Homework 5 Computational Physics

Giacomo Recine*

2 December 2025

Contents

*gr2640@nyu.edu

1 Introduction

In this homework we will learn how to solve partial differential equations using different methods. First of all, in Sec. 2 we will solve the Schrödinger equation for a particle in a box of length L. For this exercise we will use the Crank–Nicolson method, and we will also study the time evolution of a wavefunction. In Sec. 3 we will instead solve the Poisson equation using first the standard relaxation method, and then the over-relaxation method. Our goal is to optimize the computation by determining the value of the over-relaxation parameter ω that minimizes the number of iterations. For this reason, we will plot the number of iterations required for the maximum difference between any grid cell in two successive steps to be less than 1.0×10^{-10}. We will perform this optimization using the golden–search method and determine the optimal value of ω with a precision of 0.001.

2 Exercise 9.8 from Newman

Perhaps the most important partial differential equation in physics is the Schrödinger equation. In this exercise we apply the Crank–Nicolson method to solve the full time-dependent Schrödinger equation and study the time evolution of a wavefunction.

First of all, let us review how the method works. We consider the Schrödinger equation in one spatial dimension. The extension to two or three dimensions follows similar principles but requires substantially more computational effort, so we restrict ourselves to one dimension for clarity. For a particle of mass M and no potential energy, the time-dependent Schrödinger equation in one dimension reads

$$\frac{\hbar^2}{2M}\frac{\partial^2\psi}{\partial x^2} = i\hbar\frac{\partial\psi}{\partial t}.$$

We place the particle in an infinite potential well with impenetrable walls at $x = 0$ and $x = L$, so that the wavefunction satisfies

$$\psi(0,t) = \psi(L,t) = 0.$$

Replacing the spatial second derivative with a finite difference, and applying the forward Euler method, we obtain the FTCS scheme

$$\psi(x,t+h) = \psi(x,t) + h\,\frac{i\hbar}{2Ma^2}\left[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\right],$$

where a is the spatial grid spacing and h is the time step. Note that h should not be confused with Planck's constant $\hbar$. Performing a similar step in reverse, we get the implicit equation

$$\psi(x,t+h) - h\,\frac{i\hbar}{2Ma^2}\left[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\right] = \psi(x,t).$$

Taking the average of the explicit and implicit scheme yields the Crank–Nicolson method

$$\begin{aligned}\psi(x,t+h) - \frac{h\,i\hbar}{4Ma^2}\left[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\right] \\ = \psi(x,t) + \frac{h\,i\hbar}{4Ma^2}\left[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\right].\end{aligned}$$

This gives a linear system of equations, one for each spatial grid point.

Now, because $\psi(0,t) = \psi(L,t) = 0$, the unknowns are the interior values $\psi(a,t), \psi(2a,t), \psi(3a,t), \ldots$. Arrange these into the vector

$$\psi(t) = \begin{pmatrix} \psi(a,t) \\ \psi(2a,t) \\ \psi(3a,t) \\ \vdots \end{pmatrix},$$

then the Crank-Nicolson equations can be written compactly as

$$A\,\psi(t+h) = B\,\psi(t),$$

where A and B are symmetric tridiagonal matrices of the form

$$A = \begin{pmatrix} a_1 & a_2 & & & \\ a_2 & a_1 & a_2 & & \\ & a_2 & a_1 & a_2 & \\ & & \ddots & \ddots & \ddots \\ & & & a_2 & a_1 \end{pmatrix}, \qquad B = \begin{pmatrix} b_1 & b_2 & & & \\ b_2 & b_1 & b_2 & & \\ & b_2 & b_1 & b_2 & \\ & & \ddots & \ddots & \ddots \\ & & & b_2 & b_1 \end{pmatrix},$$

with coefficients

$$a_1 = 1 + \frac{h\, i\hbar}{2ma^2}, \qquad a_2 = -\frac{h\, i\hbar}{4ma^2},$$
$$b_1 = 1 - \frac{h\, i\hbar}{2ma^2}, \qquad b_2 = \frac{h\, i\hbar}{4ma^2}.$$

The system $\mathbf{A}\psi(t+h) = \mathbf{B}\psi(t)$ is of the form $\mathbf{A}x = v$ and can be solved efficiently via tridiagonal Gaussian elimination. Specifically, since the matrix $\mathbf{A}$ is tridiagonal in this case, we can use the fast tridiagonal version of Gaussian elimination.

