Une interface entre un code de calcul de laboratoire Herezh++ et le logiciel Abaqus

Hervé Laurent, Gérard Rio, Guilhem Bles

Une interface entre un code de calcul de laboratoire Herezh++ et le logiciel Abaqus

Application à un calcul industriel de type élasto-visco-hystérétique

H. Laurent* — **G. Rio*** — **G. Blès****

**Laboratoire Génie Mécanique et Matériaux, Université de Bretagne Sud, UPRES-EA 2595, 56321 LORIENT cedex, FRANCE*

herve.laurent@univ-ubs.fr

***Laboratoire Mécanique des Structures Navales, ENSIETA, 2 rue François Verny 29806 BREST cedex 09, FRANCE*

RÉSUMÉ. *Le logiciel par éléments finis ABAQUS propose une interface utilisateur, appelée Umat, qui donne accès à différentes variables globales de calcul pour le développement de nouvelles lois de comportement. Nous avons profité de cette possibilité pour lier le code ABAQUS et un code de calcul en C++, appelé HEREZH++. Dans ce papier, nous décrivons dans un premier temps le logiciel HEREZH++ puis l'architecture de cette interface de simulation. Quelques tests numériques sont ensuite présentés ainsi qu'un calcul sur une pièce industrielle avec une loi de comportement originale d'élasto-visco-hystérétique.*

ABSTRACT. *The commercial software ABAQUS offers an User-defined Material, named Umat, to introduce of a new constitutive model. We use these capabilities to link the code ABAQUS and an object-oriented FE C++ software HEREZH++. This paper describes the architecture of the simulation interface. The code coupling is made with a named piped interprocess communication method and an interface written in c/C++. Several test samples are used to verify and demonstrate the feasibility of the proposed approach. In particular, an industrial test is carried out with an original behavior model of elasto-visco-hysteresis.*

MOTS-CLÉS : *Interface entre code EF, Umat, ABAQUS, HEREZH++, programmation orienté objet*

KEYWORDS: *Code coupling, User Material interface (Umat), ABAQUS, HEREZH++, Object-oriented programming*

1. Le code Élément Finis *HEREZH++*

HEREZH++ est un code de calcul de laboratoire développé en langage orienté-objet *C++* par G. Rio, au sein du Laboratoire Génie Mécanique et Matériaux (http : //web.univ ubs.fr/lg2m/rio, 2005). Son objectif est d'être suffisamment flexible pour pouvoir intégrer facilement de nouveaux concepts numériques dans le cadre de la Méthode des Éléments Finis pour des problèmes mécaniques en grandes transformations. *HEREZH++* tire parti des potentialités offertes par le langage orienté objet *C++* et plus particulièrement de la gestion de la mémoire dynamique. Les notions d'encapsulage de données, de "template", de polymorphisme statique et dynamique comme la surcharge d'opérateur pour les notions de classes de vecteurs, tenseurs et matrices, sont largement utilisées dans le code pour définir des classes virtuelles comme les types d'éléments finis, les lois de comportement, les algorithmes de résolution, etc. Les "Standard Template Library (STL)" sont également employées pour définir les variables de listes, de tableaux variables, de containers associés ainsi que les algorithmes de bases associés tels que les notions de classement, rangement et suppression de redondance. Ces différentes notions, classiques en langage objet, assurent un "débuggage" très efficace des erreurs et une gestion du code très simplifié et rapide, ce qui n'est pas toujours le cas lors de l'implémentation de méthodes numériques dans un code de calcul commercial.

