FCM 2 HW1

Ryan Bausback

January 2021

1 Executive Summary

Our purpose is to implement four different explicit Runge Kutta (RK) methods (with between 1 and 4 stages) and then evaluate them using three different ordinary differential equation (ODE) initial value problems (IVP). The methods will be evaluated in terms of error, stability, and convergence. We found that the 4-stage explicit Runge Kutta (RK4) method was generally the most accurate and converged in the fewest steps. However, this method was also the most computationally expensive as it requires four times as many function evaluations per step as the simplest, 1-stage method.

2 Statement of Problem

During the following experiments, four different explicit Runge Kutta methods will be implemented in Java, and used in solving variations of three distinct ODE IVP. Those methods are Forward Euler (a one-stage method), a two-stage RK (RK2) parameterized by μ (where μ=1 is the explicit trapezoidal rule and μ=1/2 is the explicit midpoint rule), a three-stage RK (RK3) also parameterized by μ, and the classical explicit four stage Runge Kutta (RK4).

The three ODE IVP to be numerically solved are the classical exponential growth/decay problem ($y' = \lambda y$, $y(0) = C$), a linear, time-invariant autonomous oscillator, and a linear, time-invariant, non-autonomous driven system. The solutions that are numerically derived will then be compared against the exact solutions to find the method's error, the convergence as $h \to 0$ will be analyzed, and evidence of the regions of absolute stability will be found. The equations themselves will be also parameterized in a number of ways in order to acquire a wide range of results.

Without running any tests, we suspect that the RK4 method will prove to be the best, despite being the most computationally complex. This would fall in line with results described in the notes, where it has the tightest error bound and the largest region of absolute stability.

3 Description of Mathematics

Explicit Runge Kutta (RK) methods differ from Linear MultiStep (LMS) methods in that there is only a single time interval on which the ODE is solved, in this case referred to as $[t_{n-1}, t_n]$, compared with many time-steps for LMS [1]. This single interval is then broken up into subintervals, allowing the methods to achieve higher order over a narrow range. For the purposes of these experiments, since all of the IVP are given such that initial time point is 0, $t_{n-1} = 0$ for all RK methods, while t_n will vary.

The RK methods with s=1,2,3 and 4 stages will be evaluated. Forward Euler is the first method to be evaluated, given as:

$$y_n = y_{n-1} + hf(t_{n-1}, y_{n-1})$$

It is first-order (p=1), meaning that it only is able to find the exact solution of a constant function. It also only has one stage (s=1) per each subinterval, where the subinterval is given as $h = \frac{t_n - t_{n-1}}{N}$ where N is the number of subintervals.

The two stage explicit RK method is given as:

$$\hat{y_n} = y_{n-1} \to \hat{f}_1 = f_{n-1}$$
$$\hat{y_2} = y_{n-1} + h(\mu\hat{f}_1) \to \hat{f}_2 = f(t_{n-1} + \mu h, \hat{y_2})$$
$$y_n = y_{n-1} + h\left((1 - \frac{1}{2\mu})\hat{f}_1 + \frac{1}{2\mu}\hat{f}_2\right)$$

where $\mu = 1$ gives the explicit trapezoidal rule and $\mu = 1/2$ gives the explicit midpoint rule.

The three stage explicit RK method is given as:

$$\hat{y_n} = y_{n-1} \to \hat{f}_1 = f_{n-1}$$
$$\hat{y_2} = y_{n-1} + h(2/3\hat{f}_1) \to \hat{f}_2 = f(t_{n-1} + 2/3h, \hat{y_2})$$
$$\hat{y_3} = y_{n-1} + h((\frac{2}{3} - \frac{1}{4\mu})\hat{f}_1 + \frac{1}{4\mu}\hat{f}_2) \to \hat{f}_3 = f(t_{n-1} + 2/3h, \hat{y_3})$$
$$y_n = y_{n-1} + h\left(\frac{1}{4}\hat{f}_1 + (\frac{3}{4} - \mu)\hat{f}_2 + \mu\hat{f}_3\right)$$

where μ will be parameterized the same as in RK2.

The classical four stage RK method is given as:

$$\hat{y_n} = y_{n-1} \to \hat{f}_1 = f_{n-1}$$
$$\hat{y_2} = y_{n-1} + \frac{h}{2}\hat{f}_1 \to \hat{f}_2 = f(t_{n-1/2}, \hat{y_2})$$
$$\hat{y_3} = y_{n-1} + \frac{h}{2}\hat{f}_2 \to \hat{f}_3 = f(t_{n-1/2}, \hat{y_3})$$
$$\hat{y_4} = y_{n-1} + h\hat{f}_3 \to \hat{f}_4 = f(t_n, \hat{y_4})$$

$$y_n = y_{n-1} + h\left(\frac{1}{6}\hat{f}_1 + \frac{1}{3}\hat{f}_2 + \frac{1}{3}\hat{f}_3 + \frac{1}{6}\hat{f}_4\right)$$

All of these methods are of order that directly corresponds to the number of stages that each methods uses to solve the ODE over the fixed interval. This order can be determined by comparing the Talyor series for the difference operator with the Taylor series of the right hand side made up of a linear combination of the f_i's [2]. For instance, if we rearrange Forward Euler such that $\frac{y_n - y_{n-1}}{h} = f(t_{n-1}, y_{n-1})$ and then do the expansion of $y(t_n)$ around t_{n-1}, for the left hand side we get:

$$\frac{y(t_{n-1}) + hy'(t_{n-1}) + O(h^2) - y(t_{n-1})}{h} = y'(t_{n-1}) + O(h)$$

Since the function f is just another way to denote the derivative of y, comparing it the right hand side, we can see that the difference is in the $O(h)$ term, meaning the method is order 1. This is known as the local truncation error (d_n) [2]. For higher order methods, identities relating the derivatives of y to the linear combinations of the partials of f are used to reduce the right hand side for comparison.

