Numerical Simulation Exercise

Title: Solving the Laplace Equation for a Problem with Certain Boundary Conditions

Author: Mina Shiri

Part 1: Problem Statement

Solve the Laplace equation for the boundary condition of the diagram below.

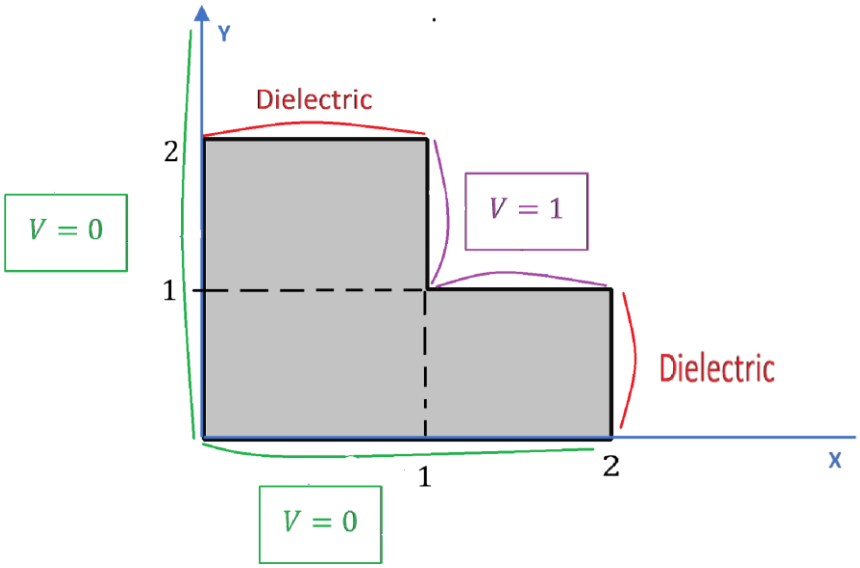


Figure 1-1) Picture of the requested issue

Questions Posed

The boundary conditions and dimensions are as follows:

1. Do you get the potential and electric field inside the high-figure space?
2. Use a mesh size the size of a small cuff. Also, check the effect of changing the size of the meshes.
3. The effect of the number of grid points on the convergence time obtained.

Part 2: Analytical Problem Solving

To solve the Laplace equation in the specified geometry and the boundary terms, we use the method of separating the variables. The steps to solve the dissolutionare as follows: $\nabla^2 V = 0$

1. Laplace's equation in the Cartesian coordinates

The Laplace equation in the Cartzian coordinates is as follows:

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$

Suppose that V(x,y) can be separated by multiplying two functions:

$$V(x, y) = X(x)Y(y)$$

2. Substitution in the Laplace equation

By placing V(x,y)=X(x)Y(y) in the Laplace equation, we have the following:

$$\frac{d^2 X}{dx^2} Y + X \frac{d^2 Y}{dy^2} = 0$$

By dividingby X(x)Y(y), the equation is separated as follows:

$$\frac{1}{X} \frac{d^2 X}{dx^2} + \frac{1}{Y} \frac{d^2 Y}{dy^2} = 0$$

Since the variables x and y are independent, each of these iodine expressions is equal to a constant valu:

$$\frac{1}{X} \frac{d^2 X}{dx^2} = -k^2, \quad \frac{1}{Y} \frac{d^2 Y}{dy^2} = k^2$$

3. Solving one-separated equations

3.1. Solution for X(x)

I have the Zair equation:

$$\frac{d^2 X}{dx^2} + k^2 X = 0$$

The solution to this equation is as follows:

$$X(x) = A \cos(kx) + B \sin(kx)$$

3.2. Solution for Y(y)

I have the Zair equation:

$$\frac{d^2Y}{dy^2} - k^2Y = 0$$

The solution to this equation is as follows:

$$Y(y) = Ce^{ky} + De^{-ky}$$

4. Composition of the whole solution

The general solution for V(x,y) is as follows:

$$V(x,y) = \sum_{n=1}^{\infty} \left[(A_n \cos(k_n x) + B_n \sin(k_n x))(C_n e^{k_n y} + D_n e^{-k_n y}) \right]$$

In which $k_n$the values are specific and are determined by the boundary line.

