Master M1 Sciences de la Matière – ENS de Lyon – 2023-2024

Physique Numérique / Computational Physics

Formation des cratères : simulation numérique d'un milieu granulaire

David FANG

4 novembre 2024

Résumé

L'objet de ce projet est l'étude de la formation des cratères dans un milieu granulaire. Ce rapport explore en détail la méthode des éléments discrets (DEM) pour la simulation d'un milieu granulaire.

Table des matières

1 Motivation

L'objet de ce projet est l'étude de la formation des cratères. Comment la forme et la taille d'un cratère varient-elles selon les caractéristiques du projectile et du milieu impacté ? Des expériences à échelle humaine sont réalisables, mais elles produisent parfois des résultats contradictoires, et l'invariance d'échelle pour les cratères réels reste incertaine. Une simulation numérique pourrait fournir des réponses complémentaires.

2 Approche : méthode des éléments discrets

Il existe deux approches principales pour simuler la formation des cratères : une approche granulaire et une approche hydrodynamique. Ici, l'approche granulaire a été retenue, car la nature granulaire du milieu, comme pour des tas de sable, est souvent cruciale dans les expériences.

Pour simplifier l'étude, seuls des projectiles sphériques et un milieu constitué de sphères identiques sont considérés. Cette invariance par rotation réduit le problème à deux dimensions spatiales, ce qui équivaut en pratique à confiner le milieu granulaire dans un plan.

Représenter le milieu par un nombre fini de sphères s'inscrit dans la méthode des éléments discrets, introduite par Cundall en 1979 [2].

L'hypothèse fondamentale dans les simulations d'écoulement granulaire, et de manière générale pour la majorité des simulations de particules, est que le comportement qui émèrge à l'échelle macroscopique résulte d'une multitude d'interactions à deux corps à l'échelle de chaque particule. Comment modéliser l'interaction inter-particules dans un milieu granulaire ?

3 Interactions inter-particules : modèle de Kelvin-Voigt

Lorsque deux particules entrent en contact, celles-ci se déforment : une force élastique dans la direction normale au contact apparaît et tend à les repousser. Cette déformation est difficile à modéliser, d'autant plus que cela imposerait de modifier la taille des particules lors de la simulation.

Dans le modèle de Cundall [2], un chevauchement virtuel est introduit entre les particules en contact. Cela permet de garder une taille fixe pour les particules, et de rendre la force élastique proportionnelle au chevauchement.

Par la suite, il n'est pas difficile de complexifier la modélisation de la force, en ajoutant par exemple une force visqueuse d'amortissement : c'est le modèle de Kelvin-Voigt. Une force de Kelvin-Voigt peut aussi être ajoutée dans la direction tangentielle.

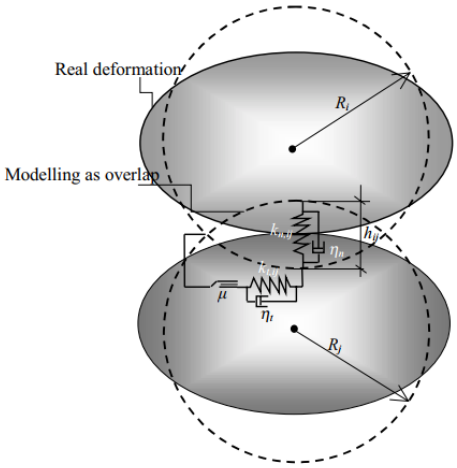

FIGURE 1 – Un modèle de contact inter-particules [1]

La force appliquée à une particule i est alors la somme des forces élastiques et visqueuses exercées par toutes les autres particules j en contact avec i.

$$\begin{aligned}
\vec{F}^{(i)} &= \vec{F}_n^{(i)} + \vec{F}_t^{(i)} + m\vec{g} \\
\vec{F}_n^{(i)} &= \sum_j \left[\vec{F}_{\text{elastique},n} + \vec{F}_{\text{visqueuse},n}\right] \\
\vec{F}_t^{(i)} &= \sum_j \left[\vec{F}_{\text{elastique},t} + \vec{F}_{\text{visqueuse},t}\right]
\end{aligned}$$

La composante normale est simplement donnée par

$$\vec{F}_{\text{elastique},n} = -kh_{ij}\vec{n} \tag{3.0.1}$$

$$\vec{F}_{\text{visqueuse},n} = -\eta(\vec{v}_{ij} \cdot \vec{n})\vec{n} \tag{3.0.2}$$

où v_ij est la vitesse relative de i par rapport à j et h_{ij} est le chevauchement virtuel. Les valeurs de k et η sont déterminées par les propriétés élastiques du matériau [4].

La composante tangentielle est plus difficile à modéliser. Celle-ci est négligée dans le cadre de cette simulation mais des expressions relativement simples existent [3].

