Math 371 FINAL PROJECT
Part 1

Bhairav Mehta [bhairavm]
Nirmal Patel [nnvpatel]
Stefano DeBellis [stefdeb]
Wyatt Kowall [wskowall]

April 15, 2016

1 Part 1: ODE Solver

This section is for Part 1 of the project. Here we use the Euler method, backward Euler method, and the fourth order Runge-Kutta method in order to solve different types of differential equations on the range of [0,1]. In addition, we will have discussion on accuracy and which methods are best for certain types of equations.

1.1 Part 1 - Question 1

Table 1: Using three methods to solve three different equations

Equation	Method	Steps	Error
Equation A	Forward Euler	536,870,912	3.43E-04
	Backward Euler	370,000,000	4.96E-05
	Runge-Kutta 4	1,770,000	4.98E-04
Equation B	Forward Euler	10,000,000	4.87E-04
	Backward Euler	1,400,000	3.38E-04
	Runge-Kutta 4	1,024	7.27E-05
Equation C	Forward Euler	524,288	1.79E-08
	Backward Euler	4	3.07E-05
	Runge-Kutta 4	362,000	1.15E-08

The step size and error for each equation with each method are delivered below in table 1. While it is mentioned above that these step sizes and errors were calculated on the range of [0,1] it was just not feasible to calculate the step size on this interval due to limitations in computing power. Thus, we reduced the interval when our group analyzed equation A, instead of [0,1] we measured from [0,1E-4]. The reason to do this for equation A is because equation A is stiff and additionally converges to zero almost instantaneously to the point that after 1E-4, the curve falls below machine precision. However, all other calculations included in the table were done with the full interval.

1.2 Part 1 - Question 2

Order of accuracy of each of the three methods we implemented.

1. The Forward Euler Method converged with an order of 1.000249, or approximately 1.
2. The Backward Euler Method converged with an order of 0.996057, or approximately 1.
3. The Runge Kutta 4 Method converged with an order of 4.122207, or approximately 4.

Appendix 2 details how the output of the calculations that were performed. For each method, a Δt was chosen to evaluate the function, much like the function was evaluated in Question 1. In addition, a smaller Δt was chosen. Specifically, this Δt was chosen to be $\Delta t/10$. Each output was then compared to the true solution, giving two arrays which contained the error at the particular timestep between the true solution and output. The absolute max was taken out of each of these error arrays, and then k, the order of convergence, was calculated as follows:

$$k = log_{10}(error(t, \Delta t)) - log_{10}(error(t, \Delta t/10))$$

1.3 Part 1 - Question 3

Results from exercise 1 have given us insight into how each of the methods work for differential equations of varying characteristics. Equation A is a very stiff equation. Equation B is a second-order differential equation, which we broke down into a system of first-order differential equations for analysis. One of the equations in this system is also a stiff equation, although less stiff than equation A. Equation C is a complex, second-order differential equation with a non-trivial solution, which we again broke down into a system of first-order differential equations. All results were obtained for an order of accuracy of at least 5E-04, or 3 digits of accuracy.

Equation A, as mentioned above, is very stiff. Forward Euler, as an explicit method, is unstable and handles stiff equations very poorly. Forward Euler requires an extremely small timestep to produce a sensible approximations for stiff equations, which is the reason 500,000,000 points were necessary. Backward Euler is an implicit method, so it is able to handle stiff equations better than forward Euler. However, because the equation is so stiff, backward Euler still took about 370,000,000 time steps. Runge-Kutta 4, on the other hand, required less than 2,000,000 time steps for convergence. RK4's approximations are much more accurate than the other two methods, giving it an enormous advantage. This is due to the approximation rules each of the methods use: RK4 uses the highly accurate Simpson's Rule approximation, while both Euler methods use crude Riemann Sum Approximations. RK4 took significantly less time steps because while forward and backward Euler converge with Δt, RK4 converges with Δt^4.

Equation B is a system of two first-order differential equations, one of which is stiff. Again, similar to equation A, forward Euler handles stiff equations very

poorly because it is an explicit method. Therefore, forward Euler required an enormous amount of steps (10,000,000). Again, backward Euler is an implicit method and therefore much more well conditioned to stiff equations. However, backward Euler still requires over 1,000,000 points to converge. Runge-Kutta 4 requires only 1,000 steps for convergence, a drastic improvement compared to both Euler methods. Again, this difference is due to the fact that RK4 converges at Δt^4 and forward and backward Euler only converge at Δt.

