Conception d'un outil adapté à la mise en données des systèmes discrets

Alexandre Martin, Marine Bagnéris, Frédéric Dubois, Rémy Mozul

CSMA 2011
10e Colloque National en Calcul des Structures
9-13 Mai 2011, Presqu'île de Giens (Var)

Conception d'un outil adapté à la mise en données des systèmes discrets

A. Martin[1], M. Bagnéris[1], F. Dubois[1], R. Mozul[1]

[1] *LMGC, Université de Montpellier 2 - CNRS, France, {alexandre.martin,marine.bagneris,frederic.dubois,remy.mozul}@univ-montp2.fr*

Résumé — Cet article présente l'élaboration d'un pré-processeur dédié à la génération de collections d'objets en interaction. Une structure de données générique permettant de définir une représentation discrète d'un objet physique est décrite. L'implémentation de cette structure de données et la procédure permettant de représenter une collection d'objets physiques sont ensuite présentées. Enfin, la mise en œuvre de cette procédure pour écrire des pré-processeurs adaptés à l'étude des milieux granulaires et des maçonneries est discutée.
Mots clés — systèmes discrets, interactions, pré-processeur, maillages éléments finis, CAO, Python.

1 Contexte

De nombreux problèmes de calcul de structure, comme l'étude du comportement des milieux granulaires ou de la stabilité des maçonneries, impliquent une collection d'objets en interaction (*e.g.* contact frottant, cohésion, *etc*). Les formes des objets, leur comportement volumique ainsi que les lois d'interaction peuvent être complexes et faire intervenir des physiques variées. Par ailleurs le nombre d'objets interagissant peut être très important.

Simuler le comportement de telles collections de corps peut s'avérer très complexe. La plate-forme ouverte LMGC90 [5, 6] est conçue pour résoudre de tels problèmes et est utilisée par une large communauté de modélisateurs, chercheurs, *etc* (académiques ou industriels).

La taille et la variété des problèmes étudiés, l'importance de leur état initial et d'une description précise des règles gouvernant leur évolution fait de l'étape de construction du modèle de calcul une étape délicate et coûteuse. Dans cette contribution nous présentons différents aspects d'un outil, le pré-processeur de la plate-forme LMGC90, adapté à ce type de mise en données.

2 Concept

L'enjeu de la conception d'un tel outil est de fournir une structure de données permettant de décrire les différents composants d'une collection d'objets en interaction :

- la discrétisation de chaque objet ainsi que ses modèles physiques (*i.e.* loi de comportement mécanique et/ou thermique) et ses paramètres matériaux,
- les zones géométriques des objets susceptibles d'entrer en interaction,
- les règles de détection des interactions (*i.e.* tables de visibilité), les modèles de comportement de ces interactions et les paramètres associés.

Le pré-processeur de la plate-forme LMGC90 s'appuie sur une représentation des objets physiques par un ensemble de classes détaillé sur la FIGURE 1 ci-après.

La représentation discrète d'un objet physique (classe *avatar*) repose sur un ensemble d'éléments (attribut *bulks* de la classe *avatar*), appuyés sur un ensemble de nœuds (attribut *nodes* de la classe *avatar*). La description topologique d'un élément (classe *bulk*) repose sur sa connectivité, listant les numéros des nœuds qui le compose. Un nœud (classe *node*) est décrit géométriquement par ses coordonnées. Le comportement volumique de l'objet physique est représenté par un modèle (classe *model*), attribué à chaque élément volumique, qui fournit l'ensemble des degrés de liberté associés à chaque nœud de même que les équations qu'ils doivent vérifier. Chaque élément volumique porte un objet (classe *material*) qui décrit localement les paramètres matériau liés au modèle. Chaque nœud porte un objet (classe *dof*) qui

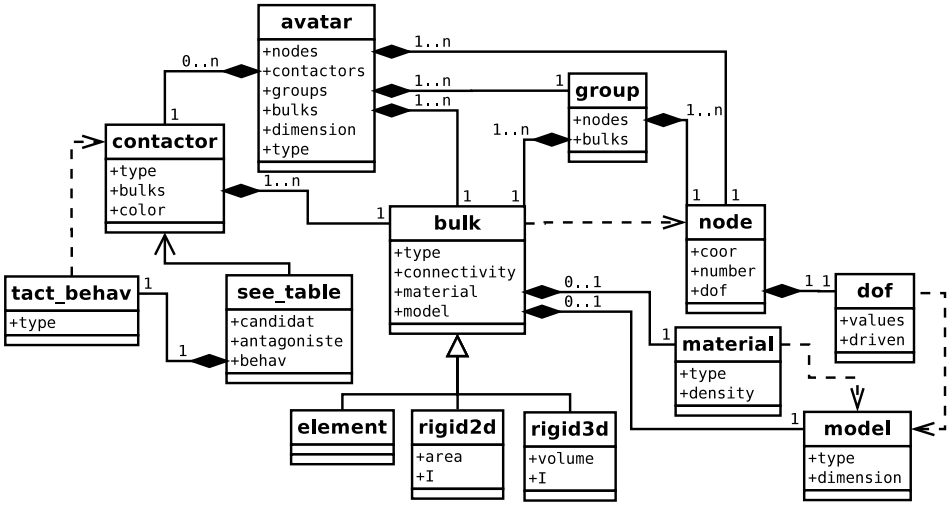


FIGURE 1 – Classes utilisées pour la représentation d'un objet mécanique.

permet de lui attribuer des conditions aux limites ou des conditions initiales.

Une telle représentation d'un objet physique est adaptée aux corps maillés par éléments finis, mais s'étend au cas des objets rigides. Un corps rigide peut être représenté par un unique élément "point", reposant sur un unique nœud : son centre d'inertie. Son mouvement est régi par les équations de Newton-Euler, qui relient réciproquement, vitesses de translation et résultantes d'une part et vitesses de rotation et moments des forces d'autre part. La formulation de ces équations nécessite des données géométriques supplémentaires : le volume de l'objet, pour le calcul des vitesses de translation ainsi que l'inertie et l'orientation de l'objet pour le calcul des vitesses de rotation. De plus, la nature de ces données change dans un cadre bidimensionnel (*e.g.* l'inertie devient scalaire). La classe représentant un élément générique (classe *bulk*) se spécialise ainsi en trois classes : une pour les éléments d'un maillage élément fini (classe *element*), une pour les corps rigides bidimensionnels (classe *rigid2D*) et une pour les corps rigides tridimensionnels (classe *rigid3D*).

