**MÉTHODE NUMÉRIQUE**

L’objectif de ce TP est de résoudre numériquement des fonctions grâce au langage C.

Nous débuterons par créer le programme permettant d’établir une fonction puis de retranscrire les résultats de celle-ci dans un fichier texte.

Ensuite nous allons programmer la résolution d’équation par la méthode de dichotomie.

Pour finir, nous établirons la résolution d’équation par la méthode de la sécante et celle de Newton.

* **Création d’un fonction simple et écriture des valeurs dans un fichier texte :**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include <math.h> /\*Appel des bibliotheques necessaire a la bonne execution du programme \*/

double y,x,a,b,h,i; /\*Definir les variables \*/

double g(double); /\*Definir la fonction\*/

void write\_values(double a, double b, double h); /\*void appel la fonction write\_values sans en afficher sa valeur \*/

int main ()

/\*On rentre les paramètres necessaire pour calculer x et f(x) entre a et b, le tout avec un pas de h.

a sert à determiner le x de la fonction

b sert à determiner à quel x le programme s’arrete

write\_values(a,b,h) calcule la fonction grace à a et b \*/

{

printf("borne inferieur de l'intervalle :\n");

scanf("%lg",&a);

printf("borne superieur de l'intervalle :\n");

scanf("%lg",&b);

printf("pas :\n");

scanf("%lg",&h);

write\_values(a,b,h);

return EXIT\_SUCCESS;

}

double g(double x) /\*Nous definissons la fonction permettant de calculer les valeurs du tableau\*/

{

return x\*x\*x-(3\*x)+1;

}

/\*Appel de la fonction pour integrer résultats dans un tableau en considerant les valeurs rentrees par l’utilisateur.

Resultats integre dans un fichier texte et s’ouvrira à la fin du programme grace a :"FILE \*f = fopen("values.txt","w")" \*/

void write\_values(double a, double b, double h)

{

double x,y;

x=a;

FILE \*f = fopen("values.txt","w");

for(i=a; i<=b; i=i+h)

{

y=g(x);

fprintf (f, "x=%lg,f(x)=%lg\n",x,y);

x=x+h;

}

* **Programme de résolution par la méthode de dichotomie :**

La dichotomie permet de résoudre l’équation f(x)=0. La fonction f change de signe donc f(x) passe forcement par 0. Nous découpons l’intervalle sur [a,b] puis nous gardons l’intervalle où f(x) change de signe. Nous avons alors un encadrement plus petit, nous devons reproduire le même système jusqu’à avoir x.

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

/\*Nous decoupons notre fonctions suivant : [-1.9;-1.8], [0.3;0.4], [1.5;1.6] \*/

double a,b,c,e,it;

double f(double x)

{

return x\*x\*x-(3\*x)+1; /\* Dichotomie affectee à cette fonction \*/

}

int main (int argc,char \*argv[]) **/**\*Integrer les resultats sous un tableau.

« Int argc » = nombre de parametres utilise par le programme

« char \*argv) = tableau de chaine de caracteres.

Nous allons ensuite integrer les variables dans le terminal.\*/

{

sscanf(argv[1], "%lg", &a);

sscanf(argv[2], "%lg", &b);

sscanf(argv[3], "%lg", &e);

c = (a+b)/2; /\*Nous decoupons l’intervalle en 2 ; while permet de definir le bon interval \*/

it = 0;

FILE g = fopen("values\_dicho.txt","w");

**/**\* Tant que b-a > e, nous continuons les itérations.

Si f(a)\*f(c) < 0, alors b vaut c, nous pouvons alors definir un nouvel intervalle.

Si f(b)\*f(c)<0 alors c’est a qui prendra la valeur de c \*/

while (fabs(b-a) > e) {

if (f(a)\*f(c) < 0) {

b = c;

}

if (f(c)\*f(b) < 0) {

a = c;

}

c = (a+b)/2;

it = it + 1;

}

**/**\* Nous affichons les resultats \*/

fprintf(g, "encadrement = [%.10f, %.10f]\n", a, b);

fprintf(g, " solution = %.10f, test = %lg\n", c, f(c));

fprintf(g, "nombre d'iteration = %lg\n", it);

return EXIT\_SUCCESS;

}

* **Programme de résolution par la méthode de Newton :**

La méthode de Newton permet de résoudre l'équation f (x) = 0.

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

double f(double x) { /\*Definition de la fonction \*/

return x\*x\*x - 3\*x + 1;

}

double df(double x) { /\*Definition de la dérivé de la fonction precedente\*/

return 3\*x\*x - 3;

}

/\*Integration des resultats sous forme de tableau \*/

int main (int argc,char \*argv[]) {

double x0, x1, eps, erreur;

int it;

sscanf(argv[1], "%lg", &x0);

sscanf(argv[2], "%lg", &eps);

iteration = 0;

/\*Nous créons une boucle do. Cette boucle permet de calculer x1=x0-.

Et erreur=x0-x1 et tant que erreur > eps, on continue les iterations \*/

do {

printf("x0 = %lg\n", x0);

x1 = x0 - f(x0) / df(x0);

erreur = fabs(x0-x1);

x0 = x1;

it = it + 1;

}

while (erreur > eps);

printf("it = %d\n", it);

return EXIT\_SUCCESS;

* **Programme de résolution par la méthode de la sécante :**

La méthode des sécantes permet de résoudre l’équation f(x)=0 par encadrement de la solution.

Nous choisissons 2 point a et b de la fonction, pour ensuite construire le segment reliant (x0,f(x0)) et (x1,f(x1)).

La droite à pour équation y-f(x1)=(x-x0)

Nous prenons alors x de tel que y=0

include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

double f(double x) {

return x\*x\*x-(3\*x)+1;

}

void secante(double x1, double x0, double eps) { /\* fonction secante sans afficher la valeur \*/

double x2, erreur;

int it = 0;

FILE \*g = fopen("values\_secante.txt","w");

Creation de la boucle do, pour le calcul de l’equation 0=(x-x0) + f(x1)

Ensuite tant que x1-x0 > eps, nous continuons les itérations. \*/

do {

x2 = ( ( x0 \* f(x1) ) - ( x1 \* f(x0) ) ) / ( f(x1) - f(x0) );

erreur = fabs(x1-x0);

x0 = x1;

x1 = x2;

it = it + 1;

}

while (erreur > eps);

fprintf(g, "endradrement = [%.10f, %.10f]\n", x1, x0);

fprintf(g, "nombre d'iteration = %d\n", it);

printf("nombre d'iteration = %d\n", it);

}

int main (int argc,char \*argv[]) { /\* Nous rangeons les resultats sous un tableau \*/

double x1,x0,eps;

sscanf(argv[1], "%lg", &x0);

sscanf(argv[2], "%lg", &x1);

sscanf(argv[3], "%lg", &eps);

secante(x1,x0,eps);

return EXIT\_SUCCESS;

}

* **Conclusion :**

Lors de ce TP nous avons étudié différentes méthodes de programmation de la résolution de l’équation f(x)=0.

Nous avons pu constaté que la résolution est particulièrement rapide grâce au programme.

Il est toute fois possible de résoudre ces équations sans outils numériques mais cela une plus grande perte de temps.