FEMJava : une approche Java pour les calculs multiphysiques multichamps couplés en mécanique

Stéphane Lejeunes, Thien An Nguyen Van, Dominique Eyheramendy, Adnane Boukamel

CSMA 2013
11e Colloque National en Calcul des Structures
13-17 Mai 2013

Une approche Java pour les calculs multiphysiques multichamps couplés en mécanique : Application à la thermo-chimio-mécanique des élastomères chargés

Stéphane Lejeunes [1] *, Thien An Nguyen Van [2], Dominique Eyheramendy [3], Adnane Boukamel [4]

1 LMA, CNRS, UPR 7051, Aix-Marseille Univ, Centrale Marseille, F-13451 Marseille, lejeunes@lma.cnrs-mrs.fr
2 LMA, CNRS, UPR 7051, Aix-Marseille Univ, Centrale Marseille, F-13451 Marseille, thienan@lma.cnrs-mrs.fr
3 LMA, CNRS, UPR 7051, Aix-Marseille Univ, Centrale Marseille, F-13451 Marseille, dominique.eyheramendy@centrale-marseille.fr
4 EHTP, Route d'El Jadida, BP 8108 Casablanca, Maroc
* Auteur correspondant

Résumé — Dans cette communication, nous présentons une architecture de code dévolues au traitement de problèmes couplés (faiblement ou fortement). Ce travail intègre une réflexion sur représentation de champs physiques et sur une vision globale de construction de formulation éléments finis couplés intégrant la modélisation du comportement. Nous illustrons l'approche sur un modèle thermo-chimio-mécanique en grandes transformations pour les élastomères chargés.
Mots clés — éléments finis, orienté objet, multiphysique, couplages faibles et forts

1. Introduction

En ingénierie moderne, la simulation est devenue un outil incontournable. Dans bien des cas, les outils industriels suffisent car ils intègrent de manière plus ou moins avancée les problématiques souhaitées par l'industriel ou le chercheur. Cependant, dans les domaines d'activité qui sont prospectifs et donc bien souvent innovants, les outils classiques n'intègre que peu ou mal les physiques et/où conditions des phénomènes étudiés. Dans ce contexte, des codes de calcul ultra-flexibles sont tout à fait opportuns. De manière générale, les approches structurées de programmation, et en particulier à objets (voir [7]), ont permis de faire évoluer très notablement les outils de simulation en mécanique. Ce travail se place dans cette continuité, et nous présentons cadre très général développé en Java pour des problèmes à physiques multiples dans un contexte mécanique fluide ou solide (voir [1][2][3]) : code orienté objet FEMJava. Ce code est structuré de manière générique afin d'intégrer naturellement les problèmes à physiques et champs multiples (voir [2][3][4]). Dans cette communication, nous présentons dans un premier temps, le concept clé de l'architecture du code : une gestion multi-champs des données discrètes, ainsi que deux concepts permettant d'améliorer la gestion de la cohérence d'un code de calcul en mécanique. Nous décrivons ensuite la base du modèle objet pour la gestion du comportement de matériau et son intégration numérique. L'approche est illustrée sur une formulation couplée de thermo-chimio-mécanique de matériaux élastomères chargés.

2. Une architecture à objets pour problèmes à champs multiples

2.1 Base de l'architecture à champs multiples

L'approche que nous proposons dans ce travail s'appuie sur une description générique des champs physiques discrets nécessaires à la simulation d'un problème quelconque : champs scalaires, champs vectoriels, champs tensoriels. Cela constitue un changement assez différent des approches classiques à objets en général basé sur l'élément, le nœud et le degré de liberté comme initialement proposé par DUB93 (voir EYH et références citées). Cette approche a pour but de dissocier les différents degrés de

liberté d'un problème en les organisant par rapport aux données physiques. Le second élément de l'approche consiste à définir dans la description du champ un niveau global (champs scalaire, vectoriel ou tensoriel définie sur un domaine physique) simple à manipuler globalement, et un niveau local, autour d'une notion proche de l'élément, donc locale, qui supporte l'interpolation. Ainsi, on peut gérer de manière plus naturelle les physiques couplées soit globalement dans une formulation ou localement lors de couplages élémentaires (calcul des contributions élémentaires par exemple).

