Ségrégration par la taille dans un milieu granulaire vibré

Christoph Charles, Jean-Baptiste Morlot

1er janvier 2011

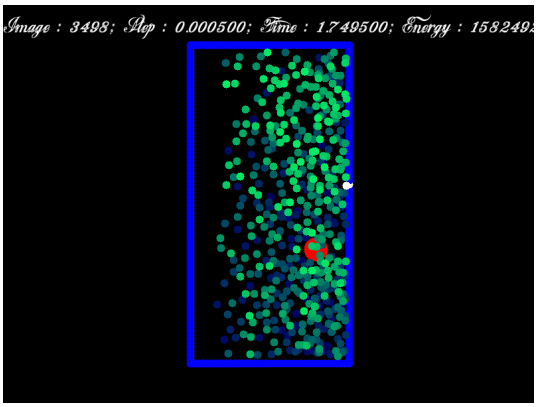
Résumé

Les matériaux granulaires possèdent des comportements très riches, dont beaucoup d'aspects sont encore mal compris. L'un d'eux est la capacité d'un mélange de grains à s'auto-organiser. Nous étudions ici ce phénomène de tri dans un milieu granulaire soumis à des vibrations répétées. L'expérience consiste à placer au fond d'un récipient une bille de diamètre important puis de le remplir par d'autres de diamètre plus petit. On soumet ensuite le récipient à des excitations horizontales, supposées sinusoïdales. Au cours du temps, on observe que la grosse bille remonte dans le récipient.

1 Modèle et implémentation

1.1 Modèle physique envisagé

Ici, nous abordons un modèle plan très simple. D'autres expériences plus complexes de ségrégation ont été étudiées, notamment la ségrégation dans un tambour en rotation (cf. [4]). En première approche, nous pouvons modéliser les particules par des sphères solides indéformables. Une telle implémentation est abordable avec certains alogorithmes comme l'algorithme de propagation de chocs (cf [2]) cependant ces algorithmes ne sont en général qu'approximatifs et des résolutions exactes amènent rapidement des problèmes de convergence. Ici, afin de faciliter les calculs pour ce modèle, nous supposons que les collisions ne sont pas parfaitement élastiques mais que la force que deux grains exercent l'un sur l'autre est proportionnelle à leur déformation. Nous modélisons alors la paroi du récipient par un ensemble de sphères de masses infinies.



1.2 Architecture du programme

Organisation des fichiers On peut séparer le programme en trois grands ensembles de fichiers.

Le groupe des outils Ce groupe contient les fichiers suivants :
- utils.c/h contient des outils mathématiques pour la simulation (MIN, MAX, résolution des équations du deuxième ordre).
- vector2D.c/h contient une implémentation simple de calcul vectoriel à deux dimensions.
- draw.c/h contient les fonctions pour l'affichage temps réel et le suivi de la simulation.

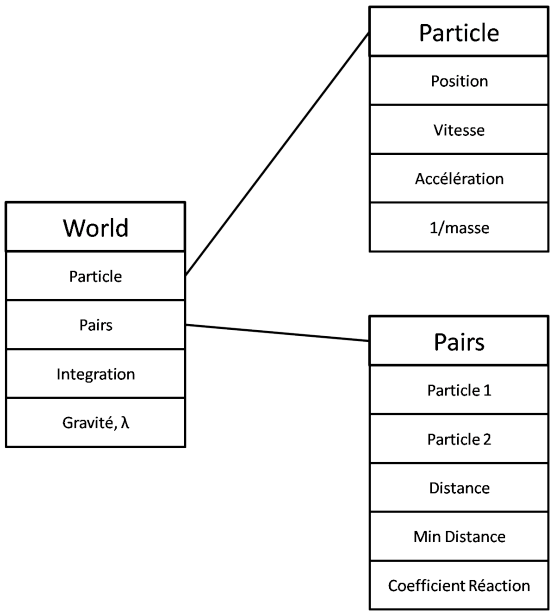
La gestion de la simulation Cela concerne l'encapsulation des données et les méthodes d'intégration :
- world.c/h contient une structure d'encapsulation de la simulation afin de proposer une interface simple.
- particle.c/h contient la structure qui décrit une particule ainsi que quelques méthodes associées.
- pair.c/h contient la structure qui définit une paire de particules. Cela contient notamment les données relatives aux interactions.
- integration.c/h et associés. Ces fichiers décrivent différentes méthodes d'intégration (méthodes de Verlet et Runge-Kutta d'ordre 2 et d'ordre 4).
- corrector.c/h contient des structures pour gérer les noyaux durs.

Le corps du programme Les données de la simulation sont entièrement définies dans deux fichiers :
- main.c qui contient la boucle de calcul et la mise en place de la scène.
- defines.h qui contient l'ensemble des constantes physiques de notre problème.

1.3 Détails du programme

Description des structures utilisées Nous utilisons des structures assez usuelles (cf [1] ou [3]). Chaque particule est décrite par une structure contenant les informations essentielles qui la concernent (inverse de sa masse, vitesse, position, ancienne vitesse, ancienne position, accumulateur de force...) et chaque paire de particules est également décrite par une structure. Cette dernière permet par exemple de fixer un coefficient pour la force de répulsion, différent pour chaque paire.

FIGURE 1 – Structures employées dans la simulation



La structure World permet d'encapsuler aisément les structures et de masquer les côtés techniques lors de la création de la scène.

