TP5- Équations et systèmes différentiels linéaires

Durée : 3h

1 Introduction

L'objectif de ce TP est d'illustrer les concepts vus l'an dernier dans sur les équations différentielles ainsi que ceux que nous verrons dans le chapitre 14 sur les systèmes différentiels linéaires.

Nous allons voir comment utiliser Python pour résoudre des équations différentielles avec le langage Python. Pour cela, nous utiliserons la commande `odeint` de la librairie `scipy.integrate`. Il est donc important de commencerpar importer cette fonction :

```
from scipy.integrate import odeint
```

Les librairies `numpy` et `matplotlib.pyplot` seront aussi utiles :

```
import numpy as np
import matplotlib.pyplot as plt
```

2 Équations différentielles d'ordre 1

2.1 Rappels théoriques de première année

Rappels

Soient I un intervalle de $\mathbb{R}$ et $f : \mathrm{I} \times \mathbb{R} \to \mathbb{R}$ une application.
Soit y une fonction définie sur I.

- On dit que y **est une solution de l'équation différentielle d'ordre 1**

$$y' = f(y, t)$$

si et seulement si y est dérivable sur I et pour tout $t \in \mathrm{I}$:

$$y'(t) = f(y(t), t).$$

- Étant donné $(t_0, y_0) \in \mathrm{I} \times \mathbb{R}$, on dit que y **est une solution du problème de Cauchy**

$$\left\{ \begin{array}{ccc} y' & = & f(y,t) \\ y(t_0) & = & y_0 \end{array} \right.$$

si et seulement si y est dérivable sur I, $y(t_0) = y_0$ et pour tout $t \in \mathrm{I}$:

$$y'(t) = f(y(t), t).$$

Exemple 1

Les équations suivantes sont des équations différentielles d'ordre 1 :

1. $y' = 2y$ *(ici* $f(y,t) = 2y$ *pour tout* $(t,y) \in \mathbb{R} \times \mathbb{R}$*) ;*
2. $y' = -3y + te^t$ *(ici* $f(y,t) = -3y + te^t$ *pour tout* $(t,y) \in \mathbb{R} \times \mathbb{R}$*) ;*
3. $y' + 2y = \ln(t)$ *(ici* $f(y,t) = -2y + \ln(t)$ *pour tout* $(t,y) \in \mathbb{R}_+^* \times \mathbb{R}$*.*

*Ces trois exemples sont des cas particuliers d'**équations différentielles linéaires d'ordre 1** qui sont les équations différentielles de la forme :*

$$y'(t) + ay(t) = b(t)$$

a un réel et b une fonction continue sur I *à valeurs dans* $\mathbb{R}$*.*

► Vérifier que les fonctions $t \mapsto 2e^{2t}$ et $t \mapsto e^{2t}$ sont solutions de l'équation différentielle $y' - 2y = 0$.

► Vérifier que les fonctions $t \mapsto 2e^{2t}$ est solution du problème de Cauchy :

$$y' - 2y = 0 \quad \text{et} \quad y(0) = 2.$$

Rappels

Soient I un intervalle de $\mathbb{R}$, a un réel et b une fonction continue sur I à valeurs dans $\mathbb{R}$. Étant donné $(t_0, y_0) \in \mathrm{I} \times \mathbb{R}$, le problème de Cauchy

$$\left\{ \begin{array}{rcl} y' + ay & = & b \\ y(t_0) & = & y_0 \end{array} \right.$$

possède une unique solution.

2.2 Résolution avec Python

Soient I un intervalle de $\mathbb{R}$ et $f : \mathrm{I} \times \mathbb{R} \to \mathbb{R}$ une application. On s'intéresse au problème de Cauchy

$$\left\{ \begin{array}{rcl} y' & = & f(y,t) \\ y(t_0) & = & y_0 \end{array} \right. .$$

Résolution avec Python

On suppose déjà déclarées dans Python :

- une variable `t` contenant un vecteur $(t_0, \dots, t_p) \in \mathrm{I}^p$;
- une variable `y0` contenant le réel y_0 ;
- une fonction `f` d'en-tête `def f(y,t)` qui définie la fonction f.