2. Fonctionnement de l'interface entre *ABAQUS* et *HEREZH++*

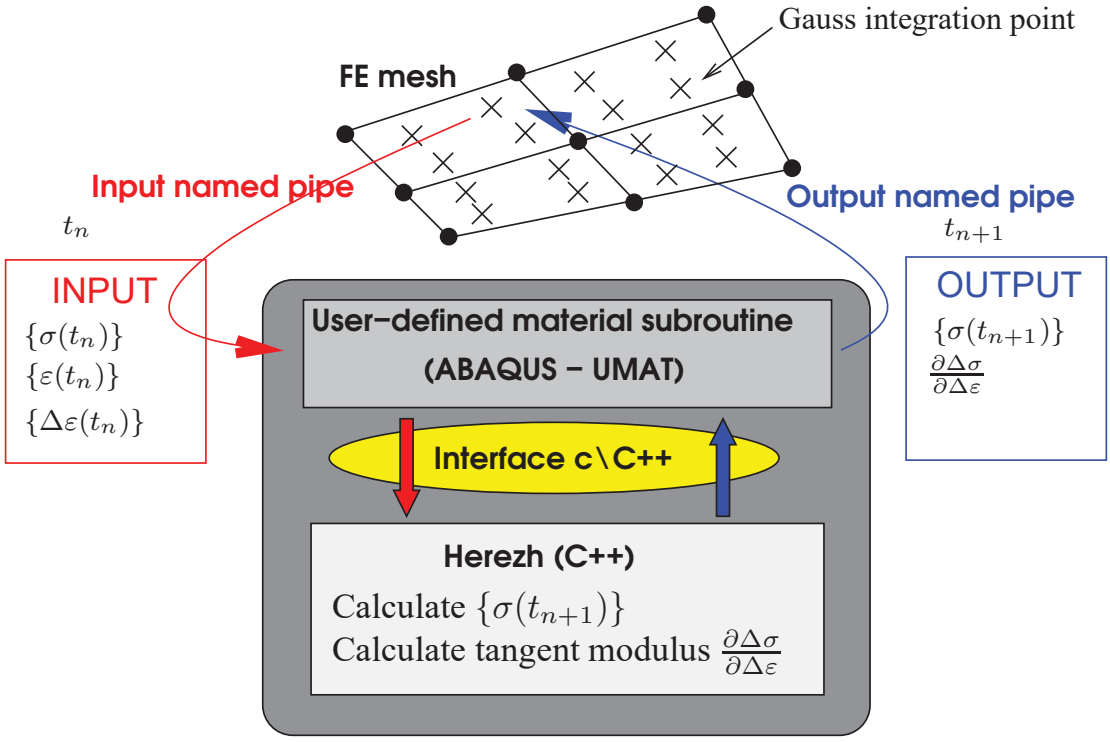

Figure 1. *Passerelle entre les codes ABAQUS et HEREZH++ par l'intermédiaire de la subroutine Umat*

La figure (1) donne un schéma de principe du fonctionnement de la passerelle entre *ABAQUS* et *HEREZH++*. Le couplage asynchrone entre les deux processus est effectué par l'intermédiaire de l'interface *Umat*, à l'aide d'une méthode de communication inter-processus par "tubes nommés" et d'une interface écrite en c/C++.

Pour chaque incrément d'équilibre t_n de la procédure de calcul non-linéaire d'*ABAQUS* et pour chaque point d'intégration de Gauss du maillage, *ABAQUS* fournit, par l'intermédiaire de l'*Umat*, le tenseur des contraintes $\{\sigma(t_n)\}$, le tenseur des déformations $\{\varepsilon(t_n)\}$ et une estimation de l'incrément de déformation $\{\Delta\varepsilon(t_n)\}$. Le code *HEREZH++* calcule le tenseur des contraintes $\{\sigma(t_{n+1})\}$ et la matrice tangente $\frac{\partial \Delta\boldsymbol{\sigma}}{\partial \Delta\boldsymbol{\varepsilon}}$ de la loi de comportement. Ces données sont ensuite transférées vers *ABAQUS* via l'interface et la *Umat*.

L'interface doit permettre un passage d'informations rapide entre les deux logiciels, nous n'envisageons donc que des dialogues par l'intermédiaire de la mémoire vive et non via une interface par fichiers. Dans notre cas, nous avons retenu un dialogue par tubes nommés pour les raisons suivantes : les données de chaque processus sont ainsi parfaitement encapsulées et les ordres de lecture et d'écriture permettent de synchroniser aisément deux processus initialement asynchrones.

Différentes structures ont été ensuite implémentées. Tout d'abord, du coté du logiciel *ABAQUS*, une subroutine *Umat* est mise en place : son rôle est de transférer les variables de passage à une routine c/C++, uniquement sous forme de pointeurs. Le dialogue par tubes nommés est alors piloté par la routine c/C++ (voir figure 1). Un "input named pipe" est utilisé pour transférer les informations entre la *Umat* à *HEREZH++* et un "output named pipe" dans l'autre sens. Les informations transmises par pipe se résument à un tableau d'octets. Il est évidemment hors de question de transmettre les données éléments par éléments, il nous faut donc sérialiser les paramètres de passage via un tableau global d'octets. Comme les paramètres de passage provenant de la subroutine Fortran ne sont pas contiguë en mémoire vive, il y a une recopie de ces paramètres dans le tableau d'octets via une structure c/C++ "union" qui établit une équivalence entre un tableau de réels, un tableau d'entiers et le tableau (ou buffer) d'octets (ou de caractères) transmis par tubes.

