Modélisation numérique des milieux granulaires.

I. Présentation des milieux granulaires

L'étude des écoulements granulaires est un domaine scientifique en pleine expansion. Assez solides pour soutenir le poids d'un immeuble, ils peuvent couler comme de l'eau dans un sablier ou être transportés par le vent pour sculpter les dunes et les déserts.

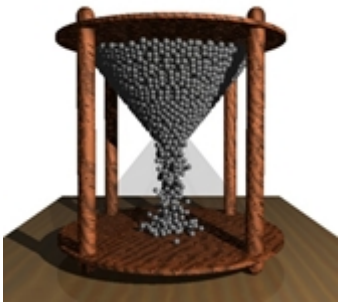


FIGURE 1 – Modélisation d'un écoulement dans un sablier.

On appelle milieu granulaire une collection de particules solides macroscopiques, typiquement de taille supérieure à 100 μm. Les particules plus fines sont appelées poudres (entre 1 μm et 100 μm) ou colloïdes (entre 1 nm et 1 μm). Pour étudier ces milieux, on néglige en général les interactions de van der Waals, les effets d'humidité et l'agitation thermique.

Une des principales motivations de l'étude des milieux granulaires est leur rôle dans de nombreux secteurs industriels. On estime en effet que plus de 50% des produits vendus dans le monde mettent en jeu des matériaux granulaires, soit dans leur élaboration, soit dans leur forme finale. Les milieux granulaires sont omniprésents dans l'activité minière (extraction des minerais, transport,...), le bâtiment et le génie civil (béton, ballast de train,...), l'industrie chimique et pharmaceutique, etc. Dans tous ces secteurs se posent des problèmes de stockage, de transport, d'écoulement, de mélange et de transformation, qu'on étudie à l'aide de simulations numériques.

FIGURE 2 – Dunes au Maroc.

Un autre grand domaine où les matériaux granulaires sont omniprésents est la géophysique, le sol étant principalement formé de grains. La nature offre ainsi les exemples les plus spectaculaires de phénomènes et de structures où interviennent les milieux granulaires : dunes de sable, plages s'étirant le long des côtes, éboulis, avalanches de neige, figures d'érosion... Ces exemples ne se limitent d'ailleurs pas à la Terre. Les dunes martiennes, les astéroïdes ou les anneaux de Saturne constitués de blocs et de poussières de glace illustrent l'ampleur des situations faisant intervenir la matière en grains. Là aussi, la simulation numérique permet l'étude des phénomènes observés.

II. Calcul de la forme d'un tas par automates cellulaires

L'objet de cette partie est la simulation numérique par un modèle de type automate cellulaire. Un automate cellulaire se présente généralement sous la forme d'un quadrillage dont chaque case peut être occupée ou vide. Un grain y est symbolisé par une case occupée. La configuration des cases, qu'on appelle état de l'automate, évolue au cours du temps selon certaines règles très simples permettant de reproduire des comportements extrêmement complexes. La physique n'intervient pas directement mais les règles d'évolution sont choisies de façon à reproduire au mieux les lois naturelles. Dans ce qui suit, nous allons simuler la formation d'un demi-tas de sable situé à droite de l'axe de symétrie vertical en appliquant les règles énoncées ci-après.

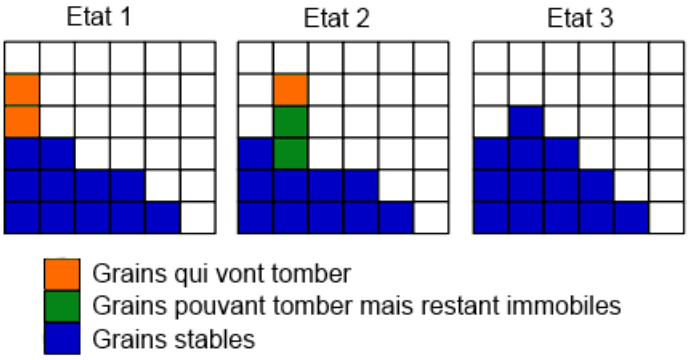


FIGURE 3 – Exemple d'évolution de l'automate cellulaire à implémenter.

Une tour de hauteur h est une pile de cases pleines consécutives dont h voisines de droite sont vides. Si h $>$ 1, on détermine arbitrairement le nombre n de grains du sommet de la tour qui vont tomber avec l'instruction python (version 2.7) :

```
n = int((h+2.0)/2 * random()) + 1
```

Dans l'exemple figure 3, lors du passage de l'état 1 à l'état 2, le hasard introduit par la fonction `random()` (qui renvoie de manière aléatoire un flottant dans l'intervalle semi-ouvert $[0.0, 1.0[$) fait que toute la tour se décale apparemment vers la droite. Ensuite, pour le passage à l'état 3 seul un grain sur les trois possibles tombe. L'état 3 est stable, plus aucun grain ne tombe.

