Université Lyon 1 Année 2018-2019
Licence Mathématiques 2ème année
Analyse numérique

TP 3

Exercice 1 - Schémas numériques

Pour $f : \mathbb{R} \mapsto \mathbb{R}$ continue, on considère le problème de Cauchy autonome suivant sur un intervalle $[0, T]$, $T > 0$

$$\begin{cases} X(0) = X_0 \in \mathbb{R} \\ X'(t) = f(X(t)),\ t \in [0, T] \end{cases} \tag{1}$$

Nous souhaitons résoudre cette équation différentielle à l'aide de schémas numériques. Pour cela, on fixe un nombre de pas N et le pas de temps associé $h = T/N$, et on considère la suite des temps d'approximation équidistribués $(t_n)_{0 \leq n \leq N} = (n\,h)_{0 \leq n \leq N}$.

Pour chacun des schémas numériques suivant, compléter si besoin le code correspondant dans le fichier `schemas.py` à télécharger au lien suivant : `http://math.univ-lyon1.fr/~brehier/L2/schemas.py`.

– *Schéma d'Euler explicite*

$$\begin{cases} x_0 = X_0 \\ \forall n \in [\![0, N-1]\!], \quad x_{n+1} = x_n + h f(x_n)\,. \end{cases} \tag{2}$$

– *Schéma d'Euler implicite*

$$\begin{cases} x_0 = X_0 \\ \forall n \in [\![0, N-1]\!], \quad x_{n+1} = x_n + h f(x_{n+1})\,. \end{cases} \tag{3}$$

– *Schéma de Crank-Nicolson*

$$\begin{cases} x_0 = X_0 \\ \forall n \in [\![0, N-1]\!], \quad x_{n+1} = x_n + \frac{h}{2}\Big[f(x_n) + f(x_{n+1})\Big]\,. \end{cases} \tag{4}$$

Les deux méthodes précédentes nécessitent en général de déterminer une approximation de x_{n+1} comme solution d'une équation non linéaire, par exemple par la méthode de Newton.
Dans ce TP, nous utilisons la fonction `fsolve` du package `scipy.optimize`.

– *Schéma du point milieu explicite (RK2)*

$$\begin{cases} x_0 = X_0 \\ \forall n \in [\![0, N-1]\!], \quad x_{n+1} = x_n + h f\Big(x_n + \dfrac{h}{2} f(x_n)\Big)\,. \end{cases} \tag{5}$$

– *Schéma de Heun*

$$\begin{cases} x_0 = X_0 \\ \forall n \in [\![0, N-1]\!], \quad x_{n+1} = x_n + \dfrac{h}{2}\left[f(x_n) + f\Big(x_n + h f(x_n)\Big)\right]\,. \end{cases} \tag{6}$$

– *Schéma de Runge-Kutta RK4*

$$\begin{cases} x_0 = X_0 \\ \forall n \in [\![0, N-1]\!], \quad k_1 = f(x_n), \\ \qquad\qquad\qquad\quad k_2 = f(x_n + \dfrac{h}{2}\, k_1), \\ \qquad\qquad\qquad\quad k_3 = f(x_n + \dfrac{h}{2}\, k_2), \\ \qquad\qquad\qquad\quad k_4 = f(x_n + h\, k_3), \\ \qquad\qquad\qquad\quad x_{n+1} = x_n + \frac{h}{6}\Big(k_1 + 2k_2 + 2k_3 + k_4\Big). \end{cases} \tag{7}$$

Exercice 2 - Ordre des schémas

Nous allons étudier l'ordre des schémas numériques précédents. Pour cela, on considère le problème de Cauchy suivant :

$$\begin{cases} \forall t \in [0,5],\ X'(t) = -X(t), \\ X(0) = 2. \end{cases} \tag{8}$$

1. Calculer la solution exacte X_{ex} du problème (8).

RÉPONSE :
$X_{ex}(t) =$

2. Dans un fichier `tp3.py`, définir la fonction `f` correspondant au problème (8).

RÉPONSE :
```
def f(x):
```

3. Dans ce même fichier, définir la fonction `sol_exacte` correspondant au problème (8).

RÉPONSE :
```
def sol_exacte(t):
```

4. *Schéma d'Euler explicite*
   (a) Créer un vecteur `tN` contenant les nombres de pas N suivants : 50, 75, 100, 125, 150, 175 et 200. Pour chacun de ces pas de temps h,
   – créer un vecteur `t` contenant la liste des points équirépartis $(t_n)_{0\leq n\leq N}$ contenus dans l'intervalle $[0,5]$,
   – créer un vecteur `Xex` contenant les valeurs de la solution exacte $(X_{ex}(t_n))_{0\leq n\leq N}$,
   – calculer les itérés du schéma d'Euler explicite, sauvegardé dans un vecteur `X1`,
   – calculer $err(h) = \max\limits_{0\leq n\leq N} \big|\texttt{Xex}[n] - \texttt{X1}[n]\big|$, sauvegardé dans un vecteur `E1`.

(b) Que signifie qu'une méthode est d'ordre p ? En déduire quelle courbe tracer pour observer l'ordre de convergence de la méthode.

RÉPONSE :

(c) Quel ordre de convergence obtient-on ? Pour le déterminer, on pourra utiliser la commande `polyfit` du package `numpy` (cf TP2).

RÉPONSE :

Ordre obtenu numériquement (à 10^{-3} près) :

Ordre théorique :

5. *Schéma d'Euler implicite*
Refaire les mêmes calculs pour le schéma d'Euler implicite et sauvegarder l'erreur dans un vecteur `E2`.

RÉPONSE :

Ordre obtenu numériquement (à 10^{-3} près) :

Ordre théorique :

6. *Schéma dde Crank-Nicolson*
Refaire les mêmes calculs pour le schéma de Crank-Nicolson et sauvegarder l'erreur dans un vecteur `E3`.

RÉPONSE :

Ordre obtenu numériquement (à 10^{-3} près) :

Ordre théorique :

7. *Point milieu*
Déterminer l'ordre de convergence numérique du schéma du point milieu en procédant comme précédemment et en sauvegardant l'erreur dans un vecteur `E4`.

RÉPONSE :

Ordre obtenu numériquement (à 10^{-3} près) :

Ordre théorique :

8. *Schéma de Heun*
   Déterminer l'ordre de convergence numérique du schéma de Heun en procédant comme précédemment et en sauvegardant l'erreur dans un vecteur `E5`.

   RÉPONSE :

   Ordre obtenu numériquement (à 10^{-3} près) :

   Ordre théorique :

9. *Schéma de Runge-Kutta 4*
   Déterminer l'ordre de convergence numérique du schéma de Runge-Kutta 4 en procédant comme précédemment et en sauvegardant l'erreur dans un vecteur `E6`.

   RÉPONSE :

   Ordre obtenu numériquement (à 10^{-3} près) :

   Ordre théorique :

10. Sur une même figure, tracer les courbes mettant en évidence l'ordre de convergence de chacun des 5 schémas à l'aide de la commande `plot` du package `matplotlib.pyplot` en utilisant les vecteurs `E1`, `E2`, `E3`, `E4`, `E5` et `E6`. Enregistrer la figure au format png sous le nom `convergence.png` en mettant une légende et un titre. Il pourrait être utile d'utiliser la fonction `savefig` du package `matplotlib.pyplot`.