Alors la commande :

```
sol = odeint(f,y0,t)
```

affectent à la variable `sol` le vecteur colonne dont les composantes sont $(y(t_0), \dots, y(t_p))$ où y est la solution du problème de Cauchy.
La commande

```
plt.plot(t,sol)
```

permet alors de représenter graphiquement y sur l'intervalle $[t_0, t_p]$.

On s'intéresse au problème de Cauchy

$$\left\{ \begin{array}{rcl} y' - 3y & = & t^2 e^{3t} \\ y(0) & = & 1 \end{array} \right. .$$

► Écrire le problème de Cauchy sous la forme

$$\left\{ \begin{array}{rcl} y' & = & f(y,t) \\ y(0) & = & 1 \end{array} \right.$$

où $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ est à préciser

- Création de la variable `t` : créer un vecteur `t` avec 100 composantes également réparties entre 0 et 1.

- Créer la variable `y0`.

- Création de la fonction f : créer une fonction `f` d'en-tête `def f(y,t)` qui prend en argument deux réels `y` et `t` et qui renvoie la valeur de f(`y`,`t`).

- Avec les commandes `odeint` et `plt.plot` déterminer puis représenter graphiquement la solution y du problème de Cauchy.

- Justifier que la solution de ce problème de Cauchy est la fonction $y : t \mapsto \left(1+\frac{t^3}{3}\right)e^{3t}$.

- Afficher la courbe représentative de la fonction ci-dessus sur le même graphique que précédemment et comparée la solution exacte avec la solution calculée par Python.

On considère le programme suivant

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

t = np.linspace(-2 ,2 , 200)
for k in range ( 1 , 10) :
  def f ( y , t ) :
    return 2*y
  z = odeint (f , k , t )
  plt.plot ( t , z , label = " Cas k = " + str ( k ) )
  plt.legend ()
plt.show ()
```

- Expliquer ce que fait ce programme.

- Quelle est le comportement en $+\infty$ des courbes obtenues ? En $-\infty$?

- Résoudre à la main les problèmes de Cauchy associés à ce programme.

- On appelle **état d'équilibre** d'une équation différentielle toute solution constante de cette équation. Déterminer le(s) état(s) d'équilibre(s) de l'équation différentielle associée à ce programme.

3 Systèmes différentiels linéaires

3.1 Résolution avec Python

On considère le **système linéaire à coefficients constant**

$$\left\{\begin{array}{ccccccccc} a_{1,1}y_1 & + & a_{1,2}y_2 & + & \ldots & + & a_{1,n}y_n & = & y_1' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,1}y_1 & + & a_{n,2}y_2 & + & \ldots & + & a_{n,n}y_n & = & y_n' \end{array}\right.$$

avec pour conditions initiales :

$$\left\{\begin{array}{ccc} y_1(t_0) & = & b_1 \\ \vdots & \vdots & \vdots \\ y_n(t_0) & = & b_n. \end{array}\right.$$

L'écriture matricielle est donnée par :

$$X' = AX$$

où $A = (a_{i,j})_{1\leqslant i,j\leqslant n} \in \mathscr{M}_n(\mathbb{R})$ et $X(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix}$.

Résolution avec Python

On suppose déjà déclarées dans Python :

- une variable `A` contenant la matrice A ;
- une variable `t` contenant un vecteur $(t_0, \ldots, t_p)$;
- une variable `X0` contenant le vecteur colonne $X(t_0)$;
- la fonction `syst` :

```
def syst(X,t):
    return np.dot(A,X)
```

Alors la commande :

```
M = odeint(syst,X0,t)
```

affectent à la variable `M` la matrice

$$\begin{pmatrix} y_1(t_0) & \cdots & y_n(t_0) \\ \vdots & \ldots & \vdots \\ y_1(t_p) & \cdots & y_n(t_p) \end{pmatrix}$$

où $(y_1, \ldots, y_n)$ est la solution du système avec condition initiale.
Dans le cas d'un système de taille 2, la trajectoire d'une solution est l'ensemble $\{(y_1(t), y_2(t)) \mid t \in \mathbb{R}\}$ que l'on peut représenter dans le plan à l'aide de la commande :

