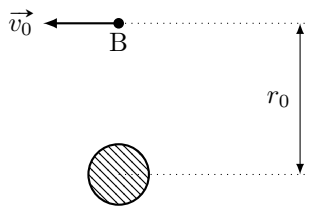
TP n°17

Mécanique : étude numérique de la trajectoire d'un satellite

MPSI 2 – 2024/2025

1 Objectif



On s'intéresse à la situation suivante : un satellite à une vitesse v_0 et est à distance r_0 du centre de la Terre. On cherche à établir la nature de la trajectoire du satellite en fonction de la valeur de v_0.
Comme nous le verrons, la trajectoire circulaire correspond à :

$$v_{\text{cercle}} = \sqrt{\frac{\mathcal{G}M_{\text{T}}}{r_0}}$$

Notons $\alpha = v_0/v_{\text{cercle}}$. C'est désormais notre paramètre ajustable.

2 L'outil `odeint`

Nous avons déjà vu en TP comment obtenir numériquement une solution approchée d'un équation différentielle par la méthode d'Euler. Cette méthode présente des défauts numériques ne permettant pas la résolution de notre problème. Nous allons utiliser le module Python appelé `scipy.integrate` contient une fonction, appelée `odeint` (pour *ordinary differential equation integration*), qui permet de résoudre directement l'équation sans avoir à programmer soi-même la méthode de résolution.

La résolution est plus précise qu'avec la méthode d'Euler car `odeint` utilise des schémas de résolution un peu plus sophistiqués que le schéma d'Euler explicite (même si les principes de base sont les mêmes).

2.1 Équation d'ordre 1

Cherchons à résoudre l'équation :

$$\frac{\mathrm{d}u_C}{\mathrm{d}t} + \frac{u_C}{3} = \frac{5}{3}$$

sur l'intervalle de temps de 0 s à 5 s, avec la condition initiale $u_C(0) = 0$. La syntaxe de la fonction `odeint` est la suivante :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate # importation du module necessaire a la resolution

def F(Y,t) : # definition de la fonction qui caracterise l'equation differentielle
 return -Y/3+5/3

dt = 0.001 # choix du pas temporel

T = np.arange(0, 5, dt) # creation de la subdivision temporelle
Y0 = 0 # condition initiale

Y = integrate.odeint(F, Y0, T) # resolution de l'equation differentielle :

plt.plot(T,Y) # trace de la solution
plt.show()
```

Exécuter ce code. La solution vous paraît-elle correcte ?

2.2 Cas d'une équation différentielle d'ordre 2

En mécanique, on rencontre le plus souvent des équations différentielles d'ordre 2. Considérons l'équation différentielle suivante (période propre de 1 s, facteur de qualité de 10) :

$$\ddot{y} + 0{,}31416\dot{y} + 39{,}4786y = 0$$

que l'on veut résoudre sur l'intervalle $[0\text{ s}; 15\text{ s}]$, avec $\dot{y}(0) = 0$ et $y(0) = 10$. En introduisant la grandeur $v = \dot{y}$, on a :

$$\dot{v} = -0{,}31416v - 39{,}4786y$$

Ainsi, si on note A le vecteur (y, v), alors :

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \begin{bmatrix} \dot{y} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -0{,}31416v - 39{,}4786y \end{bmatrix} = \begin{bmatrix} A[1] \\ -0{,}31416 * A[1] - 39{,}4786 * A[0] \end{bmatrix}$$

L'équation du second ordre est transformée en un système d'équations du premier ordre. La résolution avec `odeint` se passe alors exactement comme à l'ordre 1 sauf que l'inconnue est maintenant un vecteur.

```
def F(A,t) : # definition de la fonction qui caracterise l'equation differentielle
 return np.array([A[1],-0.31416*A[1]-39.4786*A[0]])

dt = 0.001 # choix du pas temporel

T = np.arange(0, 15, dt) # creation de la subdivision temporelle
A0 = np.array([10,0])

A = integrate.odeint(F, A0, T) # resolution de l'equation differentielle :

plt.plot(T,A) # trace de la solution
plt.show()
```

Exécuter ce code. La solution vous paraît-elle correcte ?

On remarque que Python renvoie deux courbes (l'une représentant la position y en fonction du temps et l'autre représentant la vitesse v en fonction du temps). En effet, le résultat A renvoyé est ici un tableau de deux colonnes, la première contenant toutes les valeurs de y et la deuxième les valeurs de v. Si on ne veut tracer que la position en fonction du temps, on écrira donc :

```
plt.plot(T, A[ :,0])
```

3 Revenons à nos moutons

1. Montrer que la constante des aires est donnée par :

$$C = \alpha\sqrt{r_0 \mathcal{G} M_T}$$

2. Par application du PFD, montrer que :

$$\ddot{r} = \frac{C^2}{r^3} - \frac{\mathcal{G}M_T}{r^2}$$

On peut montrer qu'en prenant $\tau = \sqrt{\dfrac{r_0^3}{\mathcal{G}M_T}}$, on a :

$$\frac{\mathrm{d}^2 r_*}{\mathrm{d}t_*^2} = \frac{\alpha^2}{r_*^3} - \frac{1}{r_*^2}$$

avec $r_* = r/r_0$ et $t_* = t/\tau$. On cherchera donc à résoudre ($r_p = \dot{r}$) :

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \begin{bmatrix} \dot{r} \\ \dot{r_p} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r_p \\ \alpha^2/r^3 - 1/r^2 \\ \dfrac{\alpha}{r^2} \end{bmatrix}$$

Résoudre l'équation. Choisir $\alpha = 1$ pour commencer. Représenter $r(t)$, $\dot{r}(t)$ et $\theta(t)$. Représenter ensuite la trajectoire. Faire varier α et observer.

Pour faire une représentation polaire :

```
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
ax.plot(theta, r)
ax.set_rmax(6)
ax.grid(True)
plt.show()
```

Exprimer l'énergie potentielle effective et l'énergie mécanique en particulier en fonction de α ; commenter.