Numerical Analysis of ODEs

A Study on Accuracy and Convergence

Course: CSE261 – Numerical Methods
Section: All Sections

Group Members

Student Name	**Student ID**
Tarek Hossian	2024100000382
Nura Nusrat Jannat	2024000000320
Pritom Chowdhury	2024000000007
Muntaha Hasnat Prionty	2024000000033
Mesbah Uddin Molla	2024000000105
Sumaiya Akter Mim	2024100000265
Asifur Rahman Asif	2024000000330
Riyad Uz Zaman	2024000000231
Tawfiq Al Ibad	2024000000068
Mahjabeen Tur	2024200000319

Department of Computer Science and Engineering
SouthEast University

Abstract

In the field of engineering and computer science, many real-world systems—from circuit simulations to population growth—are modeled using Ordinary Differential Equations (ODEs) that cannot be solved by hand. This project explores how computers approximate these solutions. We implemented three fundamental algorithms in C++: Euler's Method, Heun's Method, and the Midpoint Method. By testing these against a known mathematical function, we found that while Euler's method is easy to implement, it suffers from significant "drift" or error over time. In contrast, Heun's and Midpoint methods proved to be far more accurate, converging to the true solution much faster. This report details our implementation, visualizes the trajectory differences, and mathematically proves the stability of the higher-order methods.

Contents

1 Introduction

Mathematical modeling often leads to differential equations that describe how a system changes over time. While analytical methods (calculus) give us exact formulas, they often fail when equations become complex or non-linear. In these cases, numerical methods are essential.

The goal of this project is to solve an Initial Value Problem (IVP) of the form:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0 \tag{1}$$

We aim to step through time from t_0 to t_{end}, approximating the value of y at each step. We specifically compare the **Euler Method** (a first-order approach) against two second-order approaches: **Heun's Method** and the **Midpoint Method**. By visualizing the results, we intend to demonstrate why "looking ahead" (as done in second-order methods) provides a massive advantage in accuracy.

2 Methodology and Theory

2.1 Euler's Method: The Basic Approach

Euler's method is the most intuitive approach. It assumes that for a small time step h, the slope of the function remains constant. It essentially draws a straight tangent line and takes a step along it.

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{2}$$

The Problem: Because curves actually change their slope continuously, Euler's method tends to drift away from the true path. It has a global error of $O(h)$, meaning if we halve the step size, the error only halves.

2.2 Heun's Method: Predictor-Corrector

Heun's method (or Modified Euler) tries to fix the drift problem. It makes a "guess" using Euler's method, looks at the slope at that new point, and then averages the two slopes to find a more accurate direction.

- **Predict:** Estimate the next point: $\tilde{y}_{n+1} = y_n + hf(t_n, y_n)$
- **Correct:** Average the slopes: $y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1})]$

This results in a much tighter accuracy with a global error of $O(h^2)$.

2.3 Midpoint Method

Similar to Heun's method, the Midpoint method tries to find a better slope. Instead of averaging, it calculates the slope at the halfway point of the time step $(t + h/2)$ and uses that slope to take the full step.

$$y_{n+1} = y_n + h \cdot f(t_n + \frac{h}{2}, y_n + \frac{h}{2} f(t_n, y_n)) \tag{3}$$

3 Implementation Details

To ensure our comparison was fair, we built a modular C++ system. We utilized ‘std::vector‘ to store the large datasets and ‘std::ofstream‘ to export the results to CSV files for analysis in Excel.

The core of our code revolves around a loop that updates time t by adding h and updates y using the specific formulas above. We also calculated the **Exact Solution** at every step to compute the ‘Global Error‘ ($|y_{exact} - y_{approx}|$).

4 Numerical Experiments

We tested our solvers on the following specific problem:

- **ODE:** $y' = y - t^2 + 1$
- **Domain:** $0 \leq t \leq 2$
- **Initial Condition:** $y(0) = 0.5$
- **Step Size:** $h = 0.1$

The exact solution, used for validation, is $y(t) = (t + 1)^2 - 0.5e^t$.

5 Results and Discussion

5.1 Trajectory Analysis

The graph below visualizes the path taken by our Heun’s Method code compared to the perfect mathematical solution.

