

# MA3105: Numerical Analysis Final Project

Instructor: Koel Das

Debayan Sarkar <sup>\*</sup>  
22MS002

Diptanuj Sarkar <sup>†</sup>  
22MS038

November 26, 2024

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>ODE Solvers</b>	<b>2</b>
2.1	Forward Euler Method . . . . .	2
2.2	Runge-Kutta Methods . . . . .	2
2.2.1	Second Order . . . . .	2
2.2.2	Fourth Order . . . . .	4
<b>3</b>	<b>Problem Statement</b>	<b>5</b>
3.1	Given ODEs . . . . .	5
3.2	Direction Fields . . . . .	5
3.3	Analytical Solutions . . . . .	5
3.4	Initial Values . . . . .	6
<b>4</b>	<b>Solving the ODEs</b>	<b>6</b>
4.1	Observations and Comparisons . . . . .	6
4.2	Varying the Initial Value . . . . .	7
<b>5</b>	<b>Plotting the Errors</b>	<b>9</b>

## 1 Introduction

Ordinary Differential Equations (ODEs) are fundamental in modeling real-world phenomena across various fields of science and engineering. While analytical solutions are ideal, they are often unattainable for complex equations. Numerical methods, such as the

---

<sup>\*</sup>TheSillyCoder.github.io

<sup>†</sup>CasioWave.github.io

Forward Euler Method and Runge-Kutta Methods, provide practical alternatives. This report focuses on the Runge-Kutta methods, specifically the second and fourth-order schemes, and compares them to the Forward Euler Method.

## 2 ODE Solvers

In this section we implement the ODE solvers to compare them using the given differential equations. We will consider the differential equation given by the following equation and initial condition:

$$\boxed{\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0}$$

### 2.1 Forward Euler Method

This is the simplest way to solve ODEs numerically. This is a first order method. As in, the error in it grows as the square of the step size. The motivation for it has been shown below.

Let  $y(x)$  be a smooth function. Then, from its Taylor expansion we have

$$\begin{aligned} y(x+h) &= y(x) + y'(x)h + \sum_{n=2}^{\infty} y^{(n)}(x)h^n \\ \Rightarrow y(x+h) &= y(x) + y'(x)h + \mathcal{O}(h^2) \\ \Rightarrow y(x+h) &= y(x) + y'(x)h \quad (\text{Ignoring the second order terms}) \end{aligned}$$

This gives us the expression for iteratively calculating  $y(x)$  using the forward Euler Method:

$$y(x+h) = y(x) + hf(x, y)$$

Denoting them in terms of  $x_n$  and  $y_n$  we get the following expression:

$$\boxed{y_{n+1} = y_n + hf(x_n, y_n)}$$

### 2.2 Runge-Kutta Methods

Runge-Kutta methods are a family of iterative methods to solve ODEs which were first introduced in the 1900s. These methods improve upon the Forward Euler Method by considering intermediate slopes to achieve higher accuracy. The Runge-Kutta methods implemented here have been discussed in further detail below.

#### 2.2.1 Second Order

The most commonly used second order Runge-Kutta method is the midpoint method, where we improve upon the Euler Method using the slope of the midpoint. The algorithm

goes as follows. Given a step-size of  $h > 0$ , in each iteration we define the following:

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \end{aligned}$$

Now using these values we calculate the next term in the iteration using the following expression:

$$y_{n+1} = y_n + hk_2$$

Substituting the value of  $k_2$  in this expression we get:

$$y_{n+1} = y_n + hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}f(x_n, y_n)\right)$$

However this is not the only second order Runge-Kutta Method for solving an IVP. Any second order Runge-Kutta Method can be parameterised by  $\alpha$  and given by the formula:

$$y_{n+1} = y_n + h \left[ \left(1 - \frac{1}{2\alpha}\right) f(x_n, y_n) + \frac{1}{2\alpha} f(x_n + \alpha h, y_n + \alpha h f(x_n, y_n)) \right]$$

This formula has been derived below.

