Informatique en CPGE (2018-2019)
TP 10 : résolution numérique d'équations différentielles, méthode d'Euler

La méthode d'Euler permet de résoudre des équations différentielles du type $y' = f(x, y)$.

Exercice 1

Objectif : programmer la méthode d'Euler pour résoudre l'équation différentielle $y' = y$ sur l'intervalle $[0; 4]$ avec $y(0) = 1$.

1. Ecrire une fonction `euler` qui prend en arguments les valeurs extrêmes de l'intervalle a et b, la valeur initiale $y(0)$, le pas h et la fonction f qui sera définie dans le programme (ici par $f(x, y) = y$). La fonction `euler` construit la liste `liste_x` des abscisses x_k et la liste `liste_y` des ordonnées y_k (valeurs approchées de $y(x_k)$) en utilisant une boucle `while`, puis renvoie ces deux listes.

   Rappel du schéma : on calcule les approximations pour $k = 0, 1, 2, \ldots, n-1$ par :

   $$x_{k+1} = x_k + h \text{ et } y_{k+1} = y_k + hf(x_k, y_k)$$

   On initialise avec $y_0 = y(x_0) = y(a)$.
2. Tester le programme pour $h = 1$ avec l'instruction `print(euler(0,4,1,1,f))`. Le résultat doit être : `([0, 1, 2, 3, 4], [1, 2, 4, 8, 16])` puisque $y_k = (1 + h)^k = 2^k$.
3. La solution exacte est connue, $y(x) = \mathrm{e}^x$. Compléter le programme qui doit calculer puis afficher l'erreur $e = \max |y(x_k) - y_k|$ obtenue. Effectuer des test avec h prenant les valeurs 1, 0.5, 0.2, 0.1.
4. Déterminer en effectuant des tests le pas h afin que l'erreur maximale soit de l'ordre de 10^{-2}.
5. Tracer sur une même figure les courbes représentatives de la solution exacte et des solutions approchées pour h prenant les valeurs 1, 0.5, 0.2, 0.1.

Exercice 2

Programmer la résolution numérique avec la méthode d'Euler de l'équation différentielle $y' + y = cos(2x)$ avec $y(0) = 4$, sur l'intervalle $[0; 12]$.

Comparer graphiquement les solutions approchées obtenues pour différentes valeurs du pas h avec la solution exacte donnée par : $y(x) = 3.8\exp(-x) + 0.2\cos(2x) + 0.4\sin(2x)$

Exercice 3

Objectif : résoudre l'équation différentielle $y' = -y$ avec $y(0) = 1$ sur l'intervalle $[0; 30]$. La solution exacte est $y(x) = \mathrm{e}^{-x}$. Attention, si le pas n'est pas assez petit, on a un problème de stabilité.

1. Reprendre la fonction `euler` de l'exercice 1. Définir la fonction $f(x, y) = -y$ et tester le programme avec $h = 3$ puis $h = 2.5$ et constater qu'il y a un problème d'instabilité. On construira les courbes des solutions approchées avec Matplotlib.
2. Refaire le test avec $h = 1.5$ et construire la courbe. Déterminer une valeur approchée de h à partir de laquelle le schéma est stable.
3. Déterminer une valeur approchée de h à partir de laquelle l'erreur de discrétisation e est inférieure à 10^{-1}. (Rappel : $e = \max |y(x_k) - y_k|$).

Exercice 4

Objectif : résoudre l'équation différentielle du second ordre $y'' + y = 0$ pour $x \in [0; 10]$ avec les conditions initiales $y(0) = 0$ et $y'(0) = 1$.

On pose $Y = (y, y')$; l'équation $y'' + y = 0$ est équivalente à $(y, y')' = (y', y'') = (y', -y) = F(y, y')$, et on obtient $Y' = F(Y)$ avec la condition initiale $Y(0) = (0, 1)$.

La méthode d'Euler peut encore s'appliquer ici. On reprend les éléments du programme de l'exercice 1 avec quelques modifications puisque Y est un couple.

1. Modifier la définition de la fonction f qui prend en arguments x et Y et renvoie le couple $F(Y)$ défini ci-dessus.
2. La modification dans la définition de la fonction `euler` est la plus délicate. La liste `liste_y` est une liste de couples donc l'écriture `y=y+h*f(x,y)` ne convient pas. Il faut alors traduire proprement l'opération $Y = Y + hF(Y)$ qui permet de passer du couple (y_k, y'_k) au couple (y_{k+1}, y'_{k+1}).
3. L'appel de la fonction se fera avec $y0 = (0, 1)$ et $h = 0.01$.
4. Construire avec Matplotlib la représentation graphique de la solution approchée et celle de la solution exacte. Les ordonnées pour la solution approchée sont les premiers éléments de chaque couple dans la liste des y renvoyée par la fonction `euler`.

Exercice 5

Reprendre l'exercice précédent mais cette fois en utilisant les objets de type **array** de la bibliothèque **numpy**. On commence donc le programme avec l'instruction `from numpy import array`.

(La fonction **array** permet de convertir un objet de type **list** ou **tuple** en un objet de type **nd.array**).

Effectuer différents test sur les objets de type nd.array, par exemple :

```
a=array((3,4))
b=array((2,5))
print(a+b) # affiche [5 9]
print((a+b)[0]) # affiche 5
print(5*(a+b)) # affiche [25 45]
```

La définition de la fonction `euler` s'écrit comme dans l'exercice 1. On modifie la définition de la fonction f qui renvoie un objet de type **nd.array** et l'appel de la fonction `euler` :

```
def f(x,y):
    return array((y[1],-y[0]))

x, y = euler(0, 10, array((0,1)), 0.01, f)
```

Tester le programme en traçant avec Matplotlib la courbe représentant la solution approchée.

Exercice 6

Objectif : l'équation $\theta'' = -k_1 \sin\theta - k_2\theta'$ permet d'étudier le mouvement d'un pendule amorti et il donc est intéressant de pouvoir visualiser une approximation de la solution. (θ est fonction de t).

On pose $Y = (\theta, \theta')$; on a donc $Y' = (\theta', \theta'') = (\theta', -k_1 \sin\theta - k_2\theta') = F(Y)$ avec $F((a, b)) = (b, -k_1 \sin a - k_2 b)$.

Utiliser le programme de l'exercice 5 avec les modifications nécessaires pour pouvoir afficher les courbes représentant des solutions approchées de θ en fonction de t, et de θ' en fonction de θ (diagramme de phase), avec Matplotlib dans les deux cas suivant :

1. Pendule simple avec les conditions : $\theta_0 = \pi/6$, $\theta'_0 = 0$, $k_1 = 5$, $k_2 = 0$, $t \in [0; 20]$ et $h = 0.01$.
2. Pendule amorti avec les conditions : $\theta_0 = \pi/6$, $\theta'_0 = 0$, $k_1 = 5$, $k_2 = 0.5$, $t \in [0; 20]$ et $h = 0.01$.

Exercice 7

Reprendre l'exercice précédent en utilisant la bibliothèque **Scipy** pour la résolution des équations différentielles et **Matplotlib** pour les représentations graphiques.