❑ **Q1** – Quel est le type de n ? Déterminer un encadrement de n pour h $>$ 1

❑ **Q2** – Ecrire une fonction nommée `calcul_n` qui prend la hauteur `h` comme argument et qui renvoie le nombre `n` de grains qui vont tomber sur la pile suivante.

La représentation graphique est une image en deux dimensions mais l'automate est à une dimension car on considère le tas comme un ensemble de piles dont la hauteur future dépend uniquement des piles adjacentes. Le tas est complètement défini avec la variable **piles** qui permet de stocker la hauteur des différentes piles.

Au départ le support est vide et on vient déposer périodiquement un grain sur la première pile (à gauche) qui correspondra au sommet du demi-tas. Le support peut

recevoir `P` piles et à son extrémité droite il n'y a rien, les grains tombent et sont perdus. La pile `P+1` est donc toujours vide.

❑ **Q3** – Définir la fonction `initialisation(P)` renvoyant une variable `piles` de type liste contenant `P` piles de hauteur 0.

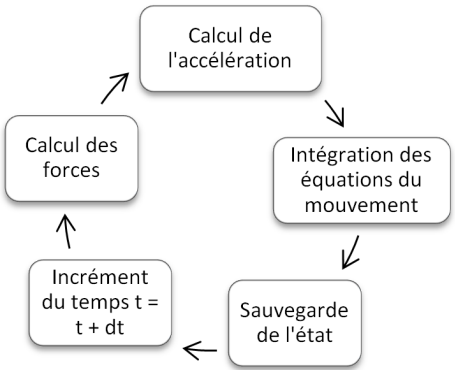
❑ **Q4** – Définir une fonction `actualise(piles,perdus)` qui va parcourir les piles de gauche à droite et les faire évoluer en utilisant les règles, fonctions et variables définies précédemment. Cette fonction doit renvoyer la variable `piles` actualisée ainsi que le nombre de grains perdus (en prenant en compte ceux qui seront tombés de la dernière pile `P`).

❑ **Q5** – Écrire le bloc d'instructions du programme principal qui permet d'ajouter 1 grain à la première pile après chaque dizaine d'exécutions de la fonction `actualise(,)` et qui s'arrêtera lorsqu'au moins 1000 grains seront sortis du support. Le nombre de piles (taille du support `P`) sera demandé à l'utilisateur lors de l'exécution. Vous utiliserez les fonctions et variables définies précédemment.

❑ **Q6** – Comment tracer simplement la forme (allure) du demi-tas à la fin de la simulation précédente.

III. Méthode des éléments distincts (DEM)

Pour remédier à certaines limitations des modèles par automates cellulaires, il est possible de modéliser les milieux granulaires par la méthode des éléments distincts DEM issue des modèles moléculaires et présentée par Cundall en 1971. Les grains sont considérés comme des corps isolés et les interactions comme des forces extérieures. Le mouvement de chaque grain est donné par les équations de la dynamique.



La boucle principale simplifiée de l'algorithme DEM est présentée ci-contre. Nous limitons l'étude à des éléments 2D (disques) et tous les grains sont supposés semblables : ils ont donc tous les mêmes caractéristiques.

Un des principaux inconvénients de la méthode est la détermination des paramètres physiques qui permettront une bonne corrélation entre la simulation et l'expérimentation. Différents laboratoires de recherche ont créé une base de données commune qui leur permet de partager des jeux de paramètres et de les noter de 1 à 5 selon les résultats obtenus lors des simulations. Une fonction d'agrégation permet de calculer la note moyenne. Il est donc possible d'effectuer des requêtes de recherche dans la relation « granular_base » définie par la table ci-après dont les lignes sont données en exemple :

Labo	Geom	Mat	Densite (g/cm3)	R (mm)	Kn (N/m)	Gamma ($N.m^{-1}s$)	Kt (N/m)	Mu	Note	Votant	...
MSC	poly	verre	2.1	-	3.10^7	20	2.10^6	0.5	2	2	...
LPMDI	poly	acier	6	-	5.10^6	1	4.10^5	0.7	3	6	...
LPGP	sphere	metal	4.7	4	10^6	10	3.10^4	0.2	3	3	...
LMGC	sphere	verre	1.56	1	2.10^7	5	10^7	0.3	4	5	...
PMMH	quartz	verre	1.46	2	7.10^8	0.5	2.10^8	0.4	2	1	...
...	...	...	...	...	...	...	...	...	...	...	...