Étude d'un pas de la simulation Nous étudions ici comment la simulation est effectuée. Étudions d'abord théoriquement, ce qui est nécessaire :

- Une équation différentielle : c'est-à-dire une équation qui donne l'accélération et la vitesse d'une particule choisie, en fonction des positions et vitesses courantes de chacune des particules. Ceci sera modélisé par `WorldStepInternal`.
- Une méthode d'intégration : il s'agit de regrouper toute l'intégration dans une unique fonction. Cette fonction représentera la méthode de Verlet ou une des méthodes de Runge-Kutta par exemple. Il faut donc utiliser des pointeurs sur fonction afin de pouvoir gagner en modularité.
- Une fonction d'encapsulation : elle permettra de jouer au niveau de l'intégration le même rôle que la structure `TWorld` pour la scène (masquer la technicité). Il s'agit ici de `WorldStep`.

L'équation différentielle Pour effectuer l'intégration proprement dit, il suffit de calculer l'accélération de chaque particule. Pour cela, on utilise un accumulateur de force (ou plutôt d'accélération) sur chaque particule. Ils sont tous d'abord remis à zéro.

```
1  void WorldStepInternal(struct TWorld* World, int Pass, double SmallStep)
2  {
3    int j = 0;
4    for (j = 0; j < World->ParticleCount; ++j)
5      ParticleResetAcceleration(&World->Particles[j]);
```

On calcule ensuite les forces d'interaction entre particules. Pour cela, on met d'abord à jour la distance entre les particules. On prend éventuellement en compte un décalage pour les méthodes de Runge-Kutta par exemple. Le décalage est stocké par la méthode d'intégration dans la structure de particule et le numéro du décalage est spécifié par `Pass`. Les forces sont effectivement calculées par `PairComputeForce`.

```
6    for (j = 0; j < World->PairCount; ++j)
7      {
8        PairUpdateDistance(Pass, SmallStep, &World->Pairs[j]);
9        PairComputeForce(&World->Pairs[j]);
10     }
```

Enfin, on calcule la force de pesanteur et les forces de frottement visqueux. Encore une fois, il ne faut pas oublier les éventuels décalages. On ne prend en compte que les particules de masse finie. Les autres seront traitées séparément.

```
  for (j = 0; j < World->ParticleCount; ++j)
    {
      if (World->Particles[j].InvMass > EPSILON)
  {
    TVector2D Amort;
    Amort = World->Particles[j].VelocityRope[Pass];
    Vect2DMul(SmallStep, &Amort);
    Vect2DAdd(Amort, World->Particles[j].CurrentVelocity, &Amort);

    Vect2DMul(-World->Particles[j].InvMass*World->Lambda, &Amort);
    Vect2DAdd(World->Particles[j].Acceleration, Amort, &World->Particles[j].Acceleration);
    Vect2DAdd(World->Particles[j].Acceleration, World->Gravity, &World->Particles[j].
        Acceleration);
  }
    }
}
```

L'intégration, l'exemple de la méthode de Verlet Pour la méthode de Verlet, on procède en deux temps. D'abord, on applique l'équation différentielle à l'ensemble du système. Elle ne sera utilisée qu'une fois car la méthode de Verlet est une méthode du premier ordre.

```
void IntegrationVerlet_func(double Step, struct TWorld* World)
{
  WorldStepInternal(World, 0, 0.);
```

Ensuite, on parcourt l'ensemble des particules et on met à jour la position et la vitesse de chacune.

```
  int i = 0;
  for (i = 0; i < World->ParticleCount; ++i)
    IntegrationVerlet_Part(Step, &World->Particles[i]);
}
```

Le travail est délégué à `IntegrationVerlet_part` qui ne fait qu'appliquer la formule de Verlet :

$$x(t+dt) = 2x(t) - x(t-dt) + dt^2 a(t)$$

```
void IntegrationVerlet_Part(double dt, TParticle* Part)
{
  // Calcul de : x(t+dt) = 2*x(t) - x(t-dt) + dt^2 * a(t)
  // x = NewPos
  // dt^2*a(t) = dt_2_Acc
  // x(t-dt) = OldPosition
  // x(t) = Position
  TVector2D NewPos, dt_2_Acc;
  double dt_2 = dt*dt;
  dt_2_Acc = Part->Acceleration;
  Vect2DMul(dt_2, &dt_2_Acc); // Calcul de dt^2 * a(t)

  NewPos = Part->Position;
  Vect2DMul(2., &NewPos);
  Vect2DAdd(NewPos, dt_2_Acc, &NewPos);
  Vect2DSub(NewPos, Part->OldPosition, &NewPos);

  // Mise a jour des variables
  Part->Position = NewPos;
  Vect2DSub(Part->Position, Part->OldPosition, &Part->Velocity);
  Vect2DMul(1./dt, &Part->Velocity);
}
```

L'encapsulation La fonction au cœur d'un pas de simulation est la fonction `WorldStep`.

