**Devoir de vacances**

*Ce devoir de vacances permet de réviser et pratiquer le programme de première année sur des questions semblables à celles d'un sujet de concours. La consigne n'est pas toujours parfaitement explicite ; c'est alors au candidat de faire preuve d'initiative et de bon sens.*

Les unités sont celles du système international.
On se propose de modéliser le système solaire en repérant le soleil, 8 planètes, 5 planètes naines, 181 lunes, 552 894 astéroïdes et 3 083 comètes dans un repère orthonormé centré sur le soleil (d'unité le mètre). On considère que le rayon du système solaire mesure 10 milliards de kilomètres.
On peut utiliser des entiers (type int) ou des nombres à virgule flottantes (type float) pour repérer ces objets.
On suppose qu'on a importé numpy avec **import** numpy **as** np.

**Q1** Quel espace mémoire sera nécessaire si on utilise des entiers ? Est-ce une contrainte forte ?

**Q2** Quel espace mémoire sera nécessaire si on utilise des nombres à virgule flottante ?

**Q3** Quel type de variable, float ou int, fournira le plus de chiffres significatifs ? Justifier.

Les objets sont classés par masse dans l'ordre décroissant et chaque objet est repéré par une référence de type entier correspondant à son rang dans ce classement. Le soleil porte le rang 0, Jupiter le rang 1, la Terre le rang 5…
On constitue une liste *masses* des masses de ces objets, de type float, triée dans l'ordre décroissant.

**Q4** Écrire une fonction d'entête **def** insere_masse(masse): qui insère une masse (float) à la bonne position dans la liste *masses* avec une complexité en O(ln *n*),
où *n* = len(masses)
On rappelle qu'une fonction Python peut modifier une liste sans qu'elle soit en argument.

On dispose d'un tableau numpy (np.array) *positions* repérant les différents objets par leurs coordonnées cartésiennes sous forme de tableau de nombres à virgules flottantes [*x, y, z*]. Ce tableau est classé dans l'ordre croissant des références.

**Q5** A quoi est égal positions[0] ?

**Q6** (attention, cette question comporte un chouïa de sciences physiques)
Écrire la fonction d'entête **def** force(i): qui renvoie la somme des forces exercées sur le corps de référence i, dues à l'attraction des autres corps. Cette fonction utilise les listes *positions* et *masses*.

On utilise également un tableau *vitesses*, similaire au tableau *positions* mais contenant les composantes des vecteurs vitesses de chaque corps.
On décide d'appliquer la méthode d'Euler pour construire une liste *res_positions* avec les positions des différents corps sur un intervalle de temps [0, *T*]. On découpe l'intervalle de temps en *N* intervalles d'amplitude *dt*.
Cette liste est assimilable à un tableau dont la première ligne contient le tableau *positions* à l'instant initial, la deuxième ligne le tableau *positions* à l'instant *dt*, … et la dernière ligne à l'instant *T*.

Dans la suite du problème, on suppose que les fonctions pourront accéder aux tableaux *vitesses*, *positions* et *masses* sans qu'ils soient passés en argument.

**Q7** Écrire la fonction d'entête **def** calcule_vitesse(vitesses, dt): qui calcule les nouvelles vitesses suivant la méthode d'Euler, au bout de *dt* secondes en appliquant le principe fondamental de la dynamique, et renvoie ces nouvelles vitesses dans un tableau. Cette fonction effectue donc une étape de la méthode d'Euler.

**Q8** Écrire la fonction d'entête **def** calcule_position(positions, dt): qui calcule les nouvelles positions au bout de *dt* secondes, de manière similaire, en réalisant une étape de la méthode d'Euler, et renvoie un tableau avec les nouvelles positions.

**Q9** Écrire la fonction d'entête **def** simulation(T,n): qui applique la méthode d'Euler sur l'intervalle [0, *T*], et renvoie la liste *res_positions*. On considère que les variables *masses*, *positions* et *vitesses* ont déjà été définies avec leurs valeurs initiales avant l'appel de la fonction. On pourra utiliser la méthode d'Euler explicite pour les vitesses et implicite pour les positions.

**Q10** Quelle est, en fonction du nombre *n* de corps étudiés et du nombre *N* d'intervalles utilisés dans la méthode d'Euler, la complexité temporelle de la fonction simulation ?

On souhaite effectuer une simulation sur une année avec un intervalle de temps *dt* d'une seconde.

