TP 10

Équations différentielles

Compétences attendues.

- ✓ Savoir représenter les trajectoires des solutions d'une équation différentielle ou d'un système différentiel.
- ✓ Visualiser l'influence des paramètres d'une équation différentielle ou d'un système différentiel.

Liste des commandes `Python` exigibles aux concours.
Le programme officiel précise qu'on pourra utiliser la fonction `odeint` de la bibliothèque `scipy.integrate`, mais que sa maitrise n'est pas exigible aux concours.

Anthony Mansuy
Professeur de Mathématiques en deuxième année de CPGE filière ECG au Lycée Clemenceau (Reims)
Page personnelle : `http://anthony-mansuy.fr`
E-mail : `mansuy.anthony@hotmail.fr`

Le langage `Python` permet de "résoudre" certaines équations différentielles et d'en visualiser les trajectoires grâce à la fonction `odeint` de la librairie `scipy.integrate`.

Pour utiliser la fonction `odeint`, il faut commander l'importation suivante :

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

On pensera également à commander les importations suivantes pour la suite :

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

1 Équations différentielles d'ordre 1

De façon la plus générale, une équation différentielle d'ordre 1 s'écrit sous la forme

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

où t est la variable des fonctions y et y'.

Exemple. L'équation $y'(t) + 2y(t) = 1 + t$ s'écrit sous la forme $y'(t) = f(y(t),t)$ avec $f(y,t) = -2y + 1 + t$.

Définition.

Considérons l'équation différentielle (E) : $y' = f(y,t)$ avec la condition initiale $y(t_0) = y_0$.

On suppose déclarées :

- Une variable `t` contenant le vecteur $(t_0, t_1, \dots, t_p)$;
- Une variable `y0` contenant le réel y_0.

Alors les commandes

```
def f(y, t):
    return expression de f
sol = odeint(f, y0, t)
```

affectent à la variable `sol` le vecteur colonne dont les composantes sont $(y(t_0), y(t_1), \dots, y(t_p))$, où y est la solution de l'équation différentielle (E).

Pour la représenter graphiquement sur l'intervalle $[t_0, t_p]$, il suffit alors de commander les instructions :

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

Exemple. Reprenons l'équation différentielle suivante étudiée au **Chapitre 14** :

$$\begin{cases} y' + 2y = 1 + t, \\ y(0) = 1. \end{cases}$$

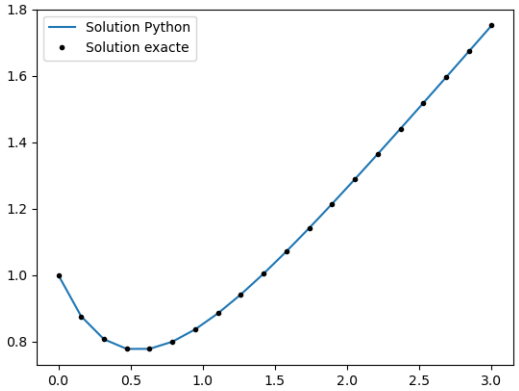
On a vu que la solution de cette équation différentielle est donnée par :

$$\forall t \in \mathbb{R}, \quad y(t) = \frac{3}{4}e^{-2t} + \frac{1}{2}t + \frac{1}{4}.$$

Vérifions ces résultats à l'aide de `Python` :

```
t = np.linspace(0, 3, 20)
y0 = 1
def f(y, t):
    return -2*y+1+t
sol = odeint(f, y0, t)
plt.plot(t, sol, label="Solution Python")
plt.plot(t, 3/4*np.exp(-2*t)+t/2+1/4, 'k.', label="Solution exacte")
plt.legend()
plt.show()
```

On obtient le graphique suivant qui permet de confirmer les résultats obtenus en cours :



2 Systèmes différentiels

On considère le système différentiel linéaire (S) suivant :

$$\begin{cases} x_1' &= a_{1,1}x_1 + \ldots + a_{1,n}x_n \\ &\vdots \\ x_n' &= a_{n,1}x_1 + \ldots + a_{n,n}x_n \end{cases}, \text{ avec les conditions initiales } \begin{cases} x_1(t_0) &= b_1 \\ &\vdots \\ x_n(t_0) &= b_n \end{cases}$$

où les $(a_{i,j})_{1\leq i,j\leq n}$, $(b_i)_{1\leq i\leq n}$ et t_0 sont des réels.

Ce système s'écrit sous la forme matricielle $X' = AX$ avec les conditions initiales $X(t_0) = X_0$ en posant

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{pmatrix} \quad \text{et} \quad X_0 = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

Définition.