La fonction `WorldStep` sert essentiellement d'interface à la fonction d'intégration. La quasi-totalité du travail est déléguée à la fonction d'intégration.

```
void WorldStep(struct TWorld* World, double TimeStep)
{
  int i = 0;
  int j = 0;
  // CurrentPosition et CurrentVelocity garderont la trace de la position et de la vitesse au
      debut du pas
  for (j = 0; j < World->ParticleCount; ++j)
    {
      World->Particles[j].CurrentPosition = World->Particles[j].Position;
      World->Particles[j].CurrentVelocity = World->Particles[j].Velocity;
    }
  // L'integration en elle-meme est deleguee a une fonction de traitement externe
  (*World->IntegrationMethod.Func)(TimeStep, World);
```

Les particules de masse infinie sont traitées séparement. Cette gestion est nécessaire pour les noyaux durs essentiellement.

```
  // On met a jour OldPosition et OldVelocity pour les methodes qui en ont besoin (Verlet)
  for (j = 0; j < World->ParticleCount; ++j)
    {
      // Le traitement n'est generalement pas correct pour les particules de masse infinie
      // On le refait comme il le faut
      if (World->Particles[j].InvMass < EPSILON)
  {
    Vect2DSub(World->Particles[j].CurrentPosition, World->Particles[j].OldPosition, &World->
        Particles[j].Velocity);
    Vect2DAdd(World->Particles[j].CurrentPosition, World->Particles[j].Velocity, &World->
        Particles[j].Position);
    Vect2DMul(1./TimeStep, &World->Particles[j].Velocity);
  }
      World->Particles[j].OldPosition = World->Particles[j].CurrentPosition;
      World->Particles[j].OldVelocity = World->Particles[j].CurrentVelocity;
    }
```

Enfin, il y a la gestion des noyaux qui sera détaillée plus tard.

```
  if (World->Corrector)
    {
      for (j = 0; j < World->ParticleCount; ++j)
  World->Particles[j].Color = World->Particles[j].OldColor;
      for (i = 0; i < World->CorrectorPass; ++i)
  {
    for (j = 0; j < World->ParticleCount; ++j)
      ParticleResetAcceleration(&World->Particles[j]);
    for (j = 0; j < World->PairCount; ++j)
      {
        if ((World->Pairs[j].Part1->InvMass > EPSILON) || (World->Pairs[j].Part2->InvMass >
            EPSILON))
    if ((!World->OnlyCorrectBorders) || (World->Pairs[j].Part1->InvMass < EPSILON) || (World->
        Pairs[j].Part2->InvMass < EPSILON))
      (*World->Corrector)(TimeStep, &World->Pairs[j]);
      }
  }
    }
}
```

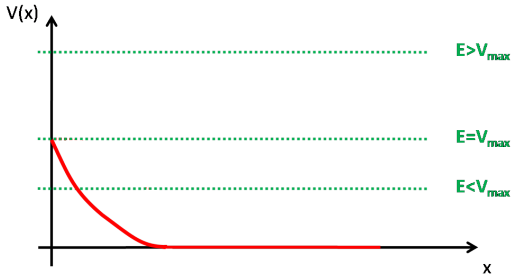
1.4 Problèmes rencontrés

On peut classer les problèmes rencontrés en deux grandes classes : ceux liés aux méthodes numériques et les problèmes intrinsèques à la modélisation physique.

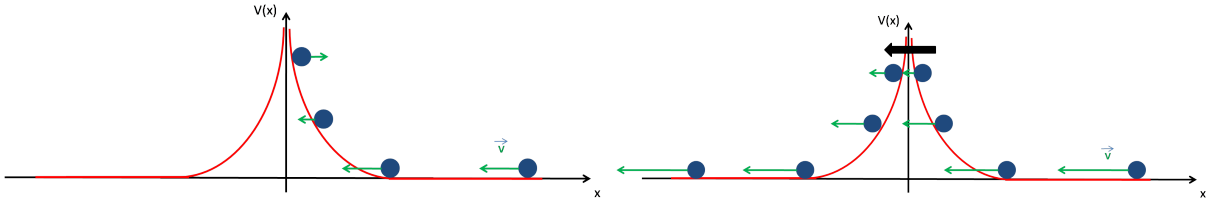
Simulation à hautes énergies La première complication provient de la forme du potentiel. En effet, à cause des forces de rappels, le potentiel prend la forme d'une parabole (à support compact néanmoins). Ainsi si l'énergie d'une boule dépasse le maximum de la parabole, il n'y aura pas de choc à proprement parler. Le problème est très important pour les parois qui peuvent alors être traversées.

Pour pallier ce problème, plusieurs solutions sont envisageables. D'abord minimiser l'énergie des boules. Cela peut passer par une force de frottement afin de diminuer les conséquences des erreurs d'intégration. On peut également changer la forme du potentiel pour qu'il tende vers l'infini lorsque la distance tend vers zéro. Cette solution sera discutée dans le prochain paragraphe. On peut également envisager la modélisation d'un noyau dur, ce qui sera étudié en fin de section.

FIGURE 2 – Potentiel parabolique initialement utilisé



Intégration numérique et stabilité Il existe deux difficultés principales issues de l'intégration numérique. D'abord la précision des méthodes qui peut provoquer une augmentation de l'énergie du système. C'est ce qui nous a encouragés à appliquer les méthodes de Runge-Kutta. Ensuite, le pas d'intégration (fini) pose problème avec toutes les méthodes si on s'approche d'un point où les dérivées ne sont pas bornées. Par exemple lorsque le potentiel tend vers l'infini, il est possible de le franchir entre deux instants consécutifs.

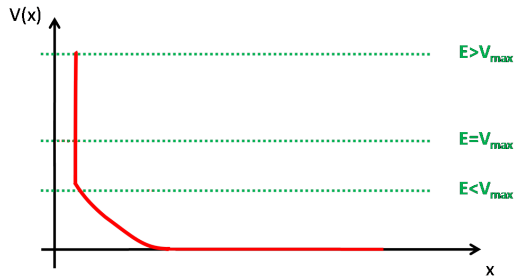


(a) Situation attendue (b) Traversée du potentiel infini

FIGURE 3 – Problème lié au pas d'intégration fini

Mise en place d'un noyau dur pour les interactions avec les parois Une solution consiste à proposer une gestion des cas où le potentiel atteint l'infini. Cela permet d'aboutir à la création d'un noyau dur (*i.e.* une zone compacte de l'espace où le potentiel atteint l'infini). Nous étudions ici la gestion des noyaux durs par une approche dite de correction.