5. Applying the Boundary Clause

5.1. V=0 conditions at green borders

- V=0 in x=0 and x=2:
  This condition causes m to remain only thesinusoidal functions in x:

  $$X(x) = \sin\left(\frac{n\pi x}{L_x}\right), \quad L_x = 2$$

- V=0 in y = 0:
  This condition proves that $C_n e^{k_n y} + D_n e^{-k_n y}$ the Turkshould be chosen as V = 0 .

5.2. V=1 conditions at purple borders:

This Sharadetermines the coefficients D_nو C_n, B_n, A_n.

6. Applying P to the Boundary ofDe-Electrics

In the boundaries between the electrons, two conditions must be met:

- Yale Potential Continuity: $V_1 = V_2$
- Continuity of the condensation of the electric fluxof one (D): $\varepsilon_1 \frac{\partial V_1}{\partial n} = \varepsilon_2 \frac{\partial V_2}{\partial n}$

These conditions should be applied to different aspects in order to determine the final coefficients.

Results

The exact solution of this problem does not require the use of sinusoidal and exponentials in each area, and by applying the boundary conditions, the coefficients are calculated. But in general, the process of solving it is complex and it is better to move on to numerical answers.

Part 3: Numerical Solution

To solve the Laplace equation according to the geometry and boundary conditions of the problem, we choose the finite differnece method. This method is simple and powerful. In the following, we will proceed by step:

1. Laplace's equation and discrete

Laplace's equation in two dimensions is in the form of equation 1-2. To solve the numbery, we first divide the space of the problem into a rectangular grid. Supposethe distance of the network nodes in both directions is h. Then, using the centrifugal differential approximation for second-order derivatives, we have:

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(i+1,j) - 2V(i,j) + V(i-1,j)}{h^2}$$

$$\frac{\partial^2 V}{\partial y^2} \approx \frac{V(i,j+1) - 2V(i,j) + V(i,j-1)}{h^2}$$

Placement in the Laplace equation:

$$\frac{V(i+1,j) - 2V(i,j) + V(i-1,j)}{h^2} + \frac{V(i,j+1) - 2V(i,j) + V(i,j-1)}{h^2} = 0$$

Simplification:

$$V(i,j) = \frac{1}{4}\big(V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j-1)\big)$$

2. Geometry gridding

2.1. Network Definition

The geometry of the problem is defined within$x \in [0,2]$و $y \in [0,2]$the scope of the problem. We divide the network into N×N nodes, where the distance betweenthese nodes is equal$h = \frac{2}{N-1}$.

2.2. Boundary Conditions

- On green borders: V = 0

- On purple borders: V = 1
- At the B-Boundary of Electrons: The Continuity of the Cell Potent and the Flux Densityof One

3. Numerical method (Gauss-Seidl repetition)

3.1. The first initializationof a

First, we determine the initial values of V(i,j) for all points in the network. For example:

- At points inside the figure: V(i,j)=0
- In boundaries: Boundary values are applied.

3.2. Repetition

For each inner node (not the Y-boundary), calculate the value of the iodine value V(i,j) using the finite differential relationship:

$$V_{\text{new}}(i,j) = \frac{1}{4}\big(V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j-1)\big)$$

This process continues for all internal nodes until the V(i,j) interval between two consecutive steps is less than one threshold () ε value.

4. Applying the Boundary ofDe-Electric

4.1. Continuity of Yell's Potential

For the nodes on the dielectric boundary, the value of the V potential will be the same:

$$V_{left} = V_{right}$$

4.2. Flux continuity

The density of flux () $D_n = \varepsilon \frac{\partial V}{\partial n}$is applied to the electric boundary:

$$\varepsilon_1 \frac{\partial V_1}{\partial n} = \varepsilon_2 \frac{\partial V_2}{\partial n}$$

In the method of finite differential, it is reduced in the form of a finite difference:

$$\varepsilon_1 \frac{V_{\text{سمت چپ}} - V_{\text{مرز}}}{h} = \varepsilon_2 \frac{V_{\text{مرز}} - V_{\text{سمت راست}}}{h}$$

5. Calculationof electric field

After calculating the distribution of V(i,j), the electric field at any point in the network is obtained using the following equations:

$$E_x = -\frac{V(i+1,j) - V(i-1,j)}{2h}, \quad E_y = -\frac{V(i,j+1) - V(i,j-1)}{2h}$$

