Mathematical modeling lab: group project

By Ankit Lamsal & Maroua Ouahbi

INTRODUCTION

In this project, we explore the fascinating dynamics of a damped harmonic oscillator. This system, described by the equation:

$$\ddot{x} + 2\zeta\dot{x} + x = 0 \tag{1}$$

is a fundamental model in physics. We examine how it behaves under different damping conditions, and more specifically for three values of the damping ratio: ζ=0.25, ζ=1, and ζ=2.

To solve this equation numerically, we use several methods: the Explicit Euler method, the Implicit Euler method, and a built-in solver from the SciPy library. We also tackle a boundary value problem for the same system using the Shooting method, the Finite Difference method, and SciPy's solve_bvp function.

Our goal is to see how these methods perform in terms of accuracy and efficiency, and to understand their strengths and weaknesses in solving both initial value and boundary value problems, and using the error analysis to determine which is the most precise of the methods.

METHODOLOGY

In this part of the report we aim to explain the Python code and give some observations on each section of the two problems.

Problem 1:

As a first step to solving the damped oscillator equation we had to convert it to a first order system so that it is easier to solve. We defined velocity as how position changes over time. Then we rearranged the equation to show acceleration and introduced the velocity variable, resulting in a system of two first-order therefore simpler equations.

$$\begin{cases} \dot{x} = v \\ \dot{v} = -2\zeta v - x \end{cases}$$

Using the explicit Euler method the solution is incrementally updated at each time step. It starts by dividing the time interval [t_0, t_f] into smaller steps of size h. The solution is initializes on the given initial conditions $x(0) = 1, \dot{x}(0) = 0$. For each step, the derivative function f(t_{i-1}, y_{i-1}) is computed, which represents the slope at the current position. Using this same slope, it predicts the next value of y by multiplying the slope with h and adding it to the current y. The iteration process approximates the solution through a given time interval. After the creation of the explicit_euler function, we proceed to call it to numerically solve the damped oscillator equation for various damping parameters ζ (0.25, 1, 2) and step sizes h (0.1, 0.01).

We create a plot titled "Phase Space Trajectories - Explicit Euler," with the x-axis representing position (x) and the y-axis representing velocity (v). The plot shows trajectories for different damping parameters (ζ) and step sizes (h), and includes a legend and grid.

The implicit Euler method updates the solution incrementally in an interval of time, similar to the explicit method. It divides the time interval into small steps and h, and starts with the same initial conditions $x(0) = 1, \dot{x}(0) = 0$. However, unlike the explicit method it computes each new y using the slope at the current position f(t_i,y_guess)f(t_i,y_guess). This requires a fixed-point iteration to refine y until the difference between successive guesses is small enough. After defining the implicit_euler function similarly to the explicit_euler function, we apply it to solve the damped oscillator equation for various damping parameters ζ (0.25, 1, 2) and step sizes h (0.1, 0.01).

We create a plot titled "Phase Space Trajectories - Implicit Euler," similar to the one that was made for the explicit Euler method, to show our trajectories for the implicit Euler method.

The SciPy `solve_ivp` function uses the RK45 method to solve our equation, which adapts the step size for better accuracy and efficiency. We set the initial conditions and time range `[t_0, t_f]`, then call `solve_ivp` to solve the damped oscillator equation for different

damping parameters ζ (0.25, 1, 2). This function returns the solution with the time points and corresponding values.

We create a plot titled "Phase Space Trajectories - SciPy Solver," similar to the ones for the explicit and implicit Euler methods. This plot shows position (x) on the x-axis and velocity (v) on the y-axis for different damping parameters (ζ). It includes a legend and grid, illustrating the phase space trajectories computed by the SciPy solver.

Now as a final step, we've calculated errors for various step sizes using both the explicit and implicit Euler methods, comparing them against the SciPy solver as our reference. For damping parameters ζ (0.25, 1, 2), we computed errors for each method and step size h. To gauge accuracy, we aligned the method's results with the reference solution's time points, then measured the discrepancies.

Next, we plotted "Error vs Step Size (log-log scale)." This graph portrays step size (h) on the x-axis and error on the y-axis, both logarithmically scaled. It features a legend and grid, revealing errors for both Euler methods across different damping parameters.

Problem 2:

For the second problem that consists of numerically solving the bounded value problem of the same equation with $\zeta = 0.25$ and $x(0) = 1$, $x(9) = 0$ across the time interval $t \in [0, 9]$.

