Project 2 :2D Poisson Solver for a Discrete Charge Distribution with Boundary Conditions and GUI Visualization

Objective

Develop a numerical solver for the two-dimensional Poisson equation for electrostatics on a rectangular domain containing a *discrete charge distribution*. The project must:

1. compute the electric potential field $V(x, y)$ on a 2D grid,

2. compute and visualize the electric field $\mathbf{E}(x, y) = -\nabla V$,

3. support multiple boundary-condition types (Dirichlet and Neumann, including mixed cases),

4. provide an interactive GUI to define charges, select boundary conditions, run the solver, and visualize results.

Governing Equation

Let $\rho(x, y)$ be the charge density and ε be the permittivity (use $\varepsilon = \varepsilon_0$ by default). The 2D Poisson equation is

$$\nabla^2 V(x, y) = -\frac{\rho(x, y)}{\varepsilon}. \tag{7}$$

On a uniform grid with spacing $\Delta x = \Delta y = h$, using the 5-point stencil,

$$\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{h^2} + \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{h^2} = -\frac{\rho_{i,j}}{\varepsilon}. \tag{8}$$

Equivalently,

$$V_{i,j} = \frac{1}{4}\left(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} + \frac{h^2}{\varepsilon}\rho_{i,j}\right), \qquad \text{(interior nodes)}. \tag{9}$$

Discrete Charge Distribution (Input Model)

Represent a set of N_c point charges in the domain:

$$\{(x_k, y_k, q_k)\}_{k=1}^{N_c}.$$

Map charges to the nearest grid node (or distribute bilinearly to the four nearest nodes). For nearest-node assignment:

$$\rho_{i_k,j_k} \leftarrow \rho_{i_k,j_k} + \frac{q_k}{h^2}, \tag{10}$$

so that $q_k \approx \rho_{i_k,j_k} h^2$.

Boundary Conditions (Required)

Implement at least the following:

1) Dirichlet Boundary Condition

Potential specified on boundary:

$$V(x, y)\big|_{\partial\Omega} = V_b(x, y). \tag{11}$$

In discrete form, boundary nodes are fixed:

$$V_{i,j} = V_b \quad \text{for } (i, j) \in \partial\Omega.$$

2) Neumann Boundary Condition

Normal derivative specified on boundary (given surface field):

$$\frac{\partial V}{\partial n}\Big|_{\partial\Omega} = g_n(x, y). \tag{12}$$

For example, on the left boundary ($x = 0$) with outward normal $-\hat{x}$:

$$\frac{V_{1,j} - V_{0,j}}{h} = -g_n(0, y_j) \quad \Rightarrow \quad V_{0,j} = V_{1,j} + h\, g_n(0, y_j). \tag{13}$$

Similarly derive formulas for right, top, bottom boundaries.

3) Mixed Boundary Conditions (Recommended)

Allow each edge to be independently set as Dirichlet or Neumann (e.g., left/right Dirichlet, top/bottom Neumann).

Suggested Input Parameters (GUI Inputs)

Domain and Grid

- Domain size: L_x, L_y [m]
- Grid resolution: N_x, N_y (or h)
- Permittivity: ε (default ε_0)

Charge Specification

- Number of charges N_c
- For each charge: position (x_k, y_k) and magnitude q_k
- Charge placement mode: manual click-to-place, or table entry

Boundary Conditions

For each boundary edge (Left/Right/Top/Bottom), select:

- Type: Dirichlet or Neumann
- Value function:
  - Dirichlet: constant V_b or function $V_b(y)/V_b(x)$
  - Neumann: constant g_n or function $g_n(y)/g_n(x)$

Solver Controls

- Solver choice: Jacobi, Gauss–Seidel, Successive Over-Relaxation (SOR)
- Convergence tolerance: τ (e.g. 10^{-6})
- Maximum iterations: $N_{\max}$
- Relaxation factor (SOR): $\omega \in (1, 2)$

Algorithms (Table Format)

Algorithm 1: Grid Setup and Charge Deposition

Step	Procedure
1	Read $L_x, L_y, N_x, N_y, \varepsilon$. Compute $h = L_x/(N_x - 1)$ and verify $h = L_y/(N_y - 1)$ or handle non-square spacing.
2	Initialize arrays $V \in \mathbb{R}^{N_x \times N_y}$ and $\rho \in \mathbb{R}^{N_x \times N_y}$ with zeros.
3	For each charge (x_k, y_k, q_k), map to grid node (i_k, j_k) and update $\rho_{i_k,j_k} \leftarrow \rho_{i_k,j_k} + q_k/h^2$.
4	Apply boundary condition initialization (set boundary V if Dirichlet; store g_n if Neumann).

