MAE 4021 FINAL REPORT

Comparing the Qualities of Numerical Methods for Solving First Order Ordinary Differential Equations

5/12/2025

Abstract

The overall objective of this report was to test six numerical methods of solving 1st order ordinary differential equations for two differential equations using three parameters. The first parameter tested was the accuracy of the explicit Euler's method for different numbers of timesteps, with it being shown that as the number of timesteps increases, so does the accuracy. The second parameter was the accuracy of each of the methods for the same number of timesteps, with the 2nd and 4th order Runge-Kutta being the most accurate. The final Parameter was comparing explicit and implicit methods, with explicit methods performing better over the small x intervals for the two functions tested.

Author: Matthew Schnese

Nomenclature:

Symbol:	Quantity:
$\frac{dy}{dt}$	Derivative of y with respect to t
f(t,y)	A function of t and y
y_i	The value of y for t=i
h	The length between t values
K	Runge-Kutta constant
N	Number of timesteps

Introduction:

In engineering, many systems are based on the changing behaviors of physical properties. For dynamics, those properties are position, velocity, and acceleration. For thermodynamics and heat transfer, those properties are heat and work. To analyze these topics, the mathematics describing the system needs to be able to model these changes as functions. The analysis of these systems requires calculus, which is based on rates of change. More specifically, these systems all use differential equations, which are equations that contain the derivative of an unknown function. Therefore, differential equations are solved for the original function. In this report, we will focus on ordinary first order differential equations, referred to as 1st order ODEs, which contain first derivatives of only one variable (y, t, etc.) [1]. The general formula for an ordinary first order differential equation can be seen below in Equation 1.

$$\frac{dy}{dt} = y'(t) = f(t, y) \quad [1]$$

The equation above shows that for a function y(t), the derivative of that function is itself a function of both y and t. Most of the engineering curriculum focuses on differential equations that are analytically solvable, which means that the original function can be found. While this is useful, in most scenarios the governing differential equation(s) of the system are often not analytically solvable. This could be due to the difficulty of the equation, with a modern example being the Navier-Stokes equation, or it could be impractical to solve, with an example being a triple pendulum.

For these differential equations, instead of an analytic solution a numerical solution is found through various methods. The numerical solution is different from the analytic solution since it is not a function but rather a specific value. This means that for an analytic solution different values for variables can be easily tested but the numerical solution would have to be recalculated for every set of new variables. For a numerical solution, the values for y(t) are calculated using Equation 2, which can be seen below.

$$y_{i+1} = y_i + \frac{dy}{dt}h \quad [2]$$

In Equation 2, y_{i+1} is the new value of y that is calculated, y_i is the current value of y, and $\frac{dy}{dt}$ is the slope of y. Each numerical method has a different way of solving for the slope, which impacts the accuracy of each method for different functions. The objective of this report is to compare the accuracies of 6 different numerical methods: explicit and implicit Euler's methods, 2nd and 4th order Runge-Kutta methods, the 4th order Adams-Bashforth method, and the 4th order Adams-Moulton method.

Method & Analysis:

To perform the analysis, each of the six methods, explicit and implicit Eulers, 2nd and 4th order Runge-Kutta, 4th order Adams-Bashforth, 4th order Adams-Moulton, were written into MATLAB script files. After the code was completed, two functions were tested. The first function was tested with the initial conditions t(0)=0, y(0)=0.2, and t_f=0.2. The second function was tested with the same initial conditions other than t_f=1.0. The first parameter was the accuracy of one of the methods based on the number of points over the interval. The next parameter was the accuracy based on the solution method for a fixed number of points. The third and final parameter was the accuracy of implicit (implicit Euler's, 4th order Adams-Moulton) and explicit (explicit Euler's, 2nd and 4th order Runge-Kutta, and 4th order Adams-Bashforth). The number of timesteps N was also converted to the length of each timestep h. The two differential equations that were tested and the conversion to h can be seen below in Equation 3-5.