Since the local truncation error is a function parameterized by h, if we plot $\log(d_n)$ against $\log(h)$ such as when analyzing convergence, we should see increasing slope for increasing order of the method. For instance, under log, $O(h^2)$ will become $2\log(O(h))$ and so forth.

The absolute stability of the method is concerned with the values of $h\lambda$ such that when tn becomes large, the function $R(z)$ causes $y_n = R(z)y_{n-1}$ to diverge away from the true function. While the notion of absolute stability is specific to the test problem $y' = \lambda y$, the general stability of the other two should be similar, since they are variations of the test problem. For the explicit RK methods we are dealing with, $R(z) = 1 + h\lambda + ... + \frac{(h\lambda)^p}{p}$, giving us Figure 1 in the $h\lambda$ complex plane. Therefore, we should see behavior consistent with this graph given a large value of t_n.

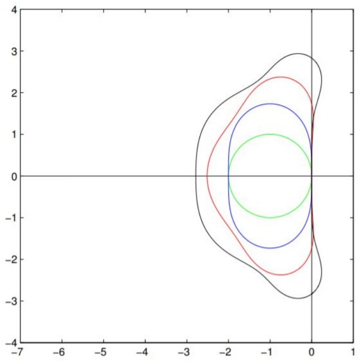


Figure 1: Absolute Stability Regions for RK 1,2,3,4 in Complex $h\lambda$ Plane [1]

4 Algorithm and Implementation

The algorithm is set up such that each of the explicit Runge Kutta methods has it's own function, while main() is treated as a sandbox utilized mostly to output results to files for later analysis. Each of the functions is then divided up based on which IVP is to be tested, since each one uses a different function for f and in Java, function names cannot be passed as variables to other methods. Instead, a tracking variable in main is used to denote which IVP is to be tested on, and then the if-statements dividing the methods call the appropriate methods within the routines.

Since each step is divided up, a for loop iterates across the step and each stage of the methods, as described in Section 3, is called for each loop iteration. This involves s function calls for each loop, where s is the number of stages, and then times the total times the loop iterates. Thus, for N subdivisions of $[t_{n-1}, t_n]$, Forward Euler has a complexity of N, RK2 of $2N$, RK3 of $3N$, and RK4 of $4N$. Therefore, as expected, the 4-stage Runge Kutta is the most complex although the order of complexity is the same for all four methods.

However, despite this, it does not actually require the most storage. Since each f_i is only only used to calculate the subsequent y_{i+1} and y_n, each only need a single storage variable to store the f_i, after which, f_{i+1} can overwrite it. This is slightly more storage efficient than RK3 for instance, where subsequent y_i's require previous f_i's in order to be computed, resulting in all of the f_i's being stored in separate variables until the end. For no method is storage for the y_i's required, since the formulas for them can be directly passed into the functions for the corresponding f_i's.

The limiting factor for this implementation was the system of ODE's from IVP2, which requires 2 components to be stored for each step within the interval. However, since only one thing can be passed back from the function, y_n was adjusted to be a 2-D array with 2 columns. For the first and third initial value problems, only the first column was used.

5 Experimental Design and Results

5.1 Initial Value Problem 1: $y' = \lambda y, y(0) = C$

The first initial value problem (IVP) to be numerically solved was the classical growth/decay problem: $y' = \lambda y, y(0) = C$. As we know from undergraduate calculus, this problem has the solution $y = Ce^{\lambda t}$. Therefore, since we know the true solution for any value of t that we may choose, we can use the true absolute error for method comparison, instead of having to do error estimation.

For the first iteration of the RK methods, the correctness of the implementation was tested using the following parameters: $C = 1$, $\lambda = 1$, on the interval $[0, 1]$, with 5 sub-divisions ($N = 5$). As discussed above, the initial point of $t_{n-1} = 0$ must be selected since that is the value given to us in the initial condition, though in later iterations, t_n may be changed. The values of C and λ

were then selected because of simplicity. This resulted in the outputs from each RK method in Figure 2.

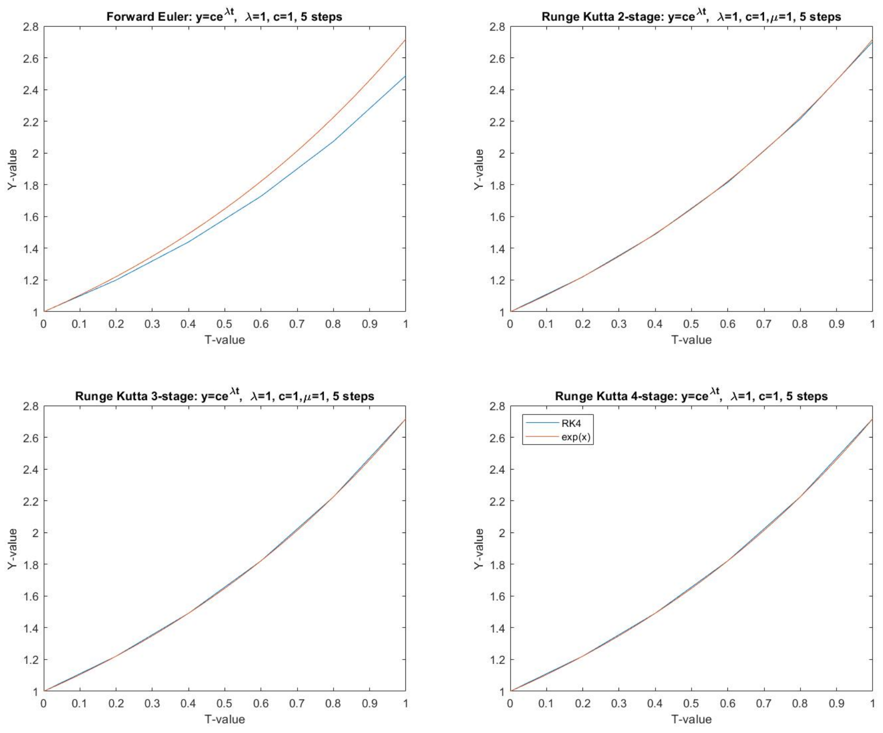
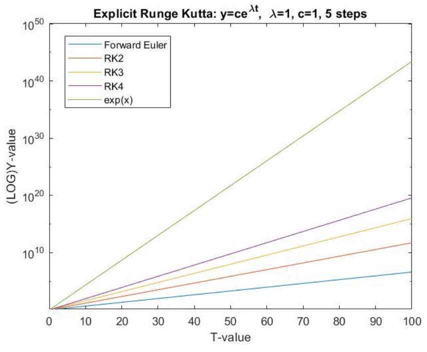


