土木所魏廷儒 r10521601
MATH 7422: Numerical Partial Differential Equations
March 8, 2023

Project 1'

Question 1. You can study the behaviors of various ODE solvers for the prototype equations. (1) $y' = ay$ with $a = -100$ (2) ${y_1}' = y_2, y_2 = -a^2 y_1$ Try various explicit, and implicit schemes. Explore the absolute stability region. What happens if $z = ak$ does not lie inside the stability region, etc.

I approached the problem in two ways. First, I implemented the implicit scheme with the 1st-order backward Euler method, and used Newton's method to find the root from scratch. I also implemented explicit schemes using the 1st-order forward Euler method from scratch.Then, I using the 1st-order forward Euler method and high-order solvers from the `Scipy` library:

1) RK45: Explicit Runge-Kutta method of order 5(4)
2) RK23: Explicit Runge-Kutta method of order 3(2)
3) DOP853: Explicit Runge-Kutta method of order 8
4) Radau: Implicit Runge-Kutta method of the Radau IIA family of order 5
5) LSODA: Adams/BDF method with automatic stiffness detection and switching.

The ODE $y' = -100y$ is a simple prototype equation with a known analytical solution $y(t) = y_0 e^{-100t}$, where y_0 is the initial value of 1 at $t = 0$.

For the implicit scheme with the 1st-order backward Euler method, denoted as BDF below, I performed numerical experiments to analyze its absolute stability region. The theoretical region is given by $\{z \in \mathbb{C} : |1 - z| \geq 1\}$ in lecture notes. To examine the sensitivity of the time step size, I varied the time step within the range of 1×10^{-1} to 1×10^{-4}.

