FCM II Programming Assignment 4

Jonathan Engle

April 18th, 2024

1 Executive Summary

In this report we consider two major tasks in Numerical Quadrature as well as analysis and computations of Numerical Ordinary Differential Equations (ODE's). This report is an extension of previously completed results of Composite Trapezoidal rule as well as extensions to the Gauss-Legendre interpolation method. For the latter half of the report we discuss numerical methods for solving ODE's.The two methods in this section we will discuss are Forward Euler and Backward Euler methods. Graphs and other visual aids are provided to further emphasize the error analysis as well as draw backs and advantages.

2 Statement of the Problem

In this assignment we consider two major tasks in Numerical Quadrature as well as analysis and computations of Numerical Ordinary Differential Equations. As for the Numerical Quadrature we implement, test and verify the global refinement algorithms for the Composite Trapezoidal Rule and Composite two-point Gauss-Legendre Method. With regards to the analysis and computations of Numerical Ordinary Differential Equations, we explore the behavior of different methods on a finite interval $0 \leq t \leq 10$ with a fixed step size h for different choices of λ, y_0 and $F(t)$. The particular type of ODE we consider is:

$$\begin{aligned} f &= \lambda(y - F(t)) + F'(t) \\ y(0) &= y_0 \\ y(t) &= (y_0 - F(0))e^{\lambda t} + F(t) \end{aligned}$$

We then compare the observed behaviors to those predicted by the theory of local error order, global error (convergence) order, and absolute stability as stated in our textbook and Section 3.

3 Description of the Algorithms and Implementation

In this section we move to discuss the theoretical aspects and motivation behind the following methods. Note that for the computations we used Matlab. Cited works of theorems and definitions used can be found at the end of the assignment.

3.1 Numerical Quadrature

3.1.1 Composite Two-Point Gauss-Legendre Method

To implement the Composite Two-Point Gauss-Legendre Method our method is as follows :

$$a_i \le x \le b_i \text{ and } -1 \le z \le 1$$

$$x = \frac{z(b_i - a_i) + b_i + a_i}{2} \implies d \quad x = \frac{b_i - a_i}{2} dz \int_{a_i}^{b_i} f(x)dx = \frac{b_i - a_i}{2} \int_{-1}^{1} f\left(\frac{z(b_i - a_i) + b_i + a_i}{2}\right) dz$$

$$I_{gl}(f(x)) = \frac{b_i - a_i}{2} I_{gl}(F(z)) = \frac{b_i - a_i}{2} \left(\gamma_0 F(z_0) + \gamma_1 F(z_1)\right) = \frac{b_i - a_i}{2} \left(\gamma_0 f(x(z_0)) + \gamma_1 f(x(z_1))\right)$$

$$\gamma_0 = \gamma_1 = 1, z_0 = -\frac{1}{\sqrt{3}}, z_1 = \frac{1}{\sqrt{3}}.$$

Where we first perform the change of basis on our interval $[a, b]$ to match that on $[-1, 1]$ then we apply the formulation above. To implement the refinement of the errors we take sub intervals if the error is greater than our tolerance.

3.1.2 Composite Trapezoidal Rule

While the use of the Composite Trapezoidal rule was not directly used in this assignment we include a key theoretical result from assignment 3.

$$E_{0,m}(f) = \sum_{k=0}^{m-1} f''(\xi_k)(H/2)^3/3 = \sum_{k=0}^{m-1} f''(\xi_k) \frac{H^2}{24} \frac{b-a}{m} = \frac{b-a}{24} H^2 f''(\xi)$$

With the assumption that $f \in C^2([a, b])$ and $\xi \in (a, b)$. With a degree of exactness equal to 1 and 2nd order method.

3.1.3 Global refinement for Composite Trapezoidal rule

The global refinement method for the Composite Trapezoidal rule was given as:

$$I_m^{ctr} = \frac{H_m}{2} \left[f_0 + f_m + 2 \sum_{i=1}^{m-1} f_i \right]$$

$$I_{2m}^{ctr} = \frac{1}{2} \left[I_m^{ctr} + H_m \sum_{m \text{ new points}} f_i \right]$$

This method allows us to recursively use the existing points for each of the methods and refines the interval of integration increasing the order of accuracy and speed of convergence. Major discussion is discussed in Section 4.2.

