TP 3: Équations différentielles

Philippe Després et Antoine Allard

Date de remise: 14 avril 2024

PHY-3500 – Physique numérique (H24)

Mécanique céleste - la Terre

La méthode de Verlet est une variation de la méthode saute-mouton (ou *leapfrog*) qui permet d'alléger les calculs pour certaines classes de problèmes, dont ceux liés au mouvement qui ont la forme

$$\frac{d^2\mathbf{r}}{dt^2} = \mathbf{f}(\mathbf{r}, t). \tag{1}$$

Avec des conditions initiales sur $\mathbf{r} = (x, y, ...)$ et $\mathbf{v} = d\mathbf{r}/dt$, la méthode consiste à faire un premier pas pour calculer

$$\mathbf{v}(t + \tfrac{1}{2}h) = \mathbf{v}(t) + \frac{1}{2}h\mathbf{f}(\mathbf{r}(t), t), \tag{2}$$

et de poursuivre en calculant successivement

$$\begin{aligned}
\mathbf{r}(t + h) &= \mathbf{r}(t) + h\mathbf{v}(t + \tfrac{1}{2}h), && (3)\\
\mathbf{k} &= h\mathbf{f}(\mathbf{r}(t + h), t + h), && (4)\\
\mathbf{v}(t + h) &= \mathbf{v}(t + \tfrac{1}{2}h) + \tfrac{1}{2}\mathbf{k}, && (5)\\
\mathbf{v}(t + \tfrac{3}{2}h) &= \mathbf{v}(t + \tfrac{1}{2}h) + \mathbf{k} && (6)
\end{aligned}$$

a. Utilisez la méthode de Verlet pour calculer l'orbite de la Terre autour du soleil, considérant que le mouvement est décrit par

$$\frac{d^2\mathbf{r}}{dt^2} = -GM\frac{\mathbf{r}}{r^3} \tag{7}$$

et sachant que la distance du périhélie est de 1.4710×10^{11} m et que la vitesse tangentielle à cette position est de 3.0287×10^4 m/s (utilisez le module `astropy` pour les autres constantes nécessaires). Utilisez un incrément temporel $h = 1$ heure. Représentez graphiquement plusieurs orbites, qui devraient appaître légèrement non-circulaires.

b. Modifier votre programme pour qu'il calcule aussi l'énergie potentielle $-GMm/r$ et cinétique $\frac{1}{2}mv^2$ à chaque pas, ainsi que la somme (énergie totale) de ces deux quantités. Rapportez ces valeurs en fonction du temps dans un graphique et commentez.

c. Utilisez maintenant la méthode d'Euler, RK2 et RK4 pour calculer l'orbite de la Terre et l'énergie totale en fonction du temps, que vous rapportez sur le même graphique que l'énergie totale en fonction du temps calculée par la méthode de Verlet. Votre graphique devrait montrer le caractère symplectique de la méthode de Verlet vs Euler/RK2/RK4.

d. Utilisez maintenant la méthode de Bulirsch-Stoer pour calculer l'orbite de la Terre, avec une précision de 1 km par année (voir l'exemple 8.7 dans le Newman pour une implémentation, que vous pouvez réutiliser). Utilisez un intervalle H de une semaine. Comparez avec la précision obtenue avec les autres méthodes.

Mécanique céleste - Mars

La NASA vous engage et vous demande de calculer la position de la planète Mars le 18 février 2021, jour d'atterissage de l'astromobile *Perseverance*, à partir des données connues le 30 juillet 2020, jour du lancement depuis la Terre.

Vous aurez besoin du module Python `jplephem`, soit les éphémérides utilisées par le JPL (`pip install jplephem`). Vous utiliserez les éphémérides `de421` (`pip install de421`), qui couvrent avec une bonne précision la période 1900–2050.

Le code suivant vous donnera la position et la vitesse initiales de Mars (attention aux unités utilisées, voir documentation à `https://pypi.org/project/jplephem/`) :

```
from astropy.time import Time

# éphémérides
import de421
from jplephem import Ephemeris

eph = Ephemeris(de421)

# dates
lancement=Time("2020-07-30")
atterissage=Time("2021-02-18")

# un nombre de jours juliens est attendu par la routine, d'où le .jd
# position en km, vitesse en km par jour
position, velocity = eph.position_and_velocity('mars',lancement.jd)
```

e. Utilisez Bulirsch-Stoer pour calculer la position de Mars 203 jours plus tard, soit le 18 février 2021 (jour de l'atterrissage). Ne lésinez pas sur la précision, ça coûte cher un astromobile. Notez que les calculs seront effectués en trois dimensions. Comparez votre position calculée à la valeur de l'éphéméride, soit

`eph.position('mars',atterissage.jd)`. Commentez, notamment sur vos limites et celles de la méthode utilisée, peut-être en vous inspirant d'informations à cette adresse [1].

f. Quelle méthode numérique la NASA utilise-t-elle pour envoyer des sondes vers Mars ?

g. Cette méthode est-elle en cause dans la perte du *Mars Climate Orbiter* en 1999 ? Si non, quelle a été la cause de cette perte ?

