Intégration numérique des équations différentielles du premier ordre

1 Introduction

1.1 Objectifs du Tp

L'objectif de ce tp est de comparer les vitesses de convergence de plusieurs schémas d'intégration d'équations différentielles :

- méthode d'Euler explicite
- méthode d'Euler implicite
- méthode de Heun
- méthode de Runge Kunta 4

1.2 Rappel du cas général

Résoudre un problème de Cauchy consiste à trouver la fonction $\mathbf{Y}$ de $[t_0, t_f] \to \mathbb{R}^N$, telle que :

$$\begin{cases} \dfrac{d\mathbf{Y}}{dt} = \mathbf{F}(\mathbf{Y}, t) \\ \mathbf{Y}(t_0) = \mathbf{Y_0} \end{cases} \quad \text{où } t \in [t_0, t_f] \text{ et } \mathbf{Y_0} \in \mathbb{R}^N \text{ sont des données.} \qquad (1)$$

Dans le problème posé (2) :

$$\begin{array}{rcl} t & \to & x \\ \mathbf{Y} & \to & y(x) \\ \mathbf{F}(\mathbf{Y}, t) & \to & f(y,x) = -\dfrac{y}{x} \\ t_0 & \to & x_0 = 1 \\ \mathbf{Y}_0 & \to & 1 \end{array}$$

1.3 Support du tp

Objectif :

L'objectif de cette partie est de calculer une solution approchée du problème de Cauchy suivant :

$$\begin{cases} y'(x) = -\dfrac{y(x)}{x}, \ \forall x \in [1,10] \\ y(1) = 1 \end{cases} \quad \text{qui admet } y(x) = \frac{1}{x} \text{ comme solution exacte.} \qquad (2)$$

Q - 1 : *Construire la fonction* `F1surX(y,x)` *associée au problème de Cauchy (2).*

Remarque : on pourra aussi tester les méthodes sur les deux problèmes suivant :

$$\begin{cases} y'(x) = -y(x), \ \forall x \in [0,10] \\ y(0) = 1 \end{cases} \qquad (3)$$

qui admet $y(x) = e^{-x}$ comme solution exacte.

$$\begin{cases} y'(x) = \frac{1}{3}.(10 - y(x)), \ \forall x \in [0,10] \\ y(0) = 0 \end{cases} \qquad (4)$$

qui admet $y(x) = 10.\left(1 - e^{(-x/3)}\right)$ comme solution exacte.

