ESPCI PARIS | PSL

PROJET DE SIMULATION NUMÉRIQUE (2A)

Réversibilité temporelle :

Modélisation d'un Billard

Raphael Törnqvist & Arthur Quairel

5 janvier 2026

1 Introduction

L'objectif de cette étude est d'étudier la réversibilité temporelle d'un système comprenant 15 boules de billard sur lequel on envoie une 16ème boule avec une vitesse v_0. Nous faisons ensuite évoluer le système pendant une durée t_r qui vaut $t_r = \frac{t_{simul}}{2}$, sachant que t_{simul} est la durée totale de la simulation. À partir de t_r, nous inversons les vitesses des boules et nous observons l'évolution du système. Théoriquement, si nous inversons toutes les vitesses à l'instant t_r, le système devrait retracer exactement sa trajectoire passée pour revenir à son état initial. Cependant, les modèles de sphères dures sont connus pour être fortement chaotiques. Ils sont très sensibles aux conditions initiales et les erreurs d'arrondi inhérentes à la représentation des nombres flottants s'amplifient exponentiellement à chaque collision, brisant artificiellement la réversibilité du système. Une deuxième partie de notre projet a été consacrée à l'utilisation de la bibliothèque **mpmath** pour pouvoir améliorer la précision de nos simulations.

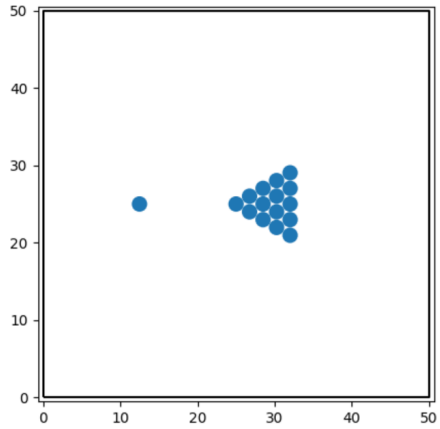

FIGURE 1 – Visualisation du billard que nous utiliserons pour les simulations

2 Architecture de la Simulation

La simulation repose sur une approche événementielle ("Event-Driven"). Contrairement à une intégration temporelle à pas fixe, notre algorithme prédit le moment exact de la prochaine interaction. Le code s'articule autour de trois modules principaux : `Simul.py` (moteur physique), `Animate.py` (rendu graphique) et `Run.py` (orchestration).

Le cycle principal de la fonction `md_step()` suit la logique suivante :

1. Détermination du temps minimal avant la prochaine collision (t_{min}), qu'elle soit contre un mur ou entre particules.
2. Avancée balistique de toutes les particules jusqu'à $t + t_{min}$.
3. Traitement de la collision (mise à jour instantanée des vitesses).
4. Répétition jusqu'à atteindre le temps total de simulation t_{simul}.

2.1 Gestion des parois (Wall Time)

Pour une particule i située en $\vec{r}_i$ avec une vitesse $\vec{v}_i$, le temps avant impact dépend de la direction du mouvement (le signe de $\vec{v}_i$) et du rayon de la particule défini par σ. La collision avec une paroi verticale (selon l'axe x) survient au temps t_{coll} défini par :

Collision Paroi 1D

$$t_{collision} = \begin{cases} \dfrac{L - \sigma - x_i}{v_{x,i}} & \text{si } v_{x,i} > 0 \\ \dfrac{\sigma - x_i}{v_{x,i}} & \text{si } v_{x,i} < 0 \end{cases} \quad (1)$$

Le `WallTime` global correspond au minimum de ces temps calculés sur toutes les particules et toutes les dimensions. Nous raisonnons de cette manière car nous effectuons nos simulations dans un billard carré, ce qui n'est pas le cas dans la réalité ; c'est une approximation de notre part.

