TP 08 – RÉSOLUTION NUMÉRIQUE DE $f(x) = 0$

I Résolution num. de $f(x) = 0$ par dichotomie

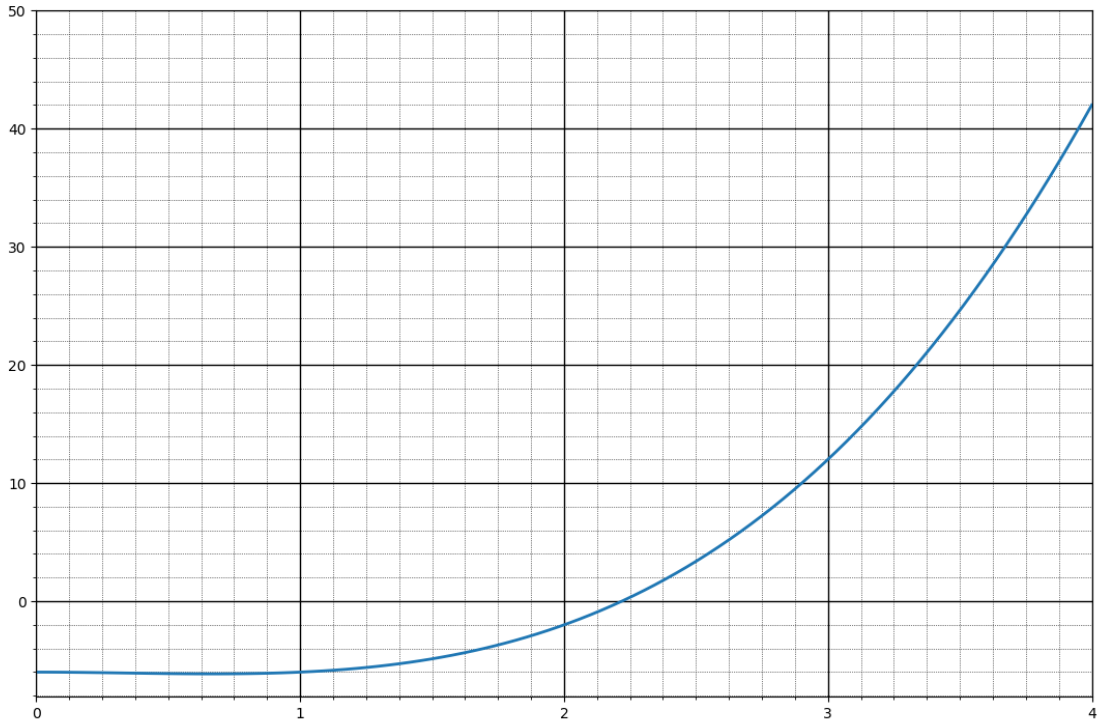
On considère une fonction $f : [a, b] \to \mathbb{R}$ continue et strictement monotone avec $f(a)$ et $f(b)$ de signes contraires. Par théorème de la bijection, on sait alors que l'équation $f(x) = 0$ admet une unique solution $\alpha \in [a, b]$. Cherchons une valeur approchée par défaut de la solution α. On cherche donc un intervalle de longueur ε (ou moins) qui contient la solution α et dans ce cas, la valeur approchée est la borne inférieure de cet intervalle. Expliquons comment procéder par dichotomie.

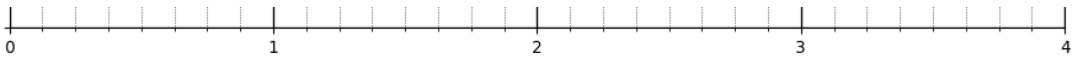
Cas particulier : $f(a) < 0$ et que $f(b) > 0$.

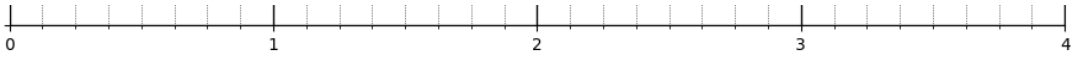
```
Données : f strictement croissante sur [a,b]
          avec f(a)<0 et f(b)>0
          epsilon > 0 (marge d'erreur)
Sortie : valeur approchée par défaut à epsilon près de f(x)=0

a = début de l'intervalle, b = fin de l'intervalle
Tant que |b-a| > epsilon
    m = milieu de [a,b]
    Si  f(m) > 0
            On actualise a ou b
        pour chercher dans la première moitié de l'intervalle
        a = a
        b = m
    Sinon (si f(m) < 0)
            On actualise a ou b
        pour chercher dans la seconde moitié de l'intervalle
        a = m
        b = b
A la fin, on a |b-a| <= epsilon.
On renvoie a qui est une valeur approchée de f(x)=0
```