2.1 Solution of the exercise

Consider an electron of mass $M = 9.109 \times 10^{-31}\,\mathrm{kg}$ in a box of length $L = 10^{-8}\,\mathrm{m}$. At time $t = 0$ the wavefunction is

$$\psi(x, 0) = \exp\left[-\frac{(x - x_0)^2}{2\sigma^2}\right] e^{i\kappa x},$$

with

$$x_0 = \frac{L}{2}, \qquad \sigma = 10^{-10}\,\mathrm{m}, \qquad \kappa = 5 \times 10^{10}\,\mathrm{m}^{-1},$$

and $\psi(0, t) = \psi(L, t) = 0$. This wavefunction is not normalized, but normalization is irrelevant for this exercise because the Schrödinger equation is linear and any overall constant cancels out.

Then, first we implement a program that performs a single Crank–Nicolson time step, computing the vector $\psi(t)$ using $N = 1000$ spatial slices with $a = L/N$. Using this, we then compute

$$v = B\,\psi(t)$$

with a time step of $h = 10^{-18}$, where the components of v are given by

$$v_i = b_1\,\psi_i + b_2\,(\psi_{i+1} + \psi_{i-1}).$$

Finally, we solve the tridiagonal system

$$A\,\psi(t+h) = v$$

using the tridiagonal Gaussian elimination method. To implement this, we employed the Python library `banded.py` (available in the online resources). This solver efficiently handles complex-valued arrays, which is necessary for representing both the wavefunction ψ and the matrix $\mathbf{A}$.

Afterwards, we extended our program to create an animation of the solution by displaying the real part of the wavefunction at each time step (see the code linked in Sec. 5). At the initial time $t = 0$, the wavefunction is shown in Fig. 1

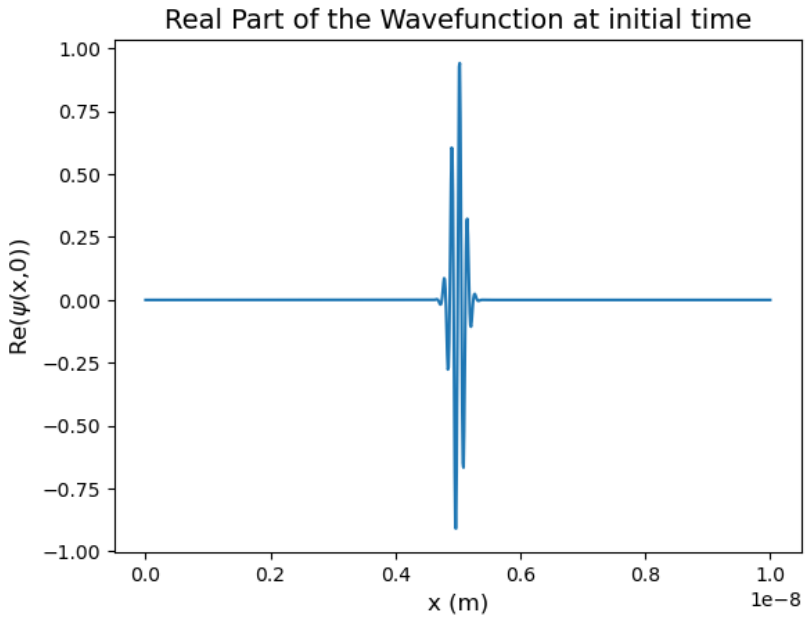

Figure 1: Real part of the wavefunction at the initial time $t = 0$.

At later times $t > 0$, the wavefunction evolves according to the Crank–Nicolson scheme. An example of the evolved wavefunction is shown in Fig. 2.