Equation C is a system of two first-order differential equations to which there is no trivial solution. Forward Euler was the slowest to converge again; however, this method converged within a more reasonable time step compared to equations A and B. Backward Euler yields an interesting result: only 4 time steps are required for convergence. The solution to equation C is a straight line, and backward Euler samples at the end of the region. Only two points are required to make a line, one at the beginning and one at the end. Backward Euler is based on Right Riemann Sums, which samples at the end of the region and uses the initial condition; therefore, with only two points, backward Euler creates a line. In other words, backward Euler converges so quickly on equation C because the method is well conditioned to linear solutions. Lastly, Runge-Kutta 4 converges in approximately 360,000 points, which is less than forward Euler, as expected.

1.4 Part 1 - Question 4

In order to consider which method we would use for each equation, one of the most important considerations we have to take is the number of steps that are required in order to get a certain amount of accuracy. A consequence of step size is memory usage and FLOPS which we would want to optimize if we were attempting to find a solution within 1E-07.

Therefore, looking at table 1 above, Runge Kutta 4 method is the best method for Equation A. It resolved the error within 1,770,000 number of steps, which is significantly less in comparison to the other methods. Moreover, equation A is a very stiff equation and explicit equations, such as forward Euler require a lot of memory and operations to get within precision as can be seen by the steps required to reach 5E-04 error size. Moreover, it is important to analyze the difference between backward euler and RK4. RK4 benefits from having a lot more points where it measures in between and applies its point weighting system onto. RK4 is a descendant of Simpson's, which gives a much better approximation that the Backwards Euler's, which is based off of a simple Right Hand Reimann Sum. RK4's extra points gives it an enormous advantage over Backwards Euler's crude approximation, which is why it takes so many fewer points to reach the desired error.
Additionally, for Equation B, the method that our team would use in order to solve the differential equation to 7 digits of accuracy would be the Runge-Kutta 4 method. For this equation, Runge-Kutta absolutely obliterates the competition. Equation B favors the fact that the equation is sinusoidal and thus has a high sampling error. To optimize sampling, it is likely that the methods from which Runge Kutta 4 are derived would give this method the advantage. Runge Kutta 4 works by utilizing slope calculations at multiple time steps at and in

between certain time value discretizations and then taking a weighted average of these multiple stages. RK4's approximations are based on Simpson's Rule, which is a much more accurate approximation that Riemann Sums, which is the approximation driving the Backwards Euler method.

Furthermore, for equation C, the best method is backward Euler. This implicit method went to the error of 5E-4 in 4 steps. Merely one iteration through our loop to reach such precision. The reason of this can be deduced by the fact that the solution curve of this equation is a straight line. When RK4 is implemented it adds curves and different weighting to each point. It does not make sense to implement such connections when the solution is in fact a straight line. While RK4 has to iterate through and flatten the induced curves and such, backward Euler is able to use 4 points and create the exact line plot.

Yet, while these methods were sufficient in solving the equations, if we were allowed to use any equation, we would always pick Gaussian quadrature. With error having an order of $O(h^{2n})$, it has been proven that it is the most accurate n-point rule of any kind. Gaussian quadrature is exact for all polynomials of degree $\leq$ 2n - 1, and if we needed seven digits of accuracy, Gaussian quadrature would be the most effective method.

2 Appendix for Pictures

12 images. 3 error plots for each equation and 1 solution curve for each equation.
EQUATION A:

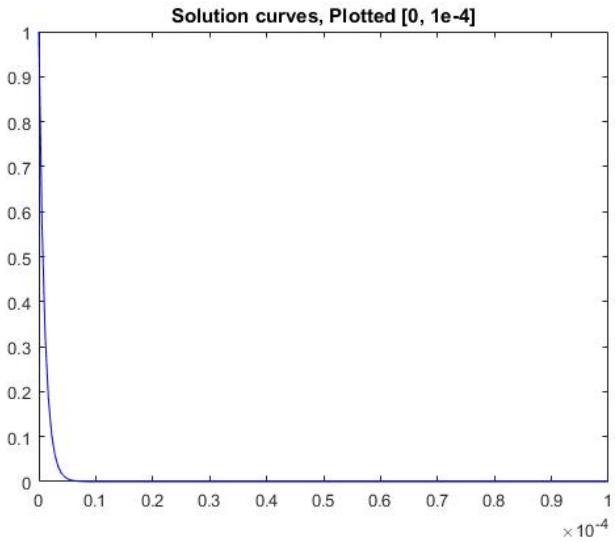

Figure 1: This Solution Curve is zoomed in on the interval [0,1E-4]. The reason for doing so was to zoom in is to analyze the immediate drop.

