, interaction fluide-structure, ajout/enlèvement de matière, dégradation de type endommagement, rupture ou usure, ...
- Performant tant sur une architecture massivement parallèle que sur un pc portable.
- Adaptable à toute nouvelle problématique : nouvelle physique, formulation, discrétisation ou algorithme.
- Se doter d'une interface type *toolbox* accessible à l'utilisateur (python) et une interface plus avancée pour développeur (C++)

Les codes de calcul de mécanique actuels du CEA (EuroPleXus et Cast3M respectivement en explicite et en implicite) remplissent la plupart des critères énoncés ci-dessus, cependant une adaptation à l'évolution des architectures modernes nécessiterait un important effort technique. Dans cette optique, des *proto-applications* ont été développées au CEA ces dernières année, avec l'objectif de tester différentes approches pour adresser les difficultés liées au calcul haute performance de systèmes mécaniques complexes. Aujourd'hui, l'expérience acquise permet d'envisager la naissance d'un code de mécanique de nouvelle génération, *opensource*, nativement tourné vers l'HPC, suivant les normes les plus récentes

du C++, et avec un souci de pérennisation et de capitalisation de l'ensemble des travaux en mécaniques des structures du CEA et de ses partenaires tant en implicite qu'en explicite : **MANTA** (Mechanical Analysis Numerical Toolbox for advanced Applications).

2 Principes de développement

Le contexte nucléaire impose de disposer d'un code de référence, validé et d'une pérennité proche de celle des installations. Cela implique d'adopter une méthodologie de développement à long terme que nous choisissons de baser sur les caractéristiques suivantes :

- Modularité
- Factorisation
- Accessibilité

2.1 Modularité

Le code MANTA s'autorise l'usage *modulaire* de composants externes. La *modularité* réfère ici à la capacité du code à remplacer un composant par un autre pour une fonctionnalité donnée. Cette agilité permet d'assurer au code sa longévité, via un renouvellement aisé des composants externes, et une facilité à évoluer vers des outils plus récents, sans être pieds et poings liés à l'un d'entre eux.

Cela a deux conséquences sur la conception logicielle :

- Les composants et bibliothèques externes ne doivent être appelées que par le biais d'interface bien définies. Celles-ci ne doivent pas être diffuses, mais restreintes à quelques fichiers sources dédiés.
- Les fonctionnalités des composants externes doivent être utilisés de façon "atomique" (la plus élémentaire possible). Cela assure un remplacement potentiel aisé en cas de non-satisfaction. Par exemple, PETSc [1] offre un *framework* complet pour développer des logiciels de simulation en parallèle, mais le choix adopté dans MANTA est de ne pas utiliser ce cadre unique et global. Seules les fonctions nécessaires à la résolution distribuée de système linéaire sont directement appelées.

Aujourd'hui Manta est interfacé avec les librairies MOAB [2], PUMI [3], MFront [4], PETSc [1] entre autre. Nous souhaitons étendre cette ouverture aux librairies Kokkos [5], Trilinos [6], Alien. Le code assure également les entrées/sorties avec les formats de Cast3M [7], GMSH [8], VTK[3] et bientôt MedCoupling [9].

2.2 Factorisation

La *factorisation* consiste à contraindre toute succession de commandes constituant une partie de ou un algorithme à n'apparaître qu'une seule fois dans le code. Nous jugeons que développer avec ce souci permanent constitue une clé permettant d'assurer la maintenance et l'évolutivité à long terme de MANTA. In fine, cela se traduit par un objectif de ratio élevé entre le nombre de fonctionnalités et le nombre de lignes.

Ceci implique :

- La factorisation des méthodes numériques. Lors du développement de nouvelles fonctionnalités, un effort de généralisation est nécessaire afin d'extraire les méthodes élémentaires qui peuvent être généralisées pour être compatibles avec d'autres fonctionnalités, dans d'autres contextes. Ainsi, une méthode générique a été pensée pour assembler les contributions des maillages distribués au système global (sous forme de champs, matrices élémentaires,...). Elle intervient pour toutes les formulations et sera désigné *pipeline*, et décrite plus loin.
- La factorisation du code source. Ce point est lié aux bonnes pratiques de développement. Il vise à avoir le meilleur compromis entre la performance, la factorisation et la lisibilité du code.

