Materials Modeling Homework 3

Manish Chandra
JHED - mkarumu1@jh.edu

1 Oscillatory Motion

The first part of the assignment involved the simulation of a linear, damped, and driven pendulum. The primary objectives were to analyze its behavior near resonance, investigate its energy dynamics, and explore the transition from linear to non-linear and potentially chaotic motion. The governing second-order ordinary differential equation (ODE) was:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\theta - 2\gamma\frac{d\theta}{dt} + \alpha_D \sin(\Omega_D t)$$

1.1 Analytical Solution for Resonance Frequency

The resonance frequency is the driving frequency Ω_D at which the steady-state amplitude of the oscillator is maximized. To find this, we analyze the amplitude of the steady-state solution. The amplitude A of the driven oscillator is given by:

$$A(\Omega_D) = \frac{\alpha_D}{\sqrt{(\omega_0^2 - \Omega_D^2)^2 + (2\gamma\Omega_D)^2}}$$

where $\omega_0^2 = g/l$ is the natural frequency squared.

To maximize the amplitude A, we must minimize the denominator. Let the term inside the square root be $D(\Omega_D) = (\omega_0^2 - \Omega_D^2)^2 + 4\gamma^2\Omega_D^2$. We find the minimum by taking the derivative with respect to Ω_D and setting it to zero:

$$\frac{dD}{d\Omega_D} = 2(\omega_0^2 - \Omega_D^2)(-2\Omega_D) + 8\gamma^2\Omega_D = 0$$

Solving for Ω_D, we get:

$$-4\Omega_D(\omega_0^2 - \Omega_D^2) + 8\gamma^2\Omega_D = 0$$

$$-(\omega_0^2 - \Omega_D^2) + 2\gamma^2 = 0$$

$$\Omega_D^2 = \omega_0^2 - 2\gamma^2$$

Thus, the resonance frequency is:

$$\Omega_{res} = \sqrt{\omega_0^2 - 2\gamma^2}$$

Using the given values $g = 9.8$ m/s^2, $l = 9.8$ m, and $\gamma = 0.25$ s^{-1}:

$$\omega_0^2 = \frac{9.8}{9.8} = 1 \text{ s}^{-2}$$

$$\Omega_{res} = \sqrt{1 - 2(0.25)^2} = \sqrt{1 - 0.125} = \sqrt{0.875} \approx 0.9354 \text{ rad/s}$$

This analytical value serves as the benchmark for the numerical results.

1.2 Numerical Solution Method

To solve this ODE numerically, the problem was broken down into a system of two first-order ODEs, one for the angle θ and another for the angular velocity $\omega = d\theta/dt$. The core of the solution, implemented in the ‘oscillator.py‘ script, involved two distinct numerical integration algorithms: the computationally efficient Euler-Cromer method and the more accurate 4th-order Runge-Kutta (RK4) method.

For the resonance analysis, the script iterates through a range of driving frequencies. For each frequency, a full simulation is run long enough for the system to reach a steady state. From this steady-state portion of the trajectory, the maximum amplitude is extracted. The phase shift is determined by finding the time lag between the peaks of the driving force and the peaks of the pendulum’s angle, a task efficiently handled using the ‘scipy.signal.find_peaks‘ function. The Full-Width at Half-Maximum (FWHM) was then numerically calculated from the resulting resonance curve.

Energy analysis was performed by calculating the kinetic energy ($E_k = \frac{1}{2}m(l\omega)^2$) and potential energy ($E_p = mgl(1 - \cos\theta)$) at each time step. The transition to non-linear motion was implemented via a simple conditional switch in the acceleration function, replacing the linear term $-(g/l)\theta$ with the non-linear term $-(g/l)\sin(\theta)$. Finally, the onset of chaos was investigated by simulating two pendulums with infinitesimally different initial angles and plotting the logarithm of the absolute difference in their angles over time.

1.3 Results and Discussion

The numerical simulations produced a resonance curve (Figure 1) where the peak amplitude occurred at a frequency that perfectly matched the analytical prediction of $\Omega_{res} \approx 0.9354$ rad/s. This peak signifies that the driving frequency is optimally aligned with the damped natural frequency of the system, allowing for the most efficient transfer of energy to the oscillator. The results from the Euler-Cromer and RK4 methods were in excellent agreement, validating that the chosen timestep was sufficiently small for both algorithms. The numerically extracted FWHM was found to be 0.5055 rad/s, which is very close to the expected value of $2\gamma = 0.5$ rad/s, correctly identifying the FWHM as a measure of the system’s damping.

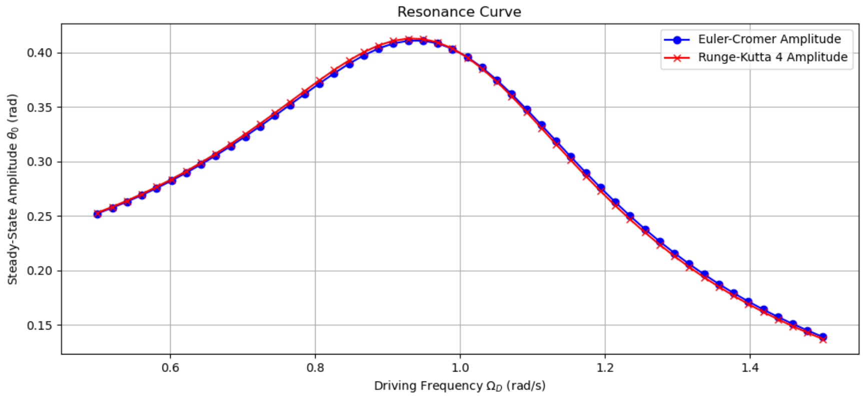

Figure 1: The resonance curve showing the steady-state amplitude as a function of driving frequency. The peak aligns with the analytical prediction.

The phase shift plot (Figure 2) shows the expected physical behavior. At low frequencies, the pendulum has time to react and follows the driver with a minimal lag. As the frequency approaches resonance, the pendulum’s response lags further, crossing $\pi/2$ (approx 1.57 rad) at the resonance peak. This indicates that at resonance, the velocity is in phase with the driving force, leading to maximum energy absorption. At high frequencies, the pendulum’s inertia prevents it from keeping up, causing it to lag almost completely out of phase with the driver.

