RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 08

Date of Experiment: 08.04.2018
Date of Submission: 22.04.2018

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: Implementation of Numerical Integration

Theory:

Given a set of data points $(x_0, y_0), (x_1, y_1), \dots\dots, (x_n, y_n)$ of a function $y = f(x)$, where f(x) is not known explicitly, it is required to compute the value of the definite integral,

$$I = \int_a^b ydx$$

Let the interval [a,b] be divided into n equal sub-intervals such that $a = x_0 < x_1 < x_2 < \cdots < x_n = b$. Clearly, $x_n = x_0 + nh$. Hence the integral becomes,

$$I = \int_{x_0}^{x_n} ydx$$

Approximating y by Newton's Forward Difference formula, we obtain,

$$I = \int_{x_0}^{x_n} \left[y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots \right] dx$$

Since, $x = x_0 + ph, dx = hdp$ and hence the above integral becomes,

$$I = h\int_0^n \left[y_0 + p\Delta y_0 + \frac{p(p-1)}{2}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{6}\Delta^3 y_0 + \cdots \right] dp$$

which gives on simplification,

$$\int_{x_0}^{x_n} ydx = nh\left[y_0 + \frac{n}{2}\Delta y_0 + \frac{n(2n-3)}{12}\Delta^2 y_0 + \frac{n(n-2)^2}{24}\Delta^3 y_0 + \cdots \right]$$

From this general formula, we can obtain different integration Formulae by putting n=1,2,3,…etc.

Trapezoidal Rule

Setting n=1 in the general formula, all differences higher than the first will become zero and we obtain,

$$\int_{x_0}^{x_1} ydx = h\left(y_0 + \frac{1}{2}\Delta y_0\right) = \frac{h}{2}(y_0 + y_1)$$

For the next interval $[x_1, x_2]$, we deduce similarly,

$$\int_{x_1}^{x_2} ydx = \frac{h}{2}(y_1 + y_2)$$

and so on. For the last interval $[x_{n-1}, x_n]$, we have,

$$\int_{x_{n-1}}^{x_n} ydx = \frac{h}{2}(y_{n-1} + y_n)$$

Combining all these expressions, we obtain the rule,

$$\int_{x_0}^{x_n} ydx = \frac{h}{2}[y_0 + 2(y_1 + y_2 + \cdots + y_{n-1}) + y_n]$$

which is known as the *trapezoidal rule*.

Simpson's 1/3-Rule

This rule is obtained by n=2 in the general equation. We obtain then,

$$\int_{x_0}^{x_2} ydx = 2h\left(y_0 + \Delta y_0 + \frac{1}{6}\Delta^2 y_0\right) = \frac{h}{3}(y_0 + 4y_1 + y_2)$$

Similarly,

$$\int_{x_2}^{x_4} ydx = \frac{h}{3}(y_2 + 4y_3 + y_4)$$

and finally,

$$\int_{x_{n-2}}^{x_n} ydx = \frac{h}{3}(y_{n-2} + 4y_{n-1} + y_n)$$

Summing up, we obtain,

$$\int_{x_0}^{x_n} ydx = \frac{h}{3}[y_0 + 4(y_1 + y_3 + y_5 + \cdots)] + 2(y_2 + y_4 + y_6 + \cdots)$$

which is known as the *Simpson's 1/3 rule*. It should be noted that this rule requires the division of the whole range into an even number of sub-intervals of width h.

Simpson's 3/8-Rule

Setting n=3 in the general formula, we observe that all the differences higher than the third will become zero and we obtain,

$$\int_{x_0}^{x_3} ydx = 3h\left(y_0 + \frac{3}{2}\Delta y_0 + \frac{3}{4}\Delta^2 y_0 + \frac{1}{8}\Delta^3 y_0\right) = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3)$$

Similarly,

$$\int_{x_3}^{x_6} ydx = \frac{3h}{8}(y_3 + 3y_4 + 3y_5 + y_6)$$

and so on. Summing up all these, we obtain,

$$\int_{x_0}^{x_n} ydx = [y_0 + 3(y_1 + y_2 + y_4 + y_5 + \cdots) + 2(y_3 + y_6 + \cdots)]$$

This is the *Simpson's 3/8 rule* but it is not as accurate as Simpson's 1/3 rule.