$$y'(t) = -400 \cdot y(t) \qquad [3]$$

$$y'(t) = \cos(\pi t) + y(t) \qquad [4]$$

$$h = \frac{(b-a)}{N} \qquad [5]$$

The first method tested was the explicit Euler's method. This method is an explicit method, which means that it is only based on current values for y and t. This method assumes that at a close enough distance h, the function is linear and therefore has a constant slope. Using this assumption, the next y value is solved using the equation of a line. The next t value is solved by adding h to the current t, which is repeated for every other numerical method in this report. The equations for the next t and y value using this method can be seen below in Equations 6 and 7.

$$t_{i+1} = t_i + h \qquad [6]$$

$$y_{i+1} = y + f(t_i, y_i)h \qquad [7]$$

The second method tested was the implicit Euler's method. Because this is an implicit method, the numerical solution is found using the next y value. Therefore, a root finding method is used to solve for the next y value. For this report, Newton's method was used for both implicit numerical methods. Newton's method was applied iteratively, so it was solved for every time step. The equations for Newton's method and the implicit Euler's method can be seen below in Equations 8 and 9.

$$y_{i+1} = y_i - \frac{f(y_i)}{f'(y_i)} \quad [8]$$

$$y_{i+1} = y + f(t_{i+1}, y_{i+1})h \quad [9]$$

The third method tested was the 2nd order Runge-Kutta method. This method is explicit and relies on two constants K_1 and K_2 to find the constant slope. While there are many different 2nd order Runge-Kutta methods, this report focuses on the modified Euler version. For this version, the equations for the constants and the next y value can be seen below in Equations 10-12.

$$K_1 = f(t_i, y_i) \quad [10]$$

$$K_2 = f(t_i + h, y_i + K_1 h) \quad [11]$$

$$y_{i+1} = y_i + \frac{h}{2}(K_1 + K_2) \quad [12]$$

The fourth method tested was the 4th order Runge-Kutta method. This method is very similar to the 2nd order Runge-Kutta, but instead of two K values there is now 4. Like the 2nd order RK, there are many different variations, but for this report the classical method was used. The equations for each constant K and the 4th order Runge-Kutta method can be seen below in Equations 13-17.

$$K_1 = f(t_i, y_i) \quad [13]$$

$$K_2 = f(t_i + \frac{h}{2}, y_i + \frac{K_1 h}{2}) \quad [14]$$

$$K_3 = f(t_i + \frac{h}{2}, y_i + \frac{K_2 h}{2}) \quad [15]$$

$$K_4 = f(t_i + h, y_i + K_3 h) \quad [16]$$

$$y_{i+1} = y_i + \left(\frac{1}{6}K_1 + \frac{2}{6}K_2 + \frac{2}{6}K_3 + \frac{1}{6}K_4\right)h \quad [17]$$

The fifth method tested was the 4th order Adams-Bashforth method. This is an explicit multi-step method, which means it is based off both current and previous values of y and t. Since this method is 4th order, it depends on the past three values for both y and t. This method is a weighted average of each of the last three values and the current value for the function. The equation for the method can be seen below in Equation 18.

$$y_{i+1} = \frac{h}{24}(55f(t_i, y_i) - 59f(t_{i-1}, y_{i-1}) + 37f(t_{i-2}, y_{i-2}) - 9f(t_{i-3}, y_{i-3})) \quad [18]$$

The last method tested was the 4th order Adams-Moulton method. This method, like the second method, is an implicit method and therefore uses newton's method each iteration. Since this method is 4th order, it relies on the previous two values, the current value, and the next value for the function. The equation for the method can be seen below in Equation 19.

$$y_{i+1} = \frac{h}{24}(9f(t_{i+1}, y_{i+1}) + 19f(t_i, y_i) - 5f(t_{i-1}, y_{i-1}) + 9f(t_{i-2}, y_{i-2})) \quad [19]$$

Results & Discussion:

The first parameter tested was the accuracy of one method based on step size, with the method used being the explicit Euler's method. This method was chosen because it is a simple first order method which makes it especially vulnerable to the step size used. The number of timesteps N tested were 10, 20, 40, 80, 160, 320, 640. It is worth noting that since N was the set parameter and not height, the size of the steps was different for the two functions. The next parameter tested was the accuracy of each of the six methods when using the largest number of time steps. The final parameter tested was the accuracy of implicit and explicit functions for each function.