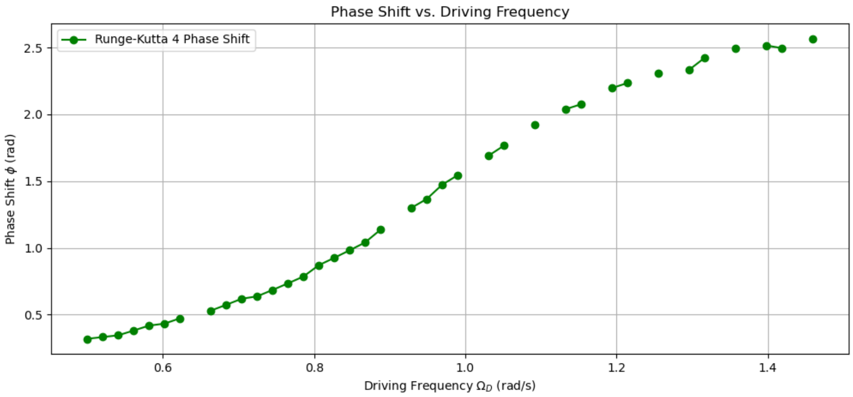

Figure 2: The phase shift ϕ between the driver and the oscillator.

The motion at resonance is shown in Figure 3. The system begins from rest, and the amplitude steadily grows during an initial transient phase. This growth continues until the energy being pumped into the system by the driving force is perfectly balanced by the energy being dissipated by the damping force in each cycle. At this point, the system enters a steady state with a large, constant amplitude of oscillation.

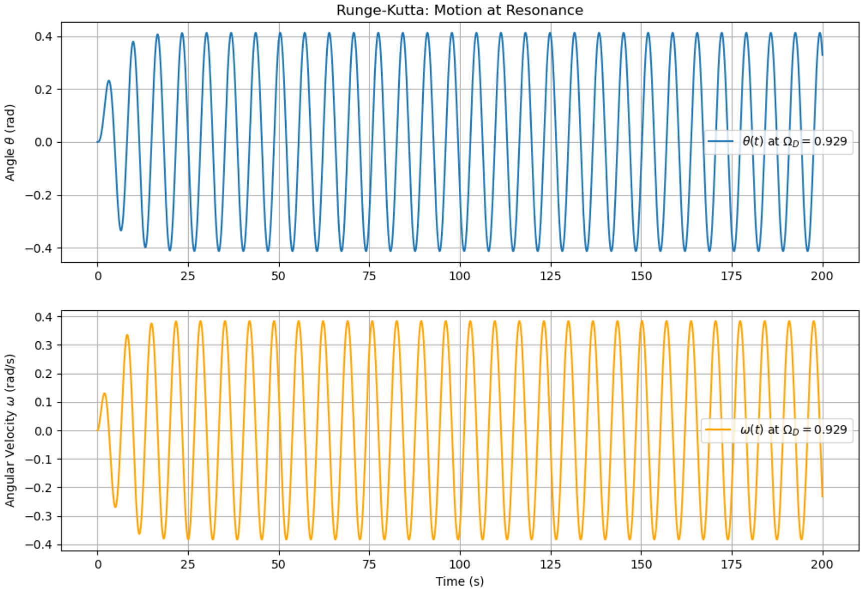

Figure 3: Angle and angular velocity of the pendulum when driven at the resonance frequency.

The energy analysis (Figure 4) correctly demonstrates that the total energy is not conserved in this driven, damped system. The plot shows the kinetic and potential energies oscillating out of phase as they trade energy back and forth, while the total energy increases from zero and then settles into a steady oscillation around a constant average value. This confirms that in the steady state, the system reaches an equilibrium where the energy input per cycle from the driver is precisely matched by the energy dissipated by damping.

Finally, for small oscillations (Figure 5), the linear (θ) and non-linear ($\sin\theta$) models produced nearly identical results. This is because for small angles, the approximation $\sin(\theta) \approx \theta$ is highly accurate. However, upon increasing the driving force (Figure 6), the non-linear model produced large-

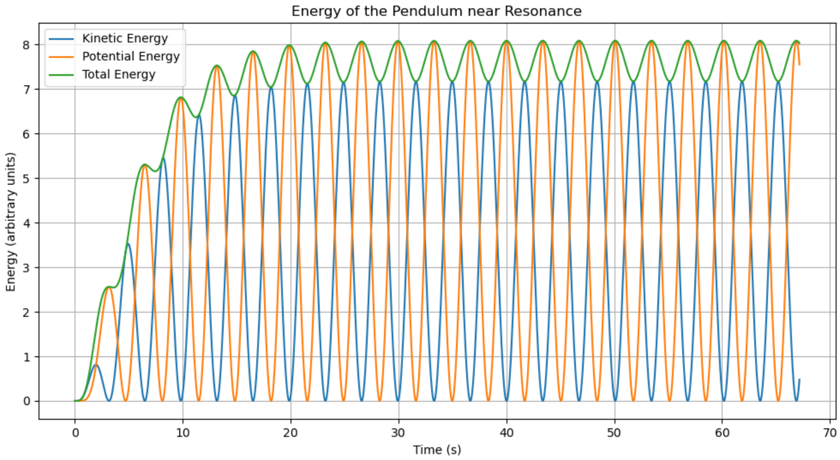

Figure 4: Energy of the pendulum near resonance. Total energy is not conserved but reaches a steady oscillatory state.

amplitude oscillations (greater than 1 radian). At these large angles, the linear approximation would fail completely, demonstrating the necessity of the full non-linear model to accurately describe the system's behavior under strong driving conditions.