FIGURE 4 – Forme d'un potentiel avec noyau dur



Le principe consiste à corriger *a posteriori* le parcours des billes. Pour cela, on envisage toutes les paires de billes et on calcule si entre les instants t et $t+dt$ les billes de la paire courante sont entrées en collision (*i.e.* leurs noyaux sont entrés en collision). Dans ce cas, on applique les formules de collision élastique lorsque les forces sont purement normales et on obtient la position corrigée des billes.
Pour calculer l'instant de collision entre les deux billes, en supposant leurs trajectoires rectilignes, il faut résoudre :

$$\overrightarrow{M_1M_2}^2 = D^2$$

si M_1 et M_2 désigne le centre de chacune des billes et D la somme des rayons des noyaux durs.
On aboutit alors à :

$$(\overrightarrow{v_2} - \overrightarrow{v_1})^2 t^2 + 2\overrightarrow{A_1A_2}.(\overrightarrow{v_2} - \overrightarrow{v_1})t + (\overrightarrow{A_1A_2}^2 - D^2) = 0$$

en appelant A_1 et A_2 la position des centres des billes à l'instant t et v_1 et v_2 les vitesses moyennes entre t et $t+dt$.

On peut également calculer les nouvelles vitesses après collision, en supposant la conservation de l'énergie et de la quantité de mouvement. On obtient, pour les vitesses normales :

$$w_1 = \frac{2M_2v_2 - (M_1 - M_2)v_1}{M_1 + M_2}$$

$$w_2 = \frac{2M_1v_1 - (M_2 - M_1)v_2}{M_1 + M_2}$$

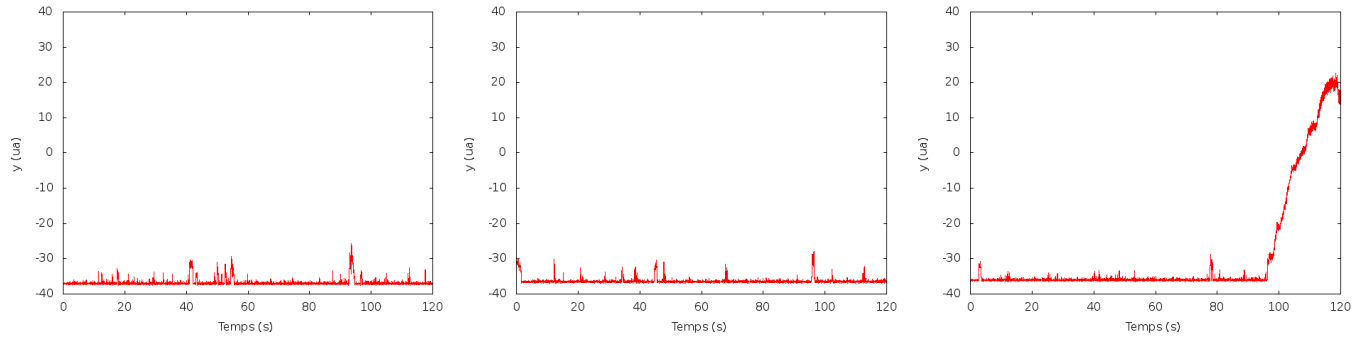
en appelant v_i les vitesses juste avant la collision et w_i les vitesses juste après.

2 Résultats et mesures

Une fois le programme établi, nous avons pu faire une série de simulations numériques. Nous avons étudié l'effet de divers paramètres sur la vitesse de ségrégation.

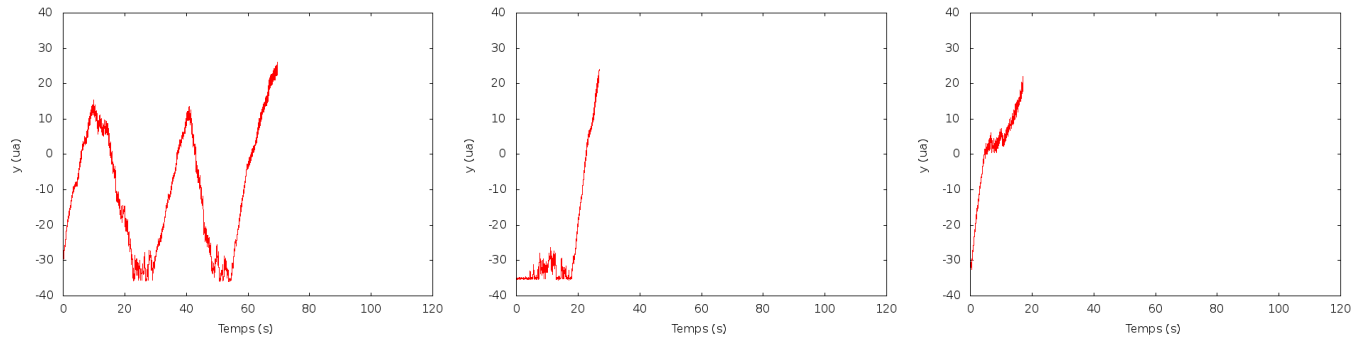
2.1 Influence du rapport des densités

Variation de la taille À masse fixée, nous avons d'abord fait varier la taille de la plus grosse boule en gardant la taille des petites constantes. On s'attend naturellement à ce qu'au delà d'une certaine densité la bille centrale reste au fond. Expérimentalement, c'est ce que l'on trouve. Dans les graphiques, on note a le rapport entre le rayon de la plus grosse bille et la plus petite, le rapport des masses étant de 9.0 dans toutes les simulations.