Avec les notations précédentes, on suppose déclarées :

- Une variable `A` contenant la matrice A ;
- Une variable `t` contenant le vecteur $(t_0, t_1, \ldots, t_p)$;
- Une variable `X0` contenant le vecteur (ou la liste) X_0.

Alors, les commandes

```
def syst(X, t):
    return np.dot(A, X)
M = odeint(syst, X0, t)
```

affectent à la variable `M` la matrice suivante :

$$\begin{pmatrix} x_1(t_0) & x_2(t_0) & \ldots & x_n(t_0) \\ x_1(t_1) & x_2(t_1) & \ldots & x_n(t_1) \\ \vdots & \vdots & & \vdots \\ x_1(t_n) & x_2(t_p) & \ldots & x_n(t_p) \end{pmatrix}.$$

La k-ième colonne de `M` est donc formée des valeurs que prend la k-ième composante de la solution du système, à savoir x_k, en chacune des composantes t_i du vecteur `t`.

Remarques.

1. La variable `t` figure parmi les paramètres d'entrée de la fonction `syst` bien qu'elle n'intervienne pas dans l'expression de la valeur retournée. C'est parce que, dans notre cas, le système est à coefficients constants mais on pourrait imaginer qu'il ne le soit pas.
2. Pour obtenir les valeurs de x_k, on commande `M[:, k-1]` (k-ième colonne de `M`).

Cas d'un système de taille $n = 2$.

Définition.

Dans le cas d'un système de taille $n = 2$, la trajectoire d'une solution est l'ensemble

$$\{(x(t), y(t)) \mid t \in \mathbb{R}\}.$$

Sa représentation graphique est donc en général une courbe du plan $\mathbb{R}^2$ (avec x en abscisse et y en ordonnée, la variable t n'apparaissant pas explicitement).
Pour représenter la trajectoire, il suffit de commander les instructions :

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

où `M` est la variable contenant la matrice :

$$\begin{pmatrix} x_1(t_0) & x_2(t_0) \\ x_1(t_1) & x_2(t_1) \\ \vdots & \vdots \\ x_1(t_n) & x_2(t_p) \end{pmatrix}.$$

Exemple. Reprenons le système différentiel suivant étudié à l'**Exercice 15** du **TD 14** :

$$\begin{cases} x' = x + 3y \\ y' = x - y \end{cases} \quad \text{et} \quad \begin{cases} x(0) = 4 \\ y(0) = 0 \end{cases}$$

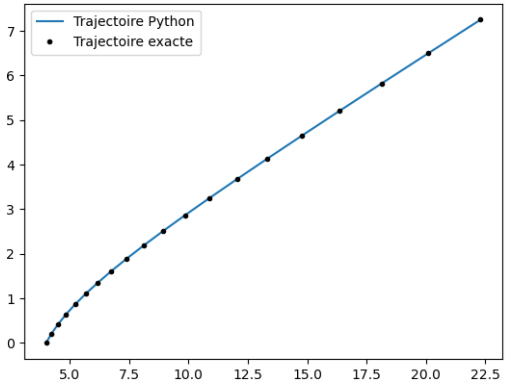
On a vu que la solution de cette équation différentielle est donnée par :

$$\forall t \in \mathbb{R}, \quad x(t) = e^{-2t} + 3e^{2t}, \quad y(t) = -e^{-2t} + e^{2t}.$$

Vérifions ces résultats à l'aide de `Python` :

```
A = np.array([[1, 3], [1, -1]])
X0 = [4, 0]
t = np.linspace(0, 1, 20)
def syst(X, t):
    return np.dot(A, X)
M = odeint(syst, X0, t)
x = M[:, 0]
y = M[:, 1]
plt.plot(x, y, label="Solution Python")
u = np.exp(-2*t)+3*np.exp(2*t)
v = -np.exp(-2*t)+np.exp(2*t)
plt.plot(u, v, 'k.', label="Trajectoire exacte")
plt.legend()
plt.show()
```

On obtient le graphique suivant qui permet de confirmer les résultats obtenus en TD :



Cas d'un système de taille $n \geq 3$.

Définition.