La représentation discrète d'un objet physique est complétée par un ensemble d'objets (classe *contactor*) représentant les zones géométriques susceptibles d'entrer en contact, appelées contacteurs dans la suite. Chaque contacteur s'appuie sur un ou plusieurs éléments, selon le type de représentation de l'objet physique. Un contacteur utilisé pour un corps maillé éléments finis repose sur un ensemble d'éléments surfaciques (ou linéiques dans un cadre bidimensionnel). Un contacteur utilisé pour un corps rigide repose sur son unique élément "point" et contribue à la description géométrique du corps. Il peut ainsi être choisi parmi les primitives suivantes : sphère, cylindre (creux ou plein) ou polyèdre, dans un cadre tridimensionnel ; disque (creux ou plein) ou polygone, dans un cadre bidimensionnel. La description géométrique d'un objet complexe peut s'appuyer sur un ensemble de ces primitives. Une étiquette (ou "couleur") est attribuée à chaque contacteur afin de disposer d'un critère supplémentaire pour distinguer des familles de contacteurs.

La détection des interactions crée des paires de contacteurs telles que l'un est appelé "candidat" et l'autre "antagoniste", de sorte que la condition de non-pénétration au contact se formule ainsi : il est interdit à un contacteur candidat de pénétrer un contacteur antagoniste. De la combinatoire de toutes les paires candidat-antagoniste existantes dans une collection d'objets donnée, des classes d'équivalence sont extraites et décrites explicitement au travers d'objets "table de visibilité" (classe *see_table*). Chaque table de visibilité définit le type de paire candidat-antagoniste selon le type de contacteur candidat, le type de corps support (*i.e.* rigide ou maillé éléments finis) du contacteur candidat, le type de contacteur antagoniste et le type de corps support du contacteur antagoniste. La description d'une interaction associe un objet table de visibilité et un objet loi d'interaction (classe *tact_behav*). Chaque loi d'interaction définit le modèle considéré et les paramètres associés.

3 Mise en œuvre

Le pré-processeur est implémenté en langage Python, qui est interprété, orienté objet et dispose de modules adaptés au calcul scientifique, comme SciPy ou NumPy [8]. La structuration des données décrite dans le paragraphe précédent est représentée informatiquement par une hiérarchie de classes *ad hoc*, fournissant un ensemble de méthodes dédiées à la construction des différents objets. Elle est assortie d'un ensemble de classes et modules utilitaires détaillés dans la suite de cette section.

3.1 Réalisation du diagramme de classe

Les différents ensembles d'objets évoqués dans la section précédente (*e.g.* ensembles d'éléments, de nœuds ou de contacteurs) sont représentés informatiquement par des objets conteneurs (*e.g.* classe *bulks* pour les éléments, classe *nodes* pour les nœuds ou classe *contactors* pour les contacteurs). Toutes ces classes de conteneurs dérivent de deux classes génériques basées, pour l'une sur le type liste et pour l'autre sur le type dictionnaire de Python. Le type liste permet de construire un ensemble (ordonné) d'objets de toute nature, alors que le type dictionnaire permet de construire un ensemble de couples (clef, valeur), où la clef est un objet non-modifiable (*e.g.* un nombre ou une chaîne de caractères) et la valeur est un objet quelconque. Les conteneurs basés sur un dictionnaire sont donc réservés au stockage des objets porteurs d'un identifiant — typiquement les nœuds identifiés par leur numéro, sur lesquels sont basés les tables de connectivité des éléments — tandis que les conteneurs basés sur des listes suffisent pour les autres objets (*e.g.* les éléments). Les fonctions qui exportent les données manipulent des conteneurs. Les représentations des objets physiques de toute la collection sont donc toujours *in fine* stockées dans un conteneur *ad hoc* (classe *avatars*). Ces conteneurs sont aussi dotés de fonctions géométriques simples (*e.g.* translation, rotation) qui permettent de repositionner toute ou partie de la collection.

Le diagramme de classes de la FIGURE 1 montre que seule la classe *bulk* (représentant les éléments) a été spécialisée et non d'autre classes qui pourraient pourtant se prêter à un mécanisme d'héritage, à l'image de la classe *contactor* (représentant les contacteurs). Il s'agit d'un choix délibéré. Le pré-processeur de la plate-forme LMGC90 est aussi un logiciel de recherche qui doit répondre à un cahier des charges évoluant sans cesse. Il est donc nécessaire de pouvoir ajouter de façon simple et peu coûteuse en termes de développement de nouveaux types pour les objets (*e.g.* de nouvelles lois d'interaction). Le mécanisme d'héritage (propre aux langages de programmation orientés objet tels que Python) est alors délaissé au profit d'un mécanisme d'abonnement. Un module du pré-processeur regroupe un ensemble d'objets dictionnaire de Python, listant pour un objet d'une classe donnée les types disponibles et les options *ad hoc*. La syntaxe de Python permet de fournir une liste variable d'arguments à une fonction. Le constructeur de chaque classe utilisant ce mécanisme d'abonnement prend donc en arguments le type de l'objet ainsi que ses différentes options et se charge de vérifier la complétude et la cohérence des attributs du nouvel objet.