Firstly we used the shooting method in order to find our initial velocity v_0 that respects the boundary conditions at t = 9. This involved solving an initial value problem and using root finding techniques. We then used the initial conditions in solving the differential equation, then we visualized the results in the plot titled "Solution of BVP using Shooting Method," which conveys how the x(t) evolves over an interval of time.

Next, we applied the finite difference method, which breaks down the differential equation into smaller time intervals. This approach required setting up and solving a system of linear equations using a tridiagonal matrix method. The resulting graph, labeled "Solution of BVP using Finite Difference Method," displays how x(t) changes over time.

Finally, we utilized SciPy's `solve_bvp` function, which is used in order to solve boundary value problems. This method automatically formulates and solves the differential equations along with the given boundary conditions, and then keeps iterating the estimates until finding the most accurate solutions. The plot titled "Solution of BVP using SciPy solve_bvp" shows how x(t) changes over time, which shows how effective the method is in handling boundary conditions.

Now that we have described in the section above the workings of the code, this part is about analyzing the results and how our experiment with different step values h and damping ratios ζ has affected the solutions and their behavior and that is using the graphs that we plotted using our python code.

Problem 1 :

For this part of the project we will start by analyzing the figure 1 below that represents the phase space trajectories for the explicit Euler method.

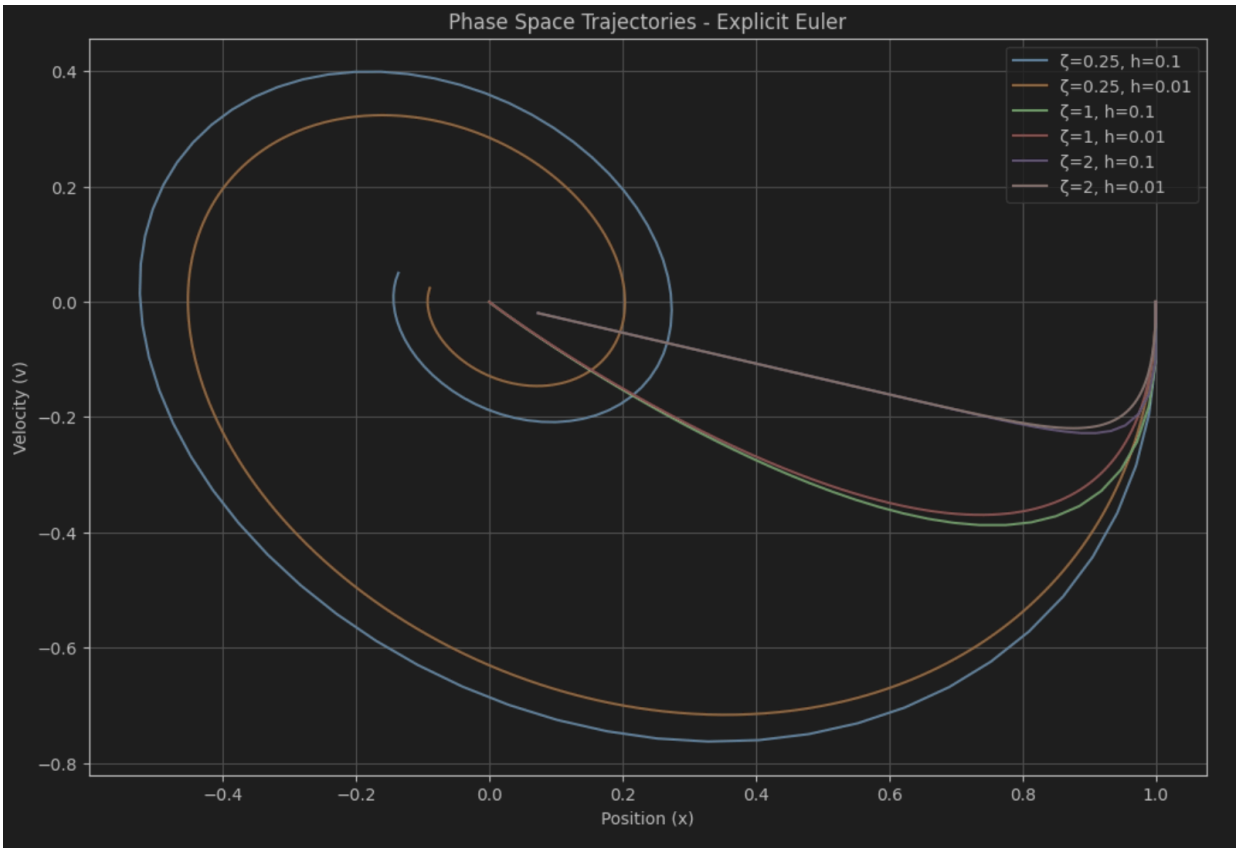

Figure 1.1: Phase Space Trajectories - Explicit method

As we can observe from the plot above, when ζ = 0.25 the oscillations are more clear and that with the large step size (h = 0.1) the trajectory extends further from the origin, indicating less accuracy compared to the smaller step size (h = 0.01). However when increasing the damping ratio to ζ = 1 and ζ = 2 the system reaches equilibrium quicker as seen above in the plot, and what explains this behavior is the damping force which is naturally strong the larger the damping ratio, and that causes the system's energy to dissipate resulting in a faster convergence of the trajectory towards the origin indicating that the system returns to a stable equilibrium state more efficiently.

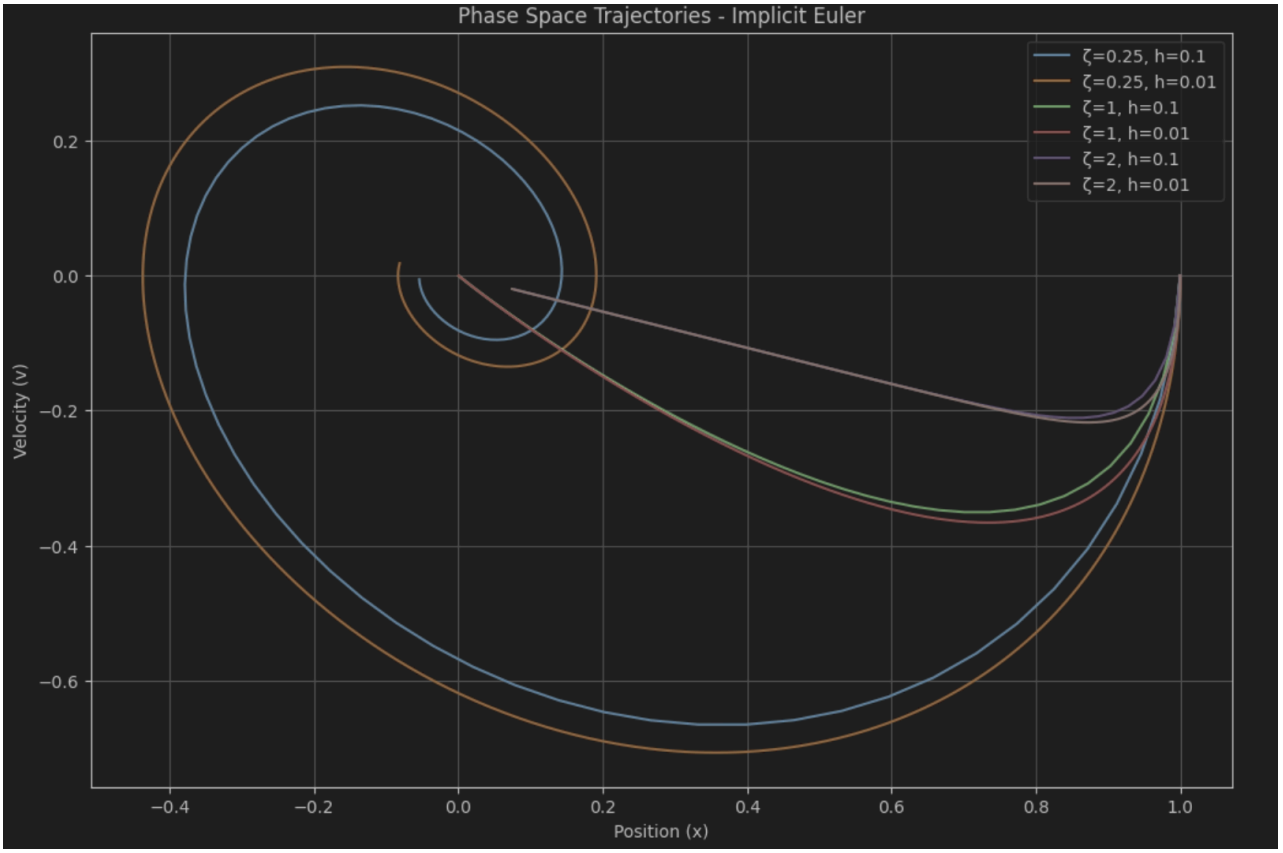

Figure 1.2: Phase Space Trajectories - Implicit method

Now in this plot above that represents the phase space trajectories on the implicit Euler method, we can tell that it is very similar to the one for the explicit methods for we have made the same changes in the damping ratios and step sizing and therefore resulting in the same variations.