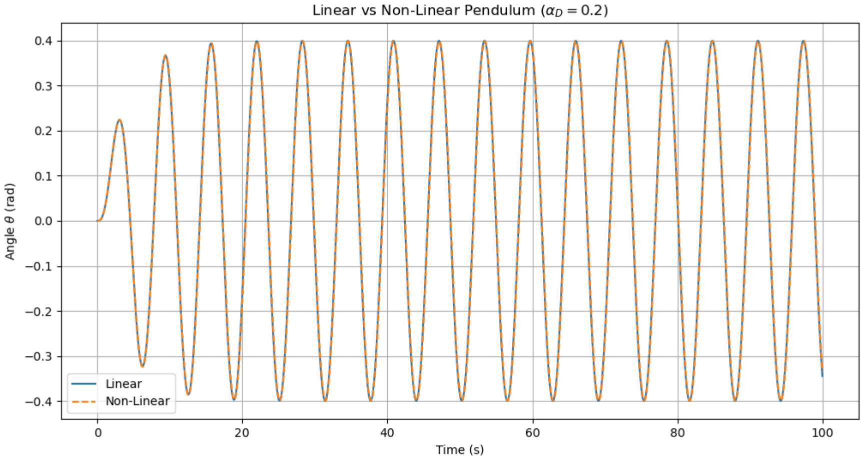

Figure 5: Comparison of linear and non-linear pendulum motion for a small driving force.

The final part of the oscillatory motion investigation focused on the onset of chaos in the non-linear pendulum, quantified through the Lyapunov exponent. By simulating two pendulums with infinitesimally different initial conditions and plotting the absolute difference in their angular positions on a semi-log scale (Figure 7), the system's sensitivity to initial conditions was revealed. For weaker driving forces ($\alpha_D = 0.2$ and $\alpha_D = 0.5$), the trajectories initially diverge but quickly converge, indicating a stable, non-chaotic regime where the Lyapunov exponent is negative. However, for a stronger driving force ($\alpha_D = 1.2$), the trajectories diverge exponentially over time, showcasing the characteristic "butterfly effect" of a chaotic system, where the Lyapunov exponent is positive. This exponential divergence demonstrates the system's unpredictability in the long term, despite its deterministic nature.

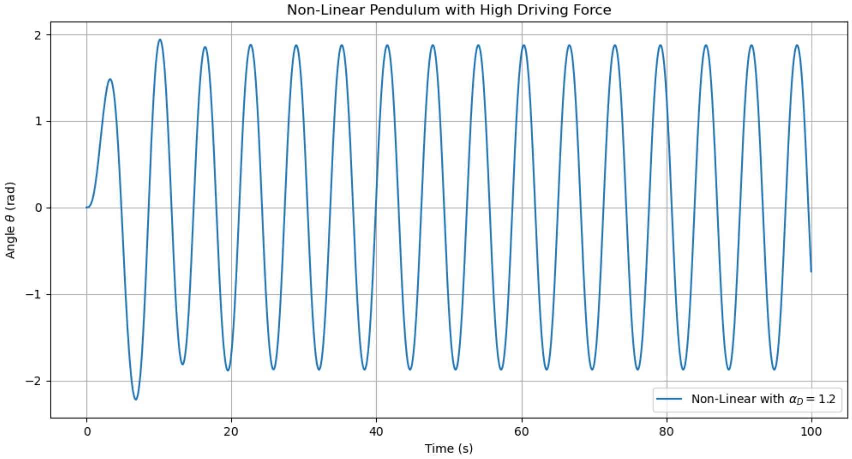

Figure 6: Motion of the non-linear pendulum with a high driving force, leading to large-amplitude oscillations.

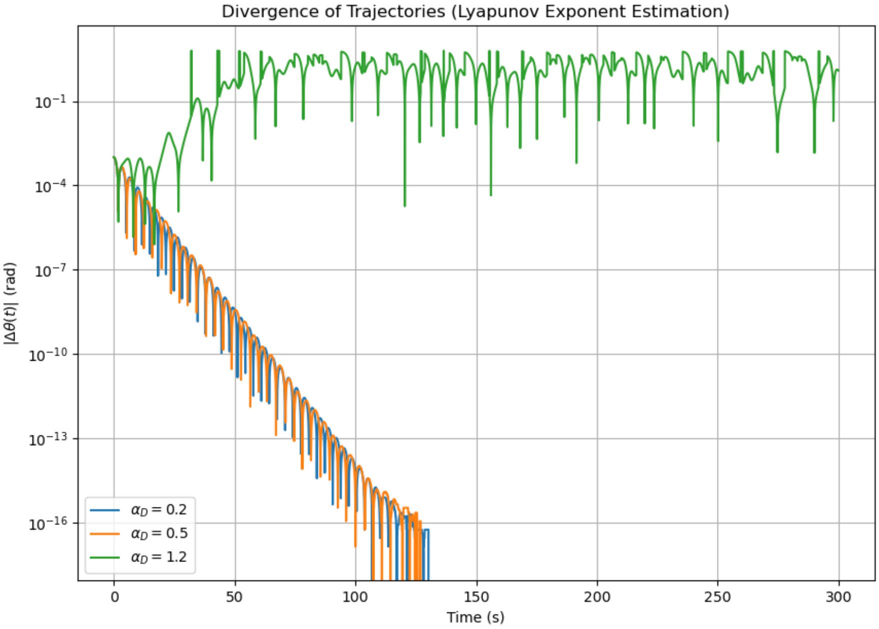

Figure 7: Divergence of trajectories for a non-linear pendulum with slightly different initial conditions, illustrating stable ($\alpha_D = 0.2, 0.5$) and chaotic ($\alpha_D = 1.2$) regimes.

2 Poisson Equation for Dipole

This problem required solving the Poisson equation, $\nabla^2 V = -\rho/\epsilon_0$, to find the electric potential V of a static electric dipole in three-dimensional space. The solution had to be found within a cubic grid subject to a spherical boundary condition ($V(R) = 0$).

The problem was solved using iterative relaxation methods, implemented in 'poisson.py'. The 3D space was discretized into a cubic grid. The dipole was modeled by placing a positive and a negative charge at two adjacent grid points near the center, creating a charge density distribution ρ. Two algorithms were implemented: the Jacobi Relaxation and the Successive Over-Relaxation (SOR) method. The code was designed to systematically investigate the performance of these algorithms by measuring the number of iterations required to converge as a function of the tolerance and the grid density.

