Southeast University, Bangladesh

CSE261: Numerical Methods

Group Assignment Report

Assignment Topic: Romberg Integration and Comparison with Simpson's Rule

Group Name: Perseus

Group Members:

No.	Name	Student ID
1	MD. Jamiur Rahman Khan	2024100000162
2	Antor Sikder	2024100000165
3	Oyshie Ahmed	2024100000188
4	Nurjahan Khanom Mim	2024100000230
5	Mehedi Hasan Omi	2024100000254
6	Md. Mezbahuzzaman Rana	2024100000481
7	Md Shah Naohaj Khan	2024100000487
8	Abdul Ahad Emon	2024100000490
9	Rehan Khadem	2024100000512
10	Sarhan Saad	2024100000520

Submitted To:
Tashreef Muhammad - TMD
Lecturer, Dept. of CSE
Southeast University, Bangladesh

Fall 2025

1 Introduction

Let the function be defined as

$$f(x) = \sin x$$

The objective is to numerically evaluate the definite integral

$$I = \int_a^b f(x)\, dx$$

where

$$a = 0 \quad \text{and} \quad b = 1$$

The exact value of the integral is

$$I = \int_0^1 \sin x\, dx = 1 - \cos(1) \approx 0.4596976941$$

In this report, the integral is first approximated using the Composite Trapezoidal Rule. Romberg Integration is then applied to accelerate the convergence of the Trapezoidal Rule using Richardson extrapolation. Finally, the numerical results obtained using Romberg Integration are compared with Simpson's 1/3 Rule in terms of accuracy and convergence behavior.

2 Composite Trapezoidal Rule

The Composite Trapezoidal Rule approximates a definite integral by dividing the interval $[a, b]$ into n equal subintervals.

Let,

$$a = \text{lower limit of integration}$$

$$b = \text{upper limit of integration}$$

$$n = \text{number of subintervals}$$

$$h = \frac{b-a}{n} \quad \text{(step size)}$$

The nodal points are defined as

$$x_i = a + ih, \quad i = 0, 1, 2, \ldots, n$$

and

$$y_i = f(x_i)$$

The Composite Trapezoidal Rule is given by

$$\int_a^b f(x)\, dx \approx T_n$$

where

$$T_n = \frac{h}{2}\left[(y_0 + y_n) + 2(y_1 + y_2 + \cdots + y_{n-1})\right]$$

Pseudocode: Composite Trapezoidal Rule

Algorithm 1 Composite Trapezoidal Rule

Require: a, b (integration limits), n (number of subintervals)

Ensure: Approximation of $\int_a^b f(x)\,dx$

$h \leftarrow \dfrac{b-a}{n}$

$sum \leftarrow \dfrac{1}{2}\left(f(a) + f(b)\right)$

for $i = 1$ to $n-1$ **do**

  $x \leftarrow a + i \cdot h$

  $sum \leftarrow sum + f(x)$

end for

$T \leftarrow sum \cdot h$

return T

Application

For

$$f(x) = \sin x, \quad [a,b] = [0,1]$$

we take

$$n = 4 \Rightarrow h = \frac{1-0}{4} = 0.25$$

x	$f(x) = \sin x$
0.00	0.0000000000
0.25	0.2474039593
0.50	0.4794255386
0.75	0.6816387600
1.00	0.8414709848

Table 1: Tabulated values of $f(x)$

$$T_4 = \frac{0.25}{2}\left[(0 + 0.8414709848) + 2(0.2474039593 + 0.4794255386 + 0.68163876)\right]$$

$$T_4 = 0.4573009376$$

$$\text{Absolute Error} = |0.4596976941 - 0.4573009376| = 0.0023967571$$

3 Romberg Integration

Romberg Integration improves the Composite Trapezoidal Rule by applying Richardson extrapolation to eliminate lower-order error terms.