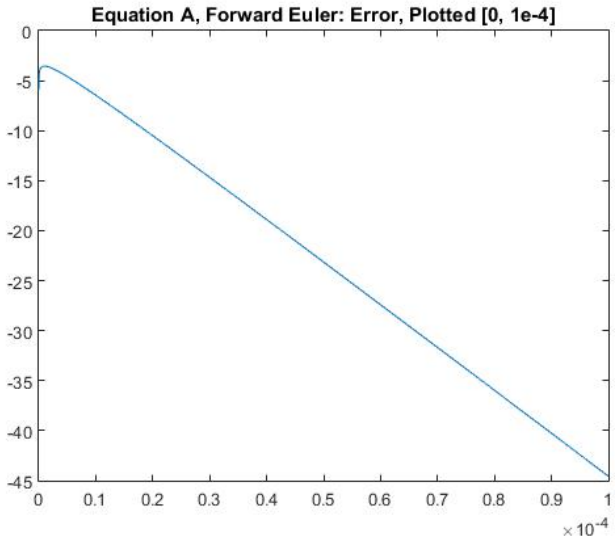

Figure 2: Error - Equation A - Forward Euler method

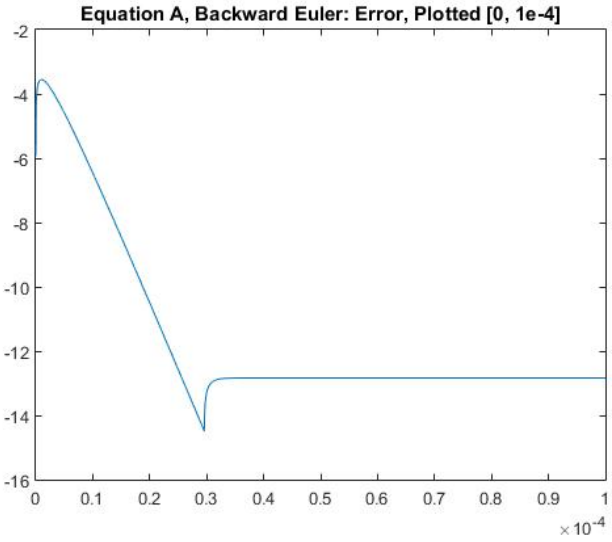

Figure 3: Error - Equation A - backward Euler method

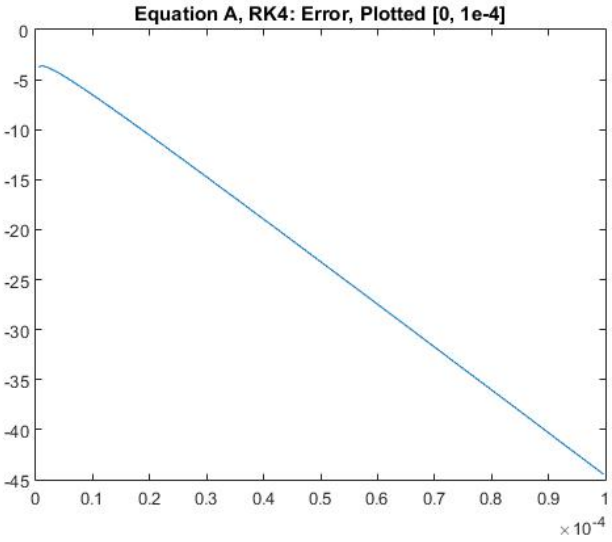

Figure 4: Error - Equation A - Runge-Kutta 4 method

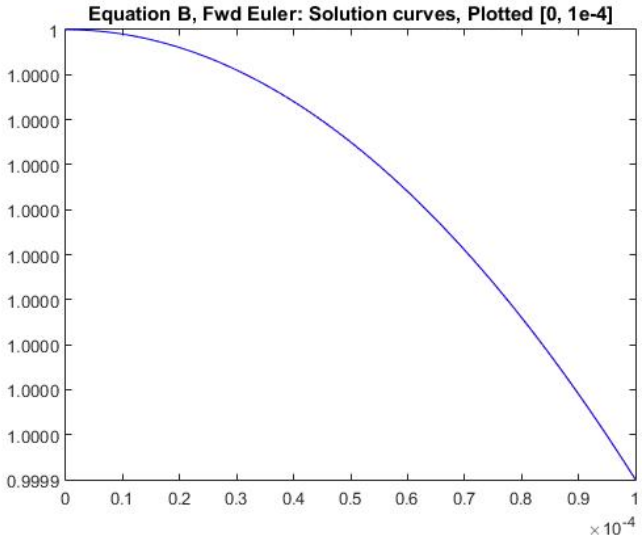

Figure 5: True Solution - Equation B

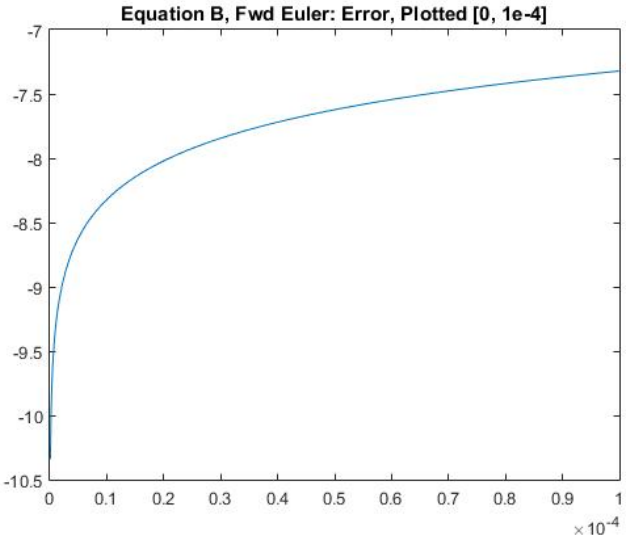

Figure 6: Error - Equation B - Forward Euler method

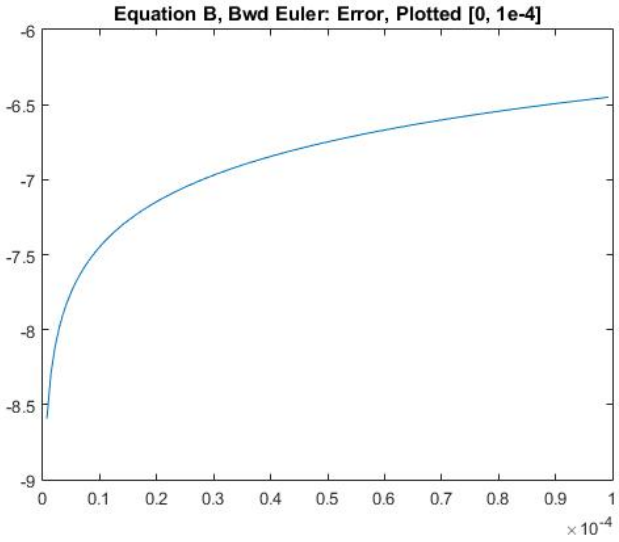

Figure 7: Error - Equation B - backward Euler method

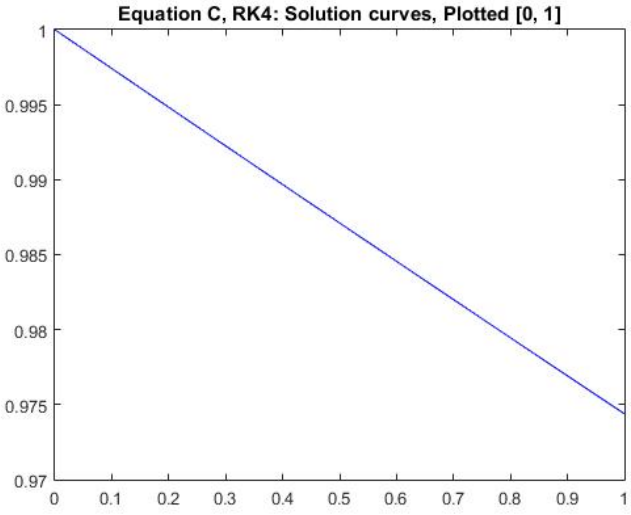

Figure 8: True Solution - Equation C

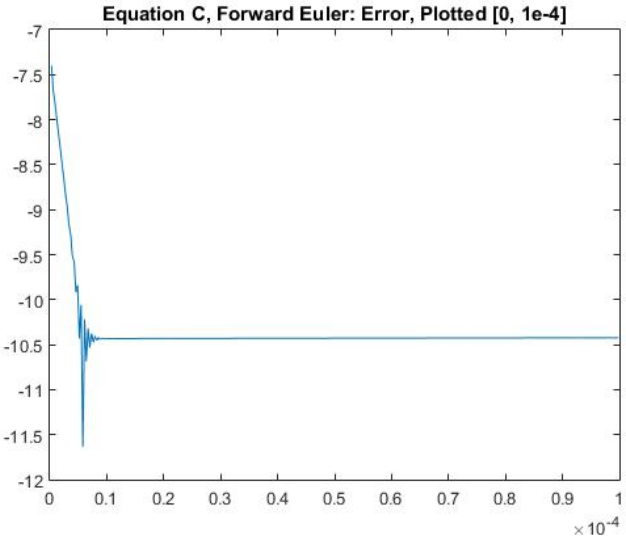

Figure 9: Error - Equation C - Forward Euler method

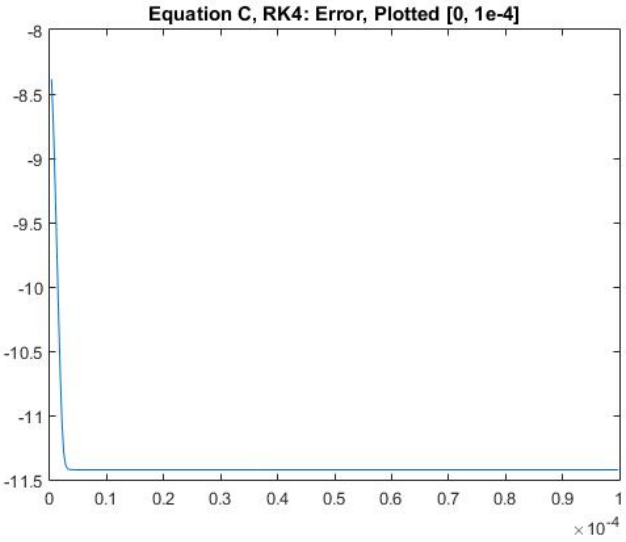

Figure 10: Error - Equation C - Runge Kutta 4 method

3 Appendix for Graphs for Part 2

Forward Euler		**Order of Convergence:**	1.000249
