INF 5 - Tracé de courbes

1. BIBLIOTHÈQUE PYPLOT ET COMMANDES DE BASE

Pour tracer des courbes avec Python, on utilisera la bibliothèque matplotlib.pyplot qu'il faudra d'abord importer ainsi :

python import matplotlib.pyplot as plt

La construction d'un graphique représentant une suite ou une fonction se fait à l'aise de plusieurs commandes à connaître. Regardons d'abord un exemple type :

```
#import des biliothèque
import matplotlib.pyplot as plt
import numpy as np

#on crée deux vecteurs, x est le vecteurs des abscisses entre 0 \
+ et 5, avec 20 points
#y est le vecteurs des exp(x) pour chacun des points d'abscisse
x = np.linspace(0,5,20)
y = np.exp(x)

plt.plot(x,y) #crée le tracé
plt.show() #affiche le tracé
```

1. `plt.plot` prend en valeur deux vecteurs de même taille et trace l'ensemble des points d'abscisse et d'ordonnées correspondantes,
2. `plt.show` affiche le tracé

On peut tracer plusieurs courbes sur la même figure. Essayer les commandes suivantes.

```
#import des biliothèque
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(- 2*np.pi, 2*np.pi ,50)
y1 = np.cos(x)
y2 = np.sin(x)

plt.plot(x,y1, 'r')
plt.plot(x,y2, 'b')
plt.show()
```

Remarque 1.1 —

1. Que font les options 'r' et 'b' ?
2. Et si on mettait :

```
#import des biliothèque
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(- 2*np.pi, 2*np.pi ,50)
y1 = np.cos(x)
y2 = np.sin(x)

plt.plot(x,y1, 'ro')
plt.plot(x,y2, 'b+')
plt.show()
```

De nombreuses options existent pour `plot`, on peut les voir dans la documentation de Python. Il y a l'épaisseur, le type de trait ...

2. PERSONNALISATION DES COURBES

Pour rendre le graphique plus beau et surtout plus clair, d'autres commandes existent :

- `plt.title` prend en entrée une chaine de caractère et la met en titre du graphique
- `plt.xlabel` et `plt.ylabel` prennent aussi des chaînes de caractère en entrée pour les mettre en nom des axes
- `plt.grid` affiche une grille sur le graphique.

On peut aussi mettre une vraie légende dans le graphique, cela rend plus clair les figures avec plusieurs courbes. La ligne plt.legend() suffit à l'afficher et règle tout toute seule. Pour cela quand on lance un tracé il faut lui donner un nom et il apparaitra dans la légende avec l'option `label`. Reprenons l'exemple des fonctions trigonométriques.

```
x = np.linspace(- 2*np.pi, 2*np.pi ,50)
y1 = np.cos(x)
y2 = np.sin(x)

plt.plot(x,y1, label = "cos(x)")
plt.plot(x,y2, label = "sin(x)")
plt.legend()
plt.show()
```

On peut personnaliser l'affichage de la légende (position, etc) : voir dans le guide de la commande.

3. EXEMPLES

Étude d'une suite connue. On va tracer la suite définie par

$$u_n = \left(1+\frac{1}{n}\right)^n.$$

On a vu, ou on verra, que cette suite converge vers le réel e. Pour illustrer la convergence, on construire la fonction suivante

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

def approxE(n):
    x = range(1,n+1) #vecteur des abcisse
    y = np.zeros(n) #initialisation ordonnées
    e = (np.e)*np.ones(n) #deuxieme vecteur ordonnees : pour la \
+ limite
    for k in range(n):
        y[k] = (1+ 1/(k+1))**(k+1) #calcul des abscisse
    plt.plot(x,y,'r+') #tracé
    plt.plot(x,e, 'b', linestyle='dashed')#tracé de la limite
    plt.show()
```

Dans ce cas là, on a utilisé une suite explicite et on avait décidé le temps d'arrêt à l'avance. On peut complexifier la chose en rajoutant des difficultés vues dans les cours précédent :

1. une boucle While avec un arrêt en fonction de la convergence. Cela rajoute une difficulté majeure : on ne sait pas à l'avance la taille du tableau à créer.
2. la construction d'une suite par récurrence.

Pour cela, modifions le code de l'algorithme de Héron pour rajouter des tracés.

```
    u = 1
    y = [u] #initialisation du tableau des ordonnées
    aux = 0.5*(u + a/u) #variable auxiliiaire
    while abs(aux-u) > eps: #condition d'arret
        u = aux
        aux = 0.5*(u + a/u) #incrémentation de la suite
        y = np.append(y,u) #rajout du terme dans le tableau des \
+ ordonnées
    n = np.size(y) #on recupere la taille deu tableau des \
+ ordonnées
    x = np.arange(n) #creation tableau abscisses
    plt.plot(x,y) #trace
    plt.plot(np.sqrt(a)*np.ones(n)) #trace de l'asymptote
    plt.show()
```

Tracé de courbes de fonctions. Rien de particulier à dire ici, si ce n'est que on doit tracer une fonction comme une suite. Par exemple pour la fonction partie entière entre sur $]-5,5[$, on tapera

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

x = np.linspace(-5,5, 100) #100 est le nombre de points du tracé, \
+ on peut changer)
y = np.floor(x)

plt.plot(x,y)
plt.show()
```

Remarque 3.1 — Modifier le code pour qu'il ne trace pas les lignes de discontinuité de la partie entière. Faire de même avec la fonction tangente.