6. Numerical execution process

To implement, we should implement these steps in the form of an orphan template:

1. Geometry Networking
2. The first componentis a
3. Repeating the Gauss-Seidl method:
   - Update V(i,j)
   - Convergence Survey
4. Applying the terms of the border and the necessity in the electric
5. Calculationof electrical field

Part 4: Python Code

According to the form of the problem, we have 6 boundary conditions. First of all, let's define them:

Then, we define a function to define boundary conditions and apply them:

Boundary conditions

```
[40] def apply_boundary_conditions(V, grid_size):

        # Boundary 1: V = 0 along x = 0 for y = [0, 2]
        V[:, 0] = 0

        # Boundary 2: V = 0 along y = 0 for x = [0, 2]
        V[0, :] = 0

        # Boundary 3: V = 1 along x = [1, 2] and y = 1
        start_x = int(grid_size / 2)
        end_x = grid_size - 1
        y = int(grid_size / 2)
        V[y, start_x:end_x] = 1

        # Boundary 4: V = 1 along x = 1 and y = [1, 2]
        x = int(grid_size / 2)
        start_y = int(grid_size / 2)
        end_y = grid_size - 1
        V[start_y:end_y, x] = 1

        # Boundary 5: V varies linearly from 0 to 1 for x = [0, 1] at y = 2
        x_vals = np.linspace(0, 1, start_x + 1)
        y = grid_size - 1
        V[y, :start_x + 1] = x_vals

        # Boundary 6: V varies linearly from 0 to 1 for y = [0, 1] at x = 2
        y_vals = np.linspace(0, 1, start_y + 1)
        x = grid_size - 1
        V[:start_y + 1, x] = y_vals

        return V
```

