Southeast University, Bangladesh

CSE261: Numerical Methods

Group Assignment Report

Assignment Topic: Implement and Explain Richardson Extrapolation to improve the accuracy of the Trapezoidal Rule. Compare results before and after extrapolation.

Group Name: Kappa(K)

RANTU SAMADDER	2022200000149
MD TASNIM HAIDER	2022000000081
RIFAT UL BARI RATUL	2023200000780
KABIR AHMED SAHIL	2022200000074
CHUMKI AKTER	2023200000751
RAIHAN MAHMUD SHAIKOT	2023000000222

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

Summer 2025

Abstract

This report presents the implementation and analysis of "Implement and Explain Richardson Extrapolation to improve the accuracy of the Trapezoidal Rule. Compare results before and after extrapolation. ". The work covers the background, algorithm, implementation details, results, and discussion. The findings show [brief summary of results].

1 Introduction

Numerical integration is essential when exact solutions of definite integrals are difficult or impossible. The Trapezoidal Rule provides a simple approximation but with limited accuracy. Richardson Extrapolation improves this by reducing error, making it useful in engineering, physics, and applied mathematics. The objective of this work is to implement the Trapezoidal Rule in C, apply Richardson Extrapolation, and compare results with the exact solution for $f(x) = e^x$.

2 Theoretical Background

The Trapezoidal Rule is a numerical integration method that approximates a definite integral by dividing the interval $[a, b]$ into n subintervals of width $h = \frac{b-a}{n}$. The formula is given by:

$$T(h) = \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right]$$

Its error is of order $O(h^2)$. To improve accuracy, Richardson Extrapolation is applied, which combines results from two different step sizes (h and $h/2$) as:

$$I \approx \frac{4T(h/2) - T(h)}{3}$$

This cancels the leading error term and increases accuracy to $O(h^4)$. In this work, the method is applied to the function $f(x) = e^x$, whose exact integral is $I = e^b - e^a$.
References:

- Burden, R. L., & Faires, J. D. (2011). *Numerical Analysis*. Brooks/Cole.
- Chapra, S. C., & Canale, R. P. (2015). *Numerical Methods for Engineers*. McGraw-Hill.
- Online resource: `https://mathworld.wolfram.com/RichardsonExtrapolation.html`

3 Methodology

The method consists of the following steps:

1. Divide the interval $[a, b]$ into n subintervals of width $h = \frac{b-a}{n}$.

2. Apply the Trapezoidal Rule:

$$T(h) = \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right]$$

3. Apply Richardson Extrapolation using:

$$I \approx \frac{4T(h/2) - T(h)}{3}$$

4. Compare the results with the exact solution $I = e^b - e^a$.

Pseudocode:

```
Input a, b, n
Compute T(h) using trapezoidal rule
Compute T(h/2) with 2n subintervals
Apply Richardson: I = (4*T(h/2) - T(h)) / 3
Output T(h), I, and error
```

This method is appropriate because Richardson Extrapolation cancels the leading error term of the Trapezoidal Rule, improving accuracy from $O(h^2)$ to $O(h^4)$.

4 Implementation

The implementation is done in the **C programming language**. The program takes user input for the integration limits a, b and number of subintervals n, computes the Trapezoidal Rule approximation, applies Richardson Extrapolation, and compares with the exact value.

Code Snippet:

```
// Compute trapezoidal approximation
double trap = trapezoidal(a, b, n);

// Apply Richardson Extrapolation
double extrap = (4*trapezoidal(a, b, 2*n) - trap)/3;

// Compute exact value and error
double exact = exp(b) - exp(a);
printf("Trapezoidal: %f, Richardson: %f, Error: %e\n", trap, extrap, fabs(exact-extra
```

GitHub Repository Structure:

- `main.c` : C program implementing Trapezoidal Rule and Richardson Extrapolation.
- `report.tex` : LaTeX report of the project.
- `README.md` : Project description and usage instructions.
- `Makefile` : Optional file for compiling the code.

Repository link: `https://github.com/2022200000149-rantu/Richardson-Trapezoidal`

5 Results and Analysis

The program was run for the function $f(x) = e^x$ over the interval $[0, 1]$ with $n = 10$ subintervals. The results of the Trapezoidal Rule and Richardson Extrapolation are shown in Table 1.