2.3 Accessibilité

L'objectif est de rendre accessible le code à tout développeur, en y faisant abstraction des aspects HPC ou des subtilités du langage de programmation, afin de pouvoir y intégrer à moindre effort son modèle ou sa méthode numérique.

A cette fin, MANTA repose sur 2 couches :

- La couche coeur (*core layer*). Elle contient tous les processus de parallélisation lié aux manipulations d'objets, de manière à ce que toutes les méthodes numériques s'appuyant dessus puissent en profiter et tirer parti des performances lié au parallélisme. Y est donc notamment défini le *pipeline* générique mentionné plus haut. On trouve également dans la couche coeur de nombreux autres *services* facilitant l'implémentations de méthodes numériques : manipulation de maillage, algèbre dense, champs, fonctions de forme, outils géométriques, gestion des configurations pour les méthodes Lagrangienne, ... Ces fonctions sont indépendantes des modèles physiques utilisés.
- La couche modélisation (*modelization layer*). Y sont développées les méthodes numériques propres aux modélisations en s'appuyant sur les fonctions de la couche coeur. Par exemple, la définition des éléments finis ou des volumes finis, les schémas d'intégration temporelle explicite ou implicite, ou encore un modèle de fluide compressible. Le développement est réalisé en faisant abstraction de l'aspect parallèle.

La figure 1 représente une vue simplifiée de l'architecture logicielle de MANTA. En italique apparaissent les fonctions prévues mais non implémentées à date.

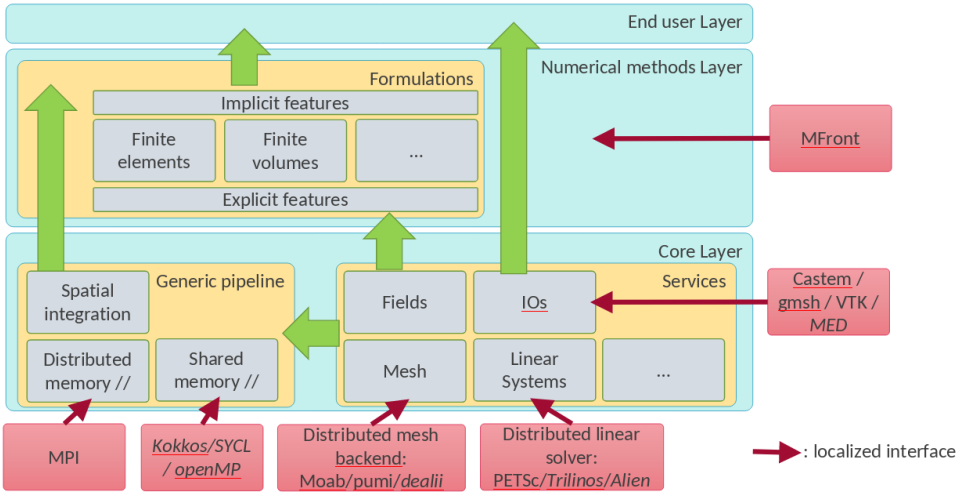

FIGURE 1 – MANTA simplified architecture

2.4 Aperçu du *pipeline*

Un des critères de developpement de MANTA est la recherche de scalabilité mémoire et l'équilibrage des processus MPI.

Ainsi, le rôle du *pipeline* est essentiel et consiste principalement à :

- assembler des matrices creuses et des seconds membres par l'adjonction de contributions élémentaires d'entités "denses" issues généralement de quantités intégrées par éléments,
- et résoudre les systèmes linéaires ainsi formés.