Figure 2: e^x (red) versus numerical solutions from RK 1-4

As the figure clearly shows, the explicit Runge Kutta methods 2-4 are able to produce the correct solution e^x with a high level of accuracy, while Forward Euler is able to do so with some deviation near $t_n = 1$. This deviation is to be excepted, however, since the absolute stability regions are in the right half of the $h\lambda$ plane and $h\lambda = 1 * (1-0)/5 = 0.2$ in this case. When this test was run again for $\mu = 0.5$, no difference in the error from the solution was found for RK2 or RK3 (i.e. explicit midpoint and explicit trapezoidal produce identical results in this case).

Despite being outside of the region of absolute stability, the methods are able to produce accurate solutions. However, if we then increase t_n to a much larger number, we would expect greater divergence from the true solution. This is exactly what happened when t_n was increased to 100, as shown in Figure 3. By examining the log of the y-values, it is easy to visual just how far off all of the RK methods are when outside the region of absolute stability. However, if we instead set $\lambda = -1$, giving us an $h\lambda = -1 * (100-0)/100 = -1$, we see a much different behavior. All of the methods are now extremely close to the true solution.



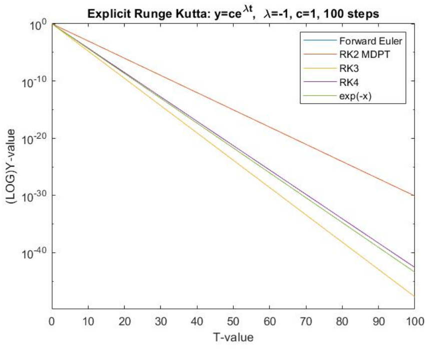
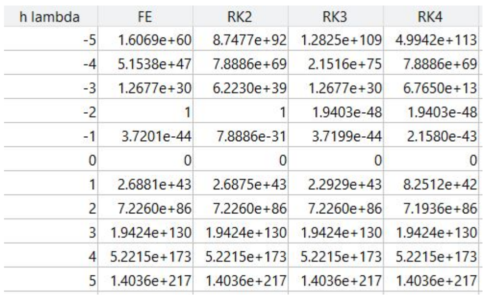


Figure 3

All of the methods are now extremely close to the true solution. We can take this analysis further by iterating lambda over the part of the real axis of most interest shown in the absolute stability plot section 3, and see if that gives us the same results. This is exactly what we ended up with, as in Figure 4.



h lambda	FE	RK2	RK3	RK4
-5	1.6069e+60	8.7477e+92	1.2825e+109	4.9942e+113
-4	5.1538e+47	7.8886e+69	2.1516e+75	7.8886e+69
-3	1.2677e+30	6.2230e+39	1.2677e+30	6.7650e+13
-2	1	1	1.9403e-48	1.9403e-48
-1	3.7201e-44	7.8886e-31	3.7199e-44	2.1580e-43
0	0	0	0	0
1	2.6881e+43	2.6875e+43	2.2929e+43	8.2512e+42
2	7.2260e+86	7.2260e+86	7.2260e+86	7.1936e+86
3	1.9424e+130	1.9424e+130	1.9424e+130	1.9424e+130
4	5.2215e+173	5.2215e+173	5.2215e+173	5.2215e+173
5	1.4036e+217	1.4036e+217	1.4036e+217	1.4036e+217

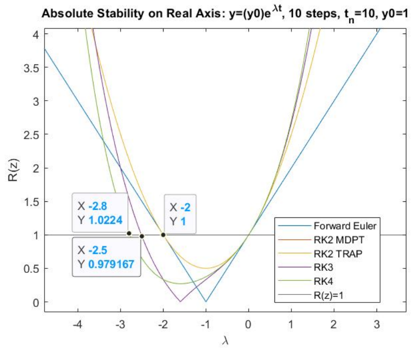


Figure 4: Absolute Global Error for Different Values of $h\lambda$ (left) and Cross-Section of Absolute Stability Region along Real Axis (right)

In order to achieve this, h was set such that $h = 1$ from $t_n = 100$ and $N = 100$ in to order for any differences in stability to be obvious. As can be seen, within the absolute stability regions, the absolute global error drops to almost 0, while outside of the regions, it explodes. For instance, for Forward Euler, the absolute stability region is a circle of radius 1 centered at -1 on the real axis, resulting in 10^{-44} error inside while 10^{30} or higher outside.