L'organisation des données est basée sur 3 niveaux de description comme montré Fig. 1. les données du problème éléments finis sont basées sur 3 niveaux de description. Les géométries de base telles que les lignes ou points (« Geometry level » sur la Fig. 1) permettent de définir la topologie générale du domaine : points, lignes, sous-domaines, domaines. La description du maillage (« Mesh level » sur la Fig. 1) est basée sur un ensemble de données décrivant les éléments géométriques : ensemble d'éléments géométriques, nœuds géométriques. Plusieurs maillages peuvent s'appuyer sur la même géométrie. A ce niveau, le maillage est décrit comme un ensemble de nœuds (entités purement géométriques à ce niveau là), et d'éléments géométriques. La cinématique de base de l'élément est définie au niveau de l'élément géométrique (fonctions d'interpolation définissant le mapping géométrique local/global). Toutes les données cinématiques nécessaires au niveau des calculs des formes élémentaires de la méthode des éléments finis seront fournies dans un objet « cinématique » donné par la géométrie élémentaire. Le champ au sens des éléments finis est décrit de manière analogue par un ensemble de champs élémentaires. Le champ élémentaire se superpose au champ géométrique et supporte l'interpolation du champ sur l'élément. La description d'un champ peut se faire soit aux nœuds, soit aux points de Gauss comme classiquement dans la méthode des éléments finis. Cette idée se retrouve de manière originale dans [8]. La description du champ élémentaire suit le même schéma multi-niveau que le champ. Comme le montre la Figure 1, la définition du champ élémentaire est basée sur la description de l'élément géométrique. Ainsi, un même élément géométrique peut servir de base à plusieurs champs élémentaires. Il peut être composé de nœuds ou de point de Gauss (données aux nœuds ou points de Gauss). Le passage d'informations entre les différents niveaux (champ élémentaire/géométrie élémentaire, champ/maillage,…) se fait grâce à une gestion de liens entre les entités des différents niveaux (sens ascendant dans les niveaux de la Fig. 1). A titre d'exemple, un champ élémentaire connaît la géométrie élémentaire sur laquelle il s'appuie. Il peut ainsi avoir accès à la définition cinématique de l'élément, principalement les fonctions d'interpolations correspondantes et leurs dérivées partielles successives. Il est important de noter que dans une telle organisation, les données ne sont ainsi jamais dupliquées.

Cette organisation de données permet ainsi au développeur de se concentrer sur la physique qui couple ces différents champs dans le contexte d'un problème complexe et sur les algorithmes de résolutions, au niveau local ou global. Cela permet une gestion assez naturelle de champs multiples dans un contexte multiphysique. De plus, cela facilite l'implémentation d'algorithmes de résolution en proposant de manière cohérente des niveaux globaux et locaux des données éléments finis.

2.2 Sur la cohérence des données dans les codes éléments finis

Nous proposons dans ce code 2 modèles de programmation afin de renforcer la cohérence des données au sein d'un code de calcul éléments finis ou similaire. On pourra consulter [8] pour les concepts de base de programmation à objets en Java.

Le premier modèle se place dans le contexte des données de champs élémentaires décrites dans le paragraphe 1. On retrouve comme dans tout code éléments finis des données globales (notion de champs) associées à des données locales (champs élémentaire ou donnée au nœud –ou au point de Gauss). Au sein d'un code de calcul, il est souvent nécessaire d'écrire les algorithmes en manipulant des données globales, soit ces données locales, tout en maintenant une cohérence de traitement entre ces deux niveaux. L'idée est d'alors d'intégrer les traitements locaux et globaux au sein du même espace de référence. On considère l'exemple de la Fig. 2, dans laquelle la classe ThermoChemoMechanicsFormulation est définie. Dans la méthode *initialize(Domain)*, les champs inconnus sont déclarés, dans cet exemple cinq champs inconnus (déplacements, pression, température, degré de cuisson et dérivée de la pression par rapport à la température). Un champ complémentaire est nécessaire : le champ des contraintes. Cette définition du problème est naturelle pour un mécanicien.