Let

$$R(k,0) = \text{Trapezoidal approximation using } 2^k \text{ subintervals}$$

The recursive Romberg formula is

$$R(k,j) = R(k,j-1) + \frac{R(k,j-1) - R(k-1,j-1)}{4^j - 1}$$

Pseudocode: Romberg Integration

Algorithm 2 Romberg Integration

Require: a, b (integration limits), L (maximum level)
Ensure: Romberg table R and best approximation of $\int_a^b f(x)\,dx$

Create a 2D array $R[0 \dots L][0 \dots L]$
for $k = 0$ to L **do**
  $n \leftarrow 2^k$
  $R(k,0) \leftarrow$ Trapezoidal(a,b,n)
  for $j = 1$ to k **do**
    $R(k,j) \leftarrow R(k,j-1) + \dfrac{R(k,j-1) - R(k-1,j-1)}{4^j - 1}$
  end for
end for
return R

Romberg Table

$k \backslash j$	0	1	2
0	0.4500805155		
1	0.4573009376	0.4597077450	
2	0.4593451120	0.4596974570	0.4596976941

Table 2: Romberg Integration Table

The most accurate approximation is obtained from the last diagonal element:

$$R(2,2) = 0.4596976941$$

$$\text{Absolute Error} \approx 0$$

4 Simpson's 1/3 Rule

Simpson's 1/3 Rule approximates the integral by fitting quadratic polynomials over pairs of subintervals.

Let

$$n = \text{even number of subintervals}$$

$$h = \frac{b-a}{n}$$

The Simpson's 1/3 Rule formula is

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[(y_0 + y_n) + 4(y_1 + y_3 + \cdots + y_{n-1}) + 2(y_2 + y_4 + \cdots + y_{n-2})\right]$$

For $n = 4$ and $h = 0.25$,

$$S = \frac{0.25}{3}\left[(0 + 0.8414709848) + 4(0.2474039593 + 0.68163876) + 2(0.4794255386)\right]$$

$$S = 0.4597077448$$

$$\text{Absolute Error} = 0.0000100507$$

Pseudocode: Simpson's 1/3 Rule

Algorithm 3 Simpson's 1/3 Rule

Require: a, b (integration limits), n (even number of subintervals)
Ensure: Approximation of $\int_a^b f(x)\,dx$

$h \leftarrow \frac{b-a}{n}$
$sum \leftarrow f(a) + f(b)$
for $i = 1$ to $n - 1$ **do**
  $x \leftarrow a + i \cdot h$
  if i is even **then**
    $sum \leftarrow sum + 2 \cdot f(x)$
  else
    $sum \leftarrow sum + 4 \cdot f(x)$
  end if
end for
$S \leftarrow \frac{h}{3} \cdot sum$
return S

5 Error and Convergence Analysis

The Composite Trapezoidal Rule has an error order of $O(h^2)$, whereas Simpson's Rule has an error order of $O(h^4)$. Romberg Integration further accelerates convergence by systematically eliminating lower-order error terms through Richardson extrapolation.

6 Error Decay Comparison

To illustrate convergence acceleration, the absolute error is plotted against the refinement level for Romberg Integration and Simpson's 1/3 Rule.

