Projet Compagnon 2022
Système à N corps et théorie du chaos

Marius Verdier

5 mars 2023

Résumé

Ce projet compagnon a pour but de résoudre numériquement un problème à N-corps afin de tenter de simuler des galaxies. Ce projet peut être mis en lien avec de la mécanique céleste.

Je modéliserai ce problème avec des équations différentielles ordinaires, et je procèderai à des résolutions numériques des différentes EDO que j'obtiendrai. Ci dessous les équations différentielles avec lesquelles je travaillerai.

$$\frac{d^2\mathbf{r}_i}{dt^2} = -G\sum_{j=1}^{n}\frac{m_j(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

$$F_{ij} = G\frac{m_i m_j(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

Dans ces équations, $\mathbf{r}_i$ et $\mathbf{r}_j$ représentent les positions du i-ème et du j-ème corps, respectivement, m_i et m_j représentent leurs masses, G est la constante gravitationnelle et F_{ij} est la force gravitationnelle agissant sur le i-ème corps en raison du j-ème corps. La somme dans la première équation est prise sur tous les corps du système, de sorte que l'accélération totale de chaque corps est la somme des forces gravitationnelles agissant sur lui en raison de tous les autres corps.

Une volonté de ce projet est également de tenter de faire un lien avec les systèmes chaotiques, en procédant à différentes simulations avec des conditions initiales légèrement différentes pour observer l'impact sur les états finals.

Language : Pour ce projet, je pense utiliser Python dans un premier temps.

Méthodes : Le problème ici est la quantité de calculs à effectuer. Tout d'abord, une approche naïve sera utilisée en effectuant tous les calculs. Par la suite, j'implémenterai la méthode de Barnes-Hut, qui vise à diviser l'espace en blocs contenant plus ou moins de corps.
Finalement, en fonction des résultats, je tenterai d'optimiser ces méthodes à l'aide de la librairie python "Numba".

1 Implémentations du problème

La première étape de ce projet est de discretiser les équations, afin de les résoudres numériquement. Dans les deux cas, j'utilise un schéma d'Euler, avec un pas de temps de $\Delta t = 0,1s$. Je rappelle ici la formule théorique du schéma d'Euler que j'utiliserai :

$$z_{n+1} = z_n + \Delta t f(t_n, z_n)$$

1.1 Méthode par "Brute force"

La première version de cette simulation est une version naïve, par brute force (voir notebook) : j'itère donc sur tous les corps présents, et calcule l'interaction gravitationnelle. Il est assez clair que cette métode est en $O(n^2)$, ce qui cause un temps d'éxecution rapidement trop important (en lançant uniquement la simulation sans retour graphique, j'obtiens à l'aide de la fonctionnalité %timeit *1min 45s ± 170 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)*

1.2 Méthode de Barnes-Hut

Cette méthode consiste à diviser l'espace en 2 dimensions en 4, puis diviser chaque sous-division en 4 jusqu'à n'avoir qu'un seul corps dans chaque partie de l'espace.
Ensuite, cette subdivision est stockée dans un QuadTree, où chaque noeud contient les informations principales comme le centre de masse de la partie, ainsi que sa masse totale. Cela permet donc de calculer les interractions de manière approximative, en se référant au centre de masse plutôt qu'à chaque corps individuellement. Cette méthode permet de passer d'une complexité en $O(n^2)$ à une complexité en $O(n \times \log(n))$. Pour les mêmes conditions initiales que la simulation précédente, le module timeit renvoie *2.52 s ± 106 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)*. L'implémentation de Barnes-Hut permet donc une réduction considérable du temps de calcul.

2 Résultat des simulations

Plusieurs problèmes sont apparus lors des différentes simulations menées. Tout d'abord, lors de des premières simulations avec la méthode "Brute force", le pas de temps n'était visiblement pas assez petit, et cela causait lorsque deux corps se rapprochaient, une force qui s'appliquait pendant trop longtemps.

