CONCOURS CENTRALE•SUPÉLEC

Informatique

MP,PC,PSI,TSI

3 heures — Calculatrices autorisées

2015

Autour de la dynamique gravitationnelle

Modéliser les interactions physiques entre un grand nombre de constituants mène à l'écriture de systèmes différentiels pour lesquels, en dehors de quelques situations particulières, il n'existe aucune solution analytique. Les problèmes de dynamique gravitationnelle et de dynamique moléculaire en sont deux exemples. Afin d'analyser le comportement temporel de tels systèmes, l'informatique peut apporter une aide substantielle en permettant leur simulation numérique. L'objet de ce sujet, composé de quatre parties, est l'étude de solutions algorithmiques en vue de simuler une dynamique gravitationnelle afin, par exemple, de prédire une éclipse ou le passage d'une comète.

Les programmes doivent être écrits en langage python et les requêtes de base de données en langage SQL. Les candidats sont libres de définir et de programmer toute fonction auxiliaire dont ils estiment avoir besoin pour répondre aux questions posées. Ils veilleront dans ce cas à définir précisément, le rôle de chaque fonction introduite, ses paramètres et son résultat. Ils peuvent également utiliser librement les fonctions de la bibliothèque standard Python, en particulier celles du module `math`.

Lorsque le sujet demande l'écriture d'une fonction python, la réponse doit commencer par l'entête de la fonction (instruction `def`*). D'autre part, si le sujet précise que la fonction prend un paramètre d'un certain type ou qui répond à une certaine condition, la fonction n'a pas à vérifier la conformité de l'argument reçu.*

La lisibilité des codes produits, tant en python qu'en SQL, est un élément important d'appréciation.

I Quelques fonctions utilitaires

I.A – Donner la valeur des expressions python suivantes :

I.A.1) `[1, 2, 3] + [4, 5, 6]`

I.A.2) `2 * [1, 2, 3]`

I.B – Écrire une fonction python `smul` à deux paramètres, un nombre et une liste de nombres, qui multiple chaque élément de la liste par le nombre et renvoie une nouvelle liste : `smul(2, [1, 2, 3])` → `[2, 4, 6]`.

I.C – ***Arithmétique de listes***

I.C.1) Écrire une fonction python `vsom` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la somme terme à terme de ces deux listes :
`vsom([1, 2, 3], [4, 5, 6])` → `[5, 7, 9]`.

I.C.2) Écrire une fonction python `vdif` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la différence terme à terme de ces deux listes (la première moins la deuxième) : `vdif([1, 2, 3], [4, 5, 6])` → `[-3, -3, -3]`.

II Étude de schémas numériques

Soient y une fonction de classe C^2 sur $\mathbb{R}$ et $t_{\min}$ et $t_{\max}$ deux réels tels que $t_{\min} < t_{\max}$. On note I l'intervalle $[t_{\min}, t_{\max}]$. On s'intéresse à une équation différentielle du second ordre de la forme :

$$\forall t \in I \qquad y''(t) = f(y(t)) \tag{II.1}$$

où f est une fonction donnée, continue sur $\mathbb{R}$. De nombreux systèmes physiques peuvent être décrits par une équation de ce type.

On suppose connues les valeurs $y_0 = y(t_{\min})$ et $z_0 = y'(t_{\min})$. On suppose également que le système physique étudié est conservatif. Ce qui entraine l'existence d'une quantité indépendante du temps (énergie, quantité de mouvement, ...), notée E, qui vérifie l'équation (II.2) où $g' = -f$.

$$\forall t \in I \qquad \frac{1}{2}y'(t)^2 + g(y(t)) = E \tag{II.2}$$

II.A – ***Mise en forme du problème***

Pour résoudre numériquement l'équation différentielle (II.1), on introduit la fonction $z : I \to \mathbb{R}$ définie par $\forall t \in I, z(t) = y'(t)$.

II.A.1) Montrer que l'équation (II.1) peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$, noté (S).