```
plt.plot(M[:,0],M[:,1])
```

On s'intéresse au système

$$\left\{\begin{array}{ccc} x'(t) & = & 2x(t) + y(t) \\ y'(t) & = & x(t) + 2y(t) \end{array}\right. \quad \text{avec} \quad x(0) = 2 \text{ et } y(0) = 0.$$

► Écrire le système sous forme matricielle. On appellera A la matrice qui intervient.

► Création de la variable `t` : créer un vecteur `t` avec 50 composantes également réparties entre 0 et 1.

- Créer la variable `X0` contenant le vecteur $\begin{pmatrix}2\\0\end{pmatrix}$.

- Définir la fonction `syst`.
- Avec les commandes `odeint` et `plt.plot` déterminer puis représenter graphiquement la trajectoire de la solution (x, y).

- Résoudre à la main le système.

- Afficher la trajectoire de la solution que vous avez trouvé ci-dessus sur le même graphique que précédemment et comparée la solution exacte avec la solution calculée par Python.

3.2 Cas des équations différentielles linéaires d'ordre 2

Soient a et b deux réels. On considère l'équation différentielle d'ordre 2 suivante :

$$y'' + ay' + by = 0$$

avec pour conditions initiales $y(t_0) = y_0$ et $y'(t_0) = y_1$.

En posant $\mathrm{Y} = \begin{pmatrix}y\\y'\end{pmatrix}$ et $\mathrm{A} = \begin{pmatrix}0 & 1\\-b & -a\end{pmatrix}$, on a vu en cours que l'équation différentielle d'ordre 2 $y'' + ay' + by = 0$ est équivalente au système différentiel $\mathrm{Y}' = \mathrm{AY}$.

Résolution avec Python

On suppose déjà déclarées dans Python :

- une variable `A` contenant la matrice A;
- une variable `t` contenant un vecteur $(t_0, \dots, t_p)$;
- une variable `Y0` contenant le vecteur colonne $\begin{pmatrix}y_0\\y_1\end{pmatrix}$;
- la fonction `syst` :

```
def syst(X,t):
    return np.dot(A,X)
```

Alors la commande :

```
M = odeint(syst,Y0,t)
```

affectent à la variable `M` la matrice

$$\begin{pmatrix} y(t_0) & y'(t_0)\\ \vdots & \vdots \\ y(t_p) & y'(t_p)\end{pmatrix}$$

où y est la solution de l'équation différentielle avec condition initiale.

On s'intéresse à l'équation différentielle d'ordre 2

$$y'' - y' - 2y = 0 \quad \text{avec} \quad y(0) = 2 \text{ et } y'(0) = 1.$$

- Écrire le système équivalent. On appellera A la matrice qui intervient.

- Création de la variable `t` : créer un vecteur `t` avec 50 composantes également réparties entre 0 et 1.

- Créer la variable `Y0` contenant le vecteur $\begin{pmatrix}2\\1\end{pmatrix}$.

- Définir la fonction `syst`.
- Avec les commandes `odeint` et `plt.plot` déterminer puis représenter graphiquement la trajectoire de la solution y.

- Résoudre à la main le système.

- Afficher la trajectoire de la solution que vous avez trouvé ci-dessus sur le même graphique que précédemment et comparée la solution exacte avec la solution calculée par Python.

4 Exercices

4.1 Un système diagonalisable

On considère le système suivant :

$$\left\{\begin{array}{rcl} x'(t) & = & -17x(t) - 3y(t) \\ y'(t) & = & 3x(t) - 7y(t) \end{array}\right. .$$

1. Écrire le système sous forme matricielle. On appellera A la matrice qui intervient.
2. Recopier et exécuter dans Python le programme suivant :

