INF 7 - Résolution de l'équation $f(x) = 0$.

1. MÉTHODE DE DICHOTOMIE

Le cadre est le suivant : on a une fonction strictement monotone F sur intervalle $[a, b]$ et on se donne un réel k strictement compris entre $F(a) = F(b)$. On sait par le théorème des valeurs intermédiaires et celui de la bijection que k admet un unique antécédent par F. Pour trouver **numériquement** antécédent on va coder les suites adjacentes qui apparaissent dans la démonstration du TVI.

L'algorithme

- supposera que la fonction F est codée en Python
- prendra en entrée a, b et la précision voulue ϵ (ou le nombre d'itération n).
  Remarque 1.1 — Prendre n ou ϵ en entrée revient au même car on a un lien entre la longueur de l'intervalle à chaque étape et le nombre d'étape :
  $$b_n - a_n \leqslant \epsilon \iff (b-a)2^{-n} \leqslant \epsilon.$$
  Cela signifie que le nombre d'itération sera la partie entière supérieure de
  $$\frac{\ln\left(\frac{b-a}{\epsilon}\right)}{\ln(2)}.$$

Commençons sur un exemple : la fonction strictement croissante sur $\mathbb{R}_+^*$,

$$f : x \mapsto x + \ln(x).$$

```python
def F(x):
    y= np.log(x) +  x##coder ici la fonction de votre choix
    return(y)
```

```python
def dichotomie(a,b,eps):
    u = a
    v = b #initialisation des suites adjacentes)
    while abs(v-u)> eps
            c = (u+v)/2
        if F(c) == 0: #test du cas trivial au milieu, facultatif
                return( (u+v)/2)
        elif F(c) < 0: #choix du demi intervalle
                v = c
        else:
            u = c
    return((u+v)/2)
```

Le cas général donne cette fonctions là :

```python
def dichotomie(a,b,eps):
    u = a
    v = b #initialisation des suites adjacentes
    while abs(a-b)> eps:
        if F((u+v)/2) *  F(u) < 0: #choix du demi intervalle où on \
+ continue
            v = (u+v)/2
        else:
            u = (u+v)/2
    return((u+v)/2)
```

Remarque 1.2 — Ici, le test `F((u+v)/2) * F(u) < 0` vérifie si $F((u+v)/2)$ est d'un signe différent de $F(u)$. Si oui, cela nous dit que la solution est entre u et $(u+v)/2$.

2. AUTRES MÉTHODES

Suites récurrentes. On a vu dans le chapitre sur les suites que si une suite récurrente définie par

$$\forall n \in u_{n+1} = f(u_n)$$

converge, alors c'est vers une solution de le l'équation

$$f(x) = x.$$

Cela permet de fournir une méthode d'approximation des solutions de $f(x) - x = 0$. Un exemple déjà est la méthode de Héron pour le calcul de racine carrée !

Utilisation d'un encadrement. Considérons les suites

$$u_n = \sin\left(\frac{\pi}{3 \times 2^n}\right) \text{ et } v_n = \cos\left(\frac{\pi}{3 \times 2^n}\right).$$

On pose

$$u_n = 9 \times 2^n \times \frac{a_n}{2 + b_n} + b_n \text{ et } v_n = 2^n \left(2a_n + \frac{a_n}{b_n}\right).$$

On peut montrer que pour tout $n \in \mathbf{N}$,,

$$u_n < \pi < v_n$$

et donc que que les deux suites tendent vers π.

1. Pourquoi a-t-on alors

$$v_n - u_n < \varepsilon \Rightarrow |u_n - \pi| < \varepsilon ? .$$

2. En déduire un code qui donne une approximation de π à ε près ?

EXERCICES

Exercice 1 Écrire un programme qui prend une entrée ε et renvoie une approximation de $\sqrt{3}$ à ε près. On utilisera un algorithme de dichotomie basé sur l'étude de la fonction $f : x \mapsto x^2 - 3$.. Sur un graphique, tracer le graphe de la fonction f ainsi que les points de la suite crée par dichotomie de coordonnées $(u_n, f(u_n))$.

Faire de même pour approximer $2^{1/3}$.

Exercice 2 Avec un algorithme de dichotomie, déterminer avec une précision de ε une solution de $\tan(x) = 10$ sur $]-\frac{\pi}{2}, \frac{\pi}{2}[$.

Exercice 3

1. Écrire un algorithme dichotomique qui prend en entrée $x \in \mathbb{R}$ et calcule une valeur approchée de $\text{Arctan}(x)$ à 10^{-6} près.
2. Tracer sur un même graphique le graphe de la fonction Arctan calculé avec fonction et calculée avec `np.arctan`. On tracera les courbes sur $]-5, 5[$.

Exercice 4

1. Montrer que la fonction cos réalise une bijection de $[0, \pi]$ sur un intervalle à déterminer. On l'appellera Arccos.
2. Utiliser la méthode de dichotomie pour programme une fonction qui prend en entrée $x \in [-1, 1]$ et renvoie $\text{Arccos}(x)$ à 10^{-6} près.
3. Tracer la fonction Arrcos sur le même graphique avec
   - la fonction codée précédemment
   - la fonction python `np.arrcos`.