Computational Physics: Homework #5

Hao Wang

December 2, 2025

Problem 1: Time-Dependent Schrödinger Equation (Exercise 9.8)

Summary

In this problem, we numerically solved the time-dependent Schrödinger equation for an electron in a one-dimensional infinite potential well. We employed the Crank-Nicolson method to evolve a Gaussian wave packet over time. The results demonstrate the movement of the wave packet and spreading of the wave packet due to dispersion.

Methods

We solve the Schrödinger equation for a particle of mass M in a box of length L:

$$-\frac{\hbar^2}{2M}\frac{\partial^2\psi}{\partial x^2} = i\hbar\frac{\partial\psi}{\partial t} \tag{1}$$

The continuous domain is discretized into $N = 1000$ spatial points with spacing $a = L/N$ and time steps of size $h = 10^{-18}$ s.

We utilize the Crank-Nicolson scheme, which is an implicit, unitary, and unconditionally stable method. The discrete equation is rearranged into a linear system of the form:

$$\mathbf{A}\psi(t+h) = \mathbf{B}\psi(t) \tag{2}$$

where $\mathbf{A}$ and $\mathbf{B}$ are tridiagonal matrices with components derived from the Hamiltonian:

$$A_{jj} = 1 + \frac{i\hbar h}{2ma^2}, \qquad A_{j,j\pm1} = -\frac{i\hbar h}{4ma^2} \tag{3}$$

$$B_{jj} = 1 - \frac{i\hbar h}{2ma^2}, \qquad B_{j,j\pm1} = \frac{i\hbar h}{4ma^2} \tag{4}$$

The code implements this by constructing the full A and B matrices using `numpy`. At each time step, the vector $v = B\psi(t)$ is computed via matrix multiplication, and the new state $\psi(t+h)$ is found by solving the linear system $Ax = v$ using `numpy.linalg.solve`.

Parameters:

- Mass $M = 9.109 \times 10^{-31}$ kg
- Box Length $L = 10^{-8}$ m
- Time step $h = 10^{-18}$ s
- Initial State: Normalized Gaussian centered at $x_0 = L/2$ with width $\sigma = 10^{-10}$ m and wavenumber $\kappa = 5 \times 10^{10}$ m^{-1}.

Results

The simulation tracks the real part of the wavefunction and the probability density $|\psi|^2$.

As shown in the wavefunction_evolution.gif (uploaded to github), the electron begins as a localized packet. As it evolves:

1. The packet translates to the right with its group velocity.
2. It broadens over time, consistent with the uncertainty principle and dispersion relation of the free particle.

Problem 2: Poisson's Equation with Relaxation

Summary

This problem explores the numerical solution of Poisson's equation, $\nabla^2\phi = -\rho/\epsilon_0$, for a system of charged particles in a 2D grounded box. We mapped point charges to a grid using the Cloud-in-Cell (CIC) method and calculated the electric potential using two iterative techniques: standard relaxation (Jacobi method) and Gauss-Seidel with Successive Over-Relaxation (SOR). We optimized the SOR parameter ω using the Golden Section Search method, achieving a significant reduction in computation time.

Methods

The simulation domain is a 100×100 grid with grid spacing $a = 1.0$ (arbitrary units). The boundary conditions are fixed at $\phi = 0$ (grounded walls).

Part (a): Charge Assignment We import particle positions from `particles.dat`. To assign point charges to the discrete grid, we use the Cloud-in-Cell (CIC) weighting scheme. A charge at (x, y) contributes to its four nearest grid points (i, j), $(i + 1, j)$, $(i, j + 1)$, and $(i + 1, j + 1)$ weighted by the overlap of a unit cell area centered on the particle with the grid cells.

Part (b): Standard Relaxation We discretize the Laplacian using finite differences. The standard relaxation update (Jacobi method) computes the new potential ϕ' at grid point (i, j) as the average of its neighbors plus a source term:

$$\phi'_{i,j} = \frac{1}{4}\left(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} + \frac{a^2\rho_{i,j}}{\epsilon_0}\right) \tag{5}$$

We iterate this update until the maximum difference between steps drops below a tolerance of $\delta = 10^{-8}$.