The rate of convergence of the error of the RK methods was also investigated for IVP 1 for the original parameter choices as $h \to 0$. As in Figure 5, higher order methods produced results with less error from the true solution.

Additionally, we can see that the higher order methods also converge faster than lower order methods such as Forward Euler, shown through the steeper slope of the those methods. This is a direct indication of the higher order of the method, since it is able to retain more of the terms from the true Taylor expansion, allowing it to get to the real solution faster. We also note that true convergence is probably impossible due to the limitations in the computer, since a solution found using "double" cannot get past machine epsilon or 10^{-16}. This is shown through the random fluctuations of RK4 for a step-size smaller than 10^{-3}, and for RK3 past 10^{-4}.

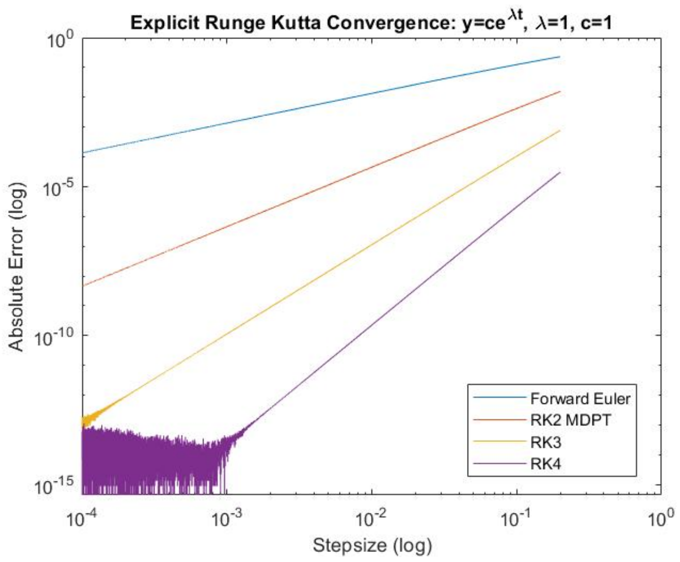


Figure 5: Absolute Global Error Convergence for RK 1-4

Changing the initial condition did not appear to have any impact on error or stability (Figure 6). It only seemed to stretch the solution by the amount of the multiplication, C.

5.2 Initial Value Problem 2: Oscillator System of ODEs

The second initial value problem to be solve was given as the following system of equations and known solution:

$$f = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 0 \\ \gamma \end{pmatrix} \rightarrow \begin{pmatrix} y_1(t) = \gamma \sin(\omega t) \\ y_2(t) = \gamma \cos(\omega t) \end{pmatrix}$$

This IVP was significantly more challenging than the first because y_1 and y_2 are interrelated during the entire solving process, requiring the matrix-vector product to be computed at every single step where the function f was used in any of the RK methods. For instance, this resulted in the following formula for y_n with RK2:

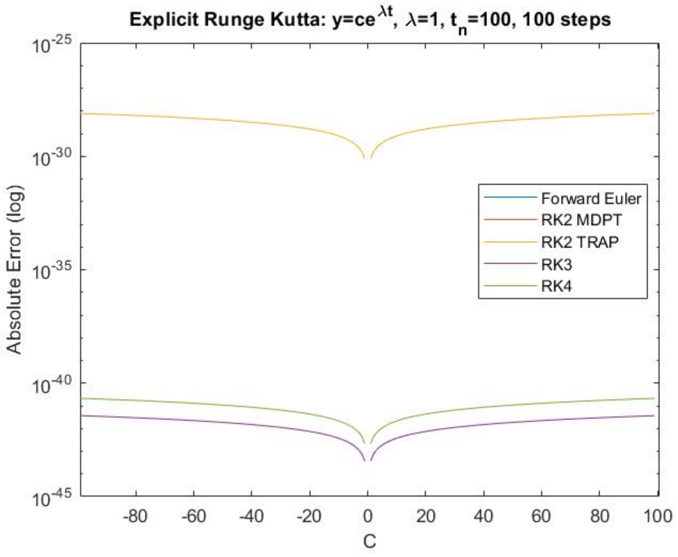


Figure 6: Absolute Global Error VS y(0)=C for RK 1-4

$$y_n = \begin{pmatrix} y_{1,n-1} \\ y_{2,n-1} \end{pmatrix} + h(1 - \frac{1}{2\mu}) \begin{pmatrix} \omega y_{2,n-1} \\ -\omega y_{1,n-1} \end{pmatrix} + \frac{h}{2\mu} \begin{pmatrix} \omega(y_{2,n-1} - h\mu\omega y_{1,n-1}) \\ -\omega(y_{1,n-1} + h\mu\omega y_{2,n-1}) \end{pmatrix}$$

These were then seperated into the top and bottom components of the matrix and stored in different columns of the 2D y_n array for each step up to N. The same process of computing the matrix vector product for each step was also applied to RK3 and RK4.

The test problem in this case for correctness of implementation involved $\omega = 1$, $\gamma = 1$, but instead of using $[0, 1]$, t_n was set to 10 with $N = 100$ so that any deviations would be highly visible. However, the high number of steps allowing the overall h value to still be close to 0, meaning that if the stability properties were similar to those of IVP1, the higher order methods would still give a very accurate solution. This was found to be the case, as Figure 7 shows.

As t_n increases, Forward Euler gets further and further away from the true solution at the peaks. But once you get even one order higher, the numerical solutions provided by the methods become indistinguishable from the true solution. Therefore, we can conclude that this implementation is correctly reproducing the algorithms for this test problem. At this level, no different was found between different μ values for the RK2 and RK3 families.