Dans le même espace de référence, à savoir la classe, une nouvelle classe est définie, ici ThermoChemoMechanics, dans laquelle les 2 principales méthodes sont la création de l'élément couplé (au sens éléments finis) et la méthode qui permet le calcul des grandeurs élémentaires. Ces structures englobantes (dite classe externe) et locale (appelée classe interne) permettent de gérer dans un même espace de référence les donnée locales et globales de la formulation, et donc d'en faciliter la maintenance cohérente. Ce schéma peut être appliqué à toute partie de code dans laquelle il est nécessaire d'avoir une cohérence de traitement entre plusieurs niveaux d'un concept.

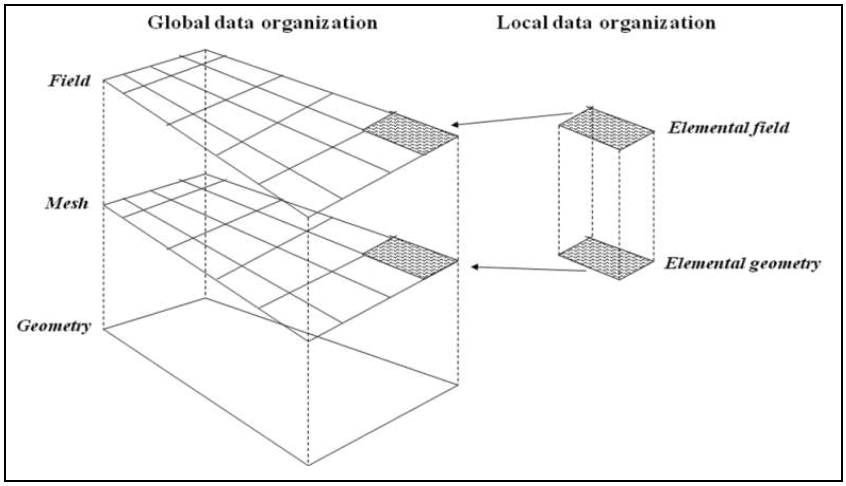

Fig. 1 – Organisation multi-niveau des champs et géométries

```
public class ThermoChemoMechanicsFormulation extends Formulation        // OUTER CLASS DEFINITION
{
   //… partial definition    //
   public void initialize( Domain domain ) {
      Field[] fields = new Field[ 6 ] ;
      fields[0] = domain.createAVectorField ( 0 ) ;  // Vector field : Displacements
      fields[1] = domain.createAScalarField ( 1 ) ;   // Scalar field : Pressure
      fields[1] = domain.createAScalarField ( 2 ) ;   // Scalar field : Temperature
      fields[1] = domain.createAScalarField ( 3 ) ;   // Scalar field : Degree of cure
      fields[1] = domain.createAScalarField ( 4 ) ;   // Scalar field : Derivative of the pressure with respect to temperature
      fields[3] = domain.createATensorField ( 5 ) ;  // Stresses
      domain.setFields ( fields ) ;
      domain.setNumberOfUnknownFields ( 5 ) ;
      Subdomain[] subdomains = domain.getSubdomains () ;
      for( int i = 0 ; i < subdomains.length ; i++ )
         this.initialize ( subdomains[i] ) ;
   }

   public Element getElement( ElementalGeometry aGeom , Quadrature aQuad , ElementalField[] flds, int nb , Material m )
   {
         return new ThermoChemoMechanics ( aGeom , aQuadrature ,flds , nb , m ) ;
   }

   public static class ThermoChemoMechanics extends Element        // INNER CLASS DEFINITION
   {
         public NonlinearElastodynamics ( ElementalGeometry aGeom
               , Quadrature aQuadrature ,  ElementalField[] flds , int n
               , Material m ) {  // … }

          public Hashtable computeElementalMatrices( TimeStep ts ) {}
          // … additional non-implemented abstract methods …
   }
}
```