Enfin, *HEREZH++* doit stocker les grandeurs relatives à la loi de comportement, ceci pour tous les points d'intégrations de tous les éléments. Or en début de calcul, *HEREZH++* ne connaît ni le nombre d'éléments, ni le nombre de points d'intégrations. La solution retenue pour résoudre cette difficulté a été de créer dans *HEREZH++* une classe particulière d'éléments finis prévu pour stocker tous les points d'intégrations d'un même élément. Le support géométrique en est le point. A la première itération du premier incrément, ces éléments sont construits dynamiquement, ainsi que les conteneurs relatifs aux points d'intégrations associés à l'élément. Au changement d'itération, l'ensemble des éléments est globalisé dans un maillage créé à cette étape. L'intérêt de cette méthode est qu'une fois le maillage initialisé, tous les mécanismes et fonctionnements existants dans *HEREZH++*, sont disponible pour ce maillage et les éléments associés. En particulier, les sauvegardes éventuelles sur disque qui peuvent être post-traitées lors d'une éventuelle mise au point. Remarquons également que la

partie *Umat* intégré dans *ABAQUS* est figé dans le temps. En particulier, l'introduction de nouvelles loi de comportement dans *HEREZH++* n'entraîne aucune modification de l'interface.

3. Tests numériques sur un essai de traction uniaxiale

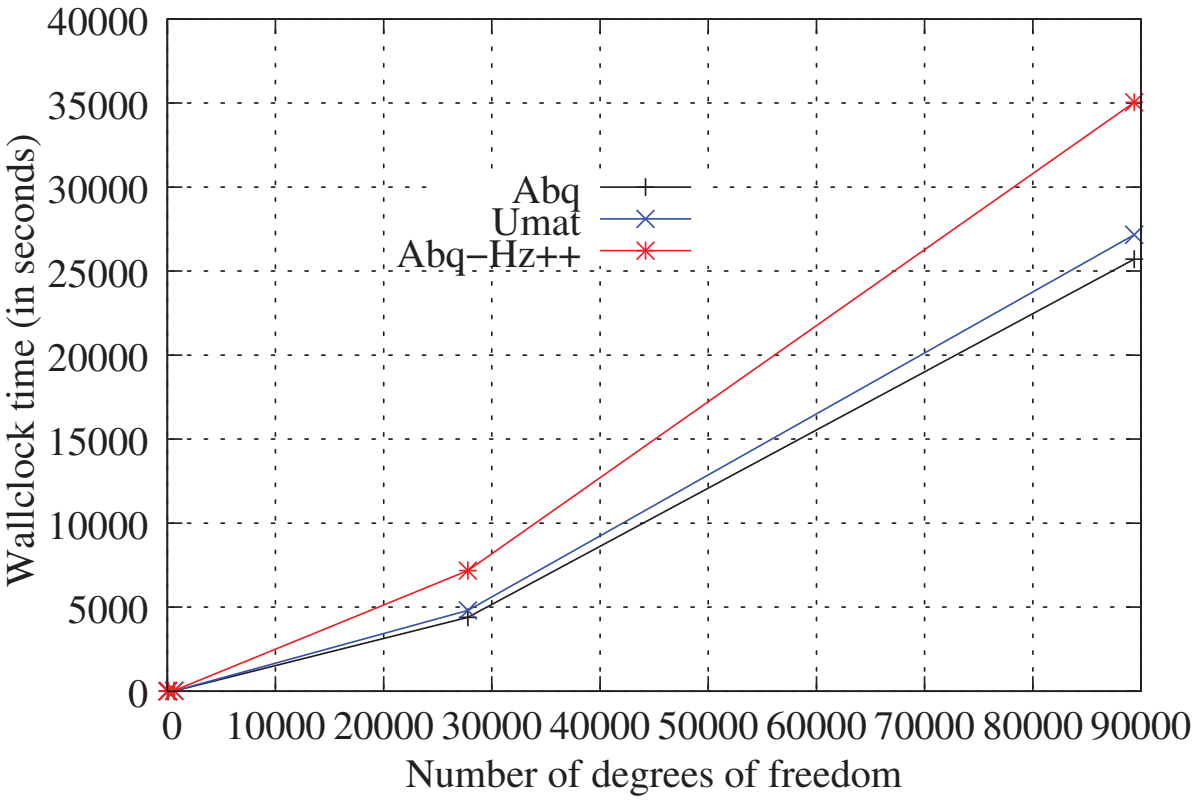

Figure 2. *Évolution du temps de calcul sur l'essai de traction en élasticité en fonction du nombre de degrés de liberté*

