GranOO : plateforme de simulation DEM en dynamique explicite

Jean-Luc Charles, Damien André, Ivan Iordanoff

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

GranOO : plateforme de simulation DEM en dynamique explicite

Jean-Luc CHARLES [1], Damien ANDRÉ [2], Ivan IORDANOFF [3]

[1] Arts et Métiers ParisTech, I2M, UMR 5295, F-33400 Talence, France, jl.Charles@i2m.u-bordeaux1.fr
[2] Arts et Métiers ParisTech, I2M, UMR 5295, F-33400 Talence, France, damien.Andre@etu.u-bordeaux1.fr
[3] Arts et Métiers ParisTech, I2M, UMR 5295, F-33400 Talence, France, i.Iordanoff@i2m.u-bordeaux1.fr

Résumé — La plateforme **GranOO** (http ://www.granoo.org) est un atelier de développement ouvert, conçu selon l'approche orientée objet, réalisé en C++, proposant un environnement complet pour le développement de simulations DEM (Discrete Element Method) en dynamique rapide. L'atelier GranOO propose des bibliothèques de classes implémentées en C++, qui permettent de manipuler efficacement tous les objets nécessaires aux simulations DEM en dynamique explicite : éléments discrets, interactions, contacts, liens cohésifs, schémas d'intégration explicites...
Un mécanisme de *pluggin* permet aux utilisateur de la plate-forme de développer du code C++ pour interagir de façon sécurisée avec les bibliothèques GranOO. Ce mécanisme permet d'étendre le comportement des objets GranOO pour traduire les besoins spécifiques propres aux simulations de chaque utilisateur.
On se propose de présenter ici le workbench GranOO, son architecture et son exploitaion pour produire des simulation DEM dynamiques, ainsi que des exemples significatifs de simulations réalisées à l'aide cette plate-forme.

Mots clés — DEM, plateforme de développement, workbench, OpenSource, C++, Python, pluggin, éléments discrets, dynamique explicite.

1 Introduction

GranOO est une plateforme ouverte dédiée aux simulations mécaniques dynamiques basées sur la modélisation DEM [2]. La plateforme fournit un ensemble de bibliothèques C++ qui permettent de construire efficacement des simulations DEM dynamiques explicites. L'atelier GranOO est basé sur 3 bibliothèques :
– une bibliothèque géométrique fournit les classes vecteur, repère, coordonnées, quaternions... en 3D. Un effort particulier a été porté sur l'utilisation des quaternions pour traduire les rotations 3D, dans le but d'optimiser performance et précision des calculs.
– une bibliothèque DEM fournit les classes nécessaires à la représentation des concepts DEM : élément discret, lien cohésif, paroie, contact, détection des contacts...
– une bibliothèque d'utiliaires regroupe des classes support de concepts aussi utiles que les pluggins, les capteurs numériques, les outils de lectures des fichiers XML...

La version actuelle de GranOO est développée sur plateforme GNU-Linux. Elle propose :
– des éléments discrets de forme sphérique, possédant les attributs : masse, volume, position, vitesse, accélération (linéaire et angulaire), et pouvant également porter des informations multi-physiques (température, moment magnétique...) ;
– des contacts réguliers, avec possibilité de frottement, cohésion ;
– des liens cohésifs de types variés : ressort unidirectionnel ou bi-directionnel, poutre 3D (traction/compression, flexion, torsion)... avec comportement élastique ou fragile ;
– des schémas d'intégration explicite, avec ou sans amortissement numérique ;
– un générateur de domaines DEM compactés isotropes permettant de réaliser des modèles de milieux continus ;
– un visualisateur de domaines DEM.

GranOO est téléchargeable librement sur le site www.granoo.org. Il est bien sûr possible de faire évoluer l'atelier en utilisant les mécanismes de l'orientée objet supportés par le C++ :
– héritage (pour la création de nouvelles classes sans tout avoir à refaire) ;
– polymorphisme dynamique (re-définition dans les classes dérivées des méthodes virtuelles présentes dans les classes de base) ;
– généricité (instanciation de classes templates pour étendre la généricté à de nouveaux cas...).