II.A.2) Soit n un entier strictement supérieur à 1 et $J_n = [\![0, n-1]\!]$. On pose $h = \dfrac{t_{\max} - t_{\min}}{n-1}$ et $\forall i \in J_n$, $t_i = t_{\min} + ih$. Montrer que, pour tout entier $i \in [\![0, n-2]\!]$,

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} z(t)\,\mathrm{d}t \qquad \text{et} \qquad z(t_{i+1}) = z(t_i) + \int_{t_i}^{t_{i+1}} f(y(t))\,\mathrm{d}t \tag{II.3}$$

La suite du problème exploite les notations introduites dans cette partie et présente deux méthodes numériques dans lesquelles les intégrales précédentes sont remplacées par une valeur approchée.

II.B – Schéma d'Euler explicite

Dans le schéma d'Euler explicite, chaque terme sous le signe intégrale est remplacé par sa valeur prise en la borne inférieure.

II.B.1) Dans ce schéma, montrer que les équations (II.3) permettent de définir deux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$, où y_i et z_i sont des valeurs approchées de $y(t_i)$ et $z(t_i)$. Donner les relations de récurrence permettant de déterminer les valeurs de y_i et z_i connaissant y_0 et z_0.

II.B.2) Écrire une fonction `euler` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$ et $(z_j)_{i\in J_n}$. Vous justifierez le choix des paramètres transmis à la fonction.

II.B.3) Pour illustrer cette méthode, on considère l'équation différentielle $\forall t \in I, y''(t) = -\omega^2 y(t)$ dans laquelle ω est un nombre réel.

a) Montrer qu'on peut définir une quantité E, indépendante du temps, vérifiant une équation de la forme (II.2).

b) On note E_i la valeur approchée de E à l'instant t_i, $i \in J_n$, calculée en utilisant les valeurs approchées de $y(t_i)$ et $z(t_i)$ obtenues à la question II.B.1. Montrer que $E_{i+1} - E_i = h^2\omega^2 E_i$.

c) Qu'aurait donné un schéma numérique qui satisfait à la conservation de E ?

d) En portant les valeurs de y_i et z_i sur l'axe des abscisses et l'axe des ordonnées respectivement, quelle serait l'allure du graphe qui respecte la conservation de E ?

e) La mise en œuvre de la méthode d'Euler explicite génère le résultat graphique donné figure 1 à gauche. Dans un système d'unités adapté, les calculs ont été menés en prenant $y_0 = 3$, $z_0 = 0$, $t_{\texttt{min}} = 0$, $t_{\texttt{max}} = 3$, $\omega = 2\pi$ et $n = 100$.

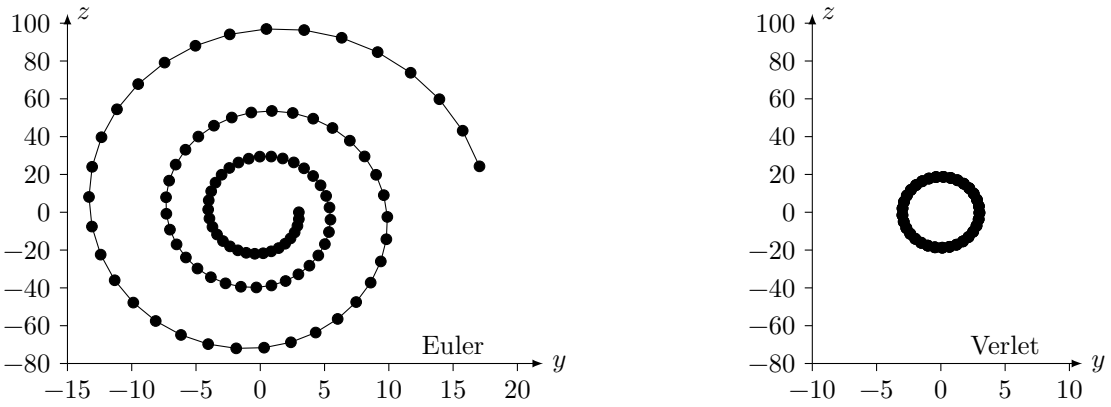

Figure 1

En quoi ce graphe confirme-t-il que le schéma numérique ne conserve pas E ? Pouvez-vous justifier son allure ?