Dans le cas d'un système de taille $n \geq 3$, on ne peut pas représenter graphiquement la trajectoire. On peut en revanche représenter graphiquement chacune des fonctions x_k (composantes de la solution).
Pour cela, il suffit de commander les instructions :

```
for k in range(n):
    plt.plot(t, M[:, k])
plt.show()
```

où `M` est la variable contenant la matrice :

$$\begin{pmatrix} x_1(t_0) & x_2(t_0) & \dots & x_n(t_0) \\ x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ \vdots & \vdots & & \vdots \\ x_1(t_n) & x_2(t_p) & \dots & x_n(t_p) \end{pmatrix}.$$

Exemple. Reprenons le système différentiel suivant étudié au **Chapitre 14** :

$$\begin{cases} x_1' &= x_1 + 2x_2 - x_3 \\ x_2' &= 2x_1 + 4x_2 - 2x_3 \\ x_3' &= -x_1 - 2x_2 + x_3 \end{cases} \quad \text{et} \quad \begin{cases} x_1(0) = 1 \\ x_2(0) = -1 \\ x_3(0) = 5 \end{cases}$$

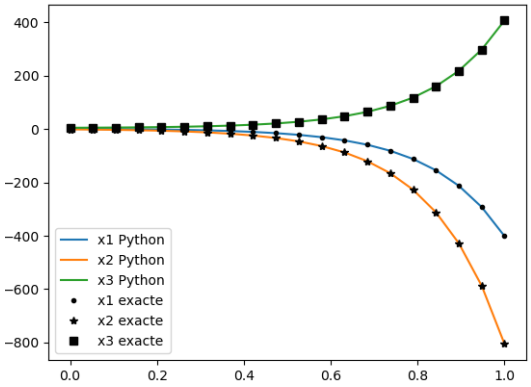
On a vu que la solution de ce système est donnée par :

$$\forall t \in \mathbb{R}, \quad x_1(t) = 2 - e^{6t},\ x_2(t) = 1 - 2e^{6t},\ x_3(t) = 4 + e^{6t}.$$

Vérifions ces résultats à l'aide de `Python` :

```
A = np.array([[1, 2, 3], [2,4,-2],[-1,-2,1]])
X0 = [1,-1,5]
t = np.linspace(0, 1, 20)
def syst(X, t):
    return np.dot(A, X)
M = odeint(syst, X0, t)
x1 = M[:, 0]
x2 = M[:, 1]
x3 = M[:, 2]
plt.plot(t, x1, label="x1 Python")
plt.plot(t, x2, label="x2 Python")
plt.plot(t, x3, label="x3 Python")
u = 2-np.exp(6*t)
v = 1-2*np.exp(6*t)
w = 4+np.exp(6*t)
plt.plot(t, u, 'k.', label="x1 exacte")
plt.plot(t, v, 'k.', label="x2 exacte")
plt.plot(t, w, 'k.', label="x3 exacte")
plt.legend()
plt.show()
```

On obtient le graphique suivant qui permet de confirmer les résultats obtenus en cours :



3 Équations différentielles linéaires d'ordre 2

Nous avons vu au chapitre 14 que l'équation différentielle linéaire d'ordre 2 à coefficients constants

$$(E) : \ y'' + ay' + by = 0$$

peut se ramener au système différentiel linéaire

$$X' = AX, \text{ où } A = \begin{pmatrix} 0 & 1 \\ -b & -a \end{pmatrix} \text{ et } X = \begin{pmatrix} y \\ y' \end{pmatrix}.$$

On peut utiliser `Python` pour résoudre (E) à laquelle on impose les conditions initiales $y(t_0) = y_0$ et $y'(t_0) = y_1$:

Définition.

Avec les notations précédentes, on suppose déclarées :

- Une variable `A = np.array([[0, 1], [-b, -a]])` ;
- Une variable `X0` contenant le vecteur (ou la liste) de composante y_0 et y_1 ;
- Une variable `t` contenant le vecteur $(t_0, t_1, \ldots, t_p)$.

Alors, les commandes

```
def syst(X, t):
    return np.dot(A, X)
M = odeint(syst, X0, t)
```

affectent à la variable `M` la matrice suivante :

$$\begin{pmatrix} y(t_0) & y'(t_0) \\ y(t_1) & y'(t_1) \\ \vdots & \vdots \\ y(t_p) & y'(t_p) \end{pmatrix}.$$

La première colonne de `M` est donc formée des valeurs que prend la solution du système en chacune des composantes t_i du vecteur `t`.