(a) $a = 1.5$ (b) $a = 2.0$ (c) $a = 2.5$

FIGURE 5 – Résultat des simulations avec un rayon variable jusqu'au décollage

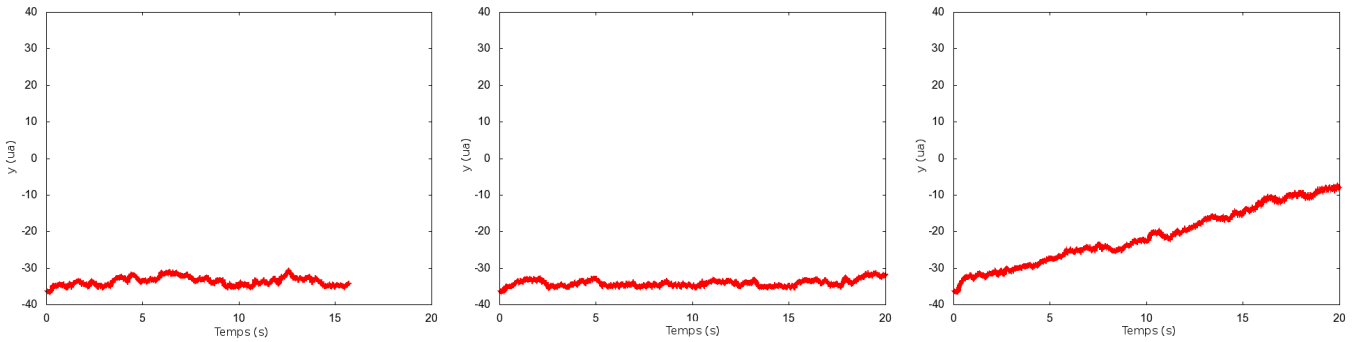


(a) $a = 3.0$ (b) $a = 3.5$ (c) $a = 4.0$

FIGURE 6 – Résultat des simulations avec un rayon variable après décollage

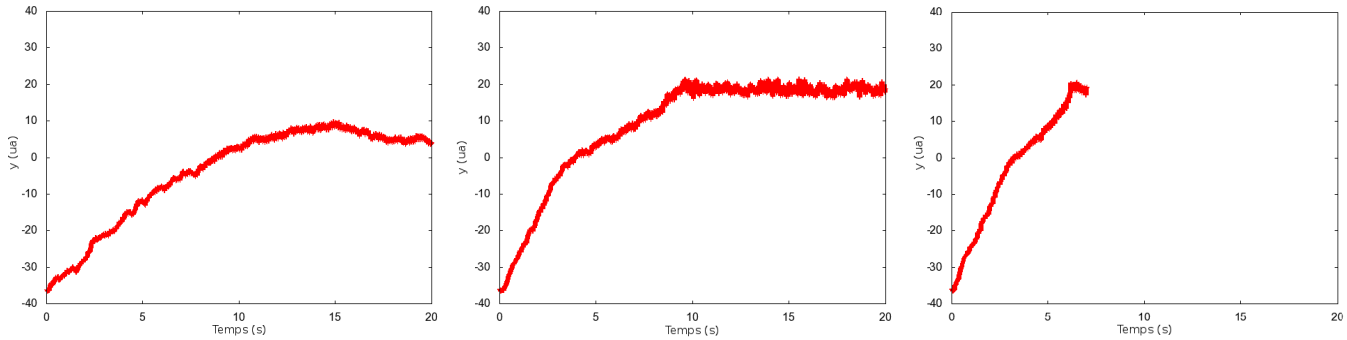
L'allure étrange pour $a = 3.0$ s'explique par le fait que la bille s'échappait du bloc central et retombait le long des parois.

Variation de la masse Afin de confirmer l'idée que la densité est maitresse, nous avons conduit une seconde série de simulation en faisant varier la masse à volume constant. Les résultats confirment notre hypothèse : il existe une densité et donc ici une masse à partir de laquelle la ségrégation n'opère plus. Dans les graphiques, a désigne le rapport $\frac{M}{m}$ où M est la masse de la bille centrale et m la masse des petites billes.



(a) $a = 13.5$ (b) $a = 12.5$ (c) $a = 11$

FIGURE 7 – Résultat des simulations avec masse variable jusqu'au décollage



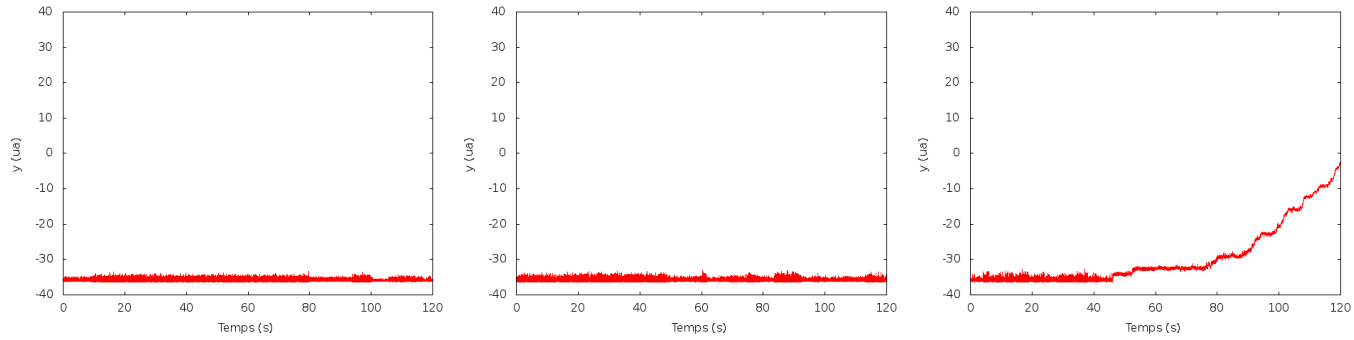
(a) $a = 1.5$ (b) $a = 2.0$ (c) $a = 2.5$

