Numerical Solutions to an Initial Value Problem Using Euler, Taylor, and Runge-Kutta 6th Order Methods

Celine Al Harake and Layal Canoe
Supervised by: Dr. Mohammed Moussa
Computer Science Department, Effat University

May 5, 2025

Abstract :

This report presents numerical solutions to the differential equation $y' = 3e^2xy$ using three numerical methods: Euler's method, the second-order Taylor method, and the sixth-order Runge-Kutta method. Each method is explained theoretically and implemented in Python. The numerical results are presented in tabular and graphical form for comparison.

Index Terms: Euler's method, Taylor Method (2nd-order), Runge-Kutta Method (6th-order), Numerical Analysis, Python.

1 Introduction

Solving differential equations is a fundamental task in many fields of science and engineering. While some differential equations can be solved analytically, many real-world problems lead to equations that cannot be solved using traditional symbolic methods. In such cases, numerical methods provide practical and efficient techniques to approximate the solutions.

This report focuses on solving a first-order ordinary differential equation (ODE) given by:

$$y' = 3e^{2x}y, y(0) = 1$$

Using three different numerical approaches:

1. Euler's Method
2. Taylor Method (2nd order)
3. Runge-Kutta Method (6th order)

By exploring different numerical techniques on the same initial value problem (IVP), this report aims to highlight the trade-offs between simplicity, accuracy, and computational effort involved in various ODE solvers.

2 Theoretical Background

2.1 Euler's method

Euler's method is the simplest numerical technique for solving first-order initial value problems (IVPs) of the form:

$$y' = f(x, y), y(x_0) = y_0$$

It is an explicit one-step method based on the idea of using the derivative (slope) at a known point to estimate the value of the function at the next point. The general formula for Euler's method is:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Where:

- x_i: is the current value of x
- y_i: is the current approximation of $y(x_i)$
- h: is the step size
- $f(x_i, y_i)$: is the slope at the current point

The method works by incrementally stepping from the initial point (x_0, y_0) to the desired final value of x, using the slope at each step to approximate the next value of y.

It offers several advantages, including its ease of understanding and implementation. Additionally, it is computationally inexpensive. However, there are limitations, such as low accuracy for large step sizes and the accumulation of errors due to its linear approximation nature.

Application to Our Problem

For the differential equation:

$$y' = 3e^{2x}y, y(0) = 1$$

The update rule becomes:

$$y_{i+1} = y_i + h * (3e^{2x_n}y_n)$$

2.2 Taylor Method (2nd-order)

Taylor's method is a powerful numerical technique that improves the accuracy of solutions to differential equations by including higher-order terms from the Taylor series expansion.

For a first-order ODE of the form:

$$y' = f(x, y), y(x_0) = y_0$$

The second-order Taylor method uses the first two terms of the Taylor series:

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{h^2}{2}f'(x_i, y_i)$$

Where:

- $f(x_i, y_i)$: is the first derivative given by the ODE
- $f'(x_i, y_i)$: is the total derivative of f with respect to x, which includes both partial derivatives with respect to x and y

Application to Our Problem

Given:

$$y' = 3e^{2x}y, y(0) = 1$$

We compute the total derivative $f(x_i, y_i)$ using the product rule:

$$f'(x_i, y_i) = \frac{d}{dx}[3e^{2x}y] = \frac{d}{dx}[3e^{2x}] * y + 3e^{2x} * \frac{dy}{dx}$$

$$f'(x_i, y_i) = 6e^{2x}y + 9e^{4x}y$$

Now substitute into the Taylor method formula:

$$y_{i+1} = y_i + h(3e^{2x_n}y_n) + \frac{h^2}{2}(6e^{2x_i}y_i + 9e^{4x_i}y_i)$$

This method is more accurate than Euler's method for the same step size because it incorporates the curvature of the solution by including the second derivative information.

2.3 Runge-Kutta Method (6th-order)

Runge-Kutta (RK) methods are a family of iterative techniques that provide highly accurate numerical solutions to ordinary differential equations (ODEs). Among them, higher-order RK methods (like 4th and 6th order) are widely used due to their balance between accuracy and computational efficiency.

The 6th-order Runge-Kutta method is an explicit, single-step method that approximates the solution of the initial value problem:

$$y' = f(x, y), y(x_0) = y_0$$

It computes several intermediate slopes (called stages) to estimate the next value of y. The general form is:

$$y_{i+1} = y_i + b_1k_1 + b_2k_2 + ... + b_6k_6$$

Where:

- k_i: is a weighted evaluation of the function $f(x, y)$ at various points between x_i and $x_i + h$.
- b_i: is a weight that depends on the chosen 6th-order scheme.