2 Conception et développement du workbench GranOO

Les points importants du cahier des initial sont de fournir aux utilisateurs (scientifiques) une plate-forme de développement :

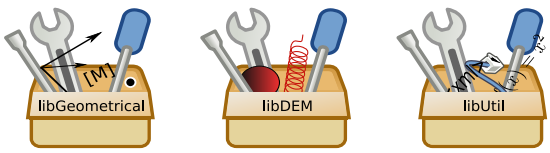
1. dédiée aux calculs DEM en dynamique explicite ;
2. permettant de **prototyper** facilement une simulation DEM ;
3. mais capable aussi de supporter des **calculs lourds** ;
4. permettant de **capitaliser** les développements faits par les uns et les autres ;
5. qui permette de **paramétrer** des simulations DEM variées grâce à un fichier d'entrée unique, lisible, décrivant tous les paramètres d'une simulation ;
6. qui puisse **évoluer** sur la durée, ouverte à la **multi-physique** ;
7. utilisable par des **scientifiques** développeurs C++ plus ou moins aguéris ;
8. portable sur tout type de plate-forme (Linux/Mac/Windows-portable/PC/Cluster) ;
9. n'utilisant que des composants logiciels libres, elle même libre d'accès ;
10. accessible, distribuable et maintenable par Internet...

La réponse à ces besoins est déclinée en 3 axes fondamentaux :
– Choix de conception : approche Orientée Objet (OO) ;
– Choix de développement : langage C++, composants logiciels libres, standards de développement ;
– Architecture informatique : séparation "espace utilisateur / espace système" grâce aux plugins.

2.1 Conception Orientée Objet

L'approche Orientée Objet (OO) permet une analyse et une coneption de haut niveau, qui préserve les concepts fondamentaux des différents domaines impliqués pour une simulation (concepts DEM, géométriques, mécaniques, mathématiques...). Elle donne une grande souplesse de conception grâce aux paradigmes de l'approche OO (héritage, encapsulation, polymorphisme, généricité...).

L'analyse OO des concepts fondamentaux utiles à la simulation DEM nous a conduit à concevoir 3 bibliothèques :



– les concepts fondamentaux de géométrie 3D (Point, Vecteur, Coordonnée, Matrice, Quaternion... Forme) sont regroupés dans la bibliothèque **libGeometrical**. Un accent particulier a été donné à l'utilisation des quaternions pour optimiser performance et précision des calculs en géométrie 3D, très utilisés par la DEM ;
– les concepts de la DEM (Élement discret, joint cohésif, interaction, contact, schéma d'intégration explicite...) sont regroupés dans la bibliothèque **libDEM** ;
– les éléments utilitairse et d'architecture informatique (pluggins, lecteur XML...) sont regroupés dans la bibliothèque **libUtil**.

Diagrammes UML

La conception OO est documentée par des diagrammes de classes qui permettent de donner des vues plus ou moins détaillées sur les classes représentant les entités fondamentales identifiées en analyse/conception. Ci dessous sont présentés des exemples de diagramems UML des classes GranOO :