L'assemblage peut se mettre sous la forme mathématique :

$$\boldsymbol{M} = \sum_i \mathcal{A}_i \int_{E_i} \boldsymbol{m}(\underline{x})\, d\underline{x} \tag{1}$$

où E_i désigne les supports d'intégration élémentaires (i. e. généralement les éléments) de la fonction $\boldsymbol{m}$. $\mathcal{A}_i$ est l'opérateur d'assemblage, qui est l'application associant un degré de liberté local (i.e un indice de ligne et de colonne) à un degré de liberté global défini de manière unique dans $\boldsymbol{M}$.

En pratique et pour les éléments finis notamment, l'intégrale est approximée par une règle de quadrature établie dans un élément de référence, conduisant à l'expression :

$$\int_{E_i} \boldsymbol{m}(\underline{x})\, d\underline{x} \approx \sum_j w_j \boldsymbol{m}(\underline{\xi}_j) |\det(\underline{\phi}_i(\underline{\xi}_j))| \tag{2}$$

avec $\underline{\xi}_j$ et w_j respectivement coordonnées et poids du $j$ème point d'intégration dans l'élément de référence, et $\underline{\phi}_i : \underline{\xi} \to \underline{x} \in E_i$ fonction de passage entre l'espace de référence et l'espace réel de l'élément E_i.

Le *pipeline* réalise cette intégration de manière générique (une seule implémentation pour toutes les formulations et méthodes). Chaque couple (modèle, méthode numérique) doit définir son propre opérateur $\mathcal{A}$ et la fonction $\boldsymbol{m}$ dans la couche *modélisation*.

3 Applications illustrant quelques unes des fonctionnalités actuelles

MANTA supporte aujourd'hui les fonctionnalités suivantes pour les calculs de structure :

- Eléments finis linéaires et quadratiques
- Éléments volumiques et coque (MITC4)
- Grandes transformations
- Lois de comportement complexes (mfront)
- Problèmes implicites/explicites
- Contact glissant
- Conditions aux limites "complexes"
- "Phasefield" (explicite/implicite)
- HDG (Hybrid Discontinuous Galerkin)

et les fonctionnalités suivantes pour les calculs fluide et d'Interaction Fluide/Structure :

- Volumes-finis ordre 2
- Problèmes implicites/explicites
- Equations d'Euler
- Ecoulement multicomposants (modèle 5 équations)
- IFS frontières immergées (Mediating Body Method)
- Méthode Chimère

Les items suivant donnent une aperçue du potentiel de calculs réalisés avec MANTA.

3.1 Modélisation d'une enceinte réacteur

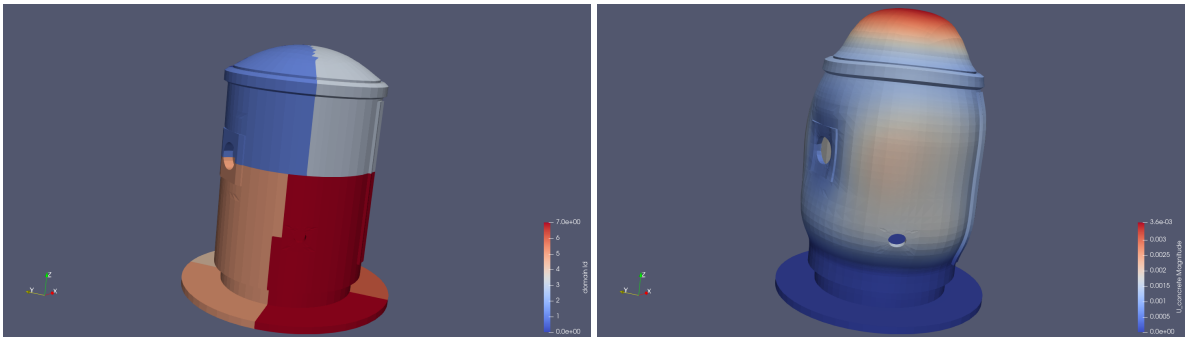

FIGURE 2 – Enceinte de confinement, maquette VERCORS - partitionnement et visualisation de la déformation à une pression interne de 4.10^5 Pa et soumis à son poids propre (échelle $\times 1000$)

