Détermination de la masse du Soleil

Doc. 8,18

Résumé

L'objectif de ce document est de déterminer la masse du Soleil à partir des périodes de révolution et des demi-grands axes des orbites planétaires, en utilisant la troisième loi de Kepler.

§1. Données

- Périodes et demi-grand axes des planètes :

Planète	Période (ans)	Demi-grand axe ($\times 10^{10}$ m)
Mercure	0,24	5,79
Vénus	0,62	10,80
Terre	1,00	15,00
Mars	1,88	22,80
Jupiter	11,90	77,80
Saturne	29,50	143,00
Uranus	84,00	287,00
Neptune	165,00	450,00
Pluton	248,00	590,00

- La constante universelle de gravitation vaut : $G = 6{,}674 \times 10^{-11}\ \mathrm{m^3 \cdot kg^{-1} \cdot s^{-2}}$.

§2. Travail expérimental

Q1. Dans la zone (Q1) du code Python, saisir les valeurs des périodes et des demi-grands axes dans les listes `T` et `a`.

Réponse :

```
T = [0.24, 0.62, 1.0, 1.88, 11.9, 29.5, 84.0, 165.0, 248.0]                    # en années
a = [5.79e10, 10.8e10, 15e10, 22.8e10, 77.8e10, 143e10, 287e10, 450e10, 590e10]     # en m
```

Q2. Compléter le code de la zone (Q2) afin de convertir toutes les périodes en secondes.

Réponse :

```
for i in range(len(T)):
    T[i] = T[i] * nbre_jours_par_an * nbre_secondes_par_jour
```

Q3. Compléter le code de la zone Q3 afin de créer la liste des carrés des périodes de révolution des planètes.

Réponse :

```
T_carre = []

for periode in T:
    T_carre.append( periode**2 )
```

Q4. Compléter le code de la zone Q4 afin de créer la liste des cubes des demi-grands axes des planètes.

Réponse :

```
a_cube = []

for demi_grand_axe in a:
    a_cube.append( demi_grand_axe**3 )
```

Q5. Compléter le code de la zone Q5 afin de créer le graphique $T^2 = f(a^3)$.

Réponse :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(a_cube, T_carre, 'o')
plt.xlabel("$a^3 \ (\mathrm{m}^3)$")
plt.ylabel("$T^2 \ (\mathrm{s}^2)$")
plt.show()
```

Q6. Compléter le code de la zone Q6 afin d'ajuster un modèle aux données expérimentales.

Réponse :

```
# Nouvelles dates pour l'affichage des grandeurs modélisées
a_cube_mod = np.linspace(min(a_cube), max(a_cube), 101)

# Fonction modèle pour le comportement de T^2 en fonction de a^3
def modele(x, a):
    return a * x

# Détermination des paramètres optimaux pour T^2 en fonction de a^3
popt, pcov = curve_fit(modele, a_cube, T_carre)
a_mod = popt[0]

# Valeurs de T^2 modélisées
T_carre_mod = modele(a_cube_mod, a_mod)

print(f"Pente de la modélisation : {a_mod}")
```

Q7. Compléter le code de la zone Q7 afin de superposer les données expérimentales et le modèle obtenu.

Réponse :

```
plt.figure(figsize=(8, 6), dpi=100)
plt.plot(a_cube, T_carre, 'o')
plt.plot(a_cube_mod, T_carre_mod, '-')
plt.xlabel("$a^3 \ (\mathrm{m}^3)$")
plt.ylabel("$T^2 \ (\mathrm{s}^2)$")
plt.show()
```

Q8. À partir des grandeurs déjà définies dans le code, et en appliquant la troisième loi de Kepler, déterminer la masse du Soleil dans la zone (Q8).

Réponse : La troisième loi de Képler, pour le système solaire, stipule que :

$$\frac{T^2}{a^3} = \frac{4\pi^2}{GM_S} \iff M_S = \frac{4\pi^2}{G}\frac{a^3}{T^2}$$

La valeur du rapport a^3/T^2 est égale à l'inverse du contenu de la variable `a_mod`, donc :

$$M_S = \frac{4\pi^2}{Ga_{\text{mod}}}$$

La formule Python à écrire est donc :

```
MS = 4 * pi**2 / (G * a_mod)
```

On obtient : $M_S = 1{,}98 \times 10^{30}$ kg.

§3. Prolongement

Lors des ECE de physique-chimie, il est possible de rencontrer un code Python utilisant la **vectorisation** du module `numpy`.
Cela signifie que *les opérations peuvent être appliquées directement à l'ensemble des valeurs d'un tableau, sans utiliser de boucles* `for`, ce qui permet d'obtenir un code plus simple à écrire et plus efficace.

Voici les modifications et simplifications à apporter au code :

- zone 1

```
T = [0.24, 0.62, 1.0, 1.88, 11.9, 29.5, 84.0, 165.0, 248.0]  # en années
T = np.array(T)  # Vectorisation de T
a = [5.79e10, 10.8e10, 15e10, 22.8e10, 77.8e10, 143e10, 287e10, 450e10, 590e10]  # en m
a = np.array(a)  # Vectorisation de a
```

- zone 2 : `T = T * nbre_jours_par_an * nbre_secondes_par_jour`
- zone 3 : `T_carre = T**2`
- zone 4 : `a_cube = a**3`

À retenir

La **vectorisation** est possible *lorsque les données ne sont pas stockées dans de simples listes Python, mais dans des tableaux du module numpy (`ndarray`).* Elle permet alors d'appliquer une même opération à l'ensemble des éléments du tableau en une seule instruction.