3.2 Numerical Ordinary Differential Equations

The overall goal of this section of implementation is to numerically solve ODE's of the following structure of initial value problems given by:

$$f = \lambda(y - F(t)) + F'(t)$$
$$y(0) = y_0$$
$$y(t) = (y_0 - F(0))\, e^{\lambda t} + F(t)$$

With the following parameters of $\lambda \in \mathbb{R}, F : \mathbb{R} \to \mathbb{R}$, and $y_0 \in \mathbb{R}$ to define an initial value problem with solution $y : \mathbb{R} \to \mathbb{R}$ on $T_L \le t \le T_U$.

3.2.1 Forward Euler Method

The forward Euler method is an explicit constructed as follows:

$$y_n = y_{n-1} + hf_{n-1}$$

This method will decay if:

$$h < \frac{-\lambda}{2}$$

3.2.2 Backward Euler Method

The backward Euler method is an A stable implicit method constructed as follows:

$$y_n = y_{n-1} + hf_n$$

This method will decay if:

$$\frac{1}{(1-\lambda h)^k} < 1$$

4 Description of the Experimental Design and Results

In this section we explore the Experimental Design and Results for Numerical Quadrature methods as well as Ordinary Differential Equation methods.

4.1 Numerical Quadrature

4.1.1 Two-Point Gauss-Legendre Method

The main takeaways of this method were that Composite Two-Point Gauss-Legendre Method is the best at convergence to a respective threshold as well as error out of the multiple interpolation methods we considered. This makes sense intuitively due to its high order of $2n-1$. We were able to see that for a small error tolerance we only needed 9 iterations of the method where as the Composite Trapezoidal method took 1024 iterations.

4.1.2 Composite Trapezoidal Rule

For this Task we were required to perform and complete the global refinement algorithms for Composite Trapezoid rule.When observing the global refinement algorithm we are able to identify via Table 4. The computations for r are seen in Table 4. We are able to identify that as k increases from zero to ten then our r value converges to 2. Furthermore, the computed error decreases drastically as this method recursively reuses the previous mesh computations and improves accuracy. Note that this is different than in the previous task when comparing this method to the Composite Trapezoidal Method. As before the Composite Trapezoidal method worked better and was able to achieve the error threshold faster than the standard Composite Midpoint rule. But with this global refinement of the Composite Midpoint rule we are able to achieve better accuracy as well as faster convergence. A graphical representation of these observations can be seen in figure 1.

4.2 Numerical Ordinary Differential Equations

4.2.1 Local Convergence

4.2.2 Absolute stability/ Zero Stability

For this section we are able to define zero stability for one step methods as follows:

Definition 11.4 (Zero-stability of one-step methods) The numerical method for the approximation of problem is zero-stable if $\exists h_0 > 0, \exists C > 0$ such that $\forall h \in (0, h_0], \forall \varepsilon > 0$ sufficiently small, if $|\delta_n| \leq \varepsilon$, $0 \leq n \leq N_h$, then

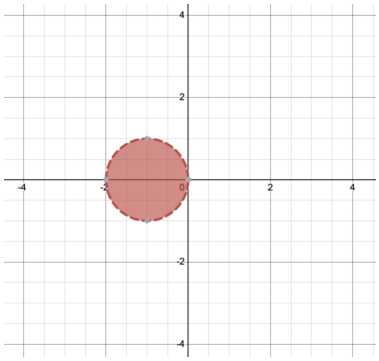
$$\left|z_n^{(h)} - u_n^{(h)}\right| \leq C\varepsilon, \quad 0 \leq n \leq N_h,$$

This is the theoretical result obtained in class and defined in our textbook. We are further able to identify this numerically for the backward and forward Euler implementation on $F(t) = 0, y(0) = 1$. This graphical representation of zero stability of one step methods can be seen in Figure 1.