Fig. 2 – Gestion des données pour une formulation de thermo-chimio-mécanique

Le second modèle permet d'imposer une cohérence au niveau purement algorithmique. L'idée est de ne permettre l'utilisation d'un algorithme que si celui-ci est déclaré compatible avec le modèle de données considéré. Pour illustrer le principe, on considère la hiérarchie de classe partielle dans laquelle est donnée la classe Behavior qui représente un modèle de comportement (voir Fig. 3). Les comportements particuliers de matériau héritent de cette classe comme par exemple la classe ViscoPlastic dans notre cas. Cette dernière contient les méthodes nécessaires pour les calculs spécifiques du modèle. La classe abstraite Integrator représente le comportement générique nécessaire à l'intégration numérique des équations du modèle de comportement (méthode abstraite *integrate*) – voir Fig. 3-. La sous-classe EulerElasticAlgorithm implémente l'algorithme d'intégration d'une loi constitutive. Il convient alors que le programmeur s'assure que l'algorithme est bien capable d'intégrer la loi considérée. Afin de le rendre conscient de ce contrôle et donc renforcer la cohérence du code de calcul, nous proposons un schéma basé sur les interfaces en Java (spécification). On considère l'interface Integrable et ses sous-interface qui spécifient les méthodes que doit posséder un comportement (sous-classes de Behavior) pour pouvoir être intégré par l'algorithme choisi (voir . Ainsi, le modèle ViscoElastic implémente l'interface EulerElasticIntegrable et possède toutes les méthodes spécifiées dans l'interface EulerElasticIntegrable afin de conduire le calcul d'intégration des équations.

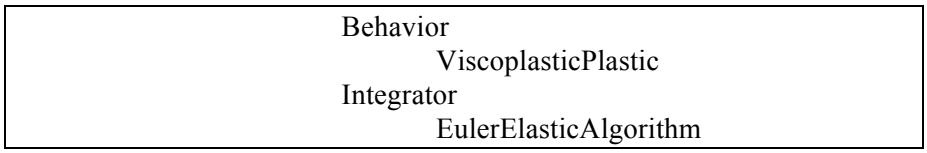

Fig. 3 – Vue partielle de la hiérarchie de classe pour les lois constitutives et les algorithmes d'intégration

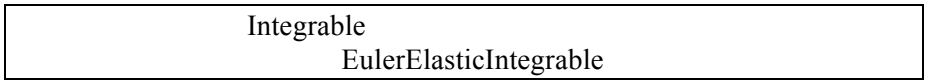

Fig. 4 – Spécifications pour les algorithmes d'intégration

3. Une architecture à objets pour un modèle matériau à plusieurs physiques

Chaque comportement de matériau est construit à partir d'un assemblage de comportements élémentaires différents. Afin de donner les principes de modélisation des objets « matériau », nous nous appuyons sur les travaux décrit dans [5][6]. Nous proposons une structure hiérarchique basée sur une classe abstraite Material pour décrire les matériaux thermo-chimio-mécanique. Ce schéma d'implémentation généralise un assemblage rhéologique en parallèle, d'éléments mécaniques partageant le même incrément de déformation, et en série (décomposition de la déformation en partie thermique, chimique et mécanique) des différentes physiques. Le modèle orienté objet du comportement matériau est présenté dans le diagramme UML Fig. 5. Nous proposons une structure hiérarchique basée sur la super-classe abstraite Material pour décrire les matériaux thermo-mécanique et thermo-chimio-mécanique. Chaque matériau est construit à partir d'un assemblage de comportements élémentaires différents (attribut *behavior* de la classe Material). Nous avons donc construit des sous-classes de Material pour décrire le comportement mécanique, comme par exemple : comportement visco-élastique de type Kelvin-Voigt (classe KelvinVoigt), comportement visco-plastique de type Bingham (classe Bingham), ….La classe Material et ses sous-classes ont pour comportement premier la gestion complète des propriétés matériaux (classe MaterialProperty). Les sous-classes de MaterialProperty gèrent les variations des propriétés matériaux par rapport aux inconnues. Ces classes gèrent ensuite la définition des variables internes aux points de Gauss. Enfin, la gestion du comportement tangent est réalisée dans ces classes (méthode *computeConstitutiveMatrix()*). Dans les classes de comportement plus spécifiques, classe Thermo-Mechanical et ses sous-classes,

