Université de Montpellier **HLMA310**
2018/2019 **Joseph Salmon**

TP N° 8 : Animation avec `matplotlib`

Objectifs du TP : Créer des films, visualiser des informations temporelles sous forme d'animation

- Animation : prise en main -

Commencer par lancer l'exemple de l'aide de `matplotlib` :

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

plt.rcParams.update({'figure.max_open_warning': 25})

# commande "magique" pour améliorer la visualiation
%matplotlib notebook

fig1 = plt.figure()


def f(x, y):
    return np.sin(x) + np.cos(y)

x = np.linspace(0, 2 * np.pi, 120)
y = np.linspace(0, 2 * np.pi, 100).reshape(-1, 1)
ims = []
for i in range(60):
    x += np.pi / 15.
    y += np.pi / 20.
    im = plt.imshow(f(x, y), animated=True)
    ims.append([im])

ani = animation.ArtistAnimation(fig1, ims, interval=50, blit=True,
                                repeat_delay=1000)
plt.show()
```

1) Quel est le type de la variable `ims` ? Modifier les options `interval` et `repeat_delay` pour comprendre leur influence.
2) Utiliser la fonction `HTML`, obtenue de la manière suivante

```
from IPython.display import HTML
```

pour afficher un lecteur (qui permet notamment de faire une pause dans la vidéo) à l'intérieur de votre notebook.

- Animation d'un pendule amorti -

Petit rappelle de physique : on peut décrire l'évolution d'un pendule avec friction par l'équation différentielle ordinaire de second ordre (https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-832-underactuated-robotics-spring-2009/readings/MIT6_832s09_read_ch02.pdf) :

$$\forall t \geqslant 0, \quad \ddot{\theta}(t) + \frac{b}{m}\dot{\theta}(t) + \frac{g}{\ell}\sin(\theta(t)) = 0 \tag{1}$$

avec

— b : coefficient de frottement

— g : accélération de la pesanteur

— m : masse du pendule

Dans la suite et pour simplifier, on prend une masse unité ($m = 1$) et une longueur unité ($\ell = 1$). En introduisant la vitesse angulaire $\omega(t) := \dot{\theta}(t)$, on obtient le système différentiel équivalent :

$$\dot{\theta}(t) = \omega(t) \tag{2}$$

$$\dot{\omega}(t) = -b\omega(t) - g\sin(\theta(t)) \tag{3}$$

ou encore en définissant $y(t) = \begin{pmatrix}\theta(t)\\ \omega(t)\end{pmatrix}$, $\dot{y}(t) = \begin{pmatrix}\omega(t)\\ -b\omega(t) - g\sin(\theta(t))\end{pmatrix}$.

3) Lancer la suite d'instruction suivante :

```
def pend(y, t, b, g):
    theta, omega = y
    dydt = [omega, -b * omega - g * np.sin(theta)]
    return dydt

b0 = 0.5
g = 9.8
y0 = [np.pi - 0.1, 0.0] # initialisation
t = np.linspace(0, 20, 301) # discretisation du temps continu

from scipy.integrate import odeint
# Intègre un système d'équations différentielles ordinaires
sol0 = odeint(pend, y0, t, args=(b0, g))

fig_angle_vitesse, axes = plt.subplots(1, 1, figsize=(7, 6))
axes[0].plot(t, sol0[:, 0], 'b', label='$\\theta(t)$')
plt.legend(loc='best')
plt.xlabel('t')
plt.show()
```

et vérifier l'impact des paramètres b_0 et y_0 sur le pendule.

4) Tester l'exemple suivant :

```
fig = plt.figure(figsize=(5, 5), facecolor='w')
ax = fig.add_subplot(1, 1, 1)
plt.rcParams['font.size'] = 15

lns = []
for i in range(len(sol)):
    ln, = ax.plot([0, np.sin(sol0[i, 0])], [0, -np.cos(sol[i, 0])],
                  color='k', lw=2)
    tm = ax.text(-1, 0.9, 'time = %.1fs' % t[i])
    lns.append([ln, tm])
ax.set_aspect('equal', 'datalim')
ax.grid()
ani = animation.ArtistAnimation(fig, lns, interval=50)
```

Partez de cet exemple pour créer l'animation suivante : on mettra côte à côte l'évolution de trois pendules (on veillera à rajouter un point au bout du pendule), afin de visualiser la différence de comportement selon que le coefficient de frottement b est positif, négatif ou nul.