Série 11

Mardi prochain : Parcourez les chapitres 5 et 6.1 des notebooks Jupyter et résolvez les exercices qui y sont proposés.

Partiellement en classe vendredi

Exercice 1

On considère l'équation différentielle suivante

$$\begin{cases} y'(t) = -ty^2(t), \quad t > 0 \\ y(0) = 2. \end{cases}$$

On veut résoudre cette équation avec les méthodes d'Euler progressive et Euler rétrograde, dans l'intervalle $[0, 4]$ avec $N_h = 20$ sous-intervalles. Ceci équivaut à un pas de temps $h = 0.2$ et donc à approcher la solution exacte $y(t_n)$ aux instants $t_n = nh$, $n = 0, 1, \ldots 20$ (donc $t_n = 0.2, 0.4, 0.6, \ldots$) par une solution numérique u_n.

On veut utiliser les programmes Python créé précédemment en utilisant les appels de fonction suivants :

— *Euler progressive*

```
f = lambda t,x = —t*x**2
Nh = 20; tspan = [0,4]; y0 = 2
t_EP, y_EP = forwardEuler(f, tspan, y0, Nh)
```

Les variables de sortie `t_EP` et `y_EP` contiennent respectivement la suite des instants t_n et les valeurs u_n correspondantes calculées par la méthode.

— *Euler rétrograde*

La fonction `beuler` utilise la même syntaxe :

```
t_ER, y_ER = backwardEuler(f, tspan, y0, Nh);
```

Comparer la solution exacte et celles obtenues par les méthodes d'Euler progressive et rétrograde. Pour cela afficher la solution exacte et celles obtenues par les deux méthodes sur un même graphe.

Exercice 2

On considère le problème de Cauchy suivant :

$$\begin{cases} y'(t) = -e^t y^2(t) & t \in [1, 3] \\ y(1) = 2 \end{cases}$$

a) Ecrivez la méthode d'Euler rétrograde pour approcher la solution $y(t)$.

b) Réécrivez la méthode d'Euler rétrograde sous la forme

$$u_{n+1} = \varphi(u_{n+1}; n, h, u_n),$$

où n, h, u_n sont considérés de paramètres, et écrivez le schéma de Newton pour résoudre cette équation non linéaire.

Exercice 3

On considère la méthode de Crank–Nicolson

$$\begin{cases} \dfrac{u_{n+1} - u_n}{h} = \dfrac{1}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right], \quad n = 0, 1, \dots \\ u_0 = y_0, \end{cases} \tag{1}$$

h étant le pas de temps, pour approcher la solution du problème de Cauchy

$$\begin{cases} y'(t) = f(t, y(t)), \quad t \in (0, T), \\ y(0) = y_0. \end{cases}$$

Soit $\lambda < 0$ un nombre réel *négatif* donné. On considère le problème modèle

$$\begin{cases} y'(t) = \lambda y(t), \quad t > 0, \\ y(0) = 1, \end{cases} \tag{2}$$

dont la solution exacte est $y(t) = e^{\lambda t}$.

1. Écrivez le schéma de Crank–Nicolson pour l'approximation numérique du problème de Cauchy (2).
2. Résolvez cette équation et donnez une expression explicite de u_n en fonction de h, λ et n.
3. Trouvez en fonction de λ les valeurs de h pour lesquelles la méthode de Crank–Nicolson appliqué à ce problème est telle que $u_n \to 0$. On dira que la méthode de Crank–Nicolson est absolument stable pour ces valeurs de h et λ.

Python

Exercice 4

Ecrire une fonction `forwardEuler` qui approche la solution du problème

$$y'(t) = f(t, y(t)),\ t \in (T_0, T_f), \quad y(0) = y_0, \tag{3}$$

en utilisant la méthode d'Euler progressif. L'entête de la fonction doit être la suivante :

```
def forwardEuler( fun, interval, y0, N ) :
    #  FORWARDEULER Solve differential equations using the forward Euler
       method.
```

```
    #  [T, U] = FORWARDEULER( FUN, INTERVAL, Y0, N ), with INTERVAL = [T0,
       TF],
    #  integrates the system of differential equations y'=f(t, y) from time
       T0
    #  to time TF, with initial condition Y0, using the forward Euler method
       on
    #  an equispaced grid of N intervals. Function FUN(T, Y) must return
    #  a column vector corresponding to f(t, y). Each row in the solution
    #  array U corresponds to a time returned in the column vector T.
```

Une fois écrite la fonction `forwardEuler`, utiliser les commandes suivantes pour calculer la solution :

```
f = lambda t,x : ( 2/15*x*(1-x/1000) ) # C=2/15 et B = 1000
tsp = [0,100]
y0 = 100; Nh = 25;
t25,u25 = forwardEuler(f,tsp,y0,Nh)

plt.plot(t25,u25,'o')
```

Approcher la solution de l'équation différentielle

$$y'(t) = \frac{2\,y}{15}(1 - y/1000),\ t \in (0, 100), \quad y(0) = 100,$$

avec $N_h = 25$.