nous trouverons la gestion de variables propres à ces formulations : chaleurs latentes, retrait chimique,… Ces comportements sont implémentés dans des classes spécifiques : classes LatentHeat, Dissipation, Kinetics,…Ce schéma d'implémentation généralise un assemblage en parallèle d'éléments mécaniques (partageant le même incrément de déformation) et en série des différentes physiques (décomposition de la déformation en partie thermique, chimique et mécanique). A l'objet matériau est associé un comportement (voir Fig. 6 classe Behavior). Les calculs propres au comportement sont réalisés dans les sous-classes de Behavior, connaissant le matériau, en utilisant des intégrateurs spécifiques (attributs *material* et *integrator*). Les intégrateurs, classes Integrator et sous-classes gèrent les calculs d'intégration locaux des lois de comportement, indépendamment de celles-ci. Le lien entre l'intégrateur et la loi de comportement est défini grâce à l'interface Integrable et ses sous-interfaces qui permet d'autoriser ou non l'utilisation d'un type d'intégrateur pour le comportement (voir interface Integrable dans Fig. 7). Cela garantie par le biais de cette spécification, le fait que le programmeur choisit l'intégrateur de manière éclairée.

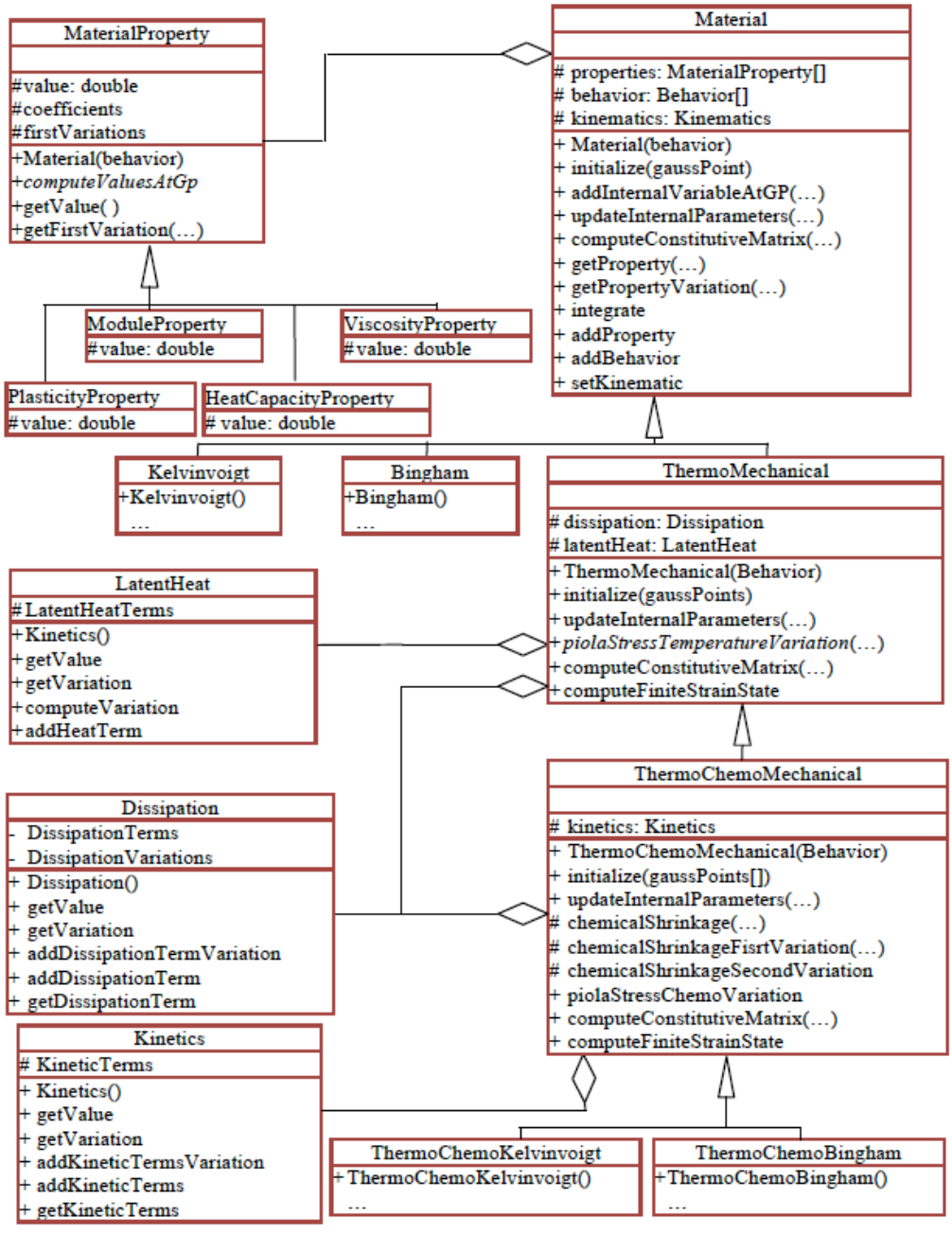

Fig. 5– Diagramme UML décrivant le matériau