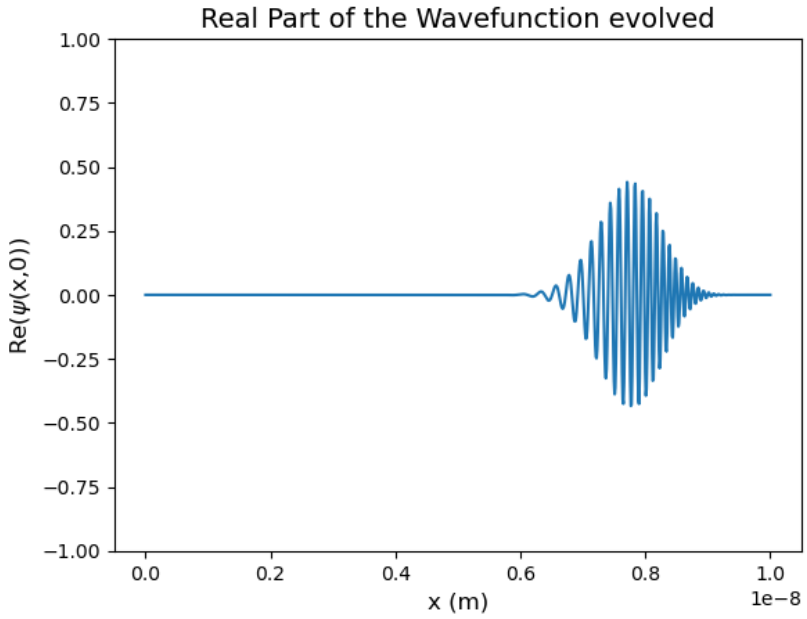

Figure 2: Real part of the wavefunction at a later time $t > 0$.

The fact that the wavefunction spreads out in time is a direct consequence of the dispersion of a free wave packet. Our initial wavefunction is indeed a superposition of many plane waves with different wavenumbers k, which means that the different Fourier components propagate with different group velocities. As a result, the wave packet gradually spreads as time evolves. Since the total probability must remain normalized to one, the amplitude of the wavefunction decreases as the packet becomes wider. When the simulation is run for longer

times, it can be seen that the real part of the wave function begins to resemble a standing wave that extends across the entire box. This behavior arises because the Gaussian wave packet undergoes multiple reflections at the walls of the well, producing both destructive and constructive interference.

3 Solving the Poisson Equation Using Relaxation and Over-Relaxation Methods

First of all we refresh the idea of solve the Poisson equation by using the relaxation method. A fundamental technique for the solution of partial differential equations is the method of finite differences. So let's consider the Poisson equation

$$\nabla^2\Phi = -\frac{\rho}{\epsilon_0}, \tag{1}$$

the method of finite differences involves dividing space into a grid of discrete points. Many kinds of grids are used, depending on the particular problem in hand, but in the simplest and commonest case we use a regular, square grid. Suppose the spacing between grid points is a, and consider a point at position (x, y). As discussed previously, second derivatives can be approximated using finite difference formulas, the same can applies to partial derivatives. Using the results for the second derivative of Φ in the x-direction is approximated by

$$\frac{\partial^2\Phi}{\partial x^2} = \frac{\Phi(x+a,y) + \Phi(x-a,y) - 2\Phi(x,y)}{a^2}. \tag{2}$$

This formula expresses the second derivative in terms of the values $\Phi(x, y)$, $\Phi(x - a, y)$, and $\Phi(x + a, y)$ at three adjacent grid points. Similarly, the second derivative in the y-direction is

$$\frac{\partial^2\Phi}{\partial y^2} = \frac{\Phi(x,y+a) + \Phi(x,y-a) - 2\Phi(x,y)}{a^2}. \tag{3}$$

The Laplacian operator in two dimensions is

$$\nabla^2\Phi(x,y) = \frac{\Phi(x+a,y) + \Phi(x-a,y) + \Phi(x,y+a) + \Phi(x,y-a) - 4\Phi(x,y)}{a^{\,2}}. \tag{4}$$

In other words, we add the values of Φ at the four grid points immediately adjacent to (x, y), subtract four times the value of Φ at (x, y), and finally divide by a^2. To apply the relaxation method, we first rearrange the previous equation, into the form

$$\Phi(x,y) = \frac{1}{4}\big[\Phi(x+a,y) + \Phi(x-a,y) + \Phi(x,y+a) + \Phi(x,y-a)\big] + \frac{a^2}{4\epsilon_0}\rho(x,y), \tag{5}$$

which states that the value of $\Phi(x, y)$ is simply the average of the values at the four nearest neighboring grid points. We now impose the boundary conditions by fixing $\Phi(x, y)$ on the edges of the domain: in the present electrostatic problem, $\Phi = 0$ on all four walls. Next, we assign initial guesses for $\Phi(x, y)$ at all interior grid points. These guesses do not need to be accurate, for instance, we could set all interior values to zero, which is clearly incorrect, but still sufficient for the algorithm. We then compute updated values $\Phi'(x, y)$ at all interior points

$$\Phi'(x,y) = \frac{1}{4}\big[\Phi(x+a,y) + \Phi(x-a,y) + \Phi(x,y+a) + \Phi(x,y-a)\big] + \frac{a^2}{4\epsilon_0}\rho(x,y). \tag{6}$$

These newly computed values are then fed back into the right-hand side to generate another updated set, and the process is repeated iteratively until Φ converges to a stable configuration. This iterative procedure for solving Laplace's equation is known as the Jacobi method.