❑ **Q7** – Écrire la requête en langage SQL qui va permettre de trouver les valeurs de paramètres possibles (Densite, R, Kn, Gamma, Kt, et mu) pour des *sphères* en *verre* avec une *note* strictement supérieure à 3 sur un échantillon d'au moins 3 *votants*

Après une phase d'initialisation qui permet d'affecter les différentes variables caractéristiques avec les valeurs issues de la base de données, on exécute le code python suivant :

```
1 class grain :                  #Objet grain i
2     def __init__(self,x,y):
3         self.pos = (x,y)       #Position (Xi,Yi)
4         self.vit = (0,0)       #Vitesse intermediaire (Vix,Viy)
5         self.force=(0,0)       #Force d'interaction (Fix,Fiy)
6         #...
```

❑ **Q8** – Analyser et expliquer le rôle de cette partie de code.

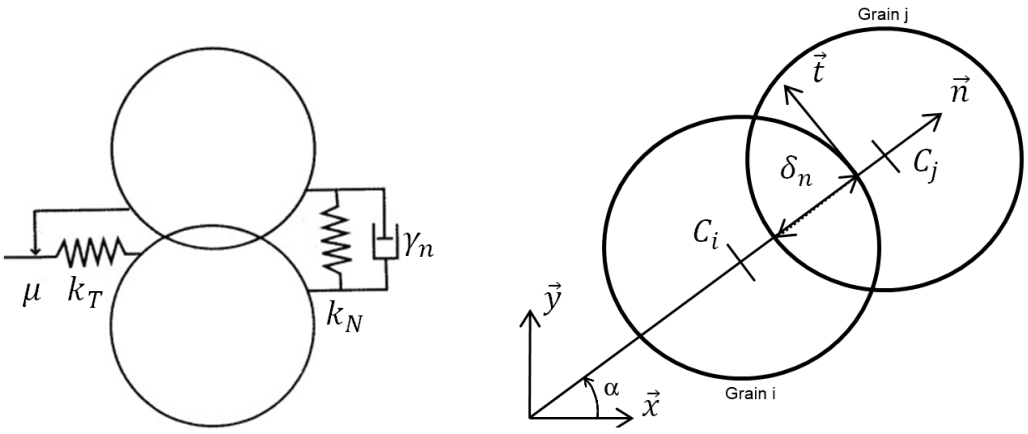
Puis on exécute :

```
1 R=1                # Rayon des grains
2 n=5                # Nombre de niveaux
3 tas=[]             # Initialisation de la liste
4 for i in range(n) :
5     for j in range(n-i):
6         tas += [grain(0+i*R+j*2*R,R+i*sqrt(3)*R)]
```

❑ **Q9** – Justifier et esquisser la forme du "tas" de grain obtenu.

❑ **Q10** – Les positions sont calculées de façon à ce que théoriquement les grains soient empilés sans aucune interpénétration, mais est-ce possible numériquement ?

Pour modéliser les interactions entre les particules, nous allons utiliser la méthode des solides déformables "sphères molles" : le calcul des forces et des moments sera réalisé en considérant que les disques sont indéformables mais peuvent s'interpénétrer légèrement avec δ_N et δ_T le déplacement normal et tangentiel à partir du contact. Ainsi, pour modéliser les forces normales, on utilise un modèle de ressort (raideur k_N) associé à une dissipation visqueuse (coefficient γ_n) permettant de reproduire une collision inélastique. Pour les forces tangentielles, un ressort (raideur k_T) couplé à un patin (limite du glissement μ) permet de modéliser la force de friction (voir figure 4).



(a) Modèle (b) Paramétrage

FIGURE 4 – Sphères molles avec interpénétration exagérée

Dans la suite du problème, nous nous intéressons à la résolution du Principe Fondamental de la Dynamique en résultante selon $\overrightarrow{x}$ et $\overrightarrow{y}$ pour chaque grain i :

$$m_i.\frac{d^2x_i}{dt^2} = F_{ix} \qquad et \qquad m_i.\frac{d^2y_i}{dt^2} = F_{iy} - m_i.g \tag{1}$$

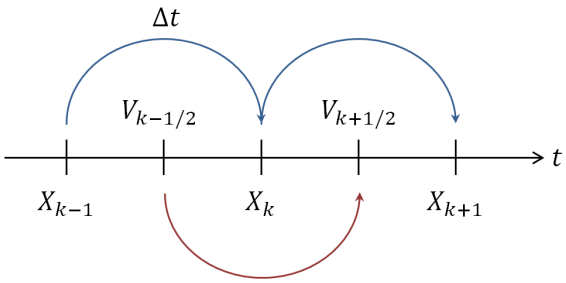
On considère qu'on dispose de la fonction `calcul_Fnt(i,j)` qui renvoie `(Fjin,Fjit)` correspondant respectivement aux composantes algébriques normales (direction $\overrightarrow{n}$) et tangentielles (direction $\overrightarrow{t}$) de la force exercée par la particule j sur la particule i. Si les deux particules ne sont pas en contact alors la fonction renvoie $(0, 0)$.