Exemple. Considérons l'équation différentielle suivante étudiée au **Chapitre 14** :

$$\begin{cases} y'' + y' - 2y = 0, \\ y(0) = 1, \\ y'(0) = 0. \end{cases}$$

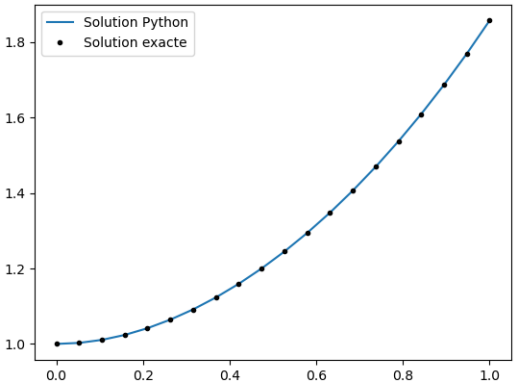
On peut montrer que la solution de cette équation différentielle est donnée par :

$$\forall t \in \mathbb{R}, \quad y(t) = \frac{1}{3}e^{-2t} + \frac{2}{3}e^{t}.$$

Vérifions ces résultats à l'aide de `Python` :

```
A = np.array([[0, 1], [2, -1]])
X0 = [1, 0]
t = np.linspace(0, 1, 20)
def syst(X, t):
    return(np.dot(A, X))
M = odeint(syst, X0, t)
y = M[:, 0]
plt.plot(t, y, label="Solution Python")
plt.plot(t, (1/3)*np.exp(-2*t)+(2/3)*np.exp(t), 'k.', label="Solution exacte")
plt.legend()
plt.show()
```

On obtient le graphique suivant qui permet de confirmer les résultats annoncés :



4 Exercices

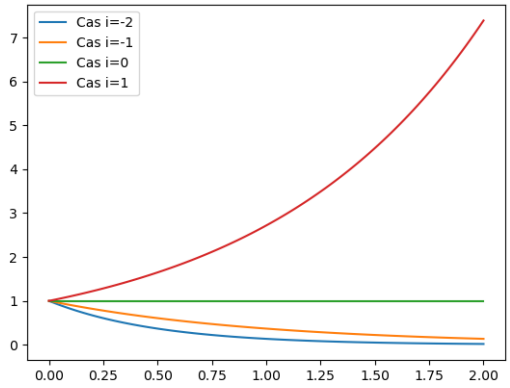
Exercice 1 (★)

On considère le script suivant :

```
t = np.linspace(0, 2, 100)
for i in range(-2, 2):
    def f(y, t):
        return i*y
    y0 = 1
    y = odeint(f, y0, t)
    plt.plot(t, y, label="Cas i="+str(i))
plt.legend()
plt.show()
```

L'option `label="Cas i="+str(i)` suivie de `plt.legend()` permet d'ajouter une légende au graphique.

Après exécution, on obtient la représentation graphique :



Justifier le comportement des courbes représentées en l'infini.

Exercice 2 (★★)
On considère le système différentiel suivant :

$$(S) : \begin{cases} x' = -17x - 3y \\ y' = 3x - 7y \end{cases}$$

1. On note A la matrice associée au système différentiel (S).
   (a) Définir la matrice A sur `Python` puis déterminer les valeurs propres de A et des vecteurs propres associés à l'aide de la commande `al.eig` (de la librairie `numpy.linalg`).
   (b) En déduire les solutions du système (S).
2. (a) Tracer avec `Python` sur un même graphique les trajectoires solutions pour les valeurs de $(x(0), y(0))$ suivantes :
   $$(-3, 1), \quad (0, 3), \quad (2, 1), \quad (1, -3), \quad (0, -3), \quad (-2, 2).$$
   (b) Justifier le phénomène de convergence des trajectoires vers le point d'équilibre $(0, 0)$.
   (c) Deux de ces trajectoires semblent être rectiligne. Expliquer pourquoi.

Exercice 3 (★)
On considère le système différentiel suivant :

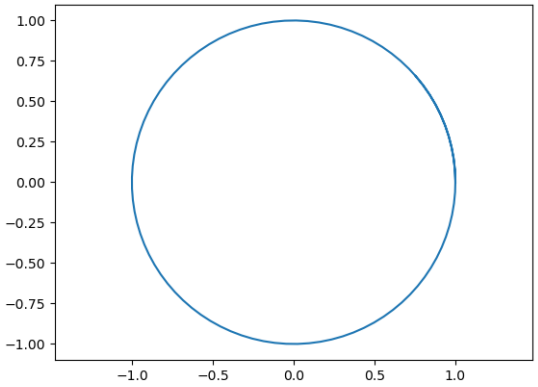
$$(S) : \begin{cases} x' = -y \\ y' = x \end{cases}$$

avec les conditions initiales $x(0) = 1$ et $y(0) = 0$.

1. Vérifier que la matrice $A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ n'est pas diagonalisable.
2. On considère le programme suivant :

```
A = np.array([[0, -1], [1, 0]])
X0 = [1, 0]
t = np.linspace(0, 7, 100)
def syst(X, t):
    return(np.dot(A, X))
M = odeint(syst, X0, t)
x = M[:, 0]
y = M[:, 1]
plt.axis("equal")
plt.plot(x, y)
plt.show()
```

Voici le graphique qu'il commande (la commande `plt.axis("equal")` rend le repère orthonormé) :



Dériver la fonction $t \mapsto x(t)^2 + y(t)^2$ et justifier ce qui est constaté sur le graphique.