To solve the Laplace equation and obtain the energy and potential for the different values of the grid, we write a general for loop as follows.

```
# Loop over different grid sizes
for grid_size in grid_sizes:
    dx = L / (grid_size - 1)  # Grid spacing

    # Initialize potential grid
    V = np.zeros((grid_size, grid_size))

    # Apply boundary conditions
    V = apply_boundary_conditions(V, grid_size)

    # Solve Laplace's equation iteratively
    max_iterations = 10000
    tolerance = 1e-6

    start_time = time.time()
    for iteration in range(max_iterations):
        V_old = V.copy()

        # Update the potential using the finite difference method
        V[1:-1, 1:-1] = 0.25 * (
            V_old[1:-1, :-2] + V_old[1:-1, 2:] + V_old[:-2, 1:-1] + V_old[2:, 1:-1]
        )

        # Reapply the boundary conditions to ensure they are maintained
        V = apply_boundary_conditions(V, grid_size)

        # Check for convergence
        if np.max(np.abs(V - V_old)) < tolerance:
            elapsed_time = time.time() - start_time
            print(f"Grid Size: {grid_size}x{grid_size}, Converged after {iteration} iterations in {elapsed_time:.2f} seconds.")
            break
    else:
```

```
        print(f"Grid Size: {grid_size}x{grid_size}, Did not converge within the maximum number of iterations.")

    # Mask the NAAN regions (x > 1 when y = [1:2] and y > 1 when x = [1:2])
    mask = np.zeros_like(V, dtype=bool)
    mask[int(grid_size / 2):, int(grid_size / 2):] = True
    V_masked = np.ma.masked_where(mask, V)

    # Compute the electric field
    Ex, Ey = np.gradient(-V, dx)
    Ex_masked = np.ma.masked_where(mask, Ex)
    Ey_masked = np.ma.masked_where(mask, Ey)

    # Plot the potential
    plt.figure(figsize=(8, 6))
    plt.contourf(
        np.linspace(0, L, grid_size),
        np.linspace(0, L, grid_size),
        V_masked, 50, cmap="viridis", extend="both"
    )
    plt.colorbar(label="Potential (V)")