FIGURE 8 – Évolution de la ségrégation avec une masse suffisamment petite

2.2 Influence de l'énergie donnée au système

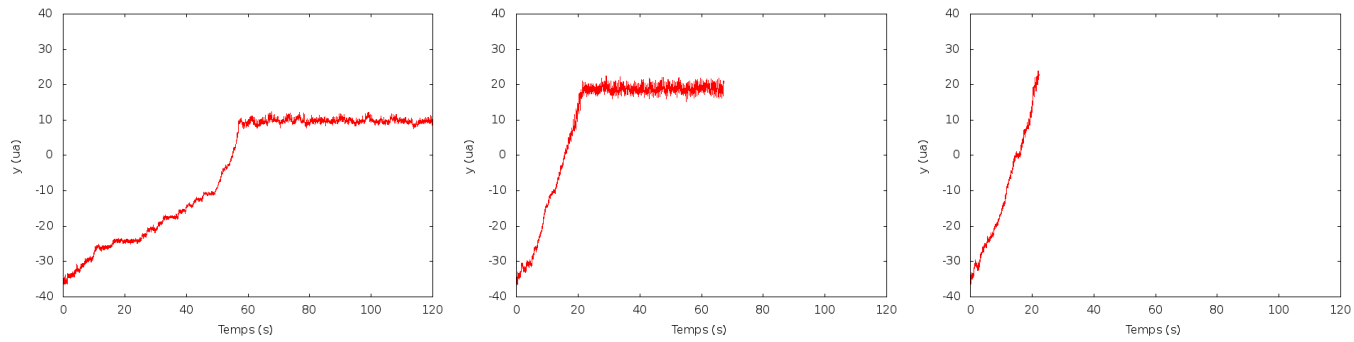
Un second paramètre est l'énergie donnée au système. Celle-ci apparaît directement dans la fréquence et l'amplitude des oscillations.

Variation de l'amplitude des simulations Nous avons d'abord étudié l'influence de l'amplitude des oscillations. Il faut noter que lors de la simulation, les billes subissaient une force visqueuse $-\lambda\overrightarrow{v}$ où $\lambda = 50.0$ $\texttt{N.cm}^{-1}\texttt{.s}$. Dans tous les diagrammes A désigne l'amplitude des oscillations (en `cm`).



(a) $A = 1.0$ `cm` (b) $A = 2.0$ `cm` (c) $A = 3.0$ `cm`

FIGURE 9 – Résultat des simulations avec une amplitude variable jusqu'au décollage



(a) $A = 4.0$ `cm` (b) $A = 8.0$ `cm` (c) $A = 9.0$ `cm`

FIGURE 10 – Évolution de la ségrégation avec l'amplitude

Influence de la fréquence d'excitation Augmenter la fréquence d'oscillation est beaucoup plus dur que d'augmenter les autres constantes. En effet, augmenter la fréquence revient à augmenter la distance caractéristique parcourue à chaque itération et donc à obtenir un nombre de collisions manquées plus important.
En terme d'ordre de grandeur, la distance caractéristique parcourue durant un pas est ωA. Disons que la distance

maximale autorisée pendant un pas est $\frac{R}{2}$ où R désigne le rayon des petites billes. On cherche :

$$\omega A \Delta t \leq \frac{R}{2}$$

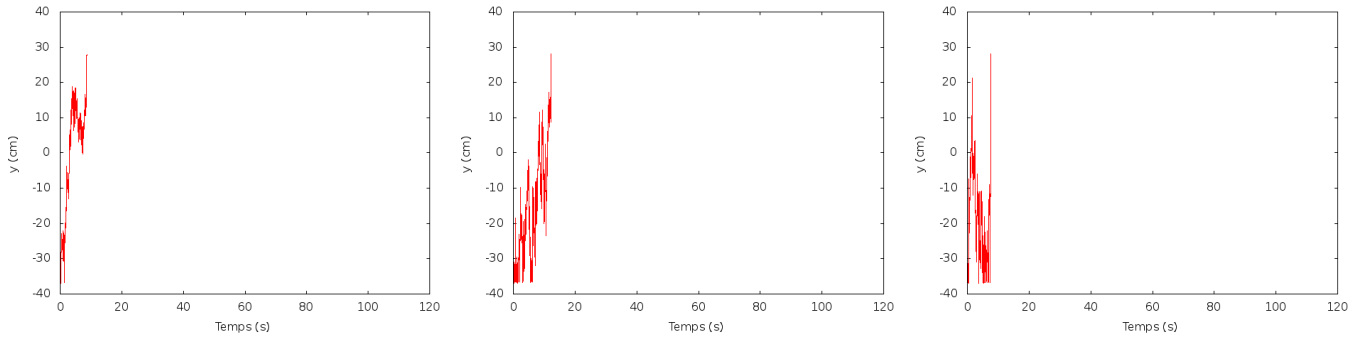
ce qui donne :

$$\omega \leq \frac{R}{2A\Delta t}$$

On a $A = 10$ `cm`, $\Delta t = 10^{-3}$ `s` et $R = 1$ `cm`. Donc :

$$\omega \leq 50 \simeq 2\pi \times 8 \text{ } \texttt{rad.s}^{-1}$$

Effectivement, on perd la stabilité de la simulation autour de 10 `Hertz`. Il fut possible de dépasser cette limite via l'introduction de noyaux durs. Néanmoins, les temps de calcul augmentent très rapidement, rendant inexploitable la simulation.