II.C – Schéma de Verlet

Le physicien français Loup Verlet a proposé en 1967 un schéma numérique d'intégration d'une équation de la forme (II.1) dans lequel, en notant $f_i = f(y_i)$ et $f_{i+1} = f(y_{i+1})$, les relations de récurrence s'écrivent

$$y_{i+1} = y_i + hz_i + \frac{h^2}{2} f_i \qquad \text{et} \qquad z_{i+1} = z_i + \frac{h}{2}\left(f_i + f_{i+1}\right)$$

II.C.1) Écrire une fonction `verlet` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$ et $(z_j)_{i\in J_n}$.

II.C.2) On reprend l'exemple de l'oscillateur harmonique (question II.B.3) et on compare les résultats obtenus à l'aide des schémas d'Euler et de Verlet.

a) Montrer que dans le schéma de Verlet, on a $E_{i+1} - E_i = O(h^3)$.

b) La mise en œuvre du schéma de Verlet avec les mêmes paramètres que ceux utilisés au II.B.3e donne le résultat de la figure 1 à droite. Interpréter l'allure de ce graphe.

c) Que peut-on conclure sur le schéma de Verlet ?

III Problème à N corps

On s'intéresse à présent à la dynamique d'un système de N corps massifs en interaction gravitationnelle. Dans la suite, les corps considérés sont assimilés à des points matériels P_j de masses m_j où $j \in [\![0, N-1]\!]$, $N \geqslant 2$ étant un entier positif donné. Le mouvement de ces points est étudié dans un référentiel galiléen muni d'une base orthonormée. L'interaction entre deux corps j et k est modélisée par la force gravitationnelle. L'action exercée par le corps k sur le le corps j est décrite par la force $\overrightarrow{F_{k/j}} = G\dfrac{m_j m_k}{r_{jk}^3}\overrightarrow{P_jP_k}$ où r_{jk} est la distance séparant les corps j et k ($r_{jk} = \|\overrightarrow{P_jP_k}\|$) et $G = 6{,}67 \times 10^{-11}$ N·m^2·kg^{-2} la constante de gravitation universelle.

À tout instant t_i avec $i \in [\![0, n]\!]$, chaque corps de masse m_j est repéré par ses coordonnées cartésiennes (x_{ij}, y_{ij}, z_{ij}) et les composantes de son vecteur vitesse $(v_{xij}, v_{yij}, v_{zij})$ dans le référentiel de référence.

Trois listes sont utilisées pour représenter ce système en python

- `masse` conserve les masses de chaque corps : `masse[j]` = m_j ;
- `position` contient les positions successives de chaque corps : `position[i][j]` = $[x_{ij}, y_{ij}, z_{ij}]$;
- `vitesse` mémorise les vitesses successives de chaque corps : `vitesse[i][j]` = $[v_{xij}, v_{yij}, v_{zij}]$.

L'objet de la suite du problème est de construire ces listes en mettant en œuvre l'algorithme de Verlet.

III.A – Position du problème

III.A.1) Exprimer la force $\overrightarrow{F_j}$ exercée sur le corps P_j par l'ensemble des autres corps P_k, avec $k \neq j$.

III.A.2) Écrire une fonction python `force2(m1, p1, m2, p2)` qui prend en paramètre les masses (`m1` et `m2` en kilogrammes) et les positions (`p1` et `p2`, sous forme de listes de trois coordonnées cartésiennes en mètres) de deux corps 1 et 2 et qui renvoie la valeur de la force exercée par le corps 2 sur le corps 1, sous la forme d'une liste à trois éléments représentant les composantes de la force dans la base de référence, en newtons.

III.A.3) Écrire une fonction `forceN(j, m, pos)` qui prend en paramètre l'indice `j` d'un corps, la liste des masses des N corps du système étudié ainsi que la liste de leurs positions et qui renvoie $\overrightarrow{F_j}$, la force exercée par tous les autres corps sur le corps `j`, sous la forme d'une liste de ses trois composantes cartésiennes.