2.2 Interactions particulaires (Pair Time)

La détection d'une collision entre deux particules i et j nécessite la résolution d'une condition géométrique de contact : $\|\vec{r}_i(t) - \vec{r}_j(t)\|^2 = (2\sigma)^2$. En injectant les trajectoires linéaires, cela se ramène à résoudre une équation quadratique en t :

$$at^2 + bt + c = 0$$

où les coefficients dépendent des positions et vitesses relatives $(\Delta\vec{r}, \Delta\vec{v})$. Une collision physique n'est validée que si le discriminant est positif et si $b < 0$ (les particules se rapprochent).

Lors du choc, la mise à jour des vitesses respecte la conservation de l'énergie cinétique et de la quantité de mouvement. Les nouvelles vitesses $\vec{v}'$ sont calculées par projection sur l'axe normal au contact $\vec{r}$:

$$\vec{v}_1' = \vec{v}_1 - [\vec{r} \cdot (\vec{v}_1 - \vec{v}_2)]\,\vec{r} \quad (2)$$

3 Configuration Initiale et Implémentation du Système

La simulation prend place dans une enceinte carrée de dimensions $L = 50$ et $W = 50$. Le système est composé de particules modélisées par des

disques durs de rayon $\sigma = 1$. L'initialisation se décompose en deux sous-systèmes : une cible cristalline et un projectile.

3.1 Placement de la Cible

La cible est constituée de 15 particules fixes arrangées selon une structure triangulaire (réseau hexagonal compact), cela est fait pour représenter le placement des boules de billard au début d'une partie.

Afin de garantir la tangence des sphères tout en respectant la géométrie du triangle équilatéral, nous avons calculé les coordonnées (x, y) de chaque particule (i, j) selon les formules suivantes :

$$x_{i,j} = \frac{L}{2} + \sqrt{3}(\sigma + 100\epsilon) \cdot i \tag{3}$$

$$y_{i,j} = \frac{W}{2} + (\sigma + 100\epsilon) \cdot (i - 2j) \tag{4}$$

Dans l'empilement compact, nous pouvons être confrontés à des chocs à N particules que notre modèle ne permet pas de traiter. Nous n'avons pas étudié la physique de ces chocs à N particules dans notre projet. Ainsi, nous avons utilisé l'astuce de "bruiter" à l'aide du paramètre ϵ les positions initiales des boules de départ pour s'affranchir de ces chocs multiples. Nous plaçons les boules à une distance $100 \times \epsilon$ les unes des autres, puis nous ajoutons un bruit aléatoire gaussien d'amplitude ϵ à la position théorique de chaque particule. La position réelle est :

$$\vec{r}_{relle} = \vec{r}_{thorique} + \epsilon \cdot \mathcal{N}(0, 1) \tag{5}$$

où ϵ est fixé à 10^{-4}. Cette étape est cruciale pour éviter les chocs à N particules lors des premières collisions.

3.2 Le Projectile

Le projectile est une particule unique ajoutée à la liste des positions après la génération de la cible. Nous l'avons placée à la position $(L/4, W/2)$, assurant ainsi un impact frontal et centré sur la structure triangulaire. Nous avons décidé d'ignorer l'angle d'impact du projectile. Nous avons ainsi pris une vitesse initiale $\vec{V}_i = (v_i, 0)$ avec V_i que nous faisons varier dans l'orchestration.

4 Analyse des Résultats et Étude de la Réversibilité

L'objectif central de notre simulation est d'éprouver la réversibilité temporelle du système de sphères dures. Théoriquement, les équations du mouvement newtonien sont symétriques par renversement du temps ($t \to -t$). Cependant, nous avons cherché à déterminer jusqu'à quel point cette réversibilité tient face à la nature chaotique du système et aux limites de la précision numérique.

4.1 Définition de la Métrique d'Erreur

Afin de quantifier l'écart entre l'état initial du système ($t = 0$) et l'état final obtenu après inversion des vitesses ($t = t_{simul}$), nous avons défini une métrique globale d'erreur sur les positions, notée $\mathcal{L}_{pos}$.

