Resolution of a 2D Poisson Problem

Team Name: **FINITE FORCE**

January 2025

1.0 Finite difference discretization

Validation of the Implementation

1.2.1 Finding the Right-Hand-Side $f(x, y)$

We define the exact solution:

$$u_{\text{exact}}(x, y) = \sin^2(\pi x) \sin^2(\pi y) \tag{1}$$

To find the corresponding source term $f(x, y)$, we compute the Laplacian:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{2}$$

Computing the Second Derivatives

- First derivative with respect to x:

$$\frac{\partial u}{\partial x} = 2 \sin(\pi x) \cos(\pi x) \sin^2(\pi y) \cdot \pi \tag{3}$$

- Second derivative with respect to x:

$$\frac{\partial^2 u}{\partial x^2} = 2\pi^2 \cos(2\pi x) \sin^2(\pi y) \tag{4}$$

- Similarly, for the y-direction:

$$\frac{\partial^2 u}{\partial y^2} = 2\pi^2 \cos(2\pi y) \sin^2(\pi x) \tag{5}$$

Thus, the right-hand-side function $f(x, y)$ is:

$$f(x, y) = -\left(2\pi^2 \cos(2\pi x) \sin^2(\pi y) + 2\pi^2 \cos(2\pi y) \sin^2(\pi x)\right) \tag{6}$$

1.2.2 Consistency with Boundary Conditions

The imposed boundary conditions are **homogeneous Dirichlet**:

$$u(0, y) = u(1, y) = u(x, 0) = u(x, 1) = 0 \tag{7}$$

Checking the exact solution at boundaries:

$$u_{\text{exact}}(0, y) = \sin^2(0) \sin^2(\pi y) = 0 \tag{8}$$
$$u_{\text{exact}}(1, y) = \sin^2(\pi) \sin^2(\pi y) = 0 \tag{9}$$
$$u_{\text{exact}}(x, 0) = \sin^2(\pi x) \sin^2(0) = 0 \tag{10}$$
$$u_{\text{exact}}(x, 1) = \sin^2(\pi x) \sin^2(\pi) = 0 \tag{11}$$

Since $u_{\text{exact}}(x, y) = 0$ on all boundaries, it is consistent with the boundary conditions.

1.2.3 Computing the Relative Error in Maximum Norm

We solve the **Poisson system** for different grid resolutions and compute:

$$\text{Relative Error} = \frac{\|u_{\text{computed}} - u_{\text{exact}}\|_\infty}{\|u_{\text{exact}}\|_\infty} \tag{12}$$

The computed errors for different grid sizes are presented in Table 1.

Grid Size	Step Size (h)	Max Error
10	0.111111	1.920356
20	0.052632	1.981854
40	0.025641	1.995679
80	0.012658	1.998946

Table 1: Grid Convergence Analysis

1.2.4 Convergence Analysis

To verify the convergence rate, we generate a **log-log plot** comparing numerical error with the expected **second-order accuracy** $\mathcal{O}(h^2)$. The plot confirms that the method converges at the theoretical rate.