3.1 Solution of Question a

For our purposes we take the data file `particle.dat`, which contains a set of (x, y) coordinates for a collection of charged particles, each with charge equal to the electron charge. The coordinates span the range

$[0, 100]$. Using the cloud-in-cell technique, we assign these charges to a two-dimensional grid of size $[0, M]$ on each side, with $M = 100$. We take the center of each grid cell (i, j) to be located at $(i + 0.5,\, j + 0.5)$. Our goal is then to produce an image of the resulting charge-density field.

First of all, let us explain the cloud-in-cell (CIC) method. For each particle p, we associate a shape function $S(x)$. Consequently, the fraction of mass assigned to a grid cell centered at x_i is determined by the weight function $W(x_p - x_i)$.

In one dimension, the fraction of mass (or charge) assigned to a grid cell centered at x_i is determined by the weight function W, obtained by integrating the particle shape function $S(x)$ over the cell width

$$W(x_p - x_i) = \int_{x_i - \frac{\Delta x}{2}}^{x_i + \frac{\Delta x}{2}} S(x_p - x')\, \mathrm{d}x'. \tag{7}$$

The generalization to two dimensions is performed by separation of variables

$$W(\vec{r}_p - \vec{r}_{i,j}) = W(x_p - x_i)\, W(y_p - y_j). \tag{8}$$

Consequently, the charge density $\rho_{i,j}$ defined on the grid is given by summing the contribution of all N particles and normalizing by the cell area

$$\rho_{i,j} = \frac{1}{\Delta x \Delta y} \sum_{p=1}^{N} q_p\, W(\vec{r}_p - \vec{r}_{i,j}). \tag{9}$$

In this work, we adopt the Cloud-in-Cell (CIC) scheme. This implies that the shape function is defined as

$$S(x) = \begin{cases} \frac{1}{\Delta x} & \text{for } |x| < \frac{\Delta x}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Then using this method explained, we have the following image of the charge density field

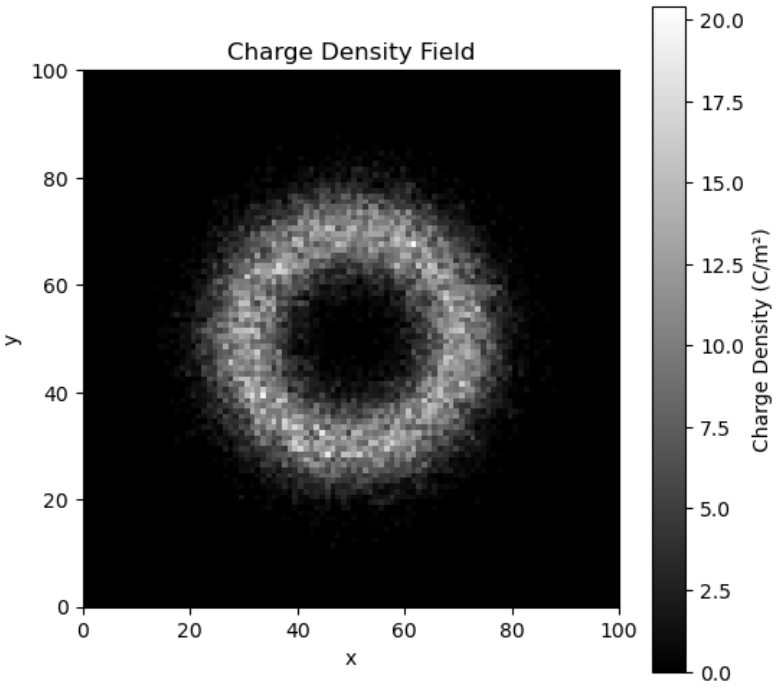

Figure 3: Image of the density charge field.

