TP n°4 : équations différentielles, fonctions (suite)

PTSI Lycée Eiffel

17 novembre 2017

1 Équations différentielles

Le but de cette première partie est d'arriver à résoudre de façon approchée des équations différentielles du premier ordre, du type $f'(t)+a(t)f(t) = b(t)$, et à tracer à l'aide du module matplolib.pyplot une allure de la courbe représentative de la fonction f. Pour celà on va créer une fonction Python prenant comme paramètres les valeurs suivantes : la valeur ti correspondant au temps initial à partir duquel on veut résoudre l'équation, la valeur tf jusqu'à laquelle on va résoudre l'équation (autrement dit, on résout sur l'intervalle $[ti, tf]$), la valeur fi correspondant à la condition initiale $f(ti)$, ainsi qu'un entier n indiquant le nombre d'étapes de calcul qu'on va effectuer. La procédure que nous allons utiliser fonctionne en effet de la façon suivante :

- on crée une liste **abscisse** de valeurs temporelles comportant exactement $n + 1$ valeurs régulièrement espacées dont la première est ti et la dernière tf. Ainsi, si on résout sur $[0, 1]$ avec $n = 10$, la liste doit être $[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$.
- On crée ensuite une liste **ordonnee** qui ne contient initialement que la valeur fi. On va ensuite compléter de proche en proche la liste avec des valeurs approchées de $f(x_i)$, où les réels x_i sont ceux apparaissant dans la liste abscisse crée ci-dessus. Pour cela, on utilisera l'approximation $f'(x_i) \simeq \dfrac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$, avec par ailleurs par définition $f'(x_i) = b(x_i) - a(x_i)f(x_i)$. Autrement dit, on calculera $f(x_{i+1}) \simeq f(x_i) + h * (b(x_i) - a(x_i)f(x_i))$, où on a noté $h = x_{i+1} - x_i$ (valeur constante tout au long de l'exécution de l'algorithme).
- On trace ensuite la courbe obtenue à partir des deux listes abscisse et ordonnee.

1. Appliquer cette méthode pour résoudre l'équation $f' = f$ sur l'intervalle $[0, 5]$ avec $fi = 1$ et $n = 10$. Modifier ensuite la valeur de n pour l'augmenter, et comparer les courbes obtenues.
2. Tracer simultanément à vos courbes approchées la courbe de la vraie solution (que vous devez normalement reconnaitre).
3. Résoudre de façon approchée l'équation différentielle $f'+f = K$, où K est une constante qu'on pourra faire varier, avec la condition initiale $fi = 0$ (on choisira un intervalle de résolution adapté).

2 Amusons-nous un peu avec les nombres entiers

Toute cette partie du TP est consacrée à l'écriture d'algorithmes faisant intervenir les nombres entiers, et notamment la notion de nombre premier. Il existe de nombreuses façons d'écrire les différents programmes demandés, notamment celui permettant de savoir si un entier donné est premier ou non. N'hésitez pas, une fois que vous avez écrit des programmes qui tournent, à comparer avec ce qu'ont pu faire vos camarades, et à réfléchir à des améliorations pour optimiser la rapidité d'exécution de vos programmes. Mais prenez quand même d'abord le temps de bien tout faire par vous-même !

1. Programmer une fonction prenant comme paramètre un entier n et déterminant s'il est premier ou non (la fonction renverra donc un booléen, autrement dit une variable ne pouvant prendre comme valeur que **True** ou **False**).

2. Programmer une fonction prenant comme paramètre un entier n et affichant la liste de tous les nombres premiers inférieurs ou égaux à n.
3. Programmer une fonction prenant comme paramètres deux entiers n et p et déterminant leur pgcd à l'aide de l'algorithme d'Euclide (si vous ne savez pas comment fonctionne cet algorithme, on prendra le temps de l'expliquer !). On pourra écrire deux versions, l'une récursive, l'autre non.
4. Programmer une fonction prenant comme paramètre un entier n et déterminant son plus petit facteur premier.
5. Programmer une fonction prenant comme paramètre un entier n et calculant sa décomposition en facteurs premiers (sous la forme d'une liste de facteurs premiers, éventuellement répétés k fois si le facteur apparait élevé à la puissance k dans la décomposition de n).