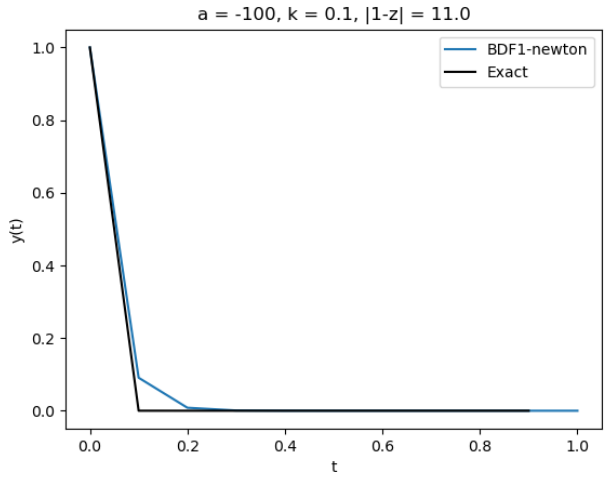

FIGURE 1. BDF: k=0.1

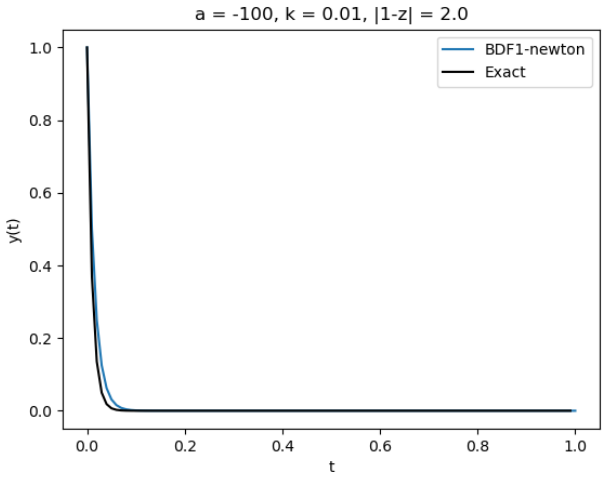

FIGURE 2. BDF: k=0.01

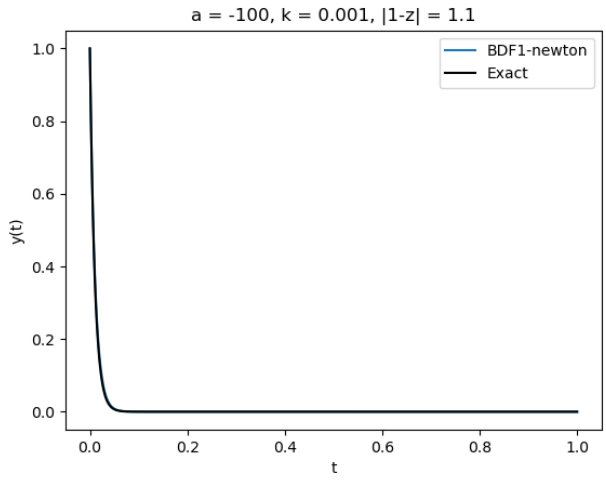

FIGURE 3. BDF: k=0.001

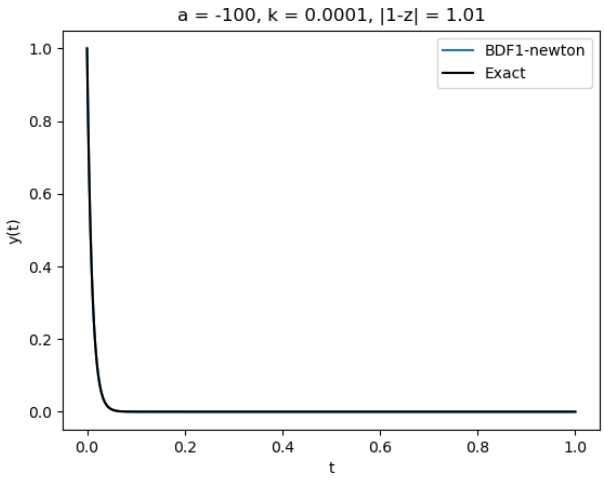

FIGURE 4. BDF: k=0.0001

In Figures 1 to 4, we observe that as the time step size decreases, the numerical error increases. Fortunately, even for the case where $k = 0.1$, the model remains within the absolute stability region. After conducting these experiments, I became curious about what would happen if the time step falls outside of the absolute stability region. To explore this, I considered the ODE $y' = 10y$ and varied the time step within the range of 1×10^{-2} to 1×10^{-4}, as shown as Figure5 - 7.

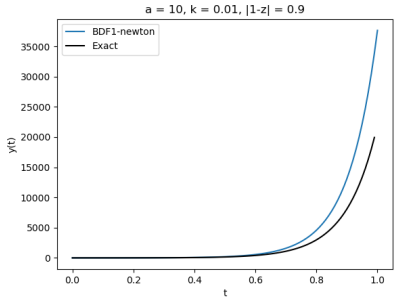

FIGURE 5. $y' = 10y$ with k=0.01

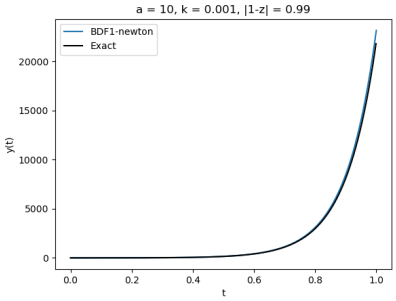

FIGURE 6. $y' = 10y$ with k=0.001

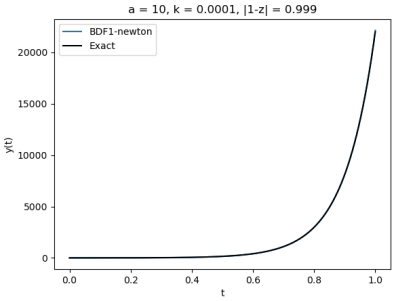

FIGURE 7. $y' = 10y$ with k=0.0001

After conducting these experiments, I became curious about what would happen if the model were to fall outside of the absolute stability region. To investigate this, I assumed that my ODE was $y' = 10y$ and varied the time step within the range of 1×10^{-2} to 1×10^{-4}. As shown in Figure 5, we can see that all of the models in this ODE are outside of the absolute stability region, even for small time step sizes. From the result, we can observed that the numerical error is significant large, especially in the Figure 5.

For explicit scheme with the 1st-order forward Euler method, denoted as Forward Euler1 below. The theoretical absolute stability region is given by $\{z \in \mathbb{C} : |1 + z| \leq 1\}$. To examine the sensitivity of the time step size, I varied the time step within the range of 1.5×10^{-1} to 1×10^{-4}