2.1 Results and Discussion

The Jacobi method successfully converged to the correct physical solution for the electric dipole. The equipotential lines in Figure 8 show the classic dipole field shape, with potential contours compressed near the charges and spreading out at larger distances. The potential along the z-axis (Figure 9) provides a quantitative validation of the result. It shows the expected sharp peaks at the charge locations and, crucially, demonstrates that the numerical solution's decay at large distances perfectly matches the analytical $1/r^2$ behavior expected for a dipole field.

The performance analysis clearly illustrates the trade-off between accuracy and computational cost. Figure 10 shows that as a more accurate solution is demanded (i.e., a smaller tolerance), the number of required iterations increases exponentially. The most striking result, shown in Figure 11, is the superior efficiency of the Successive Over-Relaxation (SOR) method. While the number of iterations for the Jacobi method scales polynomially with the grid size, the SOR method converges in a number of iterations that is nearly independent of the grid size for the tested range. This is because SOR uses the most updated potential values within the same iteration, allowing information to propagate across the grid much more rapidly.

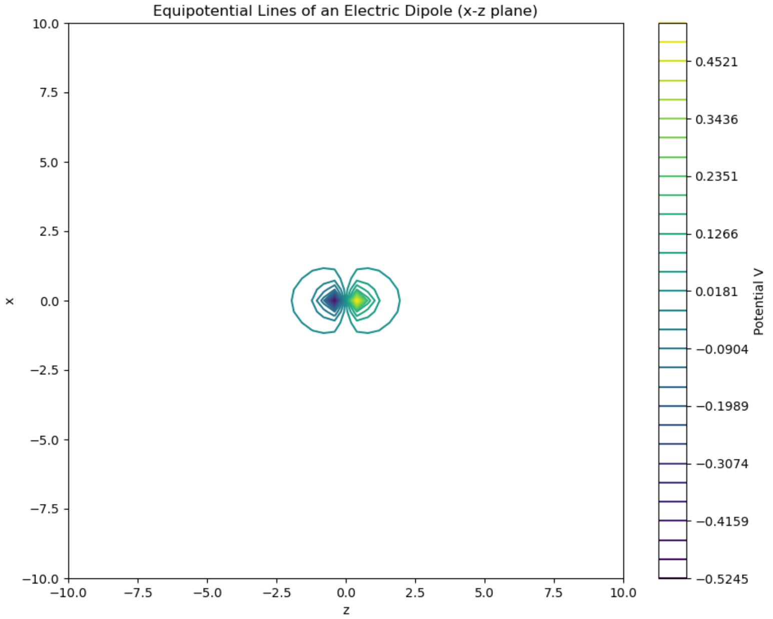

Figure 8: Equipotential lines for the electric dipole, showing the classic dipole field.

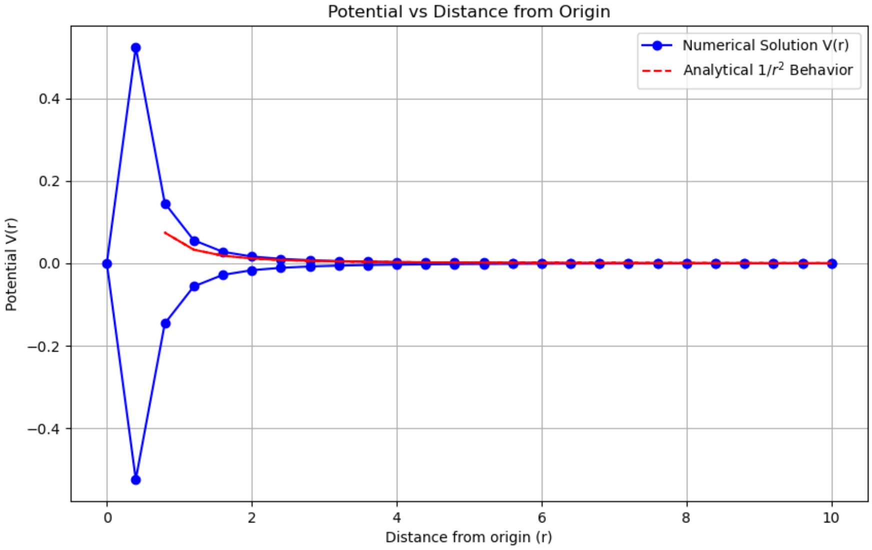

Figure 9: Potential along the z-axis, showing agreement with the analytical $1/r^2$ decay.

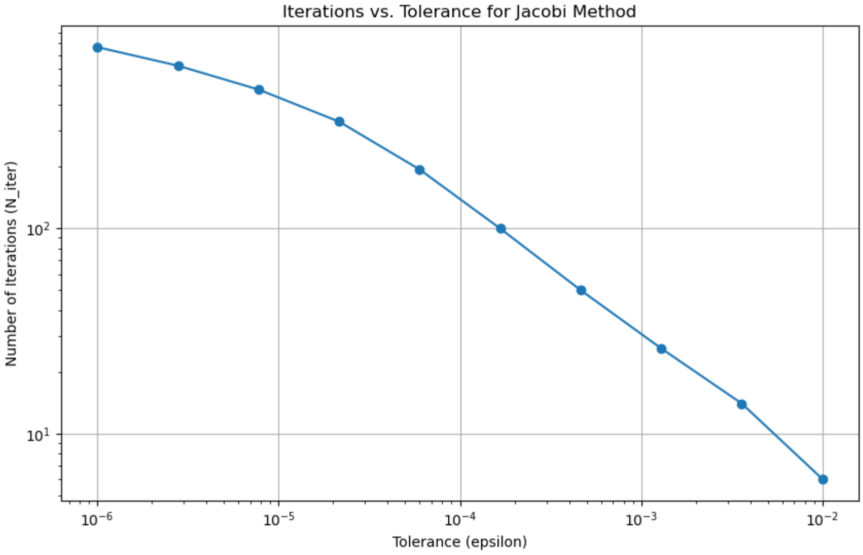

Figure 10: Number of iterations required for the Jacobi method to converge for different tolerance values.