In terms of the convergence of the error, by performing the same trick using the log/log plot on interval [0,1], we can see that lines with differing slopes appear, again indicating the differing orders of the methods (Figure 8). However, there are some notable differences for this IVP compared to the last one. None of the errors actually appear to converge all the way to machine epsilon and

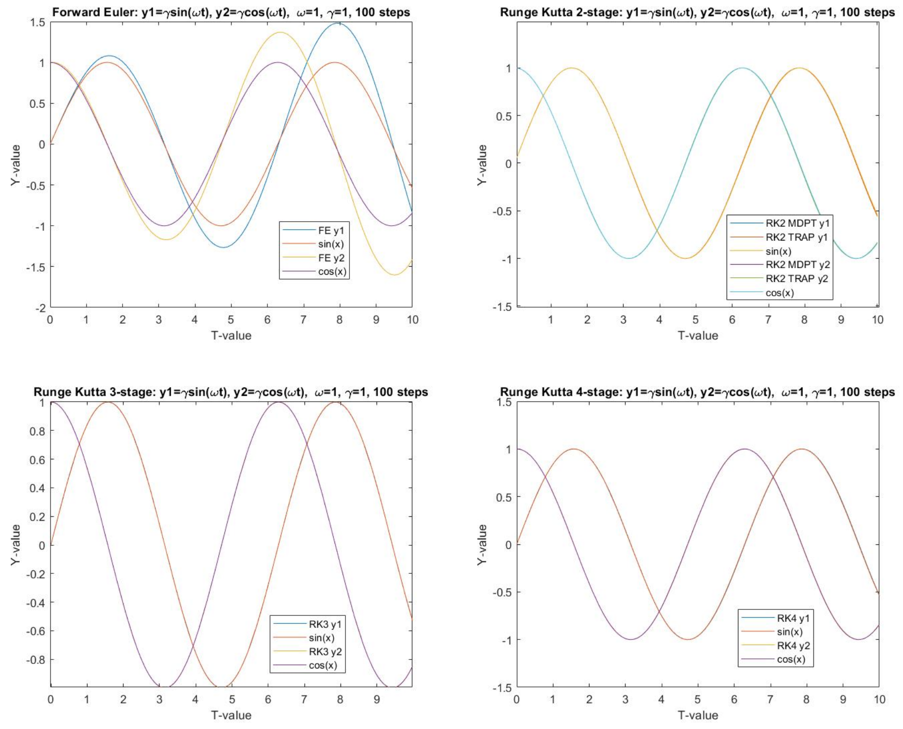


Figure 7: $\sin(t)$ and $\cos(t)$ versus numerical solutions from RK 1-4

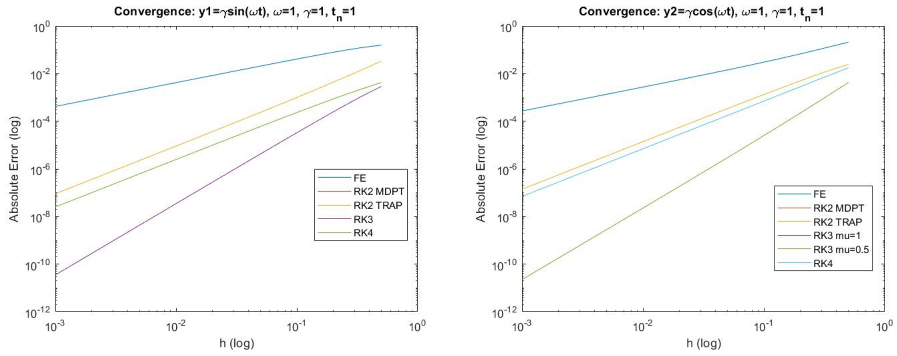


Figure 8: $\sin(t)$ and $\cos(t)$ absolute error convergence as $h \to 0$ for RK 1-4

RK4 has a slope that is less steep than RK3. Part of the reasoning for this could stem from the oscillatory, interconnected nature of the problem, since true convergence requires both the numerical solutions for y1 and y2 to converge simultaneously. No difference was found between different μ values for the RK2 and RK3 families.

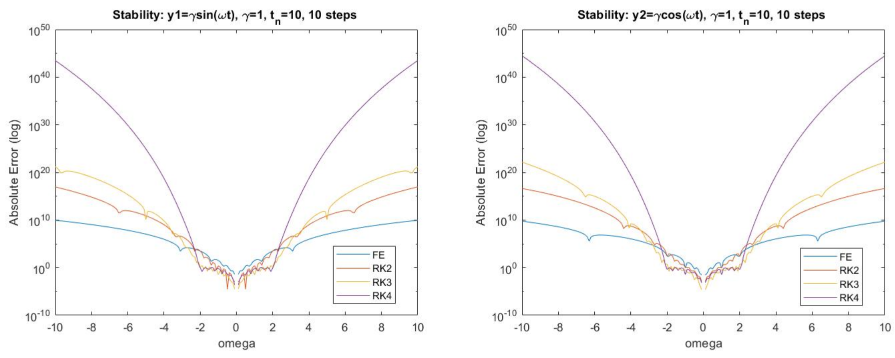


Figure 9: $\sin(\omega t)$ and $\cos(\omega t)$ stability on real $h\omega$ axis for RK 1-4

To determine the stability of the second initial value problem, we set h=1 $(t_n = 10, N = 10)$ and varied ω along the real axis, giving the results shown in Figure 9. It therefore appears that the method on this problem is stable in the same region as for the first test problem, but includes an identical mirror region flipped across the imaginary axis as well. This is especially visible with the RK4 method, which has a drastic increase in error past ≈ 2.5, which is the same border of the absolute stability region shown in the plot in Section 3. This makes sense conceptually as well, since IVP 1 is the same as the equation for $f_1 = \omega y$, and the opposite, $f_2 = -\omega y$, is also included.