Δt	Error (t , Δt)	$\Delta t \div 10$	Error (t , $\Delta t \div 10$)
4	2401	40	1099511627776
8	331776	80	11057332.32
16	305902286.3	160	0.5365047969
32	333080228940	320	0.1602614285
64	29887246146	640	0.0666544548
128	0.7721113872	1280	0.0307921794
256	0.2390833618	2560	0.01485064306
512	0.08649547115	5120	0.007304099407
1024	0.0392102896	10240	0.003622139958
2048	0.01873769236	20480	0.00180363483
4096	0.009168888846	40960	0.0008999734125
8192	0.004536933566	81920	0.0004495288808
16384	0.002256840735	163840	0.0002246500764
32768	0.001125540602	327680	0.0001122964566
65536	0.00056205403	655360	5.61E-05
131072	0.0002808483374	1310720	2.81E-05

Figure 11: Solution Curve for Forward Euler Equation B

Backward Euler		Order of Convergence:	0.996057
Δt	Error (t , Δt)	Δt ÷ 10	Error (t , Δt ÷ 10)
4	49	40	4.006737947
8	24	80	1.582084999
16	11.50000373	160	0.5365047969
32	5.251930454	320	0.09651096475
64	2.168936934	640	0.05068290654
128	0.7721113872	1280	0.02698295868
256	0.1486555955	2560	0.01392239777
512	0.06210231015	5120	0.007070635485
1024	0.03321615242	10240	0.003563524189
2048	0.01725163171	20480	0.001789007577
4096	0.008802909144	40960	0.0008963209772
8192	0.004445608025	81920	0.0004486153743
16384	0.002234012213	163840	0.0002244216332

Figure 12: Solution Curve for Forward Euler Equation B

Runge Kutta 4		Order of Convergence:	4.122207
Δt	Error (t , Δt)	Δt ÷ 10	Error (t , Δt ÷ 10)
4	49	40	4.006737947
8	24	80	1.582084999
16	11.50000373	160	0.5365047969
32	5.251930454	320	0.09651096475
64	2.168936934	640	0.05068290654
128	0.7721113872	1280	0.02698295868
256	0.1486555955	2560	0.01392239777
512	0.06210231015	5120	0.007070635485
1024	0.03321615242	10240	0.003563524189
2048	0.01725163171	20480	0.001789007577
4096	0.008802909144	40960	0.0008963209772
8192	0.004445608025	81920	0.0004486153743
16384	0.002234012213	163840	0.0002244216332

Figure 13: Solution Curve for Forward Euler Equation B