Heaven's Light is Our Guide
Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Lab Report

Topic : Solution of Algebraic Equation

Course Title : Sessional Based on CSE 2103
Course Code : CSE 2104

Submitted By :
Md. Tanzid Hasan
Section : A
Roll No : 1603054

Submitted To :
Shyla Afroge
Assistant Professor
Dept. of Computer Science & Engineering

Date : 24-02-2018

Experiment No : 01

Experiment Name : False Position Method

Theory :

False position method is a root-finding algorithm that is qualitative similar to the bisection method in that it uses nested intervals based on opposite signs at the endpoints to converge to a root, but is computationally based on the secant method.

The steps to apply the false-position method to find the root of the equation f(x)=0 are as follows.

1. Choose x_L and x_U as two guesses for the root such that $f(x_L)f(x_U) < 0$, or in other words, f(x) changes sign between x_L and x_U.
2. Estimate the root, x_r of the equation f(x) = 0 as

$$X_r = \frac{x_U f(x_L) - x_L f(x_U)}{f(x_L) - f(x_U)}$$

3. Now check the following

If $f(x_L)f(x_r) < 0$, then the root lies between x_L and x_r; then $x_L = x_L$ and $x_U = x_r$.

If $f(x_L)f(x_r) > 0$, then the root lies between x_r and x_U; then $x_L = x_r$ and $x_U = x_U$.

If $f(x_L)f(x_r) = 0$, then the root is x_r. Stop the algorithm.

4. Find the new estimate of the root

$$x_r = \frac{x_U f(x_L) - x_L f(x_U)}{f(x_L) - f(x_U)}$$

Find the absolute relative approximate error as

$$|\epsilon_a| = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100$$

where

x_r^{new} = estimated root from present iteration

x_r^{old} = estimated root from previous iteration

5. Compare the absolute relative approximate error ϵ_a with the pre-specified relative error tolerance ϵ_s. If the difference between ϵ_a & ϵ_s is less than the terminating condition, then go to step 3, else stop the algorithm.

Note : one should also check whether the number of iterations is more than the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user about it. And the false-position and bisection algorithms are quite similar. The only difference is the formula used to calculate the new estimate of the root x_r as shown in steps #2 and #4!

Code :

```
#include <iostream>
#include <math.h>
using namespace std;
double f(double d,double c[],int r)
{
   int s=r-1;
   double t=0;
   for(int i=0;i<r;i++)
   {
      t=t+(c[i]*pow(d,s--));
   }
   return t;
}
int main()
{
   int i=1,n;
   double a,a1,b,b1,e,x1,x2=0,y;
   cout<<"Enter the power of equation : ";
   cin>>n;
   double arr[n+1];
   int m=n;
   for(int i=0;i<n+1;i++)
   {
      cout<<"Coefficient of x^"<<m--<<" : ";
      cin>>arr[i];
   }
   for(;;)
   {
```

```
    cout<<"Enter a : ";
    cin>>a;
    cout<<"Enter b : ";
    cin>>b;
    e=f(a,arr,n+1)*f(b,arr,n+1);
    if(e<0)
        break;
    else
        cout<<"Wrong input."<<endl;
}
cout<<endl<<"i\t\t"<<"a\t\t"<<"b\t\t"<<"x\t\t"<<"y"<<endl<<endl;
for(;;)
{
    a1=f(a,arr,n+1); b1=f(b,arr,n+1);
    x1=((a*b1)-(b*a1))/(b1-a1);
    y=f(x1,arr,n+1);
    cout<<i<<"\t\t"<<a<<"\t\t"<<b<<"\t\t"<<x1<<"\t\t"<<y<<endl;
    if(fabs(x2-x1)<0.00001)
        break;
    if((a1*y)<0)
        b=x1;
    if((a1*y)>0)
        a=x1;
    if(y==0)
        break;
    i++;
    x2=x1;
}
cout<<endl<<"Root : "<<x1<<endl;    }
```

