TP2 : Calcul du premier entier tel qu'une condition est vérifiée

(Révisions sur la structure itérative `while`*)*

▶ Dans votre dossier `Info_2a`, créez le dossier `TP_2`.

I. Introduction du problème

Le problème qui nous intéresse ici est le suivant.

Problème.

Données :

- Une suite (u_n) et une suite (d_n) telle que $d_n \underset{n\to+\infty}{\longrightarrow} 0$.
- L'existence d'un réel α tel que : $\forall n \in \mathbb{N},\ |u_n - \alpha| \leqslant d_n$.

But :

1) Déterminer un indice N tel que le terme u_N vérifie : $|u_N - \alpha| \leqslant 10^{-4}$.

2) En déduire une valeur approchée de α à 10^{-4} près.

Remarque

- Les inégalités du type $|u_n - \alpha| \leqslant d_n$ sont fréquentes dans les exercices. On pense notamment à l'utilisation de l'inégalité des accroissements finis pour l'étude des suites du type $u_{n+1} = f(u_n)$.
- La valeur de α n'est pas forcément connue précisément. C'est par exemple le cas lorsque α est fourni par le théorème de la bijection : on sait alors dans quel intervalle se situe α mais on ne connaît pas sa valeur exacte.
- Comme $|u_n - \alpha| \leqslant d_n$ et $d_n \underset{n\to+\infty}{\longrightarrow} 0$, on en conclut, à l'aide du théorème d'encadrement, que $|u_n - \alpha| \underset{n\to+\infty}{\longrightarrow} 0$ et donc que (u_n) est convergente, de limite α. Ceci démontre que l'élément N du point ***1)*** existe bien et que l'inégalité $|u_N - \alpha| \leqslant 10^{-4}$ est vérifiée à partir d'un certain rang.
- L'idée de base pour déterminer N est de calculer successivement les termes de (u_n) jusqu'à celui qui vérifie $|u_n - \alpha| \leqslant 10^{-4}$. Cependant, on ne peut procéder de la sorte si la valeur de α n'est pas connue (calcul de $u_n - \alpha$ impossible). On se sert alors de la suite (d_n).
  Si on parvient à déterminer un entier N tel que $d_N \leqslant 10^{-4}$, alors on obtient, par transitivité :

  $$|u_N - \alpha| \ \leqslant\ d_n \ \leqslant\ 10^{-4}$$

  ce qui permet de résoudre le problème.
- Une fois l'entier N précédent déterminé, u_N est une valeur approchée de α.

II. Un exemple classique

On commence par illustrer le problème et sa résolution par l'étude d'une suite de type $u_{n+1} = f(u_n)$ dans le cadre de l'utilisation de l'inégalité des accroissements finis.

On considère la fonction $f : x \mapsto \mathrm{e}^{-\frac{x^2}{2}}$ et on définit la suite (u_n) par : $\left\{ \begin{array}{l} u_0 = \frac{1}{2} \\ \forall n \in \mathbb{N},\ u_{n+1} = f(u_n) \end{array} \right.$

Rappelons les différentes étapes de ce type d'étude. Les démonstrations sont laissées au lecteur.

1) En appliquant le théorème de la bijection à la fonction $g : x \mapsto f(x) - x$, on démontre que l'équation $f(x) = x$ admet une unique solution dans $[0, 1]$, que l'on note α.

2) Après avoir démontré que l'intervalle $[0, 1]$ est stable par f, on en déduit, par récurrence que : $\forall n \in \mathbb{N},\ u_n \in [0, 1]$.

3) a) Par étude de la fonction f', on démontre : $\forall x \in [0, 1],\ |f'(x)| \leqslant \dfrac{1}{\sqrt{\mathrm{e}}}$.

b) On est alors dans le cadre de l'application de l'IAF, qui permet de démontrer que :

$$\forall n \in \mathbb{N},\ |u_{n+1} - \alpha| \leqslant \frac{1}{\sqrt{\mathrm{e}}} |u_n - \alpha|$$

(on démontre en fait que $|f(u_n) - f(\alpha)| \leqslant \dfrac{1}{\sqrt{e}} |u_n - \alpha|$*)*

c) On en déduit que : $\forall n \in \mathbb{N},\ |u_n - \alpha| \leqslant \left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n$.

d) Comme $\left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n \underset{n \to +\infty}{\longrightarrow} 0$, on en déduit que (u_n) est convergente, de limite α.