Les flèches en pointillés apparaissant sur le diagramme de classes de la FIGURE 1 traduisent la dépendance entre certaines classes. Elles indiquent par exemple que les paramètres matériau à renseigner et les degrés de liberté portés par un nœud dépendent du modèle choisi ; ou encore que n'importe quelle loi d'interaction ne peut pas être utilisée avec n'importe quel type de contacteur. Ces dépendances sont résolues par des tests de cohérence opérés lors de l'attribution des différentes propriétés aux différents objets. Ces tests se basent sur deux types de règles définissant soit la combinatoire des associations possibles (*e.g.* la liste des types de matériau compatibles avec chaque type de modèle), soit une hiérarchisation des options nécessaires à la définition d'un certain type d'objet (*e.g.* spécifier qu'on doit préciser la formulation choisie dans le cas d'un modèle déformable en grandes transformations). La formalisation des associations utilise des objets dictionnaire de Python, tandis que la formalisation des hiérarchies repose sur des arbres n-aire. La classe arbre n-aire utilisée a été implémentée en Python au sein d'un module spécifique.

3.2 Mécanisme d'affectation des propriétés

Les mailleurs permettent d'attribuer à une entité, faisant partie de la description géométrique d'un objet physique, un identifiant dont hérite par la suite les éléments discrétisant cette entité. Les éléments du maillage résultant peuvent ainsi être classés suivant leur appartenance à un "groupe" d'éléments portant

le même identifiant. Il semble alors naturel d'utiliser cette notion de "groupe" pour attribuer une loi de comportement (*i.e.* un objet modèle et un objet matériau) à un élément volumique ou un contacteur à un élément surfacique d'un corps maillé éléments finis ; ou encore une condition limite ou une condition initiale à un nœud de ce corps. Ce mécanisme a été implémenté de la façon suivante. Chaque objet représentant un objet physique porte un ensemble d'objets "groupe", chacun associé à un identifiant. Chaque objet "groupe" porte un ensemble d'éléments qui partagent un identifiant commun, ainsi que l'ensemble des nœuds sur lesquels s'appuient les éléments du groupe. Un objet représentant un objet physique possède une méthode qui construit automatiquement les groupes, utilisable une fois que tous les éléments et tous les nœuds impliqués dans sa discrétisation ont été définis. Un groupe spécial, composé de l'ensemble de tous les éléments et de l'ensemble de tous les nœuds de la représentation discrète du corps est aussi créé par cette méthode. Il fournit un cas par défaut aux méthodes d'affectation de modèles, de matériaux, de contacteurs, de conditions limites et de conditions initiales. Ce groupe spécial permet ainsi d'étendre simplement la notion de "groupe" aux corps rigides.

La description discrète d'un corps rigide repose sur la connaissance des coordonnées de son centre d'inertie (qui constitue son unique nœud) mais doit être complétée d'informations géométriques supplémentaires : son volume, son inertie et son orientation (ou seulement sa surface et son inertie dans un cadre bidimensionnel). Les contacteurs des objets rigides participant à la définition géométrique de l'objet, chaque primitive géométrique disponible dans le pré-processeur dispose d'une méthode qui calcule ses propriétés géométriques. Elle s'appuie sur des formules analytiques disponibles dans la littérature pour une primitive continue (*i.e.* la sphère, le cylindre ou le disque) ou une méthode numérique pour une primitive discrète (*i.e.* le polygone et le polyèdre). Le calcul des propriétés géométriques d'un polyèdre utilise notamment une méthode originale, basée sur l'intégration numérique de flux à travers la surface du contacteur (*cf.* section Annexe). Pour un corps rigide dont la géométrie est décrite par un ensemble de primitives, les propriétés géométriques du corps sont obtenues comme suit. Le volume du corps est la somme des volumes des primitives. Les coordonnées du centre d'inertie du corps sont obtenues comme la moyenne des coordonnées des centres d'inertie des primitives, pondérée par les volumes des primitives. Le calcul de l'inertie du corps est obtenu en appliquant la formule de Huygens.

La manipulation des maillages éléments finis s'appuie sur une classe utilitaire dédiée, possédant comme seuls attributs un ensemble de nœuds et un ensemble d'éléments. Chaque objet maillage dispose d'une méthode lui permettant de représenter l'objet qu'il discrétise sous la forme d'un corps maillé éléments finis, à partir d'un objet modèle et d'un objet matériau. Il possède aussi une méthode spécifique à la création d'une collection d'objets distincts, capable d'extraire d'un même maillage les dicrétisations de plusieurs objets physiques. Cette méthode se base sur la valeur d'une entité attribuée à chaque élément par le mailleur, typiquement l'identifiant de l'entité géométrique à laquelle il appartient. Deux modules fournissent des fonctions qui manipulent plus spécifiquement les maillages discrétisant respectivement des objets bidimensionnels et tridimensionnels. Elles permettent notamment de construire un corps rigide représentant l'objet physique entier ou de représenter les éléments volumiques de la discrétisation par des corps rigides ou des corps maillés éléments finis distincts.

3.3 Méthode générique de construction d'une collection d'objets

Le pré-processeur se présente donc comme une bibliothèque à partir de laquelle un utilisateur peut écrire un script Python pour générer sa collection d'objets. L'algorithme de construction de la représentation d'un objet physique découle de l'ordonnancement des tâches de construction des différents objets la composant. Il est représenté sur le diagramme de la FIGURE 2. D'un point de vue plus macroscopique, l'algorithme de construction de la collection entière passe par les étapes suivantes :

- définition du cadre considéré (*i.e.* bidimensionnel ou tridimensionnel),
- définition des différents conteneurs (pour stocker les modèles, les matériaux, *etc*),
- définition des modèles et des matériaux,
- définition de la représentation de chaque objet, en appliquant l'algorithme de la FIGURE 2,
- définition des lois de contact et des tables de visibilité,
- remplissage des conteneurs,
- transfert des données au module de calcul.

