Python - Cours 3
Recherche de valeur approchée de solution d'équations

L'objectif de ce cours est d'étudier trois algorithmes de résolution d'équations du type $f(x) = 0$.

Considérons la fonction $f : x \longmapsto e^{-x} - x$, définie sur $[0;1]$. Démontrer que l'équation $f(x) = 0$ possède une unique solution, notée α, et que cette solution appartient à $[0;1]$.

Pour info...

La solution de l'équation $e^{-x} - x = 0$ est parfois appelée **constante Ω**, et est l'évaluation en 1 de la fonction de Lambert (ou parfois nommée fonction Ω), qui est la réciproque de $x \longmapsto xe^x$ (bijective de $[-1;+\infty[$ dans $[-e^{-1};+\infty[$... Il est impossible de résoudre algébriquement l'équation $e^{-x} - x = 0$ et impossible d'exprimer la constante Ω avec les fonctions usuelles. La seule chose possible est d'en obtenir une valeur approchée par des méthodes numériques !

Un peu d'histoire

Jean-Henri Lambert (1728-1777, suisse-allemand, presque français) est peu connu du grand public, même si on lui doit la première démonstration de l'irrationalité de π ; mais également d'importants travaux en géométrie, puisqu'il est l'inventeur de plusieurs systèmes de projection cartographique, donc certains sont encore utilisés...

I Par balayage

La méthode consiste à partir d'un réel a et d'incrémenter un pas constant à a jusqu'à observer un changement de signe.

A l'aide d'un algorithme de balayage, déterminer un encadrement de α d'amplitude inférieure ou égale à 10^{-6}.

Petite remarque

On pourra ajouter un compteur d'étapes...

```
import numpy as np

def f(x):
    y=np.exp(-x)-x
    return y

def valeurapp(a,b,p):
    A=a+p
    while f(a)*f(A)>0:
        a=a+p
        A=A+p
    return (a,A)
```

II Par dichotomie

On rappelle le principe de la méthode ci-dessous :

On construit deux suites (a_n) et (b_n) telles que $a_0 = a$ et $b_0 = b$ et pour tout $n \in \mathbb{N}$:

- si $f(a_n)f\left(\dfrac{a_n+b_n}{2}\right) < 0$, alors on pose $a_{n+1} = a_n$ et $b_{n+1} = \dfrac{a_n+b_n}{2}$;
- si $f(a_n)f\left(\dfrac{a_n+b_n}{2}\right) > 0$, alors on pose $a_{n+1} = \dfrac{a_n+b_n}{2}$ et $b_{n+1} = b_n$;
- si $f(a_n)f\left(\dfrac{a_n+b_n}{2}\right) = 0$, alors on pose $a_{n+1} = b_{n+1} = \dfrac{a_n+b_n}{2}$.

Rappel...

Dans le chapitre 13, nous avons démontré que la méthode de dichotomie permettait de construire deux suites adjacentes qui convergent vers une solution de l'équation $f(x) = 0$.

Petite remarque

On pourra ajouter un compteur d'étapes...

A l'aide d'un algorithme de dichotomie, déterminer un encadrement de α d'amplitude inférieure ou égale à 10^{-6}.

```python
import numpy as np

def f(x):
    y=np.exp(-x)-x
    return y

def dicho(a,b,p):
    while b-a>p:
        m=(a+b)/2
        if f(m)==0:
            a,b=m,m
        elif f(m)*f(a)<0:
            b=m
        elif f(m)*f(a)>0:
            a=m
    return (a,b)
```

III Avec une suite récurrente !

L'idée ici est de poser $g : x \longmapsto f(x)+x$, de sorte que $f(x)=0 \iff g(x)=x$.
Donc, plutôt que de chercher un zéro de f, on cherche un point fixe de g... Et on considère ainsi la suite (u_n) définie par :

$$u_0 \in [0;1] \ ; \ \forall n \in \mathbb{N},\ u_{n+1}=g(u_n)$$

On sait que si la suite (u_n) converge vers un réel ℓ, alors on obtient $\ell = g(\ell)$: ℓ est un point fixe de g.
Par conséquent, si (u_n) est convergente, alors u_n fournit une valeur approchée d'un point fixe de g pour n suffisamment grand.
De façon générale, deux problèmes pour cette méthode :

1. la méthode ne permet pas toujours de fournir une valeur approchée d'un point fixe ; en effet, il est possible que la suite (u_n) diverge...
2. si la suite (u_n) converge, on ne sait pas toujours étudier sa vitesse de convergence...

☞ Pour info...
En fait, on peut plus ou moins prédire le comportement de (u_n), mais ces résultats sont hors programme en ECG.

Petite remarque
Dans un prochain chapitre, nous verrons un théorème qui pourra être utile dans l'étude de ces suites et qui, dans certains cas, assurera la convergence de la suite (u_n)...

Reprenons toutefois le cas de la fonction $f : x \longmapsto e^{-x}-x$. Considérons donc la fonction $g : x \longmapsto e^{-x}$ et la suite (u_n) définie par :

$$\begin{cases} u_0 = 0 \\ \forall n \in \mathbb{N},\ u_{n+1}=g(u_n) \end{cases}$$

L'exercice 11 du chapitre 13 a permis d'obtenir :

- la suite (u_{2n}) est décroissante, de limite α ;
- la suite (u_{2n+1}) est croissante, de limite α.

En déduire un programme permettant d'obtenir un encadrement de α d'amplitude inférieure ou égale à 10^{-6}.

Petite remarque
On pourra ajouter un compteur d'étapes...

```python
import numpy as np

def g(x):
    y=np.exp(-x)
    return y

def valeuralpha():
    u=0 #u0
    v=1 #u1
    while abs(u-v)>10**(-6):
        u=g(v)
        v=g(u)
    return (u,v)
```

☞ Pour info...
C'est le programme type pour déterminer un encadrement de la limite commune à deux suites adjacentes...