École Polytechnique Fédérale de Lausanne
Semestre d'automne 2018

Physique Numérique I – Exercice 4

à rendre jusqu'au 21 Décembre 2018 à 23h55 sur le site
http://moodle.epfl.ch/mod/assign/view.php?id=835970

4 Système gravitationel à 1,2 et 3 corps. Apollo 13, rentrée dans l'atmosphère et asteroïdes de Troie

On s'intéresse au mouvement de N ($N \leq 3$) corps soumis aux forces de gravitation. La force exercée sur le corps i par le corps j s'écrit :

$$\vec{F}_{ij} = -\frac{G\ m_i\ m_j}{|\vec{r}_j - \vec{r}_i|^3}\ (\vec{r}_i - \vec{r}_j) \qquad (1)$$

On utilise $G = 6.674 \times 10^{-11}[\mathrm{m^3 kg^{-1} s^{-2}}]$ comme valeur de la constante de gravitation. Dans un deuxième temps, on tiendra compte de la force de traînée aérodynamique causée par la présence de l'atmosphère terrestre :

$$\vec{F}_{t,ij} = -\frac{1}{2}\rho(|\vec{r}_i - \vec{r}_j|)SC_x|\vec{v}_i - \vec{v}_j|(\vec{v}_i - \vec{v}_j) \quad (2)$$

où

$$\rho(r) = \rho_0 \exp(-(r - R_T)/\lambda) \qquad (3)$$

est la densité de l'air, ρ_0 sa valeur au niveau de la mer, R_T le rayon de la Terre, λ une épaisseur caractéristique, S la surface de section de l'objet, C_x le coefficient de traînée.

4.1 Numérique [5 pts]

Le but de cette excercice est d'étudier le problème à N corps en utilisant les équations du mouvement sous la forme

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}). \qquad (4)$$

Ici, le vecteur $\mathbf{y}$ comprend les positions *et* les vitesses de *tous les corps*. En utilisant (5), on peut programmer le schéma de Runge-Kutta sous la forme donnée par l'Eq.(2.97) des notes de cours. Vous écrirez un programme `C++` utilisant l'algorithme de Runge-Kutta du 4ème ordre en coordonnées cartésiennes et un schéma à pas de temps adaptatif (Notes de cours section 2.5.2) **vous-mêmes**. Vous pouvez vous limiter au cas $N = 3$ et au mouvement en deux dimensions ($d = 2$). On vous propose quelques indications de programmation à la

section 4.9. On écrira et programmera aussi les expressions de l'énergie mécanique et de la quantité de mouvement du système.

4.2 "Houston, we've had a problem" [10 pts]

La sonde Apollo 13, de masse $m_A = 5809$kg, diamètre $d = 3.9$m, se trouve, au temps $t = 0$, à une distance $r_0 = 314159$km du centre de la Terre, entre la Terre et la Lune. Sa vitesse, par rapport au référentiel de la Terre, est $|v_0| = 1.2$km/s. La sonde n'a plus assez de carburant pour modifier la norme de sa vitesse, mais elle peut encore ajuster sa direction. Pour ramener Apollo 13 sur Terre, il faut qu'elle se mette sur une orbite dont la distance minimale à la surface de la Terre est de $h = 10000$m (rayon $R_T = 6378.1$km, masse $m_T = 5.972 \times 10^{24}$kg).

Dans cette section, on négligera l'effet de la Lune et des autres corps célestes. Pour les calculs analytiques, cela revient à étudier le problème à 1 corps (mouvement d'Apollo 13 dans le champ terrestre). Pour les simulations numériques, cela revient à prendre une masse de la "Lune" très petite, 10^{-14} fois sa valeur réelle de $m_L = 7.3477 \times 10^{22}$kg.

Dans cette section, on négligera la présence de l'atmosphère terrestre ($\rho_0 = 0$).

(a) Calculer analytiquement les composantes du vecteur vitesse d'Apollo 13 au temps $t = 0$ pour qu'elle suive l'orbite de rentrée décrite ci-dessus. Calculer la vitesse maximale v_{max} lorqu'elle sera au point le plus proche de la Terre.

(b) Résoudre numériquement la trajectoire d'Apollo 13 avec un pas de temps Δt fixe. [N.B. : Il suffit de simuler 2 jours]. Faire une étude de convergence de l'altitude minimale h_{min} et de v_{max} avec Δt et comparer avec le résultat analytique.

(c) Implémenter le pas de temps adaptatif dans le code et calculer la trajectoire. Faire une étude de convergence, en variant ϵ, qui est la précision requise par pas de temps. Illuster comment le pas de temps Δt change au cours de la simulation. Comparer le nombre de pas de temps nécessaires pour obtenir un résultat sur h_{min} de précision donnée avec le schéma à Δt fixe.

4.3 "Houston we're going to have a real problem" [10 pts]

On tient compte maintenant de la présence de l'atmosphère terrestre, $\rho_0 = 1.2$kg/m^3, $\lambda = 7238.2$m.