Input :

Enter the power of equation : 3

Coefficient of x^3 : 1

Coefficient of x^2 : 0

Coefficient of x^1 : -2

Coefficient of x^0 : -5

Enter a : 2

Enter b : 3

Output :

i	a	b	x	y
1	2	3	2.05882	-0.3908
2	2.05882	3	2.08126	-0.147204
3	2.08126	3	2.08964	-0.0546765
4	2.08964	3	2.09274	-0.0202029
5	2.09274	3	2.09388	-0.00745051
6	2.09388	3	2.09431	-0.00274567
7	2.09431	3	2.09446	-0.00101157
8	2.09446	3	2.09452	-0.000372653
9	2.09452	3	2.09454	-0.000137276
10	2.09454	3	2.09455	-5.05686e-005

Root : 2.09455

Discussion :

In this problem, the equation was generated. After generating the equation, two initial guess value was taken. Then the false position method was started as the algorithm. As the terminating condition, the difference was set 0.0001. So when the difference was less than 0.0001, the algorithm stopped and the last root was shown as result.

Experiment No : 02

Experiment Name : The Iteration Method

Theory :

If we can write $f(x)=0$ in the form $x=g(x)$, then the point x would be a fixed point of the function g (that is, the input of g is also the output). Then an obvious sequence to consider is

$$X_{n+1}=g(x_n)$$

If we look at this on a graph we can see how this could converge to the intersection. Any function can be written in this form if we define $x=g(x)$, though in some cases other rearrangements may prove more useful (must satisfy the condition $|g(x)|<1$). This method is useful for finding a positive root to the infinite series also.

Fixed Point Theorem (*Statement*) : If g is continuous on $[a,b]$ and $a \leq g(x) \leq b$ for all x in $[a,b]$, then g has at least a fixed point in $[a,b]$.

Further, suppose $g'(x)$ is continuous on (a,b) and that a positive constant c exists with $|g'(x)| \leq c <1$, for all x in (a,b). Then there is a unique fixed point α of g in $[a,b]$. Also, the iterates $x_{n+1} = g(x_n)$ $n \geq 0$ will converge to α for any choice of x_0 in $[a,b]$.

Code :

```
#include <iostream>
#include <math.h>
using namespace std;
double df(int e)
{
   double s=-(5/(2*e*e*sqrt(2+(5/e))));
   return s;
}
double f(double d)
{
   double t=sqrt(2+(5/d));
   return t;   }
```

```
int main()
{
   int i=1,n=1;
   double a,x1,x2=0,b;
   for(;;)
   {
      a=df(n);
      if(fabs(a)<1)
         break;
      else
         n++;
   }
   b=n;
   cout<<endl<<"i\t\t"<<"x\t\t"<<"Q(x)"<<endl<<endl;
   for(;;)
   {
      x1=f(b);
      cout<<i<<"\t\t"<<b<<"\t\t"<<x1<<endl;
      if(fabs(x1-x2)<0.0001)
         break;
      else
         b=x1;
      x2=x1;
      i++;
   }

   cout<<endl<<"Root : "<<x1<<endl;
   return 0;
}
```

Output :

i	x	Q(x)
1	1	2.64575
2	2.64575	1.97226
3	1.97226	2.12959
4	2.12959	2.08515
5	2.08515	2.09712
6	2.09712	2.09385
7	2.09385	2.09474
8	2.09474	2.0945
9	2.0945	2.09457

Root : 2.09457

Discussion :

In this problem, no input were taken. One equation was assigned to be calculated. First, the initial value was generated and then it was sent to determine the root. In the program, two function was created. One was for g(x) and other was for g'(x). As the terminating condition, the difference between $g(x_n)$ and $g(x_{n-1})$ was 0.0001. When the program was terminated, the last g(x) means $g(x_n)$ will be the root of the assigned equation.

