Assignment 0

Solutions of Problems with Discretization and Sampling

Sahil Raj
2301CS41

02 August 2025

Contents

Problem 1: Monte Carlo Simulation of π

Problem Statement

The objective of this problem is to demonstrate the convergence of the estimated value of π using the Monte Carlo simulation method, with an increasing number of iterations.

NOTE: The code can be accessed using these links: MATLAB, JULIA

Methodology

The Monte Carlo method relies on generating random points inside a unit square $[0,1] \times [0,1]$ and determining the proportion of points that fall inside the quarter unit circle defined by $x^2 + y^2 \leq 1$. Since the area of this quarter circle is $\pi/4$, the ratio can be used to estimate π.

Pseudo-code

```
1. Initialize counters for points inside and outside the circle.
2. For each iteration:
    (a) Sample random $x, y \in [0,1]$.
    (b) Compute distance $d = \sqrt{x^2 + y^2}$.
    (c) If $d \leq 1$, classify point as inside; otherwise classify as outside.
3. Estimate $\pi$ as:
        $\pi \approx 4 \times \frac{\text{number of points inside}}{\text{total number of points}}$
4. Repeat with increasing iterations (e.g., $64, 640, 6400, 64000$).
5. Record and plot the results.
```

Results

The following figures illustrate the distribution of points for different iteration counts, along with the corresponding approximation of π. As the number of iterations increases, the estimate converges towards the true value of π.

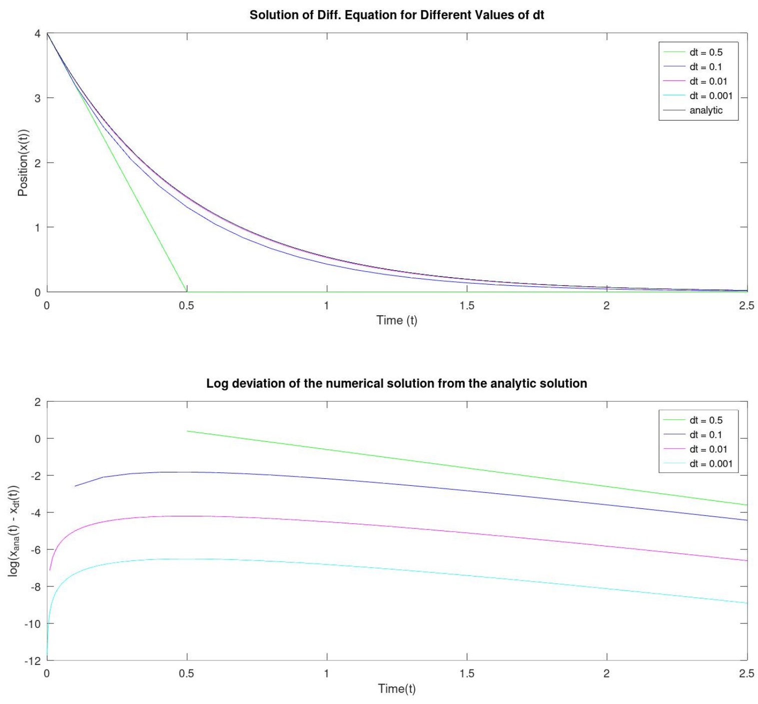


Figure 1.1: Monte Carlo simulation of π with increasing iterations.

Conclusion

The Monte Carlo simulation provides a stochastic method for estimating π. With a small number of iterations, the estimate is inaccurate, but as the iteration count increases, the estimate converges to the true value of π.

Problem 2: Numerical Solution of a First-Order ODE

Problem Statement

The task is to compute the numerical solution of the first-order ordinary differential equation

$$\frac{dx}{dt} = -kx, \quad k > 0,$$

with an initial condition $x(0) = x_0$, and compare it with the analytical solution.