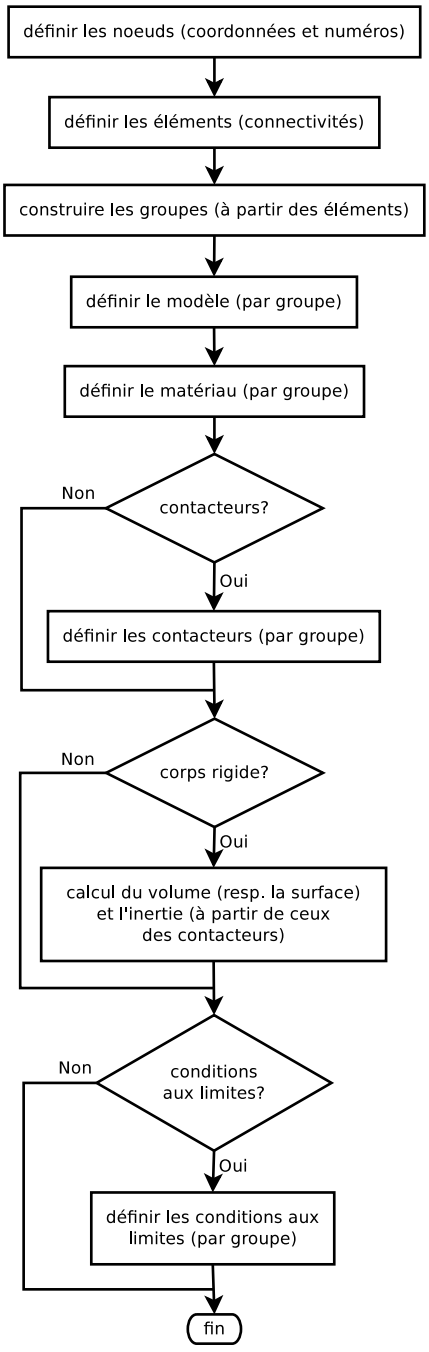
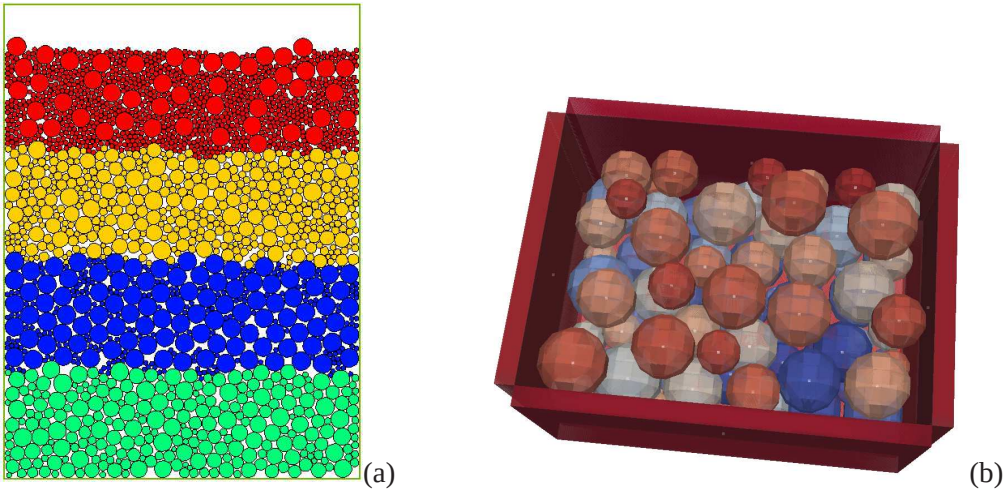


FIGURE 2 – Algorithme générique de construction de la représentation d'un objet physique.



(a) (b)

FIGURE 3 – Un échantillon composé de quatre couches de disques rigides — chaque couche suivant une granulométrie différente — déposées dans une boîte (a) et un échantillon polydisperse de sphères rigides déposées dans une boîte (b).

Il est à noter que des pré-processeurs spécifiques écrits de cette manière partagent une même structure de données générique et les routines d'exportation et peuvent faire intervenir des objets de différentes natures. Les limitations inhérentes aux programmes monolithiques : 1. réécriture systématique d'une structure de données et de routines d'exportation adaptées, 2. faible interopérabilité entre deux pré-processeurs différents ; sont ainsi levées.

4 Applications

La bibliothèque décrite dans la section précédente constitue une base pour le développement de pré-processeurs spécifiques. Les fonctions les plus coûteuses en termes de temps de calcul n'ont pas été réécrites en Python, mais placées dans des bibliothèques rendues accessibles par des modules compilés, grâce à SWIG [3].

4.1 Analyse des milieux granulaires

L'analyse des milieux granulaires (*cf.* FIGURE 3) met en jeu des post-traitements basés sur des méthodes statistiques. La qualité du résultat d'une étude dépend ainsi fortement de la représentativité statistique des échantillons analysés. Le premier enjeu du pré-processeur pour l'étude des milieux granulaires est donc sa capacité à produire des échantillons dont la granulométrie est maîtrisée. Selon le type d'étude réalisée, les résultats peuvent aussi dépendre de paramètres de texture à l'état initial tels que la compacité ou l'anisotropie des forces de contact. Un échantillon prêt à être analysé est donc toujours issu d'un premier calcul fournissant un état initial adapté à l'étude. Le pré-processeur est donc amené à raccourcir la durée de ce premier calcul, en proposant plusieurs méthodes de répartition des particules dans l'espace. Une méthode de dépôt géométrique minimisant l'énergie potentielle de pesanteur permet, par exemple, d'obtenir une configuration proche de l'état d'équilibre de l'échantillon soumis à son poids propre. Un échantillon présentant au contraire une répartition isotrope des forces de contact peut être obtenu à partir d'un calcul de compression isotrope, appliqué à une collection de particules disposées sur un réseau régulier. Enfin, la répartition des forces de contact dépendant aussi de la forme des particules, le pré-processeur doit permettre de définir le type et les paramètres caractérisant la forme des particules.