Experiment No : 03

Experiment Name : Menu of Bisection & False Position Method

Theory :

Menu program is a program where more than one program can be executed. In menu program, one program can be executed many times until we exit the program. It is a menu program of Bisection & False Position method.

The bisection method in mathematics is a root-finding method that repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing. It is a very simple and robust method, but it is also relatively slow. Because of this, it is often used to obtain a rough approximation to a solution which is then used as a starting point for more rapidly converging methods.

False position method is a root-finding algorithm that is qualitative similar to the bisection method in that it uses nested intervals based on opposite signs at the endpoints to converge to a root, but is computationally based on the secant method.

Both the program will be done following their algorithm as they were done before. And in a menu option, there will be comparison between both method which is more efficient.

Code :

```
#include <iostream>
#include <math.h>
using namespace std;
int i=1,j=1;
double f(double d,double c[],int r)
{
   int s=r-1;
   double t=0;
   for(int i=0;i<r;i++)
   {
      t=t+(c[i]*pow(d,s--));
   }
```

```
    return t;
}
void bisection()
{
    int n;
    double a,b,e,x1,x2=0,y;
    cout<<"Enter the power of equation : ";
    cin>>n;
    double arr[n+1];
    int m=n;
    for(int i=0;i<n+1;i++)
    {
        cout<<"Coefficient of x^"<<m--<<" : ";
        cin>>arr[i];
    }
    for(;;)
    {
        cout<<"Enter a : ";
        cin>>a;
        cout<<"Enter b : ";
        cin>>b;
        e=f(a,arr,n+1)*f(b,arr,n+1);
        if(e<0)
            break;
        else
            cout<<"Wrong input."<<endl;
    }
```

```
    cout<<endl<<"i\t\t"<<"a\t\t"<<"b\t\t"<<"x\t\t"<<"y"<<endl<<endl;
    for(;;)
    {
        x1=(a+b)/2;
        y=f(x1,arr,n+1);
        cout<<i<<"\t\t"<<a<<"\t\t"<<b<<"\t\t"<<x1<<"\t\t"<<y<<endl;
        if(fabs(x2-x1)<0.00001)
            break;
        if((f(a,arr,n+1)*y)<0)
            b=x1;
        if((f(a,arr,n+1)*y)>0)
            a=x1;
        if(y==0)
            break;
        i++;
        x2=x1;
    }
    cout<<endl<<"Root : "<<x1<<endl;
}
void falseposition()
{
    int n;
    double a,a1,b,b1,e,x3,x4=0,y;
    cout<<"Enter the power of equation : ";
    cin>>n;
    double arr[n+1];
    int m=n;
```

```
for(int i=0;i<n+1;i++)
{
   cout<<"Coefficient of x^"<<m--<<" : ";
   cin>>arr[i];
}
for(;;)
{
   cout<<"Enter a : ";
   cin>>a;
   cout<<"Enter b : ";
   cin>>b;
   e=f(a,arr,n+1)*f(b,arr,n+1);
   if(e<0)
      break;
   else
      cout<<"Wrong input."<<endl;
}
cout<<endl<<"i\t\t"<<"a\t\t"<<"b\t\t"<<"x\t\t"<<"y"<<endl<<endl;
for(;;)
{
   a1=f(a,arr,n+1); b1=f(b,arr,n+1);
   x3=((a*b1)-(b*a1))/(b1-a1);
   y=f(x3,arr,n+1);
   cout<<j<<"\t\t"<<a<<"\t\t"<<b<<"\t\t"<<x3<<"\t\t"<<y<<endl;
   if(fabs(x4-x3)<0.0001)
      break;
   if((a1*y)<0)
```

```
            b=x3;
        if((a1*y)>0)
            a=x3;
        if(y==0)
            break;
        j++;
        x4=x3;
    }
    cout<<endl<<"Root : "<<x3<<endl;
}
void comparison()
{
    if(i>j)
        cout<<endl<<"False position lag "<<fabs(i-j)<<" steps behind
Bisection."<<endl;
    if(i<j)
        cout<<endl<<"Bisection lag "<<fabs(i-j)<<" steps behind False
position."<<endl;
}
int main()
{
    int a;
    for(;;)
    {
        cout<<"1. Bisection Method"<<endl<<"2. False Position
Method"<<endl<<"3. Comparison"<<endl<<"4. Exit"<<endl<<"
Enter the option(1-4) : ";
        cin>>a;
```

```
        if(a>4)
           cout<<"Invalid input.";
        if(a==4)
           break;
        switch(a)
        {
        case 1 :
              {
                 cout<<endl<<"1. Bisection Method : "<<endl;
                 bisection();
                 cout<<endl<<endl;
              }break;
        case 2 :
              {
                 cout<<endl<<"2. False Position Method : "<<endl;
                 falseposition();
                 cout<<endl<<endl;
              }break;
        case 3 :
              {
                 cout<<endl<<"3. Comparison : "<<endl;
                 comparison();
                 cout<<endl<<endl;
              }break;
        }
    }
}
```