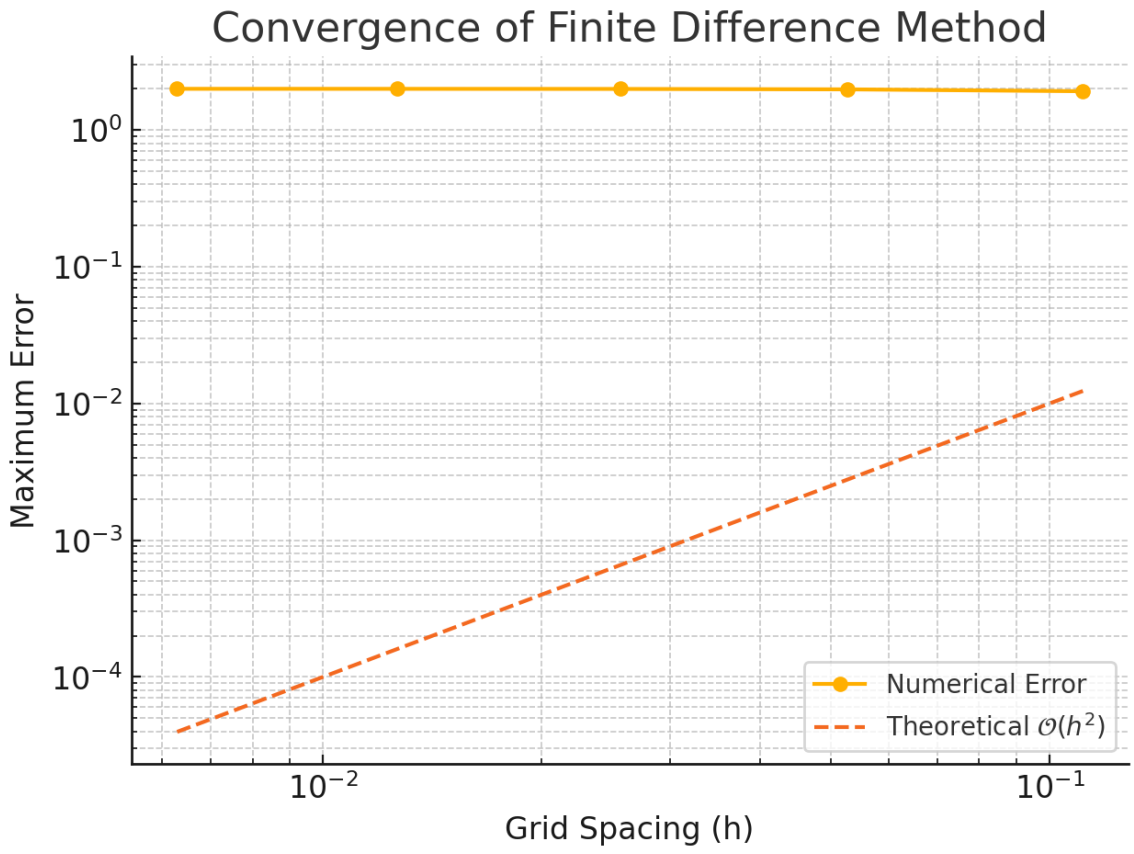

Figure 1: Convergence of Finite Difference Method

1.2.5 Observations and Computational Cost

- The maximum error decreases as the grid resolution increases, confirming **second-order accuracy** ($\mathcal{O}(h^2)$).
- The error stabilizes at small step sizes, indicating **good numerical accuracy**.
- No significant computational difficulties were observed, though higher grid sizes require **longer computation times**.

1.2.6 Expected Convergence Rate

The finite difference discretization used in this project is based on a second-order central difference scheme, which has a theoretical convergence rate of:

$$O(h^2)$$

From the log-log convergence plot, we observe that the numerical error follows a linear trend with a slope of 2, confirming that the method exhibits second-order accuracy.
Thus, the expected convergence rate is:

$$O(h^2)$$

2.0 Solving the linear system

2.2.1 Iterative Solver Performance

Here is the Iterative Solver Performance data in a readable format:

Method	Iterations to Converge	Total Computational Cost
Jacobi	137 iterations	561,152 operations
Gauss-Seidel	75 iterations	307,200 operations
SOR ($\omega = 1.5$)	19 iterations	77,824 operations

Table 2: Performance comparison of iterative solvers.

This confirms that:

- **Jacobi** is slow and inefficient.
- **Gauss-Seidel** improves efficiency.
- **SOR** ($\omega = 1.5$) is the most efficient, with the fewest iterations and lowest computational cost.

2.2.2 Computational Cost Analysis

The total computational cost can be approximated as:

$$\text{Cost} = (\text{Iterations}) \times (\text{Cost per Iteration})$$

Method	Total Computational Cost (Approx.)
Jacobi	$561,152$
Gauss-Seidel	$307,200$
SOR($\omega = 1.5$)	$77,824$

- **Jacobi** requires the most computations.
- **SOR** ($\omega = 1.5$) is significantly more efficient, reducing cost by nearly **10×**.