3.2 Solution of Question b

Now we will use the standard relaxation method in order to solve the Poisson's equation, which we have explained in the previous section. Then plotting the image of the resulting potential field, we have

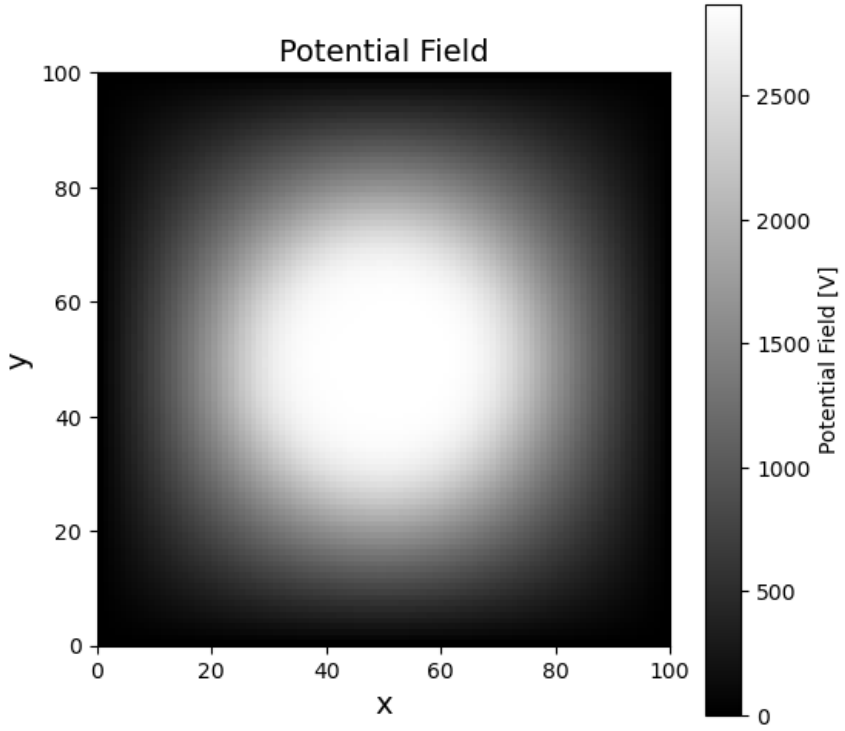

Figure 4: Image of the potential field using the relaxation.

The number of iterations required to obtain the solution is $N = 50338$. To assess convergence, we follow the procedure described by Newman, requiring that the maximum difference between any grid cell and its value from the previous iteration be less than 10^{-10}.

3.3 Solution Question c

Now we will use the Guass-Seidel overrelaxation method to solve for Poisson's equation. In that case The Jacobi method can be accelerated by updating each grid point using the newly computed values of its neighbors as soon as they are available. This variant is known as the Gauss–Seidel method, and requires storing only one array, since old values can be overwritten immediately

$$\Phi(x,y) \leftarrow \frac{1}{4}\big[\Phi(x+a,y) + \Phi(x-a,y) + \Phi(x,y+a) + \Phi(x,y-a)\big] + \frac{a^2}{4\epsilon_0}\rho(x,y). \tag{11}$$

Combining Gauss–Seidel with overrelaxation yields a faster and stable scheme:

$$\Phi(x,y) \leftarrow \frac{1+\omega}{4}\big[\Phi(x+a,y) + \phi(x-a,y) + \Phi(x,y+a) + \Phi(x,y-a)\big] - \omega\,\Phi(x,y) + \frac{a^2}{4\epsilon_0}\rho(x,y).$$

Unlike the Jacobi version, which is unstable on square grids, the Gauss–Seidel approach is both faster and more memory-efficient, making it preferable in most applications.

Indeed, following this procedure and using, for instance, $\omega = 0.95$, we obtain the previously mentioned result in only 601 iterations. The next question is to find the optimal value of ω, within a precision of 0.001. To achieve this, we apply the Golden Section Search method on the function $\phi(x,y)$ obtained from Eq. (11).

First, we briefly recall the Golden Section Search method. The Golden Section Search is a technique to find the extremum (minimum or maximum) of a unimodal function by successively narrowing the range of values inside which the extremum is known to exist. Thus the complete golden–ratio search goes as follows:

1. choose two initial outside points x_1 and x_4, then we can determine the two internal points x_2 and x_3 as follows. From the definition

$$z = \frac{x_4 - x_1}{x_3 - x_1}, \tag{12}$$

we immediately obtain

$$x_3 = x_1 + \frac{x_4 - x_1}{z}. \tag{13}$$

Using the symmetry condition of the two internal points,

$$x_2 - x_1 = x_4 - x_3, \tag{14}$$

we can derive

$$x_2 = x_4 - \frac{x_4 - x_1}{z}. \tag{15}$$

where can be find that $z = \frac{1+\sqrt{5}}{2}$, or $1/z = \frac{\sqrt{5}-1}{2}$.