Le but est alors de calculer une valeur approchée de α à 10^{-4} près.

▶ Quelle condition permet d'assurer que $|u_n - \alpha| \leqslant 10^{-4}$?

▶ Écrire un programme permettant d'afficher le premier entier n tel que $\left(\dfrac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$.

- Compléter le programme précédent afin qu'il affiche une valeur approchée à 10^{-4} près de α.

- Déterminer une formule mathématique donnant le premier entier n tel que $\left(\frac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant 10^{-4}$.

- Comparer la valeur obtenue dans la quesiton précédente et celle affichée par le programme.

- De même, donner la formule permettant d'obtenir le premier entier n tel que $\left(\frac{1}{\sqrt{\mathrm{e}}}\right)^n \leqslant \varepsilon$.

- En déduire une fonction `calcApproch` qui prend en paramètre un réel `eps` et qui calcule une valeur approchée de α à `eps` près à l'aide d'une boucle `for`. Le résultat sera stocké dans une variable `u`.

III. Les exemples aux concours

Il est fréquent de devoir coder des programmes permettant de calculer un entier n / le premier entier n tel qu'une condition est vérifiée. On retrouve ce type d'exercice sous de nombreuses variantes.

III.1. EML 2018

L'épreuve `EML 2018` comportait une étude de suite récurrente d'ordre 1 (je ne peux y croire) (u_n) définie par :

$$\begin{cases} u_0 = 4 \\ \forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n) + 2 \end{cases}$$

On devait démontrer les propriétés suivantes.

- $\forall n \in \mathbb{N},\ u_n \geqslant b$.
- $\forall n \in \mathbb{N},\ u_{n+1} - b \leqslant \dfrac{1}{2}\ (u_n - b)$.
- $\forall n \in \mathbb{N},\ 0 \leqslant u_n - b \leqslant \dfrac{1}{2^{n-1}}$.

▸ Écrire une fonction **Python** d'en-tête `def suite(n)` qui, prenant en argument un entier n de $\mathbb{N}$, renvoie la valeur de u_n.

▸ Recopier et compléter la ligne 3 de la fonction **Python** suivante afin que, prenant en argument un réel `epsilon` strictement positif, elle renvoie une valeur approchée de b à `epsilon` près.

```
1  def valeur_approchee(epsilon) :
2      n = 0
3      while ..... :
4          n = n + 1
5      b = suite(n)
6      return b
```

Commentaire

III.2. EML 2019

L'épreuve `EML 2019` comportait une étude de suite récurrente (ça a dû déstabiliser les candidats !) (u_n) définie par :

$$\begin{cases} u_1 = 1 \\ \forall n \in \mathbb{N}^*,\ u_{n+1} = u_n + \dfrac{1}{n^2\, u_n} = \dfrac{1}{n}\, f(n\, u_n) \end{cases}$$

où $f : t \mapsto t + \dfrac{1}{t}$.

On démontrait les propriétés suivantes.

- (u_n) converge vers $\ell \in [2, 3]$.
- $\forall p \geqslant 2,\ 0 \leqslant \ell - u_p \leqslant \dfrac{1}{p-1}$.

▸ En déduire une fonction **Python** qui renvoie une valeur approchée de ℓ à 10^{-4} près.

Commentaire

III.3. ECRICOME 2018

Pour tout entier naturel n non nul, on pose : $u_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n)$.

On démontrait dans cet exercice que la suite (u_n) était convergente, vers une limite notée $\gamma \in \mathbb{R}$ puis :

$$\forall n \in \mathbb{N}^*,\ |u_n - \gamma| \leqslant \frac{1}{n}$$

- On rappelle que l'instruction `np.floor(x)` renvoie la partie entière d'un réel x et on suppose que la fonction `u` de la question ***1.e)*** a été correctement programmée. Expliquer l'intérêt et le fonctionnement du script ci-dessous :

```
1 eps = float(input("Entrer un réel strictement positif : "))
2 n = int(np.floor(1/eps) + 1)
3 print(u(n))
```

III.4. ECRICOME 2019

Pour tout entier n non nul, on note h_n la fonction définie sur $\mathbb{R}_+^*$ par :

$$\forall x > 0,\quad h_n(x) \ = \ f(x^n, 1) \ = \ x^n + 1 + \frac{1}{x^n}$$

On démontrait les propriétés suivantes.