- Elasticité linéaire et éléments finis de type barre (support multi-matériau)
- relations cinématiques acier-béton, liaisons parfaites
- $1.8\text{x}10^5$ cellules béton et $8\text{x}10^4$ éléments acier - taille du système (309741×309741)
- 8 sous-domaines MPI, calcul réalisé sur laptop (Intel Core i7-10875H)

3.2 Impact sur éprouvette Charpy

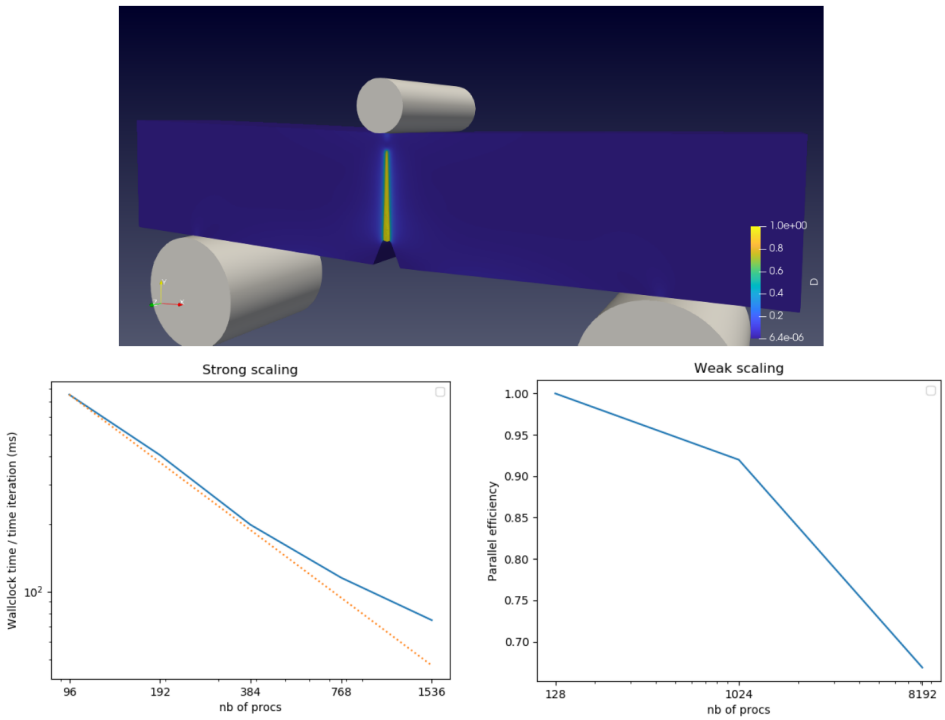

FIGURE 3 – Fracturation d'une éprouvette Charpy par champ de phase - Visualisation de l'endommagement

- Calcul éléments-finis explicite avec endommagement de type "phasefield"
- Contact glissant entre l'éprouvette, l'impacteur et les supports
- 67x10^6 cellules
- 8192 sous-domaines MPI, AMD-ROME