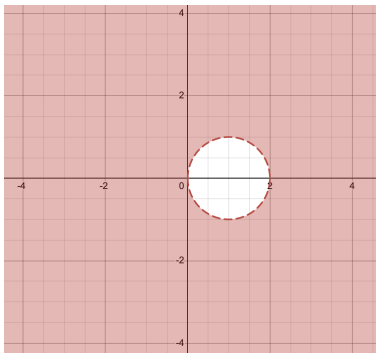
Furthermore we can discuss the stability region of the methods used. For Forward EUler Scheme, we allow λ to be a complex number such that

$$x := \lambda h \in \mathbb{C}$$

Then we obtain the stability region of:(Note: The graph represents the complex plane)



Similarly we identify the stability region for Backward Euler method but this time identifying that $|1 - z| > 1$ is the stabulitiy region for the scheme, note bellow that it includes all of $Re(z) < 0$ such that if $Re(z) < 0 \implies |x^k| \to 0$. This is precisely the definition of A-Stable. The stability region of Backward Euler method is bellow: (Note: The graph represents the complex plane)



4.2.3 Accuracy and Stability

This area of the project was pretty fascinating. We were able to test the accuracy and stability with varying parameters for $F(t) = \sin(\omega t), y(0) = 0$ i.e

$$\begin{aligned} f &= \lambda(y - \sin(\omega t) + \omega \cos(\omega t) \\ y(0) &= 0 \\ y(t) &= \sin(\omega t) \end{aligned}$$

We are able to see that when $\omega = 0.01$ i.e relatively small, that we can keep a higher step length to achieve a "good" accuracy. But when $\omega = 10$ and there are is quick change we obtain a large error best seen in Figure 4 and 3. The accumulation error throughout is large and the forward and backward methods do not match up. To combat this we need to take a small step size to achieve a higher accuracy. This matches the theory of having a high Lipschitz constant implies a smaller step size to minimize the error Table 2, Table 3 and Figures 3-5 depict these observations. Furthermore for the true solution of $F(t) = e^{\lambda t}$ we can see that for $\lambda = -1$ that the convergence of the methods is better, this is consistent with the in class theory as well as the figures above. Especially for the backward Euler method. For $\lambda = 1$ we can see that issues emerge in particular with Forward Euler method, please refer to the λ requirements above.

4.3 Additional testing

Please observe the additional testing done at the end of the document. We were further able to identify the necessary conditions obtained in class for the true solution of the exponential function. This is shown in figure 7 and 8. For a $\lambda > 0$ then we have large error terms for any perturbation of the "data" and larger errors to the true solution. But, for $\lambda < 0$ then we are able to see a tighter error bound as t gets larger. Furthermore, for the $\lambda < 0$ case we also obtain faster convergence rates as $\Delta t \to \infty$.

5 Conclusion and Comparison of Methods

Overall, we are able to analyze and compare the computational powers and accuracy of numerical integration as well as Forward and Backward Euler. The two types of quadrature that we considered through this project were: Composite Trapezoidal rule and Composite Two-Point Gauss-Legendre Method. To implement these methods we consider the test integral about the function $\int_0^3 e^x dx = e^3 - 1 \approx 19.0855$. Note that we are able to change this function to best fit real world applications and problems. The main takeaways of this project were that Composite Two-Point Gauss-Legendre Method is the best at convergence to a respective threshold as well as error. This makes sense intuitively due to its high order of $2n - 1$. As for the analysis of Forward Euler and Backward Euler methods, we were able to identify that for smooth functions without a lot of change in concavity then we are able to minimize the error. But as seen in the case when $\omega > 1$ then we ran into issues with error and had to increase our step numbers. This also makes sense since the Forward and Backward Euler schemes are $\mathcal{O}(h)$. Overall, this project was able to verify the in class numerical schemes, theories and results.

6 Theorems, Tables, and Figures

6.1 Theorems, Properties, Citations

- **Theorem 10.1** For a given $m > 0$, the quadrature formula (10.13) has degree of exactness $n + m$ iff it is of interpolatory type and the nodal polynomial ω_{n+1} (8.6) associated with the nodes $\{x_i\}$ is such that

$$\int_{-1}^{1} \omega_{n+1}(x)p(x)w(x)dx = 0, \quad \forall p \in \mathbb{P}_{m-1}$$

- **Definition 11.4** (Zero-stability of one-step methods) The numerical method for the approximation of problem is zero-stable if $\exists h_0 > 0, \exists C > 0$ such that $\forall h \in (0, h_0], \forall \varepsilon > 0$ sufficiently small, if $|\delta_n| \leq \varepsilon$, $0 \leq n \leq N_h$, then

$$\left|z_n^{(h)} - u_n^{(h)}\right| \leq C\varepsilon, \quad 0 \leq n \leq N_h,$$