Que se passe-t-il avec $N_h = 7$? Discuter les résultats obtenus.

Exercice 5

Écrivez une fonction Python `backwardEuler` qui approche la solution du problème

$$y'(t) = f(t, y(t)),\ t \in (T_0, T_f), \quad y(0) = y_0, \tag{4}$$

en utilisant la méthode d'Euler rétrograde. L'entête de la fonction doit être la suivante :

```
def backwardEuler( fun, interval, y0, N ) :
    #  BACKWARDEULER Solve differential equations using the backward Euler
       method.
    #  [T, U] = BACKWARDEULER( FUN, INTERVAL, Y0, N ), with INTERVAL = [T0,
       TF],
    #  integrates the system of differential equations y'=f(t, y) from time
       T0
    #  to time TF, with initial condition Y0, using the backward Euler
       method on
    #  an equispaced grid of N intervals. Function FUN(T, Y) must return
    #  a column vector corresponding to f(t, y). Each row in the solution
    #  array U corresponds to a time returned in the column vector T.
    from scipy.optimize import fsolve
```

Remarque : Une fois écrite la fonction `backwardEuler`. Vous disposez de la fonction 'scipy.optimize.fsolve' pour résoudre une équation non-linéaire (`fsolve` cache plusieurs méthodes de resolution, comme Newton ou le point fixe).

on peut chercher le zéro de 'F' près de 'x0' en utilisant le code suivant :

```
from scipy.optimize import fsolve
a = 5; h = 0.1;
F = lambda x :  a + np.sin(x) - h*x;

x0 = a+h
zero = fsolve(F, x0)

print(f'F({zero[0]:5.2f}) = {F(zero)[0]:4.1e}')
```

Approcher la solution de l'équation différentielle

$$y'(t) = \frac{2\,y}{15}(1 - y/1000),\ t \in (0, 100), \quad y(0) = 100,$$

avec $N_h = 25$. Que se passe-t-il avec $N_h = 7$? Discuter les résultats obtenus.

Exercice 6

On considère une population de y individus dans un environnement où au plus $B = 1000$ individus peuvent coexister. On suppose qu'initialement le nombre d'individus est $y_0 = 100$ et que le facteur de croissance est égal à une constante C. Le modèle de l'évolution de la population considérée sur 100 années est le suivant :

$$y'(t) = Cy(t)\left(1 - \frac{y(t)}{B}\right),\ t \in (0, 100), \quad y(0) = y_0, \tag{5}$$

où t est mesuré *en années*, et $C = 2/15$ an^{-1}.

Soit u_n l'approximation de $y(t_n)$, où $t_n = nh$, $n = 0, 1, 2, \ldots, N_h$, $h = 100/N_h$ étant le pas de temps, N_h étant le nombre de pas temporels. Plus N_h est grand, plus le pas de temps est petit et plus l'approximation sera précise.

1. Ecrire les schémas d'Euler progressif (explicite) et rétrograde (implicite) pour calculer une approximation u_n de $y(t_n)$ (donner les équations de récurrence définissant la suite u_n dans les deux cas).
2. On prend $h = 1/12$ an ; donner la valeur u_1 qui approche le nombre d'individus $y(t_1)$ au temps t_1 (donc après *1 mois*), obtenue *i)* par la méthode d'Euler progressive, puis *ii)* par la méthode d'Euler rétrograde. Suggestion : resoudre à la main l'équation pour trouver la nouvelle valeur u_{i+1}.
3. En utilisant la méthode d'Euler progressif (copier et modifier la fonction forwardEuler.m créé précédemment), calculer les valeurs approchées u_n, $n = 0, 1, \ldots, N_h$ en Python, pour $N_h = 20$.
4. Tracer un graphe de la solution numérique trouvée en fonction du temps.
5. Faire à nouveau le calcul précédent, mais en prenant $N_h = 1000$; tracer la nouvelle solution numérique.

6. D'après les résultats trouvés, combien d'années sont nécessaires pour que la population atteigne le nombre de 900 individus ? *Suggestion : utiliser la commande* `find`.