III.B – Approche numérique

III.B.1) Expliciter la structure et la signification de `position[i]` et `vitesse[i]`.

III.B.2) Écrire une fonction `pos_suiv(m, pos, vit, h)` qui prend en paramètres la liste des masses des N corps du système étudié (en kilogrammes), la liste de leurs positions (en mètres) à l'instant t_i, la liste de leurs vitesses (en mètres par seconde) au même instant et le pas d'intégration h (en secondes) et qui renvoie la liste des positions des N corps à l'instant t_{i+1} calculées en utilisant le schéma de Verlet.

III.B.3) Écrire une fonction `etat_suiv(m, pos, vit, h)` qui prend les mêmes paramètres que la fonction `pos_suiv` et qui renvoie la liste des positions (en mètres) et la liste des vitesses (en m/s) des N corps à l'instant t_{i+1} calculées en utilisant le schéma de Verlet.

III.B.4) En notant τ_N la durée des calculs pour un nombre N de corps, la mise en œuvre de la fonction `etat_suiv` a donné le résultat graphique de la figure 2 où on a porté $\ln(N)$ en abscisse et $\ln(\tau_N)$ en ordonnée.

a) Quelle relation simple peut-on établir entre $\ln(\tau_N)$ et $\ln(N)$ à partir de la figure 2 ?

b) Quelle hypothèse peut-on émettre quant à la complexité de l'algorithme étudié ?

III.B.5)

a) Estimer la complexité temporelle de la fonction `etat_suiv` sous la forme $O(N^\alpha)$.

b) Comparer avec le résultat obtenu à la question III.B.4.

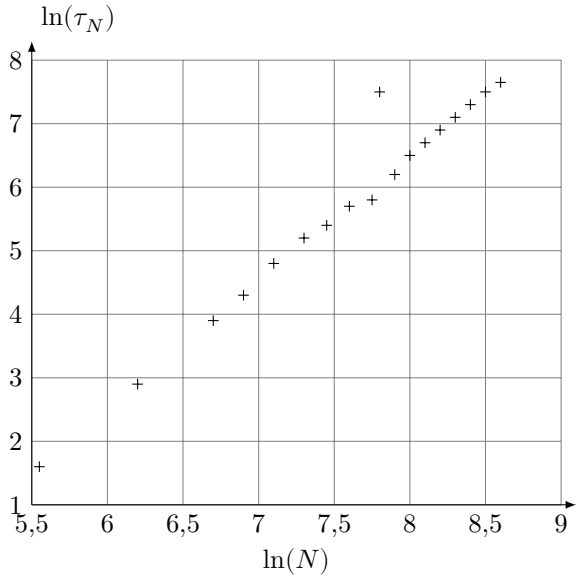

Figure 2

IV Exploitation d'une base de données

À partir de mesures régulièrement effectuées par différents observatoires, une base de données des caractéristiques et des états des corps célestes de notre Système solaire est maintenue à jour. L'objectif de cette partie est d'extraire de cette base de données les informations nécessaires à la mise en œuvre des fonctions développées dans les parties précédentes, puis de les utiliser pour prévoir les positions futures des différentes planètes. Les données à extraire sont les masses des corps étudiés et leurs états (position et vitesse) à l'instant t_{min} du début de la simulation.

Une version simplifiée, réduite à deux tables, de la base de données du Système solaire est donnée figure 3. Les masses sont exprimées en kilogrammes, les distances en unités astronomiques ($1\,\text{au} = 1{,}5 \times 10^{11}\,\text{m}$) et les vitesses en kilomètres par seconde. Le référentiel utilisé pour exprimer les composantes des positions et des vitesses est galiléen, orthonormé et son centre est situé à proximité du Soleil.