(a) Calculer numériquement la trajectoire, en utilisant le schéma avec pas de temps adaptatif, pour les conditions initiales trouvées à la section précédente. Quelle est l'accélération maximale subie par les astronautes ? Quelle est la puissance maximale de la force de traînée ? Faire une étude de convergence.

(b) Ajuster la direction initiale de Apollo13 pour minimiser l'accélération maximale subie par les astronautes, tout en atterissant en traversant une seule fois l'atmosphère.

4.4 Deux corps : Terre et Lune [5 pts]

On considère maintenant le système (Terre, Lune), avec leurs masses réelles. (On peut ignorer l'atmosphère).

(a) On cherchera des trajectoires circulaires uniformes dans le référentiel du centre de masse, avec une distance Terre-Lune $d = 384748$km. Il s'agit de trouver analytiquement les conditions initiales pour les positions et vitesses de la Terre et de la Lune.

(b) Effectuer des simulations avec ces conditions initiales et vérifier que les propriétés physiques du mouvement sont bien satisfaites : conservation de l'énergie mécanique, conservation de la quantité de mouvement et, dans ce cas, distance Terre-Lune $d_{TL} = d =$ constante. *Indication : simuler une dizaine de périodes de révolution.*

4.5 Trois corps : Terre, Lune et Apollo 13 [7 pts]

On revient sur le problème de ramener Apollo 13 près de la Terre. Attention, on se trouve maintenant dans le référentiel du centre de masse Terre-Lune, et donc il faut modifier les position et vitesse initiales d'Apollo 13 en conséquence (elles seront différentes des position et vitesse calculées dans le référentiel de la Terre).

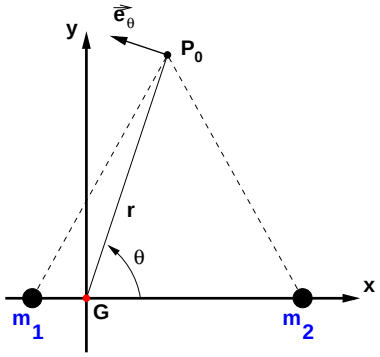
(a) En ignorant la présence de l'atmosphère ($\rho_0 = 0$), refaire une série de simulations avec le schéma adaptatif, en variant la direction de la vitesse initiale pour obtenir l'altitude $h_{min} = 10000$m avec une précision meilleure que 100m.

(b) En rajoutant la présence de l'atmosphère ($\rho_0 = 1.2\text{kg/m}^3$), ajuster de nouveau la direction initiale de Apollo13 pour minimiser l'accélération maximale comme en 4.3(b).

4.6 Trois corps : Terre, Lune et 3e corps - points de Lagrange [8 pts]

On aimerait savoir s'il est possible de mettre un 3ème corps céleste dans une orbite telle qu'elle garde des distances constantes par rapport à la Terre **et** à la Lune. On supposera ce troisième corps de masse m_A ($m_a \ll m_T, m_L$). On se place, pour les calculs analytiques, dans le référentiel $\mathcal{R}'$ tournant avec la Terre et la Lune, d'axe de rotation au centre de masse G du système Terre-Lune. Pour les calculs numériques, on reste dans le référentiel du centre de masse du système Terre-Lune. N'oubliez donc pas de transformer les positions et vitesses lors de changements de référentiel.

(a) Le 3e corps est placé en une position initiale P_0 formant un triangle équilatéral avec la Terre et la Lune, dit "point de Lagrange L4". Sa vitesse initiale est $\vec{v}_3(0) = \Omega r \vec{e}_\theta$, où $r = |\overrightarrow{GP_0}|$, Ω = vitesse angulaire du système Terre-Lune, (r, θ) = coordonnées polaires centrées en G. Montrer, par des simulations numériques, que le 3e corps reste en tous temps à égale distance de la Terre et de la Lune (simuler au moins 1 an).



(b) Placer le 3e corps à quelque distance (1000km ou plus) de ce point de Lagrange et représenter les trajectoires dans le référentiel Terre-Lune $\mathcal{R}'$.

4.7 Facultatif

(a) Bring the boys back home - essayez de trouver la bonne vitesse initiale d'Apollo 13 pour les ramener vivants sur Terre, en tenant compte à la fois de la Lune et de l'atmosphère terrestre (minimiser les valeurs maximales de l'accélération et de la puissance dissipée, tout en évitant qu'Appollo13 ne rebondisse sur l'atmosphère).

(b) trouver les positions des autres points de Lagrange, et examiner la stabilité des orbites à leur voisinage.

4.8 Soumission du rapport et du code C++ [5 pts de qualité générale]

(a) Préparer le fichier du rapport en format `pdf` portant le nom `RapportExercice4_Nom1_nom2.pdf`, ainsi que le fichier source LaTeX `RapportExercice4_Nom1_nom2.tex`.