NOTE: The code can be accessed using these links: MATLAB, JULIA

Methodology

The differential equation models exponential decay. Its exact solution is:

$$x(t) = x_0 e^{-kt}.$$

To approximate the solution numerically, we discretize the time interval $[0, t_{\max}]$ with step size Δt. Using the forward Euler method, the recurrence relation is:

$$x_n = x_{n-1}(1 - k\Delta t).$$

This process is repeated for multiple values of Δt to study convergence.

Pseudo-code

```
1. Define parameters: initial value $x_0$, coefficient $k$, maximum time $t_{\max}$.
2. Choose a time step $\Delta t$ and generate a time vector.
3. Initialize $x(0) = x_0$.
4. For each step $n$:
    (a) Update solution using $x_n = x_{n-1}(1 - k\Delta t)$.
5. Repeat for decreasing values of $\Delta t$.
6. Compare the numerical solution with the analytical solution $x(t) = x_0 e^{-kt}$.
7. Compute and plot the error $\log(x_{\text{ana}}(t) - x_{\Delta t}(t))$.
```

Results

The following figures illustrate:

- Numerical solutions for different time step sizes compared against the analytical solution.
- The logarithmic deviation of numerical solutions from the analytical solution.

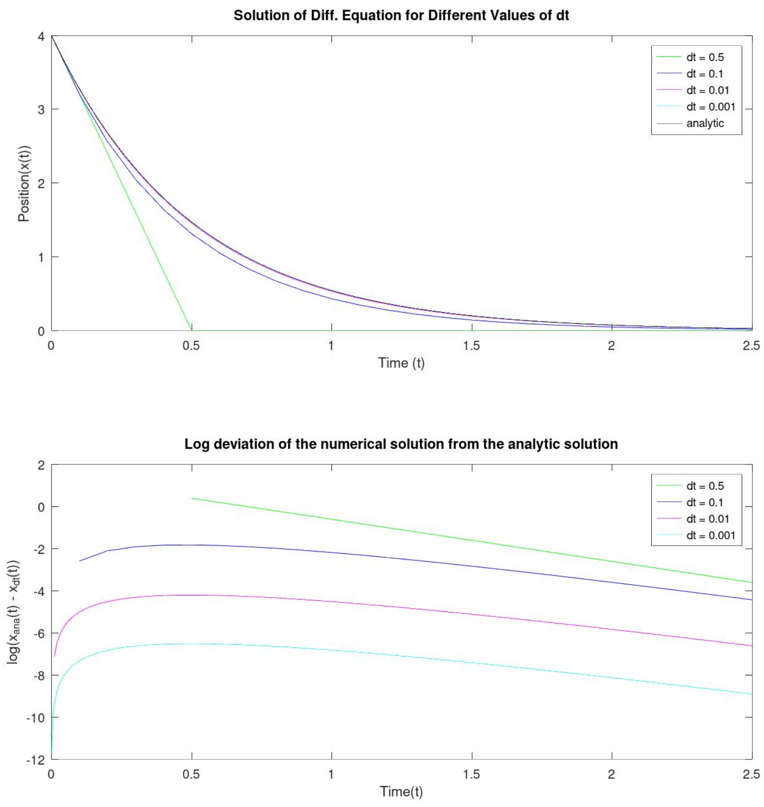


Figure 2.1: Numerical solutions for different Δt compared with the analytical solution.

Conclusion

The Euler method successfully approximates the solution of the differential equation. However, the accuracy depends strongly on the time step Δt. Larger Δt values produce significant deviation, while smaller Δt values converge towards the exact solution. The error plots clearly demonstrate the improvement in accuracy with smaller step sizes.

Problem 3: Numerical Solution of a Second-Order ODE

Problem Statement

The problem is to compute the numerical solution of the second-order ordinary differential equation

$$m\frac{d^2x}{dt^2} = -kx(t),$$

where $k > 0$ and $m > 0$, with given initial conditions $x(0) = x_0$, $x'(0) = v_0$. The numerical solution is compared against the exact analytical solution.

NOTE: The code can be accessed using these links: MATLAB, JULIA

Methodology

The given equation represents the equation of motion of a simple harmonic oscillator. The analytical solution is

$$x(t) = A\sin(\omega t + \phi),$$

where $\omega = \sqrt{\frac{k}{m}}$, and A, ϕ are determined from the initial conditions.

