Rajshahi University of Engineering & Technology

CSE 2104: Sessional Based on CSE 2103

Lab Report 02

Dated: 20.02.18

Submitted to

Shyla Afroge

Assistant Professor

Dept. of Computer Science & Engineering

&

Instructor, CSE 2104

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of Computer Science & Engineering

Problem#01: Compare the iterations between Bisection method and False Position method

**THEORY:** The algorithm used to find the root of an equation by these two methods - Bisection and False position - are given below.

*Algorithm of Bisection Method*

1. Start
2. Decide initial values for x1 and x2 and stopping criterion, E.
3. Compute f1 = f(x1) and f2 = f(x2).
4. If f1 \* f2>0, x1 and x2 do not bracket any root and go to step 7;

Otherwise continue.

1. Compute x0 = (x1+x2)/2 and compute f0 = f(x0)
2. If f1\*f0 < 0 then

set x2 = x0

Else

set x1 = x0

set f1 = f0

1. If absolute value of (x2 – x1)/x2 is less than error E, then

root = (x1 + x2)/2

write the value of root

go to step 7

else

go to step 4

1. Stop.

*Algorithm of False Position Method*

1. Start
2. Decide initial values for x1 and x2 and stopping criterion, E.
3. Compute f1 = f(x1) and f2 = f(x2).
4. If f1 \* f2>0, x1 and x2 do not bracket any root and go to step 7;

Otherwise continue.

1. Compute x0 = ((x1 \* f(x2)) - (x2 \* f(x1))) / (f(x2) - f(x1)) and compute f0 = f(x0)
2. If f1\*f0 < 0 then

set x2 = x0

Else

set x1 = x0

set f1 = f0

1. If absolute value of (x2 – x1)/x2 is less than error E, then

root = (x1 + x2)/2

write the value of root

go to step 7

else

go to step 4

1. Stop.

#include <stdio.h>

#include <math.h>

#define ERR .0001

int bisecITER = 0;

int falPosITER = 0;

//main function

double f(double x)

{

return x\*x\*x - 2\*x - 5;

}

//differentiate the function

double fd(double x)

{

return 3\*x\*x - 2;

}

double fdd(double x)

{

return 3\*x;

}

//Formatted print function

void print(int ITER, double a, double b, double c)

{

printf(" %2d | %lf | %lf | %lf | %9lf \n", ITER, a, b, c, f(c));

}

//bisection method function

void bisec(double a, double b)

{

bisecITER = 0;

if(f(a) \* f(b) >= 0)

{

printf("The assumed values are wrong...\nTry new values...\n\n\ta = ");

scanf("%lf", &a);

printf("\tb = ");

scanf("%lf", &b);

printf("\n\n");

return;

}

else

{

++bisecITER;

double c = a;

printf("==================\n Bisection Method \n==================\n");

printf("\n Iter. No. | a | b | x | f(x) \n");

printf(" ------------------------------------------------------------------\n");

print(bisecITER, a, b, c);

while (fabs(b - a) >= ERR)

{

++bisecITER;

c = (a+b)/2;

if (f(c) == 0.0){

print(bisecITER, a, b, c);

break;

}

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

{

b = c;

print(bisecITER, a, b, c);

}

else if (f(c) \* f(a) > 0)

{

a = c;

print(bisecITER, a, b, c);

}

else

{

printf("This is a RIDICULUS function\n");

return;

}

}

printf("\n\nThe value of root is : %lf\n", c);

printf("Absolute error : %lf\n", ERR);

printf("Number of Iterations : %d\n\n", bisecITER);

}

}

void falPos(double a, double b)

{

double c, temp, error;

falPosITER = 0;

if(f(a) \* f(b) > 0)

{

printf("The assumed values are wrong...\nTry new values...\n\n\ta = ");

scanf("%lf", &a);

printf("\tb = ");

scanf("%lf", &b);

printf("\n\n");

return;

}

else

{

printf("\n\n=======================\n False Position Method \n=======================\n");

printf("\n Iter. No. | a | b | x | f(x) \n");

printf(" ------------------------------------------------------------------\n");

if(f(a) == 0 || f(b) == 0)

{

printf("Root:\t%f\n", f(a) == 0 ? a:b);

return;

}

do

{

temp = c;

c = (((a \* f(b)) - (b \* f(a))) / (f(b) - f(a)));

print(++falPosITER, a, b, c);

if(f(c) == 0)

{

break;

}

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

{

b = c;

}

else

{

a = c;

}

error = fabs(c - temp);