Although the implicit Euler method shows similar trajectory variations when damping ratios and step sizes change, it typically provides more stability than the explicit method.

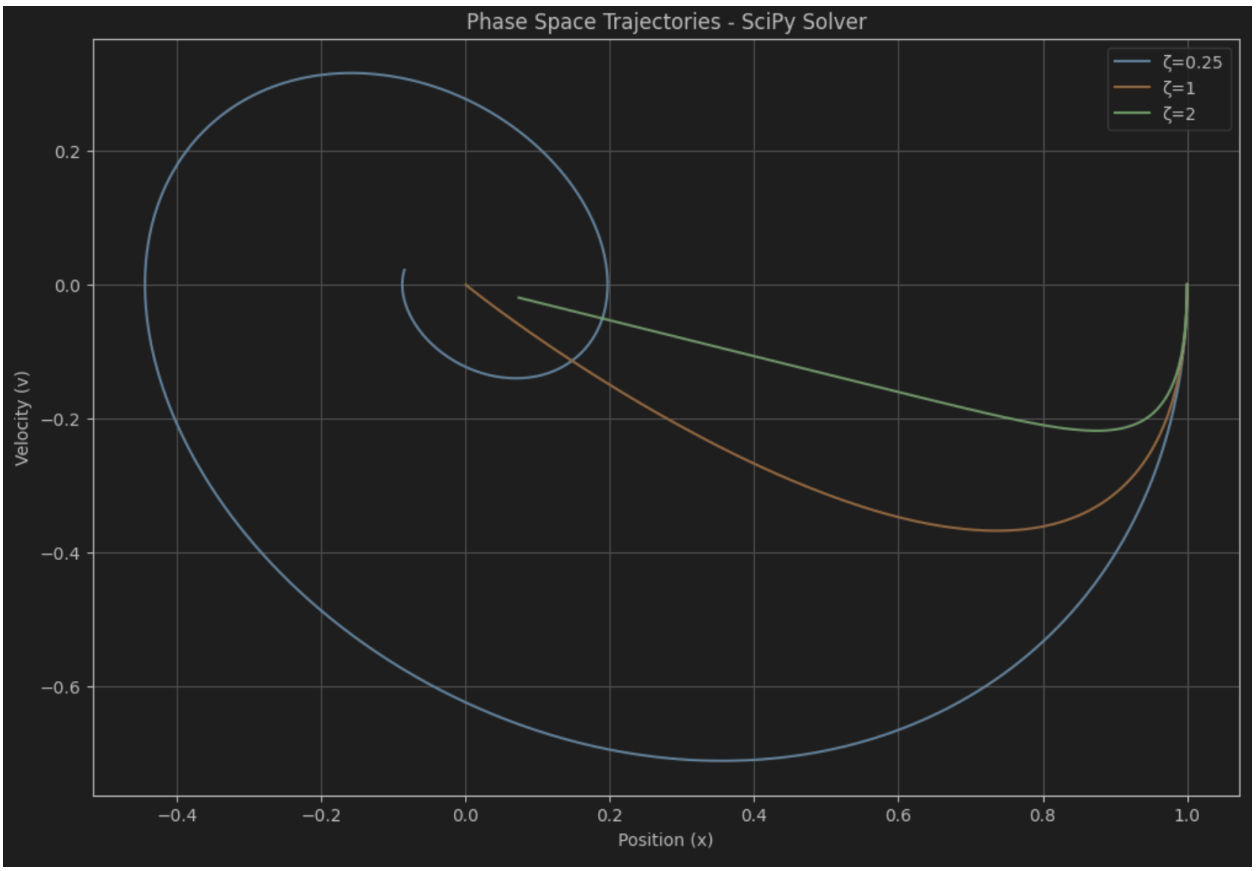

Figure 1.3: Phase Space Trajectories - SciPy Solver

This plot shows how the position and velocity of a damped oscillator evolve over time using the SciPy solver. Each line represents a trajectory for different damping ratios $\zeta = 0.25, 1, 2$. It visualizes how damping affects the oscillator's movement, with each curve illustrating how quickly or slowly the system reaches equilibrium.

