

TP3

Equations différentielles

25 octobre 2017

Dans ce TP, on présente quelques méthodes pour intégrer numériquement sur un intervalle $t \in [0, T]$, une équation différentielle ordinaire - dite EDO - c'est-à-dire de la forme

$$u'(t) = f(t, u(t)), \quad u(0) = u_0 \quad (1)$$

La fonction f , qui prend en argument un couple de réels et qui renvoie un réel, ainsi que la valeur initiale u_0 sont donnés; la fonction u , qui prend un réel en argument et qui renvoie un réel, est inconnue; c'est la solution cherchée.

1. Exemple : considérons l'EDO très simple

$$u'(t) = -u(t), \quad u(0) = 1.0$$

Le problème est ici de déterminer la solution u .

- (a) Que vaut la fonction f dans cet exemple
 - (b) Calculer à la main la solution u de l'EDO ci-dessus, et la dessiner sur papier; on prendra comme intervalle $t \in [0, 2]$.
 - (c) Sur ordinateur, représenter graphiquement u à l'aide du module `matplotlib`; on créera le vecteur d'abscisses au moyen de `numpy.linspace` et on sauvera la figure au format `png` dans le répertoire courant.
2. Calcul approché de u au moyen de la **méthode d'Euler**.

On divise l'intervalle $[0, T]$ en n parties égales, puis on pose $h = T/n$, ce qui fournit une subdivision $t_0 = 0, t_1 = h, t_2 = 2h, \dots, t_n = T$. A partir de là veut calculer, approximativement, la valeur de u aux points t_k de la subdivision. Voici l'idée :

Supposons que l'on connaisse, pour un indice k , une approximation u_k de la valeur exacte $u(t_k)$; comment alors calculer une approximation u_{k+1} de la valeur exacte $u(t_{k+1})$? Réponse :

On écrit $u'(t_k) \approx \frac{u(t_{k+1}) - u(t_k)}{h}$, approximation d'autant meilleure que h est petit;

on a donc $\frac{u(t_{k+1}) - u(t_k)}{h} \approx u'(t_k) = f(t_k, u(t_k))$, puis $u(t_{k+1}) \approx u(t_k) + hf(t_k, u(t_k))$

et enfin $u(t_{k+1}) \approx u_k + hf(t_k, u_k)$. On a donc trouvé une approximation u_{k+1} de la valeur exacte $u(t_{k+1})$, à savoir

$$u_{k+1} = u_k + hf(t_k, u_k) \quad (2)$$

Cette formule permet, par récurrence, de calculer une suite d'approximations u_k , à condition que l'on connaisse une première approximation u_0 de $u(t_0)$; mais pour u_0 on peut bien sûr prendre la valeur de la condition initiale donnée par l'EDO.

- (a) Calculer la suite u_k , définie par la méthode d'Euler, pour l'exemple ci-dessus; on prendra $T = 2.0$ et $n = 10$.
- (b) Représenter sur le même graphique la solution exacte u et les points (t_k, u_k) ; faire le travail sur papier et numériquement à l'aide de `matplotlib`.

3. Ecrire une fonction python `euler` qui prend en arguments une fonction f de deux variables scalaires t, u , une valeur initiale u_0 , un réel T , un entier n , et qui renvoie deux `numpy arrays`, tt contenant les valeurs t_k et uu contenant les valeurs u_k , obtenues par la méthode d'Euler.
4. (a) Appliquer la fonction `euler` à l'exemple précédentci-dessus (on aura besoin d'écrire la fonction f de l'EDO - fonction de deux variables - dans une fonction python) et retrouver les résultats précédents.
 (b) Examiner avec les EDOs suivantes (attention, certaines équations peuvent être facilement intégrées à la main, d'autres non).

$$\begin{array}{ll} u'(t) = -u(t) + t, & u(0) = 1.0 \\ u'(t) = u^2(t), & u(0) = 1.0 \\ u'(t) = u^2(t) - t, & u(0) = 1.0 \end{array}$$

5. Nous montrons maintenant comment intégrer une EDO d'ordre supérieur. Prenons l'exemple de l'oscillateur harmonique (modélisation du ressort) qui est une EDO d'ordre 2

$$u''(t) + \omega^2 u(t) = 0, \quad u(0) = u_0, \quad u'(0) = v_0 \quad (3)$$

où ω, u_0, v_0 sont des scalaires donnés ; ω s'appelle la pulsation, u_0 la position initiale, v_0 la vitesse initiale. Posons ensuite $v = u'$ et écrivons l'EDO sous la forme

$$\begin{cases} u'(t) = v(t) \\ v'(t) = -\omega^2 u(t) \end{cases} \quad (4)$$

On pose $U = [u, v]$ et $U_0 = [u_0, v_0]$ de sorte que l'EDO s'écrit maintenant

$$U'(t) = F(t, U(t)), \quad U(0) = U_0 \quad (5)$$

avec $F(t, [u, v]) = [v, -\omega^2 u]$; F est donc une fonction de $\mathbb{R} \times \mathbb{R}^2$ dans \mathbb{R}^2 .

On a ainsi réécrit l'EDO (3) d'ordre 2 en l'EDO (5) d'ordre 1. Attention ! dans le premier cas, la fonction f va de $\mathbb{R} \times \mathbb{R}$ dans \mathbb{R} ; dans le deuxième cas, la fonction F va de $\mathbb{R} \times \mathbb{R}^2$ dans \mathbb{R}^2 . Il est maintenant facile de résoudre cette EDO numériquement, il suffit de lui appliquer la méthode d'Euler sous la forme $U_{k+1} = U_k + hf(t_k, U_k)$, avec U_0 qui est donnée. C'est une récurrence qui fournit la suite des points U_k de \mathbb{R}^2 , approximations des valeurs $U(t_k)$, où U est la solution exacte de l'EDO.

- (a) Calculer, à la main, la solution exacte de l'EDO (3).
- (b) Modifier la fonction `euler`, pour qu'elle prenne en arguments F , une fonction de deux variables $t \in \mathbb{R}, U \in \mathbb{R}^2, U_0 \in \mathbb{R}^2$, un réel T , un entier n , et qui renvoie deux `numpy arrays`, tt contenant les valeurs t_k et UU contenant les valeurs U_k , obtenues par la méthode d'Euler.
- (c) Résoudre numériquement l'EDO (3), avec $\omega = 1.0, T = 4\pi$ et les conditions initiales $u(0) = 1.0, u'(0) = 0.0$; on écrira la fonction F dans une fonction python ; on prendra différentes valeurs de n .
- (d) Faire trois graphiques :
 - i. position u en fonction du temps t
 - ii. vitesse v en fonction du temps t
 - iii. vitesse v en fonction de position u

Sur chaque graphique, représenter la solution exacte et la solution approchée donnée par la méthode d'Euler.