**Q11** Quelle quantité de mémoire nécessite-t-on ?

**Q12** La simulation est-elle réalisable ?

**Q13** Proposer une piste pour améliorer la simulation.

On suppose maintenant que l'on a modifié la fonction simulation et qu'elle renvoie une liste *res_vitesses* similaire à la liste *res_positions* mais contenant les vecteurs vitesses des différents corps à chaque instant de la simulation. La distance parcourue par un corps au long de la simulation est égale à $\int_0^T \left\|\vec{V}(t)\right\| \mathrm{d}t$.

**Q14** Écrire une fonction d'entête **def** distance_parcourue(i,res_vitesses,T): calculant la distance parcourue par le corps de référence *i* lors de la simulation par la méthode des trapèzes.

La résolution de l'équation de Kepler permet de déterminer la position d'une planète à une date donnée. Dans le cas d'une orbite elliptique, on a

$$E - e.\sin(E) = M.$$

- *e* est l'excentricité de l'ellipse.
- *E* est l'anomalie excentrique et donne la position de la planète.
- *M* est l'anomalie moyenne et est liée à la date.

**Q15** Écrire une fonction d'entête **def** resolution_Kepler(e,M,epsilon): déterminant une approximation de *E* à partir des valeurs de *e* et *M* par la méthode de Newton. On rappelle que cette méthode construit une suite $(E_n)$ ayant pour limite la solution de l'équation considérée.
On prendra $E_0 = M$ et on interrompra les calculs dès que $|E_{n+1} - E_n| \leq$ epsilon.

Pour initialiser les variables *masses, positions, vitesses,* on utilise d'une base de données constituée d'une table corps_celestes dont nous donnons la première ligne :

| id | nom | date | heure | masse | x | y | z | vx | vy | vz |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 'Soleil' | 1972-02-15 | 09:45:00 | 1.989E30 | 0 | 0 | 0 | 0 | 0 | 0 |

Cette table contient tous les objets répertoriés dans le système solaire ainsi que d'autres objets extérieurs au système solaire.

**Q16** Écrire une requête SQL renvoyant pour chaque date le nombre de données disponibles.

**Q17** Écrire une requête SQL renvoyant les données relatives aux objets du système solaire à une date *d*.

On dispose d'une fonction requete_BDD(sql) qui retourne le résultat d'une requête SQL sous forme de liste s'il n'y a qu'une valeur par enregistrement, ou de liste de listes s'il y a plusieurs valeurs par enregistrement.

**Q18** Écrire une fonction d'entête **def** initialisation(d): permettant d'initialiser les variables globales *masses*, *positions*, *vitesses* en utilisant les mesures réalisées à une date d.

La plupart des objets du système solaire évoluent à peu près dans un même plan. On souhaite déterminer le plan d'équation $z = ax + by + c$ passant le plus près d'une liste *objets* d'objets repérés par leurs coordonnées $[[x_1, y_1, z_1],\ldots, [x_n, y_n, z_n]]$, ce qui revient à minimiser $\|AX - B\|^2$ avec

$$\text{A} = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix}, \text{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \text{ et B} = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}.$$

On appelle pseudo-solution du système $AX = B$ une solution du système $A^TAX = A^TB$. On démontre que $X$ est une pseudo-solution du système $AX = B$ si et seulement si $AX$ est le projeté orthogonal de $B$ sur Im $A$, c'est-à-dire que $X$ minimise $\|AX - B\|^2$.

<u>On traitera les questions **Q19**, **Q20** et **Q21** sans utiliser la bibliothèque numpy.</u>

**Q19** Écrire une fonction d'entête **def** ata(objets): qui prend la liste *objets* en arguments et retourne une liste de listes représentant la matrice $A^TA$.

**Q20** Écrire une fonction d'entête **def** atb(objets): qui retourne la liste des coefficients de la matrice $A^TB$.

On admet que la matrice $A^TA$ est inversible, ce qui est le cas si rg $A = 3$.

**Q21** Écrire une fonction d'entête **def** resoudre(ATA,ATB): résolvant le système $A^TAX = A^TB$ par la méthode du pivot de Gauss. Cette fonction renverra la liste $[a, b, c]$ des coefficients de l'équation du plan. ATA et ATB sont les résultats renvoyés respectivement par les fonctions ata et atb.

<u>On décide maintenant de reprendre le problème traité dans les trois questions précédentes en utilisant les fonctionnalités de la bibliothèque numpy.</u>

**Q22** Écrire une fonction d'entête **def p**lan(objets): renvoyant une matrice colonne $[a, b, c]$ des coefficients de l'équation du plan passant au plus près des objets $[[x_1, y_1, z_1],\ldots, [x_n, y_n, z_n]]$, en utilisant le calcul matriciel de la bibliothèque numpy.