First let us define the following terms:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \alpha h, y_n + \beta k_1) \\ y_{n+1} &= y_n + ak_1 + bk_2 \end{aligned} \tag{1}$$

Where,  $a, b, \alpha, \beta$  are constants. Now, consider the Taylor series expansion of  $y(x)$  upto the second order.

$$\begin{aligned} y(x+h) &= y(x) + h \frac{dy}{dx} + \frac{h^2}{2} \frac{d^2y}{dx^2} + \mathcal{O}(h^3) \\ \Rightarrow y(x+h) &= y(x) + hf(x, y) + \frac{h^2}{2} \left( \frac{\partial f}{\partial x} + f(x, y) \frac{\partial f}{\partial y} \right) + \mathcal{O}(h^3) \\ \Rightarrow y_{n+1} &= y_n + hf(x_n, y_n) + \frac{h^2}{2} \left( \frac{\partial f}{\partial x} + f(x_n, y_n) \frac{\partial f}{\partial y} \right) + \mathcal{O}(h^3) \end{aligned} \tag{2}$$

We can also do a Taylor series expansion for  $k_2$  upto an error of  $\mathcal{O}(h^3)$ .

$$\begin{aligned} k_2 &= hf(x_n + \alpha h, y_n + \beta k_1) \\ &= h \left[ f(x_n, y_n) + \frac{\partial f}{\partial x} \alpha h + \frac{\partial f}{\partial y} \beta k_1 \right] + \mathcal{O}(h^3) \end{aligned}$$

Substituting this value of  $k_2$  into (1) we get,

$$y_{n+1} = y_n + h(a + b)f(x_n, y_n) + bh^2 \left[ \alpha \frac{\partial f}{\partial x} + \beta f(x_n, y_n) \frac{\partial f}{\partial y} \right] + \mathcal{O}(h^3) \quad (3)$$

Comparing (2) and (3) and equating the coefficients we get the following equations.

$$\alpha b = \frac{1}{2} \Rightarrow b = \frac{1}{2\alpha}$$

$$\beta b = \frac{1}{2} \Rightarrow \beta = \alpha$$

$$a + b = 1 \Rightarrow a = 1 - \frac{1}{2\alpha}$$

Then (1) can be re-written parameterised only in terms of  $\alpha$  as,

$$y_{n+1} = y_n + h \left[ \left( 1 - \frac{1}{2\alpha} \right) f(x_n, y_n) + \frac{1}{2\alpha} f(x_n + \alpha h, y_n + \alpha h f(x_n, y_n)) \right]$$

As is clear from this derivation, this method is valid upto an error of the order of  $\mathcal{O}(h^3)$ .

### Error Estimates

A brief discussion for the error estimates for the RK2 method.

#### Local Truncation Error: Fehlberg Methods

We simultaneously compare the numerical solutions for a lower order RK method and a higher order RK method. The local truncation error at the  $n^{th}$  point is given by,

$$T_n \approx \hat{y}_n - y_{n+1}$$

It can be shown to be the correct asymptotic result, as  $h \rightarrow 0$ .

#### Global Error: Richardson Extrapolation

This involves solving a particular Runge-Kutta method at step sizes of  $h$  and  $2h$  then the difference between the numerically obtained solutions of these two different step sizes can be used with richardson extrapolation to derive an estimation of the local truncation error in, the numerical solution for step-size  $h$ . For RK2 in particular it is given by:

$$y(x_n) - y_h(x_n) \approx \frac{1}{3} [y_h(x_n) - y_{2h}(x_n)]$$

### 2.2.2 Fourth Order

The fourth order Runge-Kutta Method, also known as the classic Runge-Kutta Method is the most widely known member of the Runge-Kutta Family, and is known as 'RK4'.