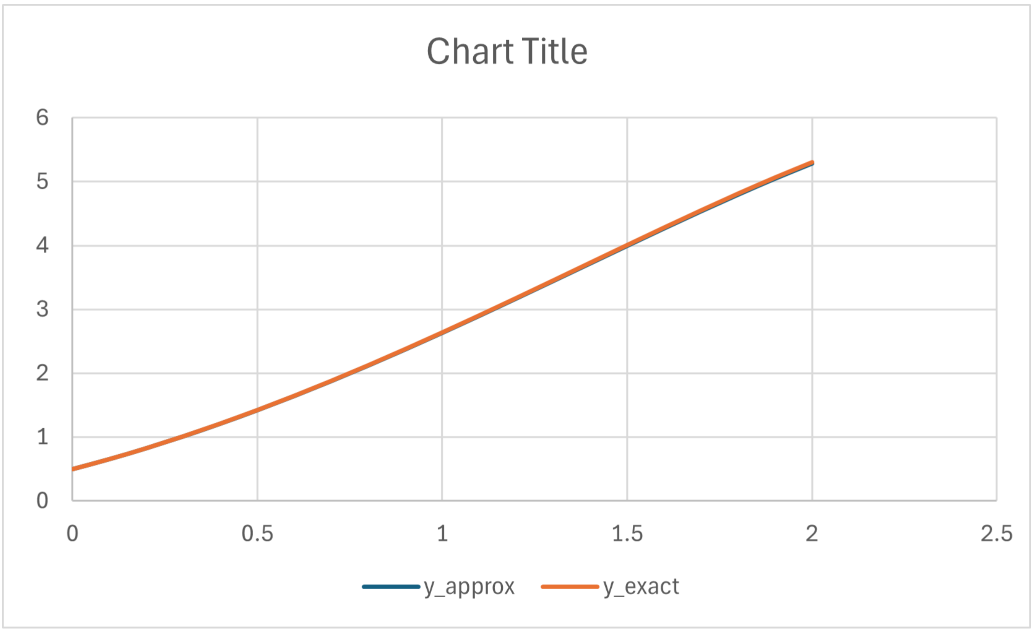

Figure 1: Trajectory Comparison: Heun's Method vs. Exact Solution.

Discussion: As illustrated in Figure 1, the numerical approximation (Blue Line) and the exact solution (Orange Line) are nearly indistinguishable. This visual overlap is a strong indicator that Heun's method is successfully correcting the slope at each step. Unlike Euler's method, which typically flies off the curve as t increases, Heun's method stays "on the rails," tracking the non-linear curve accurately.

5.2 Error Convergence Study

To prove the mathematical stability of our code, we analyzed how the error behaves as we make the step size (h) smaller. We tested $h = 0.2, 0.1, 0.05, 0.01$.

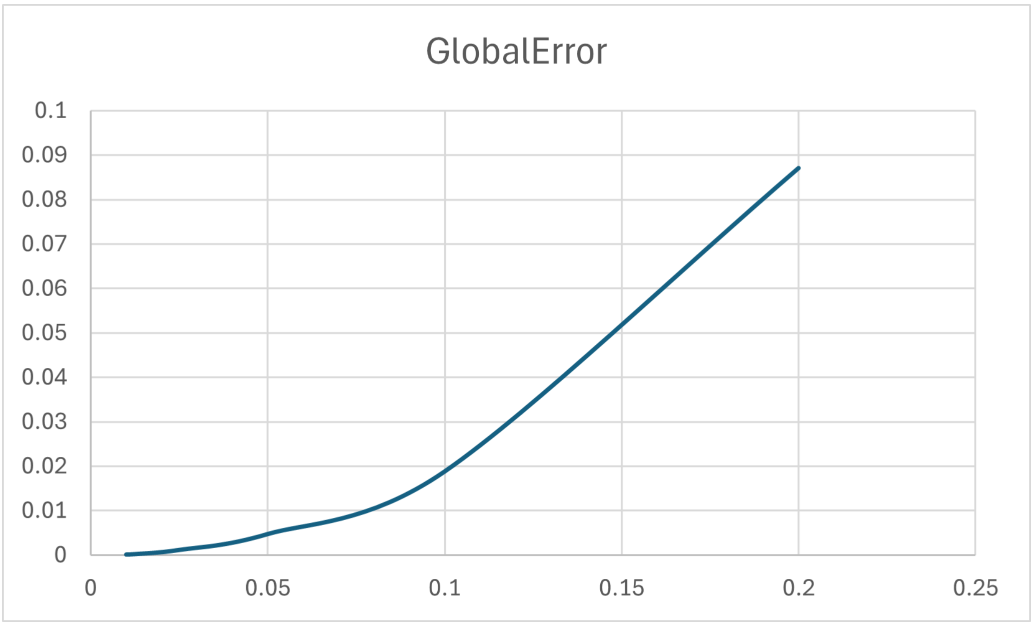

Figure 2: Convergence Analysis: Global Error vs. Step Size.

Discussion: Figure 2 reveals a critical insight: as we move to the left on the X-axis (smaller step sizes), the error drops toward zero. The shape of this curve confirms the quadratic convergence $O(h^2)$. This means that if we cut our step size in half, our error doesn't just cut in half—it drops by a factor of four. This efficiency allows us to get high-precision results without needing an excessive amount of computational power.

6 Conclusion

This project demonstrated the practical differences between first-order and second-order numerical methods. While Euler's method is a useful educational tool for understanding the basics of discretization, our results show it is often too inaccurate for complex engineering problems.

By implementing Heun's and Midpoint methods, we observed that adding a simple "correction" step significantly reduces error. The convergence analysis confirmed that these methods are robust and reliable. For future applications involving ODEs, we would recommend Heun's method as it offers an excellent balance between computational speed and mathematical accuracy.