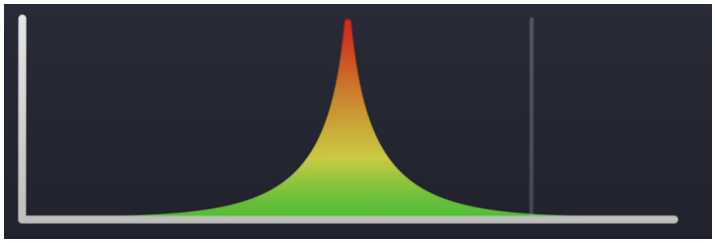

FIGURE 1 – Intensité de la force - temps continu

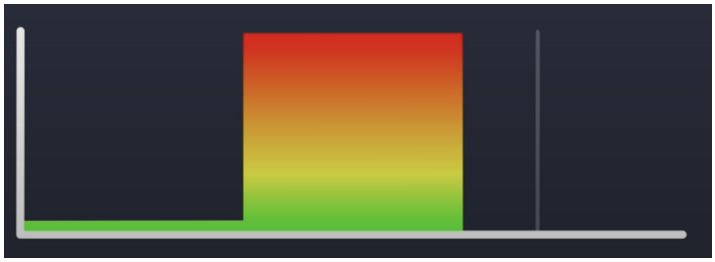

FIGURE 2 – Intensité de la force - temps discret

Cela a causé un phénomène d'éjection de certains corps, que j'ai tenté de corriger en ajoutant une correction dans la force subie $F_{ij} = G\frac{m_i m_j(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3 + \mathbf{k}}$. Après certains essais, $k = 1$ convient.

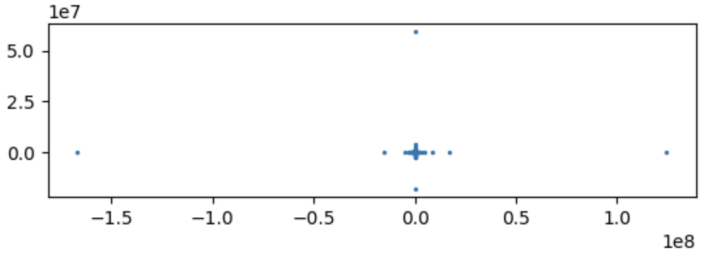

FIGURE 3 – Corps éjectés - avant correction

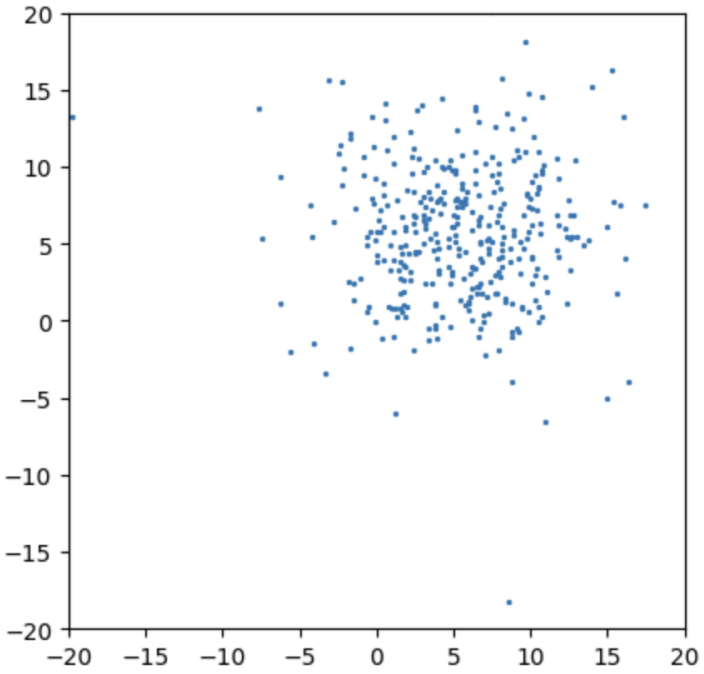

FIGURE 4 – Corps non éjectés - après correction

Ensuite, un autre problème est survenu durant l'implémentation de l'algorithme de Barnes-Hut. Lors de la subdivision de l'espace de manière récursive, j'atteignais la profondeur maximum de récursion imposée par Python très rapidement. J'ai donc du définir un quandrant minimum.
Finalement, les simulations corrigées permettent une résolution approximative de ce problème (comme le montre la vidéo jointe). Cependant, malgré de nettes améliorations amenées par l'implémentation de l'algorithme de Barnes-Hut, la résolution reste assez lente. Pour palier à ce problème, il pourrait être intéressant d'implémenter le problème dans un langage de plus bas niveau comme le C, et de faire exécuter les calculs par la carte graphique.

En réalisant plusieurs simulations avec un nombre plus restreint de corps, j'ai également pris conscience du caractère chaotique.