Le pré-processeur de la plate-forme LMGC90 dispose d'un module fournissant différents types de granulométrie, présentées sur la FIGURE 3(a) : une répartition aléatoire des tailles dans un intervalle donné (couche verte), une répartition des tailles dans un intervalle donné garantissant une répartition uniforme des volumes (couche orange), une répartition en deux catégories de tailles où le pourcentage du volume occupé par chaque catégorie est fixé (couches rouge et bleue), ou une granulométrie pré-calculée (*i.e.* lue dans un fichier ; non représentée sur la FIGURE 3(a)). Le pré-processeur dispose aussi

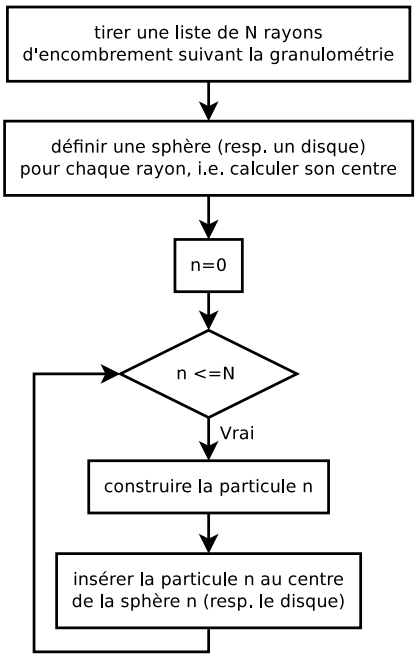
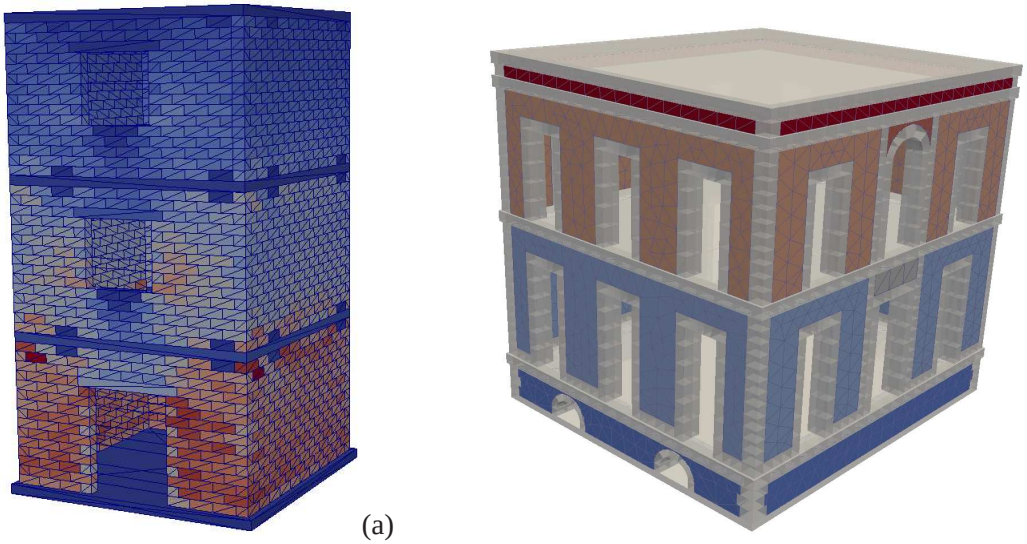


FIGURE 4 – Algorithme générique de construction d'un échantillon granulaire.

de modules dédiés à la génération de particules prédéfinies. Dans un cadre bidimensionel, différentes particules rigides sont disponibles : des disques, des polygones réguliers (dont le nombre de faces est paramétrable) et des clusters de disques (le nombre de disques dans un cluster est paramétrable). Il est à noter que les particules circulaires sont aussi disponibles sous la forme de corps maillés éléments finis. Dans un cadre tridimensionnel, seules les sphères rigides sont disponibles sous la forme de particules rigides prédéfinies. Un autre module du pré-processeur permet cependant d'extruder un échantillon bidimensionnel pour obtenir un échantillon tridimensionnel. Les disques peuvent alors devenir des sphères ou des cylindres (pleins) tandis que les polygones deviennent des polyèdres. Enfin, l'utilisateur peut à loisir définir ses propres fonctions pour générer un type de particule particulier, en se basant sur l'algorithme de construction de la représentation d'un objet physique présenté dans la section précédente (*cf.* FIGURE 2).

Les deux méthodes de répartition des particules dans l'espace introduites dans le premier paragraphe de la section (*i.e.* la méthode de dépôt géométrique minimisant l'énergie potentielle de pesanteur et la disposition des particules sur un réseau régulier) font l'objet de modules dédiés du pré-processeur. Ces méthodes de dépôt sont conçues pour positionner des particules sphériques (ou circulaires, dans un cadre bidimensionnel) connaissant leur rayon. Que l'une ou l'autre des deux méthodes de dépôt soit employée, le cas des particules non-sphériques (ou non-circulaires, dans un cadre bidimensionnel) est traité en utilisant leur rayon d'encombrement. Un échantillon granulaire obtenu en insérant des particules de formes quelconques dans des cavités sphériques présente une faible compacité initiale et peu de contacts. Il en résulte que le calcul d'un état initial pour ce type d'échantillon est plus long que pour un échantillon composé de sphères.

La méthode de dépôt géométrique est conçue pour réaliser des échantillons compacts, disposés dans divers conteneurs. Son implémentation est différente selon le cadre considéré. Dans le cadre bidimensionnel, le dépôt se déroule en deux étapes : 1. un premier échantillon est obtenu en déposant les particules dans une boîte rectangulaire et 2. un échantillon dont la forme s'insère au mieux dans le conteneur visé est extrait du premier échantillon. Il est ainsi possible de réaliser des dépôts dans des boîtes pleines rectangulaires ou circulaires. De plus, deux autres fonctions ont été développées afin de réaliser des remplisages partiels destinés à des essais spécifiques : analyse de tambours tournant et cisaillement de Couette. Enfin, la méthode de découpe faisant l'objet d'une fonction séparée, l'utilisateur peut définir la géométrie d'un conteneur particulier en décrivant son contour comme un polygone (ouvert ou fermé). Dans le cadre tridimensionnel, le dépôt des particules est réalisé directement dans le conteneur choisi : un parallélépipède, un cylindre ou une sphère. Dans tous les cas, le nombre de particules nécessaires pour remplir le conteneur visé est inconnu à l'avance. Il est donc tentant de partir d'un nombre de particules



(a) (b)

FIGURE 5 – Le résultat d'une descente de charge sur un bâtiment régulier dont chaque partie est représentée par un corps rigide (a) et une modélisation du musée Schœlcher (d'après [10]) mêlant blocs rigides (en transparence) et remplissages maillés éléments finis (en dégradé de couleurs) (b).

arbitrairement grand, de sorte que le conteneur soit rempli correctement *in fine* ; les particules superflues étant mises de côté par la méthode de découpe, ou n'étant tout simplement pas déposées. Cependant, ce tri arbitré par la méthode de dépôt ne permet pas de garantir que les particules de l'échantillon déposé vérifient encore la granulométrie initiale.