3.3 Explosion

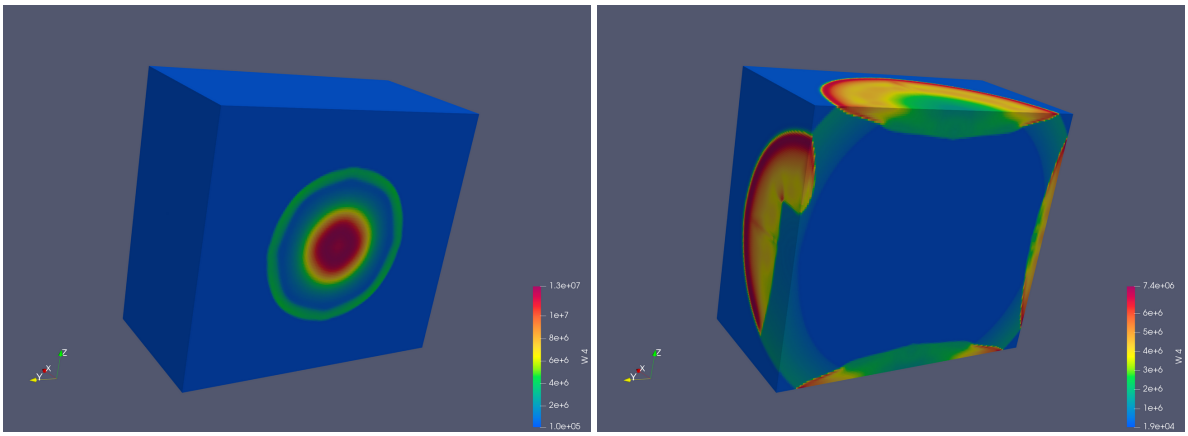

FIGURE 4 – Explosion d'une charge de TNT dans un bunker - Visualisation de la propagation de l'onde de pression

- Calcul Volumes Finis mono- ou multi-constituants
- Schéma explicite d'ordre 2 en temps et espace : MUSCL-Hancock
- 150x10^6 cellules
- 8192 sous-domaines MPI, Intel-Skylake

3.4 Mise en forme par emboutissage

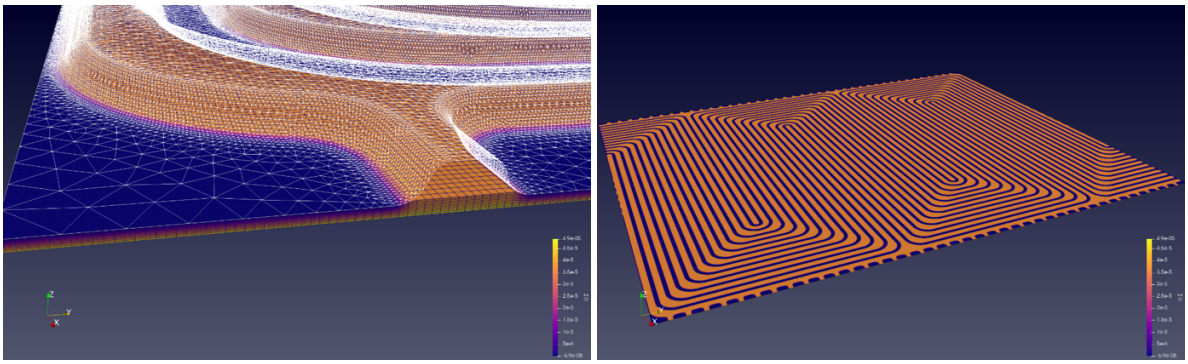

FIGURE 5 – Essais de mise en forme d'une plaque, pile à combustible (la surface poinçon apparaît en blanc à gauche)

- Calcul explicite
- Contact glissant
- 84x10^6 degrés de liberté, 24x10^6 cellules
- Surface « poinçon » : 12x10^6 cellules
- 1000 sous-domaines MPI, Intel-Skylake

4 Perspectives

De nombreux développements sont en cours et visent à intégrer rapidement :

- le raffinement hiérarchique adaptatif (AMR) et des critères de raffinement,
- la modélisation du contact frottant,
- la modélisation de propagation de fissure (érosion d'éléments, déboutonnage, champs de phase, remaillage, ...)
- des approches "micromorphes" pour l'endommagement des structures,
- des modèles fluides plus représentatifs de la physique des explosions,
- de l'"anti-hourglass" pour permettre l'utilisation d'éléments sous-intégrés,
- une API C++ et python.

Coté performance, le traitement efficace de tels problèmes « complexes » faisant intervenir plusieurs dizaines de milliards de degrés de libertés sur des centaines de milliers de coeurs de calcul des futures machines exascale représente un challenge ambitieux que nous souhaitons relever via le code MANTA d'ici 2030. Les 3 axes suivants seront abordés :