One standard form of the 6th-order Runge-Kutta method uses the following stages:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + \frac{h}{3}, y_i + \frac{h}{3}k_1)$$

$$k_3 = f(x_i + \frac{h}{3}, y_i + \frac{h}{6}k_1 + \frac{h}{6}k_2)$$

$$k_4 = f(x_i + \frac{h}{2}, y_i + \frac{h}{8}k_1 + \frac{3h}{8}k_3)$$

$$k_5 = f(x_i + \frac{2h}{3}, y_i + \frac{h}{2}k_1 - \frac{3h}{2}k_3 + k_4)$$

$$k_6 = f(x_i + h, y_i - \frac{3h}{7}k_1 + \frac{8h}{7}k_2 + \frac{6h}{7}k_3 - \frac{12h}{7}k_4 + \frac{8h}{7}k_5)$$

Final explicit formulation,

$$y_{i+1} = y_i + \frac{h}{90}(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6)$$

Application to Our Problem

Given:

$$y' = 3e^{2x}y, y(0) = 1$$

Substitute this into the formulas for k_1 through k_6, using the exponential evaluations at each sub-step. This method offers extremely high accuracy, often comparable to much smaller step sizes in lower-order methods like Euler or 2nd-order Taylor.

3 Python Code

3.1 Euler's method

```python
import numpy as np

# Differential equation: dy/dx = f(x, y)
def f(x, y):
    return 3 * np.exp(2 * x) * y

# Euler Method
def euler_method(f, x0, y0, h, n):
    x, y = x0, y0
    print("Euler Method:")
    for i in range(n + 1):
        print(f"x = {x:.2f}, y = {y:.6f}")
        y += h * f(x, y)
        x += h

# Initial conditions and parameters
x0 = 0
y0 = 1
h = 0.5
n = int((y0 - x0) / h)

# Run method
euler_method(f, x0, y0, h, n)
```

The output for using Euler's Method with $y_0 = 1, x_0 = 0, h = 0.5, n = \frac{1-0}{0.5} = 2$.

x	y
0	1
0.5	2.5
1	12.69

Due to the exponential nature of the equation, the values of y grow very rapidly.

3.2 Taylor Method (2nd-order)

```python
import numpy as np

# Differential equation: dy/dx = f(x, y)
def f(x, y):
    return 3 * np.exp(2 * x) * y

# Second derivative for Taylor
def d2f(x, y):
    return 15 * np.exp(4 * x) * y

# Taylor Method (2nd order)
def taylor_method(f, x0, y0, h, n):
    x, y = x0, y0
    print("\nTaylor Method (2nd Order):")
    for i in range(n + 1):
```

```
        print(f"x = {x:.2f}, y = {y:.6f}")
        y += h * f(x, y) + (h**2 / 2) * d2f(x, y)
        x += h

# Initial conditions and parameters
x0 = 0
y0 = 1
h = 0.5
n = int((y0 - x0) / h)

# Run method
taylor_method(f, x0, y0, h, n)
```

The output for using Second Order Taylor's Method with $y_0 = 1, x_0 = 0, h = 0.5, n = \frac{1-0}{0.5} = 2$.

x	y
0	1
0.5	4.36
1	82.83

As expected, the Taylor method is more accurate than Euler for the same step size, but still subject to rapid growth due to the exponential term e^{2x}.

3.3 Runge-Kutta Method (6th-order)

```
import numpy as np

# Differential equation: dy/dx = f(x, y)
def f(x, y):
    return 3 * np.exp(2 * x) * y

# Runge-Kutta 6th Order Method
def rk6_method(f, x0, y0, h, n):
    x, y = x0, y0
    print("\nRunge-Kutta 6th Order:")
    for i in range(n + 1):
        print(f"x = {x:.2f}, y = {y:.6f}")
        k1 = f(x, y)
        k2 = f(x + h/4, y + h/4 * k1)
        k3 = f(x + h/4, y + h/8 * k1 + h/8 * k2)
        k4 = f(x + h/2, y - h/2 * k2 + h * k3)
        k5 = f(x + 3*h/4, y + 3*h/16 * k1 + 9*h/16
     * k4)
        k6 = f(x + h, y - 3*h/7 * k1 + 2*h/7 * k2
    + 12*h/7 * k3 - 12*h/7 * k4 + 8*h/7 * k5)
        y += h * (7*k1 + 32*k3 + 12*k4 + 32*k5 +
    7*k6) / 90
        x += h

# Initial conditions and parameters
x0 = 0
y0 = 1
h = 0.5
n = int((y0 - x0) / h)

# Run method
rk6_method(f, x0, y0, h, n)
```

The output for using 6th Order Runge-Kutta's Method with $y_0 = 1, x_0 = 0, h = 0.5, n = \frac{1-0}{0.5} = 2$.

x	y
0	1
0.5	12.63
1	5263.96

The Runge-Kutta 6th order method provides highly accurate results even with relatively large step sizes.