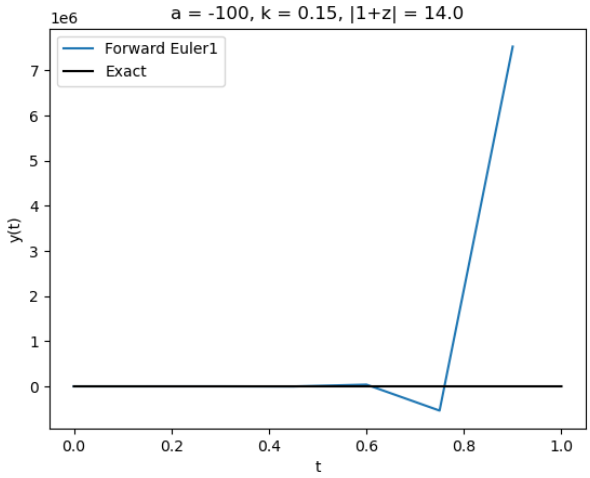

FIGURE 8. Forward Euler1: k=0.15

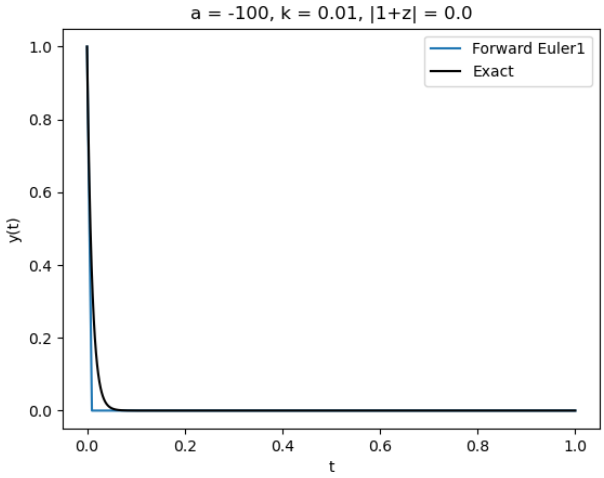

FIGURE 9. Forward Euler1: k=0.01

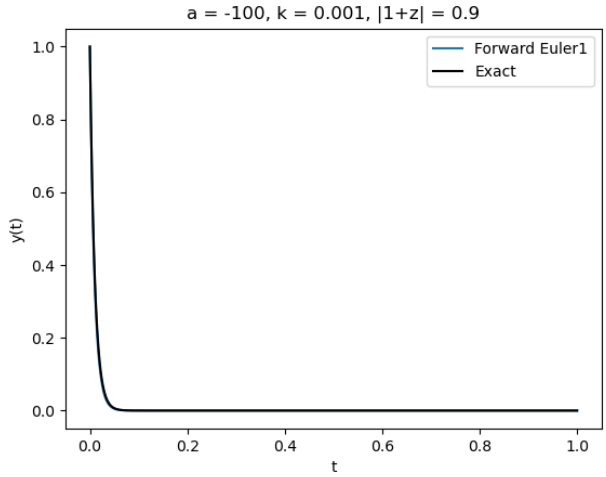

FIGURE 10. Forward Euler1: k=0.001

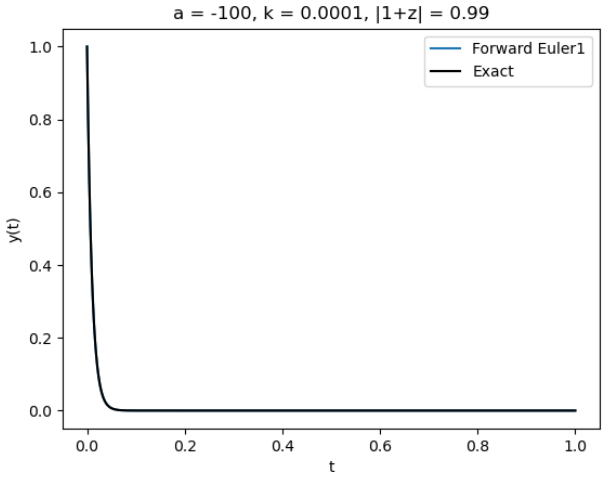

FIGURE 11. Forward Euler1: k=0.0001

Figures 9 to 11 demonstrate that the numerical results converge as the time step size decreases, except for when the time step size is 0.15. This failure to converge is likely due to the fact that the model falls outside the absolute stability region, as shown in Figure 8.

Below is chatGPT thought of this issue: In explicit methods, the stability region is determined by the absolute value of the amplification factor $|1 + k\lambda|$, where λ is the eigenvalue of the linear operator in the differential equation. For a stable method, we want $|1 + k\lambda| < 1$ for all eigenvalues λ. This means that as k increases, the amplification factor increases and the method becomes more unstable. In contrast, implicit methods have larger stability regions because they use backward differences, which involve solving a linear system of equations at each time step. This allows implicit methods to use larger time steps while maintaining stability. However, implicit methods require more computational resources at each time step, which can make them slower than explicit methods for some problems.