4 Implémentation du programme

Le problème est décrit par la donnée de N particules, chacune ayant une position, une vitesse et une rotation variant dans le temps. L'état initial est donné par une distribution aléatoire des particules dans un domaine donné, en s'assurant que les particules ne se chevauchent pas. Puis on utilise un schéma d'intégration explicite pour résoudre les équations du mouvement. Pour cela, il faut à chaque instant, pour chaque particule,

1. Détecter quels particules sont en contact avec elle

2. Calculer la force totale $\vec{F}$ qui s'exerce sur elle en utilisant les expressions précédentes
3. Intégrer la vitesse de la particule $\vec{v}(t+dt) = \vec{v}(t) + \vec{F}dt/m$
4. Intégrer la positsion de la particule $\vec{r}(t+dt) = \vec{r}(t) + \vec{v}(t)dt/m$

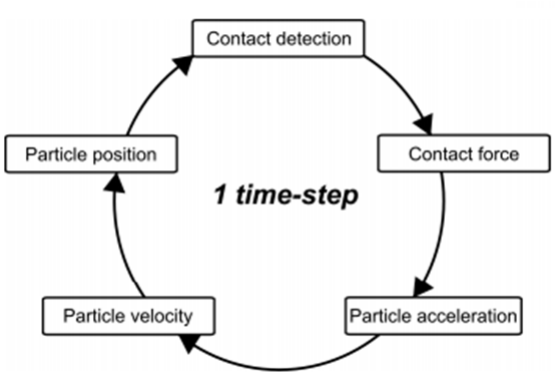

FIGURE 2 – Un cycle de calcul DEM [1]

Pour simplifier le problème, toutes les particules sont considérées comme identiques, suivant les caractéristiques suivantes :

Caractéristique	Valeur
Densité (ρ)	$2.7 \times 10^3\,\mathrm{kg.m^{-3}}$
Module de Young (E)	$94 \times 10^9\,\mathrm{Pa}$
Coefficient de Poisson (ν)	0.17
Pesanteur (g)	$9.81\,\mathrm{m.s^{-2}}$
Rayon (r)	$1 \times 10^{-3}\,\mathrm{m}$

TABLE 1 – Caractéristiques des grains de sable

Ces caractéristiques correspondent à des grains de sable de quartz.

Cet algorithme a l'avantage d'être simple sur le principe. Cependant, il devient rapidement très coûteux en temps de calcul : pour un petit tas de sable de 1L, cela équivaut à environ $N \sim 10^6$ particules. Comment détécter les voisins efficacement ? Quel pas de temps d'intégration choisir ?

4.1 Pas de temps

Le pas de temps d'intégration doit être ni trop long, ce qui conduirait à des imprécisions non négligeables, ni trop court, ce qui conduirait à des calculs inutiles.

La plupart des auteurs s'accordent sur l'existence d'une durée critique

$$\Delta t_{\text{critique}} = \sqrt{\frac{m}{k}} \tag{4.1.1}$$

Cela correspond à la durée nécessaire pour qu'une onde élastique se propage d'une particule à l'autre, ou de manière équivalente, à la durée de déformation d'une particule

lors d'un choc. Si le pas de temps d'intégration est plus grand que cette durée, alors les particules ne peuvent pas réagir à temps et la force est calculée "trop tard". Dans mon programme, il a été effectivement observé que les particules étaient systématiquenent éjectées de manière violente lorsque le pas de temps est inférieur à la durée critique.

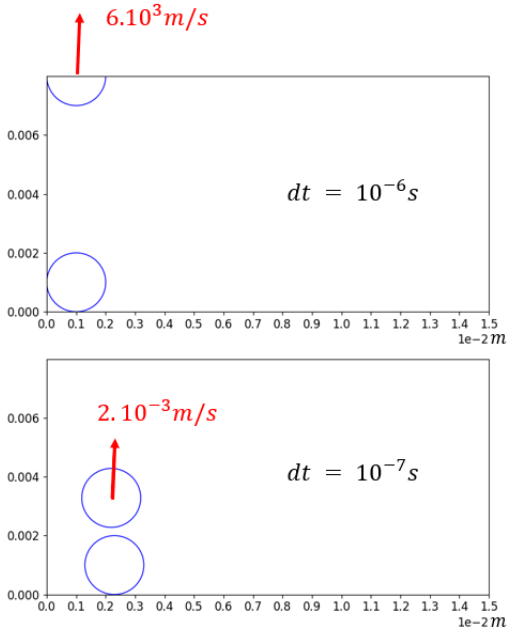

FIGURE 3 – Impact de 2 particules pour deux pas de temps différents, d$t = 10^{-5}$s et d$t = 10^{-6}$s, pour les mêmes conditions initiales.

