

```

# -*- coding: utf-8 -*-
"""
Created on Sat Aug 27 14:58:46 2022

@author: fabrice
"""

##    double pendule avec animation deux pendules.py

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
import matplotlib.animation as animation

#pour Les graphiques
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, autoscale_on=False, xlim=(-2.2, 2.2), ylim=(-2.2, 2.2))
ax.set_aspect('equal')
ax.grid()

# accélération de La gravité
#Les longueurs des deux pendules et Les masses sont identiquement égales
#à L'unité
g = 9.8

#conditions initiales pendule 1
départ = 120
u110 = np.radians(départ) #u10=theta10: angle de lâcher initial
u120 = 0 #vitesse de lâcher initial theta1
u130 = np.radians(-10) #u20=theta20: angle de lâcher initial
u140 = 0 #vitesse de lâcher initial theta2

u10 = u110 , u120 , u130 , u140 #vecteur des conditions initiales

#conditions initiales pendule 2: écart est Le décalage de L'angle initial theta1
écart = 1
u210 = np.radians(départ + écart) #u10=theta10: angle de lâcher initial
u220 = 0 #vitesse de lâcher initial theta1
u230 = np.radians(-10) #u20=theta20: angle de lâcher initial
u240 = 0 #vitesse de lâcher initial theta2

u20 = u210 , u220 , u230 , u240 #vecteur des conditions initiales

#vecteur final des conditions initiales pour Les deux pendules
vect_init = u10 , u20

#intervalle de temps pour L'intégration
t0 , tf = 0 , 50

#nombre de valeurs de t pour L'intégration
tmax = 500

#fonction qui prend Le vecteur u et en calcule La dérivée du
#ce qui nous interesse c'est u1 et u3, soit Les angles theta1 et theta 2
#équations différentielles selon wikipédia

def deriv(t , u):
    u1 , u2 , u3 , u4 = u
    du1 = u2
    du3 = u4

```

```

v12 = np.cos(u1 - u3)
w12 = np.sin(u1 - u3)

den = 2-v12**2

du4 = ((u4**2)*w12*v12 + 2*g*np.sin(u1)*v12 + 2*(u2**2)*w12 - 2*g*np.sin(u3)) / den

du2 = (-g*np.sin(u3) + (u2**2)*w12 -du4 )/ v12

return du1 , du2 , du3 , du4

#boucle permettant de calculer Les fichiers des angles theta1 et theta2 pour Les
#deux pendules j=0 et j=1

j = 0 #compteur pour La boucle

#déclaration des fichiers à 2D pour Les angles
theta1 , theta2 = np.empty((2 , tmax)) , np.empty((2 , tmax))

#boucle commence...pendule1=conditions initiales1 et pendule2=conditions initiales2
for u0 in vect_init:

    #soln vecteur contenant Les solutions
    soln = solve_ivp(deriv , (t0 , tf) , u0 , dense_output = True)

    #on extrait Les solutions qui nous intéressent theta1, theta2, avec La base de temps t
    t = np.linspace(t0 , tf , tmax)
    sol = soln.sol(t)
    t1 , t2 = sol[0] , sol[2]

    #là c'est une boucle un peu pénible qui copie Les solutions extraites dans Les
    #fichiers des angles
    for i in range(tmax):
        theta1[j][i] = t1[i]
        theta2[j][i] = t2[i]

    j+=1

#caclul des coordonnées cartésiennes de L'articulation P1 et de L'extrémité P2 pendule 1
x11 , y11 = np.sin(theta1[0]) , -np.cos(theta1[0])
x12 , y12 = np.sin(theta2[0]) + x11 , -np.cos(theta2[0]) + y11

#caclul des coordonnées cartésiennes de L'articulation P1 et de L'extrémité P2 pendule 2
x21 , y21 = np.sin(theta1[1]) , -np.cos(theta1[1])
x22 , y22 = np.sin(theta2[1]) + x21 , -np.cos(theta2[1]) + y21

#partie pour L'animation avec déclaration des graphiques vides
#prépare le graphique pendule 1
line1, = ax.plot([], [], 'o-', lw=2 , label="Pendule 1,  $\theta_{10}="+str(départ)+"^{\circ}"$ )
#prépare le graphique pendule 2
line2, = ax.plot([], [], 'o-', lw=2 , label="Pendule 2,  $\theta_{10}="+str(départ+écart)+"^{\circ}"$ )

#pour L'animation
def init():
    line1.set_data([], [])
    line2.set_data([], [])
    return line1, line2

def animate(i):

```

```

coordx1 = [0, x11[i], x12[i]] #coordonnées x des trois points dessinés
        #fixation, P1 et P2 pendule 1
coordy1 = [0, y11[i], y12[i]] #coordonnées y des trois points dessinés
        #fixation, P1 et P2 pendule 1
coordx2 = [0, x21[i], x22[i]] #coordonnées x des trois points dessinés
        #fixation, P1 et P2 pendule 2
coordy2 = [0, y21[i], y22[i]] #coordonnées y des trois points dessinés
        #fixation, P1 et P2 pendule 2

line1.set_data(coordx1, coordy1)
line2.set_data(coordx2, coordy2)

return line1, line2

#commande L'animation
#régler la vitesse plus c'est grand plus c'est lent !
vitesse = 200
ani = animation.FuncAnimation(fig, animate,
                             interval=vitesse, blit=True, init_func=init)

plt.title("""Deux pendules doubles !! pour deux angles
        initiaux de lâcher  $\theta_{10}$  proches
        toutes autres grandeurs égales""")
plt.legend()

#sauvegarde fichier(optionnel)
ani.save('double_pendule_double.mp4', fps=6, extra_args=['-vcodec', 'libx264'])

plt.show()

```