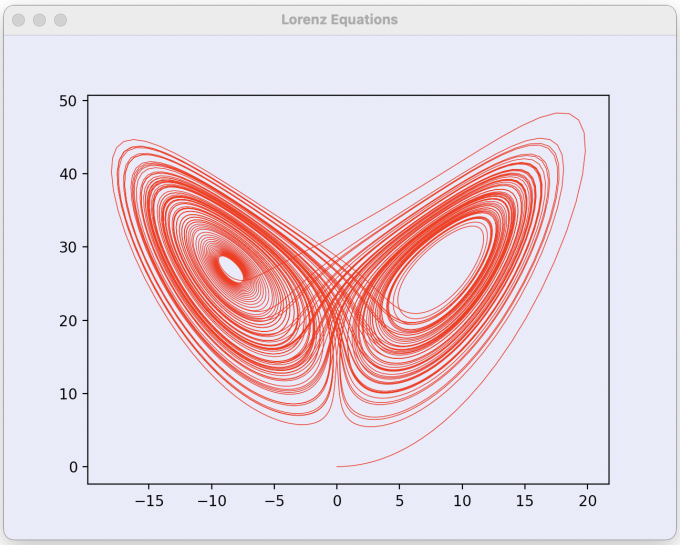
Python 22-23 for dummies : problème 6
Un modèle-jouet :
Les équations de Lorenz !



Il s'agit de résoudre le système d'équations différentielles ordinaires donnés par :

$$\begin{cases} u'(t) &= 10v(t) - 10u(t) \\ v'(t) &= 28u(t) - v(t) - u(t)w(t) \\ w'(t) &= u(t)v(t) - \frac{8}{3}w(t) \end{cases}$$

Il s'agit d'un exemple de la famille des équations de Lorenz. En prenant comme conditions initiales $u(0) = w(0) = 0$ et $v(0) = 1$, on souhaite obtenir les trois composantes pour $0 \leq t \leq 100$

En 1963, Edward Lorenz (1917-2008), qui s'intéressait au problème de la convection dans l'atmosphère terrestre, simplifia drastiquement les équations de Navier-Stokes de la mécanique des fluides, réputées pour leur inextricable complexité. Le modèle atmosphérique de Lorenz est ce que les physiciens appellent un modèle-jouet : bien qu'il n'ait probablement pas grand-chose à voir avec la réalité, Lorenz ne tarda pas à réaliser qu'il s'agissait d'un modèle mathématique très intéressant. Les équations de Lorenz ne font intervenir que trois nombres u, v et w, de sorte que chaque point (u, v, w) de l'espace symbolise un état de l'atmosphère et l'évolution consiste à suivre un champ de vecteurs. Comprendre l'évolution du temps qu'il fait dans l'atmosphère virtuelle de Lorenz revient à suivre une trajectoire de ce champ de vecteurs. N'oublions pas qu'il s'agit d'un modèle-jouet et que l'objectif est d'essayer de comprendre les grandes lignes d'un comportement complexe. Si l'on considère deux atmosphères presque identiques, donc représentées par les centres de deux petites boules extrêmement proches, rapidement les deux évolutions se séparent de manière significative : les deux atmosphères deviennent complètement différentes. Lorenz a pu constater sur son modèle la dépendance sensible aux conditions initiales, le chaos. Mais plus intéressant, partant d'un grand nombre d'atmosphères virtuelles, bien qu'un peu folles et bien peu prévisibles, les trajectoires semblent toutes s'accumuler sur un même objet en forme de papillon, popularisé sous le nom d'attracteur de Lorenz, un attracteur bien étrange...[1]

Pour intégrer ces équations, on se propose d'implémenter la méthode de Runge-Kutta :

$$\begin{aligned} \mathbf{U}_{i+1} &= \mathbf{U}_i + \tfrac{h}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K_4}) \\ \mathbf{K}_1 &= f(T_i, \mathbf{U}_i) \\ \mathbf{K}_2 &= f(T_i + \tfrac{h}{2}, \mathbf{U}_i + \tfrac{h}{2}\mathbf{K}_1) \\ \mathbf{K}_3 &= f(T_i + \tfrac{h}{2}, \mathbf{U}_i + \tfrac{h}{2}\mathbf{K}_2) \\ \mathbf{K}_4 &= f(T_i + h, \mathbf{U}_i + h\mathbf{K}_3) \end{aligned}$$

[1] https://www.chaos-math.org/en/chaos-vii-strange-attractors.html

Plus précisément, on vous demande de :

1. Implémenter la fonction suivante

   `T,U = lorenz(Tstart,Tend,Ustart,n)`.

   - Les arguments `Tstart` et `Tend` définissent l'intervalle d'intégration temporelle.
   - L'argument `Ustart` définit les 3 valeurs initiales pour le vecteur (u, v, w) dans une liste ou un tableau de longueur 3.
   - L'argument `n` contient le nombre des pas égaux à effectuer avec la méthode de Runge-Kutta.
   - Votre fonction fournira tout d'abord les $n + 1$ temps T_i et les valeurs discrète (U_i, V_i, W_i). Le tableau `T` sera un tableau unidimensionnel `numpy.ndarray` de longueur $n + 1$, tandis que le tableau `U` sera un tableau bidimensionnel `numpy.array` de taille $(n + 1) \times 3$.

2. Un programme `lorenzTest.py` vous est fourni pour tester votre fonction.

```
from numpy import *
from matplotlib import pyplot as plt

plt.figure("Lorenz Equations")
Xstart = 0; Xend = 100; Ustart = [0,1,0]; n = 10000

X,U = lorenz(Xstart,Xend,Ustart,n)
plt.plot(U[:,0],U[:,2],'-r',linewidth=0.5)
plt.show()
```

   Evidemment, sur le serveur, nous utiliserons d'autres arguments que celles de ce programme : il est donc nécessaire d'écrire le code de manière tout-à-fait générale :-)

3. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera soumise via le site web du cours.

4. Attention : il ne faut pas recopier le programme de test `main` dans votre soumission : uniquement les deux fonctions que vous avez écrites. Vérifier bien que votre programme fonctionne correctement sur le serveur et pas uniquement sur votre ordinateur : aucun recours ne sera valable si le devoir n'est pas exécuté correctement sur le serveur. Pour rappel, toutes vos soumissions seront systématiquement analysées par un logiciel anti-plagiat. Faites vraiment votre programme seul... en vous inspirant uniquement des programmes fournis par l'enseignant :-)