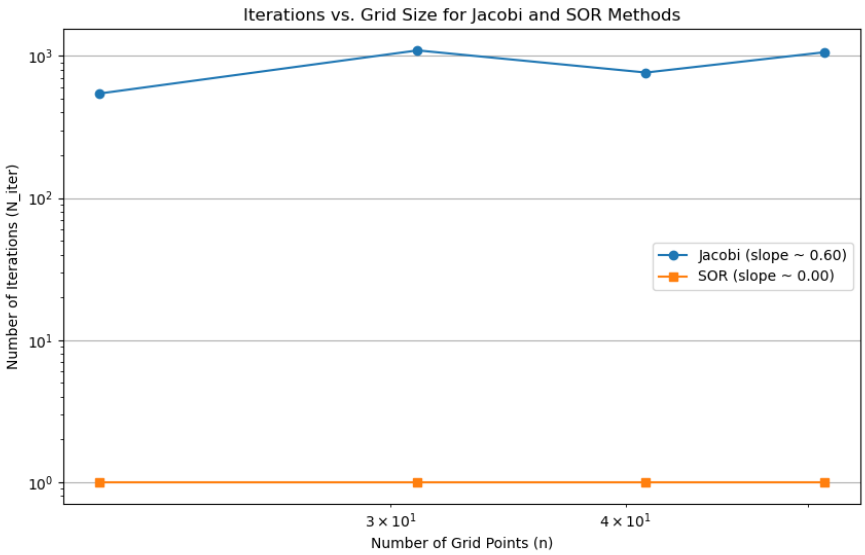

Figure 11: Comparison of convergence speed for Jacobi and SOR methods, demonstrating the superior efficiency of SOR.

3 Random Numbers and Walks

This section focused on the generation, analysis, and application of random numbers. The first part involved verifying the properties of uniform and Gaussian random number generators. The second part used these generators to simulate a 2D random walk and demonstrate its diffusive nature.

For the uniform distribution, the solution utilized ‘numpy.random.rand()‘. For the Gaussian distribution, the Box-Muller algorithm was implemented. The 2D random walk simulation involved tracking the path of a "walker" on a discrete lattice, averaging over 10,000 independent walks to obtain statistically meaningful results for the mean position and mean square displacement.

3.1 Results and Discussion

The histogram analysis effectively demonstrates the law of large numbers. With only 1,000 samples (Figure 12), the uniform distribution is "lumpy" and shows significant statistical noise. However, when the sample size is increased to 1,000,000 (Figure 13), this noise averages out, resulting in a near-perfectly flat distribution. A similar result is seen for the Gaussian generator. The smaller sample size (Figure 14) only loosely follows the theoretical curve, while the larger sample (Figure 15) produces a histogram that is almost indistinguishable from the theoretical bell curve, validating the Box-Muller implementation.

The random walk simulation demonstrated the key properties of diffusive motion. Figure 16 shows that the ensemble average of the walker's position, $\langle x_n \rangle$, fluctuates around zero. This is expected, as a step in any direction is equally probable. The mean square position, $\langle x_n^2 \rangle$, however, grows linearly with the number of steps, indicating that while the walker has no preferred direction, it does, on average, move away from its starting point. This is the definition of a random walk. Figure 17 confirms this by showing that the total mean square distance from the origin, $\langle r^2 \rangle$, is directly proportional to time (number of steps). The excellent linear fit validates the diffusive relationship, $\langle r^2 \rangle \propto t$.

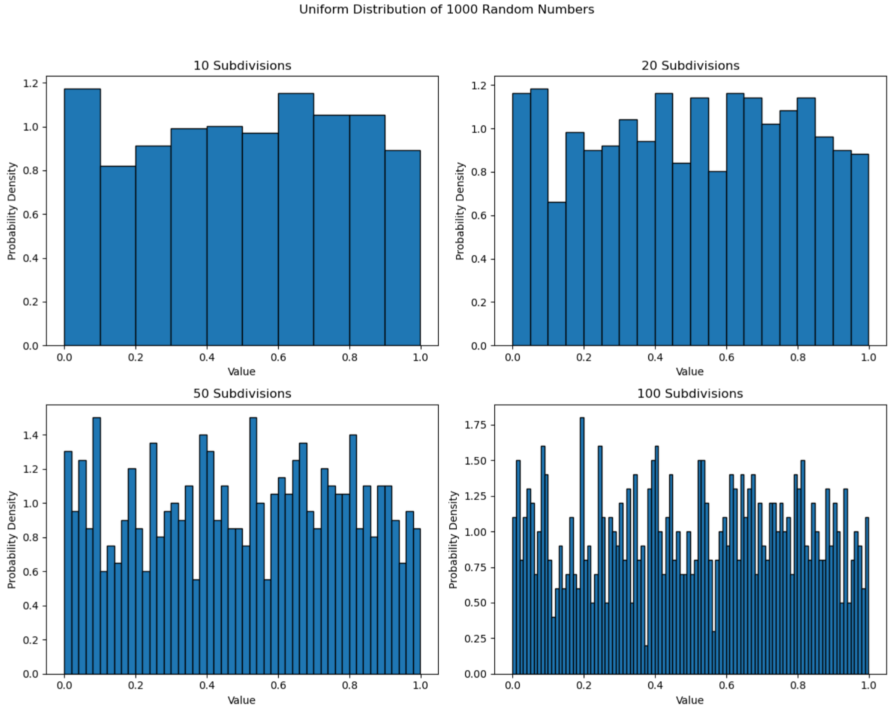

Figure 12: Histogram of 1,000 uniformly distributed random numbers.

