Tracés de trajectoires

Code de partage avec Capytale : 1a40-8382209

On utilisera les bibliothèques suivantes :
`import numpy as np`
`import matplotlib.pyplot as plt`

On s'intéresse à un système différentiel du type :

$$(S)\ : \qquad \forall t \in \mathbb{R} \quad \left\{ \begin{array}{rcrcr} x'(t) & = & a_{1,1}x(t) & + & a_{1,2}y(t) \\ y'(t) & = & a_{2,1}x(t) & + & a_{2,2}y(t) \end{array} \right.$$

On rappelle qu'une trajectoire est l'ensemble $\{(x(t), y(t)), t \in \mathbb{R}\}$
Graphiquement, il s'agit de l'ensemble des points (x, y) parcourus par la solution. Dans un problème physique par exemple, il s'agit donc vraiment d'une « trajectoire ».

Cas n°1 - matrice diagonale à valeurs propres négatives

Dans ce premier cas, on va étudier les trajectoires d'une matrice diagonale dont les éléments diagonaux sont négatifs.

1. Rappeler les solutions dans ce cas

   On exécutera les programmes suivants, en cherchant à comprendre leurs résultats, sans rentrer dans le détail du programme.

```python
from scipy.integrate import odeint
# On définit ici le système différentiel
def syst(X, t, a11, a12, a21, a22) :
    x, y = X
    dXdt = [a11 * x + a12 * y, a21 * x + a22 * y]
    return dXdt

# Matrice A (choisir les coefficients)
a11, a12 = -1,0
a21, a22 = 0,-3
# Intervalle de temps sur lequel on trace la solution
t = np.linspace(0, 2, 100)

plt.close()

# Choix de la condition initiale
X0 = [0.75,0.5]
# On trace y en fonction de x
# plot en temps positif
sol = odeint(syst, X0, t, args=(a11, a12, a21, a22))
plt.plot(sol[:, 0], sol[:, 1], 'b')
# plot en temps négatif
sol = odeint(syst, X0, -t, args=(a11, a12, a21, a22))
plt.plot(sol[:, 0], sol[:, 1], 'b')
plt.show()
```

```python
# Configuration fenêtre graphique
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.axis([-1,1,-1,1])

# Représentation des tangentes aux trajectoires
g1 = np.linspace(-1,1,15)
g2 = np.linspace(-1,1,15)
A1, A2 = np.meshgrid(g1,g2)
V1,V2 = syst((A1,A2),t, a11, a12, a21, a22)
r1 = np.sqrt(1+V1**2)
r2 = np.sqrt(1+V2**2)
plt.quiver(A1, A2, V1/r1, V2/r2)
plt.show()
```

On complète le graphique précédent avec de nouvelles trajectoires

```python
# Matrice A (choisir les coefficients)
a11, a12 = -1,0
a21, a22 = 0,-3
# Intervalle de temps sur lequel on trace la solution
t = np.linspace(0, 2, 100)
plt.close()
L=[[0.75,0.5], [0.5,0.5],[0.25,0.75],[1,0],
[0,1],[-0.75,0.5], [-0.5,0.5],[-0.25,0.75],
[0,-1],[0.75,-0.5], [0.5,-0.5],[0.25,-0.75],
[-1,-0],[-0.75,-0.5], [-0.5,-0.5],[-0.25,-0.75]]

for X0 in L :

    # On trace y en fonction de x
    # plot en temps positif
    sol = odeint(syst, X0, t, args=(a11, a12, a21, a22))
    plt.plot(sol[:, 0], sol[:, 1], 'b')
    # plot en temps négatif
    sol = odeint(syst, X0, -t, args=(a11, a12, a21, a22))
    plt.plot(sol[:, 0], sol[:, 1], 'b')
plt.show()
```

2. Expliquer le graphique précédent et interpréter ses résultats.

Cas n°2 - matrice diagonale à valeurs propres positives

Reproduire le schéma précédent avec la matrice $A = \begin{pmatrix} 5 & 0 \\ 0 & 2 \end{pmatrix}$

Cas n°3 - matrice diagonale à valeurs propres de signes opposés

Reproduire le schéma précédent avec la matrice $A = \begin{pmatrix} 1 & 0 \\ 0 & -3 \end{pmatrix}$

Cas n°4 - matrice diagonale avec une valeur propre nulle

Reproduire le schéma précédent avec la matrice $A = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}$

Cas suivants - matrices diagonalisables

Reprendre les cas précédents quand A n'est pas diagonale mais est diagonalisable