(b) Préparer le fichier source C++ `Exercice4_Nom1_nom2.cpp`.

(c) Préparer le fichier script Matlab `Exercice4_Nom1_nom2.m`.

(d) Le lien de soumission se trouve dans la boîte du premier exercice sur notre site "Moodle" et peut directement être atteint en cliquant ici.

4.9 Guide de programmation

Cet exercice vous offre une expérience plus riche que les autres, parce que vous allez écrire votre code vous-même. Afin d'en faciliter l'écriture, veuillez tenir compte des points suivants qui pourront vous être utiles.

4.9.1 Conventions physiques

Le but de cet excercice est d'étudier le problème à N corps en utilisant les équations du mouvement sous la forme

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}) \tag{5}$$

Ici, le vecteur $\mathbf{y}$ comprend les positions *et* les vitesses de *tous les corps*. Vous devez choisir une convention pour $\mathbf{y}$. Une possibilité est donnée par $\mathbf{y} = (\mathbf{r}_1, \ldots, \mathbf{r}_N, \mathbf{v}_1, \ldots, \mathbf{v}_N)$ avec $\mathbf{r}_i$ la position du corps i, et $\mathbf{v}_i$ sa vitesse. Pour le cas à deux dimensions, cela devient par exemple définir $\mathbf{y} = (x_1, y_1, x_2, y_2, \ldots, v_{1;x}, v_{1;y}, \ldots)$.

Choisir une convention pour y et en déduire la forme correspondante de f !

On programmera le schéma de Runge-Kutta d'ordre 4 sous la forme donnée par l'Eq.(2.97) des notes de cours.

4.9.2 La structure générale

On propose d'organiser l'architecture de l'outil numérique autour de trois étapes principales : l'initialisation des conditions initiales, l'évolution du système physique et le post-traitement de la sortie.

Pour l'initialisation, vous pouvez reprendre le template `ConfigFile` de l'exercice 3. Vous pourrez ainsi adapter le script Matlab `ParameterScan.m` pour varier un ou plusieurs paramètres.

On propose d'utiliser une structure similaire aux exercices précédents, avec une classe `Exercice4` dont le constructeur appelle `ConfigFile` pour lire les paramètres, et dont la méthode `run()` effectue l'itération temporelle en appelant la méthode `step()` et l'écriture des résultats `printOut()`.

Dans la boucle temporelle de `run()`, vous devrez inclure des modifications pour pouvoir exécuter soit avec un pas de temps Δt fixe, soit avec le schéma Δt adaptatif.

La méthode `step()` pourra faire appel à une fonction `std::valarray acceleration(std::valarray const& y)` qui calculera les composantes du vecteur $\mathbf{f}(\mathbf{y})$ (voir la sous-section suivante pour l'utilisation des `valarrays`). L'utilisation des références constantes permet d'éviter la copie du tableau à chaque appel de la fonction.

La méthode `printOut()` peut utiliser une fréquence d'échantillonage pour écrire tous les `nsampling` pas de temps afin d'éviter les fichiers de sortie trop volumineux.

Une fois que vous avez généré votre fichier de sortie, il vous faut le lire au sein de fonctions ou de scripts Matlab afin d'en faire ressortir l'information pertinente. Commencez avec une fonction simple comme par exemple le traçage de trajectoires, puis soyez créatifs et essayez d'avoir des figures simples qui synthétisent votre compréhension.

4.9.3 Quelques propositions techniques et d'approche

(a) **Utiliser** `valarrays` pour les vecteurs. Cela facilitera le calcul. En effet, pour des vecteurs **a**, **b**, **c** de la même dimension codés sous forme de valarrays, il suffit d'écrire

```
c=a+b
```

pour faire la somme sans avoir à écrire une boucle sur les indices.

(b) Le concept de `slices` est extrêmement utile pour accéder à un groupe d'indices d'un vecteur. Par exemple, imaginons un vecteur

$$\mathbf{m} = (m_0, m_1, \ldots, m_3)$$

dont on veut remplacer les composantes m_2 et m_3 par les composantes n_0 et n_1 du vecteur $\mathbf{n} = (n_0, n_1)$.

Si **m** and **n** sont du type `valarray`, en utilisant la commande slice, il suffit d'crire

```
m[std::slice(2,2,1)] = n
```

L'opration inverse

```
n = m[std::slice(2,2,1)]
```

est aussi possible.

En fait, la fonction `slice(i,j,k)` commence avec l'indice i du vecteur et sélectionne j indices régulièrement espacés de k positions.

(c) Il est indispensable que vous structuriez et planifiez votre code avant de vous lancer dans l'écriture :
— Essayer de faire un dessin avec des liaisons, les tâches et les variables principales.
— Discuter en groupe votre modèle.

(d) Utiliser les documentations pour `C++` (par exemple `www.cplusplus.com`)