(a) $f = 8$ `Hz` (b) $f = 9$ `Hz` (c) $f = 10$ `Hz`

FIGURE 11 – Quelques simulations faisant varier la fréquence d'excitation

3 Interprétation des résultats

Malheureusement, il n'existe pas de théorie générale pour traiter la ségrégation par la taille. Il faudrait trouver une condition de ségrégation et dans ce cas, rechercher une loi pour la vitesse de ségrégation.

3.1 Étude de la vitesse de ségrégation

Nous abordons d'abord le temps caractéristique de ségrégation. L'influence du rayon est de forme hyperbolique. On s'attendrait à une dépendance directe à la densité, c'est-à-dire à une dépendance en $\frac{1}{R^2}$. Néanmoins, il faudra utiliser la densité *apparente* pour expliquer totalement les résultats.
De manière étonnante, la meilleure concordance est obtenue pour : $T = a\left(\rho^{5/2} - \rho_{milieu}^{5/2}\right)$

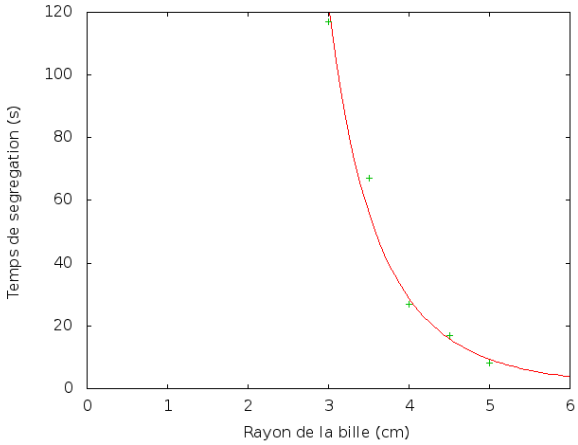


FIGURE 12 – Temps de ségrégation en fonction du rayon de la bille

On peut aussi étudier l'influence de l'énergie injectée dans le système. On s'attend à une dépendance en $1/A$, ce qui est bien obtenu. On remarque tout de même que la loi n'est plus correcte à haute amplitude. En effet, les lois étant de nature probabiliste, on s'attend à ce que les effets parasites deviennent prépondérants à court temps de ségrégation.

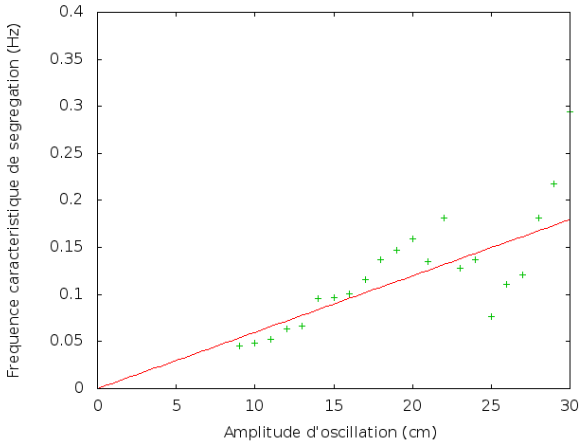


FIGURE 13 – Fréquence caractéristique de ségrégation en fonction de l'amplitude d'oscillation

3.2 Condition d'existence de la ségrégation

Comme déjà mentionné, on s'attend pour observer la ségrégation à ce que la masse volumique (surfacique dans notre modélisation) de la bille centrale doive être inférieure à la masse volumique des billes (dans l'approximation d'une faible déformation de celles-ci). Cette condition est d'ailleurs bien confirmée.

On aimerait néanmoins également comprendre pourquoi une amplitude d'oscillation minimum existe. Sachant qu'il y a conservation de l'énergie et de l'impulsion, il est équivalent de supposer qu'une bille parte de la paroi et transmette, à elle seule, l'information jusqu'à la bille centrale. Notons δ la phase de l'oscillation au moment du décollage de la bille. Elle part avec une vitesse $2A\omega\cos\delta$ et d'une position $A\sin\delta$. Elle suit l'équation :

$$m\frac{d\overrightarrow{v}}{dt} = -\lambda\overrightarrow{v}$$

Donc :

$$\overrightarrow{v} = \overrightarrow{v_0}\exp\left(-\frac{\lambda t}{m}\right)$$

et ensuite :

$$d_{atteint} = A\sin\delta + 2A\omega\cos\delta\frac{m}{\lambda}\left(1-\exp\left(-\frac{\lambda t}{m}\right)\right)$$

puis

$$d_{max} = A\left(\sin\delta + 2\cos\delta\frac{m\omega}{\lambda}\right)$$

qui atteint son maximum pour une certaine valeur de δ :

$$d = A\sqrt{1+\left(\frac{2m\omega}{\lambda}\right)^2}$$

On trouve alors l'amplitude nécessaire pour toucher la bille centrale :

$$A = 4,5 \ \texttt{cm}$$

Cette valeur est légérement plus grande ($\simeq 4$ cm) que celle trouvée expérimentalement.