Cette contrainte est très forte, car pour les grains de sable que nous considérons, la durée critique est de l'ordre de 10^{-6}s, donc il faut choisir d$t \leq 10^{-7}$ Or, la durée d'un impact d'un cratère est de l'ordre de 1s : cela signifie qu'il faut réaliser 10^7 intégrations pour simuler un impact. Pour une distribution de $N = 10^6$ particules, cela correspond à 10^{13} opérations, ce qui est irréalisable sans optimisation.

4.2 Détection des contacts

À chaque instant, il est nécessaire de détecter quelles particules sont en contact avec une particule donnée pour calculer les forces appliquées sur celle-ci.

La condition de contact entre une particule i et j est

$$|\vec{r}_i - \vec{r}_j| \leq R$$

La méthode la plus intuitive pour détecter les contacts est la suivante : pour la particule donnée, on parcourt la totalité des $N-1$ autres particules et on regarde si la condition de contact est vérifiée. Cet algorithme est irréalisable pour un nombre macroscopique de particules, car sa complexité est de l'ordre de $O(N^2)$. Il est possible de réduire la complexité à $O(N)$ en utilisant une grille de détection.

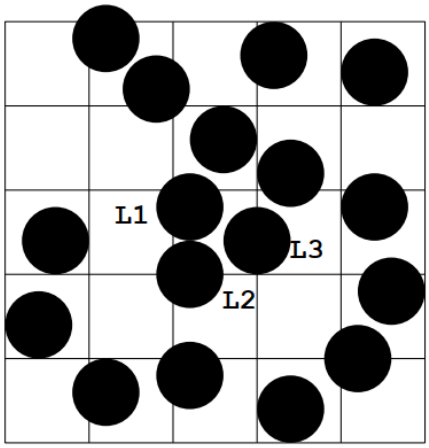

FIGURE 4 – Une grille de détection

L'espace est divisée en cellules de taille égales d_{cell}. Plusieurs choix sont possibles. Par exemple, si $d_{\text{cell}} \leq d$, les particules en contact avec i sont forcément contenues dans les 24 cellules voisines. Si $d_{\text{cell}} > d$, alors celles-ci sont forcément contenues dans les 8 cellules voisines. Mais dans ce cas, il faut s'assurer que la taille de la cellule n'est pas trop grande devant le diamètre des grains, auquel un nombre trop grand de grains peut être présent dans chaque cellule, ce qui fait perdre l'interêt de la grille puisque le nombre de calculs devient trop grand.

Il existe en fait de nombreuses variantes de grilles de détection. Dans tous les cas, l'avantage est non seulement de réduire la complexité tout en pouvant traiter des particules de taille différente.

4.3 Conditions aux bords

Au niveau des parois de l'enceinte, il est nécessaire de modifier le comportement des particules. Par simplicité, l'algorithme implémenté considère simplement que la vitesse normale y est nulle, ce qui signifie que les particules rebondissent de manière parfaitement élastique contre les parois. Il est néanmoins assez facile d'implémenter des conditions aux bords plus complexes, avec par exemple un frottement solide sur la composante tangentielle, ou même de considérer des interactions électrostatiques dans les modèles les plus complexes [3].

5 Parallélisation

L'algorithme de calcul des forces entre particules devient très coûteux en temps de calcul lorsque le nombre de particules est grand. Pour surmonter cette limitation, la parallélisation de l'exécution de l'algorithme est une solution tentante, permettant d'exploiter les différents cœurs du processeur et de réduire significativement le temps de calcul.

Python, cependant, ne permet pas de réaliser des calculs en parallèle de manière efficace en raison du Global Interpreter Lock (GIL). Le GIL est un verrou qui permet

à un seul thread d'exécuter du code Python à la fois, même sur un processeur multi-cœurs. Pour contourner cette limitation, on peut utiliser le module `multiprocessing`, qui crée des processus séparés, chacun avec son propre GIL.

La première approche explorée a consisté à attribuer des particules différentes à différents cœurs. Cependant, cette méthode s'est avérée peu avantageuse car, à chaque pas de temps, il est nécessaire de synchroniser la mémoire partagée entre les processus, ce qui s'est avéré être plus coûteux que de réaliser les calculs en série.

Ainsi, certains auteurs [3] suggèrent de diviser l'espace en plusieurs parties indépendantes, en supposant que les particules présentes dans chaque partie restent les mêmes. Par exemple, pour un processeur de 16 cœurs, il est possible de diviser l'espace en 16 colonnes, gagnant ainsi un facteur 16 sur le temps d'exécution du programme. Cette méthode est peu pertinente la formation d'un cratère, où les particules peuvent se déplacer entre différentes colonnes, et ejectées de manière assez violente.

Une implémentation en C ou C++ pourrait améliorer les performances, mais Python a l'avantage de fournir un code clair et lisible. Mais la conversion d'un code initialement en Python s'avère souvent difficile.

