Méthodes numériques concernant les fonctions

I/ Représentations graphiques

1. Méthode générale

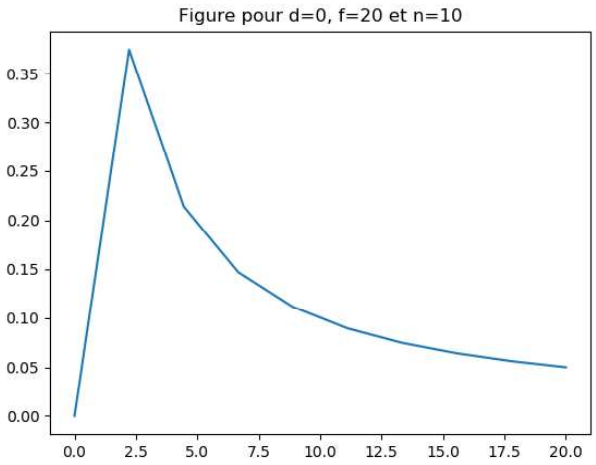
Pour tracer la représentation graphique d'une fonction, la méthode est fondamentalement la même que pour tracer la représentation graphique d'une suite ; il faut créer une liste d'abscisses, créer une liste d'ordonnées et faire afficher à l'aide de la bibliothèque **matplotlib.pyplot**. Si chacune des deux listes contient un très grand nombre de points et qu'on relie ces points, on obtiendra l'apparence d'une courbe.

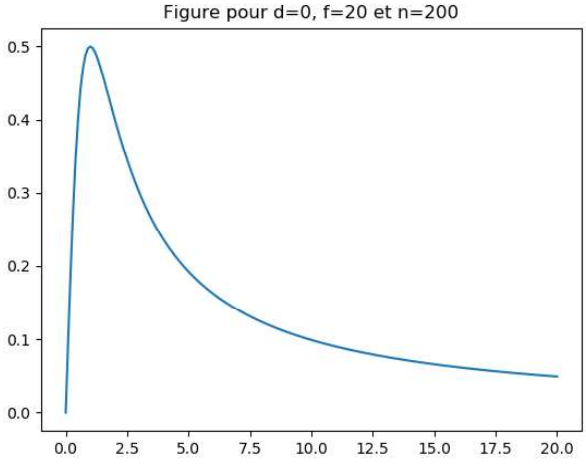
Pour créer la liste des abscisses, on utilise la fonction **linspace** de la bibliothèque **numpy**. Cette fonction prend en argument trois entiers **d**, **f** et **n** et renvoie un tableau unidimensionnel de première valeur **d**, de dernière valeur **f** et qui contient **n** réels répartis régulièrement entre **d** et **f**.

2. Exemple

On souhaite tracer la représentation graphique de la fonction $f : x \mapsto \dfrac{x}{x^2+1}$ sur l'intervalle $[d, f]$ avec une précision de n points. Compléter le programme suivant dans ce but :

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

def fonction(x):
    return x/(x**2+1)

def graphique(d,f,n):

    X = ...

    Y = ...

    plt.plot(X,Y)
    plt.title("Figure pour d="+str(d)+", f="+str(f)+" et n="+str(n))
    plt.show()
```





II/ Résolution d'équations

1. Principe général : recherche par dichotomie

On considère une fonction f continue et strictement monotone sur un intervalle $[a, b]$ ($a, b \in \mathbb{R}$ avec $a < b$).

On suppose que f s'annule exactement une fois sur $[a, b]$ en un réel que l'on note x_0 et on souhaite obtenir un encadrement de x_0.

On définit les deux suites (a_n) et (b_n) de la façon suivante :

→ $a_0 = a$ et $b_0 = b$.

→ Pour tout entier naturel n, on note $c_n = \dfrac{a_n + b_n}{2}$ et :

$$(\, a_{n+1}\,,\, b_{n+1}\,) = \begin{cases} (\, a_n\,,\, c_n\,) & \text{si } f(a_n)\, f(c_n) \leqslant 0 \\ (\, c_n\,,\, b_n\,) & \text{sinon} \end{cases}$$

Remarque : on peut également utiliser une recherche par dichotomie sous les hypothèses suivantes :

→ f est continue strictement monotone sur $[a, b]$

→ $f(a)\, f(b) < 0$

2. Programmation en Python

```
def dichotomie_fonction(f,a,b,e) :
    while (b-a) >= e :
        c = (a+b)/2
        if f(a)*f(c) <= 0 :
            b = c
        else :
            a = c
    return (a,b)
```

Explications concernant les arguments :

→ Pour indiquer à Python à quelle fonction f on fait référence, on utilise la commande **lambda**.

→ a et b désignent les valeurs de a_0 et b_0.

→ e désigne la précision de la recherche.

III/ Calculs d'intégrales

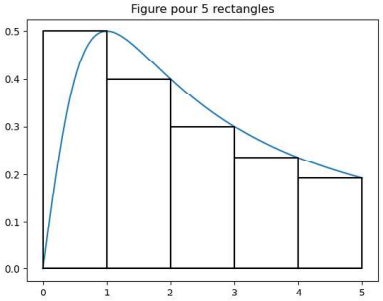
1. Principe général : méthode des rectangles

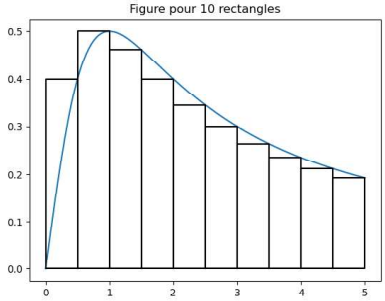
On considère une fonction f continue et positive sur un intervalle $[a, b]$ ($a, b \in \mathbb{R}$ avec $a < b$) et on souhaite obtenir une valeur approchée de $\int_a^b f(t)\,dt$.

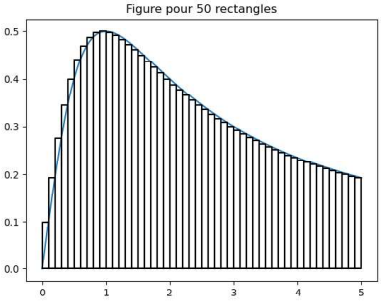
Pour cela, on approche la courbe de f par des rectangles.

On traite l'exemple de la fonction $f : x \mapsto \dfrac{x}{x^2+1}$ sur l'intervalle $[0, 5]$.

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

def fonction(x):
    return x/(x**2+1)

def illustration(a,b,n,N):
    Xf = np.linspace(a,b,n)
    Yf = [fonction(x) for x in Xf]
    plt.plot(Xf,Yf)
    Xr = np.linspace(a,b,N+1)
    for i in range(N):
        plt.plot([Xr[i],Xr[i+1],Xr[i+1],Xr[i],Xr[i]],
        [0,0,fonction(Xr[i+1]),fonction(Xr[i+1]),0],'k')
    plt.title("Figure pour "+str(N)+" rectangles")
    plt.show()
```







Mathématiquement, cela revient à approcher la valeur de l'intégrale $\int_a^b f(t)\,dt$ par la somme :

2. Programmation en Python

La programmation sera faite en TP. Le but sera d'écrire une fonction Python qui prend en arguments a *et* b *et qui renvoie une valeur approchée de* $\int_a^b f(t)\,dt$.