

Dichotomie

I/ Recherche par dichotomie d'un zéro d'une fonction continue et strictement monotone sur un intervalle

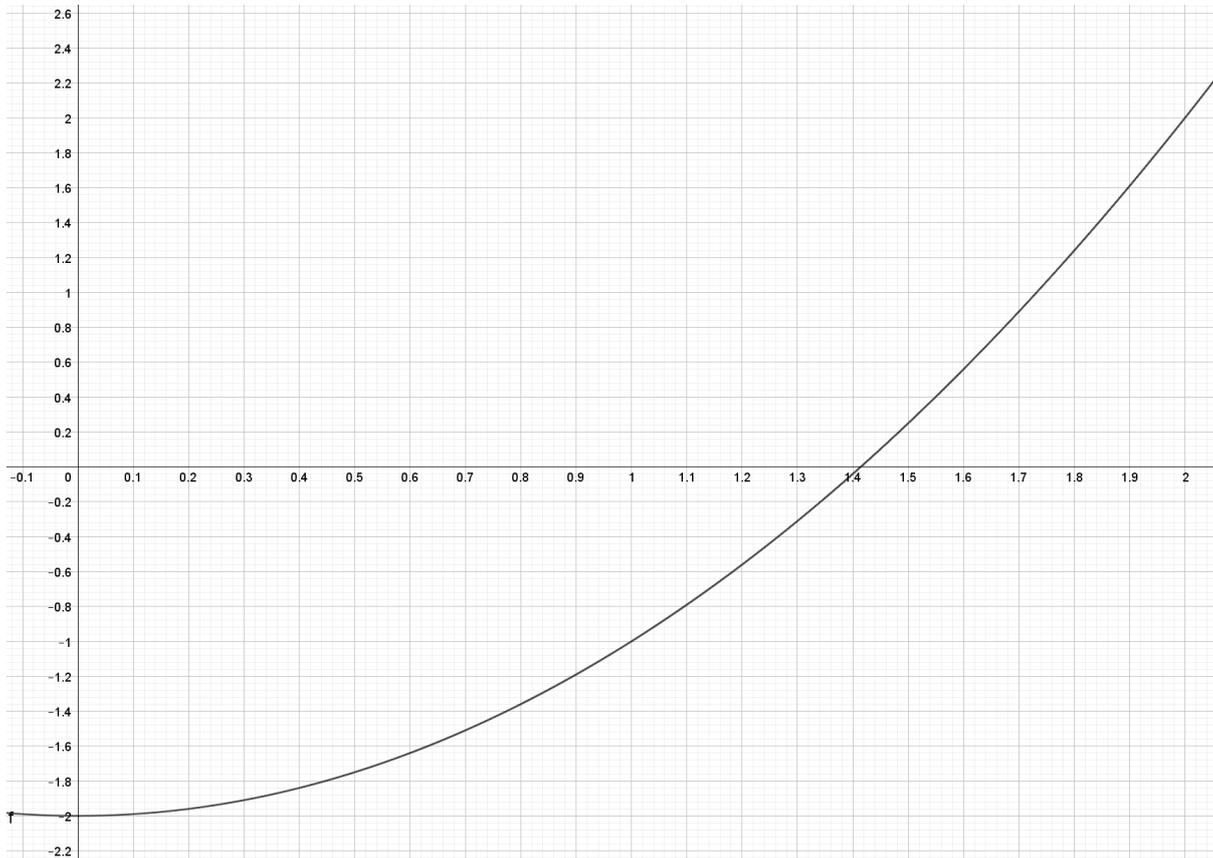
- Illustration 1

On considère la fonction $f : x \mapsto x^2 - 2$ sur l'intervalle $I = [0, 2]$.

Sur l'intervalle I , f est continue et strictement croissante.

De plus, $f(0) = -2 < 0$ et $f(2) = 2 > 0$.

On peut donc démontrer qu'il existe un unique $x_0 \in [0, 2]$ tel que $f(x_0) = 0$ (voir chapitre "Limites et continuité").



- Illustration 2

On considère la fonction $f : x \mapsto x^2 - 2$ sur l'intervalle $I = [-2, 0]$.

Sur l'intervalle I , f est continue et strictement décroissante.

De plus, $f(-2) = 2 > 0$ et $f(0) = -2 < 0$.

On peut donc démontrer qu'il existe un unique $x_0 \in [-2, 0]$ tel que $f(x_0) = 0$ (voir chapitre "Limites et continuité").



- Principe général

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 .

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 $c_n = \frac{a_n + b_n}{2}$ et :

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

- Programmation en Python

```

1 def dichotomie_fonction(f, a, b, e) :
2     while (b-a) >= e :
3         c = (a+b)/2
4         if f(a)*f(c) <= 0 :
5             b = c
6         else :
7             a = c
8     return (a, b)

```

En se plaçant avec une précision $e = 10^{-5}$:

→ Pour l'illustration 1, on utilise l'instruction : `dichotomie_fonction(lambda x :x**2-2 , 0 , 2 , 10**(-5))`

→ Pour l'illustration 2, on utilise l'instruction : `dichotomie_fonction(lambda x :x**2-2 , -2 , 0 , 10**(-5))`

II/ Recherche par dichotomie dans une liste

- Principe général

On considère une liste de nombres **L**.

On suppose que les éléments de **L** sont triés par ordre croissant.

Pour chercher si un nombre **x** est présent dans la liste, on procède de la façon suivante :

- On compare **x** à l'élément "au milieu" de **L**, qu'on note **L[m]**.
- Si c'est **x**, on s'arrête.
- Si $x > L[m]$, on continue la recherche dans la fin de la liste.
- Si $x < L[m]$, on continue la recherche dans le début de la liste.
- On s'arrête soit quand on a trouvé **x** soit quand on a épuisé le champ de recherche.

- Programmation en Python

```
1 def recherche_dichotomie(L, x):
2     a = 0
3     b = len(L)-1
4     while a <= b:
5         m = (a+b) // 2
6         if L[m] == x:
7             return True
8         elif L[m] < x :
9             a = m + 1
10        else :
11            b = m - 1
12    return False
```