TP N_2 : Méthode de Dichotomie (Bissection)

1 But du TP:

Le problème est de trouver (par la programmation sous Matlab de la méthode de Dichotomie) des valeurs approchées des solutions d'une equation $f(x) = 0$ où f est une fonction non linéaire, le plus souvent continue et dérivable, sur un intervalle I. dans le cas général, en utilisant des methods itératives, qui donnent une suite d'approximations successives s'approchantde la solution exacte.

2.Principales méthodes de résolutions approchées de $f(x) = 0$

On considère un intervalle $[a, b]$ et une fonction f continue de $[a, b]$ dans R.

On suppose que $f(a) * f(b) < 0$ et que l'équation $f(x) = 0$ admet une unique solution α sur l'intervalle $]a, b[$.

2.1.Méthode de Dichotomie

La méthode de Dichotomie consiste à construire une suite (x_n) qui converge vers la racine α de la manière suivante :

-Principe :

on prend pour $x0$ le milieu de l'intervalle $[a, b]$, $(x0 = \frac{a+b}{2})$.

La racine se trouve alors dans l'un des deux intervalles $]a, x0[$ ou $]x_0 , b[$ ou bien elle est égale à $x0$.

1. Si $f(x0) = 0$, c'est la racine de f et le problème est résolu.
2. Si $f(x0) \neq 0$, nous regardons le signe de $f(a) * f(x0)$

a). Si $f(a) * f(x0) < 0$, alors $\alpha \in]a, x0[$

b). Si $f(x0) * f(b) < 0$, alors $\alpha \in]x0, b[$

On recommence le processus en prenant l'intervalle $[a, x0]$ au lieu de $[a, b]$ dans le premier cas, et l'intervalle $[x0, b]$ au lieu de $[a, b]$ dans le second cas.

De cette manière, on construit par récurrence sur n trois suites (an), (b_n) et $(x0_n)$ telles que $a1 = a$, $b1 = b$ et telles que pour tout $n \geq 0$,

- si $f(a) * f(x0) < 0$, alors $\alpha \in]a, x0[$. On pose $a1 = a$, $b1 = x0$.
- si $f(a) * f(x0) = 0$, alors $\alpha = x0$.

- Algorithme de la méthode de Dichotomie

Paramètres d'entrées : f, a, b
f fonction continue sur un intervalle I
deux réels a et b de I tels que $f(a)f(b) < 0$
Paramètres de sorties : x, n
x approximation du zéro de la fonction f
n nombre d'itérations nécessaires pour atteindre x

```
a_0 ← a, b_0 ← b
x_0 ← (a + b)/2
Tant que (x_n > a_n & x_n < b_n) Faire
    Si f(a_n)f(x_n) > 0 alors
        a_{n+1} ← x_n, b_{n+1} ← b_n
        x_{n+1} ← (x_n + b_n)/2
    Sinon
        a_{n+1} ← a_n, b_{n+1} ← x_n
        x_{n+1} ← (x_n + a_n)/2
    Fin Si
Fin Tant que
```

3. Les données:

On travaillera avec les fonctions et intervalles suivants :

$$\begin{cases} f1(x) = x - e^{\sin(x)} \\ [a,b] = [1,10] \end{cases} \ldots\ldots(1)$$

$$\begin{cases} f2(x) = x^3 - 15x^2 - 100x + 89 \\ [a,b] = [0,1] \end{cases} \ldots\ldots(2)$$

$$\begin{cases} f3(x) = 15x^3 - e^{15x} \\ [a,b] = [0,1] \end{cases} \ldots\ldots(3)$$

$$\begin{cases} f3(x) = \cos(x) - 3x + 100 \\ [a,b] = [0,1] \end{cases} \ldots\ldots(4)$$

4. Travail à realizer:

-Programmation de la méthode de Dichotomie :

Écrire les programmes sous MATLAB permettant d'appliquer la méthode en question aux fonctions précédemment dé_nies.

On prendra un test d'arrêt de la forme $|xn+1 - xn| < \epsilon$ et on prendra soin de prévoir un compteur d'itérations qui permettra d'interrompre le traitement dès que *Nmax* d'itérations sont effectuées sans que la précision ϵ ne soit atteinte. On pourra prendre par exemple $Nmax = 100$.

- Les données seront *a, b* et ϵ, *Nmax* et la fonction utilisée ; Les résultats seront la racine obtenue ainsi que son image par la fonction utilisée, le nombre d'itérations effectuées, et l'erreur de calcul.

- Tester sur les fonctions f_1, f_2, f_3 et f_4 pour $\epsilon = 10^{-3}$, 10^{-6}, 10^{-9} et 10^{-12}.

--Programme sous Matlab

On prend l'exemple de la solution d'une fonction : $f(x) = x - e\sin(x)$:

Solutions

```
clc;clear all;
x=1 :0.1 :10 ;
f=inline('x-exp(sin(x))') ;
plot(x,f(x)), grid ;
a=1 ; fa=f(a) ;
b=10 ; fb=f(b) ;
i=0 ; Nmax=20 ;
eps=1.0e-3 ;
if ((fa*fb)<0)
for (i=0 :Nmax)
x=(a+b)/2 ;
i=i+1 ;
if (sign(f(x)) == sign(fa))
a=x ; fa=f(x) ;
else
b=x ; fb=f(x) ;
end
fprintf('d''itération i=%d \t la solution est x0=f \t f(x0)= %f\n',i,x,f(x))
end ;
fprintf('La solution finale est x0 = %f \n',x)
else
disp('On ne peut pas faire de Bisction dans cet intervalle ! !')
end
```

-En utilisant la même méthode, complétez la solution d'équations précédente