Code de partage avec Capytale : c0d2-8228870

On utilisera les bibliothèques suivantes :

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

Exercice 1 - illustration de l'instabilité et des conditions initiales

On considère le système différentiel suivant :

$$(\mathscr{S}) : \quad \forall t \in \mathbb{R} \quad \begin{cases} x'(t) &= 3x(t) &- y(t) &+ z(t) \\ y'(t) &= 3x(t) &- 2y(t) &+ 2z(t) \\ z'(t) &= 3x(t) &+ y(t) &- z(t) \end{cases}$$

On notera $X(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$ pour tout $t \in \mathbb{R}$

1. Définir la matrice A sous Python.
2. Déterminer les valeurs propres de A (utiliser Python, on rappelle que `al.eig` renvoie les valeurs propres et des vecteurs propres associés).
   On pourra utiliser la matrice Q définie avec la commande suivante et la commande `np.diag` pour D

```
spectre, Q=al.eig(A)
for i in range(3):
    Q[:, i]=Q[:, i]/Q[2, i]
print (spectre, Q)
```

3. Déterminer une matrice P inversible et une matrice D diagonale telles que : la dernière ligne de P est $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$ et les éléments diagonaux de D sont rangés dans l'ordre croissant.
4. Résoudre $(\mathscr{S})$
   On écrira $X(t)$ sous la forme :

$$X(t) = C_0 e^{r_0 t} U + C_1 e^{r_1 t} V + C_2 e^{r_2 t} W$$

   U, V, W étant trois vecteurs colonnes à déterminer, r_0, r_1, r_2 trois réels à déterminer, C_0, C_1, C_2 étant des paramètres (constantes quelconques).

   La résolution de $(\mathscr{S})$ donne :

$$\exists (C_0, C_1, C_2) \in \mathbb{R}^3,\ \forall t \in \mathbb{R},\ X(t) = C_0 e^{-3t} U + C_1 V + C_2 e^{3t} W$$

   où : U, V, W sont les colonnes de P (dans cet ordre), c'est-à-dire :

$$U = \begin{pmatrix} -1/3 \\ -1 \\ 1 \end{pmatrix} \qquad V = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \qquad W = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

5. La fonction Python suivante définit la fonction `X(t, condition)` renvoyant un triplet `[x(t), y(t), z(t)]` avec $t \mapsto X(t)$ solution de $(\mathscr{S})$ sous la condition initiale $X(0) = {}^t(a \quad b \quad c)$. `condition` sera la liste `[a, b, c]`.
Expliquer les lignes 4 et 7.

```
def SolX(t, condition):
    P=np.array([[-1/3, 0, 1], [-1, 1, 1], [1, 1, 1]])
    r=np.array([-3, 0, 3])
    C=al.inv(P)@np.transpose(condition)
    X=np.zeros(3)
    for k in range(3):
        X=X+C[k]*np.exp(r[k]*t)*P[:, k]
    return X
```

6. Le code suivant trace les courbes x, y, z solution du problème de Cauchy avec $x(0) = -1/3$, $y(0) = -1$, $z(0) = 1$ sur trois graphiques différents.
Commentez le résultat.

```
T=np.linspace(0, 5, 100)
X=[]
Cauchy=[-1/3, -1, 1]
fonction=['x', 'y', 'z']
for t in T:
    X.append(SolX(t, Cauchy))

for k in range(3):
    plt.close(k)
    plt.figure(k)
    x=[X[i][k] for i in range(100)]
    plt.plot(T, x, label=fonction[k])
    plt.legend()
    plt.show()
```

7. Le code suivant trace les courbes x, y, z solution du problème de Cauchy avec $x(0) = -0.333333$, $y(0) = -1$, $z(0) = 1$ sur trois graphiques différents.
Commentez le résultat.

```
T=np.linspace(0, 5, 100)
X=[]
Cauchy=[-0.333333, -1, 1]
fonction=['x', 'y', 'z']
for t in T:
    X.append(SolX(t, Cauchy))

for k in range(3):
    plt.close(k)
    plt.figure(k)
    x=[X[i][k] for i in range(100)]
    plt.plot(T, x, label=fonction[k])
    plt.legend()
    plt.show()
```