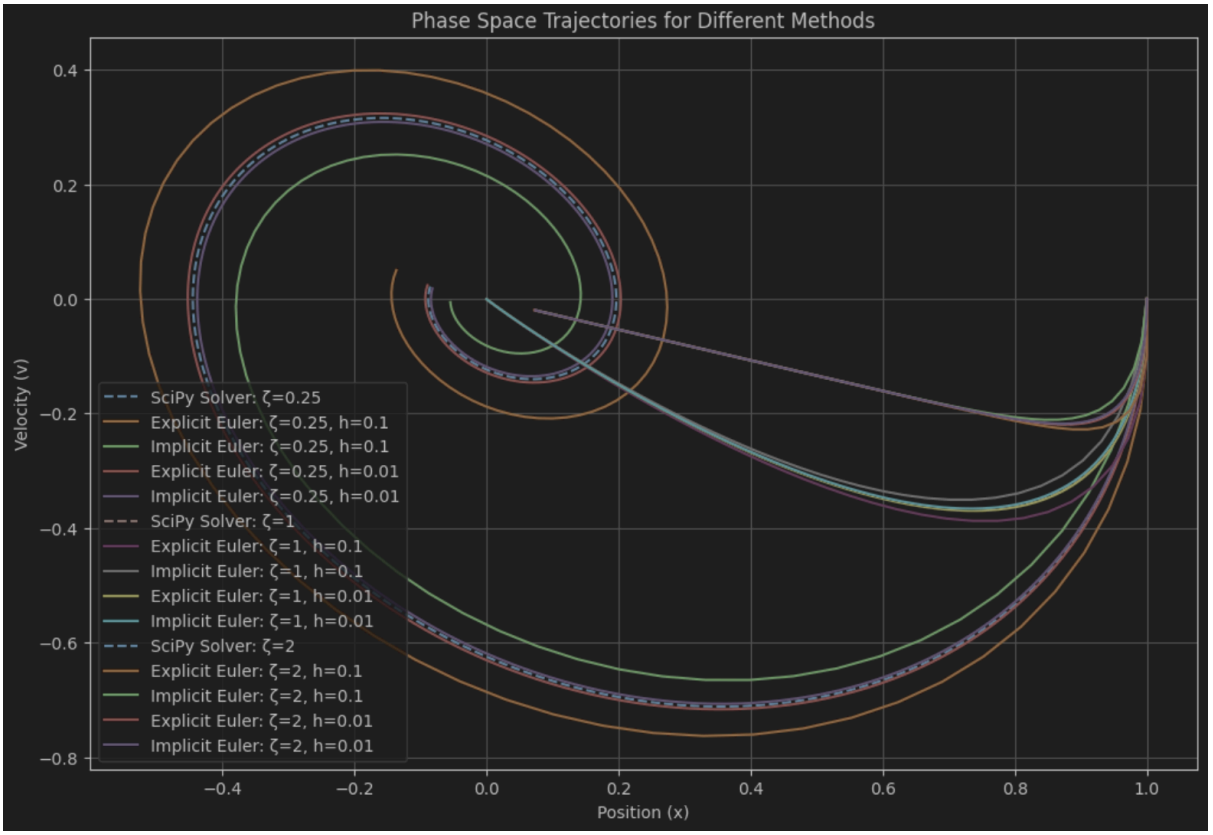

Figure 1.4: Phase Space Trajectories of the system

In this plot we get the bigger picture. Knowing that the SciPy solver trajectories are the most accurate we can how the two different Euler methods vary in accuracy and efficiency using the SciPy solver trajectories are references, and what we can see is that for both the implicit and explicit Euler method while not entirely accurate, when the damping ratio and stepping sizes we get the closest results to the ones of the SciPy solver and as seen in the plot the trajectories almost collide.

