Projets de simulation numérique en C++
Licence 3 de Physique

V. Doyeux

17 juillet 2012

Problème à deux corps

1 Description

Dans ce projet on s'intéresse à la résolution numérique du très célèbre problème de deux corps interagissant par forces gravitationelles. Ce problème décrit par exemple très bien le mouvement d'un satellite autour de son astre, ou l'éjection d'un astéroïde passant à proximité d'une grosse planète. D'un point de vue numérique, il s'agit d'intégrer les équations de Newton, soit résoudre des équations différentielles ordinaire (**EDO**).

2 Problème

On considère le système de deux masses m_1 et m_2 dans un référentiel galiléen. En appliquant le théorème du moment cinétique au centre de masse des deux particules, on peut montrer que leur trajectoires seront contenues dans un plan (cf cours de mécanique du point) . Notre problème se résume donc à l'étude 2d des trajectoires. On se munit d'un repère cartésien (xOy). Les quantités qui nous intéressent sont les positions des corps $\boldsymbol{X}_1 = (x_1, y_1)$, $\boldsymbol{X}_2 = (x_2, y_2)$ ainsi que leur vitesses $\boldsymbol{V}_1 = (v_{x1}, v_{y1})$ et $\boldsymbol{V}_2 = (v_{x2}, v_{y2})$. La force de gravitation exercée par la masse 1 sur la masse 2 s'écrit :

$$\boldsymbol{F}_{1/2} = -\frac{Gm_1m_2}{r^2}\boldsymbol{u}_r$$

avec G la constante de gravitation universelle, r la distance entre les centres de masse des corps, et $\boldsymbol{u}_r$ le vecteur unitaire allant de 1 vers 2. Par symétrie, ou en appliquant le principe d'action réaction, on a $\boldsymbol{F}_{2/1} = -\boldsymbol{F}_{1/2}$. En appliquant le principe fondamental de la dynamique :

$$m\frac{d\boldsymbol{V}}{dt} = \boldsymbol{F}$$

on obtient une équation différentielle d'ordre 1 sur le vecteur vitesse, soit 2 équations par particule. Puis par définition de la vitesse :

$$\frac{d\boldsymbol{X}}{dt} = \boldsymbol{V}$$

on obtient encore 2 équations différentielles d'ordre 1 par particule. Au final, le problème numérique consiste à résoudre ces 8 équations différentielles couplées et obtenir à chaque instant les 8 quantités : $x_1, y_1, v_{x1}, v_{y1}, x_2, y_2, v_{x2}, v_{y2}$.

3 Méthode numérique

De nombreuses méthodes existent pour résoudre une équation différentielle ordinaire du type

$$\frac{dy}{dt} = f(y,t)$$

(y pouvant être un vecteur ou un scalaire).

Dans un premier temps, on utilisera la méthode dite d'**Euler explicite** pour la résolution de notre système d'équations. De manière générale, les méthodes d'Euler consistent à discrétiser la variable y dans le temps et approximer la dérivée d'une quantité par son développent de Taylor à l'ordre 1, soit :

$$\frac{dy}{dt} = \frac{y^{n+1} - y^n}{\Delta t}$$

où le temps à été discrétisé : $t = n\Delta t$ avec Δt le *pas de temps* et n le numéro de l'itération courante. De même, $y(t)$ est discrétisée dans le temps $y^n = y(n\Delta t)$ et $y^{n+1} = y((n+1)\Delta t)$.

Enfin, une méthode est dite explicite si toute les quantités à l'itération $n+1$ peuvent s'écrire comme une fonction de quantités dépendant uniquement des instants précédents. Dans le cas de l'équation, $\frac{dy}{dt} = f(y,t)$, cela revient à dire que la dérivée de y à l'itération $n+1$ est égale à la fonction f calculée avec le y précédent et le temps précédent. Ceci mis en équation :

$$\frac{y^{n+1} - y^n}{\Delta t} = f(y^n, t^n)$$

nous donne la méthode d'Euler explicite. Si on connaît y^n, on peut calculer la valeur de y à l'itération suivante et ainsi de suite. On peut montrer (voir [1]) que l'erreur accumulée à chaque itération sur le calcul de y est de l'ordre de Δt. On dit que la méthode est d'ordre 1 (on verra plus loin le principe de la méthode **RK4** d'ordre 4, c'est à dire pour laquelle l'erreur accumulée est de l'ordre de Δt^4).