Then we evaluate $f(x)$ at each of the four points and we check that at least one of the values at x_2 and x_3 is less than the values at both x_1 and x_4. Also we choose a target accuracy for the position of the minimum;

2. if $f(x_2) < f(x_3)$ then the minimum lies between x_1 and x_3. In this case, x_3 becomes the new x_4, x_2 becomes the new x_3, and there will be a new value for x_2, chosen once again according to the golden–ratio rule. Thus, we evaluate $f(x)$ at this new point;

3. otherwise, the minimum lies between x_2 and x_4. Then x_2 becomes the new x_1, x_3 becomes the new x_2, and there will be a new value for x_3, and thus we valuate $f(x)$ at this new point;

4. if $x_4 - x_1$ is greater than the target accuracy, repeat from step 2. Otherwise, we calculate $\frac{1}{2}(x_2 + x_3)$, which is the final estimate of the position of the minimum.

This method efficiently narrows the search interval to converge to the optimal value of ω.

Using this method, we have plotted the number of iteration steps required, with a precision of 0.001, as a function of the different values of ω, obtaining

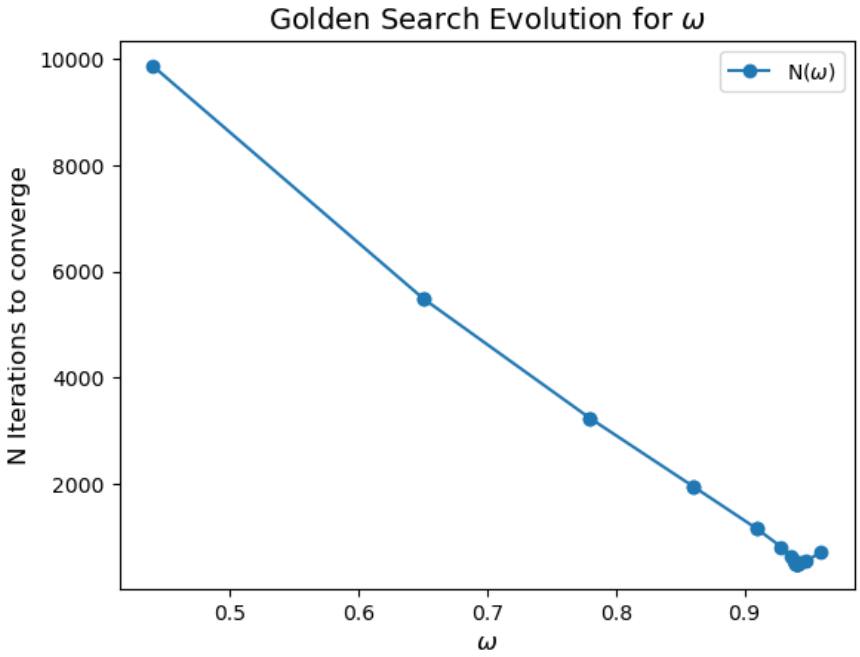

Figure 5: We study the behavior of the number of iterations required to solve the partial differential equation as a function of the relaxation parameter ω.

We see that, as the value of ω increases, the number of iterations required to achieve the accuracy of 10^{-10} decreases, with the optimal value found being $\omega = 0.940$, which corresponds to $N = 601$ iterations.

4 Conclusion

We have learned how to implement some of the most useful methods for solving partial differential equations, which appear in many areas of Physics. In Sec. 2 we solved the problem using the Crank–Nicolson method in order to obtain the time evolution of the Schrödinger equation for a particle in a box. Finally, in Sec. 3 we solved another crucial partial differential equation, the Poisson equation. We first applied the standard relaxation method and then the over-relaxation method plus the Gauss-Seidel method to optimize the computation. Using the Golden Search method, we determined the optimal value of ω such that the number of steps required to achieve a maximum difference of 1.0×10^{-10} between any grid cell in two successive iterations is minimized, with a precision of 0.001.

5 Code

The code containing all the results can be found at the following Repository GitHub.