Pour un système de N particules, où $\vec{r}_k^{(i)}$ et $\vec{r}_k^{(f)}$ sont respectivement les positions initiales et finales de la particule k, la métrique est définie par la somme des normes des écarts :

$$\mathcal{L}_{pos} = \sum_{k=1}^{N} \|\vec{r}_k^{(f)} - \vec{r}_k^{(i)}\| \tag{6}$$

Si la réversibilité était parfaite, nous devrions avoir $\mathcal{L}_{pos} = 0$. En pratique, cette valeur capture l'accumulation des divergences numériques au cours de la simulation. De manière analogue, nous avons défini $\mathcal{L}_{vit}$ qui mesure l'écart entre les vitesses initiales et finales de toutes les particules.

4.2 Corrélation entre Vitesse, Temps et Nombre de Chocs

Dans notre protocole expérimental, nous avons fait varier la vitesse initiale du projectile V_i tout en gardant la durée de simulation fixe (c'est-à-dire le nombre de frames). En effet, nous possédons un contrôle précis sur les valeurs des vitesses initiales, ce qui n'est pas le cas du nombre de frames. Il est important de noter que dans un système de sphères dures, augmenter la vitesse des particules revient à augmenter la fréquence des collisions et donc le nombre de chocs que subit le système. On remarque que N_{choc} varie linéairement en fonction de V_i. Ainsi, cela valide le fait que notre étude de $\mathcal{L}_{pos}$ en fonction de V_i est analogue à l'étude de $\mathcal{L}_{pos}$ en fonction du nombre d'événements subis par le système.

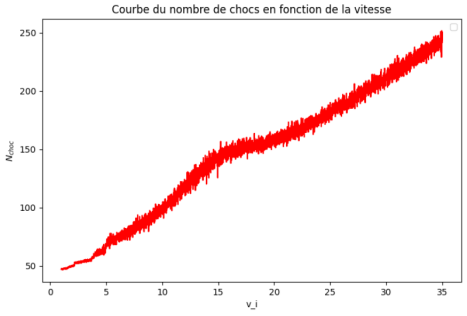

FIGURE 2 – Évolution du nombre de chocs en fonction de la vitesse initiale V_i.

4.3 Observation de la Rupture de Réversibilité

L'analyse des courbes d'erreur nous permet de distinguer trois régimes distincts dans le comportement du système :

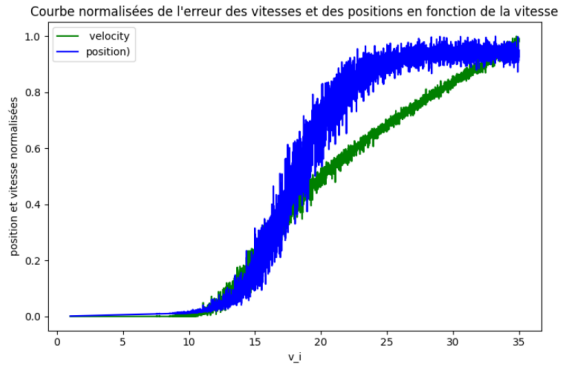

FIGURE 3 – Évolution de $\mathcal{L}_{pos}$ et $\mathcal{L}_{vit}$ en fonction de la vitesse initiale V_i pour $n_{frames} = 1000$.

1. **Régime de Réversibilité Numérique :** Pour des vitesses faibles (et donc un nombre de chocs restreint), l'erreur $\mathcal{L}_{pos}$ reste négligeable. Le système "se souvient" de son état initial, il n'est pas impacté par les approximations numériques.
2. **Régime d'Expansion Chaotique (Transition) :** À partir d'une vitesse critique (graphiquement $V_i = 12$), nous observons une croissance abrupte de l'erreur. C'est la signature du chaos hamiltonien. Chaque collision agit comme un multiplicateur d'erreur. Si l'on considère une perturbation infinitésimale δ_0, l'erreur au temps t évolue selon $\delta(t) \sim \delta_0 e^{\lambda t}$, où λ est le plus grand exposant de Lyapunov du système [1].
3. **Régime de Saturation (Perte totale d'information) :** Pour les vitesses élevées (graphiquement $V_i = 25$), l'erreur atteint un plateau. À ce stade, $\mathcal{L}_{pos}$ d'une particule est de l'ordre de grandeur des dimensions de la boîte ($L = 50$). Cela signifie que la position finale calculée n'a plus aucune corrélation avec la position initiale réelle. La "mémoire" du système a été totalement effacée par le chaos ; la réversibilité n'est plus valable.