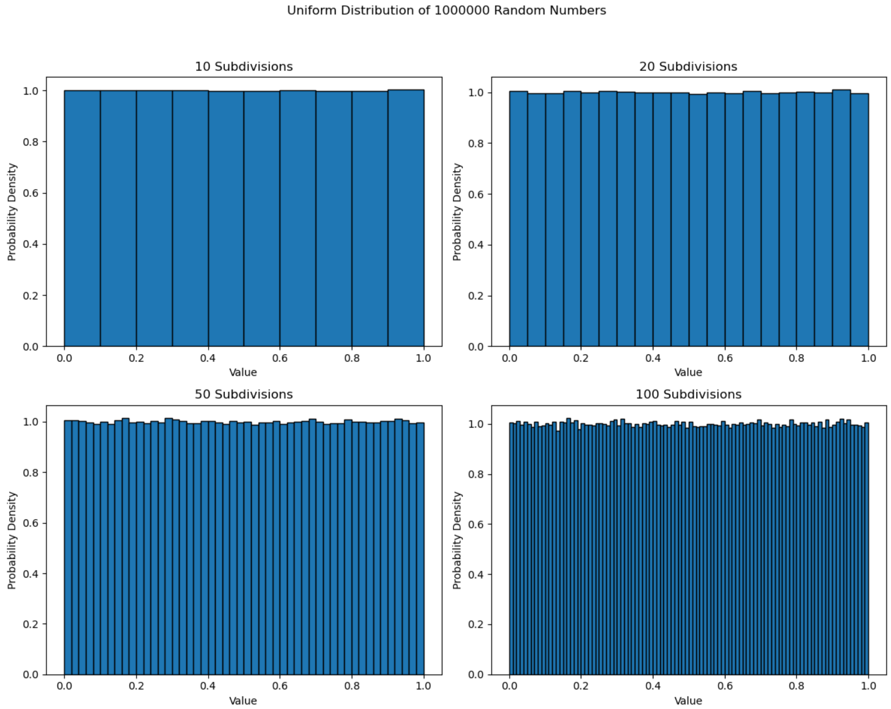

Figure 13: Histogram of 1,000,000 uniformly distributed random numbers, showing a flat distribution.

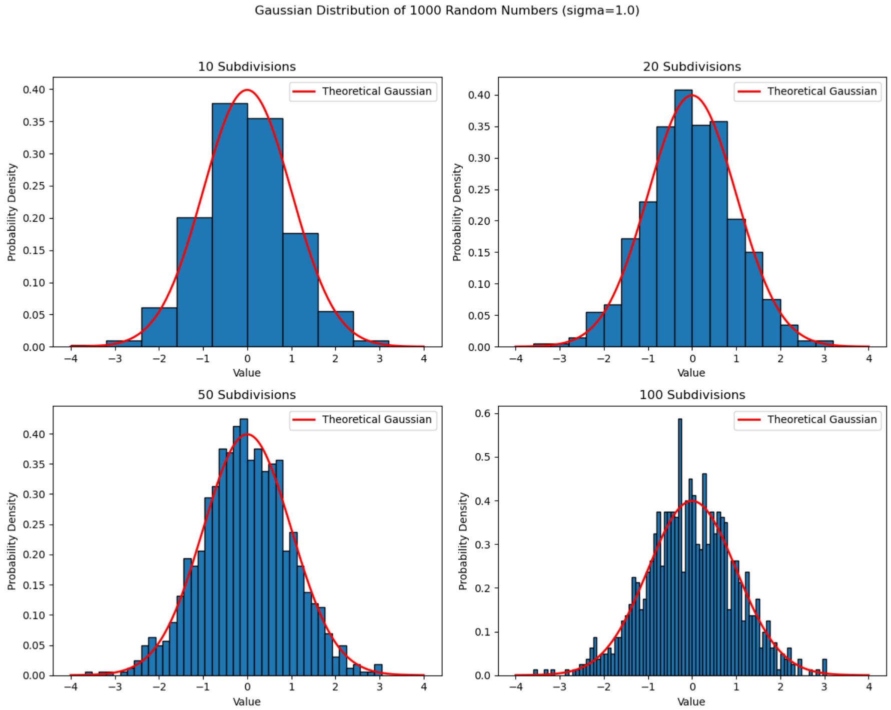

Figure 14: Histogram of 1,000 Gaussian-distributed random numbers.

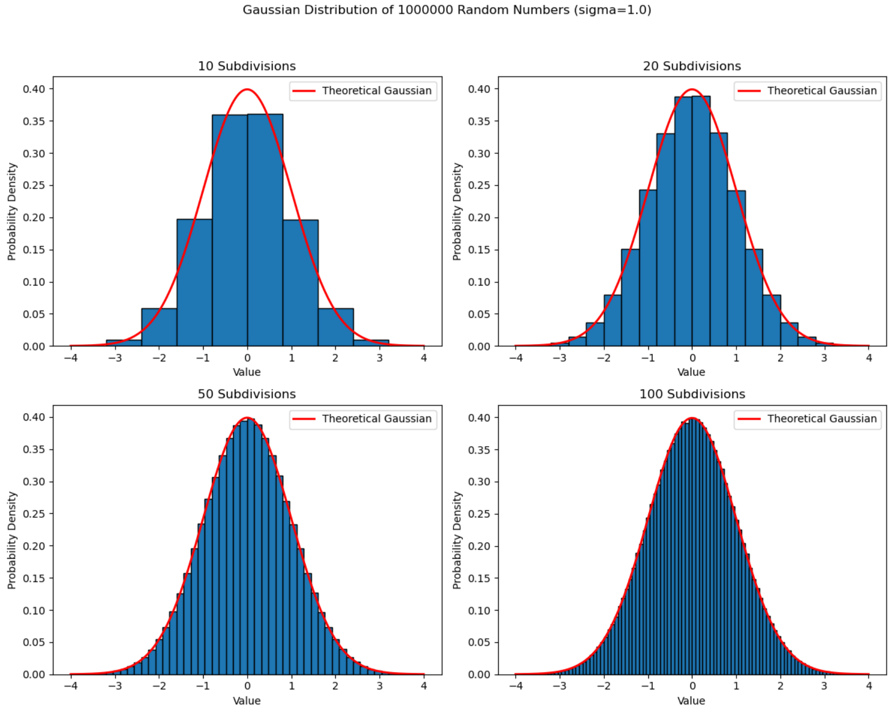

Figure 15: Histogram of 1,000,000 Gaussian-distributed random numbers, matching the theoretical curve.

