```python
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 19 07:08:56 2022

@author: fabrice
"""

#système différentiel de lorentz résolution et représentation graphique en 3D
#et visualisation du "papillon de lorentz "

import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

from matplotlib import rc
font_properties = {'size' : 10}
rc('font' , **font_properties)

##déclaration graphique
fig = plt.figure()

#constantes k1 = sigma k2 = rho k3 =béta
#on reprend les valeurs données par Lorentz (wiki)
k1 , k2 , k3 = 10 , 28 , 8/3

#conditions initiales (y1=x, y2=y , y3=z) choix peu conséquent
y10 , y20 , y30 = 0.1 , 0.1 , 0.1

#temps maximal
tmax = 200

#définition des variables fonction de t et écriture des trois équations couplées
def dydt(t , y , k1 , k2, k3):
    y1 , y2 , y3 = y
    dy1dt = k1*(y2 - y1)
    dy2dt = k2*y1 - y2 - y1*y3
    dy3dt = y1*y2 - k3*y3
    return dy1dt , dy2dt , dy3dt

#intégration vecteur des conditions initiales
y0 = y10 , y20 , y30

#résolution
solution = solve_ivp(dydt , (0 , tmax) , y0 , dense_output = True, args=(k1,k2,k3))

#fichier avec les valeurs du temps et extraction des réponses
#la visibilité du graphique peut être améliorée en jouant sur les valeurs de tmax
#et le nombre de valeurs du fichier t
t = np.linspace(0 , tmax , 2000)
x , y , z = solution.sol(t)

#graphique
ax = fig.add_subplot(projection = '3d')
```

```
ax.plot(x , y , z , color = 'k' , lw=0.5 )
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

plt.title('Papillon de Lorentz, solution des équations de Lorentz')
plt.tight_layout()
plt.grid()
plt.show()
```