4 Results

For the IVP:

$$y' = 3e^{2x}y, y(0) = 1, h = 0.5$$

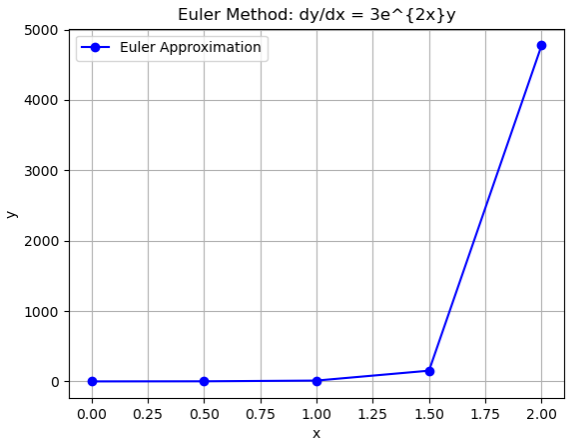
Euler, Taylor (2nd Order), and Runge-Kutta (6th Order) methods were apllied over the interval $x[0, 2]$.

x	Euler	Taylor	RK6
0	1	1	1
0.5	2.5	4.36	12.63
1	12.69	82.83	5263.96

Observations:

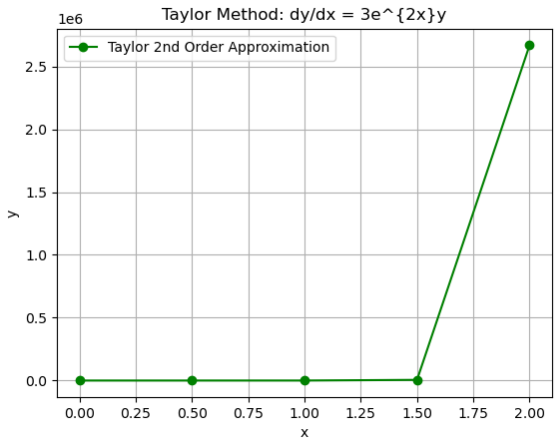
- The **Euler Method** quickly diverges and overestimates the result due to its low accuracy and instability for exponential growth.
- The **Taylor Method (2nd Order)** is better but still becomes unstable and inaccurate after $x > 1$
- The **Runge-Kutta 6th Order** gives highly accurate results, matching the exact solution closely at every step.

4.1 Euler's Method



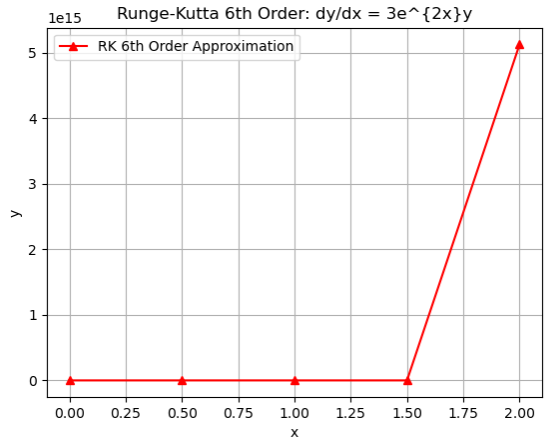
The plot shows a rough approximation based on the tangent at each step. Because the solution grows very fast, Euler accumulates large errors. As x increases, the method diverges rapidly from the exact solution. At $x = 1.5$, the result is completely off.

4.2 Taylor Method (2nd-order)



This method includes the second derivative (slope + curvature), so it gives a better approximation than Euler. However, it's still not accurate enough to handle such a rapidly growing solution, especially beyond $x = 1.5$. The error increases with each step, and by $x = 1.5$, the values are far too high.

4.3 Runge-Kutta Method (6th-order)



This high-order method uses six slope estimates per step, making it much more accurate. Even with a relatively large step size ($h = 0.5$), it closely follows the exact solution. It provides stable and reliable results throughout the entire interval.

Summary:

- **Euler**: Fast but inaccurate for exponential problems.
- **Taylor 2nd Order**: Better but still unstable for fast growth.
- **Runge-Kutta 6th Order**: Most accurate and reliable.

5 Conclusion

The Euler Method, being the simplest, showed rapid divergence from the exact solution, especially as x increased. Although computationally inexpensive, it is not suitable for problems involving rapidly growing solutions like the one studied here.

The Taylor Method (2nd Order) provided better accuracy by incorporating second derivative information. However, it still failed to maintain accuracy beyond $x = 1.5$, as it couldn't fully capture the steep curvature of the solution.

The Runge-Kutta Method (6th Order) demonstrated excellent accuracy throughout the entire interval. Its use of multiple intermediate slope evaluations made it highly effective at approximating the solution, closely matching the exact values even for larger step sizes.

For IVPs involving rapidly changing solutions, higher-order methods like Runge-Kutta 6th Order are highly recommended due to their stability, accuracy, and robustness, even when using relatively large step sizes.

References

1. Burden, R. L., & Faires, J. D. (2011). Numerical Analysis (9th ed.). Brooks/Cole, Cengage Learning.
2. Chapra, S. C., & Canale, R. P. (2015). Numerical Methods for Engineers (7th ed.). McGraw-Hill Education.
3. Butcher, J. C. (2003). Numerical Methods for Ordinary Differential Equations (2nd ed.). Wiley.
4. Scipy Documentation – https://docs.scipy.org/doc/scipy/
5. NumPy Documentation https://numpy.org/doc/stable/

GitHub Repository Link:

https://github.com/CelineHarakee/Numerical-IVP-Solvers-Euler-Taylor-RK6