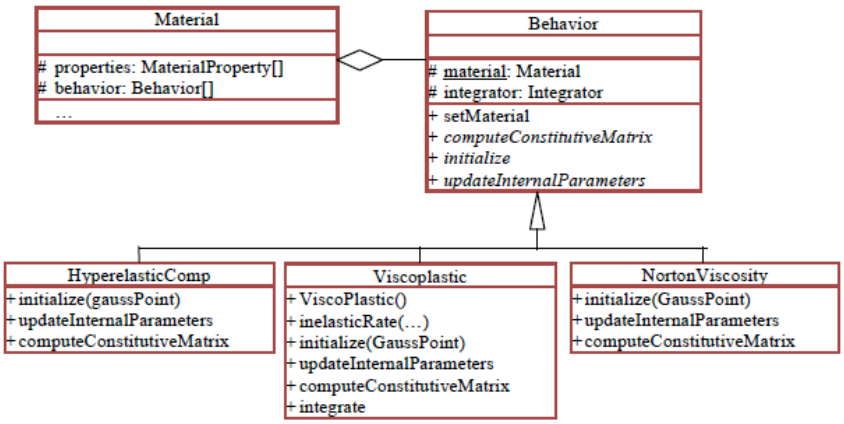

Fig. 6- Diagramme UML décrivant le comportement

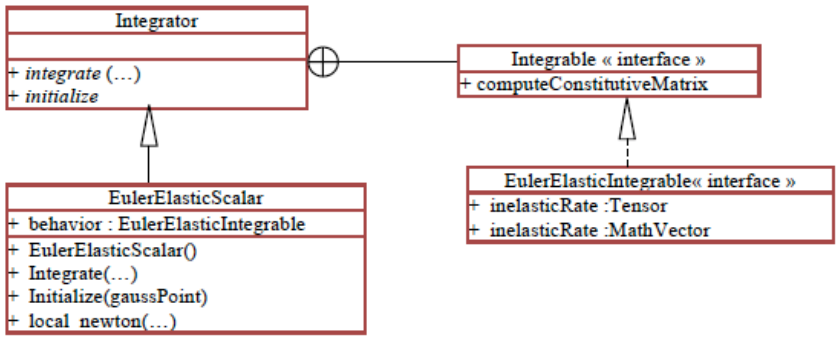

Fig. 7– Diagramme UML décrivant l'intégration de loi d'écoulement mécanique

4. Application à un modèle de comportement thermo-chimio-mécanique pour les élastomères chargés

Nous illustrons l'approche sur un modèle de comportement de matériau à physiques fortement couplées comme le montre le schéma Fig. 8. Nous n'entrons pas ici dans les détails du modèle que l'on pourra trouver dans [5][6][8]. Nous considérons un modèle constitutif couplé : mécanique, thermique et chimique. Le processus chimique modélise la cuisson d'un élastomère chargé. La réaction chimique est caractérisée par le degré de réticulation, qui est reflète le nombre de ponts entre chaînes de polymères, qui entraine la rigidification du matériau. La mécanique produit par auto-échauffement de la chaleur et influe par la pression sur la réaction chimique. La température influe sur les paramètres matériaux mécaniques et par un terme de dilatation thermique. Elle influe également sur la réaction chimique. La réaction chimique est exothermique. Le degré de cuisson du matériau modifie les propriétés mécaniques (rigidification quand le degré de réticulation augmente, pas de réversibilité). Le modèle d'équations est basé sur la résolution :