Pour tester les temps de transfert entre *ABAQUS* et *HEREZH++*,[1] nous avons traité un cas de calcul non-linéaire en élasticité (100 incréments de temps) sur un cube en traction uniaxiale. Sur la figure 2, nous comparons les temps de calcul donnés par le calcul direct sur *ABAQUS* (noté *Abq*), celui donné par une interface *Umat* en élasticité (*Umat*), et les temps de calcul donné par l'interface entre *Umat* et *ABAQUS* (*Abq-Hz++*). Nous faisons évoluer le nombre d'éléments du maillage, nous utilisons une loi de comportement particulière dans *HEREZH++* qui ne calcule aucune grandeur, le calcul de la loi élastique étant réalisé dans la *Umat* d'*ABAQUS*. Ainsi, nous évaluons simplement le temps de transfert par les tubes nommés ainsi que les transformations effectuées par *HEREZH++*.

Avec un maillage de $20 \times 20 \times 20$ éléments (soit 89373 degrés de liberté), en comparant les temps de calcul entre la *Umat* et l'interface entre la *Umat* et *HEREZH++*

1. Tous les calculs présentés dans la suite ont été effectués sur une machine Dell Precision Workstation 450 comprenant deux processeurs *Intel(R) Xeon(TM) CPU 2.40GHz* sur environnement *Linux version 2.6.11.11, Debian 1 :3.3.5-8ubuntu2* et la version 6.5 d'*ABAQUS*.

(voir courbe *Abq-Hz++*), nous obtenons un facteur 3 ce qui s'avère globalement satisfaisant car conduisant à des temps de transferts convenables et non-prohibitifs.

4. Exemple d'application avec une loi de comportement complexe et un calcul de type industriel

L'objectif de ce test est de montrer que l'interface proposée est utilisable dans le cadre d'une loi complexe associée à un maillage représentatif de pièce réelle simulée. Ce test qualitatif illustre également l'intérêt de l'utilisation d'une plateforme logiciel *HEREZH++* strictement externe à *ABAQUS* permettant le développement d'une loi externe complexe dont l'implantation directe dans une routine *Umat* Fortran classique aurait été difficile. Ce calcul provient de la documentation d'Abaqus (Hibbit *et al.*, 2005). Il correspond à un calcul sur un joint de cardan en caoutchouc qui subit de fortes rotations angulaires.

La loi de comportement originale utilisée est de type élasto-visco-hystérétique qui présente la particularité de calculer la contrainte finale sous la forme additive suivante :

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_e + \boldsymbol{\sigma}_v + \boldsymbol{\sigma}_h \qquad [1]$$

dans laquelle $\boldsymbol{\sigma}_e$ représente une contribution élastique de Hooke, $\boldsymbol{\sigma}_v$ représente une contribution visco-élastique constituée par deux branches de Maxwell (un ressort et un dashpot en série), $\boldsymbol{\sigma}_h$ représente une contribution hystérétique originale. Alors que les deux premiers comportements sont classiques, on se reportera aux références (Guélin, 1980), (Manach *et al.*, 2002) pour plus d'information sur le comportement complexe d'hystérésis. Ce comportement nécessite, d'une part, l'intégration d'une équation constitutive qui décrit de manière implicite l'évolution du déviateur des contraintes $\boldsymbol{S}$ en fonction du déviateur des vitesses de déformation $\bar{\boldsymbol{D}}$, d'une variation finie de la contrainte $\Delta_R^t \boldsymbol{S}$ entre un instant privilégié de référence R et l'instant final t et d'une puissance non réversible Φ fonction des autres grandeurs ainsi que μ et β deux paramètres matériaux :

$$\dot{\boldsymbol{S}} = 2\mu\bar{\boldsymbol{D}} + \beta\Phi(\boldsymbol{S}, \boldsymbol{D})\Delta_R^t \boldsymbol{S} \qquad [2]$$

Le modèle d'hystérésis nécessite, d'autre part, la mise en place d'un algorithme gérant ces points de référence de l'espace des tenseurs de contraintes. Ces points constituent une "mémorisation" discrète de l'histoire du chargement, qui peut-être a priori différente en chaque point du matériau. Une des difficultés introduites par l'hystérésis est alors la gestion et le stockage dynamique des contraintes de références. La solution développée s'appuie sur les listes doublement chaîné de la librairie STL associée au C++. L'utilisation de classe génériques de tenseurs permet également une transposition simple des expressions analytiques à l'implantation informatique.