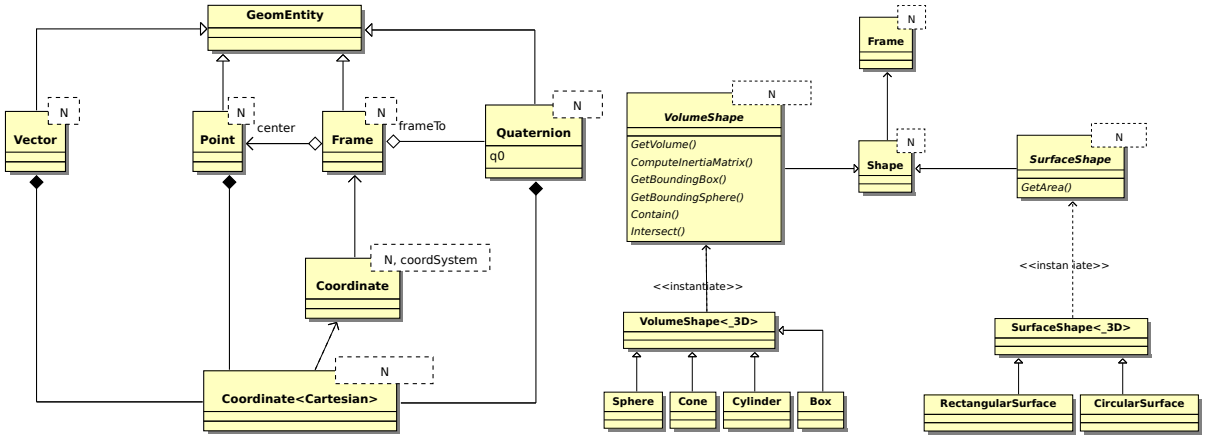


Fig. 1 – libGeometrical : vue globale et exxtrait de l avue détaillée de la classe `Shape`

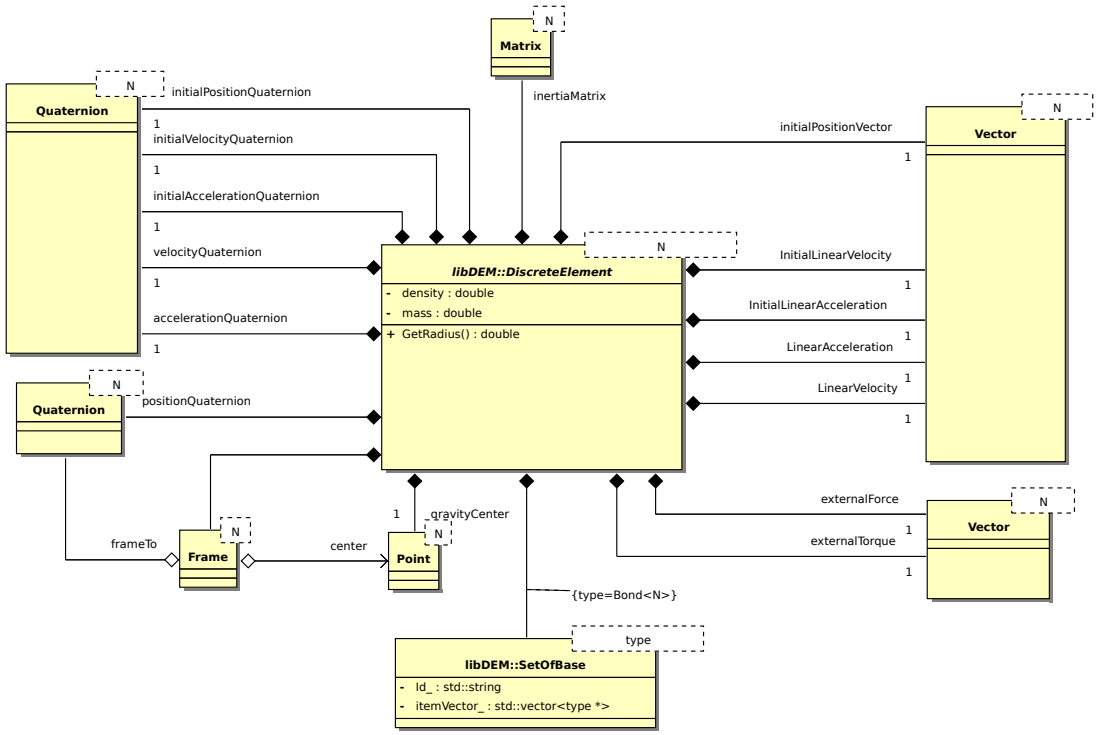


Fig. 2 – libDEM : extrait de la vue de la classe `DiscreteElement`

2.2 Les choix de développement : langage C++ et outils standards libres

Le langage choisi pour développer les bibliothèques GranOO est le C++ , langage OO mature, supportant les *template*, maintenant performant. Il est à noter que le C++ supporte beaucoup de styles de programmation (OO, "à la C", "à la Fortran"...). Beaucoup de bibliothèques libres, de très grande qualité, sont disponibles en C++ (scientifiques, graphiques, informatiques...).
La garantie d'un développement pérenne et efficace du workbench GranOO s'appuie des choix en terme de plate-forme, d'outils et de règles :

– Plate forme de développement GNU-Linux : plate-forme libre, très bien équipée en outils de développement libres !