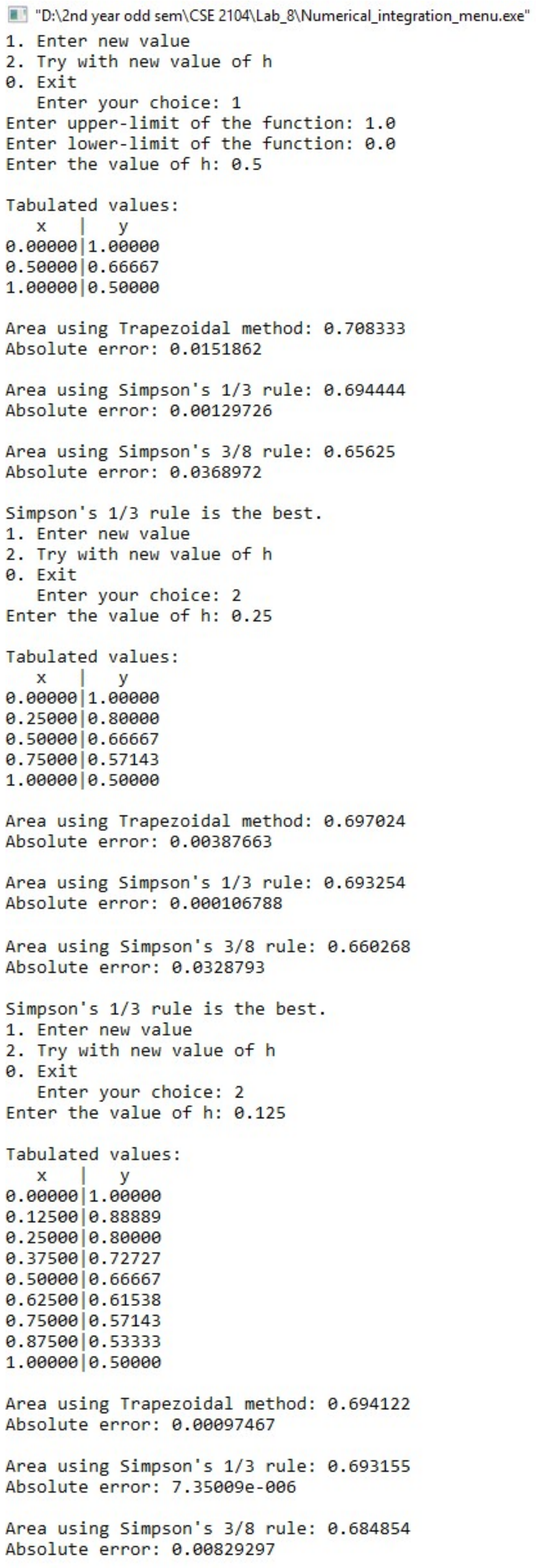
Code:

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

int main(void)
{
    double upper,lower,h;
    int steps;
    int i;
    int checker;
    double trap,trap_error;
    double simp_1_3,simp_1_3_error;
    double simp_3_8,simp_3_8_error;
    double calculated;

    while(1)
    {
        printf("1. Enter new value\n2. Try with new value of h\n0.
Exit\n   Enter your choice: ");
        cin>>checker;

        switch(checker)
        {
```

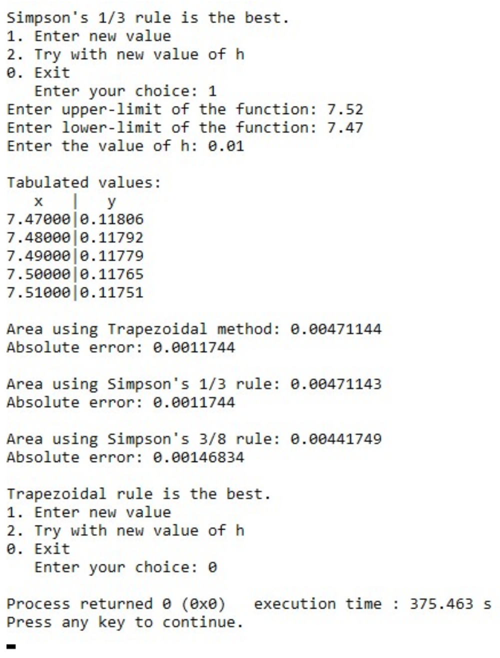
```
case 0:
    return 0;

case 1:
    {
        printf("Enter upper-limit of the function: ");
        cin>>upper;
        printf("Enter lower-limit of the function: ");
        cin>>lower;

        xx:
        printf("Enter the value of h: ");
        cin>>h;
        steps=(int)(((upper-lower)/h)+1);
        double x[steps];
        double y[steps];
        calculated=log(1+upper)-log(1+lower);
        x[0]=lower;
        y[0]=1/(1+x[0]);

        for(i=1;i<steps;i++)
        {
            x[i]=x[i-1]+h;
            y[i]=1/(1+x[i]);
        }
        printf("\nTabulated values:\n   x   |   y\n");
        for(i=0;i<steps;i++)
            printf("%0.5f|%0.5f\n",x[i],y[i]);

        trap=y[0]+y[steps-1];
        for(i=1;i<steps-1;i++)
            trap+=(2*y[i]);
        trap*=(h/2);
        trap_error=abs(calculated-trap);
        printf("\nArea using Trapezoidal method: ");
        cout<<trap<<endl;
        printf("Absolute error: ");
        cout<<trap_error<<endl;

