Exercice type *examen final*

Approximation numérique de la solution d'une équation différentielle

Le problème :

Supposons que l'on vous demande de calculer la solution $y(t)$ correspondant à l'évolution temporelle de l'équation différentielle :

$$\dot{y}(t) + ay(t) = \mathrm{b}$$

avec a et b des constantes strictement positives.

Cette trajectoire débutera au temps $t = 0$ avec la condition initiale $y(0) = y_0$

La solution mathématique :

La solution exacte (à retrouver !) s'écrit :

$$y(t) = \frac{b}{a} + (y_0 - \frac{b}{a})e^{-at}$$

Imaginez maintenant que l'équation différentielle soit plus complexe (a et b non constants par exemple). Le calcul de la solution exacte devient alors difficile voire impossible à faire. Nous allons voir comment il est possible d'approximer cette solution numériquement.

La solution informatique :

D'un point de vue informatique il est possible de réaliser cette approximation de la solution avec le principe suivant :

On suppose la fonction $\boldsymbol{f}$ qui décrit l'équation différentielle qui est dans notre cas d'ordre 1 :

$$\dot{y}(t) = f(y, t)$$

On considère un petit intervalle de temps h. En partant d'un point particulier de la trajectoire $y(T)$, puisque $f(y(T), T)$ représente la pente à la trajectoire en ce point particulier, il est possible d'approximer le point *suivant* $y(T + h)$ comme :

$$y(T + h) = y(T) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

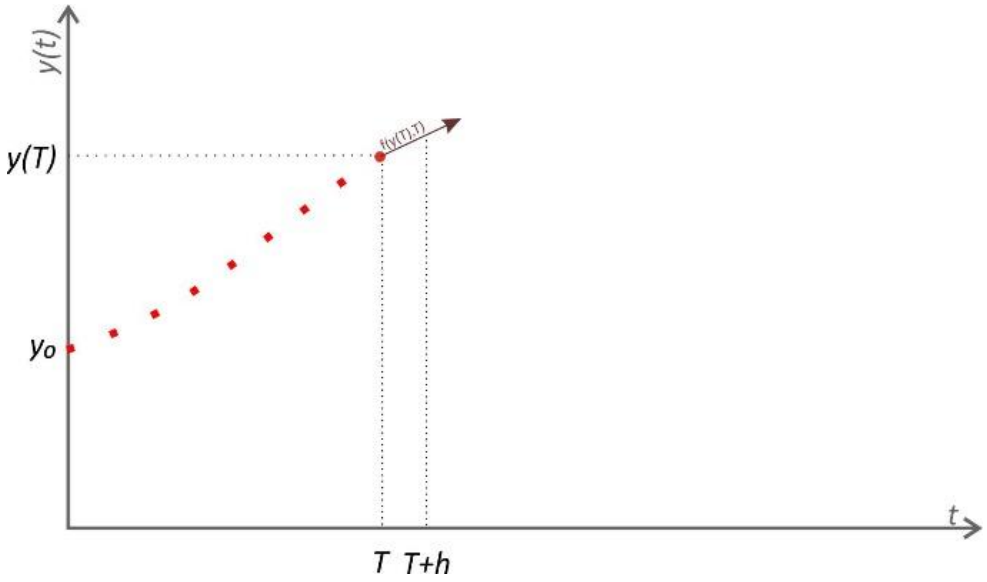
où les coefficient k_i sont calculés de proche en proche comme :

$$\begin{cases} k_1 = f(y(T), T) \\ k_2 = f\left(y(T) + \frac{h}{2}k_1, T + \frac{h}{2}\right) \\ k_3 = f(y(T) + \frac{h}{2}k_2, T + \frac{h}{2}) \\ k_4 = f(y(T) + hk_3, T + h) \end{cases}$$

L'algorithme de la construction est simple : en partant de la condition initiale y_0, il est possible de calculer un premier point $y(h)$. Puis de ce premier point, on peut en calculer un second $y(2h)$,… etc.

Si l'on cherche à estimer la solution sur un horizon temporel $[0, t_f]$, il est alors nécessaire de faire N boucles itératives de ce calcul avec $N = \frac{t_f}{h}$

La figure suivante illustre ce principe de construction de la trajectoire par calculs successifs.



Le travail à faire :

Etape 1 : Ecrire un script qui saisit les différents paramètres du problème $(a, b, y0, tf, h)$ et qui construit 2 « listes de floats » correspondant au calcul de la solution exacte :

- la première liste (`Temps`) contiendra l'évolution du temps (de 0 à tf avec un pas de h) ;
- la seconde liste (`TrajExacte`) contiendra l'évolution de la trajectoire pour les différents instants de `Temps`;
- `Temps` et `TrajExacte` seront donc deux listes de même dimension.

Le script se terminera par le tracé de la courbe « solution exacte » avec les lignes de code suivantes (en important le module matplotlib.pyplot) :

```
import matplotlib.pyplot as pl

fig = pl.figure(10, facecolor=[0.92, 0.92, 0.81])
ax = fig.add_axes([0.05,0.2,.90,.75])
ax.plot(Temps,TrajExacte,'+r')
pl.show()
fig.close
```

Etape 2 : Créer la fonction $Derive(y, t, a, b)$ qui calcule la dérivée de l'équation différentielle en un point y et à un temps t.

Etape 3 : Ecrire une fonction $RK(y0, tf, h)$ qui calculera l'approximation de la solution avec la méthode présentée ci-dessus. Cette fonction fera évidement appel à la fonction $Derive$ pour le calcul des coefficients k_i. La fonction retournera une liste qui contiendra les différentes valeurs de l'approximation de la solution. Il est possible d'effectuer le tracé des deux solutions (celle exacte et celle approximée) en adaptant les lignes précédentes comme :

```
. . .
TrajApprox = RK(y0,tf,h)
fig = pl.figure(10, facecolor=[0.92, 0.92, 0.81])
ax = fig.add_axes([0.05,0.2,.90,.75])
ax.plot(Temps,TrajExacte,'+r', Temps,TrajApprox,'*b')
pl.show()
fig.close
```