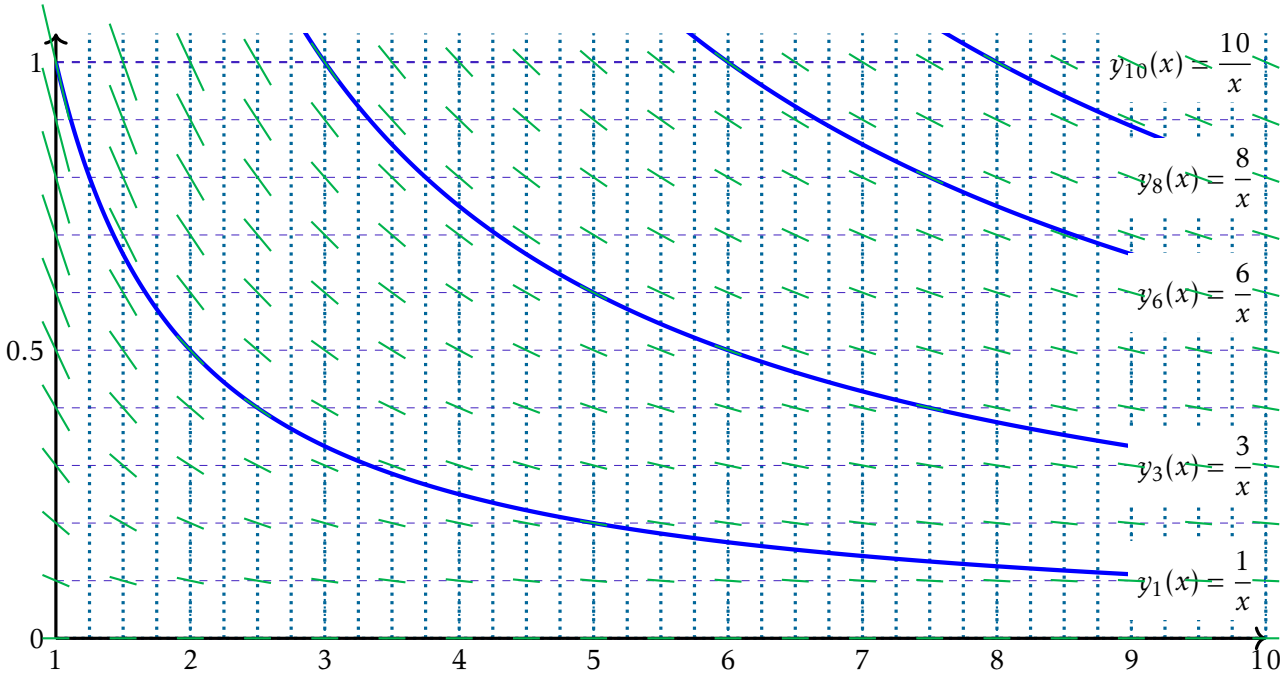

FIGURE 1 – Fonctions $y_k(x) = \dfrac{k}{x}$ avec pour $k = 1$, y_k solution du problème de Cauchy (2).

1.4 Arguments

Pour construire une fonction permettant de trouver les $(y_i)_0^n$, il est possible de choisir différents types d'arguments. On peut :

- entrer les bornes a, b et le nombre de points n.

```
def EulerExp(f,y0,a,b,n):
```

- entrer les bornes a, b et le pas h.

```
def EulerExp(f,y0,a,b,h):
```

- entrer la borne de gauche a, le nombre de points n et le pas h.

```
def EulerExp(f,y0,a,n,h):
```

- entrer un vecteur de points en abscisses.

```
def EulerExp(f,y0,X):
```

Nous opterons pour le dernier cas qui permet d'intégrer le premier cas avec **linspace** ou le deuxième avec **arange** :

```
EulerExp(f,y0,numpy.linspace(a,b,n)):
```

```
EulerExp(f,y0,numpy.arange(a,b+h,h)):
```

et qui correspond aux arguments de la fonction **odeint** :

```
scipy.integrate.odeint(f,y0,numpy.linspace(a,b,n))
scipy.integrate.odeint(f,y0,numpy.arange(a,b+h,h))
```

2 Comparaisons des méthodes

2.1 Intégration numérique d'une équation différentielle du premier ordre

Résoudre un problème de Cauchy en dimension 1 consiste à trouver la fonction $y(t)$ de $[t_0, t_f] \to \mathbb{R}$, telle que :

$$\begin{cases} \dfrac{dy}{dt} = f(y,t) \\ \\ y(t_0) = y_0 \end{cases} \tag{5}$$

où $t \in [t_0, t_f]$ et $y_0 \in \mathbb{R}$ sont des données.

Les méthodes de résolution numérique des équations différentielles sont basées sur des techniques d'estimation de l'intégrale de la fonction f.

On s'intéressera dans cette partie aux méthodes de résolution à un pas qui se mettent sous la forme générale :

$$\begin{cases} y_{i+1} = y_i + h.\Phi(y,t,h) \\ y(t_0) = y_0 \end{cases} \text{ où } \Phi \text{ est donné par la méthode.} \tag{6}$$

2.2 Méthodes d'Euler

2.2.1 Méthode d'Euler explicite

Dans la méthode d'Euler explicite, la fonction d'approximation au pas i est $\Phi = f(y_i, t_i)$. Le schéma d'Euler explicite permet de passer donc de l'état i à l'état $i+1$ grâce à la relation de récurrence : $y_{i+1} = y_i + h.f(y_i, x_i)$, où $h = x_{i+1} - x_i$.

Q - 2 : *Construire, en utilisant un schéma d'Euler explicite, la fonction* `EulerExp(f, y0, X)` *qui permet de trouver l'évolution de* $y(x)$ *à partir de l'état initial* `y0` *pour toutes les valeurs de* x *contenues dans* `X` *et rangées par ordre croissant .*

Q - 3 : *Tracer l'évolution de* y *pour* $x \in [1, 10]$ *pour différentes valeur de pas* h *ou différentes valeurs du nombre de points* n.

Q - 4 : *Que se passe-t-il si* $h = 1$, $h > 1$ *et* $h \in [0, 1[$?

2.2.2 Méthode d'Euler implicite

Dans la méthode d'Euler explicite, la fonction d'approximation au pas i est $\Phi = f(y_{i+1}, t_{i+1})$.