- Gestion des conditions aux limites complexes lors de la résolution des systèmes linéaires distribués. Des méthodes spécifiques devront être développées et mises en œuvre pour traiter efficacement les grands systèmes creux avec des solveurs linéaires itératifs. Des travaux sur la définition de préconditionneurs efficaces pour ce type de problème seront également à mener.
- Equilibrage de charge dynamique pour le parallélisme à mémoire distribué. Cette axe consiste à réviser régulièrement au cours du calcul la décomposition de domaine afin de garantir une bonne répartition de charge entre les unités de calcul. Une des principales difficultés pour les calculs visés réside dans le fait que la décomposition optimale en sous-domaines peut être différente pour chacune des étapes de calcul lors de la résolution d'une sous-étape temporelle. Des stratégies permettant de traiter cette difficulté seront à mettre en place. L'AMR (Adaptive Mesh Refinement) rend aussi l'équilibrage dynamique particulièrement difficile sur les très grandes plate-formes de calcul.
- Portabilité des performances, en particulier sur GPU. Afin d'aborder cette question des GPUs dans le cadre plus général de la portabilité des performances, nous envisageons d'utiliser des outils tels que Kokkos [5] ou DPC++/SYCL [10]. Un important travail d'adaptation et d'optimisation de l'architecture logicielle et des structures de données de la couche « core » de MANTA sera nécessaire pour tirer un maximum de performance sur des architectures GPUs. La question

du traitement des lois de comportement sur GPU sera un point central de cet axe car il s'agit d'un centre de coûts important en terme de temps d'exécution - la bibliothèque MFront [1] est utilisée pour le traitement de ces lois.

Afin d'illustrer les capacités de MANTA et de répondre aux besoins de ses futurs utilisateurs, les développements évoqués s'accompagneront de la mise oeuvre de calculs cibles dans des domaines d'applications variées (ex : calculs thermomécaniques de composants nucléaire divers, vibrations en interaction avec les fluides, déformation d'assemblages combustibles, usure, explosion, réponse au séisme avec ISS, modélisation multi-échelle de composites, ...).

Références

[1] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.16, Argonne National Laboratory, 2021.

[2] T. J. Tautges, R. Meyers, K. Merkley, C. Stimpson, and C. Ernst. MOAB : a mesh-oriented database. SAND2004-1592, Sandia National Laboratories, April 2004. Report.

[3] Daniel A. Ibanez, E. Seegyoung Seol, Cameron W. Smith, and Mark S. Shephard. Pumi : Parallel unstructured mesh infrastructure. *ACM Trans. Math. Softw.*, 42(3), may 2016.

[4] Thomas Helfer, Bruno Michel, Jean-Michel Proix, Maxime Salvo, Jérôme Sercombe, and Michel Casella. Introducing the open-source mfront code generator : Application to mechanical behaviours and material knowledge management within the pleiades fuel element modelling platform. *Computers & Mathematics with Applications*, 70(5) :994–1023, 2015.

[5] Christian R. Trott, Damien Lebrun-Grandié, Daniel Arndt, Jan Ciesko, Vinh Dang, Nathan Ellingwood, Rahulkumar Gayatri, Evan Harvey, Daisy S. Hollman, Dan Ibanez, Nevin Liber, Jonathan Madsen, Jeff Miles, David Poliakoff, Amy Powell, Sivasankaran Rajamanickam, Mikael Simberg, Dan Sunderland, Bruno Turcksin, and Jeremiah Wilke. Kokkos 3 : Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, 33(4) :805–817, 2022.

[6] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[7] CEA. Cast3m. http ://www-cast3m.cea.fr, 2021. Code de calcul pour l'analyse de structures par la méthode des éléments finis et la modélisation en mécanique des fluides.

[8] Christophe Geuzaine and Jean-François Remacle. Gmsh : A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11) :1309–1331, 2009.

[9] CEA, EDF, and Opencascade. Salome. https ://www.salome-platform.org/, 2021. The open source integration platform for numerical simulation.

[10] Khronos group. Sycl. https ://www.khronos.org/sycl/, 2021. The open source integration platform for numerical simulation.

1. MFRONT : voir "New functionalities of Versions 3.3, 3.4 and 4.0 of the TFEL/MFront project and Version 1.0, 1.1 and 1.2 of the MGIS project", par T. Helfer *et al.*, à cette conférence.