TD6 - Matplotlib et tableaux Numpy

Dans ce TD, on utilisera :
- les listes par compréhension,
- les tableaux du package Numpy,
- des commandes du package Matplotlib pour tracer des graphes.

On peut travailler avec un fichier par exercice. On chargera les modules nécessaires avec :

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

Tableaux Numpy (à une dimension)

1 – *Diverses commandes pour créer des tableaux (une dimension)*
Exécuter, observer comprendre :

```
X = np.array([1,3,5,7])
print(type(X))
print(X)

Y = np.arange(-np.pi,np.pi,0.1)
print(type(Y))
print(Y)

Z = np.linspace(0,5,10)
print(type(Z))
print(np.size(Z))

T = np.zeros(7)
print(T)
```

2 – *Appliquer une fonction à un tableau*
On souhaite créer un tableau de valeurs numériques en appliquant une même fonction à chacun des éléments d'un autre tableau. Il y a plusieurs manières :

1. On peut bien sûr parcourir le tableau à l'aide d'une boucle.
```
X = np.linspace(-2,2,10)
Y = np.zeros(10)
for k in range(10):
  Y[k] = (X[k])**2
print(Y)
```
2. Les fonctions usuelles ont été réécrites dans Numpy afin de pouvoir s'appliquer directement à un tableau (vectorisation des fonctions).
```
X = np.arange(-np.pi,np.pi,0.1)
Y = np.sin(X)
print(Y)
```
3. On peut aussi vectoriser nos propres fonctions.
```
X = np.arange(1,10,1)
print(X)

def fact(n):
  p = 1
  for k in range(1,n):
    p = p*k
  return(p)
vfact = np.vectorize(fact)
Y = vfact(X)
print(Y)
```

3 – Sinon, on peut utiliser les listes Python , en utilisant la définition des listes par compréhension. Ecrire les commandes permettant de créer les listes correspondant aux tableaux ci-dessus.

Premiers exemples de graphes

4 – *Une parabole*
Exécuter, observer comprendre :

```
X = np.linspace(-2,2,10)
Y = X**2
plt.axis("equal")
plt.plot(X,Y)
plt.show()
```

5 – *Une spirale*
Exécuter, observer comprendre :

```
T = np.linspace(0,10,100)
X = T*np.cos(T)
Y = T*np.sin(T)
plt.axis("equal")
plt.plot(X,Y)
plt.show()
```

6 – Reproduire ce qui vient d'être fait mais avec des listes aux lieu des tableaux Numpy.

De la pratique

7 – *Une marmite*
Soit f la fonction définie sur $\mathbb{R}$ par : $f(x) = \arctan(x^2 + 5x)$.

1. Tracer sa courbe sur $[-10, 10]$.
2. Ajouter un titre à l'aide de la commande `plt.title("Titre")`.
3. Modifier la couleur en ajoutant l'option `color="green"`.
4. Modifier le type de trait et son épaisseur à l'aide des options linestyle et line width (par exemple `linestyle = "-"` et `linewidth = 2`).
5. Faire un zoom sur une partie de la courbe à l'aide de la commande `plt.axis([xmin,xmax,ymin,ymax])`.

8 – *Une fonction et sa réciproque*

1. Tracer sur le même graphique les courbes des fonctions sin et arcsin (np.arcsin).
2. Ajouter la première bissectrice (d'équation $y = x$) en pointillé.
3. Ajouter une légende à l'aide de l'option `label` à chaque commande `plot` (il faut ensuite afficher la légende avec la commande `plt.legend`).

9 – *Suite récurrente*
Soit f une fonction définie sur un intervalle I de $\mathbb{R}$ telle que $f(I) \subset I$ et $u_0 \in I$. On peut alors définir la suite (u_n) par la relation de récurrence $u_{n+1} = f(u_n)$ pour $n \in \mathbb{N}$. On voudrait tracer sur un même graphe la courbe $y = f(x)$, la droite $y = x$ et l'évolution de la suite (u_n). La fonction `escargot`, que l'on veut écrire, doit comporter trois arguments : la fonction f , le premier terme u_0 de la suite et le nombre n d'itérations. Pour être tout à fait complète, cette procédure doit aussi calculer la bonne échelle sur les axes en fonction des données (il faut que les termes de la suites soient dans l'intervalle du graphique).

1. Écrire la fonction demandée. Si possible, fournir une fonction utilisant les tableaux Numpy et une version utilisant les listes.
2. Tester cette procédure avec la fonction cos, $u_0 = \frac{\pi}{8}$ et $n = 9$. Vers quoi converge la suite (u_n) ainsi définie ?
3. Même chose avec $f(x) = \frac{1}{2}(x + \frac{2}{x})$, $u_0 = 12$ et $n = 5$. Vers quoi converge la suite (u_n) ainsi définie ?
   Indications :
   — on dispose des fonctions `np.min` et `np.max` pour calculer le minimum et le maximum d'un tableau de nombres ;
   — on rappelle que la commande `plot([x1,x2],[y1,y2])` permet de tracer le segment reliant les points de coordonnées (x_1, y_1) et (x_2, y_2).