The algorithm goes as follows. Given a step-size of  $h > 0$ , in each iteration we define the following:

$$\begin{aligned}k_1 &= f(x_n, y_n) \\k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\k_4 &= f(x_n + h, y_n + hk_3)\end{aligned}$$

Now, using these intermediate slopes, we calculate the next term in the iteration using the following expression:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

A derivation similar to that of RK2 can be done for this as well and it can be shown that RK4 is a fourth order method i.e. the error in the numerically obtained solution from the analytical solution is of the order of  $\mathcal{O}(h^5)$ .

### 3 Problem Statement

#### 3.1 Given ODEs

We were asked to solve the following differential equations using the aforementioned ODE solvers.

$$\begin{aligned}\frac{dy}{dx} &= y - x \\ \frac{dy}{dx} &= y - x^2\end{aligned}$$

#### 3.2 Direction Fields

Direction Fields help us to visualise how the solutions of the differential equation will look, before actually going about solving them.

#### 3.3 Analytical Solutions

The analytical solutions to these equations can be computed by converting these into exact differentials using the integrating factor  $\mu(x) = e^{-x}$ . Solving these equations with the initial condition  $y(0)$  gives us the solutions

$$\begin{aligned}y_1(x) &= 1 + x + (y(0) - 1)e^x \\ y_2(x) &= 2 + 2x + x^2 + (y(0) - 2)e^x\end{aligned}$$

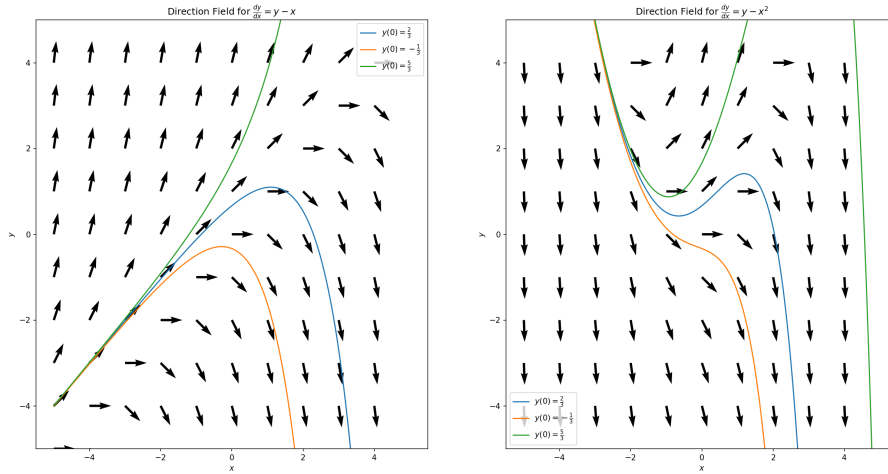


Figure 1: Direction Fields for the given ODEs

### 3.4 Initial Values

The initial values provided for both these equations is  $y(0) = \frac{2}{3}$

## 4 Solving the ODEs

The given IVPs were solved using the priorly mentioned numerical methods, implemented in python. A plot of the solutions obtained on the interval  $x \in [0, 5]$  has been included below.

### 4.1 Observations and Comparisions

Looking at the plots, we can compare the methods and conclude the following:

- **Forward Euler Method:** Produced noticeable deviations from the exact solution, and as the value of  $x$  grew, the deviation became larger.
- **RK2 Method:** Significantly reduced errors compared to Forward Euler. The inclusion of intermediate slopes provided better stability and accuracy.
- **RK4 Method:** Delivered solutions nearly indistinguishable from the exact solution. The higher accuracy of  $\mathcal{O}(h^5)$  made RK4 the most reliable among the three methods for the given ODEs.