When testing the explicit Euler's method for each step size, the reactions were different for the two functions. For the first function, the numerical method was not stable until N=80. For timesteps lower than N=80, the values for y would keep increasing in an exponential oscillation. For N=80 and greater, the function showed the general shape of the analytic solution, which exponentially decayed to zero, while reducing error while the number of timesteps increased. For the second function, the numerical method was stable for every size, with the error reduced for every increase in the number of timesteps. While the error was decreasing, the amount of reduction was lower for every increase in the number of timesteps. Additionally, none of the timesteps were particularly close to the solution. This may be due to the relatively small t interval being analyzed. The graphs of each number of timesteps for both functions one and two can be seen below in Figure 1.

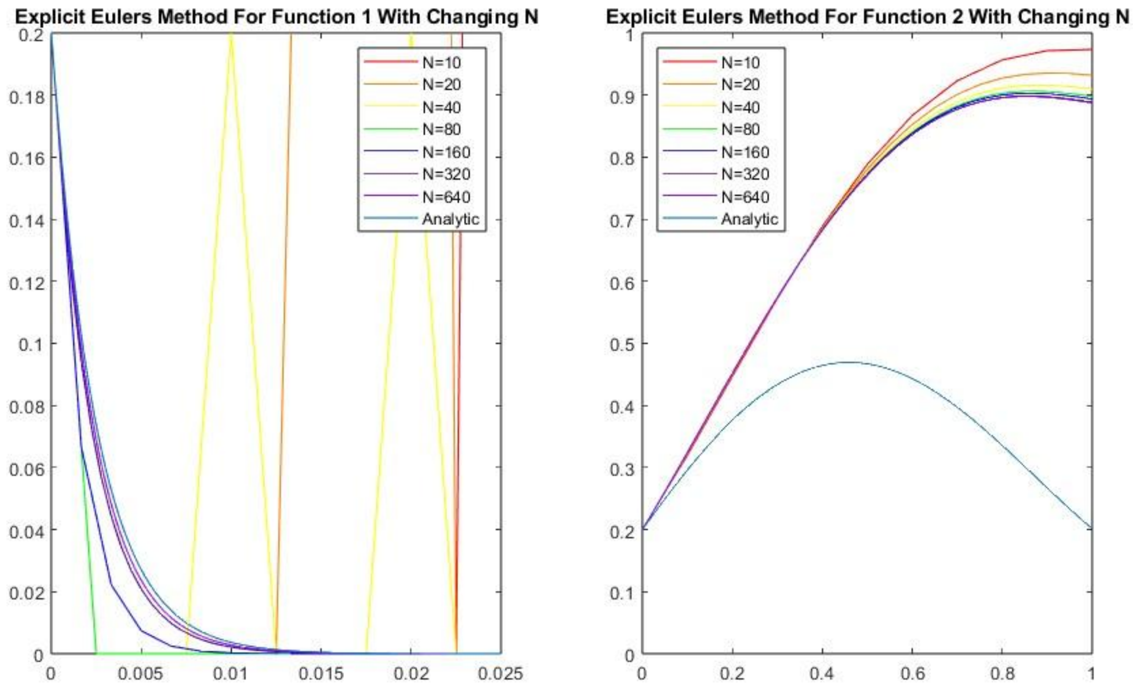

Figure 1: Euler's Explicit Method for Different Timesteps for Functions 1 and 2

The behavior of the total error for each number of timesteps behaved in the same way that the numerical solutions did for each function. For function one, the error of the unstable graphs, the number of timesteps being less than N=80, continued increasing over the whole interval. For N=80 and higher however, the error of the numerical solution converged to zero for each number of timesteps. As the number of timesteps increased, the error converged more quickly. For the second function, the error behaved similarly for each number of timesteps. While each error function behaved similarly, the overall error did decrease as the number of timesteps increased. This can be seen below in Figure 2.