Nous remarquons qu'il est aisé de noter ces trois régimes en regardant l'évolution de la perte positionnelle. Cependant, la perte des vitesses n'est point bornée : elle diverge. Nous avons dans notre système conservation de l'énergie cinétique ; ainsi, plus la vitesse initiale est importante, plus en moyenne les vitesses des particules seront importantes et donc plus $\mathcal{L}_{vit}$ sera importante car on répartit l'énergie cinétique en moyenne sur toutes les particules. En particulier, dans le cas de figure où l'on dépasse V_c, les écarts de vitesses vont s'accentuer avec la vitesse initiale.

Ainsi, nous avons mis en évidence l'existence d'une limite numérique de la réversibilité. Celle-ci est liée à l'utilisation de flottants codés sur 64 bits (mantisse de 52 bits et exposant de 11 bits), ce qui produit une précision d'environ 16 chiffres significatifs. Nous pouvons augmenter le nombre de chiffres significatifs à l'aide, par exemple, de la bibliothèque **mpmath**.

5 Amélioration de la précision numérique avec mpmath

La limite de la précision informatique est liée au nombre de chiffres significatifs des vitesses et des positions des particules. Par exemple, les flottants de 64 bits, qui présentent environ 16 chiffres significatifs, présentent moins d'erreur que les flottants de 32 bits qui en ont 7. La bibliothèque **mpmath** permet d'augmenter le nombre de chiffres significatifs, donc d'augmenter la précision des résultats et ainsi réduire les erreurs numériques, contre un temps d'exécution plus long.

5.1 Implémentation de la Précision Arbitraire

Dans notre code, le contrôle de la précision s'effectue via une variable **mp.dps** qui configure de manière globale le nombre de décimales que l'on utilise.

Cette instruction redéfinit la manière dont tous les objets de type `mp.mpf` sont traités lors des opérations arithmétiques.

Une contrainte technique importante est apparue lors de la phase de visualisation. La bibliothèque `matplotlib`, utilisée pour l'animation et les tracés, n'est pas compatible nativement avec les objets `mpf` de haute précision.

Pour contourner ce problème sans sacrifier la précision des calculs, nous avons dû implémenter une méthode de conversion `convert_to_float` au sein de la classe `Animate`. Ainsi, les positions sont calculées avec des types **mp.mpf** tout au long de la dynamique, et ne sont converties en flottants standards (avec perte de précision) qu'au dernier moment, uniquement pour l'affichage graphique, ce qui n'impacte pas la précision de nos calculs ni de nos simulations.

5.2 Influence de la Précision sur le Seuil de Chaos

L'apport de la précision arbitraire se mesure directement sur la stabilité de la réversibilité tempo-

relle. Théoriquement, plus la précision numérique est élevée, plus le système est capable de conserver la "mémoire" de sa trajectoire face à l'instabilité exponentielle des exposants de Lyapunov.

Avec une précision standard, la réversibilité est perdue pour des vitesses initiales inférieures à 12 (la vitesse critique étant à 12).

En passant à `mp.dps = 35`, nous avons observé que la vitesse critique est repoussée vers des valeurs plus élevées.

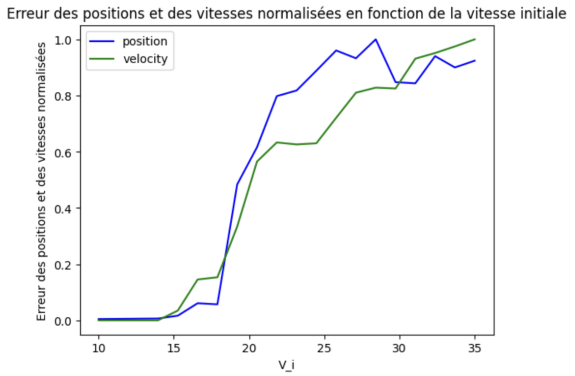

FIGURE 4 – Évolution de $\mathcal{L}_{pos}$ et $\mathcal{L}_{vit}$ en fonction de V_i pour mpf = 35 et $n_{frames} = 1000$.