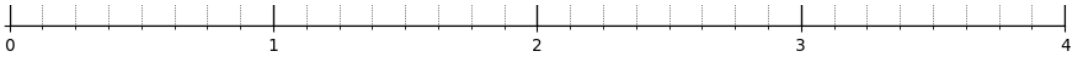
Cas général : c'est-à-dire où $f(a)$ et $f(b)$ sont seulement de signe contraire

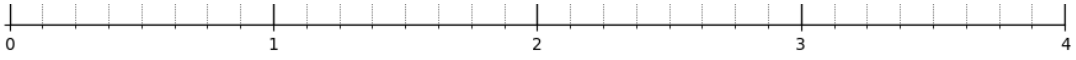
```
Données : f strictement monotone sur [a,b]
          avec f(a) et f(b) de signes contraires
          epsilon > 0 (marge d'erreur)
Sortie : valeur approchée par défaut à epsilon près de f(x)=0

a = début de l'intervalle, b = fin de l'intervalle
Tant que |b-a| > epsilon
    m = milieu de [a,b]
    Si f(a) et f(m) sont de signes contraires
            On actualise a ou b
        pour chercher dans la première moitié de l'intervalle
        a = a
        b = m
    Sinon
            On actualise a ou b
        pour chercher dans la seconde moitié de l'intervalle
        a = m
        b = a
A la fin, on a |b-a| <= epsilon.
On renvoie a qui est une valeur approchée de f(x)=0
```

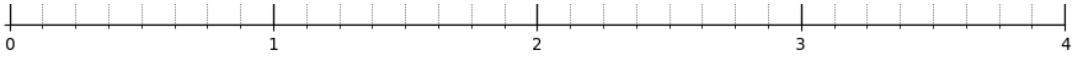

50
40
30
20
10
0
0
1
2
3
4

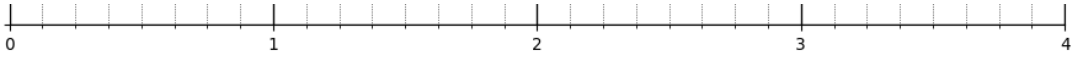

0
1
2
3
4


0
1
2
3
4


0
1
2
3
4


0
1
2
3
4


0
1
2
3
4


0
1
2
3
4

a	b	m	f(m)	Conséquence
0	4	2	f(2) <= 0	On cherche dans la seconde moitié
2	4	3	f(3) >= 0	On cherche dans la première moitié
2	3	2,5	f(2,5) >= 0	On cherche dans la première moitié
2	2,5	2,25	f(2,25) >= 0	On cherche dans la première moitié
2	2,25	2,125	f(2,125) <= 0	On cherche dans la seconde moitié

Finalement, on sait que

$$2,125 \leqslant \alpha \leqslant 2,5.$$

① Écrire un fonction `resolutionpart(f,a,b,epsilon)` qui renvoie une valeur approchée (par défaut, à *epsilon* près) de la solution de l'équation $f(x) = 0$ sur $[a, b]$, en suivant l'algorithme dichotomique du cours dans le cas particulier où f est strictement croissante, $f(a) < 0$ et $f(b) > 0$.

Entrée [1]:
```
def resolutionpart(f,a,b,epsilon):
    while b-a > epsilon:
        m = (a+b)/2
        if f(m)>0 :
            b=m
        else:
            a=m
    return a
```

Tester la résolution avec l'équation $x - 3 = 0$ sur $[1, 6]$ avec $\varepsilon = 10^{-2}$.

Entrée [2]:
```
def f(x):
    return x-3

resolutionpart(f,1,6,0.01)
```

Out [2]:
```
2.9921875
```

② Écrire un fonction `resolutionpart(f,a,b,epsilon)` qui renvoie une valeur approchée (par défaut, à *epsilon* près) de la solution de l'équation $f(x) = 0$ sur $[a, b]$, en suivant l'algorithme dichotomique du cours dans le cas général

Entrée [3]:
```
def resolution(f,a,b,epsilon):
    while b-a > epsilon:
        m = (a+b)/2
        if f(a)*f(m)<0 :
            (a,b) = (a,m)
        else:
            (a,b) = (m,b)
    return a
```

Tester la résolution avec l'équation $x - 3 = 0$ sur $[1, 6]$ avec $\varepsilon = 10^{-2}$.