In my opinion, the results of our experiments suggest that the explicit scheme is generally more challenging to converge than the implicit scheme. This may be due to the absolute stability region condition, which involves the term $|1 + ka|$. When a is negative, as in our prototype equation $y' = -100y$, the value of $|1 + ka|$ increases as k increases. Consequently, when using explicit schemes with large values of k, the resulting time step may fall outside of the absolute stability region, leading to numerical instability and convergence issues.

Additionally, I also experimented with some high-order numerical methods to solve this ODE. I varied the time step size from 0.1 to 0.0001. As shown in Figure 12 - 14, all the numerical methods tend to converge, including the explicit scheme with a large time step size (RK45).

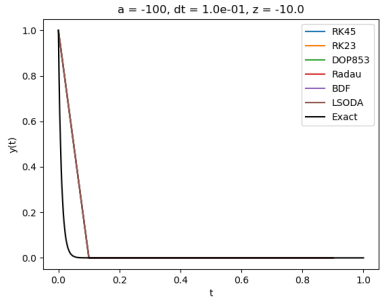

FIGURE 12. k=0.1

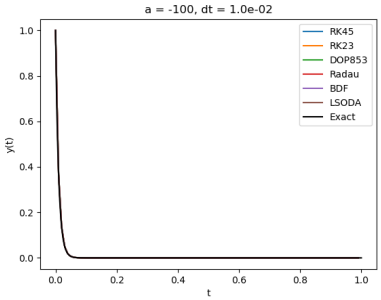

FIGURE 13. k=0.001

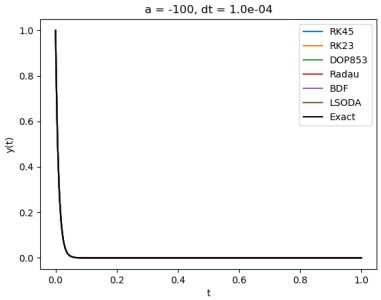

FIGURE 14. k=0.0001

Question 2. $y_1{}' = y_2, y_2{}' = -a^2 y_1$

First, I assigned $a = 10$ and the initial conditions $y_1(0) = 1$ and $y_2(0) = 0$. The system of ODEs can be written as:

$$\begin{bmatrix} y_1{}' \\ y_2{}' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -100 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

The coefficient matrix has eigenvalues $\lambda_1 = ai$ and $\lambda_2 = -ai$, and corresponding eigenvectors $v_1 = \begin{bmatrix} 1 & i/a \end{bmatrix}$ and $v_2 = \begin{bmatrix} 1 & -i/a \end{bmatrix}$.

The exact solution can be written as:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = c_1 v_1 e^{-iat} + c_2 v_3 e^{iat} = \begin{bmatrix} A\cos(at) + B\sin(at) \\ C\cos(at) + D\sin(at) \end{bmatrix}$$

Using the initial conditions, we can solve for the exact solution as:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \cos(10t) \\ \frac{1}{10}\sin(10t) \end{bmatrix}$$

In this subproblem, I will use numerical methods to solve the ODE system $y_1{}' = y_2, y_2{}' = -a^2 y_1$ with the initial conditions $y_1(0) = 1$ and $y_2(0) = 0$. Specifically, I will use the 1st-order backward Euler method and 1st-order forward Euler method, as well as high-order solvers from the Scipy library, to compare their convergence abilities.

In the 1st-order backward Euler method, I examined the sensitivity of the time step size by varying the time step within the range of 1×10^{-2} to 1×10^{-4}, as shown in Figure 15 and Figure 16. The theoretical region is given by $z \in \mathbb{C} : |1 - z| \geq 1$. It can be observed that regardless of the smallness of the time step size, the numerical solution falls outside the absolute stability region and results in divergent behavior. If we look deep in, we can found that the nurmical result is decrease exponentially; however, the phase has no lag with the exact solution. If we take a closer look at the numerical solution, we can see that it decreases exponentially, which cause the numerical error. However, the phase of the numerical solution has no lag with respect to the exact solution. This means that the backward Euler method is not introducing any phase error. In Figure 15, we can see that if we have a small enough time step size, we can still obtain a numerical solution that is somewhat acceptable.