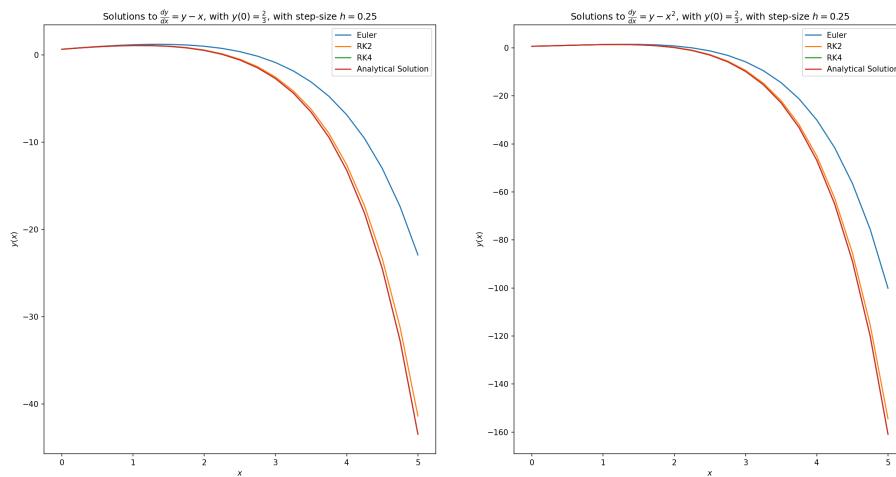


Figure 2: Plot of the solutions of the ODEs using the priorly mentioned numerical methods

## 4.2 Varying the Initial Value

We varied the initial value  $y(0)$  and plotted the analytical solution and RK4 solutions to see how they compare. The step-size taken for the solutions was  $h = 0.025$ .

As we can see in the above figure, the RK4 solutions still coincide with the analytical solutions.

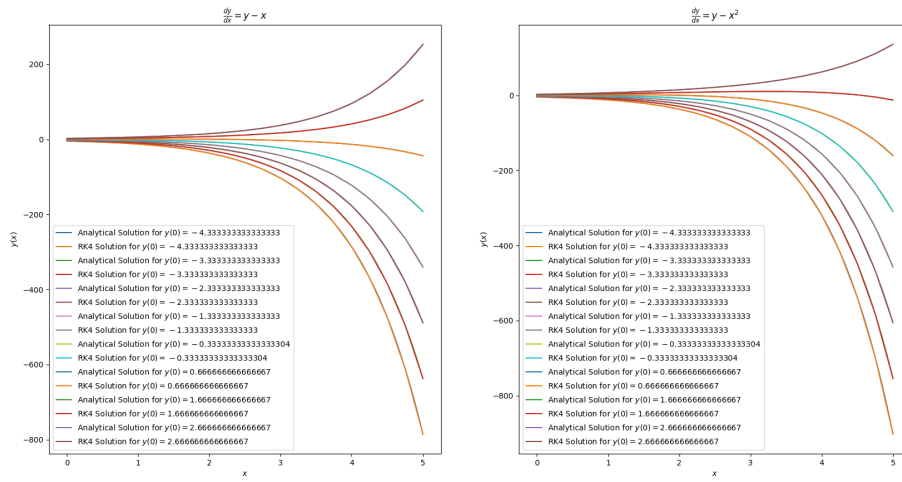


Figure 3: Plots of the Solutions of the given Differential Equations with varying values of  $y(0)$