- des équations de la mécanique
- d'une équation de la chaleur issue du 1er principe de la thermodynamique
- d'une équation d'évolution de la réaction chimique (évolution du degré de réticulation du matériau)

A cela, s'ajoute une condition de faible compressibilité du milieu. Le code développé sur le modèle permet de vérifier qualitativement les couplages entre les différentes physiques sur ce modèle développés en transformations finies. On considère pour illustrer l'approche une lame d'élastomère

cisaillé cycliquement comme le montre la Fig. 9 – Lame d'élastomère cisaillé - ProblèmeFig. 9. La température est imposée sur le bord du domaine et il est chauffé durant les 2 premiers cycles de chargement mécanique. La température imposée au bord et la température initiale du domaine sont de 293K. Dans la Fig. 10, on voit qu'après les 2 premiers cycles la température continue à augmenter à cause de l'auto-échauffement du matériau. On observe l'hystérésis qui conduit à cet échauffement. Le degré de cuisson de matériau augmente avec un temps de latence important de la réaction de réticulation de l'élastomère. Sur la courbe d'hystérésis, on observe alors une rigidification du comportement de l'élastomère (évolution des propriétés mécaniques)

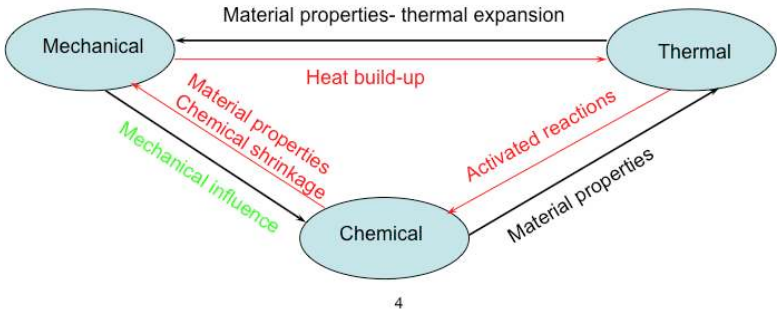

Fig. 8 – Modèle de couplage thermo-chimio-mécanique

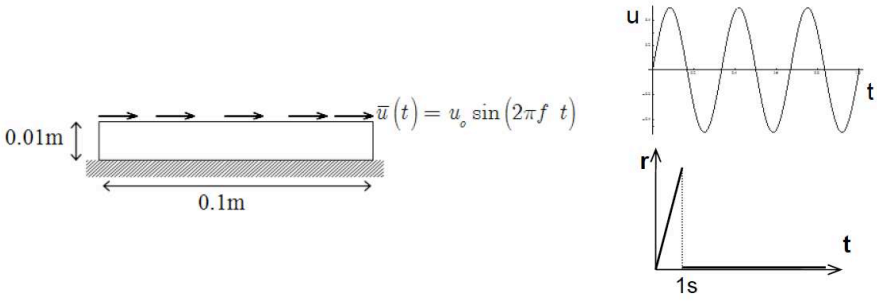

Fig. 9 – Lame d'élastomère cisaillé - Problème

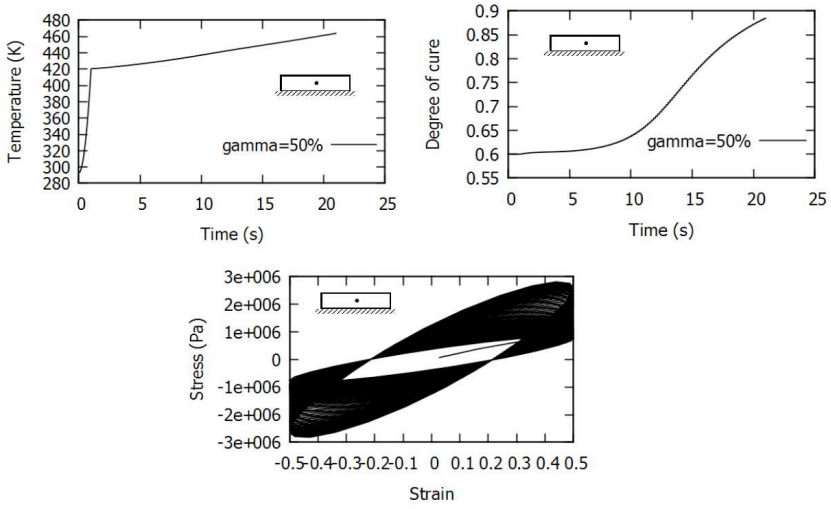

Fig. 10 – Evolution de la température et du degré de cuisson – Courbe d'hystérésis

5. Conclusion

Nous avons présenté dans cette communication deux concepts clés d'un environnement de calcul pour multiphysiques couplées. Ceux-ci ont pour but de maintenir plus efficacement la cohérence du code dans son évolution dans le cadre de physiques multiples et couplées de manière complexe. L'implémentation proposée ici est en Java, cela pouvant être étendue à d'autres langages acceptant ou permettant de simuler le mécanisme d'interface. L'exemple présenté est un modèle de thermo-chimio-mécanique de matériau L'architecture présentée dans ce travail a été étendue naturellement dans de