Pour information, une méthode est dite implicite si le calcul d'une quantité à une itération $n+1$ dépend des quantités de l'itération $n+1$, soit pour notre problème :

$$\frac{y^{n+1} - y^n}{\Delta t} = f(y^{n+1}, t^{n+1})$$

Les méthodes implicites sont beaucoup plus stables que les méthodes explicites. Néanmoins elles sont plus complexes à mettre en oeuvre.

4 Mise en oeuvre en C++

Tout d'abord, écrivez sur un papier explicitement les 8 équations mentionnées plus haut. Puis, réécrivez les sous la forme :

$$\frac{d\boldsymbol{U}}{dt} = \boldsymbol{F}(\boldsymbol{U})$$

avec $\boldsymbol{U} = (v_{x1}, v_{y1}, x_1, y_1, v_{x2}, v_{y2}, x_2, y_2)$. A vous de chercher $\boldsymbol{F}$.

Il vous suffira enfin d'écrire une fonction C++ qui prend en argument un vecteur `U` et renvoie un autre vecteur `F(U)` et d'appliquer le schéma d'Euler explicite décrit précédemment.

Pour mettre des variables dans un vecteur de façon simple en C++, utilisez la classe `std::vector`.

Il vous faut ensuite décrire un système, c'est à dire donner des valeurs numériques aux données initiales du problème. Pour cela vous pouvez considérer des systèmes réels, par exemple Terre / Soleil, ou Terre / Lune ou encore Jupiter / Astéroïde ... Toutes les valeurs des masses, distances et vitesses sont facilement accessible sur internet par exemple sur wikipedia ou le site de la NASA (http ://solarsystem.nasa.gov/planets). Attention, il vous faut choisir un système d'unités adapté aux dimensions de votre

système. Par exemple, le kg n'est sûrement pas l'unité de masse la mieux adaptée, 10^{20}kg l'est sûrement davantage si vous considérez une planète... N'oubliez pas que G a aussi des dimensions et doit être écrite dans le système d'unité que vous aurez choisi. De manière générale, on choisi un système d'unité dans lequel on peut écrire un maximum de quantité du problème sans avoir à les exprimer avec beaucoup de puissances de 10.

5 Traitement des données

A la fin de la simulation, vous tracerez les trajectoires des deux corps sur un même graphe. Pour cela enregistrez les variables calculées à chaque instant dans un fichier et tracez les avec un logiciel vous permettant de faire facilement des graphes comme `gnuplot`, `octave` ou `scilab` par exemple. Une manière de quantifier l'erreur numérique est de calculer la variation d'énergie totale du système. Celle-ci s'écrit facilement grâce aux données calculées. Vous tracerez la variation d'énergie totale du système dans le temps.

Vous ferez ensuite plusieurs expériences numérique pour valider votre code. Vous pourrez par exemple essayer de vérifier si les lois de Kepler sont bien satisfaites.

Le site http ://www.imcce.fr/fr/ephemerides/formulaire/form_ephepos.php vous permet de télécharger les valeurs des trajectoires calculées d'un grand nombre d'astres. Téléchargez ces données et tracez les sur le même graphe que les vôtres pour voir la différence. Expliquez les éventuelles différences.

Pour aller plus loin dans la présentation des résultats, vous pouvez enregistrer automatiquement un grand nombre d'images de vos astres à des temps différents, et assembler ces images pour en faire une vidéo grâce au logiciel `ffmpeg`.

6 Méthode d'ordre supérieur

Si l'erreur numérique est trop grande, il vous faudra sûrement utiliser une méthode d'intégration numérique plus précise. Les méthodes de Runge-Kutta sont une classe de méthode pour résoudre des problèmes du type $\frac{dy}{dt} = f(y,t)$. L'idée est que dans un même pas de temps, on évalue la fonction $f(y,t)$ plusieurs fois. Évidemment, elles sont plus coûteuses en temps de calcul. Très souvent, la méthode **RK4** (pour Runge-Kutta d'ordre 4) est choisie comme un bon compromis entre temps de calcul et précision. Cette méthode est d'ordre 4. L'algorithme de RK4 est décrit sur wikipedia. Implémentez le dans votre code et comparez à la méthode d'Euler.

Références :

[1] J-P. Demailly *Analyse numérique et équations différentielles*, Collection Grenoble Sciences (2006)