```python
# -*- coding: utf-8 -*-
"""
Created on Sat Aug 27 09:18:49 2022

@author: fabrice
"""

# double pendule visualisation chaos sur x de P1

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

# pour le graphique
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot()

from matplotlib import rc
font_properties = {'size' : 14}
rc('font' , **font_properties)

linestyles = [{'ls': '-'}, {'ls': '--'}, {'ls': ':'}, {'ls': '-.'},
              {'dashes': [2, 4, 2, 4, 8, 4]}]


g = 9.8  # accélération de la gravité
# les longueurs des deux pendules et les masses sont identiquement égales
# à l'unité

# conditions initiales
initheta10 = 120
u10 = np.radians(initheta10)  # u10=theta10: angle de lâcher initial
u20 = 0  # vitesse de lâcher initial theta1
u30 = np.radians(-10)  # u20=theta20: angle de lâcher initial
u40 = 0  # vitesse de lâcher initial theta2

écartheta10 = 0.5  # écart à l'angle initial
écart = np.radians(écartheta10)  # décalage sur u10 pour voir le chaos

u0 = u10, u20, u30, u40  # 1er vecteur des conditions initiales
u00 = u10 + écart, u20, u30, u40  # 2ème vecteur des conditions initiales

vect_init = u0, u00  # vecteur des conditions initiales pour boucle ci-dessous

# intervalle de temps pour l'intégration
t0, tf = 0, 20

# 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 qui fait calculer la simulation pour deux angles initiaux theta1 proches
i = 0  # compteur hyper utile pour ax.plot

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, avec la base de temps t
    t = np.linspace(t0, tf, 500)
    sol = soln.sol(t)
    theta1 = sol[0]

    # caclul des coordonnées cartésiennes de l'articulation P1 et de l'extrémité P2
    x1 = np.sin(theta1)

    ax.plot(t, x1, label="θ₁₀ =  " + str(round((np.degrees(vect_init[i][0])) , 1)) + '
            c='k', **linestyles[i])

    i += 1

# fignolage du graphique
plt.title("""Pendule double, coordonnées x de l'articulation pour deux angles
          initiaux de lâcher θ₁₀ proches
          toutes autres grandeurs égales""")
ax.set_xlabel("temps (s)")
ax.set_ylabel("x (m)")
plt.legend(loc='upper right')
plt.grid()
plt.show()
```