Travaux dirigés – Outils Numériques

Feuille numéro 2

Manuel en ligne de *Python* : `https://docs.python.org/fr/3.5/tutorial/`

1 Courbes paramétriques

1.1 En 2D : Courbes de Lissajous

On importera les "libraries" suivantes au début du programme python :

```
import numpy as np
import matplotlib.pyplot as plt
```

Une masse est attachée à un chariot à deux degrés de liberté pouvant effectuer une translation suivant x et une autre suivant y. La position de la masse à l'instant t est définie par $x(t) = x_0 \cos(\omega_x t)$ et $y(t) = y_0 \sin(\omega_y t)$. On prendra $x_0 = 1$, $\omega_x = 1$

1. Créez un vecteur `t` allant de 0 à $3T_x$ avec $T_x = 2\pi/\omega_x$ (période de $x(t)$). On choisira un pas de $T_x/100$.
2. Calculer x puis y dans le cas où $\omega_x = \omega_y$, $y_0 = x_0$.
3. Tracez y en fonction de x (trajectoire de la masse dans le plan (x, y)). On utilisera un repère orthonormé. Placez des labels sur les axes et un titre à votre courbe.
4. Tracez la trajectoire quand $\omega_y = \omega_x$, $y_0 = 2x_0$. Placez une légende pour différencier les deux courbes (commande plt.legend())
5. Dans une autre fenêtre, tracez la trajectoire quand $\omega_y = 2\omega_x$ avec $y_0 = x_0$. Par dessus la précédente, tracez la trajectoire obtenue pour $\omega_y = 3\omega_x$. Les courbes que vous observez sont dites de Lissajous.

Rappel : pour tracer une courbe, on utilise plt.plot(x,y) et plt.show() où x et y sont vecteurs.

1.2 En 3D: piège de Penning

La libraries à importer, en plus des précédentes, est la suivante

```
from mpl_toolkits.mplot3d import Axes3D
```

Le piège de Penning est un dispositif permettant de stocker des particules chargées, grâce à la combinaison d'un champ magnétique uniforme et d'un champ électrique quadrupolaire. La géométrie est celle de la figure ci-après, les deux électrodes supérieure et inférieure en forme de coupelle sont portées au potentiel V_0, tandis que l'électrode latérale torique est au potentiel $-V_0$. On montre que le mouvement d'un ion dans ce piège est décrit par les équations suivantes :

$$\begin{aligned} x(t) &= r_1 \sin(\omega_+ t) - r_2 \sin(\omega_- t) \\ y(t) &= r_1 \cos(\omega_+ t) - r_2 \cos(\omega_- t) \\ z(t) &= z_0 \cos(\omega_z t) \end{aligned}$$

où, en notant B le champ magnétique, q et m la charge et la masse de l'ion, et $\omega_c = qB/m$ la pulsation "cyclotron", $\omega_z = \frac{1}{z_0}\sqrt{\frac{qV_0}{m}}$, $\omega_+ = \frac{1}{2}\left(\omega_c + \sqrt{\omega_c^2 - 2\omega_z^2}\right)$, et $\omega_- = \frac{1}{2}\left(\omega_c - \sqrt{\omega_c^2 - 2\omega_z^2}\right)$. Le but de l'exercice est de tracer la trajectoire d'un ion dans le piège, et de vérifier qu'il y reste confiné.

1. Calculer les constantes ω_z, ω_c, ω_+, et ω_-. On prendra les valeurs numériques suivantes : $B = 6$ T, $V_0 = 5$ V, $q = 1.6 \cdot 10^{-19}$ C, $m = 1.6 \cdot 10^{-27}$ kg, $z_0 = 1$ mm.
2. Créer un tableau pour le temps t, allant de 0 à $100\pi/\omega_z$ par pas de $\pi/(100\omega_z)$.
3. Calculer $x(t)$, $y(t)$ et $z(t)$. On prendra $r_2 = 10$ cm et $r_1 = 1$ mm.

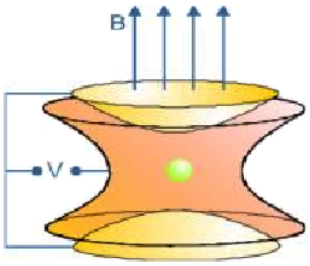

Figure 1: Géométrie du piège de Penning.

4. Représenter la trajectoire de l'ion dans l'espace (x, y, z). Il s'agit d'un graphe en perspective : pour tracer la figure, utiliser les commandes suivantes :

```
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(x, y, z, label='trajectoire')
```

2 Dessiner une surface

Les "libraries" à importer sont les mêmes que pour l'exercice précédent (piège de Penning). Le but est de dessiner des surface $(x, y, z = f(x, y))$. Les commandes sont les suivantes

```
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, linewidth=0, antialiased=False)
```

avec X, Y, Z des matrices.

1. Pour définir X, Y on utilise la fonction "np.meshgrid". Taper les lignes suivantes et observer les matrices X et Y : que représentent-elles ?

   ```
   [X, Y]=np.meshgrid([15, 5, 8], [0 , 1, 2, 3])
   print(X)
   print(Y)
   ```

   Taper ensuite

   ```
   Z=X*Y
   print(Z)
   ```

2. Programmer le calcul de la matrice Z en utilisant des boucles for. On appellera Z_1 le résultat. Comparer Z_1 et Z.

   Rappels :

   - `s=np.shape(X)` permet de calculer la taille d'une matrice X et de placer le résultat dans un vecteur s.
   - `Z1=zeros([nl,nc])` crée une matrice nulle de taille nc (nb de lignes) par nl (nb de colonnes)

3. En imaginant que $x \in [-5, 5]$ et $y \in [-5, 5]$, dessiner les surfaces définies par les équations suivantes. On prendra un pas de 0.25 sur x et sur y.

   - $z = f(x, y) = \exp(-[x^2 + y^2])$
   - $z = f(x, y) = \exp(y)$
   - $z = f(x, y) = \frac{\cos(x^2+y^2/4))}{x^2+y^2+1}$