Enfin, la parallélisation est mieux exploitée lorsqu'elle est implémentée sur le GPU. En Python, il est possible d'exploiter l'architecture CUDA des GPU Nvidia pour gagner un temps de calcul encore plus important. Cependant, cette tentative n'a pas abouti en raison de la complexité de l'architecture CUDA. De plus, une implémentation en C++ est souvent plus pratique et efficace.

6 Simulation

En raison du très grand nombre de pas de temps nécessaire, il n'a pas été possible de simuler des tailles de l'ordre du mètre, demandant tout de même un pas de temps de $10^{-4}s$ et plus de 100 000 pas de temps avant d'observer une réelle évolution du milieu. Pour ajouter l'impact du projectile, il suffit d'ajouter une particule de taille significativement plus grande, avec une condition initiale différente. Cependant, la vitesse d'impact de celle-ci étant très grande, il est nécessaire de choisir un pas de temps encore plus petit pour éviter les instabilités. Malheuresement, en gardant un pas de temps raisonnable, l'algorithme a systématiquement échoué lors de l'impact : les particules sont éjectées trop violemment. Néanmoins, la simulation du milieu granulaire a été réalisée avec succès, malgré le temps nécessaire pour réaliser les calculs.

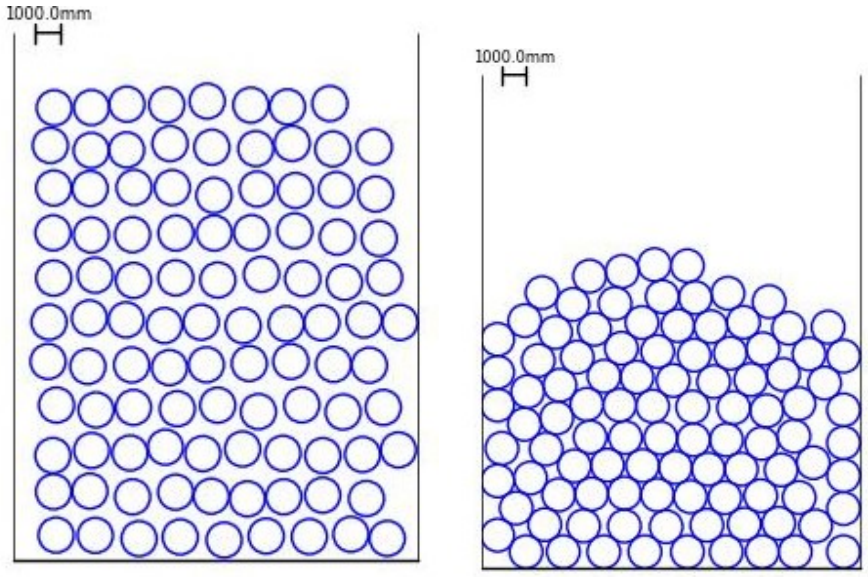

FIGURE 5 – Evolution du lit granulaire au bout de 100 000 pas de temps

Il est même possible de faire apparaître des lignes de forces en affichant la force appliquée sur chaque particule. D'autres résultats ainsi que l'animation de cette évolution sont accessibles dans le Github. Il s'avère que les résultats seraient beaucoup plus intéressants si les grains sont de tailles différentes.

7 Conclusion

La méthode des éléments discrets est un outil puissant pour simuler le comportement des milieux granulaires et peut être appliquée également aux problèmes de mécanique des fluides et des solides. Cependant, pour des simulations à petites échelles spatiales, telles que la formation de cratères de l'ordre du millimètre, la DEM reste extrêmement coûteuse en temps de calcul, même en optimisant la détection des voisins. Cette difficulté provient principalement du grand nombre de pas de temps nécessaires pour suivre les interactions entre particules.

La parallélisation de l'algorithme semble donc indispensable pour réduire les délais d'exécution et rendre ces simulations réalisables en temps raisonnable. L'implémentation sur GPU ou le recours à des langages de bas niveau pourrait offrir des gains significatifs, permettant de tirer plein potentiel du matériel disponible.

Références

[1] Robertas Balevičius, Algis Džiugys, and Rimantas Kačianauskas. Discrete element method and its application to the analysis of penetration into granular media. *Journal of Civil Engineering and Management*, 10(1) :3–14, 2004.

[2] Peter A Cundall and Otto DL Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1) :47–65, 1979.

[3] GA Kohring. Dynamical simulations of granular flows on multi-processor computers. *Computational methods in applied sciences*, 96 :190–196, 1996.

[4] Koji Wada, Hiroki Senshu, and Takafumi Matsui. Numerical simulation of impact cratering on granular material. *Icarus*, 180(2) :528–545, 2006.