Entrée [4]:
```
def f(x):
    return x-3

resolution(f,1,6,0.01)
```

Out [4]:
```
2.9921875
```

Exercice 1

1. Définir une fonction appelée `f` et renvoyant $f(x) = x^3 + 2x^2 + 3x + 4$ puis tracer sa courbe sur $[-10, 10]$. *On peut facilement montrer (exercice) que l'équation $f(x) = 0$ admet une unique solution dans $\mathbb{R}$. On cherche une valeur approchée de celle-ci.*

Entrée [5]:
```
def f(x):
    return x**3+2*x**2+3*x+4

import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10,10,100)
y = f(x)
plt.plot(x,y)
plt.grid() # pour afficher une grille, cela facilite la lecture
plt.show()
```

2. À l'aide du graphique, trouver un intervalle $[a, b]$ avec $f(a)$ et $f(b)$ de signes contraires.

   L'intervalle $[-3, 1]$ semble convenir.

3. Utiliser la fonction `resolution` pour trouver une valeur approchée de la solution de l'équation $f(x) = 0$ avec $\varepsilon = 10^{-3}$.

Entrée [6]:
```
resolution(f,-3,2,0.001)
```

Out [6]:
```
-1.651123046875
```

Exercice 2 En suivant la même méthode que dans l'exercice précédent, donner une valeur approchée de chacune des solutions de l'équation $x - \ln(x) = 2$.

Entrée [7]:
```
def f(x):
    return x-np.log(x)-2

x = np.linspace(0.1,10,100)
y = f(x)
plt.plot(x,y)
plt.grid() # pour afficher une grille, cela facilite la lecture
plt.show()
```

Entrée [8]:
```
resolution(f,0.1,1,0.001)
```

Out [8]:
```
0.1580078125
```

Entrée [9]:
```
resolution(f,1,4,0.001)
```

Out [9]:
```
3.14599609375
```

II Résolution num. de $f(x) = 0$ via des suites récurrentes

Dans cette section, on souhaite trouver une valeur approchée de l'unique solution de l'équation $x^2 - 2 = 0$ sur $[0, +\infty[$ (attention, cette équation admet deux solutions sur $\mathbb{R}$). Autrement dit, on souhaite calculer une valeur approchée du réel $\sqrt{2}$.

Pour cela, on introduit la fonction h définie sur $\mathbb{R}^*$ par

$$\forall x \in \mathbb{R}^*, \qquad h(x) = \frac{1}{3}\left(2x + \frac{2}{x}\right)$$

1. Tout d'abord, on peut calculer que
$$h(\sqrt{2}) = \sqrt{2}.$$
2. De plus, la fonction h vérifie les propriétés suivantes
   - La fonction h est dérivable sur $[1, +\infty[$.
   - Il existe $k = \frac{2}{3} > 0$ tel que, pour tout $x \in [1, +\infty[$, $|h'(x)| \leqslant k$.

   Donc par inégalité des accroissements finis,
$$\forall a, b \in [1, +\infty[, \qquad |f(b) - f(a)| \leqslant \frac{2}{3}|b - a|$$
   En particulier,
$$\forall x \in [1, +\infty[, \qquad |f(x) - \sqrt{2}| \leqslant \frac{2}{3}|x - \sqrt{2}|$$

De plus, on définit la suite $(u_n)_{n\in\mathbb{N}}$ par $u_0 = 2$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = h(u_n)$.

1. On peut montrer par récurrence que, pour tout $n \in \mathbb{N}$, $u_n \geqslant 1$.
2. Donc, en utilisant l'égalité démontrée sur h, on obtient que
$$\forall n \in \mathbb{N}, \qquad |u_{n+1} - \sqrt{2}| \leqslant \frac{2}{3}|u_n - \sqrt{2}|$$
3. On peut alors en déduire par récurrence que
$$\forall n \in \mathbb{N}, \qquad |u_n - \sqrt{2}| \leqslant \left(\frac{2}{3}\right)^n$$
4. On en déduit que la suite $(u_n)_{n\in\mathbb{N}}$ converge vers $\sqrt{2}$.

① Écrire un programme qui permette d'afficher les 11 premiers termes de la suite (u_n) (donc de u_0 à u_{10}). Vérifier que (u_n) semble tendre vers $\sqrt{2}$.

Entrée [10]:
```
u=2
    for k in range(10):
        u = 1/3*(2*u+2/u)
        print(u)
```

② Écrire un programme qui affiche un entier $n \in \mathbb{N}$ tel que u_n soit une valeur approchée de $\sqrt{2}$ à $\varepsilon = 10^{-4}$ près. On affichera également la valeur de u_n correspondante.

Entrée [11]:
```
n=0
u=2
while (2/3)**n > 10**(-4):
    n = n+1
    u = 1/3*(2*u+2/u)
print('valeur approchée =', u)
```

Out [11]:
```
valeur approchée = 1.4142135623833116
```