3 Chaos

Pour illustrer cette propriété, j'ai décidé de travailler avec 3 corps. Pour l'expliciter, il m'a fallu réaliser différentes simulations avec des conditions initiales très proches. Ici, j'ai choisi de jouer sur la position du corps 1 sur l'axe x, avec des valeurs qui diffèrent à 10^{-6} unité près.

Ces simulations sont très satisfaisantes, et montrent clairement le caractère chaotique du problème que j'ai cherché à étudier.

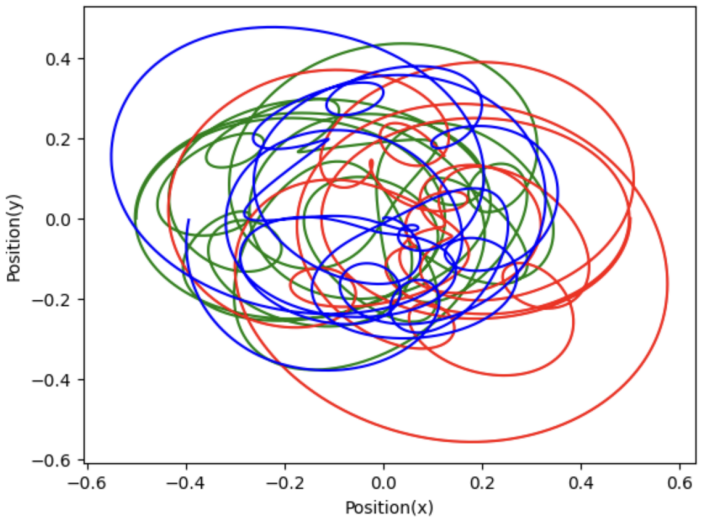

FIGURE 5 – Trajectoires - $x_1 = -0.5000001$

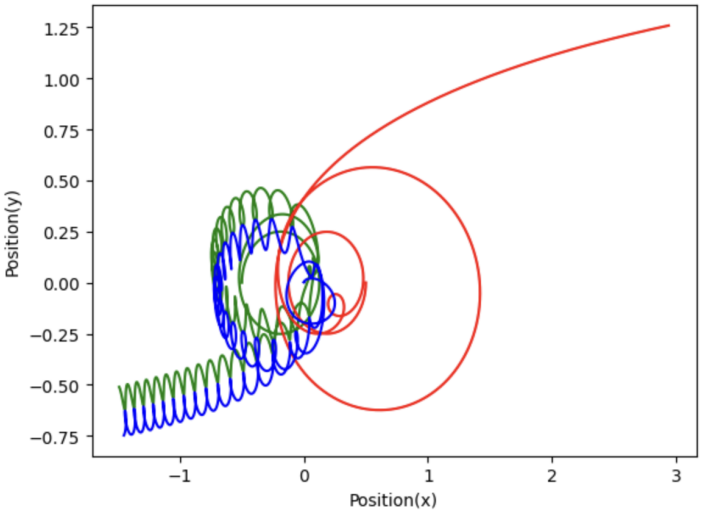

FIGURE 6 – Trajectoires - $x_1 = -0.500001$

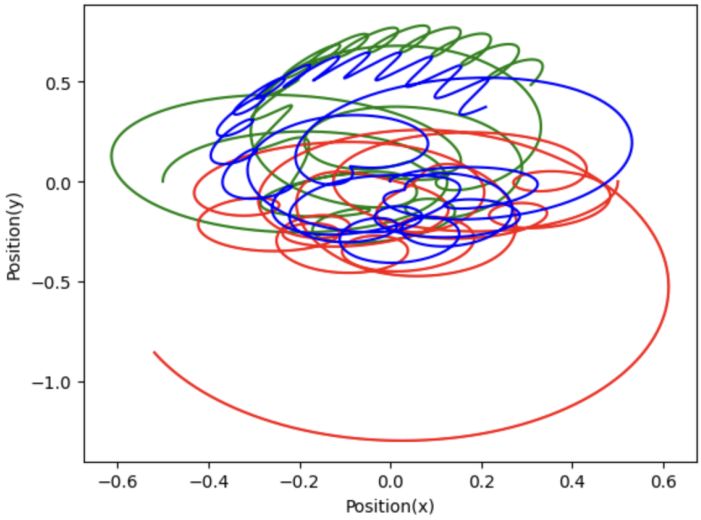

FIGURE 7 – Trajectoires - $x_1 = -0.500004$