3.3 Modèlisation de la position de stagnation

Lors des simulations sur l'amplitude, on trouve une position de stagnation de la bille à des hauteurs de plus en plus élevées mais sans atteindre le bord supérieur. Nous chercherons ici à modéliser ce phénomène.

On va modéliser le contenu du récipent par un gaz parfait. On supposera que les oscillations ne font que le maintenir à une certaine énergie. On suppose donc que l'équation

$$PV = nk_BT$$

est vérifiée. On posera $k_BT = E = \alpha A^2 + \beta$, β étant dû à l'énergie de pesanteur du système. On a : $n = \frac{\rho V}{m}$ donc :

$$P = \frac{\rho E}{m}$$

Puis,

$$\vec{\nabla} P = \frac{E}{m} \vec{\nabla} \rho$$

Et donc :

$$\vec{\nabla} \rho = \frac{m\rho}{E} \vec{g}$$

Ce qui se résout en :

$$\rho = \rho_0 \exp\left(-\frac{mg}{E} z\right)$$

Avec les conditions aux limites, on obtient :

$$\rho = \frac{m^2 N g}{LE} \exp\left(-\frac{mg}{E} z\right)$$

et donc

$$z = \frac{E}{mg} \ln \frac{m^2 N g}{LE\rho}$$

où L est la taille du fond du récipient et N le nombre de particules.

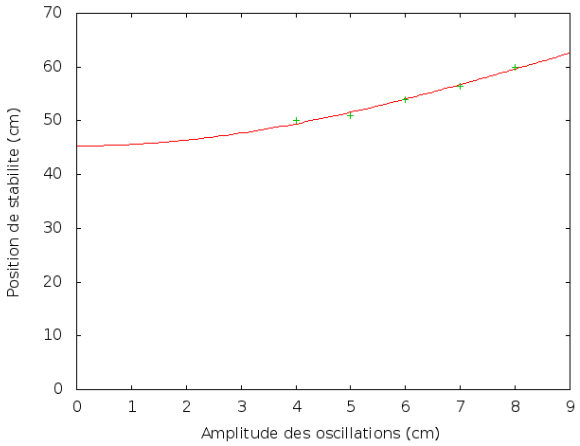


FIGURE 14 – Position de stabilité de la bille centrale en fonction de l'amplitude d'oscillation

On a une position stable pour $\rho_{bille} = \rho_{milieu}$. On obtient alors la figure ci-dessus. On obtient pour la constante de couplage : $\alpha = 1,8.10^2$ `Kg.s`$^{-2}$ qui est bien du même ordre de grandeur que $m\omega^2 = 1,6.10^2$ `Kg.s`$^{-2}$.

Conclusion

La simulation effectuée a le mérite de révéler que les équations de la mécanique sont suffisantes pour expliquer le phénomène de ségrégation. Tous les modèles proposés s'inspirent bien sûr des modèles fluides car le problème est physiquement très ressemblant. Néanmoins, on comprend que l'approximation des particules ponctuelles et l'échelle du problème peuvent avoir leur importance. On note par exemple que la ségrégation a lieu toujours avant que la théorie ne le prédise. Sans doute, les approximations ne sont plus valables dans ces régimes de fonctionnement.
Afin de valider cette approche, nous avons tenté de lancer une simulation avec le rapport $\frac{R}{r}$ très grand devant 1. Cela nécessite l'utilisation de méthodes très précises. Aussi fut-il impossible (pour des questions de temps) de lancer un grand nombre de simulations. Néanmoins, on comprend bien que, dans ce cadre, on se rapproche de l'approximation des milieux continus.

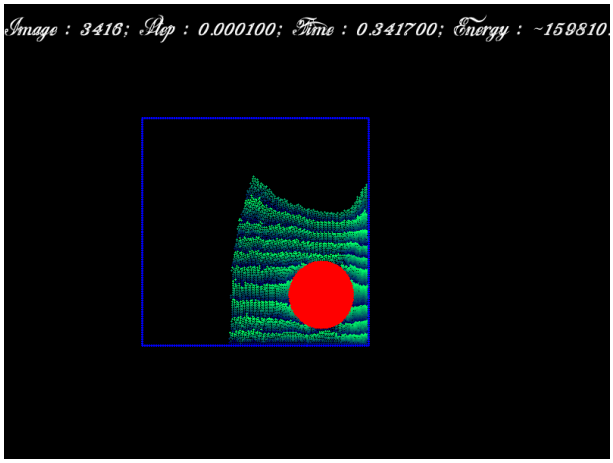


FIGURE 15 – Simulation avec un rapport $\frac{R}{r} = 30$

Références

[1] Gregory Corgié. Conception d'un moteur physique. 2008. `ftp://ftp-developpez.com/gregorycorgie/tutoriels/physic/moteur_physique.pdf`.

[2] Ronald Fedkiw Eran Guendelman, Robert Bridson. Nonconvex rigid bodies with stacking. 2003. `http://www.cs.ubc.ca/~rbridson/docs/rigid_bodies.pdf`.

[3] Julien Peyre Grégory Corgié. Implémentation d'un moteur physique. 2007. `ftp://ftp-developpez.com/gregorycorgie/tutoriels/physic/fichiers/Rapport_PhysicEngine.pdf`.

[4] Patrick Richard Nicolas Taberlet. Diffusion of a granular pulse in a rotating drum. 2006. `http://arxiv.org/abs/cond-mat/06038740`.