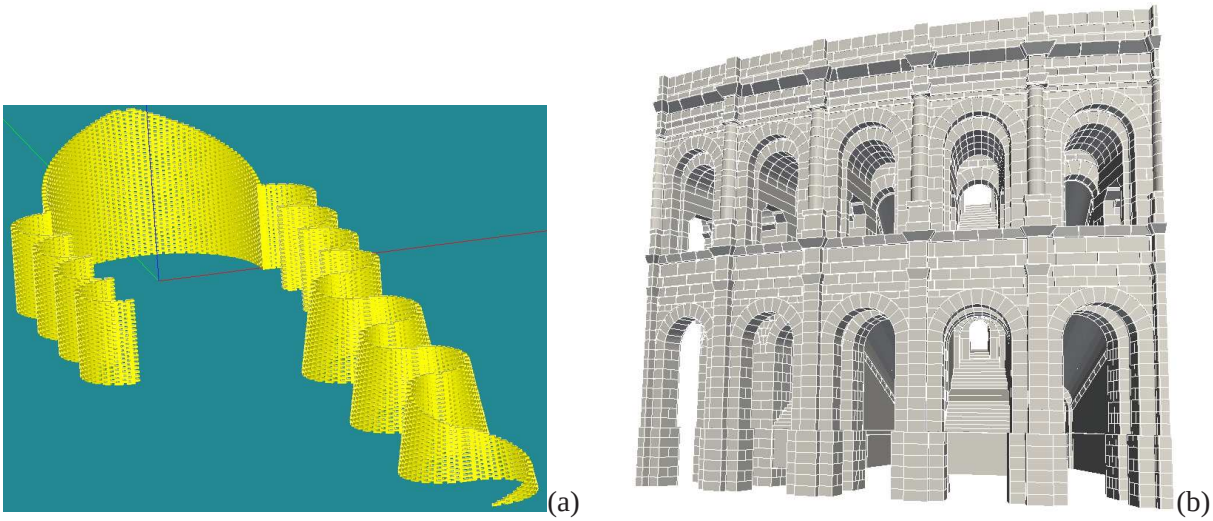
La disposition des particules sur un réseau est une méthode adaptée à la génération d'échantillons devant présenter une répartition isotrope des forces de contact et peut aussi permettre de remplir des conteneurs non-gérés par la méthode de dépôt géométrique. Dans un cadre bidimensionnel, les réseaux triangulaires et rectangulaires ont été implémentés, tandis que seul le réseau hexaédrique est disponible dans un cadre tridimensionnel. Contrairement à la méthode de dépôt géométrique, le nombre de particules du réseau peut être calculé à l'avance. Les échantillons obtenus par cette méthode respectent ainsi la granulométrie considérée. Cependant, ils présentent une plus faible compacité que ceux obtenus par la méthode de dépôt géométrique. Le calcul d'un état initial pour ce type d'échantillon est donc plus long.

Un algorithme générique de construction d'un échantillon granulaire découle naturellement de l'ordonnancement des tâches décrites précédemment : 1. tirage d'une liste de rayons suivant une granulométrie donnée, 2. répartition spatiale des particules en utilisant une méthode de dépôt et 3. construction des particules. Il est représenté sur la FIGURE 4.

4.2 Étude des maçonneries

Pour l'étude des maçonneries, un module proposant des classes d'objets "blocs" (bidimensionnels et tridimensionnels) a été écrit. Chaque objet "bloc" est décrit comme un parallélépipède rectangle (ou un rectangle dans un cadre bidimensionnel) et peut être représenté par un corps rigide ou un corps maillé éléments finis. L'utilisateur du pré-processeur peut ainsi écrire l'algorithme de construction adapté à ces besoins. La richesse des lois d'interaction disponibles dans la plate-forme LMGC90 lui permet, par exemple, de construire un mur où les joints peuvent se rompre et les briques se fissurer (suivant un faciès prédéfini), en utilisant des lois cohésives endommageables (type modèle de zones cohésives).

Afin de pouvoir étudier des bâtiments réguliers (*cf.* FIGURE 5(a)), un module de génération de bâtiments caractérisé par un ensemble de paramètres a été écrit. Sa conception repose sur une décomposition hiérarchique du bâtiment. Un immeuble peut ainsi être vu comme un ensemble d'étages liés entre eux par des diaphragmes présentant ou non des chaînages horizontaux. Un étage peut lui-même se décomposer en quatre murs (éventuellement assortis de refends) liés entre eux par harpage, chaînages verticaux... Un mur est quant à lui décrit au travers d'un ensemble de trumeaux liés entre eux par le hargape naturel de la maçonnerie, avec ou non des chaînages verticaux. Enfin, se distinguent parmi les trumeaux, les trumeaux "pleins" ou contenant une ouverture ; ce dernier type préfigurant une décomposition en allège,



(a) (b)

FIGURE 6 – La modélisation sous Salome d'une cathédrale réalisée aux Grands Ateliers de l'Ecole d'architecture de Paris Belleville (a) et une modélisation de cinq travées des arènes de Nîmes où les blocs sont représentés par des corps rigides (b).

ouverture et linteau. Cette décomposition se traduit informatiquement en relations dictant le processus de construction des objets d'une part (*e.g.* la construction d'un mur appelle la construction des trumeaux qui le composent) et définissant les tables de visibilité entre objets d'autre part.