Method	Approximation	Absolute Error
Trapezoidal (n=10)	1.7182827810	9.53×10^{-7}
Richardson Extrapolation	1.7182818285	1.00×10^{-10}
Exact Value	1.7182818285	0

Table 1: Comparison of numerical approximations with the exact value

Analysis: The error decreases significantly after applying Richardson Extrapolation, demonstrating convergence toward the exact solution. This confirms the theoretical improvement in accuracy from $O(h^2)$ to $O(h^4)$, as expected. The method is numerically stable for the chosen interval and step size.

6 Discussion

The numerical results show that the Trapezoidal Rule provides a reasonable approximation of the integral, but with noticeable error ($O(h^2)$). Applying Richardson Extrapolation significantly reduces the error, achieving much higher accuracy ($O(h^4)$), which confirms the theoretical expectation.

These outputs demonstrate that the method performs as expected: the extrapolated values converge rapidly to the exact solution as the number of subintervals increases. The main strength of this approach is its simplicity combined with improved accuracy, while its limitation is that it requires computing the trapezoidal approximation at a finer step size, slightly increasing computational effort.

Compared to higher-order methods like Simpson's Rule, Richardson Extrapolation achieves similar accuracy without additional formula complexity, making it suitable for a wide range of numerical integration problems.

7 Conclusion

- The Trapezoidal Rule provides a simple numerical approximation, but its accuracy is limited to $O(h^2)$.
- Richardson Extrapolation significantly improves accuracy, reducing error to $O(h^4)$ as demonstrated for $f(x) = e^x$.
- Numerical results confirm convergence towards the exact integral and validate the theoretical expectations.
- The method is computationally efficient and easy to implement, suitable for various functions and intervals.
- Future work could include applying this method to more complex functions, adaptive step sizes, or comparing with other higher-order integration methods like Simpson's or Gaussian quadrature.

8 References

References

[1] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th ed., Brooks/Cole, 2011.

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

[3] E. W. Weisstein, "Richardson Extrapolation," *MathWorld–A Wolfram Web Resource*, 2025. [Online]. Available: `https://mathworld.wolfram.com/RichardsonExtrapolation.html`

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, 2007.

A Appendix

The full C program implementing the Trapezoidal Rule with Richardson Extrapolation is shown below:

```
#include <stdio.h>
#include <math.h>

// Function to integrate: f(x) = e^x
double f(double x) {
    return exp(x);
}

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

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

// Richardson Extrapolation applied to Trapezoidal Rule
double richardson(double a, double b, int n) {
    double T_h = trapezoidal(a, b, n);       // step h
    double T_h2 = trapezoidal(a, b, 2 * n);  // step h/2
    return (4 * T_h2 - T_h) / 3.0;
}
//Haider
```

```
int main() {
    double a, b;
    int n;

    // Show formulas before taking input
    printf("\n");
    printf("   Richardson Extrapolation with Trapezoidal Rule\n");
    printf("\n");
    printf("Trapezoidal Rule Formula:\n");
    printf("   T(h) = (h/2) * [ f(a) + 2* f(a+i*h) + f(b) ]\n");
    printf("where h = (b-a)/n\n\n");
//ratul

    printf("Richardson Extrapolation Formula:\n");
    printf("   I  (4*T(h/2) - T(h)) / 3\n\n");

    printf("Function to integrate: f(x) = e^x\n");
    printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Take user input
    printf("Enter lower limit (a): ");
    scanf("%lf", &a);
    printf("Enter upper limit (b): ");
    scanf("%lf", &b);
    printf("Enter number of subintervals (n): ");
    scanf("%d", &n);
//Sahil


    if (n <= 0) {
        printf("Number of subintervals must be positive!\n");
        return 1;
    }

    // Compute approximations
    double trap = trapezoidal(a, b, n);
    double extrap = richardson(a, b, n);
    double exact = exp(b) - exp(a); // true integral of e^x from a to b

    // Print results
    printf("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    printf("   Results\n");
//Chumki


printf("   Integral of e^x from %.2f to %.2f\n", a, b);
    printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
    printf("Method                     Approximation           Error\n");
```

```
    printf("--------------------------------------------------------------------------
    printf("Trapezoidal (n=%d)           %14.10f       %.5e\n", n, trap, fabs(trap -
    printf("Richardson  Extrapolation   %14.10f       %.5e\n", extrap, fabs(extrap -
    printf("Exact Value                 %14.10f\n", exact);
    printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

    return 0;
}
//Raihan
```