1. `https://en.wikipedia.org/wiki/Jet_Propulsion_Laboratory_Development_Ephemeris`

L'équation de Schrödinger et la méthode spectrale

Soit l'équation de Schrödinger dépendante du temps

$$-\frac{\hbar^2}{2M}\frac{\partial^2\psi}{\partial x^2} = i\hbar\frac{\partial\psi}{\partial t} \tag{8}$$

que vous devrez résoudre pour trouver la fonction d'onde d'une particule de masse M dans une boîte unidimensionnelle de longueur L avec des parois impénétrables (problème sans potentiel).

La fonction d'onde dans cette boîte est nécessairement nulle aux parois et une solution possible a la forme

$$\psi_k(x,t) = \sin\left(\frac{\pi k x}{L}\right) e^{iEt/\hbar} \tag{9}$$

où l'énergie E peut être trouvée en substituant dans l'équation de Schrödinger pour donner

$$E = \frac{\pi^2\hbar^2 k^2}{2ML^2}. \tag{10}$$

Nous avons vu en classe que la transformée en sinus n'était pas très utilisée car elle requiert une fonction nulle aux extrémités considérées ; or c'est justement le cas ici.

Nous pouvons supposer que la solution au problème est une combinaison linéaire de solutions de la forme de l'équation 9, qui pour une grille de points $x_n = nL/N$ est

$$\psi(x_n,t) = \frac{1}{N}\sum_{k=1}^{N-1} b_k \sin\left(\frac{\pi k n}{N}\right) \exp\left(i\frac{\pi^2\hbar k^2}{2ML^2}t\right), \tag{11}$$

où les b_k sont des coefficients (potentiellement complexes) qui déterminent la forme de la fonction d'onde. Le facteur $1/N$ est optionnel mais utile.

Puisque l'équation de Schrödinger est du premier ordre selon le temps (contrairement à l'équation d'onde), nous avons besoin d'une seule condition initiale pour $\psi(x,t)$ pour déterminer les coefficients b_k.

Nous allons considérer un électron ($M = 9.109\times10^{-31}$ kg) dans une boîte unidimensionnelle de longueur $L = 10^{-8}$ m. Au temps $t = 0$, nous allons supposer que la fonction d'onde de l'électron a la forme

$$\psi(x,0) = \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right] e^{i\kappa x} \tag{12}$$

où

$$x_0 = \frac{L}{2}, \quad \sigma = 1\times10^{-10}\text{m}, \quad \kappa = 5\times10^{10}\text{m}^{-1} \tag{13}$$

et $\psi = 0$ aux parois à $x = 0$ et $x = L$. Cette expression pour $\psi(x,0)$ n'est pas normalisée mais ceci importe peu pour l'instant parce que l'équation de Schrödinger est linéaire.

Vous pourrez toujours renormaliser vos solutions, pour que l'aire sous la courbe de la densité de probabilité soit égale à 1, tel que requis pour respecter la physique du problème.

h. Écrivez un programme pour calculer les valeurs des coefficients b_k, qui peuvent de façon pratique être séparés en composantes réelles et imaginaires ($b_k = \alpha_k + i\eta_k$). Divisez la boîte en $N = 1000$ tranches et créez deux conteneurs pour les parties rélles et imaginaires de $\psi(x_n, 0)$ à chaque point de la grille. Effectuez une transformée en sinus pour chaque conteneur séparément, *i.e.* calculez les valeurs de α_k et β_k pour $k = 1...N-1$. Vous utiliserez les fonctions du module `dcst` de Newman (Appendix E ou son site web).

i. Expliquez pourquoi Newman s'en remet à la fonction `rfft` de `numpy` pour calculer la transformée en sinus.

j. En injectant $b_k = \alpha_k + i\eta_k$ dans la solution et en prenant la partie réelle, nous obtenons

$$\mathrm{Re}\psi(x_n, t) = \frac{1}{N}\sum_{k=1}^{N-1}\left[\alpha_k \cos\left(\frac{\pi^2\hbar k^2}{2ML^2}t\right) - \eta_k \sin\left(\frac{\pi^2\hbar k^2}{2ML^2}t\right)\right]\sin\left(\frac{\pi kn}{N}\right) \qquad (14)$$

pour la partie réelle de la fonction d'onde.

Une inspection vous convaincra qu'il s'agit de l'inverse d'une transformée en sinus pour la quantité entre crochets. Complétez votre programme pour qu'il calcule la partie réelle de $\psi(x_n, t)$ pour un t arbitraire à l'aide de la fonction `idst` fournie par Newman. Testez votre programme en illustrant la fonction d'onde à $t = 10^{-16}$ s.

k. Maintenant que tout est en place, vous pouvez produire une animation de l'évolution de la fonction d'onde (normalisée) au cours du temps, en générant une image à plusieurs t (utilisez un incrément de 10^{-18} s). Il y a probablement plusieurs façons d'y arriver, l'une d'elle étant le package `visual`. Prenez soin d'ajuster le taux de rafraîchissement et les échelles pour bien apprécier la dynamique de la fonction d'onde.

l. Laissez tourner votre animation un moment et décrivez ce que vous observez, en utilisant bien entendu un langage se rapportant à la physique du problème.

Instructions pour la remise

Le travail devra être complété en trinômes sous format de cahier de bord `jupyter` (.ipynb) et remis dans la boîte de dépôt créée à cette fin. Ce document contiendra **toutes informations pertinentes** permettant au lecteur d'apprécier vos résultats et conclusions, incluant le code Python utilisé et d'éventuelles références bibliographiques. La qualité de la présentation est très importante (utilisation de sections, de graphiques appropriés, de mise en contexte, etc.).