## 5 Plotting the Errors

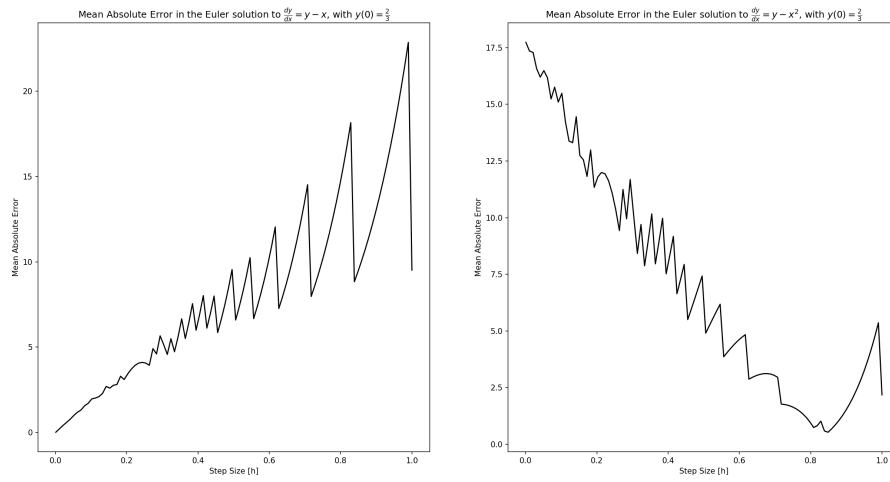


Figure 4: Plots of Mean Absolute Error vs Step-size for Euler Method

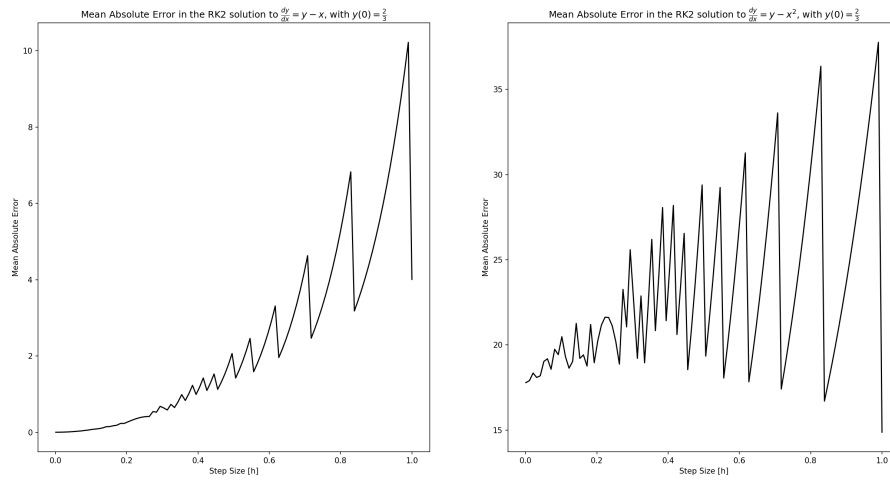


Figure 5: Plots of Mean Absolute Error vs Step-size for RK2 Method

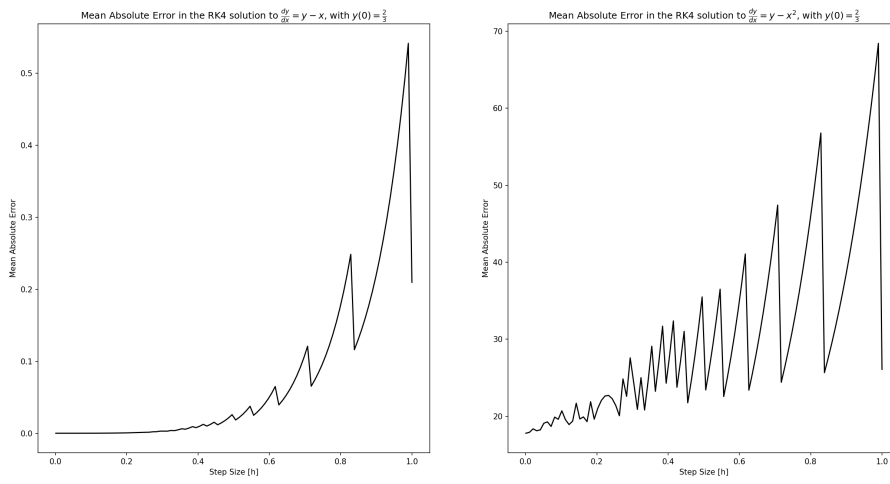


Figure 6: Plots of Mean Absolute Error vs Step-size for RK4 Method