– Production des exécutables par fichiers Makefile (standard du génie logiciel) ;
– Utilisation de bibliothèques C++ libres incontournables (performance, pérennité) :
  STL (*Standard Template Library* : *generic containers, strings, IOstream*)
  boost (robustesse, performance, diversité des domaines traités...)
  libxml2 (E/S XML)
  CGAL (*Computational Geometry Algorithms Library*)...
– Programmation agile en C++ objet avec généricité (performance, évolutivité...) ;
– Programmation par contrat (Bertrand Meyer, `assert` : robustesse, confiance dans les calculs !) ;
– Gestion des sources sur plate-forme SVN (partage, gestion des versions ...) ;
– Effort de documentation (UML/Doxygen/PDF) pour les utilisateurs et les développeurs des bibliothèques.

2.3 Architecture informatique

Les grands points de l'architecture informatique retenue pour la plate-forme GranOO sont :
– la séparation **espace utilisateur** (scientifique) / **espace système** (bibliothèques GranOO...) ;
– l'organisation en **bibliothèques** de classes thématiques (géométrie 3D, DEM, outils...) ;
– l'utilisation de **pluggins** $\rightarrow$ simplicité, souplesse, évolutivité, capitalisation...

La séparation "espace utilisateur / espace système" repose sur un mécanisme de *plugins*, qui permet à chaque utilisateur de développer du code C++ dans son espace : chacun peut ainsi faire "tout ce qu'il veut" dans son espace de développement, sans risquer de compromettre les bibliothèques système. Ce mécanisme de pluggin protège les bibliothèques fournies avec GranOO, tout en permettant leur extension pour les besoins spécifiques de chaque utilisateur. C'est dans l'espace système que sont conçues et développées les classes fournies avec les bibliothèques GranOO (*libGeometrical*, *libDEM* et *libUtil*). Le code C++ écrit dans ces classes doit être performant et robuste, car toutes les simulations DEM crées par les utilisateurs dépendent de ces classes.

La conception des plugins repose les points suivants :
– La bibliothèque libUtil fournit des classes de base qui modélisent les mécanismes des Pluggins ;
– La bibliothèque libDEM implémente les principaux Pluggins nécessaires à la simulation DEM complète d'un problème de mécanique dynamique ;
– Un Pluggin utilisateur peut être compilé et combiné aux bibliothèques GranOO pour produire un exécutable utilisateur ;
– Un Pluggin intéressant plusieurs utilisateurs peut être validé/optimisé/sécurisé pour intégration dans une bibliothèque système (libDEM par exemple).

L'architecture à base de pluggin est la clef de la capitalisation des développements effectués par les doctorants et les chercheurs qui utilisent le workbench GranOO.

3 Architecture d'exploitation de l'atelier GranOO

3.1 Fichiers d'entrée et pluggins

Une simulation réalisée avec GranOO utilise un fichier d'entrée (ASCII XML) dont les balises permettent de décrire les traitements à effectuer dans 3 sections :
– **pre-processing** (avant la boucle d'intégration temporelle) : lecture du domaine discret, création d'éléments discrets et de liens...
– **processing** (boucle temporelle) : détection des contacts, calcul des efforts appliqués aux éléments discrets, intégration explicite, conditions limites, enregistrement du domaine DEM tous les n pas de temps...
– **post-processing** (après la boucle temporelle) : post-traitements divers, enregistrement du domaine discret...

```
<PreProcessing>
  <PlugIn Id="ConvertBondToBeam3D"   ... />     to create cohesif bonds
</PreProcessing>

<Processing>
  <PlugIn Id="Check3D" />                       to performe computing validation
  <PlugIn Id="ResetLoad3D" />                   to reset all loads
  <PlugIn Id="ApplyBoundaryCondition3D" />      to apply all boundary conditions
  <PlugIn Id="ApplyLoad3D" />                  to compute all the loads
  <PlugIn Id="ApplyBondLoad3D"/>              to compute all the loads on bonds
  <PlugIn Id="IntegrateAccelerationLinear3D" /> to integrate linear acceleration
  <PlugIn Id="IntegrateAccelerationAngular3D"/> to integrate angular acceleration
  <PlugIn Id="SaveDomain3D" IterLoop="50"/>     to save the domain state
</Processing>

<PostProcessing>
</PostProcessing>
```