} while(error > ERR);

printf("\n\nThe value of root is : %lf\n", c);

printf("Absolute error : %lf\n", ERR);

printf("Number of Iterations : %d\n\n", falPosITER);

}

}

void compare()

{

printf("Number of iterations in Bisection method: %d\nNumber of iterations in False Position method: %d\n", bisecITER, falPosITER);

if(bisecITER == 0 || falPosITER == 0)

{

printf("\nBoth or one of the methods are not run before...\nTo calculate the number of iterations, FIRST, run the both methods and then compare.\n\n");

}

else if(falPosITER < bisecITER)

{

printf("So, False Position Method is efficient.\n\n");

}

else if(bisecITER < falPosITER)

{

printf("So, Bisection method is efficient.\n\n");

}

else

{

printf("Both method has the same iterations.\n\n");

}

}

int main()

{

double a = 2.0, b = 3.0;

while(1)

{

printf("Enter your choice:\n\t1. Bisection Method\n\t2. False Position Method\n\t3. Comparison\n\t4. Exit\n\n");

int c;

scanf("%d", &c);

switch(c)

{

case 1: bisec(a, b); break;

case 2: falPos(a, b); break;

case 3: compare(); break;

case 4: return 0;

default : printf("\nI N V A L I D C H O I C E. . .\nP R O G R A M T E R M I N A T I N G...\n");

return 0;

}

}

return 0;

}

TERMINAL:

Musafeer@DESKTOP-OGK5MQH MINGW64 ~/desktop/S T U D Y/CSE 2104/Lab 2

$ gcc bisection.c

Musafeer@DESKTOP-OGK5MQH MINGW64 ~/desktop/S T U D Y/CSE 2104/Lab 2

$ ./a.exe

Enter your choice:

1. Bisection Method

2. False Position Method

3. Comparison

4. Exit

1

==================

Bisection Method

==================

Iter. No. | a | b | x | f(x)

------------------------------------------------------------------

1 | 2.000000 | 3.000000 | 2.000000 | -1.000000

2 | 2.000000 | 2.500000 | 2.500000 | 5.625000

3 | 2.000000 | 2.250000 | 2.250000 | 1.890625

4 | 2.000000 | 2.125000 | 2.125000 | 0.345703

5 | 2.062500 | 2.125000 | 2.062500 | -0.351318

6 | 2.093750 | 2.125000 | 2.093750 | -0.008942

7 | 2.093750 | 2.109375 | 2.109375 | 0.166836

8 | 2.093750 | 2.101563 | 2.101563 | 0.078562

9 | 2.093750 | 2.097656 | 2.097656 | 0.034714

10 | 2.093750 | 2.095703 | 2.095703 | 0.012862

11 | 2.093750 | 2.094727 | 2.094727 | 0.001954

12 | 2.094238 | 2.094727 | 2.094238 | -0.003495

13 | 2.094482 | 2.094727 | 2.094482 | -0.000771

14 | 2.094482 | 2.094604 | 2.094604 | 0.000592

15 | 2.094543 | 2.094604 | 2.094543 | -0.000090

The value of root is : 2.094543

Absolute error : 0.000100

Number of Iterations : 15

Enter your choice:

1. Bisection Method

2. False Position Method

3. Comparison

4. Exit

2

=======================

False Position Method

=======================

Iter. No. | a | b | x | f(x)

------------------------------------------------------------------

1 | 2.000000 | 3.000000 | 2.058824 | -0.390800

2 | 2.058824 | 3.000000 | 2.081264 | -0.147204

3 | 2.081264 | 3.000000 | 2.089639 | -0.054677

4 | 2.089639 | 3.000000 | 2.092740 | -0.020203

5 | 2.092740 | 3.000000 | 2.093884 | -0.007451

6 | 2.093884 | 3.000000 | 2.094305 | -0.002746

7 | 2.094305 | 3.000000 | 2.094461 | -0.001012

8 | 2.094461 | 3.000000 | 2.094518 | -0.000373

The value of root is : 2.094518

Absolute error : 0.000100

Number of Iterations : 8

Enter your choice:

1. Bisection Method

2. False Position Method

3. Comparison

4. Exit

3

Number of iterations in Bisection method: 15

Number of iterations in False Position method: 8

So, False Position Method is efficient.

Enter your choice:

1. Bisection Method

2. False Position Method

3. Comparison

4. Exit

4

**DISCUSSION**:As we seen from the menu program, the false position method is much more efficient from the bisection method. Whereas, there may occur some exceptional cases as well.