- **Theorem 11.1** (Zero-stability) Consider the explicit one-step method for the numerical solution of the Cauchy problem . Assume that the increment function Φ is Lipschitz continuous with respect to the second argument, with constant Λ independent of h and of the nodes $t_j \in [t_0, t_0 + T]$, that is

$$\begin{aligned}
&\exists h_0 > 0, \exists \Lambda > 0 : \forall h \in (0, h_0] \\
&\left|\Phi\left(t_n, u_n^{(h)}, f\left(t_n, u_n^{(h)}\right); h\right) - \Phi\left(t_n, z_n^{(h)}, f\left(t_n, z_n^{(h)}\right); h\right)\right| \\
&\leq \Lambda\left|u_n^{(h)} - z_n^{(h)}\right|, 0 \leq n \leq N_h.
\end{aligned}$$

  Then, the single step method is zero-stable.

- **Definition 11.6** A numerical method for approximating ODE's is absolutely stable if

$$|u_n| \longrightarrow 0 \text{ as } t_n \longrightarrow +\infty$$

6.2 Tables

Table 1: Iterations for Composite Trapezoidal and Composite Two-point Gauss Legendre methods

Method	$\|Error\| \leq 0.01$	$\|Error\| \leq 0.0001$
Composite Trapezoidal	128	1024
Composite Two-point Gauss Legendre	4	9
Theoretical 2pt GL	4	11

Table 2: Euler errors for $F(t) = 0, f(0) = 1, h = 0.001$

Method	Error
Forward Euler $\lambda = 1$	0
Forward Euler $\lambda = -1$	0
Backward Euler $\lambda = 1$	0
Backward Euler $\lambda = -1$	0

Table 3: Euler errors for $F(t) = \sin(\omega t), f(0) = 0, h = 0.1$

Method	Error
Forward Euler $\lambda = -1, \omega = 0.01$	$4.5103e{-}07$
Forward Euler $\lambda = -1, \omega = 10$	0.9057
Forward Euler $\lambda = -0.01, \omega = 0.01$	$2.4099e{-}06$
Forward Euler $\lambda = -0.01, \omega = 10$	0.9958
Backward Euler $\lambda = -1, \omega = 0.01$	$4.5899e{-}06$
Backward Euler $\lambda = -1, \omega = 10$	1.5237
Backward Euler $\lambda = -0.01, \omega = 0.01$	$2.5883e{-}06$
Backward Euler $\lambda = -0.01, \omega = 10$	1.0500

Table 4: Euler errors for $F(t) = \sin(\omega t), f(0) = 0, h = 0.0001$

Method	Error
Forward Euler $\lambda = -1, \omega = 0.01$	$4.4940e-10$
Forward Euler $\lambda = -1, \omega = 10$	$8.6411e-04$
Forward Euler $\lambda = -0.01, \omega = 0.01$	$2.4168e-09$
Forward Euler $\lambda = -0.01, \omega = 10$	$9.9843e-04$
Backward Euler $\lambda = -1, \omega = 0.01$	$4.5465e-09$
Backward Euler $\lambda = -1, \omega = 10$	0.0015
Backward Euler $\lambda = -0.01, \omega = 0.01$	$2.5791e-09$
Backward Euler $\lambda = -0.01, \omega = 10$	0.0010

Table 5: Euler errors for $F(t) = e^{(\lambda t)}, f(0) = 0, h = 0.1$

Method	Error
Forward Euler $\lambda = -1$	0.0192
Forward Euler $\lambda = 1, \omega = 10$	$8.2459e+03$
Backward Euler $\lambda = -1,$	0.100
Backward Euler $\lambda = 1$	$6.8679e+03$

Table 6: Euler errors for $F(t) = e^{(\lambda t)}, f(0) = 0, h = 0.0001$

Method	Error
Forward Euler $\lambda = -1$	$1.8395e-05$
Forward Euler $\lambda = 1, \omega = 10$	11.0097
Backward Euler $\lambda = -1$,	$9.9973e-05$
Backward Euler $\lambda = 1$	8.8083