nombreux domaines de la simulation numérique en mécanique (voir références dans [7]). Ces concepts d'organisation peuvent s'étendre assez naturellement à la définition symbolique des modèles d'équations couplées (voir [7]). Dans la suite de ce travail, nous nous attacherons à étendre de type d'approche à la gestion d'autres types de formulations discrètes telles que la méthode isogéométrique.

Références

[1] D. Eyheramendy, Java for nonlinear computational mechanics: An initial approach to handle complexity, NATO Advanced Workshop: Multi-physics and Multi-scale computer models in nonlinear analysis and optimal design of engineering structures under extreme conditions, Bled, Slovenia, Eds. Ibrahimbegovic & Brank, (2004) pp. 476-480.

[2] D. Eyheramendy, New tracks for future computational platforms for engineering applications, Numerics in Geotechnics, Ed. Elmepress Int., (2006) pp. 1-15.

[3] D. Eyheramendy, High abstraction level frameworks for the next decade in computational mechanics, Innovation in Engineering Computational Technology, Eds. B.H.V. Topping, G. Montero and R. Montenegro, ©Saxe-Cobourg Publications, Chap. 3, (2006) pp. 41-61.

[4] D. Eyheramendy and F. Oudin, Advanced object-oriented techniques for coupled multiphysics, In Civil engineering computations: Tools and Techniques Ed. B.H.V. Topping, ©Saxe-Cobourg Publications, Chap. 3 (2007) pp. 37-60.

[5] Thien An Nguyen Van, Sur la modélisation et la simulation du couplage thermo-chimio-mécanique au sein des élastomères chargés, Thèse de doctorat, Aix-Marseille Université, Novembre 2012.

[6] T.A. Nguyen Van, S.Lejeunes, D. Eyheramendy and A.Boukamel, A finite strain thermo-chemo-mechanical coupled formulation for filled rubber, ECT 2012, Sept. 2012, Dubrovnik, Croatia.

[7] R. Saad, Sur une approche à objets généralisée pour la mécanique nonlinéaire, Thèse de doctorat, Aix-Marseille Université, Décembre 2011.

[8] D. Flanagan, Examples in Java in a Nutschell, Eds. O'Reilly, 2000.

[9] T.A. Nguyen Van, S. Lejeunes, D. Eyheramendy, A. Boukamel, Sur un modèle de couplage Thermo-Chimio-Mécanique pour les élastomères chargés, Actes du 11ème Colloque CSMA, Giens 2013.