Cela démontre que l'erreur observée n'est pas uniquement un bruit de fond numérique, mais bien le résultat d'une amplification progressive des infimes erreurs d'arrondi : en retardant l'apparition de ces erreurs, on retarde l'explosion chaotique, sans toutefois pouvoir l'annuler totalement.

Par la suite, nous avons étudié, pour une vitesse initiale fixe, l'influence des erreurs des positions et des vitesses moyennes et normalisées en fonction du nombre de décimales utilisé.

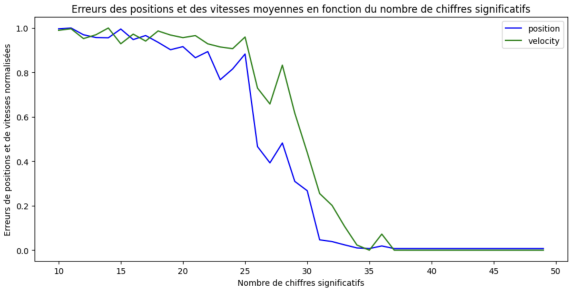

FIGURE 5 – Évolution de $\mathcal{L}_{pos}$ et $\mathcal{L}_{vit}$ en fonction du nombre de mpf.

Nous remarquons bien que plus le nombre de décimales est important, plus les erreurs des positions et des vitesses sont faibles. Cela renforce ainsi notre analyse effectuée précédemment sur l'influence du nombre de décimales sur l'erreur numérique.

5.3 Analyse du Coût Computationnel

L'utilisation de `mpmath` introduit un compromis significatif entre la fidélité physique et la performance informatique. Contrairement aux flottants standards (norme IEEE 754) qui sont gérés directement par le matériel, les opérations sur les nombres `mpf` sont émulées par logiciel.

Nous avons constaté une forte augmentation du temps d'exécution de notre simulation entre l'utilisation de flottants codés sur 64 bits (0,3 seconde d'exécution) et l'utilisation des nombres `mpf` (10 secondes d'exécution). En effet, les flottants de 64 bits sont codés sur 8 octets contre 50 à 100 pour une précision `mpf` de 50. Par ailleurs, l'utilisation de `mpmath` n'est pas directement compatible avec `numpy` ; il faut ainsi ajouter des étapes supplémentaires pour permettre la vectorisation. Ainsi, l'utilisation de flottants permet d'optimiser considérablement la vitesse des calculs numériques.

Notons que ce surcoût est inutile pour des systèmes stables ou intégrables, mais il devient indispensable dans notre cas d'étude : il permet de distinguer le chaos physique intrinsèque (sensibilité aux conditions initiales) du chaos numérique artificiel (erreurs d'arrondi).

6 Conclusion et Perspectives

Ce projet nous a permis d'explorer les limites de la réversibilité temporelle dans un système hamiltonien chaotique : le gaz de sphères dures. En développant un code de dynamique moléculaire événementiel en Python, nous avons mis en évidence le phénomène d'écho de Loschmidt et son effondrement.

L'originalité de notre approche réside dans l'utilisation de l'arithmétique à précision arbitraire via `mpmath`. Nos résultats montrent que la perte de réversibilité n'est pas un simple artefact numérique que l'on pourrait éliminer avec un ordinateur parfait. Même avec 50 décimales de précision, l'information initiale finit inéluctablement par être perdue, diluée par la dynamique expansive des collisions.

Une extension naturelle de ce travail consisterait à quantifier précisément le caractère chaotique du système, que nous n'avons ici qu'observé qualitativement via la perte de réversibilité. Pour ce faire, il conviendrait de calculer le plus grand exposant de Lyapunov, noté λ_{max}.

Références

[1] J. R. Dorfman, *An Introduction to Chaos in Nonequilibrium Statistical Mechanics*, Cambridge University Press, 1999.