        simp_1_3=y[0]+y[steps-1];
        for(i=1;i<steps-1;i++)
        {
            if(i%2==0)
                simp_1_3+=(2*y[i]);
            else
                simp_1_3+=(4*y[i]);
        }
        simp_1_3*=(h/3);
        simp_1_3_error=abs(calculated-simp_1_3);
        printf("\nArea using Simpson's 1/3 rule: ");
        cout<<simp_1_3<<endl;
        printf("Absolute error: ");
        cout<<simp_1_3_error<<endl;

        simp_3_8=y[0]+y[steps-1];
        for(i=1;i<steps-1;i++)
```

```
                {
                    if(i%3==0)
                        simp_3_8+=(2*y[i]);
                    else
                        simp_3_8+=(3*y[i]);
                }
                simp_3_8*=(3*h/8);
                simp_3_8_error=abs(calculated-simp_3_8);
                printf("\nArea using Simpson's 3/8 rule: ");
                cout<<simp_3_8<<endl;
                printf("Absolute error: ");
                cout<<simp_3_8_error<<endl;

                if(trap_error<simp_1_3_error)
                {
                    if(trap_error<simp_3_8_error)
                        printf("\nTrapezoidal rule is the best.\n");
                    else
                        printf("\nSimpson's   3/8   rule   is   the
best.\n");
                }
                else
                {
                    if(simp_1_3_error<simp_3_8_error)
                        printf("\nSimpson's   1/3   rule   is   the
best.\n");
                    else
                        printf("\nSimpson's   3/8   rule   is   the
best.\n");
                }
            }
            break;

        case 2:
            goto xx;
            break;

        default:
            printf("Wrong input...\n");
        }
    }
}
```

Output:



```
"D:\2nd year odd sem\CSE 2104\Lab_8\Numerical_integration_menu.exe"
1. Enter new value
2. Try with new value of h
0. Exit
   Enter your choice: 1
Enter upper-limit of the function: 1.0
Enter lower-limit of the function: 0.0
Enter the value of h: 0.5

Tabulated values:
   x    |    y
0.00000|1.00000
0.50000|0.66667
1.00000|0.50000

Area using Trapezoidal method: 0.708333
Absolute error: 0.0151862

Area using Simpson's 1/3 rule: 0.694444
Absolute error: 0.00129726

Area using Simpson's 3/8 rule: 0.65625
Absolute error: 0.0368972

Simpson's 1/3 rule is the best.
1. Enter new value
2. Try with new value of h
0. Exit
   Enter your choice: 2
Enter the value of h: 0.25

Tabulated values:
   x    |    y
0.00000|1.00000
0.25000|0.80000
0.50000|0.66667
0.75000|0.57143
1.00000|0.50000

Area using Trapezoidal method: 0.697024
Absolute error: 0.00387663

Area using Simpson's 1/3 rule: 0.693254
Absolute error: 0.000106788

Area using Simpson's 3/8 rule: 0.660268
Absolute error: 0.0328793

Simpson's 1/3 rule is the best.
1. Enter new value
2. Try with new value of h
0. Exit
   Enter your choice: 2
Enter the value of h: 0.125

Tabulated values:
   x    |    y
0.00000|1.00000
0.12500|0.88889
0.25000|0.80000
0.37500|0.72727
0.50000|0.66667
0.62500|0.61538
0.75000|0.57143
0.87500|0.53333
1.00000|0.50000

Area using Trapezoidal method: 0.694122
Absolute error: 0.00097467

Area using Simpson's 1/3 rule: 0.693155
Absolute error: 7.35009e-006

Area using Simpson's 3/8 rule: 0.684854
Absolute error: 0.00829297
```



```
Simpson's 1/3 rule is the best.
1. Enter new value
2. Try with new value of h
0. Exit
   Enter your choice: 1
Enter upper-limit of the function: 7.52
Enter lower-limit of the function: 7.47
Enter the value of h: 0.01

Tabulated values:
   x   |   y
7.47000|0.11806
7.48000|0.11792
7.49000|0.11779
7.50000|0.11765
7.51000|0.11751

Area using Trapezoidal method: 0.00471144
Absolute error: 0.0011744

Area using Simpson's 1/3 rule: 0.00471143
Absolute error: 0.0011744

Area using Simpson's 3/8 rule: 0.00441749
Absolute error: 0.00146834

Trapezoidal rule is the best.
1. Enter new value
2. Try with new value of h
0. Exit
   Enter your choice: 0

Process returned 0 (0x0)   execution time : 375.463 s
Press any key to continue.
```

Discussion:

For the advantages of Trapezoidal rule, it is more accurate approximation than a single Riemann sum. It can be used regardless if we have even or odd number of subintervals. Its concept and derivation of formula is easier than the Simpson’s rule using average for two consecutive function values. As for its disadvantages, it is less accurate than the Simpson’s rule. This method is preferred when there are odd numbers of subintervals.
For the advantages of Simpson’s rule, it is more accurate due to the use of parabolic top. It also achieves higher level of accuracy faster using less number of sub-intervals. For its disadvantages, it is restricted to even subintervals since it has to consider a pair of subintervals for each parabolic top. Thus, this method is preferred when there is quadratic polynomial or even number of subintervals.