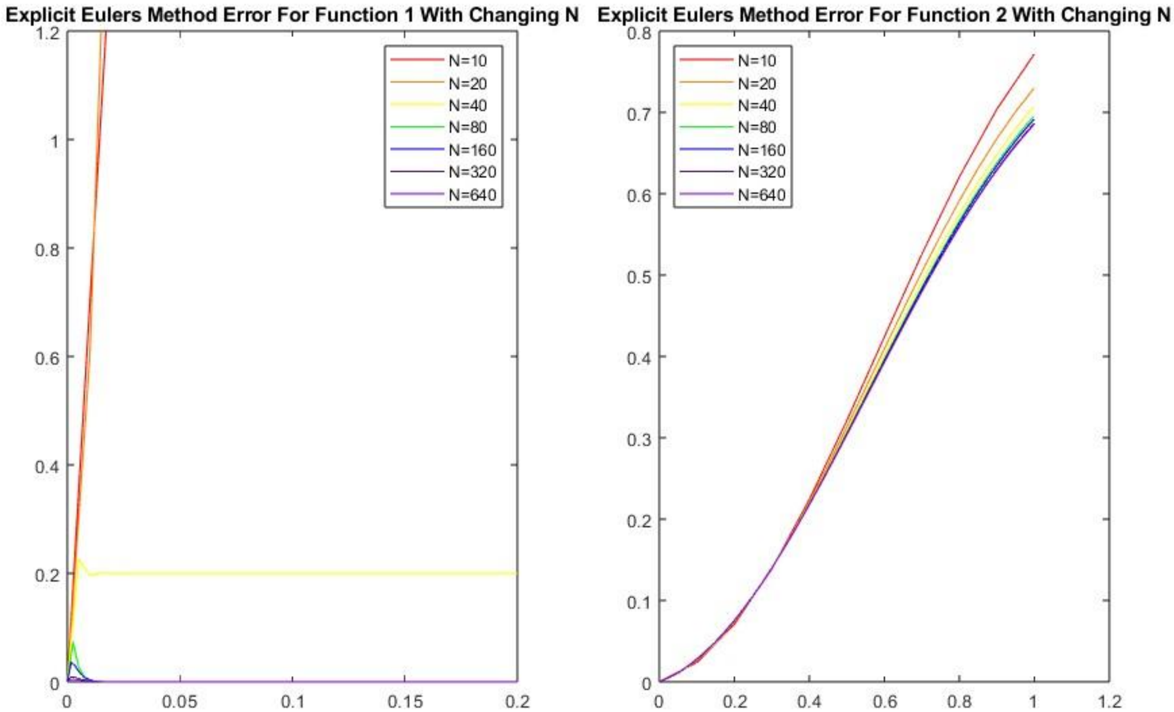

Figure 2: Error of Euler's Explicit Method for Different Timesteps Plotted for both Functions

When comparing each numerical method with the largest number of timesteps, every method behaved similarly, sharing the same general shape. For function one, the implicit Euler's, explicit Euler's, and the 4th order Adams-Moulton method were furthest from the analytical solution. While they correctly converged, they were on either side of the line containing every other numerical method. For the implicit methods, this could be caused by the stiffness of the differential equation, since it decays to zero so rapidly. Implicit numerical methods are poor at handling stiff differential equations such as this. For both Euler's methods, this error could also be caused by the accuracy of the method, since they are only 1st order. For the second function, each method was so identical in nature that they could not be discerned on the graph. This suggests that for less stiff functions, like function two, the accuracy of the 6 methods tested is more dependent on the number of timesteps rather than the method. The graphs of each of the six numerical methods tested at N=640 for both functions can be seen below in Figure 3.