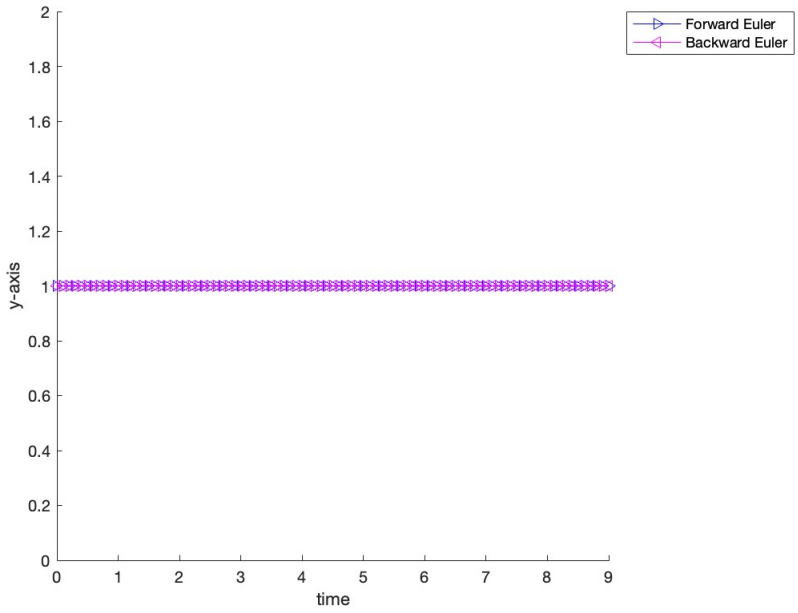
Table 6: Global refinements for Composite Trapezoidal rule, varying k :

k	$m = 2^k$	Function Evals	r	Error
1	2	3	1.8169	3.4511
2	4	5	1.9612	0.8864
3	8	9	1.9897	0.2231
4	16	17	1.9994	0.0559
5	32	33	1.9997	0.0140
6	64	65	1.9998	0.0035
7	128	129	1.9999	$8.7366e-04$
8	256	257	1.9999	$2.1842e-04$
9	512	513	2	$5.4604e-05$
10	1024	1025	2	$1.3651e-05$