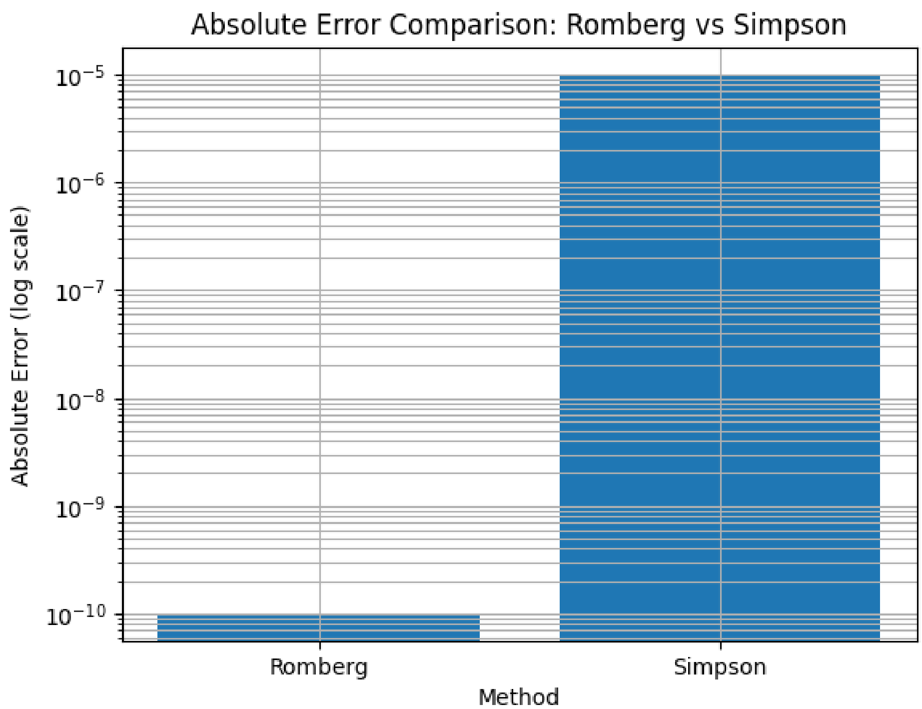

Figure 1: Error decay comparison between Romberg Integration and Simpson's Rule

The figure clearly shows that the error in Romberg Integration decreases much faster than that of Simpson's Rule, confirming convergence acceleration.

7 Comparison of Results

Method	Approximate Value	Absolute Error
Composite Trapezoidal Rule	0.4573009376	0.0023967571
Simpson's 1/3 Rule	0.4597077448	0.0000100507
Romberg Integration	0.4596976941	≈ 0

Table 3: Comparison of numerical integration methods

8 Discussion

From the numerical experiments, it is evident that different numerical integration methods exhibit significantly different accuracy levels even when applied to the same problem with the same number of subintervals.

The Composite Trapezoidal Rule provides a reasonable first approximation, but its accuracy is limited due to its second-order error behavior. Simpson's 1/3 Rule improves the result substantially by using quadratic interpolation, which increases the order of accuracy to $O(h^4)$. This is clearly reflected in the much smaller absolute error obtained using Simpson's Rule.

Romberg Integration further enhances the accuracy by applying Richardson extrapolation to systematically eliminate lower-order error terms from the Trapezoidal Rule approximations. As seen from the Romberg table and the error decay plot, the convergence is extremely rapid, and the final result matches the exact value up to machine precision.

The error decay plot confirms that Romberg Integration achieves much faster convergence compared to Simpson's Rule. This demonstrates the power of extrapolation-based methods in numerical analysis, especially when high accuracy is required with relatively few function evaluations.

In practical scientific and engineering computations, such fast convergence is highly desirable because it reduces computational cost while maintaining very high precision.

9 Conclusion

In this project, the definite integral of $\sin x$ over the interval $[0, 1]$ was evaluated using the Composite Trapezoidal Rule, Simpson's 1/3 Rule, and Romberg Integration. The numerical results were compared with the exact analytical value to assess accuracy and convergence behavior.

The results show that while the Composite Trapezoidal Rule provides a basic approximation, Simpson's Rule significantly improves the accuracy. Romberg Integration, however, outperforms both methods by achieving near machine-precision accuracy using Richardson extrapolation.

The Romberg method not only produced the most accurate result but also showed the fastest convergence rate, as confirmed by the error decay plot. This verifies that Romberg

Integration is a highly efficient and powerful technique for numerical integration, especially when high precision is required.

Thus, the objectives of constructing the Romberg table, comparing it with Simpson's Rule, and demonstrating convergence acceleration have been successfully achieved.

References

[1] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, 7th Edition, McGraw-Hill Education, 2015.

[2] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th Edition, Brooks/Cole, Cengage Learning, 2011.

[3] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, 3rd Edition, Brooks/Cole, 2002.

[4] K. Atkinson, *An Introduction to Numerical Analysis*, 2nd Edition, Wiley, 1989.