    plt.title(f"Electric Potential\nGrid Size: {grid_size}x{grid_size}, Convergence: {iteration} iterations\nTime: {elapsed_time:.2f} seconds")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()

    # Plot the electric field
    plt.figure(figsize=(8, 6))
    plt.quiver(
        np.linspace(0, L, grid_size),
        np.linspace(0, L, grid_size),
        Ex_masked, Ey_masked,
        scale=20,
        color="r",
    )
    plt.title(f"Electric Field\nGrid Size: {grid_size}x{grid_size}, Convergence: {iteration} iterations\nTime: {elapsed_time:.2f} seconds")
```

```
plt.title(f"Electric Field\nGrid Size: {grid_size}x{grid_size}, Convergence: {iteration} iterations\nTime: {elapsed_time:.2f} seconds")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

In the first part of this code, it solves the Laplace equation, then updates its value each time, after calculating the potential. Finally, it calculates the electric field and plots it.

Section 4: Answers to the Questions

4-1: How is the potential and electric field inside the high-figure space?

For grid size = 50, the potential and electric field are as follows:

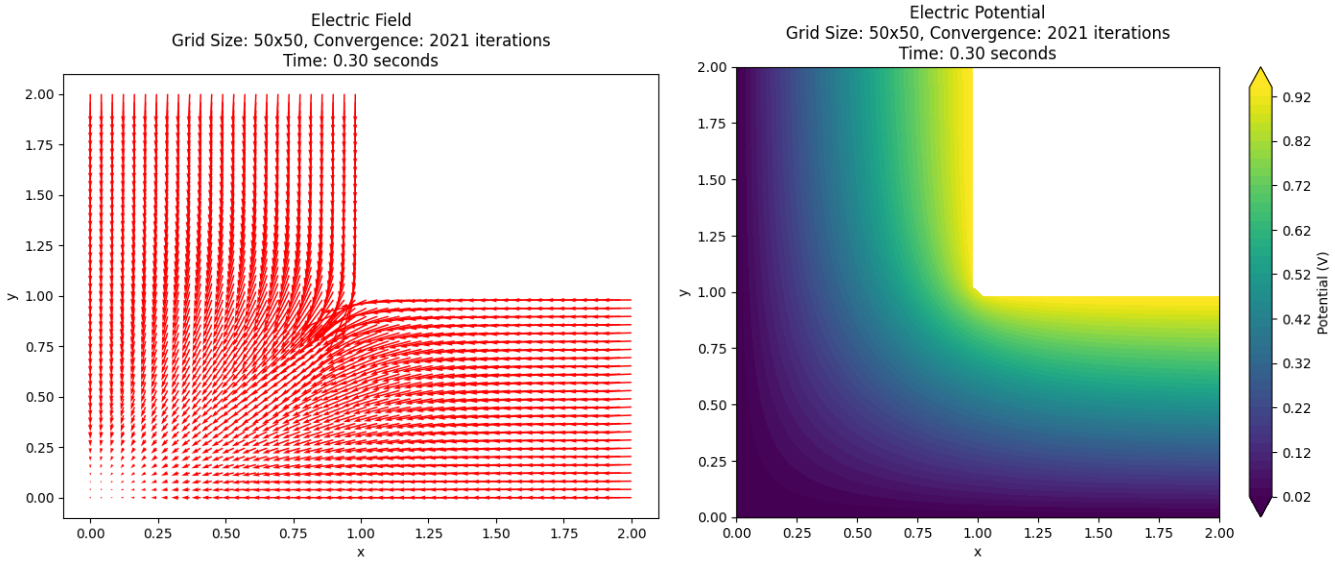
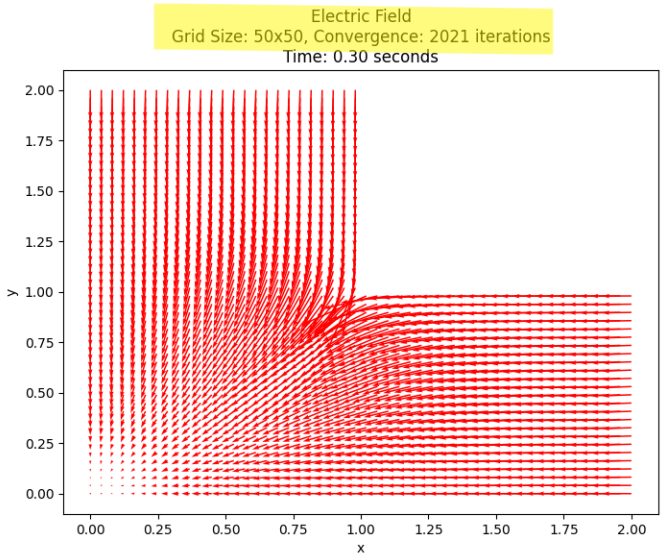
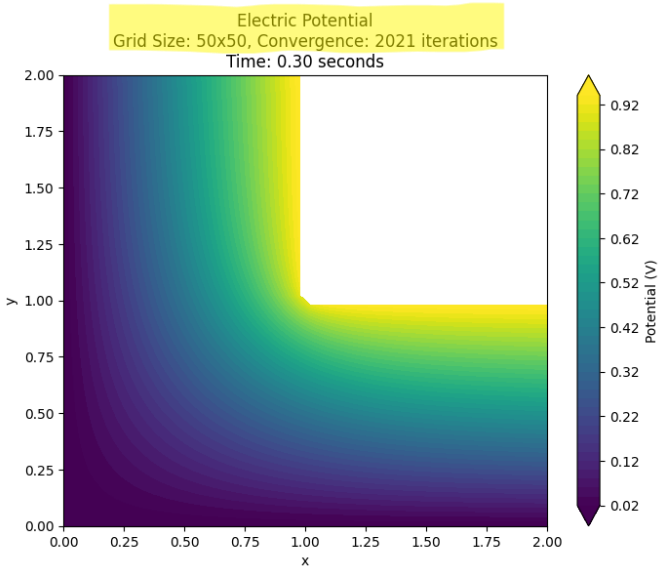
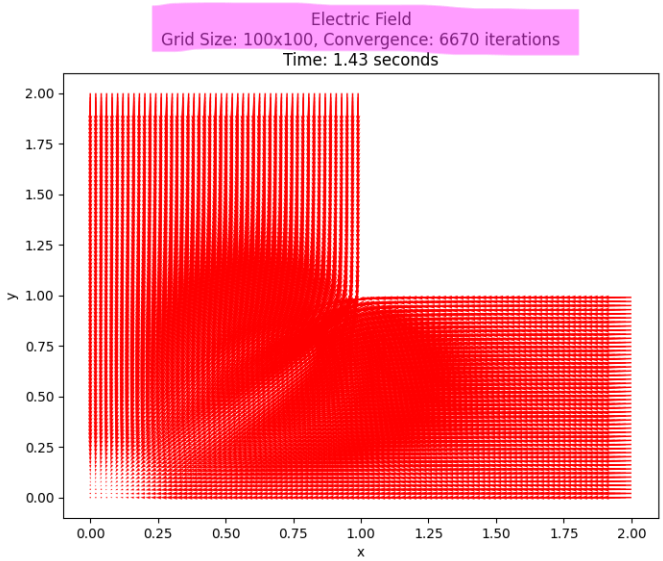


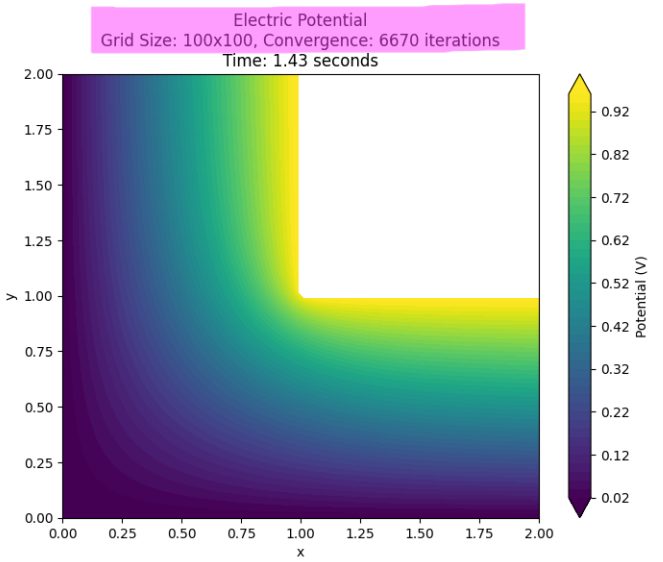
Figure 4-1) Potential and electric field diagram for grid size=50

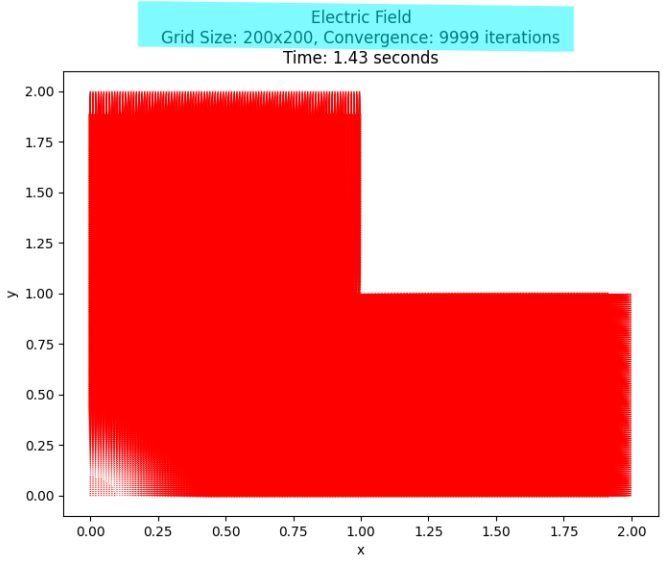
4-2: Use a mesh size which is small enough. Also, check the effect of changing the size of the meshes.

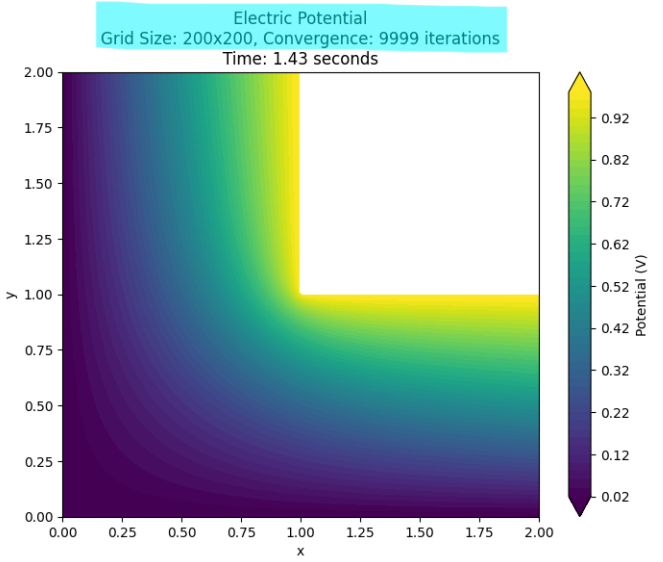












4-3: The effect of number of grid points on convergence time.

The effect of the number of network points on convergence time shows that with increasing the number of network points, the number of iterations required for convergence increases significantly. Specifically:

- **50×50 Network**
  - Electric's Potential: Convergence After 2021 Repeats in 0.30 Seconds