Part (c): Successive Over-Relaxation (SOR) To accelerate convergence, we employ the Gauss-Seidel method with over-relaxation. The grid is updated in-place:

$$\phi_{new} = (1 - \omega)\phi_{old} + \omega\phi^*_{Gauss-Seidel} \tag{6}$$

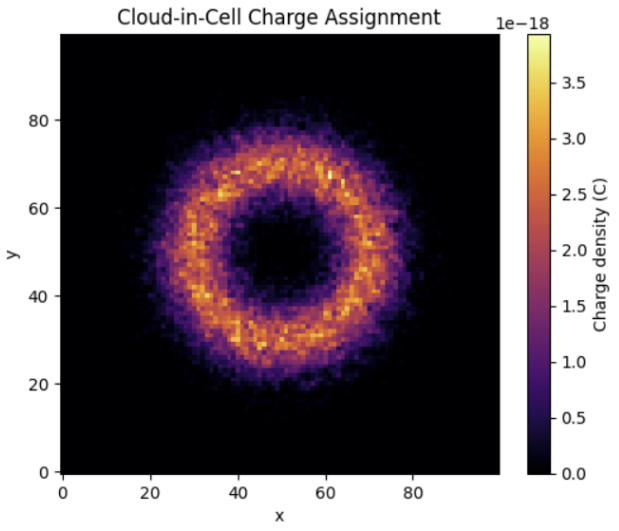
where $\phi^*_{Gauss-Seidel}$ is the value calculated using the most recent neighbor values. We determine the optimal relaxation parameter ω by minimizing the number of iterations required for convergence using the Golden Section Search algorithm.

Results

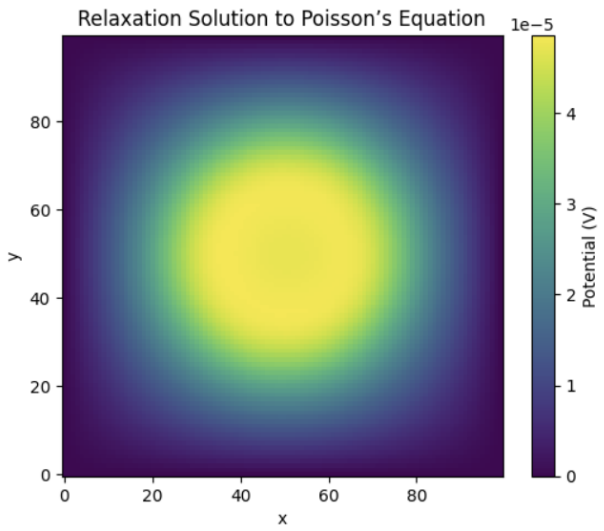
Charge Density and Potential: The CIC method successfully produced a smooth charge density grid from the point particles. The resulting potential field reflects the distribution of charges, zeroing out at the boundaries as expected.

Convergence Comparison:

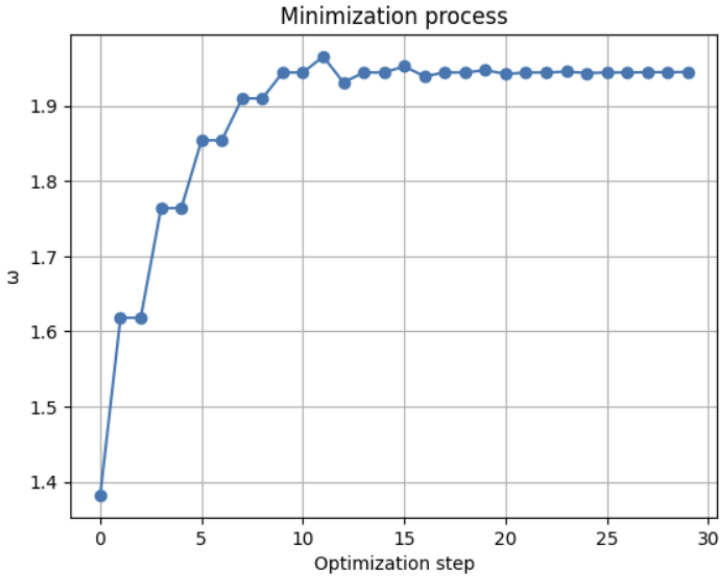
- **Standard Relaxation:** The method required **2506 iterations** to converge to a tolerance of 10^{-8}.
- **SOR Optimization:** The Golden Section Search identified an optimal over-relaxation parameter of approximately $\omega \approx 1.945$.
- **Performance Gain:** Using the optimal ω, the SOR solver converged in only **133 iterations**. This represents a speedup factor of nearly $20\times$ compared to the standard relaxation method.



(a) Charge density (ρ) via Cloud-in-Cell.



(b) Calculated Potential (ϕ).



(c) Golden Section Search for optimal ω.

Figure 1: Solution to Poisson's equation and optimization of the SOR parameter.

Code

The Python code for both problems, including the Crank-Nicolson solver, the Cloud-in-Cell implementation, and the Golden Section Search for SOR optimization, is hosted in the following repository:

`https://github.com/hw3926/Computational-Physics/tree/main/HW5`