To compute the numerical solution, the central difference method is used:

$$x_n = (2 - \tfrac{k\Delta t^2}{m})x_{n-1} - x_{n-2},$$

with $x(0) = x_0$ and $x(\Delta t) = x_0 + v_0\Delta t$.

Pseudo-code

1. Define parameters: $m, k, x_0, v_0, t_{\max}$.
2. Initialize solution array:

$$x(0) = x_0,$$
$$x(\Delta t) = x_0 + v_0\Delta t.$$

3. For $n = 2, 3, \ldots, N$:

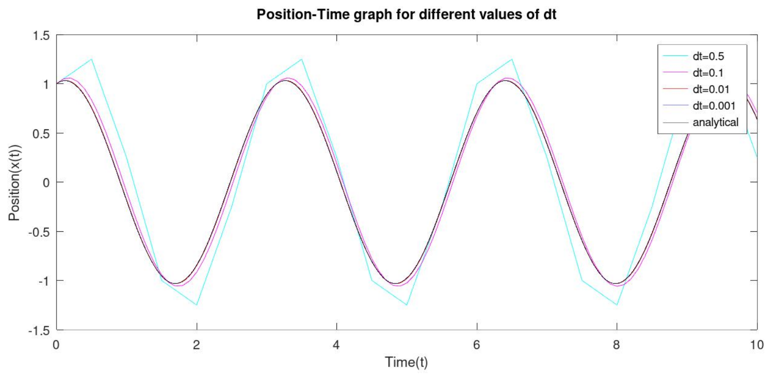
$$x_n = \left(2 - \frac{k\Delta t^2}{m}\right)x_{n-1} - x_{n-2}.$$

4. Repeat for different values of Δt.
5. Compare numerical solutions with analytical solution
$$x(t) = A\sin(\omega t + \phi).$$
6. Plot solution curves and error curves.

Results

The following figures illustrate:

- The computed solutions for different step sizes Δt along with the analytical solution.
- The logarithmic absolute deviation of the numerical solutions from the analytical solution.



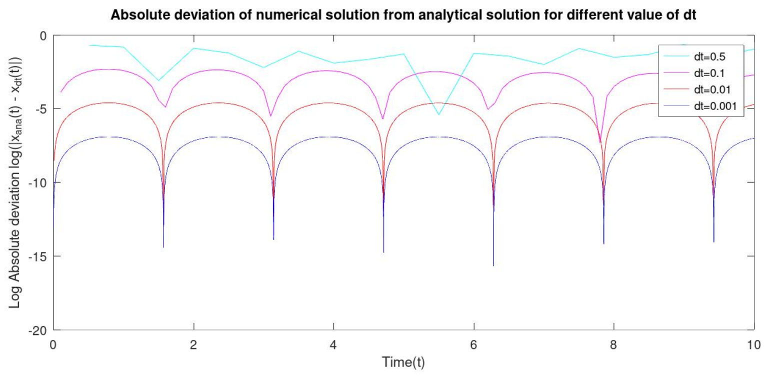


Figure 3.1: Position-time graph for different Δt compared with the analytical solution.

Conclusion

The central difference method provides a stable and accurate approximation of the harmonic oscillator when the time step Δt is sufficiently small. Large values of Δt lead to noticeable deviations, while smaller steps converge towards the analytical solution. The error plots clearly demonstrate the dependence of accuracy on the step size.

Problem 4: Numerical Solution of a Non-linear Second-Order ODE

Problem Statement

We aim to compute the numerical solution of the non-linear second-order ordinary differential equation

$$m\frac{d^2x}{dt^2} = -kx - lx^3 - b\frac{dx}{dt} + F_0\cos(\omega t),$$

where $k, l, b > 0$ and F_0, ω are constants. This equation describes a driven, damped, non-linear oscillator.

NOTE: The code can be accessed using these links: MATLAB, JULIA

Methodology

The given equation cannot be solved in closed form due to the cubic nonlinearity. A numerical scheme based on finite differences is applied to approximate the solution.