- Pour tout entier naturel n non nul, la fonction h_n est strictement décroissante sur $]0, 1[$ et strictement croissante sur $[1, +\infty[$.
- Pour tout entier n non nul, l'équation $h_n(x) = 4$ admet exactement deux solutions, notées u_n et v_n et vérifiant : $0 < u_n < 1 < v_n$.

▸ Écrire une fonction **Python** d'en-tête `def h(n,x)` qui renvoie la valeur de $h_n(x)$ lorsqu'on lui fournit un entier naturel n non nul et un réel $x \in \mathbb{R}_+^*$ en entrée.

▸ Compléter la fonction suivante pour qu'elle renvoie une valeur approchée à 10^{-5} près de v_n par la méthode de dichotomie lorsqu'on lui fournit un entier $n \geqslant 1$ en entrée :

```
1  def v(n) :
2      a = 1
3      b = 3
4      while (b-a) > 10**(-5) :
5          c = (a+b) / 2
6          if h(n,c) < 4 :
7              ........
8          else :
9              ........
10     return ..........
```

Commençons par rappeler le cadre de la recherche par dichotomie.

Calcul approché d'un zéro d'une fonction par dichotomie

Données :

× une fonction $f : \mathbb{R} \to \mathbb{R}$,

× un intervalle de recherche $[a, b]$,

× une précision de recherche ε.

Résultat : une valeur approchée à ε près d'un zéro (sur l'intervalle $[a, b]$) de la fonction f.

Autrement dit, une valeur approchée (à ε près) d'un réel $x \in [a, b]$ tel que : $f(x) = 0$.

- La dichotomie est une méthode itérative dont le principe, comme son nom l'indique, est de découper à chaque itération l'intervalle de recherche en deux nouveaux intervalles. L'intervalle de recherche est découpé en son milieu. On obtient deux nouveaux intervalles :

  × un intervalle dans lequel on sait que l'on va trouver un zéro de f.
  Cet intervalle est conservé pour l'itération suivante.

  × un intervalle dans lequel ne se trouve pas forcément un zéro de f.
  Cet intervalle n'est pas conservé dans la suite de l'algorithme.

  La largeur de l'intervalle de recherche est ainsi divisée par 2 à chaque itération.
  On itère jusqu'à obtenir un intervalle I contenant un zéro de f et de largeur plus faible que ε.
  Les points de cet intervalle I sont tous de bonnes approximations du zéro contenu dans I.

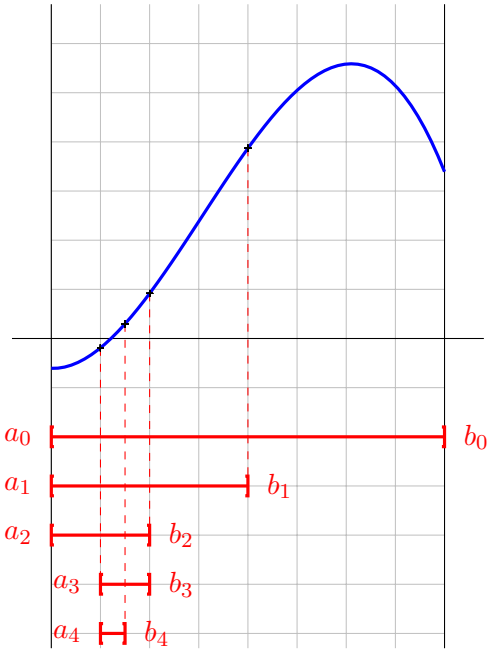
- C'est le **théorème des valeurs intermédiaires** qui permet de choisir l'intervalle qu'il faut garder à chaque étape. Rappelons son énoncé et précisons maintenant l'algorithme :

Théorème des Valeurs Intermédiaires

Soit $f : [a, b] \to \mathbb{R}$ continue sur l'**intervalle** $[a, b]$.

Supposons : $f(a)\ f(b) \leqslant 0$.

Alors il existe $c \in [a, b]$ tel que $f(c) = 0$.