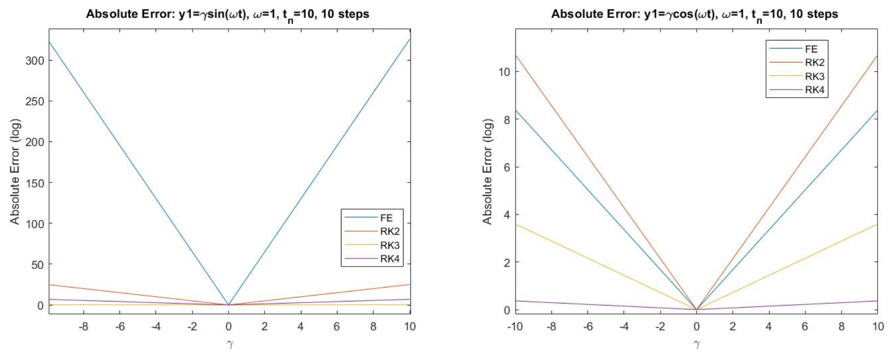


Figure 10: $\gamma \sin(t)$ and $\gamma \cos(t)$ absolute error with γ variation for RK 1-4

This time, changing the initial condition, γ, caused a significant impact on the ability of the RK methods to accurately solve the ODE. This had the most impact on the low order methods and the least impact on the higher order methods. The larger the value of γ, the more difficulty the RK methods had, in general, with solving it. This makes sense since changing γ amounts to changing the amplitude or the height of the peaks and troughs of the sin(x) and cos(x) functions, making the magnitudes of the fluctuations more intense and therefore more difficult to capture in a numerical solution.

5.3 Initial Value Problem 3: Driven System

The third initial value problem to be solved was $f = \lambda(y - F(t)) + F'(t), y(0) = c$ with solution $y(t) = (c - F(0))e^{\lambda t} + F(t)$. This is basically the same problem as in 5.1, except this time with a driving function, F(t).

To test the correctness of our implementation, all parameters were first set to 1 and the driving function was set to $F(t) = e^t$. This results in $c - F(0) = 1 - e^0 = 0$, meaning that $y = e^t$. This results in an identical problem to IVP 1 after additional computation and the algorithm was easily able to produce an identical result, as expected. In the graph, RK 2-4 are not visible since they are almost identical to the true solution, as in section 5.1. Correctness tests were also performed for $F(t) = \sin(t)$, $F(t) = \cos(t)$, and $F(t) = t^2$, and implementation in all cases was found to be accurate.

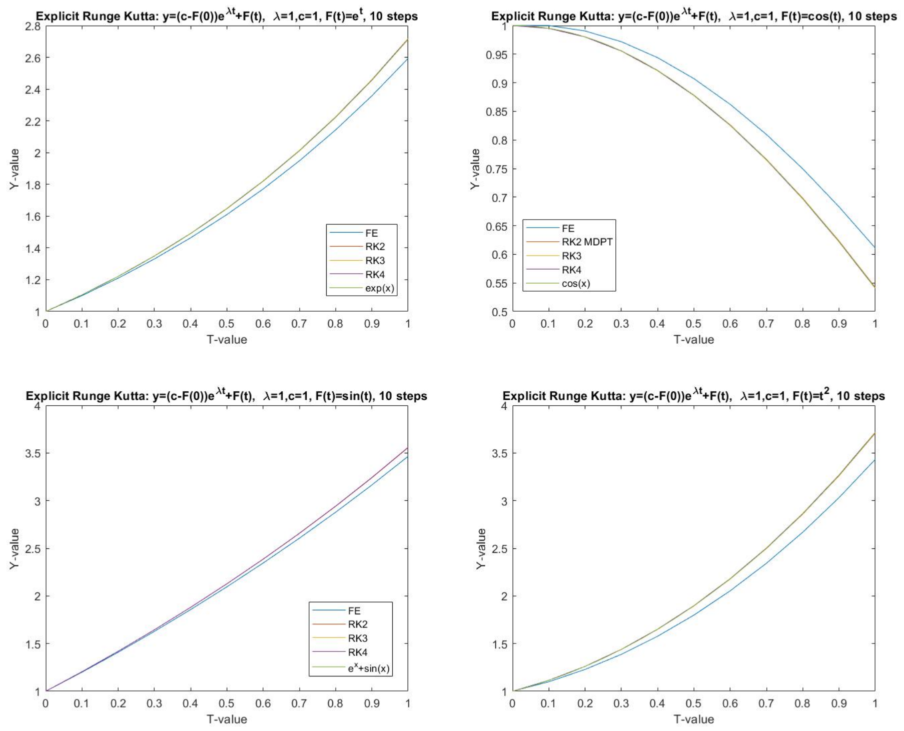


Figure 11: Correctness plots for different $F(t)$

Next, the convergence of error as $h \to 0$ was investigated, as before using the same log-log plot scheme, for the above functions excluding the trivial $F(t) = e^t$ case.