  - Electric-One Field: Convergence After 2021 Repeat in 0.30 Seconds
- **100×100 Network**
  - ElectronPotential One: Convergence after 6670 repetitions in 1.43 seconds
  - Electron FieldOne: Convergence after 6670 repetitions in 1.43 seconds
- **200×200 Network**
  - Electron Potential One: Convergence after 9999 repetitions in 1.43 seconds
  - Electron FieldOne: Convergence after 9999 repetitions in 1.43 seconds

The results show that by increasing the network size from 50×50 to 100×100, the number of iterations required for convergence increases significantly (from 2021 to 6670) and the convergence time increases (from 0.30). seconds to 1.43 seconds). As the network size is further increased to 200×200, the number of iterations required increases again (9,999), but the convergence time remains the same as the network of 100×100 (1.43 seconds). Thissuggests that although the number of iterations required for convergence increases with increasing network size, the convergence time does not increase proportionately and may be due to the computational limitations of the simulations used in the orphan algorithm. remain steady.

Section 5: Conclusion

Inthis exercise, the effectof the number of network points on the convergence time was investigated. The results showed that with increasing the size of the network, the number of iterations required for convergence increases. However, the convergence time did not remain constant due to the increase in the number of network points, and the convergence time may reach a constant value due to computational limitations and algorithm configurations. This issue highlights the importance of choosing the appropriate network size for solving Laplace equations andthe representation of the electric fieldwith appropriate accuracy and efficiency. Such analyses can be effective in

improving computational methods and achieving more accurate and faster results in physical and engineering problems.