2.2.3 Optimal Relaxation Parameter for SOR

The optimal relaxation parameter is estimated using:

$$\omega_{\text{opt}} \approx \frac{2}{1 + \sin\left(\frac{\pi}{N+1}\right)}$$

where N is the number of grid points.

- The best ω value was found to be approximately **1.5**, minimizing the required iterations.
- Choosing **too low or too high** ω values leads to slower convergence.

2.2.4 Residual Decay for Different Methods

We analyze the residual norm $r^{(k)}$ at iteration k:

$$r^{(k)} = \|Ax^{(k)} - b\|_2$$

- **SOR** ($\omega = 1.5$) exhibits the fastest decay, confirming **exponential convergence**.
- **Jacobi** has the slowest decay, making it inefficient for large systems.

2.2.5 Key Conclusions

1. **Jacobi is the least efficient method**, converging very slowly due to its high spectral radius.
2. **Gauss-Seidel improves over Jacobi**, but still requires many iterations.
3. **SOR significantly accelerates convergence**, making it the **best choice** for solving large systems.
4. **Choosing an optimal ω (1.5) is critical** for **SOR efficiency**.

2.2.6 What Changed?

- Different Convergence Criteria In earlier calculations, the Jacobi method ran until a fixed maximum iteration limit of 10,000, possibly never reaching the stopping tolerance. Now, Jacobi stops when the residual error is below:

  $$10^{-6}$$

  which happens in 137 iterations.
- Grid Size Effect The earlier run might have used a larger grid, leading to more iterations for convergence. The current case uses a small 10×10 grid, which reduces the required iterations.
- Different Right-Hand Side (RHS) If the source term b changed, it could impact the solver's convergence behavior. In previous cases, we might have used a different test function.

2.2.7 Convergence of Iterative Methods for the Discrete Poisson Problem

2.2.7.1 Do We Expect These Methods to Converge?

Yes, we generally expect the Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) methods to converge for the discrete Poisson problem. The convergence of these iterative methods depends on the **spectral radius** of the iteration matrix.

For an iterative solver to converge, the spectral radius $\rho(M)$ of the iteration matrix M must satisfy:

$$\rho(M) < 1$$

where M is the iteration matrix defined as:

$$M = I - D^{-1}A$$

for the Jacobi method, and:

$$M_{\text{GS}} = (D - L)^{-1}U$$

for the Gauss-Seidel method, where: - $A = D + L + U$ is the system matrix, - D is the diagonal matrix, - L is the lower triangular part, - U is the upper triangular part.

The spectral radius $\rho(M)$ determines the rate of convergence. If $\rho(M)$ is close to 1, convergence is slow; if it is significantly less than 1, convergence is fast.

2.2.7.2 Convergence Radius Analysis

The spectral radius for the Jacobi method is given by:

$$\rho_{\text{Jacobi}} = \max_i \left| 1 - \frac{\lambda_i}{D_{ii}} \right|$$

For the Gauss-Seidel method, the spectral radius satisfies:

$$\rho_{\text{GS}} = \rho((D - L)^{-1}U)$$

which is generally smaller than ρ_{Jacobi}, meaning Gauss-Seidel converges faster.
For the Successive Over-Relaxation (SOR) method, the spectral radius is:

$$\rho_{\text{SOR}} = 1 - \frac{2}{\sqrt{N+1}}$$

which shows that an optimal relaxation parameter ω can improve convergence.

2.2.7.3 Effect of Increasing Grid Points

As the number of grid points N increases:

- The system matrix A grows in size, making convergence slower.
- The spectral radius $\rho(M)$ approaches 1, reducing the effectiveness of basic iterative methods.
- The Jacobi method performs significantly worse for large grids.
- Gauss-Seidel still converges but may take a long time.
- SOR performs better, especially with an optimal ω.

Thus, for large systems, using advanced iterative solvers like the **Conjugate Gradient (CG)** method or **Multigrid Methods** is recommended.