La figure 3 présente les résultats de calcul obtenus par l'interface entre *ABAQUS* et *HEREZH++* sur le joint de cardan. Le temps de calcul total est de $15339\,s$ avec

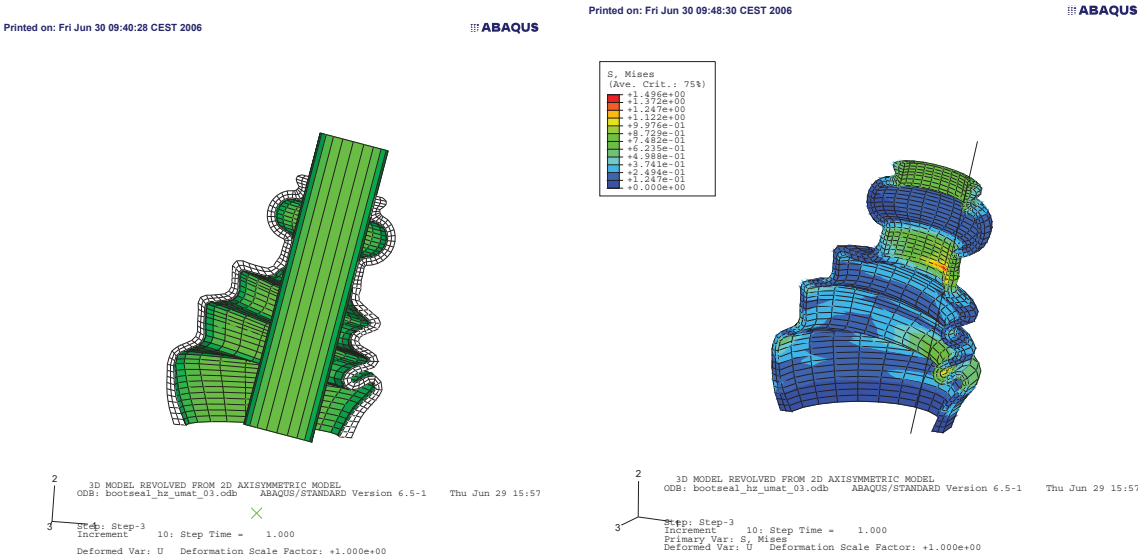

Figure 3. *Configuration déformée de la moitié du modèle et isovaleurs des contraintes principales sur le calcul du joint de cardan.*

14116 s pour le calcul de la loi de comportement de *HEREZH++*. Le temps de transfert est d'environ 270 s ce qui montre que dans ce test, c'est le calcul de la loi de comportement qui est le plus pénalisant.

5. Conclusion

Dans ce papier, nous avons proposé une technique permettant de coupler un code de calcul de laboratoire *HEREZH++* avec le code *ABAQUS*. Nous avons montré que ce couplage permet le développement rapide de loi de comportement sophistiquée dans un code "maison". Sur un test industriel, nous avons utilisé une nouvelle loi de comportement d'élasto-visco-hystérésis qui nécessite une importante place mémoire ainsi qu'une sauvegarde des informations au cours du calcul. Cette implantation aurait été très difficile à réaliser avec une simple subroutine *Umat* en Fortran77.

6. Bibliographie

Guélin P., « Remarques sur l'hystérésis mécanique », *J. Mécanique Théorique et Appliquée*, vol. 19, n° 2, p. 217-247, 1980.

Hibbit, Karlson, Inc. S., « Abaqus », *Theory Manual - version 6.5*, 2005.

http ://web.univ ubs.fr/lg2m/rio, « Herezh++ », *certification IDDN-FR-010-0106078-000-R-P-2006-035-20600*, 2005.

Manach P., Couty N., « Elastoviscohysteresis constitutive law in convected coordinate frames : application to finite deformation shear tests », *Computational Mechanics*, vol. 28, p. 17-25, 2002.