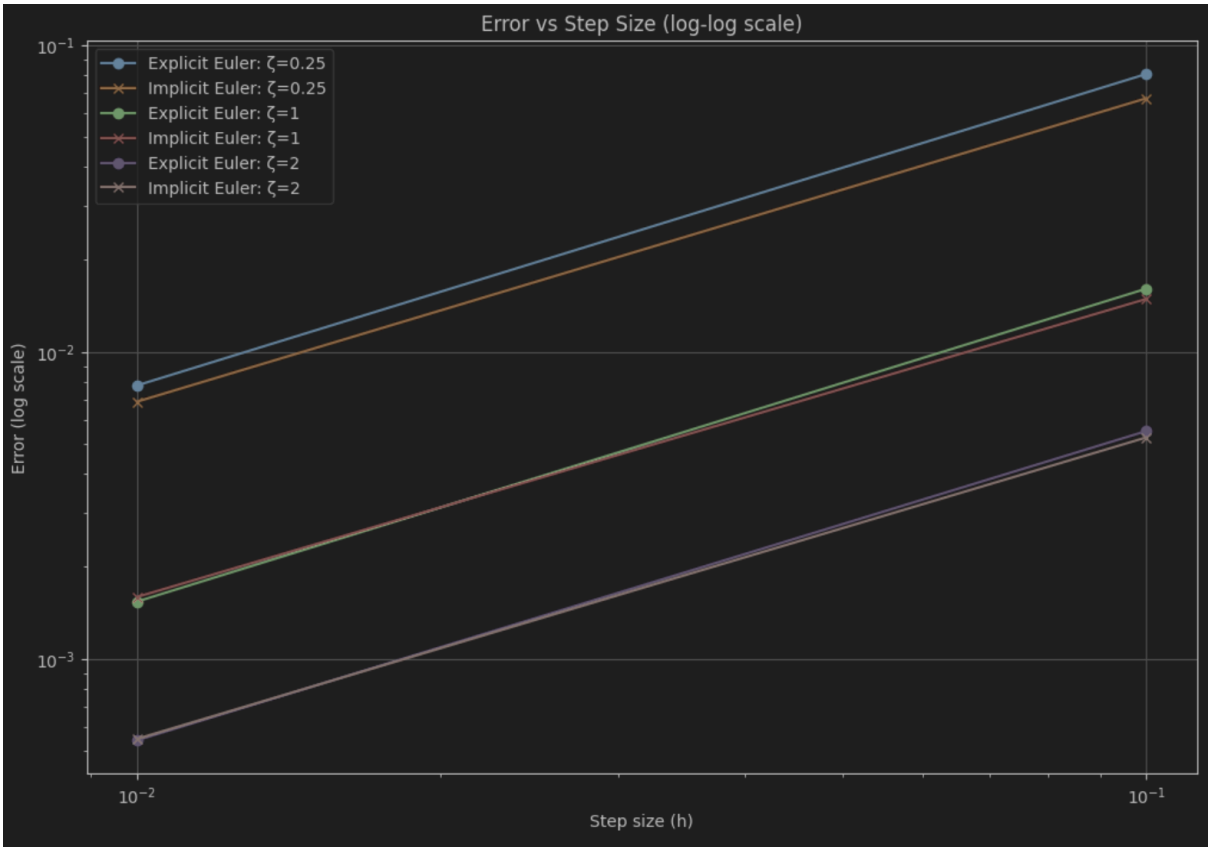

Figure 1.5: Error vs Step Size (log-log scale)

In this final plot we have a comparison of the Explicit Euler and Implicit Euler methods in solving the damped oscillator problem using a log-log scale to visualize errors. Each line represents how much the solutions differ from the reference obtained with the SciPy solver which we know is the most accurate. And as far as common sense goes the smaller the error the more accurate the method, and while analyzing this plot we can tell that for each damping ratio the error margin is greater for the explicit method.

Problem 2:

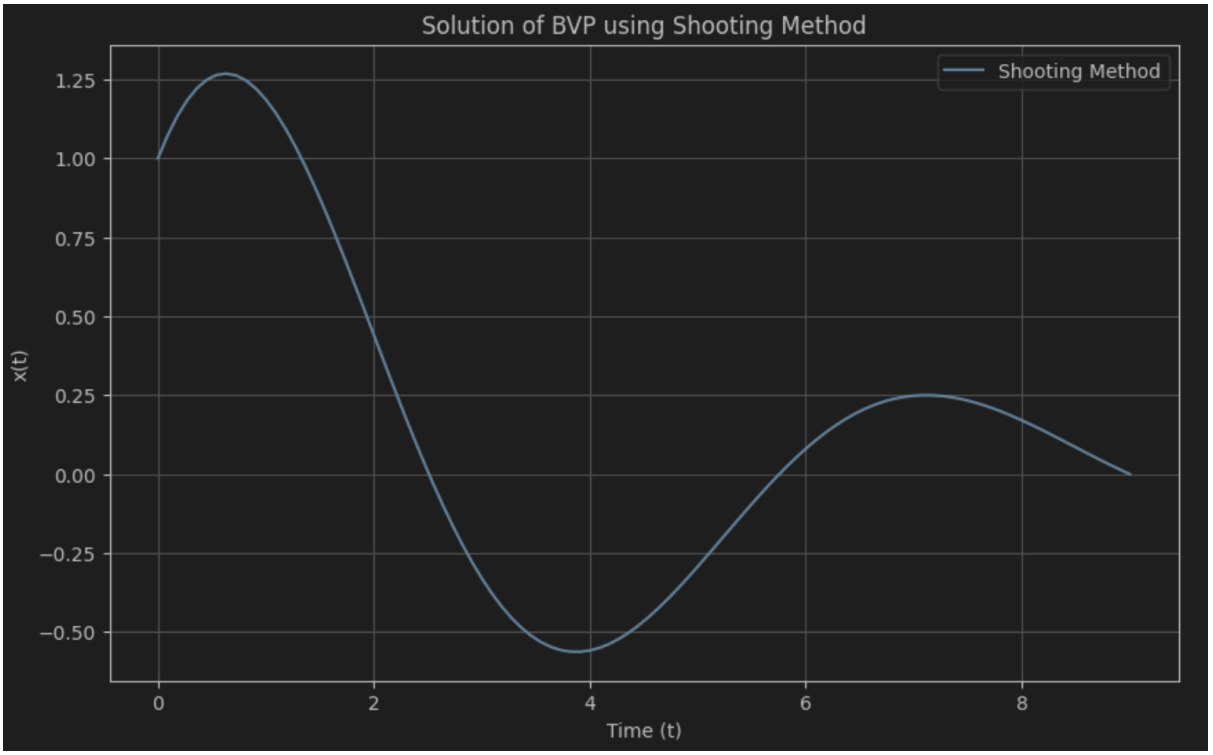

Figure 2.1: Error vs Step Size (log-log scale)

In Figure 2.1, the shooting method gives us a solution where the oscillations are smoother and less frequent compared to the other methods. This method focuses on finding an initial condition that meets the boundary conditions at t = 9. The x(t) curve shows a gradual decrease toward zero without sharp, rapid oscillations, indicating a smooth approach to equilibrium over time.

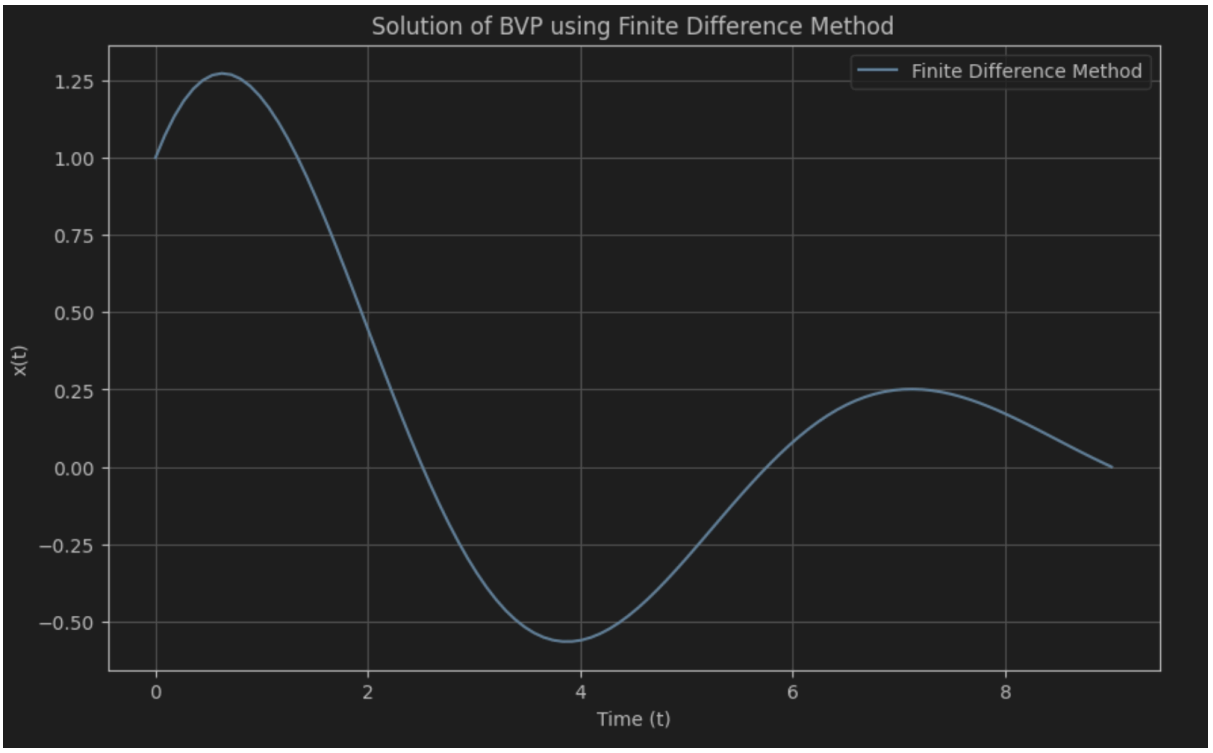

Figure 2.2: Error vs Step Size (log-log scale)

Figure 2.2 shows the result of the finite difference method, where we observe more pronounced oscillations in x(t) compared to the shooting method. Here, the differential equation is transformed into a set of linear equations, capturing the dynamics of the oscillator more accurately. This method reveals faster oscillations around the equilibrium position, reflecting more detailed aspects of the oscillator's response.