Fig. 3 – Mise en oeuvre des pluggins dans le fichier d'entrée au format XML

Tous les traitements mentionnés dans le fichier d'entrée sont réalisés par des pluggins (livrés avec GranOO ou développés par l'utilisateur), qui agissent sur un ensemble d'objets identifiés. Les ensemble d'objets (éléments discrets, joints cohésifs, paroies...) sont modélisés par la classe template `SetOf<...>` (basée sur la classe STL `std::vector<...>`). Les pluggins sont appelés dans un fichier d'entrée unique, qui permet de piloter en un seul endroit tous les détails d'une simulation DEM.

Deux applications importantes font partie de la plate-forme GranOO, `cooker` et `gddviewer` :

cooker est un générateur de domaines DEM compactés isotropes, permettant de simuler le comportement de domaines continus, isotropes. `cooker` permet de générer des domaines de forme simple (cube, sphère, cylindre...) en maîtrisant parfaitement le nombre d'éléments, le nombre de coordination (nombre moyen de contacts entre éléments), ainsi qu'un certain nombre d'autres paramètres caractérisant le domaine discret (loi de probabilité des rayons des éléments discrets...). Les fichiers créés par cooker représentant des domaines DEM ont l'extension `.gdd` (GranOO Discrete Domain).

gddviewer est un visualisateur de domaines .gdd qui permet la visualisation de l'état d'un domaine DEM au cours de la simulation

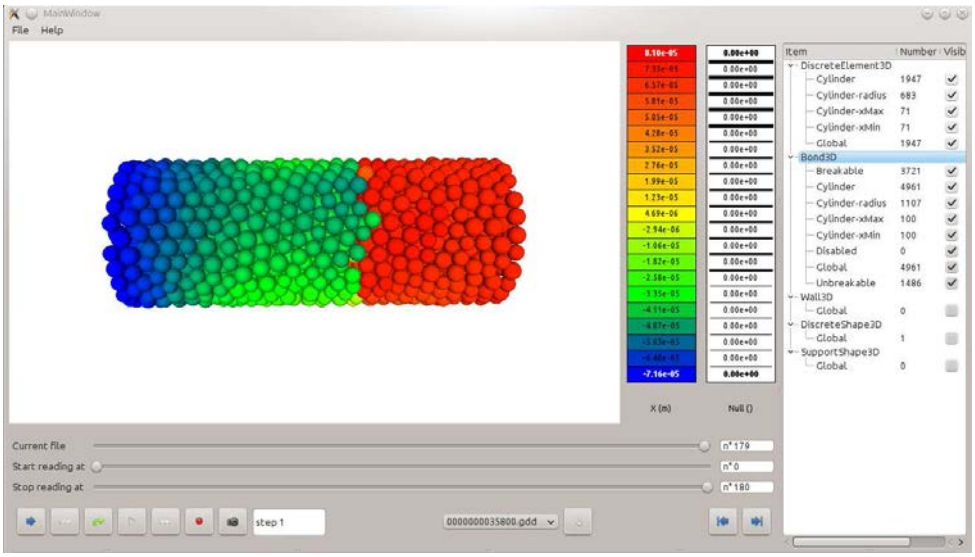


Fig. 4 – Visualisation d'un domaine DEM avec l'application gddViewer

La figure 5 résume l'architecture d'exploitation de GranOO :

