RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 02

Date of Experiment: 13.02.18
Date of Submission: 20.02.18

Submitted to:
Shyla Afroge
Assistant Professor,
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Submitted by:
Riyad Morshed Shoeb
Roll No: 1603013
Section: A
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Name of the experiment: Comparison between Bisection Method, False Position Method and Iteration Method

Theory:

False Position Method

This method is applicable for numerically solving the equation $f(x) = 0$ for the real variable x, where f is a continuous function defined on an interval $[a, b]$ and where $f(a)$ and $f(b)$ have opposite signs. In this case a and b are said to bracket a root since the continuous function f must have at least one root in the interval (a, b).
At each step the method divides the interval in two by

$$x = \frac{a \times f(b) - b \times f(a)}{f(b) - f(a)}$$

of the interval and the value of the function $f(x)$ at that point. Unless x is itself a root (which is very unlikely, but possible) there are now only two possibilities: either $f(a)$ and $f(x)$ have opposite signs and bracket a root, or $f(x)$ and $f(b)$ have opposite signs and bracket a root. The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. The process is continued until the interval is sufficiently small.
Explicitly, if $f(a)$ and $f(x)$ have opposite signs, then the method sets x as the new value for b, and if $f(b)$ and $f(x)$ have opposite signs then the method sets x as the new a. (If $f(x)=0$ then x may be taken as the solution and the process stops.) In both cases, the new $f(a)$ and $f(b)$ have opposite signs, so the method is applicable to this smaller interval.

Iteration Method

It is a mathematical procedure that uses an initial guess to generate a sequence of improving approximate solutions for a class of problems, in which the n-th approximation is derived from the previous ones. A specific implementation of an iterative method, including the termination criteria, is an algorithm of the iterative method.
If an equation can be put into the form $f(x) = x$, and a solution **x** is an attractive fixed point of the function f, then one may begin with a point x_1 in the basin of attraction of **x**, and let $x_{n+1} = f(x_n)$ for $n \geq 1$, and the sequence $\{x_n\}_{n \geq 1}$ will converge to the solution **x**. Here x_n is the nth approximation or iteration of x and x_{n+1} is the next or $n + 1$ iteration of x.
Alternately, superscripts in parentheses are often used in numerical methods, so as not to interfere with subscripts with other meanings. If the function f is continuously differentiable, a sufficient condition for convergence is that the spectral radius of the derivative is strictly bounded by one in a neighborhood of the fixed point. If this condition holds at the fixed point, then a sufficiently small neighborhood (basin of attraction) must exist.

Code:

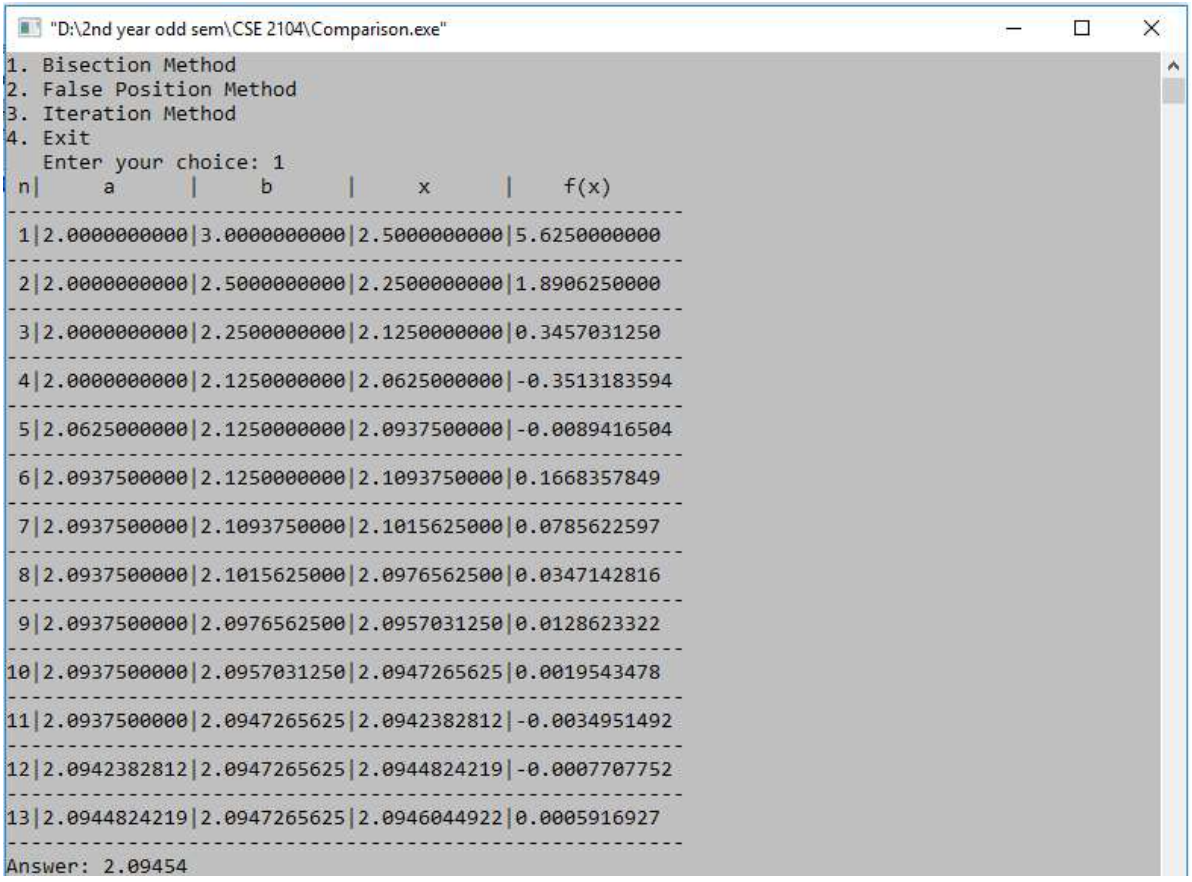
```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<algorithm>
using namespace std;

double x0;
double x=0;

double f(double x)
{
    return ((x*x*x) - (2*x) - 5);
}
```

```
double phi(double x)
{
    double temp= sqrt(2+(5/x)) ;
        return temp;
}

int bisection(double a, double b)
{
    int n=0;

    printf(" n|     a      |     b      |     x      |    f(x)    \n");
    printf("-----------------------------------------------------------\n");
    while(1)
    {
        x0=x;
        x=(a+b)/2;
        if(abs(x0-x)>=0.0001)
        {
            n++;
            if(f(a)*f(x)>0)
            {
                printf("%2d|%3.10f|%3.10f|%3.10f|%3.10f\n",n,a,b,x,f(x));
                a=x;
            }
            else
            {
                printf("%2d|%3.10f|%3.10f|%3.10f|%3.10f\n",n,a,b,x,f(x));
                b=x;
            }
            printf("-----------------------------------------------------------
-\n");
        }
        else
            break;
    }
    printf("Answer: ");
    cout<<x<<endl;

    return n;
}

int false_position(double a, double b)
{
    int n=0;

    printf(" n|     a      |     b      |     x      |    f(x)    \n");
    printf("-----------------------------------------------------------\n");
    while(1)
    {
        x0=x;
        x=((a*f(b)-b*f(a))/(f(b)-f(a)));
        if(abs(x0-x)>=0.0001)
        {
            n++;
            if(f(a)*f(x)>0)
            {
                printf("%2d|%3.10f|%3.10f|%3.10f|%3.10f\n",n,a,b,x,f(x));
                a=x;
            }
            else
            {
```

```
                printf("%2d|%3.10f|%3.10f|%3.10f|%3.10f\n",n,a,b,x,f(x));
                b=x;
            }
            printf("-------------------------------------------------------
-\n");
        }
        else
            break;
    }
    printf("Answer: ");
    cout<<x<<endl;

    return n;
}

int iteration(double a)
{
    int n=0;
    printf(" n|      a     |    f(a)    \n");
    printf("----------------------------\n");
    while(1)
    {
        n++;
        printf("%2d|%3.10f|%3.10f\n",n,a,phi(a));
        if(abs(a-phi(a))<=.00001)
            {
                printf("Answer: ");
                cout<<phi(a)<<endl;
                break;
            }
        else
            a=phi(a);
        printf("----------------------------\n");
    }
    return n;
}

int main(void)
{
    int ck;
    int bi, fp, it;

    while(1)
    {
        cout<<"1. Bisection Method\n2. False Position Method\n3. Iteration
Method\n4. Exit\n";
        cout<<"   Enter your choice: ";
        cin>>ck;

        switch(ck)
        {
        case 1:
            bi=bisection(2,3);
            break;

        case 2:
            fp=false_position(2,3);
            break;

        case 3:
            it=iteration(2);
```

```
            break;

        case 4:
            goto xx;

        default:
            cout<<"Wrong Choice...";
        }
    }

    xx:
    printf("\nIteration needed for Bisection Method: %d\n",bi);
    printf("Iteration needed for False Position Method: %d\n",fp);
    printf("Iteration needed for Iteration Method: %d\n\n",it);

    if(fp<bi&&fp<it)
        cout<<"False Position is better."<<endl;
    else if(it<bi&&it<fp)
        cout<<"Iteration Method is better."<<endl;
    else
        cout<<"Bisection Method is better."<<endl;
}
```

Output:



```
1. Bisection Method
2. False Position Method
3. Iteration Method
4. Exit
   Enter your choice: 2
 n|     a      |     b      |     x      |    f(x)
-------------------------------------------------------
 1|2.0000000000|3.0000000000|2.0588235294|-0.3907999186
-------------------------------------------------------
 2|2.0588235294|3.0000000000|2.0812636598|-0.1472040596
-------------------------------------------------------
 3|2.0812636598|3.0000000000|2.0896392101|-0.0546765033
-------------------------------------------------------
 4|2.0896392101|3.0000000000|2.0927395743|-0.0202028663
-------------------------------------------------------
 5|2.0927395743|3.0000000000|2.0938837085|-0.0074505059
-------------------------------------------------------
 6|2.0938837085|3.0000000000|2.0943054511|-0.0027456728
-------------------------------------------------------
 7|2.0943054511|3.0000000000|2.0944608458|-0.0010115739
-------------------------------------------------------
Answer: 2.09452
```

```
1. Bisection Method
2. False Position Method
3. Iteration Method
4. Exit
   Enter your choice: 3
 n|      a     |    f(a)
----------------------------
 1|2.0000000000|2.1213203436
----------------------------
 2|2.1213203436|2.0873482230
----------------------------
 3|2.0873482230|2.0965170506
----------------------------
 4|2.0965170506|2.0940171593
----------------------------
 5|2.0940171593|2.0946968820
----------------------------
 6|2.0946968820|2.0945119261
----------------------------
 7|2.0945119261|2.0945622432
----------------------------
 8|2.0945622432|2.0945485537
----------------------------
 9|2.0945485537|2.0945522781
Answer: 2.09455
1. Bisection Method
2. False Position Method
3. Iteration Method
4. Exit
   Enter your choice: 4

Iteration needed for Bisection Method: 13
Iteration needed for False Position Method: 7
Iteration needed for Iteration Method: 9

False Position is better.

Process returned 0 (0x0)   execution time : 360.257 s
Press any key to continue.
```