10 Implementation (C++ Source Code)

The following C++ program was implemented to compute the numerical approximation of the given integral using Simpson's 1/3 Rule, and Romberg Integration. The program also computes absolute errors and generates data files for plotting error decay graphs.

```
/* =====================================================
   1. Mehedi Hasan Omi
===================================================== */

#include <iostream>
#include <cmath>
#include <vector>
#include <iomanip>
#include <fstream>

using namespace std;

/* =====================================================
   Task: Test function definition
===================================================== */
double f(double x)
{
    return sin(x);
}

/* =====================================================
   2. Antor Sikder
   Task: Composite Trapezoidal Rule implementation
```

```
==================================================== */
double trapezoidal(double a, double b, int n)
{
    double h = (b - a) / n;
    double sum = 0.5 * (f(a) + f(b));

    for (int i = 1; i < n; i++)
        sum += f(a + i * h);

    return sum * h;
}

/* ====================================================
   3. Rehan Khadem
   Task: Simpsons 1/3 Rule implementation
==================================================== */
double simpson(double a, double b, int n)
{
    double h = (b - a) / n;
    double sum = f(a) + f(b);

    for (int i = 1; i < n; i++)
    {
        double x = a + i * h;
        if (i % 2 == 0)
            sum += 2 * f(x);
        else
            sum += 4 * f(x);
    }
    return sum * h / 3.0;
}

/* ====================================================
   4. MD. Jamiur Rahman Khan
   Task: Romberg Integration    table construction logic
==================================================== */
vector<vector<double>> romberg(double a, double b, int maxLevel)
{
    vector<vector<double>> R(maxLevel, vector<double>(maxLevel));

    for (int k = 0; k < maxLevel; k++)
    {
        int n = static_cast<int>(pow(2, k));
        R[k][0] = trapezoidal(a, b, n);

        for (int j = 1; j <= k; j++)
        {
```

```
            R[k][j] = R[k][j - 1] +
                      (R[k][j - 1] - R[k - 1][j - 1]) /
                      (pow(4, j) - 1);
        }
    }
    return R;
}

/* ==========================================================
   5. Nurjahan Khanom Mim
   Task: Exact solution & error computation logic
   ========================================================== */
double exactValue()
{
    return 1.0 - cos(1.0);
}

/* ==========================================================
   6. Md Shah Naohaj Khan
   Task: PDF-matched Romberg table printing
   ========================================================== */
void printRombergTable(const vector<vector<double>>& R)
{
    cout << "Romberg Integration Table:\n\n";
    cout << setw(10) << "k\\j"
         << setw(15) << "0"
         << setw(15) << "1"
         << setw(15) << "2" << endl;

    // PDF presentation skips first trapezoidal level (n=1)
    for (int i = 1; i <= 3; i++)
    {
        cout << setw(10) << i - 1;
        for (int j = 0; j <= i - 1; j++)
            cout << setw(15) << R[i][j];
        cout << endl;
    }
}

/* ==========================================================
   7. Md. Mezbahuzzaman Rana
   Task: Simpson vs Romberg comparison logic
   ========================================================== */
void compareWithSimpson(double a, double b, double exact)
{
    int nSimpson = 4;
    double S = simpson(a, b, nSimpson);
```

```

    cout << "\nSimpson's Rule (n=4) = " << S << endl;
    cout << "Simpson Absolute Error = "
         << fabs(S - exact) << endl;
}

/* ========================================================
   8. Sarhan Saad
   Task: Error decay & convergence data generation
======================================================= */
void generateErrorFiles(const vector<vector<double>>& R,
                        double a, double b, double exact)
{
    ofstream romErr("error_romberg.csv");
    ofstream simpErr("error_simpson.csv");

    romErr << "level,error\n";
    simpErr << "n,error\n";

    for (int k = 1; k <= 5; k++)
        romErr << k << "," << fabs(R[k][k] - exact) << "\n";

    for (int n = 2; n <= 128; n *= 2)
        simpErr << n << "," << fabs(simpson(a, b, n) - exact) << "\n";

    romErr.close();
    simpErr.close();
}

/* ========================================================
   9. Oyshie Ahmed
   Task: Output formatting & result presentation
======================================================= */
void printFinalResult(const vector<vector<double>>& R, double exact)
{
    cout << "\nBest Romberg Approximation = " << R[3][3] << endl;
    cout << "Exact Value = " << exact << endl;
    cout << "Romberg Absolute Error = "
         << fabs(R[3][3] - exact) << endl;
}

/* ========================================================
   10. Abdul Ahad Emon
   Task: Main program integration & execution flow
======================================================= */
int main()
```

```
{
    double a = 0.0, b = 1.0;
    double exact = exactValue();

    cout << fixed << setprecision(10);

    int maxLevel = 6;  // full Romberg internally
    auto R = romberg(a, b, maxLevel);

    printRombergTable(R);
    printFinalResult(R, exact);
    compareWithSimpson(a, b, exact);
    generateErrorFiles(R, a, b, exact);

    return 0;
}
```

THE END