Problème à N corps

Simuler un (petit) univers

version 0

Ismail Bennani
<ismail.lahkim.bennani@ens.fr>

16 mars 2020

Mesures exceptionnelles
Étant donnée la situation actuelle, j'ai créé un slack sur lequel vous pourrez poser vos questions et me faire part de vos problèmes. Je tacherai aussi de répondre à mes mails le plus vite possible.
Je serai aussi disponible via Skype ou Zoom pour faire des points réguliers avec vous.

1 Projet

1.1 Description du problème

Ce problème consiste à résoudre les équations du mouvement de N corps quelconques interagissant dans un champs gravitationnel en dimension $n \in \{2, 3\}$. Soient $(c_i)_{i\in[0,N]}$ nos corps. Pour chaque corps c_i, on note $m_i \in \mathbb{R}$ sa masse, $\overrightarrow{p_i} \in \mathbb{R}^n$ sa position, $\overrightarrow{v_i} \in \mathbb{R}^n$ sa vitesse et $\overrightarrow{a_i} \in \mathbb{R}^n$ son accélération.
Pour chaque couple de corps (distincts) c_i, c_j, on note $r_{i,j} = \|p_j - p_i\|$ la distance euclidienne entre ces corps et $\overrightarrow{u_{i,j}} = \frac{1}{r_{i,j}}(\overrightarrow{p_j} - \overrightarrow{p_i})$ le vecteur unitaire colinéaire à $\overrightarrow{p_i p_j}$.

La force gravitationnelle appliquée par un corps $j \in [0, N]$ sur un corps $i \in [0, N], i \neq j$ s'écrit :

$$\overrightarrow{F_{i,j}} = G\frac{m_i m_j}{r_{i,j}}\overrightarrow{u_{i,j}} \tag{1}$$

La dynamique du système est alors décrite par les équations suivantes :

$$\forall i \in [0, N], \begin{cases} \dot{\overrightarrow{p_i}} = \overrightarrow{v_i} \\ \dot{\overrightarrow{v_i}} = \overrightarrow{a_i} \\ \dot{\overrightarrow{a_i}} = \dfrac{1}{m_i} \displaystyle\sum_{j\in[0,N],j\neq i} \overrightarrow{F_{i,j}} \end{cases} \tag{2}$$

1.2 Travail attendu et évaluation

Vous écrirez un simulateur capable de calculer l'évolution des positions (en 2D **OU** 3D) des N corps dans le temps, ce simulateur devra utiliser un solveur d'équations différentielles que vous aurez écrit. Vous écrirez aussi un affichage graphique pour visualiser cette évolution.
Le projet est volontairement peu cadré pour que vous puissiez approfondir les parties qui vous intéressent le plus. Vous pourrez par exemple

- implémenter des optimisations pour accélérer le calcul du système d'équations différentielles (cf. section 2.1)
- implémenter un solveur d'équations différentielles plus précis que la méthode d'Euler (cf. section 2.2)
- implémenter des fonctionnalités pour votre interface graphique (cf. section 2.3)

Le rendu attendu est une archive contenant

- l'intégralité du code source
- pour C++ : des instructions de compilation (en particulier les librairies nécessaires et la commande de compilation)
- pour Python : un fichier requirements.txt (cf fichier requirement.txt)
- un rapport concis expliquant votre travail, en particulier :
  - la répartition des taches dans votre binôme
  - les librairies externes utilisées
  - le fonctionnement de votre programme
  - ce que vous avez approfondi
  - les difficultés rencontrées et vos solutions, les choix techniques effectués

Je vous suggère d'utiliser Python ou C++ comme langage de programmation, mais vous pouvez me consulter si vous désirez utiliser autre chose.
En section 2.4, je vous suggère quelques librairies utiles que vous pouvez utiliser avec Python et C++, vous êtes libres d'utiliser les librairies que vous voulez. Par contre, veillez à bien noter dans votre rapport tout ce que vous avez implémenté vous-même et tout ce qui vient d'une librairie externe.

Attention, la valeur d'un code ne se mesure pas au nombre de lignes ! Un bon code doit être commenté (pas besoin de trop en faire) et clair. Veillez par exemple à choisir des noms de variable explicites et informatifs.

2 Implémentation

Dans cette section, je vous décrit une façon d'implémenter le simulateur en trois parties bien délimitées et indépendantes les unes des autres. Vous **n'êtes pas** tenus de respecter cette implémentation.

2.1 Moteur Physique

Le rôle du moteur physique est de calculer une approximation de la dynamique du problème. Il s'occupe d'une part de fournir au solveur d'équations différentielles les valeurs à intégrer, et d'autre part de fournir à l'interface graphique les données nécessaires à l'affichage des corps (positions, vitesses, ...).

Une approximation simple dans notre cas est celle où les réelles sont représentés par des flottants machines, et où l'accélération d'un corps est obtenue en calculant les $N-1$ forces qu'il subit.
Cependant, on peut être plus efficace en remarquant que

$$\overrightarrow{F_{i,j}} = -\overrightarrow{F_{j,i}} \tag{3}$$

Grâce à ça, on peut obtenir le même résultat que précédemment en économisant la moitié des calculs.

Notez que l'efficacité de ce module sera cruciale si vous voulez pouvoir simuler un grand nombre de corps. En effet, la dynamique du système doit être calculée au moins une fois à chaque pas de calcul du solveur d'équations différentielles.
Une façon d'accélérer les calculs est de sacrifier en partie la précision du résultats. Plusieurs méthodes existent pour faire ça dans notre cas :

— Simulation de Barnes-Hut

Note Lors de l'implémentation de ce module, il sera judicieux d'écrire une classe de vecteurs pour représenter les positions, vitesses, accélérations et forces.
Vous pourrez aussi surcharger les opérateurs arithmétiques ($+$, $*$, ...) pour manipuler les instances de cette classe.
Surcharge opérateurs Python : explication, liste des opérateurs
Surcharge opérateurs C++ : explication, liste des opérateurs

2.2 Solveur d'équations différentielles

2.3 Interface graphique

2.4 Outils