On rappelle qu'un cercle de centre $A(x_A, y_A)$ et de rayon $r > 0$ est l'ensemble :

$$\mathscr{C}(A, r) = \left\{M \in \mathbb{R}^2 \mid d(A, M) = r\right\} = \left\{(x_M, y_M) \in \mathbb{R}^2 \mid (x_M - x_A)^2 + (y_M - y_A)^2 = r^2\right\}.$$

Exercice 4 (★★)

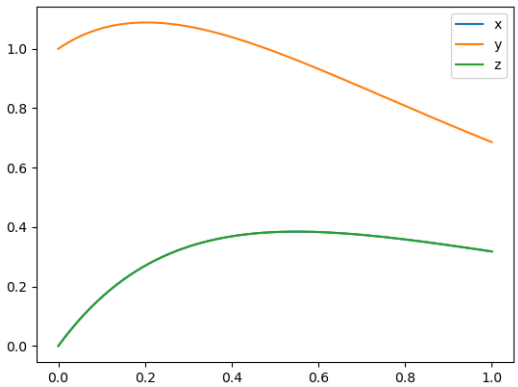
On considère le système différentiel suivant :

$$(S) : \begin{cases} x' &= -10x + 2y + 5z \\ y' &= -9x + y + 5z \\ z' &= -12x + 2y + 7z \end{cases}$$

1. On note A la matrice associée au système différentiel (S).
   (a) Définir la matrice A sur `Python` puis déterminer les valeurs propres de A et des vecteurs propres associés à l'aide de la commande `al.eig` (de la librairie `numpy.linalg`).
   (b) La matrice A est-elle diagonalisable ?
2. On considère le programme suivant :

```
A = np.array([[-10, 2, 5], [-9, 1, 5], [-12, 2, 7]])
X0 = [0, 1, 0]
t = np.linspace(0, 5, 200)
def syst(X, t):
    return(np.dot(A, X))
M = odeint(syst, X0, t)
x = M[:, 0]
y = M[:, 1]
z = M[:, 2]
plt.plot(t, x, label="x")
plt.plot(t, y, label="y")
plt.plot(t, z, label="z")
plt.legend()
plt.show()
```

Voici le graphique qu'il commande :



(a) Pourquoi ne voit-on que deux courbes au lieu de trois ?
(b) La trajectoire converge-t-elle et, si oui, quel est le point limite ?

Exercice 5 (★★)
On considère l'équation différentielle suivante :

$$(E) : \ y'' - y' - 2y = 0.$$

On veut étudier la convergence de la solution de cette équation dans les trois cas suivants :

$$(y(0) = 1, y'(0) = -2); \quad (y(0) = 1, y'(0) = -1); \quad (y(0) = 1, y'(0) = 1).$$

Pour chacun de ces trois cas :

1. Écrire un programme `Python` permettant de représenter graphiquement la solution.
2. A l'aide du graphique obtenu, conjecturer la convergence de la solution.
3. Valider la conjecture par le calcul.

Exercice 6 (★★)
On considère l'équation différentielle d'ordre 3 suivante :

$$(E) : \ y''' + 6y'' + 11y' + 6y = 0$$

avec les conditions initiales $y(0) = 3$, $y'(0) = -6$ et $y''(0) = 14$.

1. Montrer que résoudre cette équation différentielle se ramène à résoudre un système différentiel linéaire $X' = AX$ où $X = \begin{pmatrix} y \\ y' \\ y'' \end{pmatrix}$ et A est une matrice de $\mathscr{M}_3(\mathbb{R})$ que l'on déterminera.
2. (a) Déterminer les valeurs propres de A et des vecteurs propres associés à l'aide de la commande `al.eig` (de la librairie `numpy.linalg`).
   (b) En déduire le spectre de A ainsi qu'une matrice $P \in \mathscr{M}_3(\mathbb{R})$ inversible dont la première ligne n'est formée que de 1 et telle que $P^{-1}AP = D$, où D est diagonale avec les coefficients diagonaux rangés par ordre croissant.
   (c) Définir une variable `P` contenant la matrice P puis exécuter les instructions suivantes :

```
>>> X0 = [3, -6, 14]
>>> al.solve(P, X0)
```

   En déduire l'expression de l'unique solution de (E) telle que $y(0) = 3$, $y'(0) = -6$ et $y''(0) = 14$.