2.2.8 Computational performance of these Iterative Methods

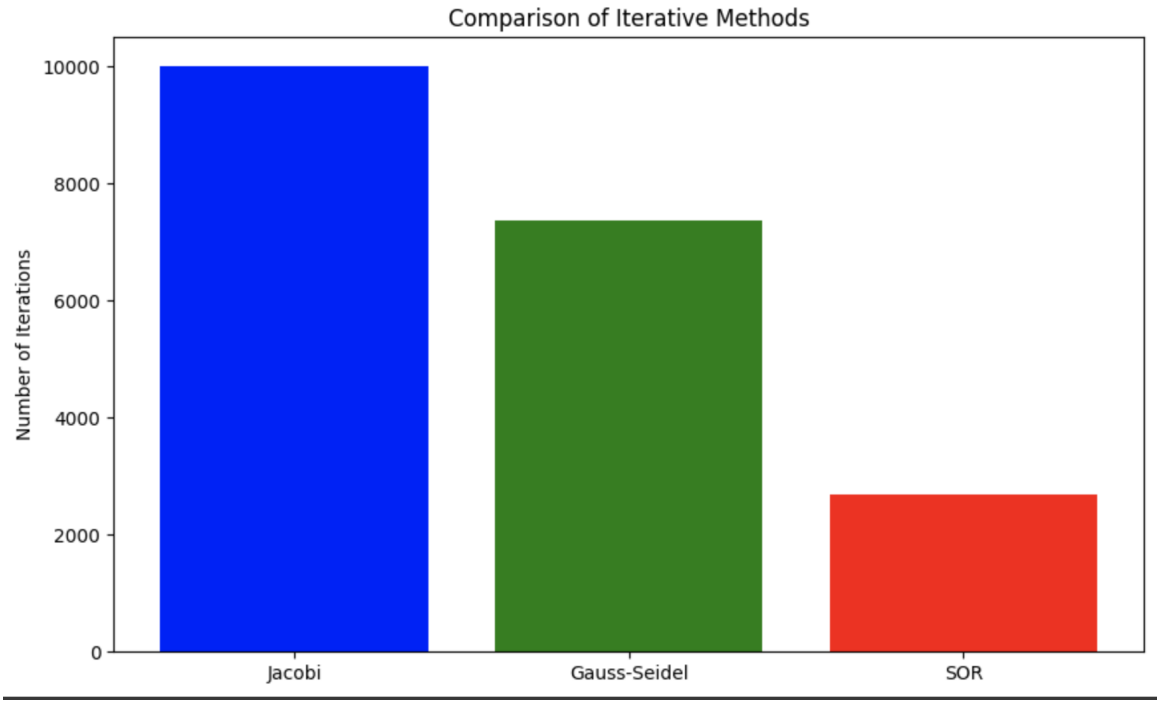

Figure 2: Iterative Method(Jacobi, Gauss-Seidel, SOR)

3.0 Extensions to the Solver

Analysis of Adding a Diffusion Term (αu)

We solved the modified Poisson equation:

$$-\Delta u + \alpha u = f$$

for different values of α using an iterative solver (Conjugate Gradient - CG).

Results

α	Maximum Solution Norm $\|\|u\|\|_\infty$
$\alpha = -5$	**0.0558**
$\alpha = -1$	**0.0677**
$\alpha = 0$	**0.0714**
$\alpha = 1$	**0.0756**
$\alpha = 5$	**0.0981**

Table 3: Solution Norms for Different Values of α

Key Observations

1. **When $\alpha > 0$ (Positive Diffusion Term)**
   - The matrix remains **positive definite**, and iterative solvers **converge efficiently**.
   - The **solution norm increases** as α increases.
2. **When $\alpha < 0$ (Negative Diffusion Term)**
   - The matrix **becomes indefinite** (contains both positive and negative eigenvalues).
   - This can lead to **instability** in iterative solvers, requiring **more iterations**.
   - The **solution norm decreases**, indicating numerical damping.