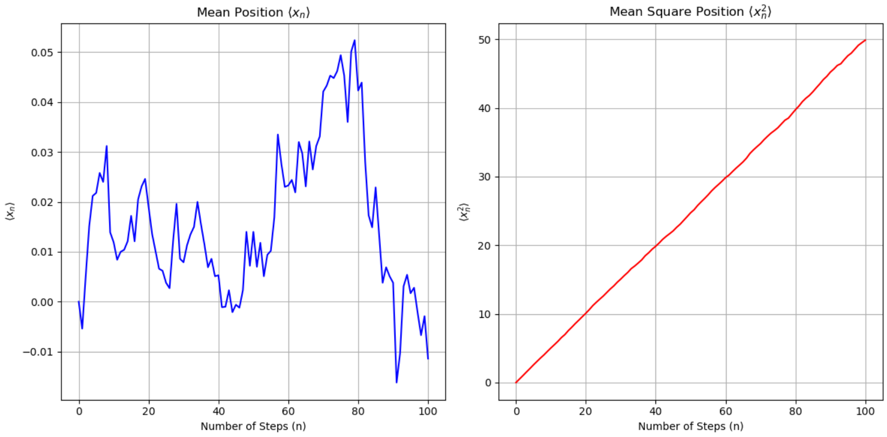

Figure 16: Left: The mean position $\langle x_n \rangle$ fluctuates around zero. Right: The mean square position $\langle x_n^2 \rangle$ grows linearly with time.

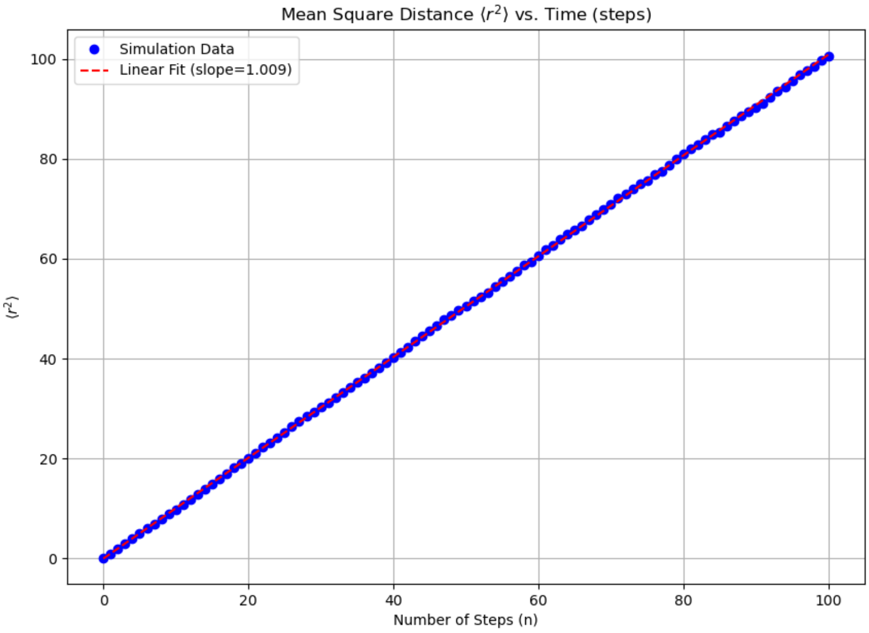

Figure 17: The mean square distance from the origin grows linearly with time, confirming diffusive motion.

4 Diffusion Equation and Gas Mixing

The final section explored the concept of diffusion by directly solving the 1D diffusion equation and by creating a particle-based simulation of two gases mixing in a 2D container.

The 1D diffusion equation, $\frac{\partial \rho}{\partial t} = D\frac{\partial^2 \rho}{\partial x^2}$, was solved numerically using the Forward-Time Centered-Space (FTCS) finite difference method. The extra credit problem simulated diffusion from a microscopic perspective by modeling individual gas particles moving randomly on a 2D grid.

4.1 Results and Discussion

The numerical solution to the 1D diffusion equation (Figure 18) correctly shows the physical process of diffusion. The initial, sharp "box" profile, representing a high concentration gradient, immediately begins to smooth out and spread. Over time, the profile evolves into the characteristic bell curve shape of a Gaussian distribution. The dashed lines, representing Gaussian fits to the data at each time step, align perfectly with the simulation, and the extracted width of these fits, σ, was confirmed to grow according to the theoretical relation $\sigma(t) = \sqrt{2Dt}$.

The gas mixing simulation provides a powerful link between microscopic random motion and macroscopic diffusion. The snapshots in Figure 19 give a clear visual representation of this process, as the initially separated gases intermingle over time. This is quantified in the density plots. The plot from a single trial (Figure 20) is highly stochastic and noisy, reflecting the random path of each individual particle. However, by averaging the densities over 100 trials (Figure 21), this statistical noise is smoothed out, revealing the underlying deterministic behavior. The resulting smooth curves clearly depict the same diffusive spreading seen in the direct solution of the diffusion equation, providing a microscopic validation for the macroscopic model.