6.3 Figures

6.3.1 $F(t) = 0, y(0) = 1$

Figure 1: $\lambda = 0, f(0) = 1$



6.3.2 $F(t) = \sin(\omega t), y(0) = 0$

Figure 2: $h = 0.1, \lambda = -0.01, \omega = 0.01$ Similar for $\lambda = -1$

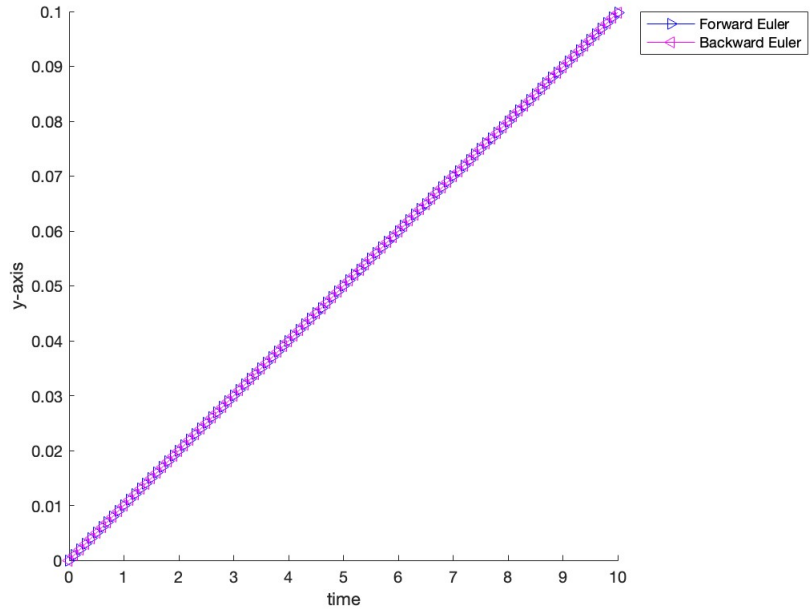


Figure 3: $h = 0.1\lambda = -0.01, \omega = 10$

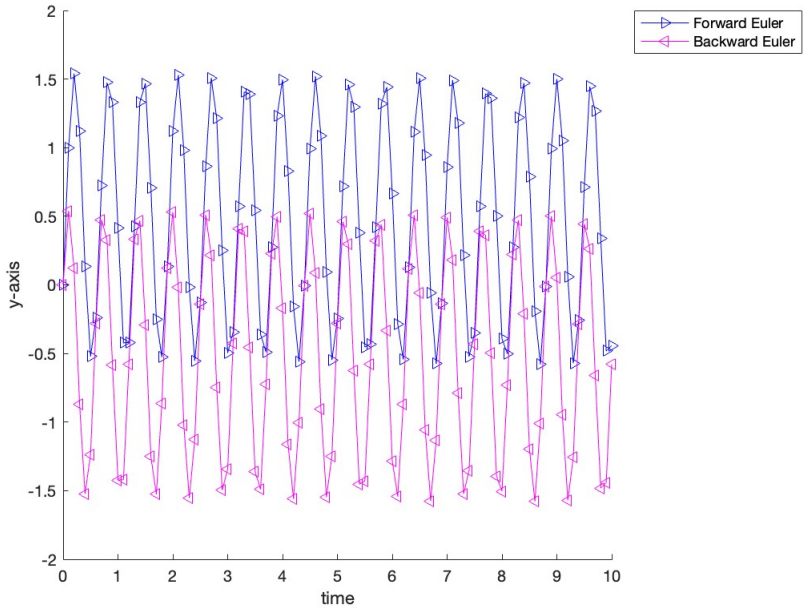


Figure 4: $h = 0.1$

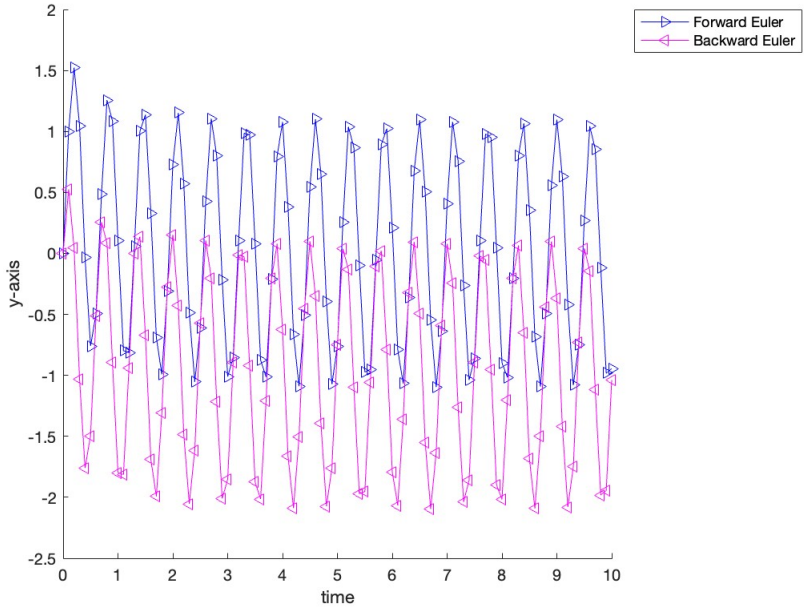
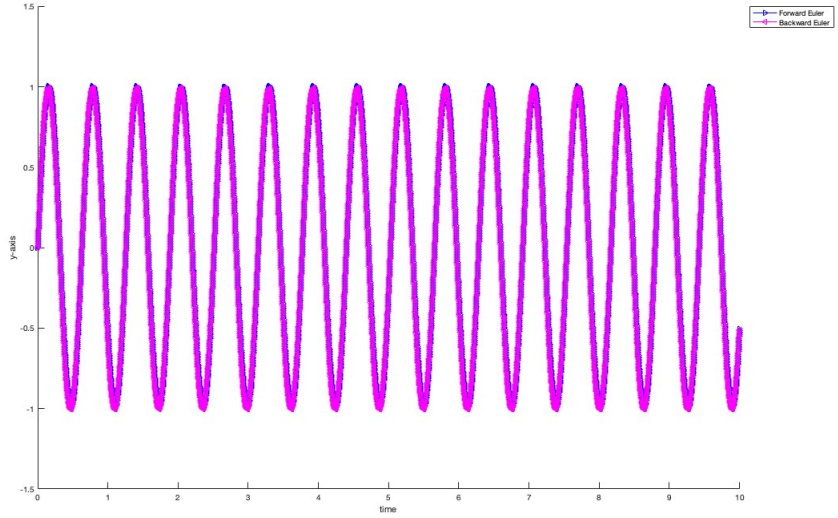


Figure 5: $h = 0.0.001, \lambda = -1, \omega = 10$ Error still decently big just need to make smaller step sizes.