La relation de récurrence dans le cas à une dimension est : $y_{i+1} = y_i + h.f(y_{i+1}, t_{i+1})$

Il s'agit alors de résoudre à chaque itération l'équation :

$$g_i(y) = 0 \quad \text{et} \quad g_i(y) = y_i + h.f(y, t_{i+1}) - y \quad \text{avec} \quad g_i'(y) = h.\frac{\partial f(y, t_{i+1})}{\partial y} - 1$$

Grâce à g_i' on pourrait utiliser une programme de résolution basé sur l'algorithme de Newton que nous avons créé au SIM-NUM-1-Tp-1. Cependant la résolution sera faite à l'aide de la fonction `newton` de la bibliothèque **scipy.optimize**. Ainsi pour la récurrence, on pourra utliser :

```
newton(lambda Y:y[i-1]+h.f(Y,T[i])-Y, y[i-1])
```

On constate qu'on n'est pas obligé de donner la dérivée de g pour la résolution. Cependant, si on veut le faire, il suffit de l'écrire en troisième argument. Mais alors `EulerImp` prendre 4 argument au lieu de 3 pour `EulerExp`.

Q - 5 : *Programmer la méthode d'Euler implicite sans prise en compte de la dérivée dans les arguments.*

2.3 Méthodes de Runge Kutta

La méthode générale de Runge Kutta consiste à déterminer Φ à partir d'approximations successives. On choisit :

$$\Phi(y,t,h) = \sum_{m=1}^{q} \omega_m.k_m(y,t,h) \quad \text{avec} \quad k_1(y,t,h) = f(y,t)$$

$$\text{pour} \quad m \geq 2, \qquad k_m(y,t,h) = f\left(y + \sum_{j=1}^{m-1} \beta_{mj}.k_j(y,t,h)\ ,\ t + \alpha_m.h\right)$$

2.3.1 Méthode de Heun (RK2)

Pour cette méthode, $q = 2$ Le relation de récurrence est $y_{i+1} = y_i + \frac{h}{2}.f(y_i,t_i) + \frac{h}{2}.f\left(y_i + h.f(y_i,t_i)\ ,\ t_i + h\right)$.
On a donc pris $\Phi(y,t,h) = \frac{k_1+k_2}{2}$ avec : $k_1 = f(y_i,t_i)$ et $k_2 = f(y_i + h.k_1\ ,\ t_i + h)$.

$$y_{i+1} = y_i + \frac{h}{2}.\left(\underbrace{f(y_i,t_i)}_{k_1} + \underbrace{f(y_i + h.\overbrace{f(y_i,t_i)}^{k_1}, t_i + h)}_{k_2}\right)$$

Q - 6 : *Construire la fonction `Heun(f,y0,T)` relative à la méthode de Heun détaillée précédemment.*

Q - 7 : *Déterminer expérimentalement l'ordre de la méthode.*

2.3.2 Méthode de Runge Kutta 4 (RK4)

La fonction Φ est $\Phi(y,t,h) = \frac{1}{6}.(k_1 + 2.k_2 + 2.k_3 + k_4)$ avec quatre évaluations successives de f :

$$\begin{aligned}
k_1(y,t,h) &= f(y,t) \\
k_2(y,t,h) &= f\left(y + \frac{h}{2}.k_1(y,t,h)\ ,\ t + \frac{h}{2}\right) \\
k_3(y,t,h) &= f\left(y + \frac{h}{2}.k_2(y,t,h), t + \frac{h}{2}\right) \\
k_4(y,t,h) &= f(y + h.k_3(y,t,h), t + h)
\end{aligned}$$

d'où $y_{i+1} = y_i + \frac{h}{6}.(k_1 + 2.k_2 + 2.k_3 + k_4)$

Q - 8 : *Construire la fonction* `RK4(f,y0,X)` *relative à la méthode de Runge Kutta 4 détaillée précédemment.*

2.4 Synthèse

La résolution du problème (5) est implémentée dans le module **scipy**. Il suffit d'appeler la fonction **scipy.integrate.odei** pour l'utiliser. Cette fonction est optimisée par rapport à celle que nous venons de créer.

Q - 9 : *Comparer graphiquement les solutions approchées calculées par chacune des méthodes.*

Q - 10 : *Comparer les erreurs de consistance et les vitesses de convergences des différentes méthodes ainsi que les temps de calculs.*

Q - 11 : *Recommencer l'étude avec les problèmes (3) et (4) ou encore, avec le problème ci-dessous :*

$$\begin{cases} y'(x) &= -2.\dfrac{y}{x},\ \forall x \in [1,10] \\ y(1) &= 1 \end{cases} \qquad \textit{qui admet } y(x) = \frac{1}{x^2} \textit{ comme solution exacte.}$$