TP 5

Résolution numérique d'équations différentielles

1 La méthode d'Euler

La méthode d'Euler est une méthode de résolution approchée d'équations différentielles, basée sur un développement limité d'ordre 1 (on l'appelle aussi méthode de Runge-Kutta d'ordre 1).

Plus précisément, on considère l'équation

$$y'(t) = f(t, y(t)),$$

où f est une fonction.

La méthode d'Euler est basée sur l'approximation $y'(t) = \frac{y(t+h)-y(t)}{h}$, et on a donc

$$y(t+h) = y(t) + hf(t, y(t)).$$

Ainsi, à chaque point, on approxime $y(t)$ par sa tangente.

Implémenter alors la méthode d'Euler. On testera les résultats pour l'équation

$$y'(t) = \cos(t) - y \text{ et } y(0) = 1.$$

2 La méthode d'Euler pour les systèmes

On considère maintenant le système d'équations différentielles

$$\begin{cases} x'(t) = f(t, x(t), y(t)) \\ y'(t) = g(t, x(t), y(t)) \end{cases}.$$

On peut alors à nouveau implémenter la méthode d'Euler en utilisant la même approximation que précédemment.

Tester cette méthode de résolution pour le système de Lokta-Volterra

$$\begin{cases} x'(t) = x(t)(3 - 2y(t)) \\ y'(t) = y(t)(-4 + x(t)) \end{cases}.$$

En utilisant un système, on peut aussi trouver des approximations pour les équations différentielles d'ordre 2 : pour l'équation $y'' = f(t, y(t), y'(t))$, on se ramènera au système, en posant $x = y'$:

$$\begin{cases} x'(t) = f(t, y(t), x(t)) \\ y'(t) = x(t) \end{cases}.$$

Implémenter cette méthode, et la tester pour l'équation du pendule simple

$$y'' + \sin(y) = 0.$$

On comparera cette solution avec celle de $y'' + y = 0$, utilisée en physique.

3 Méthode de Runge-Kutta d'ordre 4

Pour la méthode de Runge-Kutta d'ordre 4 (souvent appelée RK4) pour résoudre $y'(t) = f(t, y(t))$, on pousse le développement limité un peu plus loin :

$$\begin{aligned} K_1 &= hf(t, y(t)) \\ K_2 &= hf\left(t + \frac{1}{2}h, \frac{1}{2}K_1 + y(t)\right) \\ K_3 &= hf\left(t + \frac{1}{2}h, \frac{1}{2}K_2 + y(t)\right) \\ K_4 &= hf(t + h, K_3 + y(t)) \\ y(t + h) &= y(t) + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right) \end{aligned}$$

Implémenter cette méthode, et tester sur le premier exemple.

4 La fonction `odeint`

Dans le module `scipy.integrate`, il existe une fonction `odeint`, qui permet de résoudre numériquement des équations différentielles.

On peut l'utiliser pour résoudre l'équation $y'(t) = f(y(t)t)$ (attention à l'inversion des variables), avec éventuellement des valeurs vectorielles, avec la syntaxe `odeint(f,y0,t)` où `f` est une fonction `y0` la condition initiale et `t` une liste d'abscisses pour lesquelles on veut résoudre l'équation.

Tester en comparer avec les méthodes précédentes.

5 Attracteur de Lorenz

On considère les équations de Lorenz, provenant de la mécanique des fluides :

$$\begin{cases} x'(t) = \sigma(y(t) - x(t)) \\ y'(t) = \rho x(t) - y(t) - x(t)z(t) \\ z'(t) = x(t)y(t) - \beta z(t) \end{cases}$$

Résoudre numériquement cette équation, en prenant $\sigma = 10$, $\beta = 2.66$ et $\rho = 30$.

Pour représenter les solutions, on pourra utiliser la ligne

```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.gca(projection='3d')
```

et en utilisant dans la suite `ax.plot` au lieu de `plt.plot`.

On considèrera plusieurs conditions initiales.