6.3.3 $F(t) = \lambda y, y(0) = 1$

Figure 7: $h = 0.001, \lambda = 1$

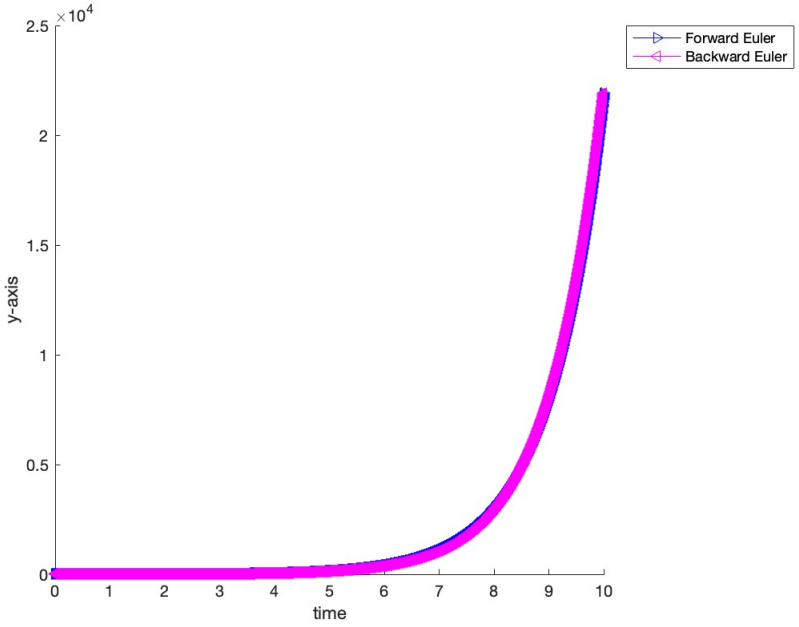
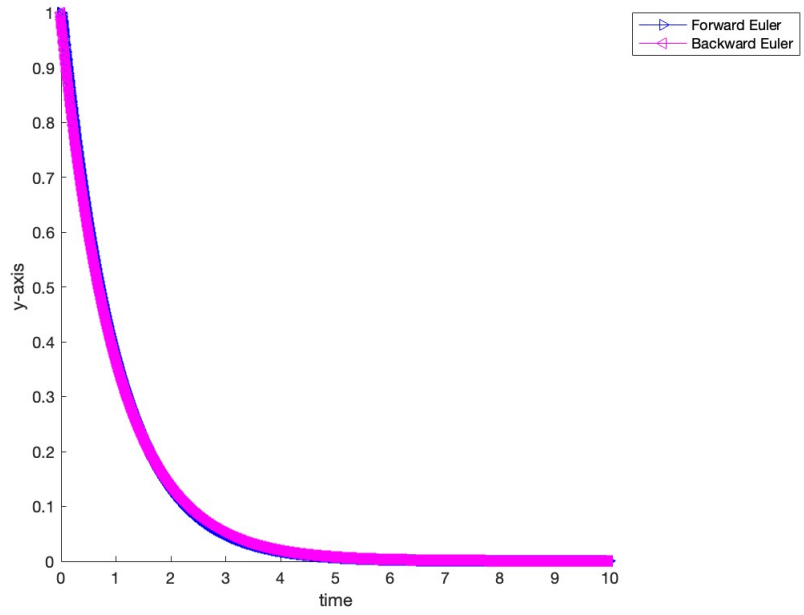


Figure 8: $h = 0.001, \lambda = -1$



6.3.4 Additional Testing and Cases

Figure 6: $F(t, y) = 1, 5, 10, 15, f(0) = 1, h = 0.1$