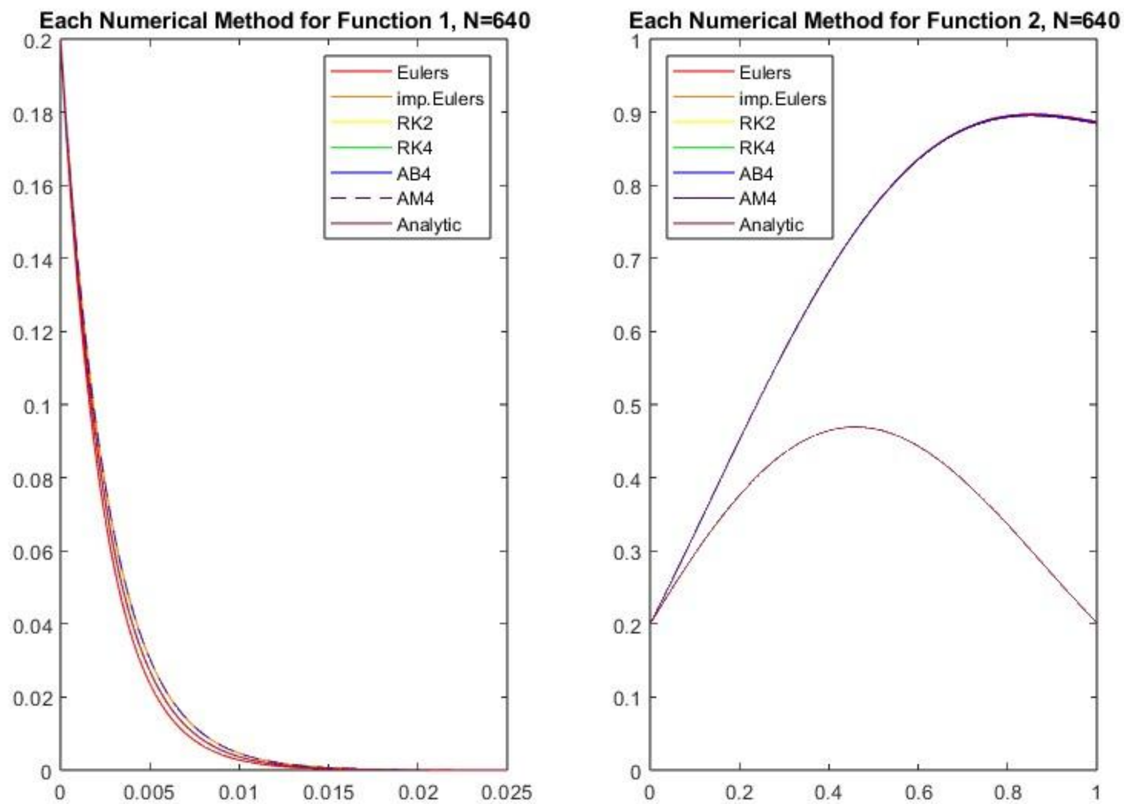

Figure 3: Each Numerical Method Plotted for Functions 1 and 2, for N=640

When analyzing the errors of function one, it is important to point out the scaling. Each of the six numerical methods had very low true errors when evaluated at N=640. With that observation in mind, the explicit Euler's method had by far the most error for this number of timesteps. Both the implicit Euler's method and the 4th order Adams-Moulton method had the exact same error curve, which is an interesting discovery. Every other numerical method was very accurate, with both Runge-Kutta methods standing out. Specifically, the 4th order Runge-Kutta method had so little error that it was not visible on the plot when the axes were normalized to the other errors. For the second function, each of the error graphs shared the same shape, but did not converge to zero. Instead, each method's error increased over the entire interval. These error plots reinforce the observations and inferences that were made previously for these functions. The error plots for each numerical method at N=640 can be seen below in Figure 4.