The update formula for the position is derived by discretizing time with step size Δt, leading to:

$$x_{n+1} = c_1x_n + c_2x_n^3 + c_3x_{n-1} + c_4\cos(\omega t_n),$$

where the coefficients are functions of the parameters m, k, l, b, F_0, and Δt.

Pseudo-code

```
1. Initialize parameters: $m, k, l, b, F_0, \omega, \Delta t, t_{\max}$.
2. Initialize solution:
       $x(0) = x_0, \quad x(\Delta t) = x_0 + v_0\Delta t.$
3. Compute coefficients $c, c_1, c_2, c_3, c_4$ from system parameters.
4. For $n = 2, 3, \ldots, N$:
       $x_{n+1} = c_1x_n + c_2x_n^3 + c_3x_{n-1} + c_4\cos(\omega t_n).$
5. Plot $x(t)$ versus $t$.
```

Results

The numerical simulation yields a time-domain trajectory of the oscillator. The solution exhibits oscillatory behavior with nonlinear distortion due to the cubic restoring force, damping, and external driving force.

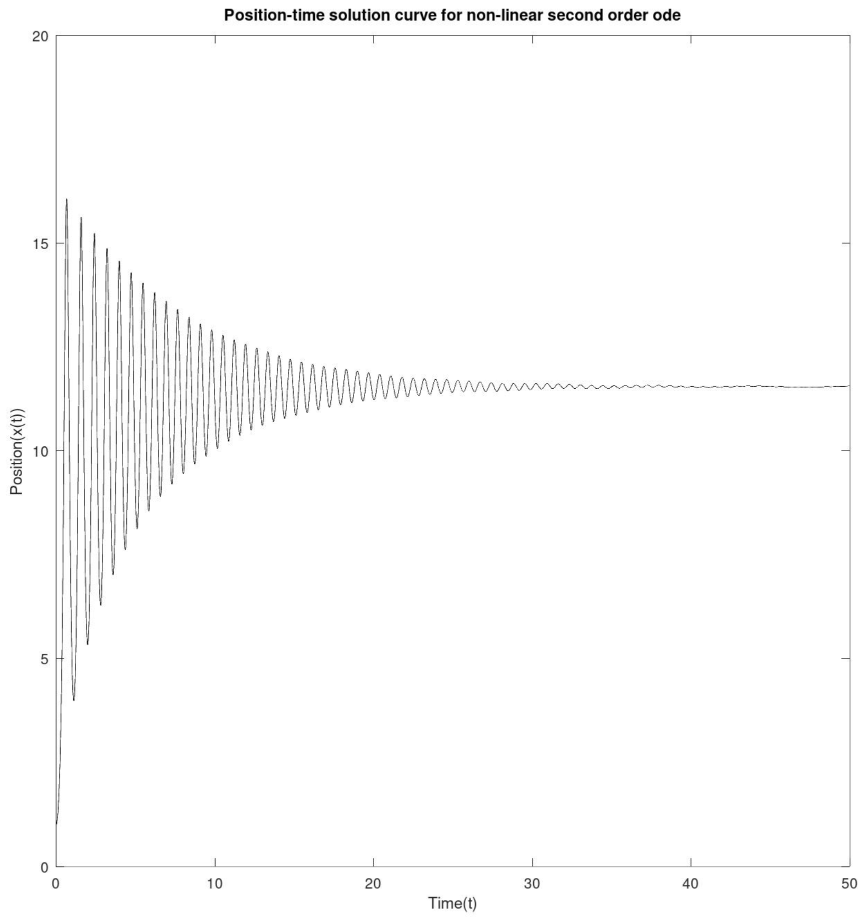


Figure 4.1: Position-time solution curve for the non-linear second-order ODE.

Conclusion

The non-linear oscillator demonstrates rich dynamics that cannot be captured analytically. Numerical integration allows us to visualize the system's trajectory and analyze the influence of non-linearity, damping, and external driving. The results highlight the importance of computational methods in studying complex dynamical systems.