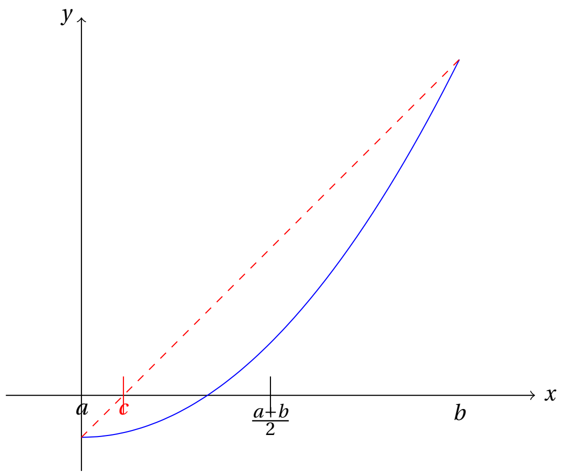
TD informatique du chapitre 8 : Algorithmes dichotomiques

Exercice 1 : Méthode de la fausse position

Soient $(a,b) \in \mathbb{R}^2$, avec $a < b$ et $f \in \mathcal{C}^0([a,b],\mathbb{R})$ telle que $f(a)f(b) \leq 0$.
On cherche une valeur approchée d'un zéro de f sur $[a,b]$.
Le principe de la méthode de dichotomie consiste à couper en deux parties égales d'intervalle. Cependant, on peut penser que si $|f(a)|$ est plus petite que $|f(b)|$ le zéro recherché est plus probablement proche de a que de b. Ainsi, plutôt que de comparer avec le milieu de l'intervalle $[a,b]$, on va comparer avec le point d'intersection de la corde reliant les points de f d'abscisse a et b avec l'axe des abscisses.



L'équation de la corde reliant les points de f d'abscisse a et b est :

$$y = \frac{f(b)-f(a)}{b-a}(x-a) + f(a).$$

Ainsi :

$$0 = \frac{f(b)-f(a)}{b-a}(c-a) + f(a) \Leftrightarrow c = a - \frac{f(a)(b-a)}{f(b)-f(a)} = \frac{af(b)-bf(a)}{f(b)-f(a)}.$$

On définit donc les suites suivantes :

$$x_0 = a,\ y_0 = b$$

$$\forall n \in \mathbb{N},\ (x_{n+1}, y_{n+1}) = \begin{cases} \left(x_n, \frac{x_n f(y_n) - y_n f(x_n)}{f(y_n)-f(x_n)}\right) & \text{si } f(x_n) f\left(\frac{x_n f(y_n) - y_n f(x_n)}{f(y_n)-f(x_n)}\right) \leq 0 \\ \left(\frac{x_n f(y_n) - y_n f(x_n)}{f(y_n)-f(x_n)}, y_n\right) & \text{sinon} \end{cases}$$

1. Ecrire une fonction `FaussePosition` qui prend comme arguments une fonction f, deux réels a et b et un réel strictement positif ε et qui renvoie, en utilisant la méthode de la fausse position, une liste $[x,y]$ telle que f admette au moins un zéro dans $[x,y]$ et $|x-y| \leq \varepsilon$.
   Tester cette fonction pour rechercher un zéro de $x \mapsto x^2 - \cos(x)$ sur $\left[\frac{1}{2}, 1\right]$ avec une précision de 10^{-5}.
2. (a) Modifier les fonctions `dichotomie` et `FaussePosition` afin qu'elles renvoient également le nombre d'itérations effectuées.
   (b) Comparer le nombre d'itérations effectuées pour la recherche d'un zéro de $x \mapsto x^2 - \cos(x)$ sur $\left[\frac{1}{2}, 1\right]$ avec une précision de 10^{-15} par les méthodes de dichotomie et de fausse position.

Exercice 2 : Insertion dans une liste triée

Ecrire une fonction dichotomique `Insertion` qui prend comme arguments une liste `l` de flottants triés par ordre croissant et un flottant `x` et qui renvoie une liste `L` constituée de la liste `l` dans laquelle est inséré l'élément `x` en conservant l'ordre croissant.
Par exemple : `Insertion([1,2,8,9,12],5)` renvoie `[1,2,5,8,9,12]` et `Insertion([1,2,8,9,12],8)` renvoie `[1,2,8,8,9,12]`.

Exercice 3 : Jeu contre l'ordinateur

Ecrire une fonction `Jeu` qui prend comme argument un entier naturel N et qui consiste à faire deviner par l'ordinateur un nombre entier compris entre 0 et N en utilisant une méthode de dichotomie. Pour cela, l'utilisateur doit penser à un entier compris entre 0 et N et la fonction doit :

- proposer une valeur,
- demander à l'utilisateur de taper :
  - + si la réponse est plus grande que la proposition,
  - - si la réponse est plus petite que la proposition,
  - = si la proposition est juste,
- si l'utilisateur répond + ou -, l'ordinateur propose alors une autre valeur,
- si l'utilisateur répond =, la fonction s'arrête et affiche le nombre de coups ayant permis à l'ordinateur de gagner,
- sinon, l'ordinateur affiche "Tu triches" et propose à nouveau la valeur.

Exercice 4 : Exponentiation

On considère un flottant `x` et un entier naturel `n`. Le but de cet exercice est d'écrire une fonction qui renvoie x^n **sans utiliser les commandes de puissances** de Python.

1. En utilisant une seule boucle, écrire une fonction simple qui réponde à la question.
2. On cherche maintenant à améliorer la rapidité de cette fonction en remarquant que :

$$x^n = \begin{cases} x^{n/2}.x^{n/2} & \text{si } n \text{ est pair,} \\ x.x^{(n-1)/2}.x^{(n-1)/2} & \text{si } n \text{ est impair.} \end{cases}$$

En utilisant la notation Python pour la division euclidienne, on a donc :

$$x^n = \begin{cases} x^{n//2}.x^{n//2} & \text{si } n \text{ est pair,} \\ x.x^{n//2}.x^{n//2} & \text{si } n \text{ est impair.} \end{cases}$$

Ecrire une fonction basée sur ce principe qui réponde à la question.