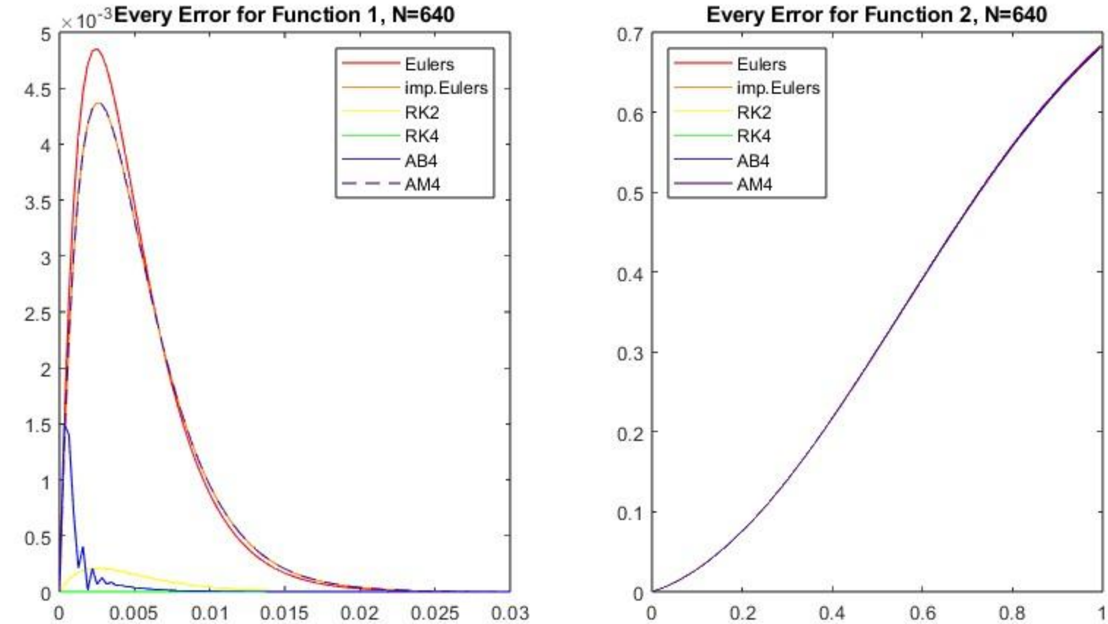

Figure 4: Error for each Numerical Method for both Functions for N=640

The third parameter tested was the accuracy of implicit and explicit functions. For the first function, the implicit methods performed better than the explicit methods at higher orders and worse than the explicit methods at higher orders. This can be seen in the errors for the explicit and implicit methods, which are 1st order. For higher orders, the 4th order Runge-Kutta and the 4th order Adams-Moulton can be compared. While the 4th order Adams-Moulton is multi step, it was analyzed using the 4th order Runge-Kutta for the first 3 points. Even though both methods are 4th order, the 4th order Runge-Kutta has orders of magnitude of less error. Furthermore, the 2nd order Runge-Kutta method, which is explicit, has less error than the 4th order Adams-Moulton method. For the second function, the error was indistinguishable for the implicit and explicit methods, so neither species of method had less error than the other.

Conclusion:

The objective of this report was to test six different numerical methods for solving first order ordinary differential equations through three parameters, accuracy through step size, accuracy through method, and accuracy through method type. The explicit Euler's method was chosen for the first parameter since it is first order and therefore more susceptible to error. The result of this analysis was that for a certain number of N, the explicit Euler's method was not stable, let alone accurate. For higher numbers of timesteps, the method became far more accurate, but still accumulated errors. For the second parameter, the 4^{th} and 2^{nd} order Runge-Kutta were the most accurate numerical methods, with the explicit Euler's being the least accurate. For the third parameter, for the two equations analyzed explicit methods generally performed as well if not better than the implicit methods over small timescales. Overall, this analysis showed that the 4^{th} order Runge-Kutta was the most accurate numerical method for the two equations given over short intervals of t.

In future analysis, it is recommended for the scope of the project to become bigger. N=640 is still relatively small, the range of t values was very small, and the number of functions analyzed was also small. For the next analysis, more functions and larger t intervals could better demonstrate the parameters analyzed.

References:

[1] Gilat, A., Numerical Methods for Engineers and Scientists, Third ed., Wiley Global Education, 2014.