TP3 : Représentation graphique
Suites et convergence

1 Instructions de base

Pour faire de jolies représentations graphique, on a besoin de deux modules : numpy et matplotlib.pyplot

Commencez par installer puis importer les bibliothèques en question :

Aller dans Outils -> Gérer les paquets, installer matplotlib et numpy en les recherchant dans la barre de recherche.

```
from matplotlib.pyplot import *
from numpy import *
```

Ces bibliothèques contiennent de nombreux outils, dont les suivants :

Dans "matplotlib.pyplot " :

- **plot(x,y)**, avec x et y deux arrays ou listes de même taille $1 \times n$: construit une ligne brisée passant par les n points dont les abscisses sont donnés dans la liste x, les ordonnées dans la liste y.
- **close()** : ferme la fenêtre graphique
- **xlabel**(*chaîne de caractères*) permet de légender l'axe des abscisses ;
- **ylabel**(*chaîne de caractères*) permet de légender l'axe des ordonnées ;
- **title**(*chaîne de caractères*) permet d'afficher un titre.
- **show()** : affiche la fenêtre graphique. Tant qu'on a pas exécuté "show()", les commandes précédentes n'affichent rien.

Essayez les commandes suivantes :

dans l'éditeur, puis charger (CTRL+E)

```
close()
x=[1,3,5,5,4,2,1,1]
y=[2,1,2,5,4,4,5,2]
plot(x,y)
z=[3,3]
t=[1.5,2]
plot(z,t)
show()
```

Exercice 1 Vous avez compris le fonctionnement ? Dessinez moi un carré pour voir !

D'autres commandes de la bibliothèque **matplotlib.pyplot** peuvent être utilisées :

- On peut imposer la couleur des courbes avec un paramètre optionnel et les codes suivants :

'r'	'b'	'g'	'y'	'k'
rouge	bleu	vert	jaune	noir

  Ainsi, `plot(x,y,'g')` tracera en vert.

- On peut changer le type de ligne :

'-'	'- -'	' :'	'x'	'o'	's'
ligne continue	ligne discontinue	pointillés	croix	rond	carré

  Par exemple `plot(x,y,'o')` ne reliera pas les points donnés par x et y, mais fera des petits ronds à la place des points indiqués (pratique pour les suites)

- On peut combiner les deux pour avoir, par exemple, des pointillés rouge : `plot(x,y,'r:')`.

- La commande `legend([chaîne1,chaîne2, ...])` permet de légender les graphiques générés par des tracé de plot. La chaine1 sera associé au premier plot rencontré, la chaine2 au second, etc...
- La commande **axis('equal')** permet d'imposer un repère orthonormé .
- La commande **axis([xmin,xmax,ymin,ymax])** permet d'imposer le cadrage $[xmin, xmax] \times [ymin, ymax]$ de la fenêtre graphique.

Testez avec l'exemple suivant (on revient sur l'instruction linspace juste après) :

dans l'éditeur puis chargé dans le shell

```
close()
x=linspace(-3,3,5)
y1=x**2
y2=exp(x)
plot(x,y1,'--')
plot(x,y2,':')
legend(['carré','exponentiel'])
plot(x,y1,'ro')
axis([-4,4,-1,20])
show()
```

2 Tracés de courbes de fonctions

2.0.1 Fonctions classiques

Le tracé d'une courbe de fonction est une utilisation fréquente de la fonction plot, et se déroule toujours en trois étapes :

1. préparation des points en abscisses,
2. calcul de l'image de ces points par la fonction f,
3. affichage du tout.

Pour préparer les points en abscisses, l'instruction `linspace(a,b,n)` est très pratique. Elle est dans le module `numpy` et elle donne une liste de n nombres répartis de manière égale entre a et b.

Par exemple, si on veut tracer $x \mapsto x^2$ entre -1 et -1, on peut taper le script suivant :

dans l'éditeur puis chargé dans le shell

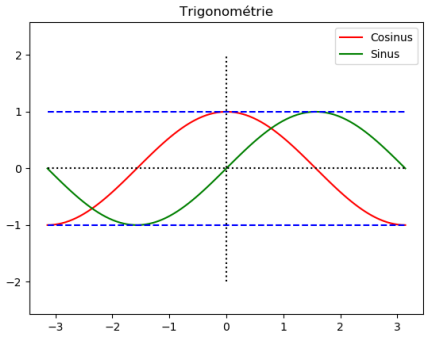
```
close()
x=linspace(-1,1,10)
y=x**2
plot(x,y)
show()
```

En modifiant x, on obtient un tracé plus arrondi. Testez avec `x=linspace(-1,1,100)` par exemple...

Exercice 2

1. Tracer sur l'intervalle $[-\pi, \pi]$ (pi est simplement obtenu l'instruction `pi`) les fonctions cos et sin. On veut que cos soit en rouge, sin en vert, que le repère soit orthonormé, qu'il y ait une légende et un titre.
2. Ajouter en traits discontinus les droites d'équations $y = 1$ et $y = -1$.