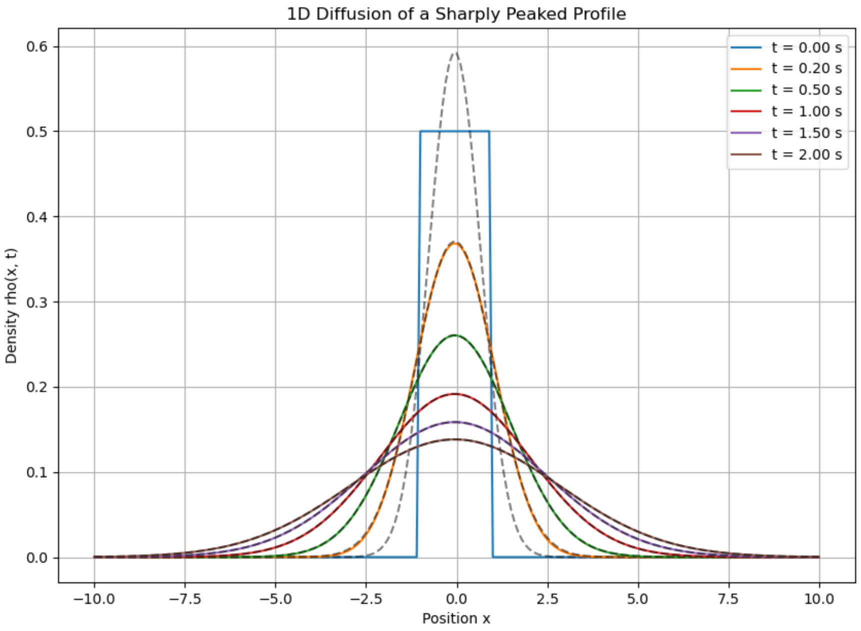

Figure 18: Numerical solution to the 1D diffusion equation, showing a box profile spreading into a Gaussian.

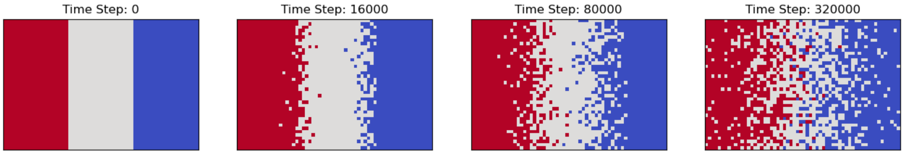

Figure 19: Snapshots of the 2D gas mixing simulation at different time steps.

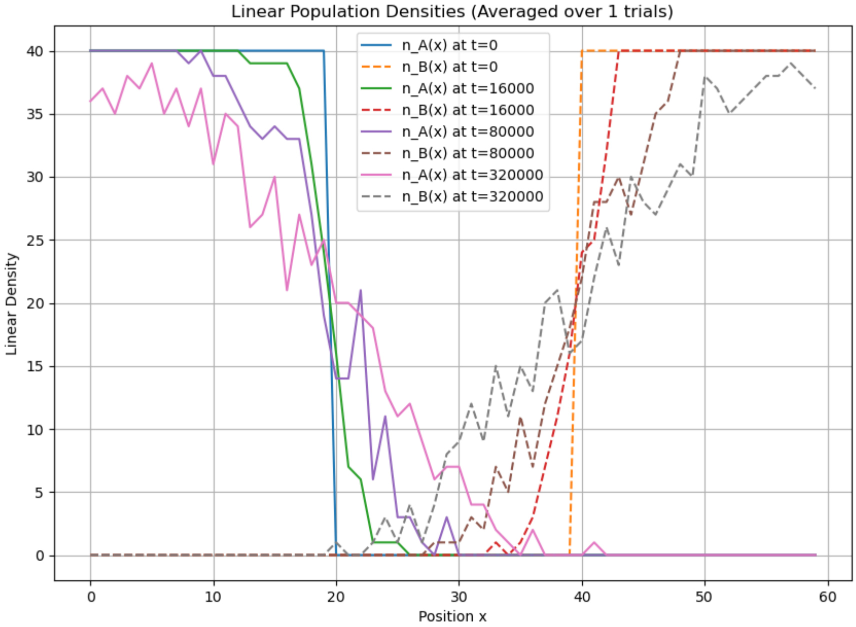

Figure 20: Linear population densities from a single simulation trial, showing significant statistical noise.

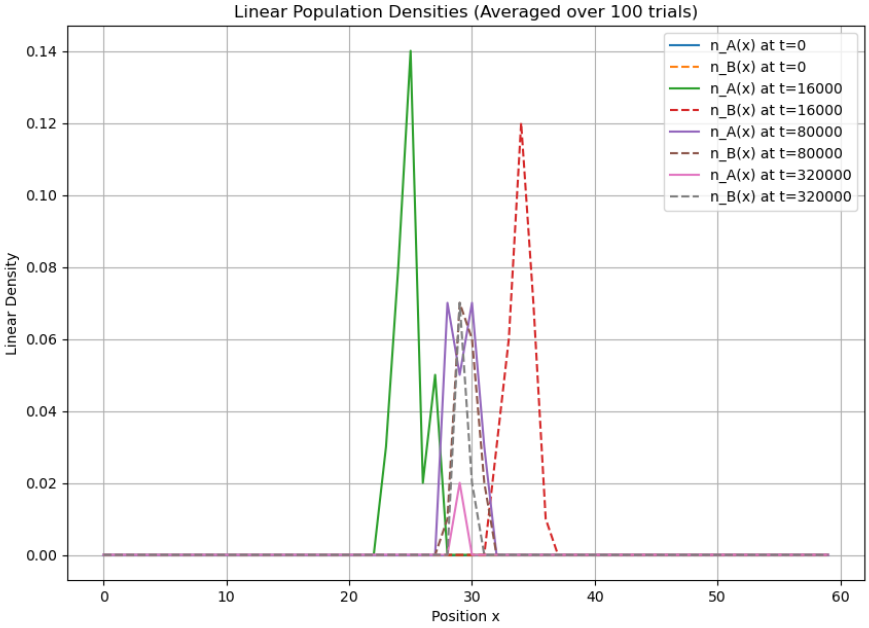

Figure 21: Averaged linear densities of the two gases over 100 trials, showing smooth diffusive spreading.

Contributions

All contributors to this homework assignment contributed equally to the final submission.

References

The following online video resources were consulted for understanding the implementation of the numerical methods used in this assignment:

- 4th-Order Runge Kutta Method for ODEs
- Runge-Kutta Integrator Overview: All Purpose Numerical Integration of Differential Equations
- Euler-Cromer Method for Beginners
- The Jacobi Method
- 3.4.6-Linear Algebra: Gauss-Seidel and Jacobi
- Numerical Methods for Linear Systems - SOR
- Box-Muller Transform + R Demo
- The Finite Difference Method (1D)

Now python related resources -

- Python Tutorial: Using Try/Except Blocks for Error Handling
- NumPy vs SciPy
- Matplotlib Animations in Python