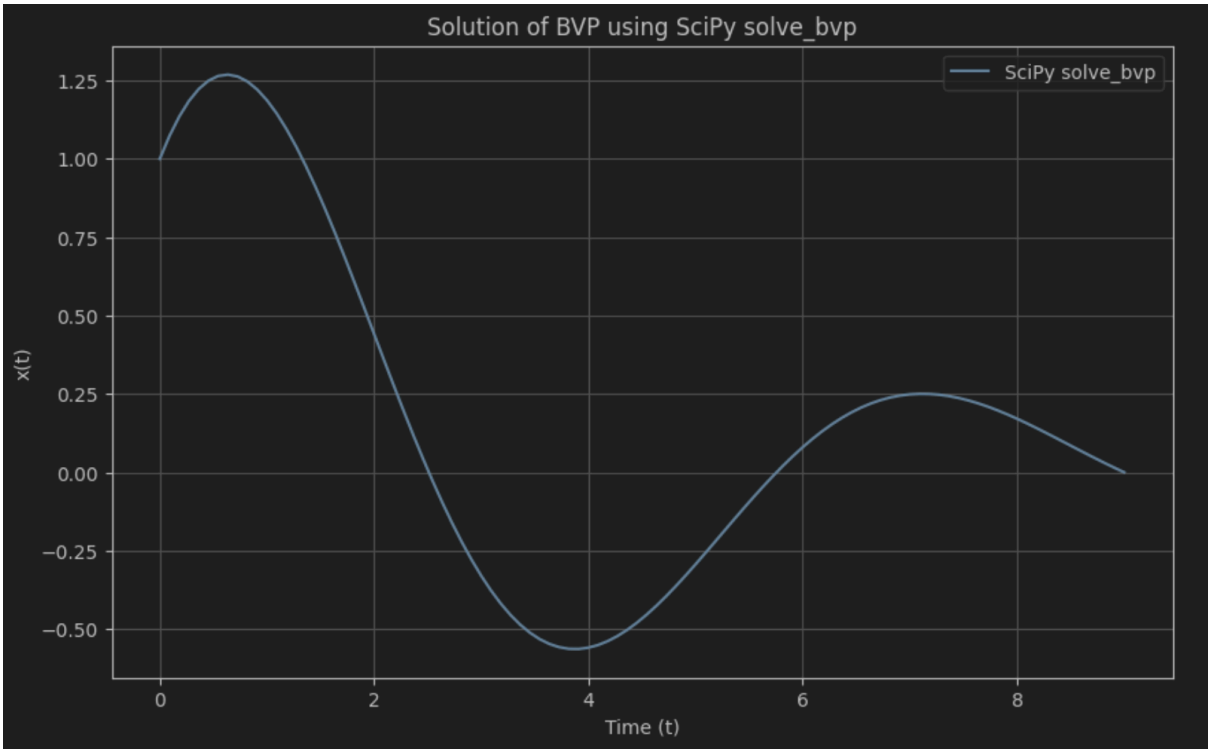

Figure 2.3: Error vs Step Size (log-log scale)

Finally, Figure 2.3 presents the solution obtained using the SciPy, which shows the highest frequency oscillations among the methods. This approach employs advanced numerical techniques to directly solve the boundary value problem, refining an initial guess iteratively. Thex(t) curve exhibits rapid oscillations and sharp transitions, closely representing the true oscillatory behavior of the damped oscillator.

DISCUSSION AND CONCLUSION

After the thorough analysis we have done in the previous section we can confidently say the closest method to accuracy when it comes to initial conditions problems is the implicit Euler method, considering it has the closest results and trajectories to the ones in the SciPy solver and also with the smallest error margin, that is compared to explicit Euler method. We can also deduct that the more accurate results occur when we use a small damping ratio and small stepping sizes.

As we move from simpler methods like the shooting method to more advanced ones like the finite difference method and SciPy solve_bvp, we notice a clear improvement in how accurately the oscillator's behavior is represented. The oscillations become more frequent in the figures, showing that these methods can capture finer details of how the system behaves over time. Each method helps us understand better how the damped oscillator reacts to damping and initial conditions, with SciPy solve_bvp offering the most precise and detailed portrayal of its oscillatory motion.