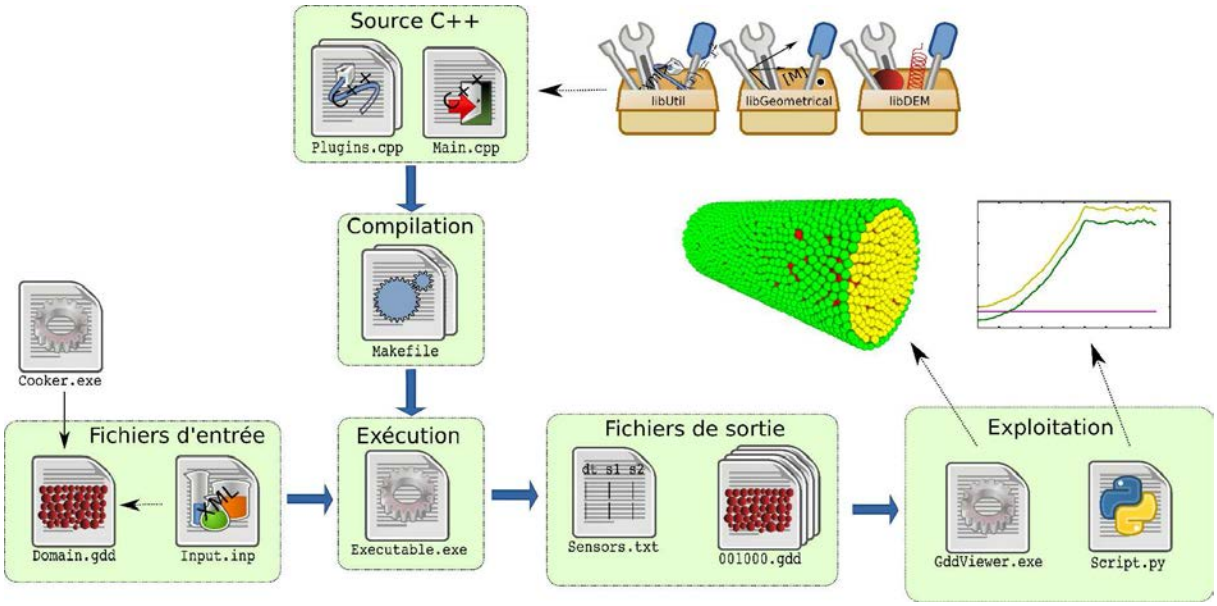


Fig. 5 – Architecture d'exploitation de la plate-forme GranOO

La classe `Sensor` implémentée dans la bibliothèque libUtil permet de créer des objets 'capteurs numériques' grâce auxquels il est possible d'instrumenter l'expérience numérique que constitue une simulation DEM. Les valeurs mesurées par ces capteurs numériques sont enregistrées dans le fichier `sensors.txt` qui peut être traité par des scripts Python pour les tracés graphiques 2D et 3D.

4 Exemple de simulations

Les exemples qui suivent illustrent la diversité des situations de modélisation DEM qu'il est possible de traiter avec la plate-forme GranOO.