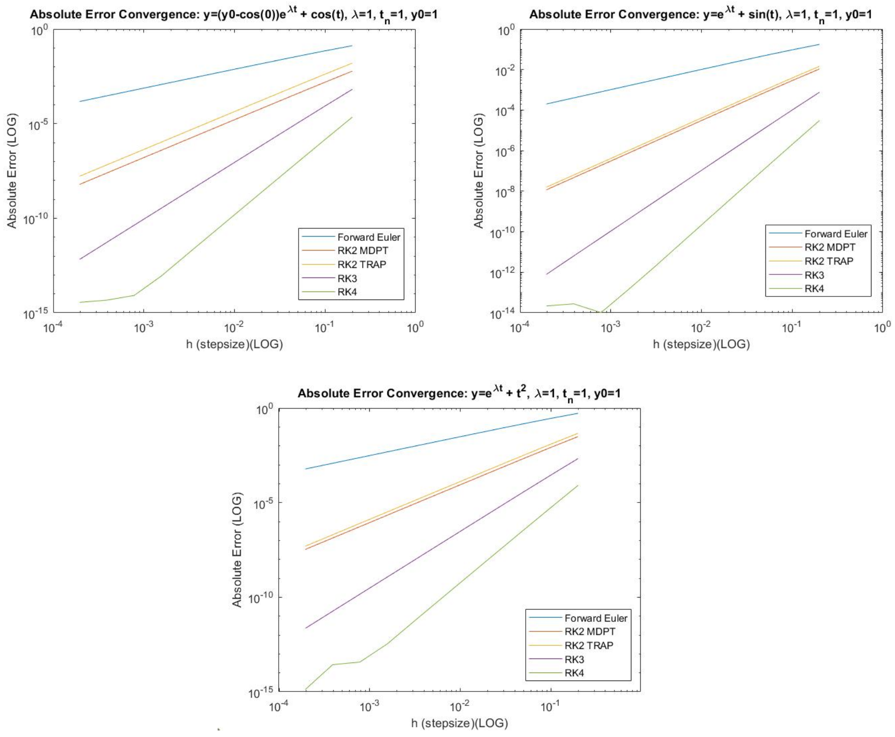


Figure 12: Error Convergence Plots for Different $F(t)$

Once again, we see the same evidence of differing orders between the methods from the differing slopes. We can also clearly see that differing the value of μ for RK2 does not change the order, confirming the theory. In all cases, explicit midpoint rule was slightly more accurate than the explicit trapezoidal rule. For RK3, μ was taken as 0.5, just like for explicit midpoint. Additionally, as with IVP 1, RK4 appears to achieve true convergence since the error levels off close to 10^{-15} or machine epsilon for double.

Although the absolute stability of the Runge Kutta methods is defined in terms of IVP 1, we can find evidence that the stability of the test problem with the additional driver function $F(t)$ has similar attributes, as was also true for the oscillator system. [3] If we plot the absolute global error against variation in λ, while keeping h constant, we can see a real-axis cross section of what appears to be a similar region to the one plotted in section 3. The error for all methods remains fairly constant and close to 0 about 1 unit in either direction around -1, before suddenly ballooning outside this interval (Figure 13). This is consistent with the behavior in the table in Figure 4. It is impossible to determine any

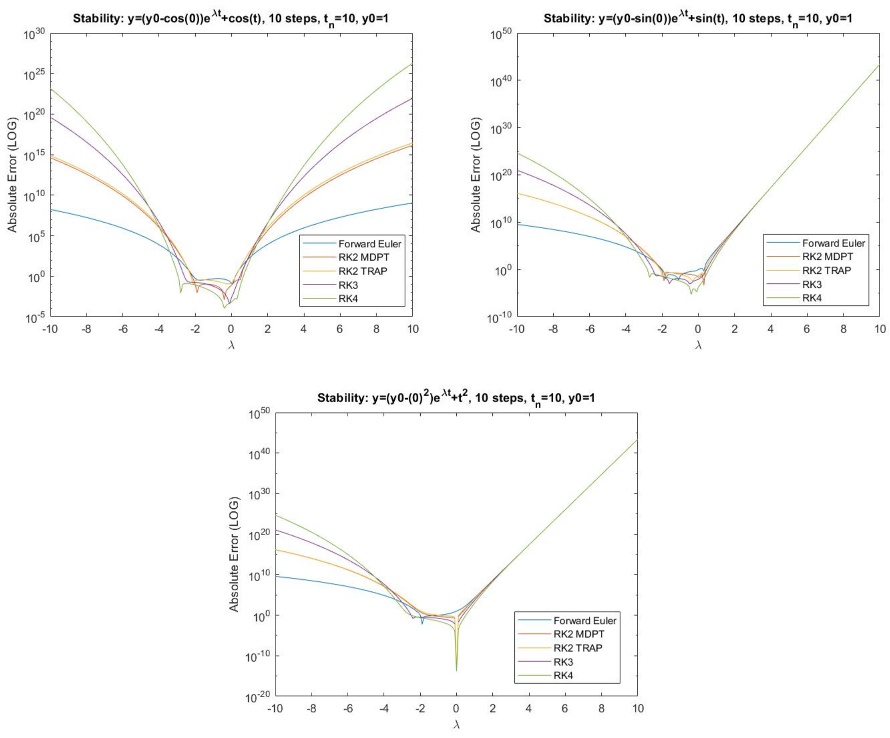


Figure 13: Absolute Error for different λ, constant h

distinct difference between different choices of μ in RK2 and no difference was found for changing μ in RK3 either.

