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

Department of Computer Science & Engineering

Lab Report

Topic : Error

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 : 14-02-2018

Experiment No : 01

Experiment Name : Absolute Error

Theory :

The absolute error is defined as the absolute value (magnitude) of the difference between the measured value and the true value. Let :

e_a = absolute error
x_m = measured value
x_t = true value

The formula for computing absolute error is : $e_a = | x_m - x_t |$

But here, first value will be taken as x which is Infinite Decimal Fractional Number. Then, three value will be taken as x_1, x_2 & x_3 which are very close to x. The modulus difference between x and x_1, x and x_2 & x and x_3 need to be calculated. Between these three difference, the absolute error will be the one which is the smallest difference.

Code :

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    double x,x1,x2,x3,e1,e2,e3;
    cout<<"Enter X : ";
    cin>>x;
    cout<<"Enter x1 : ";
    cin>>x1;
    cout<<"Enter x2 : ";
    cin>>x2;
    cout<<"Enter x3 : ";
    cin>>x3;
    e1=fabs(x-x1);
    e2=fabs(x-x2);
```

```
    e3=fabs(x-x3);
    if(e1<e2 && e1<e3)
        cout<<"Output : "<<x1<<endl;
    if(e2<e1 && e2<e3)
        cout<<"Output : "<<x2<<endl;
    if(e3<e2 && e3<e1)
        cout<<"Output : "<<x3<<endl;

    return 0;
}
```

Input :

0.3333333333

0.30

0.33

0.34

Output :

0.33

Discussion :

In this experiment, Infinite Decimal Fractional Number were Rational Number not Irrational Number. After that, x_1, x_2 & x_3 was taken two decimal fractional number. Absolute error is the difference between these number. But in this case, absolute error is the smallest difference.

Experiment No : 02

Experiment Name : Error of Square-root.

Theory :

The absolute error is defined as the absolute value (magnitude) of the difference between the measured value and the true value. Let :

e_a = absolute error

x_m = measured value

x_t = true value

The formula for computing absolute error is : $e_a = | x_m - x_t |$

The relative error is defined as the absolute error relative to the size of the measurement. All you need to do is divide the absolute error by the measured value. In addition to the variables above. Let :

e_r = relative error

Then the formula for computing relative error is : $e_r = e_a / x_m$

The percentage error expresses as a percentage the difference between an approximate or measured value and an exact or known value. Let :

e_p = percentage error

Then the formula for computing percentage error is : $e_p = e_r*100$

But here, first three value will be taken as x_1, x_2 & x_3 which are square-root of integer. Then, three value will be taken as y_1, y_2 & y_3 which are Finite Decimal Fractional Number of square-root of x_1, x_2 & x_3 respectively. Then, absolute error, relative error & percentage error will be determined between summation of x_1, x_2 & x_3 and summation of y_1, y_2 & y_3 .

Code :

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
   double x1,x2,x3,y1,y2,y3,x,y;
   cout<<"Enter x1 : ";
   cin>>x1;
```

```
    x1=sqrt(x1);
    cout<<"Enter x2 : ";
    cin>>x2;
    x2=sqrt(x2);
    cout<<"Enter x3 : ";
    cin>>x3;
    x3=sqrt(x3);
    cout<<"Enter y1 : ";
    cin>>y1;
    cout<<"Enter y2 : ";
    cin>>y2;
    cout<<"Enter y2 : ";
    cin>>y3;
    x=x1+x2+x3;
    y=y1+y2+y3;
    double eA=fabs(x-y);
    double eR=eA/x;
    double eP=eR*100;

    cout<<"Absolute Error : "<<eA<<endl<<"Relative Error : 
"<<eR<<endl<<"Percentage Error : "<<eP<<endl;

    return 0;
}
```

Input :

2

3

5

1.41421

1.73205

2.23607

Output :

2.34744e-006

4.36138e-007

4.36138e-005

Discussion :

In this experiment, x_1, x_2 & x_3 were taken as integer but it was saved as square-root of x_1, x_2 & x_3 respectively. And y_1, y_2 & y_3 was taken five decimal fractional number. Absolute error is difference between summation of square-root of x_1, x_2 & x_3 and summation y_1, y_2 & y_3 . Relative error is determined dividing absolute error by summation of square-root of x_1, x_2 & x_3 . Percentage error is determined multiplying relative error by 100.

Experiment No : 03

Experiment Name : Bi-section Method.

Theory :

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.

The 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, by the intermediate value theorem, 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 computing the midpoint $c = (a+b) / 2$ of the interval and the value of the function $f(c)$ at that point. Unless c is itself a root (which is very unlikely, but possible) there are now only two possibilities: either $f(a)$ and $f(c)$ have opposite signs and bracket a root, or $f(c)$ and $f(b)$ have opposite signs and bracket a root.[5]The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

Explicitly, if $f(a)$ and $f(c)$ have opposite signs, then the method sets c as the new value for b, and if $f(b)$ and $f(c)$ have opposite signs then the method sets c as the new a. (If $f(c)$=0 then c 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.[6]

Code :

```
#include <iostream>

#include <math.h>


using namespace std;


double f(double d,double c[],double 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,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<<"Value : "<<x1<<endl;

}
```

Input :

3

1

0

-2

-5

2

3

Output :

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

2.09455

Discussion :

In this experiment, first equation f(x) was taken. Two value, a & b were taken as a>b and sign of f(a) and f(b) was opposite. Average value of a & b,x_0 , was put in f(x) which was positive. Then, b was replaced by x_0. Average value of a & x_0 ,x_1, was put in f(x) which was positive. Then, x_0 was replaced by x_1. If $f(x_1)$ was negative, a would replace by x_1. And as the terminating condition, the modulus difference between x_1 and x_0 was less than 0.0001. When this condition was fulfilled, the loop was terminated. As a root of the equation, the last value of x was the root.