Pour d'avantage de richesse morphologique, des modules complémentaires ont été développés pour interagir avec des outils de CAO ou manipuler des géométries exportées de ces outils (*cf.* FIGURE 5(b)). Ces modules sont basés sur l'interface Python du modeleur géométrique de la plate-forme libre Salome [9], *via* le module *geompy*. Un premier module est dédié à la génération de la représentation d'un solide (issu du modeleur de Salome ou importé) sous la forme d'un corps rigide. Le volume, la position du centre d'inertie et l'inertie du corps sont obtenus en utilisant les fonctions *ad hoc* du module *geompy*. Le corps est équipé d'un contacteur polyèdre obtenu en discrétisant l'enveloppe du solide grâce au mailleur libre Gmsh [7]. Une collection d'objets géométriques est représentée sous Salome sous la forme d'un objet *compound*. Le module *geompy* dispose de fonctions permettant d'ouvrir un *compound* et d'appliquer un traitement à tous les solides qu'il contient. Chaque solide contenu dans ce *compound* génère ainsi un corps rigide. Une géométrie exportée depuis un modeleur CAO peut aussi être directement importée et discrétisée par Gmsh. Une fois le maillage résultant importé par le pré-processeur, les méthodes de la classe maillage (*cf.* section 3.2) permettent de représenter chaque pièce ou l'intégralité de la structure sous la forme d'un corps rigide ou maillé éléments finis.

L'interface Python du modeleur de Salome a aussi permis d'écrire un module dédié à l'importation de fichiers DXF — le format d'échange fourni par le modeleur propriétaire AutoCAD — dans Salome. Ce module est issu de la modification de modules Python dédiés à l'importation de fichiers DXF (au format ASCII uniquement) dans le modeleur Blender et distribués sous une licence libre. Dans la version originale, un premier module lit le fichier DXF et remplit une base de données représentant son contenu, tandis qu'un second module la traite en créant des objets Python capables de se représenter sous Blender. Le premier module ne contient aucun appel à des fonctions spécifiques de Blender et n'a ainsi pas été modifié. Le second module a été repris pour que les objets Python créés puissent se représenter sous la forme d'objets géométriques de Salome. Il est à noter que les solides manipulés par AutoCAD sont exportés dans un fichier DXF sous une forme binaire non documentée et sont donc inexploitables. De plus, la vocation de Blender étant seulement de représenter des objets tridimensionnels, le module original ne vérifiait pas la cohérence de la description topologique de l'enveloppe des solides. La partie la plus importante du travail de modification du module a donc été la mise en place de procédures de nettoyage des maillages décrivant les enveloppes des objets (*e.g.* suppression des sommets et des faces redondants, "couture" des faces pour fermer les enveloppes), afin de rendre exploitables ces fichiers par Salome pour la création de solides.

Afin de pouvoir analyser des structures présentant de multiples courbures (*cf.* FIGURE 6(a)), une bibliothèque de formes paramétrées pFormes [1, 2] a été implémentée. L'utilisation conjointe de cette

bibliothèque et des modules dédiés à la manipulation de géométries issues d'outils de CAO présentés précédemment, a permis d'importer dans LMGC90 un modèle discret de cinq travées des arènes de Nîmes (*cf.* FIGURE 6(b)) et ainsi de réaliser un calcul de descente de charge pour ces cinq travées.

5 Conclusion

L'élaboration du pré-processeur de la plate-forme LMGC90 a été présentée, depuis la conception d'une structure de donnée générique adaptée à la représentation d'un objet physique jusqu'aux exemples d'application. De nombreux détails de mise en œuvre ont été abordés afin de justifier les choix techniques (*e.g.* le choix du langage Python) et de montrer la façon dont les modules de ce pré-processeur générique peuvent être exploités pour écrire de nouvelles classes de pré-processeurs spécifiques. L'écriture de modules dédiés à l'étude des milieux granulaires et des maçonneries a ainsi été discutée.

La capacité du pré-processeur à générer la représentation d'une collection d'objets physiques, sous la forme de corps rigides ou maillés éléments finis, pouvant avoir des formes complexes, faire intervenir diverses physiques et interagir selon diverses lois d'interaction a été démontrée. Cependant, la définition des tables de visibilité utilisées lors de la détection des interactions est laissée à la discrétion de l'utilisateur. Ceci devient problématique quand les limites des méthodes de détection d'interactions sont atteintes. La capacité du code de calcul à sélectionner uniquement les interactions ayant un sens physique repose alors entièrement sur la définition faite par l'utilisateur des tables de visibilité excluant les interactions aberrantes.

Le prochain défi que doit relever le pré-processeur est donc la définition automatique de tables de visibilité capables d'exclure les interactions aberrantes et de soulager ainsi l'utilisateur d'une tâche pénible et source d'erreurs. L'idée envisagée pour y parvenir est d'introduire la notion d'entité. Une entité représente directement un objet physique et dispose de méthodes lui permettant de se représenter sous la forme d'un corps rigide ou maillé éléments finis. Une entité dispose aussi de méthodes de subdivision lui permettant de se définir elle-même comme une collection d'entités en interaction (*e.g.* un mur peut se subdiviser en blocs). Les tables de visibilité entre entités atomiques (*i.e.* non-subdivisées) découlent ainsi des subdivisions successives dont ces entités atomiques sont issues.

Remerciements Les auteurs remercient les organismes qui financent le développement de la plate-forme LMGC90 : l'ANR qui finance le pojet Saladyn, d'une part et FEDER, OSEO et la région Languedoc-Roussillon qui financent le projet Degrip, d'autre part.

Annexe

La méthode de calcul des propriétés géométriques d'un polyèdre utilisée par le pré-processeur de la plate-forme LMGC90 est détaillée dans cette section. Soit P un polyèdre occupant le domaine Ω de l'espace et $\partial\Omega$ la frontière de Ω. Le volume V du polyèdre P est défini par

$$V = \int_\Omega dV. \tag{1}$$

Soit $\underline{f} : \mathbb{R}^3 \to \mathbb{R}^3$ une fonction à valeur vectorielle telle que $\operatorname{div} \underline{f} = 1$. De l'application du théorème de la divergence à l'équation (1), il vient

$$V = \int_{\partial\Omega} \underline{f} \cdot \underline{n}\, dS, \tag{2}$$

où $\underline{n} \in \mathbb{R}^3$ est la normale à la surface $\partial\Omega$.