Algorithm 2: Iterative Poisson Solver (Gauss–Seidel / SOR)

Step	Procedure
1	Choose method: Gauss–Seidel ($\omega = 1$) or SOR ($1 < \omega < 2$). Set tolerance τ and $N_{\max}$.
2	For iter $= 1$ to $N_{\max}$:
3	Enforce Neumann boundary relations (update ghost or boundary nodes using discrete normal-derivative formulas).
4	For each interior node (i, j): compute $V^{\star}_{i,j} = \frac{1}{4}\left(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} + \frac{h^2}{\varepsilon}\rho_{i,j}\right)$.
5	Update (SOR form): $V_{i,j} \leftarrow (1 - \omega)V_{i,j} + \omega V^{\star}_{i,j}$.
6	Re-impose Dirichlet boundary nodes (keep fixed).
7	Compute residual metric (e.g. max-norm): $r = \max_{i,j} \left\lvert \nabla_h^2 V_{i,j} + \rho_{i,j}/\varepsilon \right\rvert$ or update norm $\lVert V^{(new)} - V^{(old)} \rVert_\infty$. If $r < \tau$, stop.

Algorithm 3: Electric Field Computation

Step	Procedure
1	After convergence, compute $\mathbf{E} = -\nabla V$ using centered differences for interior nodes.
2	For interior nodes: $E_x(i, j) = -(V_{i+1,j} - V_{i-1,j})/(2h)$, $E_y(i, j) = -(V_{i,j+1} - V_{i,j-1})/(2h)$.
3	Use one-sided differences at boundaries as needed.

Algorithm 4: GUI Workflow

Step	Procedure
1	User defines domain/grid and boundary conditions (edge-by-edge selection of Dirichlet/Neumann and values).
2	User places charges via mouse clicks (each click adds (x_k, y_k) and prompts for q_k) or via a table editor.
3	User selects solver type (Jacobi / Gauss–Seidel / SOR), tolerance τ, max iterations, and ω (if SOR).
4	On `Run`, execute Algorithms 1–3. During iteration, update a convergence plot (residual vs iteration).
5	Display outputs: potential colormap/contours, electric-field vectors, and charge locations overlay.
6	Provide `Reset` (clear charges and fields) and `Export` (save V, $\mathbf{E}$, and figures).

Output Criteria (Required Deliverables)

1. **Potential field:** 2D visualization of $V(x, y)$ as a heatmap and/or contour plot.
2. **Electric field:** vector plot (quiver) of $\mathbf{E}(x, y)$ over the domain.
3. **Charge overlay:** markers showing positions and signs/magnitudes of the discrete charges.
4. **Convergence monitoring:** plot of residual (or update norm) versus iteration number.
5. **Boundary condition verification:** clear indication of boundary type and value on each edge, and numerical evidence that conditions are satisfied (e.g. boundary node values for Dirichlet; approximate normal derivative for Neumann).
6. **GUI:** interactive controls for domain/grid, charge editing, boundary selection, solver settings, and visualization panels.

GUI Requirements (Minimum)

The GUI must include:

- input fields for $L_x, L_y, N_x, N_y, \varepsilon$,
- boundary-condition selectors (Dirichlet/Neumann) for each of the four edges and corresponding value inputs,
- an interactive charge editor (click-to-place and/or editable table of (x_k, y_k, q_k)),
- solver selection (Jacobi/Gauss–Seidel/SOR), tolerance τ, max iterations $N_{\max}$, and ω for SOR,
- visualization panels for: (i) V heatmap/contours, (ii) $\mathbf{E}$ quiver plot, (iii) convergence plot,
- control buttons: `Run`, `Pause/Stop`, `Reset`, and `Export`.