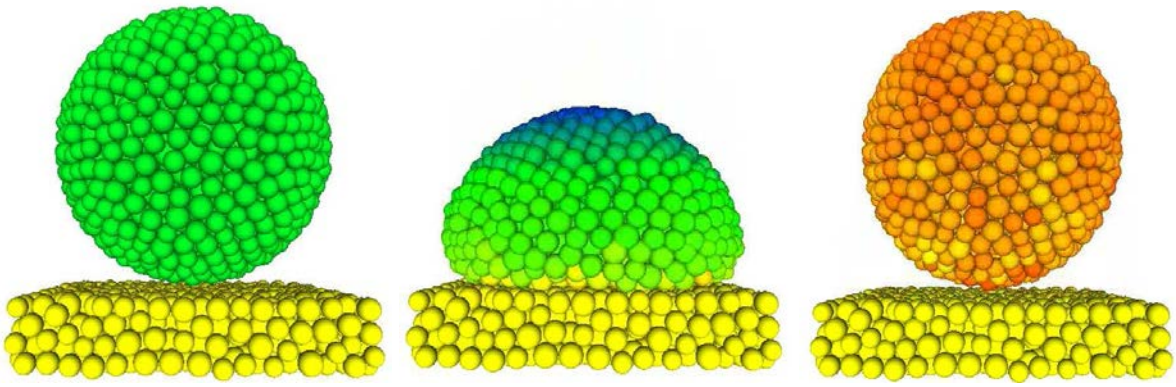


Fig. 6 – Choc mou

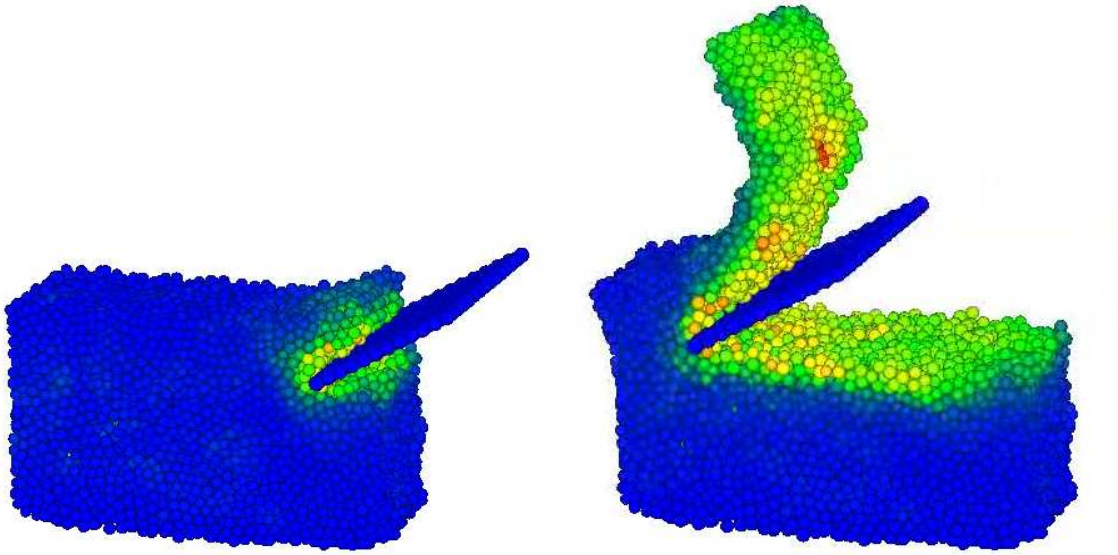


Fig. 7 – Usinage avec formation de copeau

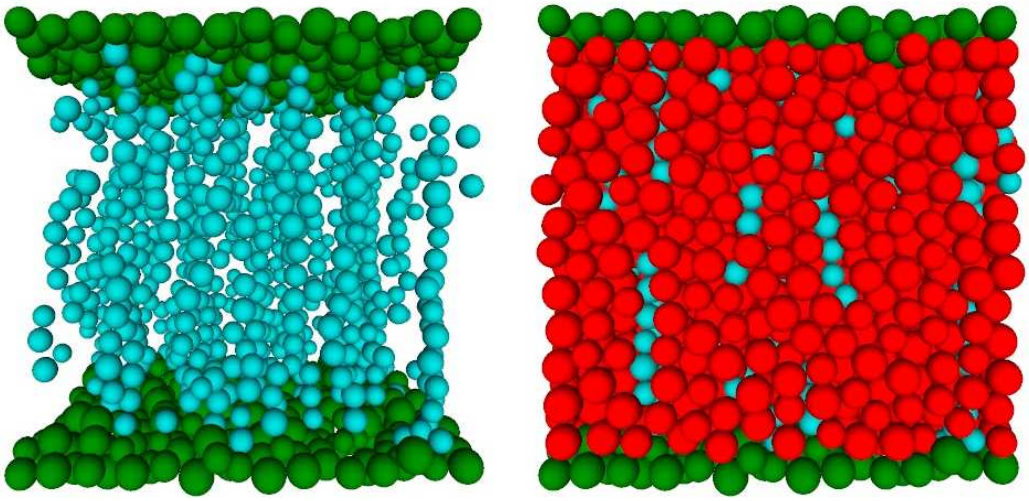


Fig. 8 – Polymère chargé en particules magnétiques

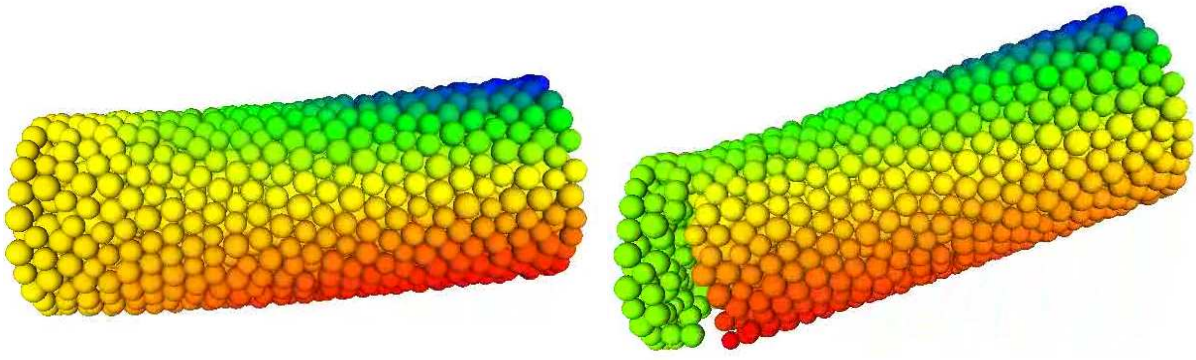


Fig. 9 – Ruptue en flexion d’une éprouvette de matériau continu fragile

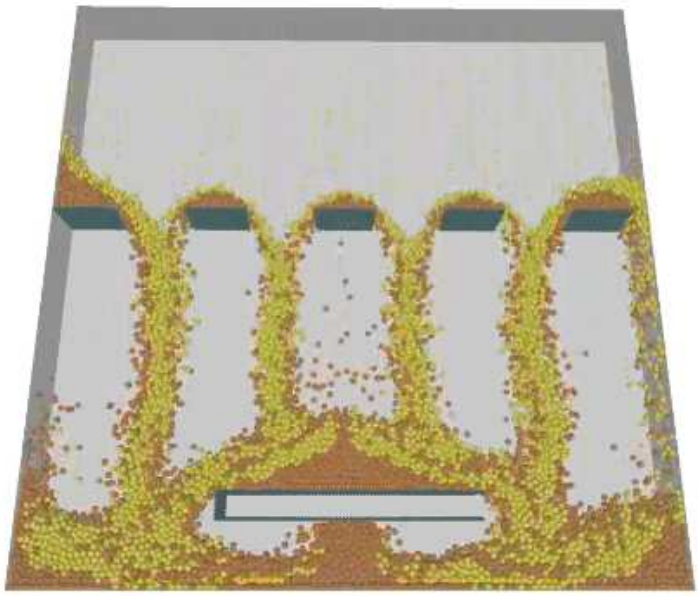


Fig. 10 – Écoulement de sable pour le remplissage de moules en fonderie

Références

[1] D. André, *Modélisation par éléments discrets des phases d'ébauchage et de doucissage de la silice*, Thèse soutenue le 15 mars 2012, Université de Bordeaxu 1.

[2] D. André, I. Iordanoff, J.L. Charles, J. Néauport, *Discrete element method to simulate continuous material by using the cohesive beam model*, Computer Methods in Applied Mechanics and Engineering, Ed. Elsevier, Vol. 213-216, pp. 113-125, 2012.

[3] Jebahi M., Charles J.L., Dau F., Iloul L., Iordanoff I., *3D coupling approach between discrete and continuum models for dynamic simulations (DEM–CNEM)*, Computer Methods in Applied Mechanics and Engineering, Vol. 255, pp. 196-209, 2013, DOI http ://dx.doi.org/10.1016/j.cma.2012.11.021

[4] Terreros I., Iordanoff I., Charles J.L., *Simulation of continuum heat conduction using DEM domains*, Computational Material Science, Vol. 69, pp. 46-52, 2013.