On choisit $\underline{f}$ définie par $\underline{f}(\underline{x}) = (x,0,0)^T$, où $\underline{x} = (x,y,z)^T$. Le calcul de V se ramène finalement au calcul de l'intégrale de surface suivante :

$$V = \int_{\partial\Omega} (x,0,0)^T \cdot \underline{n}\, dS. \tag{3}$$

Le même raisonnement peut être appliqué au calcul des coordonnées du centre d'inertie et de la matrice d'inertie du polyèdre. Soit $\underline{x_G} = (x_G, y_G, z_G)^T$ les coordonnées du centre d'inertie G du polyèdre.

Le calcul des coordonnées de G se ramène alors (par exemple) aux trois calculs d'intégrale de surface suivants :

$$x_G = \frac{1}{V}\int_\Omega x\, dV = \frac{1}{V}\int_{\partial\Omega} \frac{1}{2}(x^2,0,0)^T \cdot \underline{n}\, dS, \tag{4}$$

$$y_G = \frac{1}{V}\int_\Omega y\, dV = \frac{1}{V}\int_{\partial\Omega} \frac{1}{2}(0,y^2,0)^T \cdot \underline{n}\, dS, \tag{5}$$

$$z_G = \frac{1}{V}\int_\Omega z\, dV = \frac{1}{V}\int_{\partial\Omega} \frac{1}{2}(0,0,z^2)^T \cdot \underline{n}\, dS. \tag{6}$$

Soit

$$\underline{\underline{I}} = \begin{pmatrix} I_{11} & I_{21} & I_{31} \\ I_{21} & I_{22} & I_{32} \\ I_{31} & I_{32} & I_{33} \end{pmatrix}, \tag{7}$$

la matrice d'inertie du polyèdre exprimée en G. De même, le calcul des six composantes indépendantes de $\underline{\underline{I}}$ se ramène alors (par exemple) aux six calculs d'intégrale de surface suivants :

$$I_{11} = \int_\Omega \left[(y-y_G)^2 + (z-z_G)^2\right]\, dV = \int_{\partial\Omega} \frac{1}{3}(0,(y-y_G)^3,(z-z_G)^3)^T \cdot \underline{n}\, dS, \tag{8}$$

$$I_{22} = \int_\Omega \left[(x-x_G)^2 + (z-z_G)^2\right]\, dV = \int_{\partial\Omega} \frac{1}{3}((x-x_G)^3,0,(z-z_G)^3)^T \cdot \underline{n}\, dS, \tag{9}$$

$$I_{33} = \int_\Omega \left[(x-x_G)^2 + (y-y_G)^2\right]\, dV = \int_{\partial\Omega} \frac{1}{3}((x-x_G)^3,(y-y_G)^3,0)^T \cdot \underline{n}\, dS, \tag{10}$$

$$I_{32} = -\int_\Omega (y-y_G)(z-z_G)\, dV = -\int_{\partial\Omega} \frac{1}{2}(0,(y-y_G)^2(z-z_G),0)^T \cdot \underline{n}\, dS, \tag{11}$$

$$I_{31} = -\int_\Omega (x-x_G)(z-z_G)\, dV = -\int_{\partial\Omega} \frac{1}{2}((x-x_G)^2(z-z_G),0,0)^T \cdot \underline{n}\, dS, \tag{12}$$

$$I_{21} = -\int_\Omega (x-x_G)(y-y_G)\, dV = -\int_{\partial\Omega} \frac{1}{2}((x-x_G)^2(y-y_G),0,0)^T \cdot \underline{n}\, dS. \tag{13}$$

Au sein du pré-processeur, un polyèdre est décrit par un maillage de surface, composé exclusivement de triangles. Les intégrales de surface des équations (3), (4) à (6) et (8) à (13) peuvent donc être évaluées numériquement par une méthode de quadrature [4].

Références

[1] M. Bagnéris, A. Marty, B. Maurin, R. Motro et N. Pauli. *Pascalian forms as morphogenetic tool.* Journal of the International Association for Shells and Spatial Structures, 51 :165–181, 2010.

[2] M. Bagnéris, F. Dubois, A. Martin, R. Mozul et P. Taforel. *De l'intérêt de la courbure dans les interfaces : application aux structures en pierre.* In 10ème Colloque National en Calcul des Structures, Giens, 2011.

[3] D. M. Beazley. *Automated scientific software scripting with SWIG.* Future Generation Computer Systems, 19(5) :599–609, 2003.

[4] G. Dhatt et G. Touzot. *Une présentation de la méthode des éléments finis.* Maloine S.A. Editeur Paris et Les presses de l'Université de Laval Québec, 1981.

[5] F. Dubois et M. Jean. *LMGC90 une plateforme de développement dédiée à la modélisation des problèmes d'interaction.* In M. Potier-Ferry, M. Bonnet, et A. Bignonnet, éditeurs, 6ème Colloque National en Calcul des Structures, volume 1, pages 111–118, Giens, 2003.

[6] F. Dubois, M. Jean, M. Renouf, R. Mozul, A. Martin et M. Bagnéris. *LMGC90.* In 10ème Colloque National en Calcul des Structures, Giens, 2011.

[7] C. Geuzaine et J.-F. 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 :1309–1331, 2009.

[8] T. E. Oliphant. *Python for Scientific Computing.* Computing in Science and Engineering, 9 :10–20, 2007.

[9] A. Ribes et C. Caremoli. *Salome platform component model for numerical simulation.* 31st Annual International Computer Software and Applications Conference, 2 :553–564, 2007.

[10] P. Taforel, F. Dubois, A. Martin et S. Pagano. *Développement de formulations dynamiques adaptées à la modélisation par éléments discrets de structures maçonnées sous chargements sismiques.* In 10ème Colloque National en Calcul des Structures, Giens, 2011.