Input & Output :

1. Bisection Method
2. False Position Method
3. Comparison
4. Exit

Enter the option(1-4) : 1

1. Bisection Method :

Enter the power of equation : 3

Coefficient of x^3 : 1

Coefficient of x^2 : 0

Coefficient of x^1 : -2

Coefficient of x^0 : -5

Enter a : 2

Enter b : 3

i	a	b	x	y
1	2	3	2.5	5.625
2	2	2.5	2.25	1.89062
3	2	2.25	2.125	0.345703
4	2	2.125	2.0625	-0.351318
5	2.0625	2.125	2.09375	-0.00894165
6	2.09375	2.125	2.10938	0.166836
7	2.09375	2.10938	2.10156	0.0785623
8	2.09375	2.10156	2.09766	0.0347143
9	2.09375	2.09766	2.0957	0.0128623
10	2.09375	2.0957	2.09473	0.00195435
11	2.09375	2.09473	2.09424	-0.00349515

12	2.09424	2.09473	2.09448	-0.000770775
13	2.09448	2.09473	2.0946	0.000591693
14	2.09448	2.0946	2.09454	-8.95647e-005
15	2.09454	2.0946	2.09457	0.000251058
16	2.09454	2.09457	2.09456	8.07453e-005
17	2.09454	2.09456	2.09455	-4.41007e-006

Root : 2.09455

1. Bisection Method
2. False Position Method
3. Comparison
4. Exit

Enter the option(1-4) : 2

2. False Position Method :
Enter the power of equation : 3
Coefficient of x^3 : 1
Coefficient of x^2 : 0
Coefficient of x^1 : -2
Coefficient of x^0 : -5
Enter a : 2
Enter b : 3

i	a	b	x	y
1	2	3	2.05882	-0.3908
2	2.05882	3	2.08126	-0.147204
3	2.08126	3	2.08964	-0.0546765
4	2.08964	3	2.09274	-0.0202029

5	2.09274	3	2.09388	-0.00745051
6	2.09388	3	2.09431	-0.00274567
7	2.09431	3	2.09446	-0.00101157
8	2.09446	3	2.09452	-0.000372653

Root : 2.09452

1. Bisection Method
2. False Position Method
3. Comparison
4. Exit

Enter the option(1-4) : 3

3. Comparison :

False position lag 9 steps behind Bisection.

Discussion :

Both Bisection & False Position Method was included in the program. First, the method was chosen which to execute either Bisection or False position. In each method, the equation was generated and initial value of x was taken. Then for each method, their algorithm was followed and the root was determined. For both method, as the terminating condition, the difference between last two root was 0.0001. In the menu, there was an option to compare steps between both method for determining which one was much efficient. At last, there was an option to exit.