Au final, vous devrez avoir quelque chose comme ça :



2.1 Courbes paramétrées

On peut aussi faire varier simultanément x et y pour obtenir des jolis dessins. Essayez l'exemple suivant :

dans l'éditeur puis chargé dans le shell

```
close()
xlim(-1,1); ylim(-2,0.5)
t = arange(0,2 * pi,0.01)
x = (sin(t)) ** 3
y = cos(t) - (cos(t)) ** 4
plot(x,y,"r")
axis('equal')
```

Ce genre de très jolies courbes est malheureusement hors programme en ECG :(

3 Suites

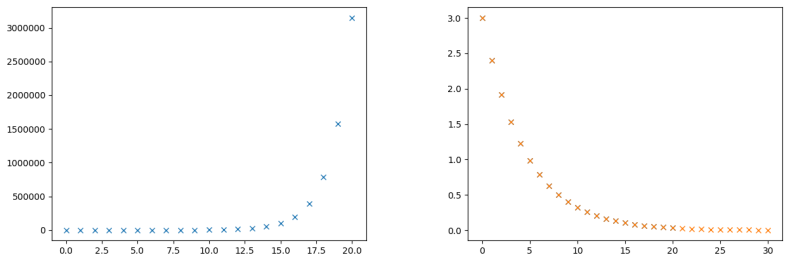
3.1 Tracé de suites

Le tracé des suites suit le même principe que celui des fonctions, avec quelques petites différences :

- La liste des abscisses est constituée d'entiers. On utilisera l'instruction `arange`, similaire à `range`. Ainsi, `x=arange(0,100)` donne une liste des entiers de 0 à 99.
- En général, on ne relie pas les points (on utilisera donc plutôt `plot(x,y,'x')` par exemple)

Exercice 3 Ecrire un programme `geom(a,q,n)` qui trace les n premières valeurs d'une suite géométrique, de raison q avec $u_0 = a$.

Votre programme affichera des choses comme ceci (selon si $|q| > 1$ ou $|q| < 1$) :



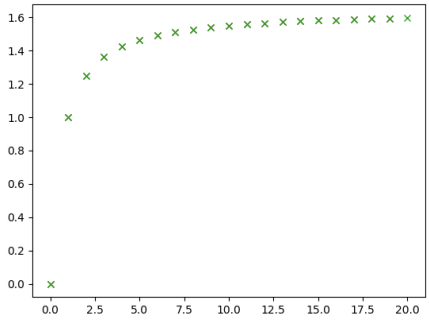
3.2 Illustration de convergence

Exercice 4 Pour tout $n \in \mathbb{N}^\star$, on pose

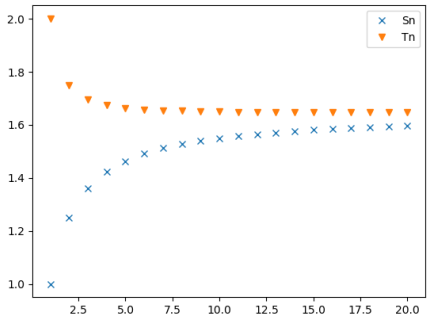
$$S_n = \sum_{k=1}^{n} \frac{1}{k^2} \text{ et } T_n = S_n + \frac{1}{n}$$

1. Ecrire une fonction `sommeSn(n)` qui calcule S_n.

2. Ecrire une fonction `convergence(n)` qui affiche les n premiers termes de S_n. Vous devez obtenir quelque chose comme ceci :



3. Modifier le programme précédent pour qu'il affiche également T_n :



Exercice 5

Soient les suites (u_n) et (v_n) définies par $u_0 = 0$, $v_0 = 1$ et, pour tout $n \in \mathbb{N}$, par :

$$u_{n+1} = \frac{u_n + v_n}{2} \quad \text{et} \quad v_{n+1} = \sqrt{u_{n+1} v_n}$$

On admet que ces suites sont adjacentes et convergent vers $\frac{2}{\pi}$. [1]

1. Illustrez la convergence de ces suites vers $\frac{2}{\pi}$ de la même manière qu'à l'exercice précédent.
2. Utilisez ces suites pour écrire une fonction qui calcule π à 10^{-p} près et qui affiche le nombre d'étapes nécessaires à l'obtention de ce résultat.

Exercice 6 (Entraînement) Pour les suites suivantes, déterminez d'abord sur une feuille la limite, puis illustrez la convergence vers cette limite avec Python :

1. $\forall n \in \mathbb{N}, u_n = (-1)^n + 2$;
2. $\forall n \in \mathbb{N}, u_n = \dfrac{\exp(n^2 + 2n - 3) + 1}{\exp(n^2 + n + 5)}$;
3. $u_0 = 1$ et, pour tout entier $n \in \mathbb{N}$, $u_{n+1} = \dfrac{3}{4} u_n + 3$;
4. $u_0 = 1$, $u_1 = 0$ et pour tout entier n positif on a $3u_{n+2} = 8u_{n+1} - 3u_n$.

Exercice 7 (Vos premières séries : la série géométrique et la série géométrique dérivée !) Soit $(u_n)_{n\in\mathbb{N}}$ définie par : $\forall n \in \mathbb{N}, u_n = 2^{-n}$.

1. Écrire un programme qui prend comme entrée un entier n et retourne $\displaystyle\sum_{k=0}^{n} u_k$.

1. Cette méthode est attribuée à un cardinal allemand du XVe siècle, Nicolas Krebs. Ce n'est pas la méthode la plus rapide pour avoir une valeur approchée de π, mais elle est déjà très rapide et a le mérite d'être très facile à programmer.

2. Illustrer la convergence de la suite $(S_n)_{n\in\mathbb{N}}$ où $S_n = \sum_{k=0}^{n} u_k$, conjecturer la limite de la suite.
3. Retrouver ce résultat sans ordinateur !
4. Entraînement :Enregistrer vos programmes précédents, les effacer, puis faire les mêmes questions où cette fois $\forall n \in \mathbb{N}, u_n = n2^{1-n}$.

Remarque : cette étude de la somme des termes d'une suite, nous la reverrons en détail pendant l'année, elle est très importante en maths et sert entre autres en probas.