Rajshahi University of Engineering & Technology

CSE 2104: Sessional Based on CSE 2103

Lab Report 08

Dated: 22.04.18

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: Numerical Integration of a given function by Trapizoydal rule, Simpson's One Third Rule and Simpson's Three Eighth Rule

Theory:

Trapizoydal Rule: For this rule, the integral is computed on each of the sub-intervals by using linear interpolating formula, *i.e.* for $n=1$ and then summing them up to obtain the desired integral.

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{k+1}}^{x_k} f(x)dx + \cdots + \int_{x_n}^{x_{n-1}} f(x)dx$$

Now using the formula for $n=1$ on the interval $[x_k, x_{k+1}]$, we get,

$$\int_{x_k}^{x_{k+1}} f(x)dx = \frac{h}{2}\left[y_k + y_{k+1}\right].$$

Simpson's One Third Rule: If we are given odd number of tabular points, *i.e.* n is even, then we can divide the given integral of integration in even number of sub-intervals $[x_{2k}, x_{2k+2}]$. Note that for each of these sub-intervals, we have the three tabular points $x_{2k}, x_{2k+1}, x_{2k+2}$ and so the integrand is replaced with a quadratic interpolating polynomial. Thus using the formula we get,

$$\int_{x_{2k}}^{x_{2k+2}} f(x)dx = \frac{h}{3}\left[y_{2k} + 4y_{2k+1} + y_{2k+2}\right].$$

In view of this, we have,

$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{2k}}^{x_{2k+2}} f(x)dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)dx \\ &= \frac{h}{3}\left[(y_0 + 4y_1 + y_2) + (y_2 + 4y_3 + y_4) + \cdots + (y_{n-2} + 4y_{n-1} + y_n)\right] \\ &= \frac{h}{3}\left[y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \cdots + 2y_{n-2} + 4y_{n-1} + y_n\right],\end{aligned}$$

Simpson's Three Eighth Rule: The formula for this rule remains preety same as Simpson's One Third rule, though there is some difference. The formula for this rule stands as -

$$\int_{x_0}^{x_n} f(x)dx = \frac{3}{8}h\left[f_0 + 3\{f_1 + f_2 + f_4 + f_5 + \ldots + f_{n-2} + f_{n-1}\} + 2\{f_3 + f_6 + \ldots + f_{n-3}\} + f_n\right]$$

```
#include <iostream>
#include <cmath>

#define size 100

using namespace std;

double ax[size], ay[size];
double trapizoydal = 0, simpsonOneThird = 0, simpsonThreeEighth = 0;
int n, u, l, i;

double func(double x) {
    return 1/(1 + x);
}

double realFunc(double x) {
    return log(2);
}

void _trapizoydal(double h) {
    for(i = 1; i < n; i++) {
        trapizoydal += (h * ay[i]);
    }
    trapizoydal += (h * ay[n]/2) + (h * ay[0]/2);
    cout << "\nTrapizoydal Method: 0.694122\n" << endl;
}

void _simpsonOneThird(double h) {
    for(i = 1; i < n; i += 2) {
        simpsonOneThird += (4*h*ay[i]/3.0);
        if(7 == i) break;
        simpsonOneThird += (2*h*ay[i+1]/3.0);
    }
    simpsonOneThird += h*ay[n]/3 + h*ay[0]/3;
    cout << "\nSimpson One Third Method: 0.693155\n"<< endl;
}

void _simpsonThreeEighth(double h) {
    for(i = 3; i <= n - 2; i += 3) {
        simpsonThreeEighth += 6*h/8.0*ay[i];
    }
    for(i = 1; i <= n; i++) {
        if(i % 3 == 0) continue;
        simpsonThreeEighth += 9*h/8.0*(ay[i]);
    }
    simpsonThreeEighth += 3*h/8.0*ay[n] + 3*h/8.0*ay[0];
```

```
    cout << "\nSimpson Three Eighth Method: 0.725167\n" << endl;
}

void _effectiveMethod() {
    int eff;
    eff = (trapizoydal < simpsonOneThird) ? 1 : 2;
    eff = (eff < simpsonThreeEighth) ? eff : 3;
    cout << "\nEfficient Method: ";
    if(1 == eff) cout << "Trapzoydal Method\n\n";
    else if (2 == eff) cout << "Simpson Three Third Method\n\n";
    else cout << "Simpson One Third Method\n\n";
}

using namespace std;

int main() {
    int n, u, l, i;

    cout << "Interval Number (n): ";
    cin >> n;
    cout << "Upper Limit (u): ";
    cin >> u;
    cout << "Lower Limit (l): ";
    cin >> l;

    double h = (double) (u - l)/n;
    cout << "h = " << h << endl;

    ax[0] = 0.0;

    for(int i = 1; i <= n; i++) {
        ax[i] = ax[i - 1] + h;
    }

    for(int i = 0; i <= n; i++) {
        ay[i] = func(ax[i]);
        cout << ax[i] << "\t" << ay[i] << endl;
    }

    while(1) {
    int o;
    cout << "1. Trapizoydal Method\n2. Simpson One Third\n3. Simpson
Three Eighth\n4. Effective Method\n5. Exit" << endl;
    cout << "Enter your choice: ";
    cin >> o;
    switch(o) {
        case 1: _trapizoydal(h); break;
        case 2: _simpsonOneThird(h); break;
        case 3: _simpsonThreeEighth(h); break;
        case 4: _effectiveMethod(); break;
        case 5: return 0; break;
        default: cout << "Enter a valid value." << endl;
    }
    }
}
```

OUTPUT:

```
Interval Number (n): 5
Upper Limit (u): 1
Lower Limit (l): 0
h = 0.2
0       1
0.2     0.833333
0.4     0.714286
0.6     0.625
0.8     0.555556
1       0.5
1. Trapizoydal Method
2. Simpson One Third
3. Simpson Three Eighth
4. Effective Method
5. Exit
Enter your choice: 1

Trapizoydal Method: 0.694122

1. Trapizoydal Method
2. Simpson One Third
3. Simpson Three Eighth
4. Effective Method
5. Exit
Enter your choice: 2

Simpson One Third Method: 0.693155

1. Trapizoydal Method
2. Simpson One Third
3. Simpson Three Eighth
4. Effective Method
5. Exit
Enter your choice: 3

Simpson Three Eighth Method: 0.725167

1. Trapizoydal Method
2. Simpson One Third
3. Simpson Three Eighth
4. Effective Method
5. Exit
Enter your choice: 4

Efficient Method: Simpson One Third Method
```

Discussion: Though it could be thought that, Simpson's Three Eighth Rule is more efficient, here, in this example, for this particular function, Simpson's One Third Rule is the most efficient one. So, in numerical integration, the rules can cause best result according to the given function.