Calcul des suites (a_m), (b_m), (c_m)

Cas $f(a) \leqslant 0$ et $f(b) \geqslant 0$

- Initialement, $a_0 = a$, $b_0 = b$
- À chaque tour de boucle (tant que $b_m - a_m > \varepsilon$) :
  - × $c_m = \dfrac{a_m + b_m}{2}$ (point milieu de $[a_m, b_m]$)
    - × si $f(c_m) < 0$ alors :
      - ∗ $a_{m+1} = c_m$
      - ∗ $b_{m+1} = b_m$
    - × si $f(c_m) \geqslant 0$ alors :
      - ∗ $a_{m+1} = a_m$
      - ∗ $b_{m+1} = c_m$

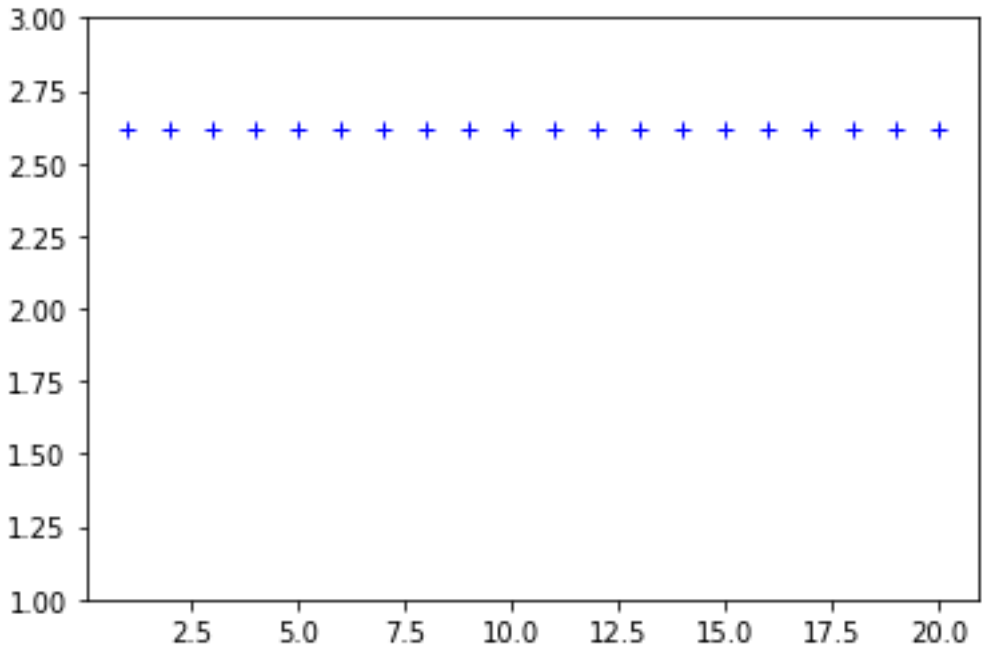
- On construit ainsi une suite $\left([a_m, b_m]\right)_{m\in\mathbb{N}}$ de segments emboîtés :
  - × contenant tous un zéro de f,
  - × dont la largeur est divisée par deux d'un rang au suivant.

Commentaire

- À la suite de la fonction `v`, on écrit le code suivant :

```
1  import matplotlib.pyplot as plt
2  X = [k for k in range(1,21)]
3  Y = []
4  for k in range(1,21) :
5      Y.append( v(k)**k )
6  plt.plot(X, Y, ‘‘b+‘‘)
7  ply.ylim(1, 3)
```

À l'exécution du programme, on obtient la sortie graphique suivante :



Expliquer ce qui est affiché sur le graphique ci-dessus.
Que peut-on conjecturer ?

III.5. ESSEC-I 2020

Cet énoncé introduisait, pour tout $n \in \mathbb{N}^*$ un événement G_n et faisait démontrer les résultats suivants :

$$\mathbb{P}(G_1) \;=\; \frac{p}{q} - \left(1 - \frac{p}{q}\right) pq \qquad \text{et} \qquad \mathbb{P}(G_{n+1}) \;=\; \mathbb{P}(G_n) - \left(1 - \frac{p}{q}\right)\; (pq)^{n+1}\; \binom{2n+1}{n+1}$$

où $q = 1 - p$.
Après avoir codé une fonction `CoeffBin` permettant de calculer les coefficients binomiaux (voir TP1), l'énoncé demandait d'écrire un script **Python** qui déterminer n_p, le plus petit entier n tel que $\mathbb{P}(G_n) \leqslant \varepsilon$ pour $p < \dfrac{1}{2}$ et $\varepsilon > 0$ saisis au clavier par l'utilisateur.