❑ **Q11** – A partir du paramétrage fourni figure 4, écrire un bloc d'instructions qui permet de déterminer la valeur de $\cos(\alpha)$ et $\sin(\alpha)$ avec $\alpha = (\overrightarrow{x}, \overrightarrow{n})$ en fonction des coordonnés `(Xi,Yi)` et `(Xj,Yj)` pour deux grains i et j. On pourra utiliser la fonction `sqrt(nb_flottant)` qui renvoie la racine carrée de `nb_flottant`.

❑ **Q12** – En déduire une fonction `somme_int()` effectuant la somme des interactions `Fix` et `Fiy` (selon $\overrightarrow{x}$ et $\overrightarrow{y}$) sur chaque grain i, exercées par **tous les autres** grains j. On utilisera la fonction `calcul_Fnt(i,j)` et la structure de donnée précédente (on accède au tuple contenant les composantes de l'effort d'interaction `(Fix,Fiy)` exercées sur le grain i avec l'expression `tas[i].force` et on procède de même pour la position du centre du grain avec `tas[i].pos`).

Le schéma d'intégration utilisé dans la suite sera celui de l'algorithme de Verlet "saute-mouton" (leapfrog) qui possède certains avantages par rapport à la méthode d'Euler notamment en termes de réversibilité du temps, de précision et de stabilité. On calcule les positions des grains aux temps $t = 0,\ \Delta t,\ 2\Delta t \ldots$ où Δt est le pas de temps. Les vitesses sont calculées et mémorisées pour des temps intermédiaires, $t = \Delta t/2,\ 3\Delta t/2 \ldots$. On utilise les vitesses intermédiaires pour déterminer les positions aux instants $k\Delta t$.

On note :



1. x_k la position du grain au temps $t = k\Delta t$
2. $v_{k+1/2}$ la vitesse au temps $t = (k+1/2)\Delta t$
3. a_k l'accélération au temps $t = k\Delta t$

❑ **Q13** – Donner une formule de calcul de $v_{k+1/2}$ en fonction de $v_{k-1/2}$, a_k *et* Δt ainsi que x_{k+1} en fonction de x_k, $v_{k+1/2}$ et Δt.

❑ **Q14** – En utilisant le schéma d'intégration ainsi défini, écrire le bloc d'instructions qui permet de calculer les positions à chaque pas de temps Δt=T_{stop}/inc avec T_{stop} le temps de simulation et inc le nombre d'incrément définis par l'utilisateur. On utilisera `somme_int()` définie précédemment. On prendra aussi en compte la pesanteur telle que définie dans l'équation de la dynamique (1) avec une même masse $m_i =$ `M` pour tous les grains.

❑ **Q15** – Expliquer pourquoi le choix du pas de temps est crucial pour ce modèle numérique.

Un problème d'oscillation apparaît dans le modèle précédent. Pour le résoudre, on propose d'introduire un amortissement supplémentaire (utilisant un paramètre $\tau > 0$) dans l'équation de la dynamique :

$$m.\frac{d^2x_i}{dt^2} = F_{ix} - \frac{1}{\tau}\frac{dx_i}{dt} \qquad et \qquad m.\frac{d^2y_i}{dt^2} = F_{iy} - \frac{1}{\tau}\frac{dy_i}{dt} - m.g \tag{2}$$

❑ **Q16** – Pourquoi cette equation (2) est-elle problématique par rapport au schéma d'intégration mis en place et comment résoudre ce problème ?

IV. Optimisation de l'algorithme et exploitation

La simulation des modèles par éléments distincts (DEM) est très gourmande en temps processeur et en allocation mémoire, il est donc nécessaire d'optimiser le code.

❑ **Q17** – Quelle est la partie de l'algorithme proposé qui est la plus pénalisante ? Evaluer sa complexité.

❑ **Q18** – Proposer (sans coder) des améliorations possibles afin de diminuer cette complexité et donner une estimation du gain ainsi obtenu.

❑ **Q19** – Expliquer (sans coder) comment générer une animation graphique 2D fluide des grains en ayant une incidence minimale sur l'espace occupé en mémoire et sur le temps de simulation (temps de résolution du modèle jusqu'à l'instant $t = T_{stop}$) ?

Fin de l'épreuve.