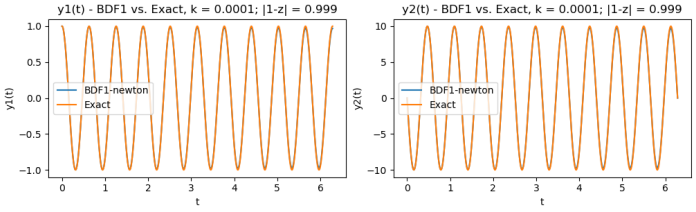

FIGURE 15. BDF: k=0.0001

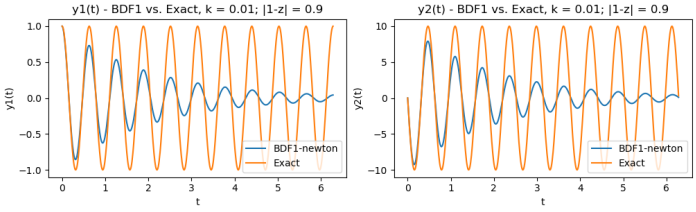

FIGURE 16. BDF: k=0.01

In the 1st-order forward Euler method, I also examined the sensitivity of the time step size by varying the time step within the range of 1×10^{-2} to 1×10^{-3}, as shown in Figure 17 and Figure 18. The theoretical absolute stability region is given by $z \in \mathbb{C} : |1 + z| \leq 1$. As shown in Figure 17 and Figure 18, the numerical model falls outside the absolute stability region. In contrast to the backward Euler method, the numerical result grows exponentially. However, both numerical solvers exhibit no phase lag.

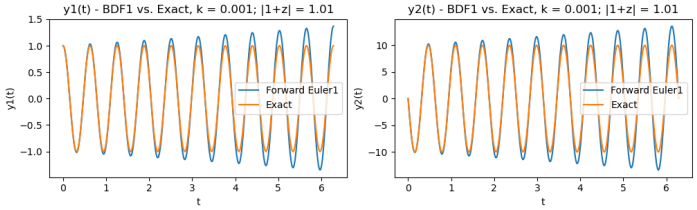

FIGURE 17. Forward Euler1: k=0.001

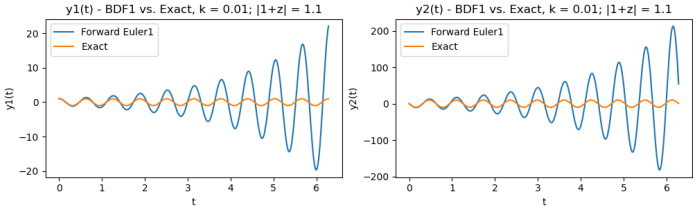

FIGURE 18. Forward Euler1: k=0.01

The utilization of a high-order solver has resulted in more stable numerical results compared to the two previously mentioned solvers, as depicted in Figures 19 through 24.

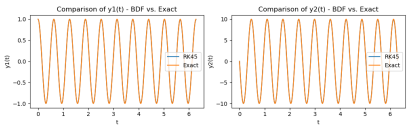

FIGURE 19. BDF k=0.1

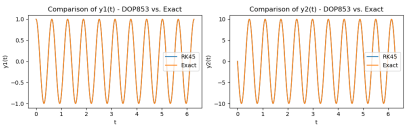

FIGURE 20. DOP853 k=0.001

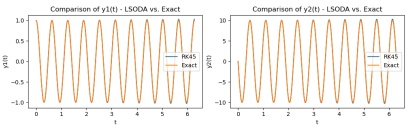

FIGURE 21. LSODA k=0.0001

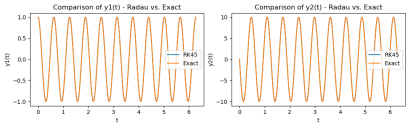

FIGURE 22. Radau k=0.1

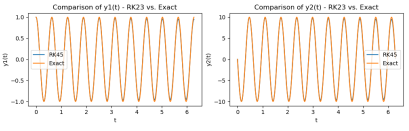

FIGURE 23. RK23 k=0.001

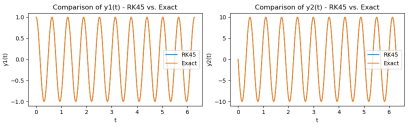

FIGURE 24. RK45 k=0.0001

In this project, the backward Euler method and forward Euler method are implemented in the `ODESolver` class. All of the code can be found in the repository (https://github.com/BerryWei/nurmical-PDE/tree/main/project1).

R10521601