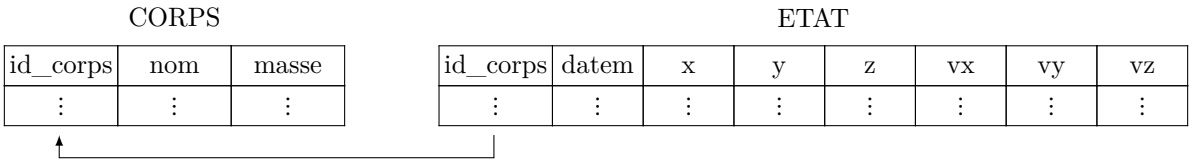

Figure 3 Schéma de la base de données

La table `CORPS` répertorie les corps étudiés, elle contient les colonnes
- `id_corps` (clé primaire) entier identifiant chaque corps ;
- `nom`, chaine de caractères, désigne le nom usuel du corps ;
- `masse` de type flottant, contient la masse du corps.

La table `ETAT` rassemble l'historique des états successifs (positions et vitesses) des corps étudiés. Elle est constituée de huit colonnes :
- `id_corps` de type entier, identifie le corps concerné ;
- `datem` est la date de la mesure, sous forme d'un entier donnant le nombre de secondes écoulées depuis un instant d'origine ;
- trois colonnes de type flottant pour les composantes de la position `x`, `y`, `z` ;
- trois colonnes de type flottant pour les composantes de la vitesse `vx`, `vy`, `vz`.

IV.A – Écrire une requête SQL qui renvoie la liste des masses de tous les corps étudiés.

IV.B – Les états des différents corps ne sont pas forcément tous déterminés exactement au même instant. Nous allons assimiler l'état initial (à la date t_{min}) de chaque corps à son dernier état connu antérieur à t_{min}.

Dans toute la suite, on supposera que la valeur de t_{min}, sous le format utilisé dans la table `ETAT`, est accessible à toute requête SQL via l'expression `tmin()`.

IV.B.1) On souhaite d'abord vérifier que tous les corps étudiés disposent d'un état connu antérieur à `tmin()`. Le nombre de corps présents dans la base est obtenu grâce à la requête `SELECT count(*) FROM corps`. Écrire une requête SQL qui renvoie le nombre de corps qui ont au moins un état connu antérieur à `tmin()`.

IV.B.2) Écrire une requête SQL qui renvoie, pour chaque corps, son identifiant et la date de son dernier état antérieur à `tmin()`.

IV.B.3) Le résultat de la requête précédente est stocké dans une nouvelle table `date_mesure` à deux colonnes :
- `id_corps` de type entier, contient l'identifiant du corps considéré ;
- `date_der` de type entier, correspond à la date du dernier état connu du corps considéré, antérieur à `tmin()`.

Pour simplifier la simulation, on décide de négliger l'influence des corps ayant une masse strictement inférieure à une valeur fixée `masse_min()` et de ne s'intéresser qu'aux corps situés dans un cube, centré sur l'origine du référentiel de référence et d'arête `arete()` donnée. Les faces de ce cube sont parallèles aux plans formés par les axes du référentiel de référence.

Écrire une requête SQL qui renvoie la masse et l'état initial (sous la forme `masse, x, y, z, vx, vy, vz`) de chaque corps retenu pour participer à la simulation. Classez les corps dans l'ordre croissant par rapport à leur distance à l'origine du référentiel.

IV.C – On dispose des variables python `t0`, `p0`, `v0` et `masse` initialisées à partir du résultat de la requête précédente. `t0` est un entier qui donne la date des conditions initiales : il correspond à t_{min} et à `tmin()`. `p0` est une liste de longueur N, chaque élément de `p0` est une liste à 3 éléments de la forme `[x, y, z]` représentant la position initiale d'un corps, en unité astronomique. `v0` a une structure identique mais indique les vitesses initiales des corps considérés, en km/s. `masse` est décrite en partie III.

Écrire la fonction python `simulation_verlet(deltat, n)` qui prend en paramètre un incrément de temps en secondes (`deltat > 0`) et un nombre d'itérations (`n > 0`) et qui renvoie la liste des positions des corps considérés pour chaque instant `t0`, `t0 + deltat`, ..., `t0 + n*deltat` (cf variable `position` définie en partie III). Les calculs seront menés en utilisant le schéma d'intégration de Verlet, le résultat sera fourni en unité astronomique.

• • • FIN • • •