```
from numpy import linalg as al
A = np.array([[-17,-3],[3,-7]])
Sp,VP = al.eig(A)
print(Sp)
print(Vp)
```

Quel est la fonction de la commande `al.eig`?

3. Déterminer les valeurs propres de A et une base de chaque sous-espace propre. Comparer avec ce que renvoie le programme de la question précédente.
4. Avec Python, tracer les trajectoires des solutions du système avec pour conditions initiales $(x(0), y(0)) = (0,3)$.
5. Même question avec $(x(0), y(0)) = (2,2)$ puis $(x(0), y(0)) = (1,-3)$, $(x(0), y(0)) = (0,-3)$, $(x(0), y(0)) = (-2,-2)$ et $(x(0), y(0)) = (-3,1)$.
6. Deux trajectoires semblent rectilignes, lesquelles? Expliquer pourquoi.
7. Les trajectoires semblent converger vers un point, lequel? Expliquer pourquoi.

4.2 Un système non diagonalisable

On considère le système suivant :

$$\left\{\begin{array}{rcl} x'(t) & = & -y(t) \\ y'(t) & = & x(t) \end{array}\right. \quad \text{avec} \quad (x(0), y(0)) = (1,0).$$

1. Écrire le système sous forme matricielle. On appellera A la matrice qui intervient.
2. Vérifier que A n'est pas diagonalisable.
3. Tracer la trajectoire de la solution. On ajoutera la commande `plt.axis("equal")` avant la commande `plt.plot` afin de rendre le repère orthonormé.
   Que remarquez-vous?
4. En dérivant la fonction $t \mapsto x(t)^2 + y(t)^2$ justifier le constat de la question précédente.
5. Avec Python, tracer les trajectoires des solutions du système avec pour conditions initiales $(x(0), y(0)) = (0,3)$.
6. Même question avec $(x(0), y(0)) = (2,2)$ puis $(x(0), y(0)) = (1,-3)$, $(x(0), y(0)) = (0,-3)$, $(x(0), y(0)) = (-2,-2)$ et $(x(0), y(0)) = (-3,1)$.
7. Deux trajectoires semblent rectilignes, lesquelles? Expliquer pourquoi.
8. Les trajectoires semblent converger vers un point, lequel? Expliquer pourquoi.

4.3 Un système à paramètre

Soit $k \in \mathbb{N}$. On considère le système suivant :

$$\left\{\begin{array}{rcl} x'(t) & = & (k-12)x(t) + 4y(t) \\ y'(t) & = & -8x(t) + ky(t) \end{array}\right. \quad \text{avec} \quad (x(0), y(0)) = (0,1).$$

1. On considère la matrice $A_k = \begin{pmatrix} k-12 & 4 \\ -8 & k \end{pmatrix}$.
   (a) Vérifier que les valeurs propres de A_k sont les solutions de l'équation :
   $$\lambda^2 - 2(k-6)\lambda - 12k + 32 = 0.$$
   (b) En déduire que A_k possède deux valeurs propres distinctes que l'on exprimera en fonction de k et déterminer une base de chaque sous-espace propre.
   La matrice A_k est-elle diagonalisable?
2. Recopier et exécuter le programme suivant :

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import odeint

plt.xlim(-0.1,2.7)
plt.ylim(-0.1,4)

t = np.linspace(0,10,300)
X0 = [0,1]
for k in range(1,9) :
    Ak = np.array([[k-12,4],[-8,k]])
    def syst(X,t) :
        return np.dot(Ak,X)
    M = odeint(syst,X0,t)
    x = M[:,0]
    y = M[:,1]
```

```
    plt.plot(x,y,label='Cas k='+str(k))
plt.legend()
plt.plot(0,0,'ko')
plt.plot(0,1,'k+')
plt.plot(1,2,'k*')
plt.show()
```

Les commandes

```
plt.plot(0,0,'ko')
plt.plot(0,1,'k+')
plt.plot(1,2,'k*')
plt.show()
```

permettent de marquer d'un rond le point $(0,0)$, d'un + le point $(0,1)$ et d'une étoile le point $(1,2)$.

3. (a) Justifier le comportement des trajectoires en $+\infty$ correspondant aux valeurs $k \in [\![5,8]\!]$.
   (b) Justifier la convergence de la trajectoire pour $k = 4$.
   (c) Justifier le phénomène de convergence observé pour les valeurs $k \in [\![1,3]\!]$.
4. On remplace la condition initiale $(x(0), y(0)) = (0,1)$ par la condition initiale $(x(0), y(0)) = (1,1)$.
   (a) Modifier le programme de la question 2 en conséquence et observer le résultat obtenu.
   (b) Justifier l'observation faite à la question précédente.