Varying the initial condition over the range [-10,10] produced absolute error that remained constant in general for each method. $\lambda = -1$ and $t_n = 10$ with $N = 10$ were chosen to maximize stability so that any changes in error could only be attributed to the initial condition. However, there were different constant levels of error for each method and we can finally see some differences between different choices of μ for the RK2 family. However, it is not clear which choice of μ is universally superior. It instead seems dependent on the choice of $F(t)$. For instance, with $F(t) = cos(t)$, the explicit midpoint $\mu = 0.5$ produced marginally lower error than explicit trapezoidal $\mu = 1$, and vice-versa for $F(t) = sin(t)$ and $F(t) = t^2$ (Figure 14). It is interesting that in all three cases, the lines were parallel and remained that way when sloping slightly for $F(t) = sin(t)$. This is perhaps why it was impossible to distinguish them when zoomed out from a 10^a log setting, since the difference is only marginal to begin with. No differences were apparent for different choices of μ for the 3-stage Runge Kutta once again.

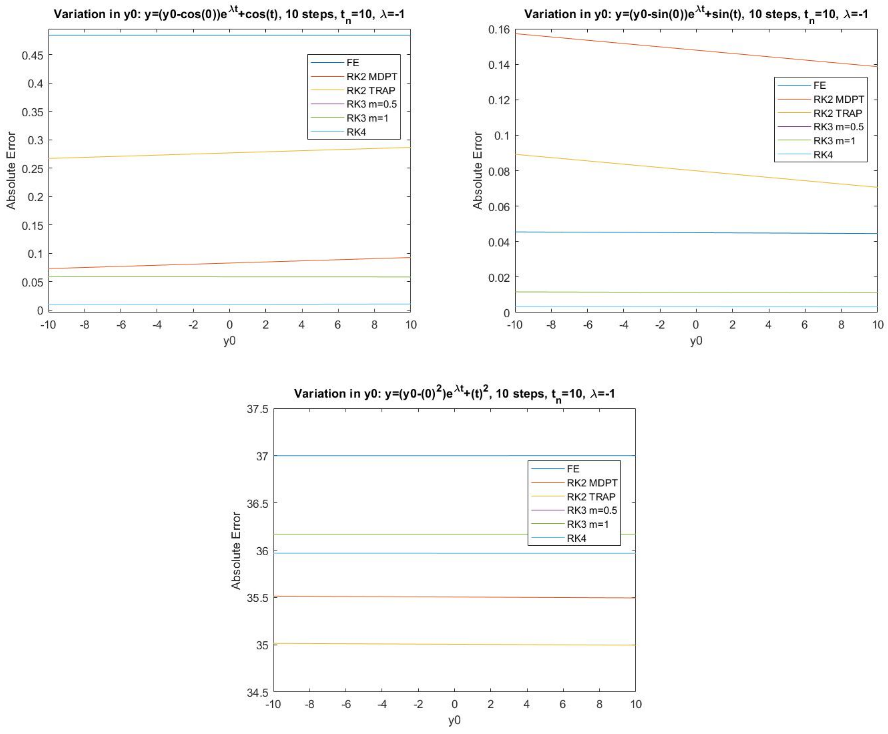


Figure 14: Absolute Error for Different Initial Conditions

6 Conclusions

Overall, the hypothesis that higher order methods would perform better in the majority of cases proved to be true. The 4-stage Runge Kutta confirmed the theory by converging the fastest for all three initial value problems, providing evidence of its increased order due to the steeper slope in the log-log plots. The 3-stage Runge Kutta also had a steeper slope than 2-stage Runge Kutta, which had a steeper slope than Forward Euler, also indicating the differing orders of the methods.

We also found behavior consistent with the absolute stability regions by taking a cross section of those plots along the real axis and then comparing the intervals where $R(z) = 1$ to the width of the stability region plot along the real axis. Those regions turned out be identical for the test problem that defines absolutes stability (IVP 1: $y = y0e^{\lambda t}$), confirming the theory. The absolute global error for differing values of λ was also examined, and a direct correspondence between the stability region and magnitude of the absolute error was found. This was then applied to the second and third initial value problems. An analysis of their absolute global errors for varying λ but constant h revealed identical stability evidence for the third and also for the second but symmetric about the imaginary axis in the $h\omega$ plane.

Changing the initial condition had an effect that was dependent on the initial value problem. For the first and third, changing $y(0)$ did not effect absolute global error much at all. However, for the second initial value problem, since the solution is a system of intertwined sin and cos functions, changing $y(0)$ changed the amplitude of the oscillations, which become harder for the Runge Kutta methods to numerically mimic for large values of t_n (i.e. after many oscillations). Higher order methods had an easier time handling this because more adjustments per time step allowed them to more effectively handle the constant dramatic changes in the slope of the solution.

The effect of changing the value of μ for the RK3 and RK2 families of methods was barely able to be distinguished. The convergence of RK3 to true solution very quickly prevented an effect from being seen at all. For RK2, explicit midpoint and explicit trapezoidal each resulted in lower absolute error depending on the choice of $F(t)$ for IVP 3. This situational evidence makes sense since the difference in the methods comes from different coefficients in the error term derived for a comparison of Taylor expansions. Different problems will have different Taylor expansions and so depending on what the coefficient is on the $O(h^3)$ term, the resulting subtraction will lead to different error coefficients [2].

7 Program Files

The program file "explicitRK.java" was compiled on my local machine with Java SE 9 using "javac explicitRK.java" and then "java explicitRK" within the Windows 10 powershell. The output .txt files were then imported into Matlab for the creation of all the graphs used in this document. To utilize different elements within main(), uncomment the appropriate sections relevant to the initial value problem that is to be solved.

References

[1] www.math.fsu.edu/ gallivan/courses/FCM1new/Locked/Sets/set30.pdf

[2] www.math.fsu.edu/ gallivan/courses/FCM1new/Locked/Sets/set27.pdf

[3] archive.siam.org/books/ot98/sample/OT98Chapter7.pdf