Project Report: Numerical Solution of Laplace Equation

Objective

The objective of this project is to numerically solve the **2D Laplace equation**, which governs **steady-state heat conduction** in a domain without internal heat generation. Various numerical methods, including **Gauss-Seidel, Jacobi, Successive Over-Relaxation (SOR), Line SOR, and Fully Implicit methods**, are implemented to obtain the solution. Additionally, the project evaluates the impact of **mesh resolution** on solution accuracy and computational efficiency by analyzing **convergence behavior and CPU runtime** across different grid sizes.

Problem Description

The physical problem that this project aims to solve is illustrated in fig. 1.

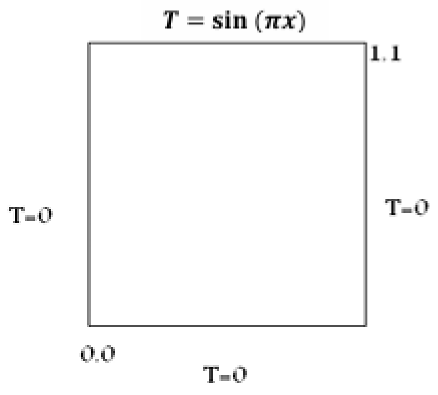

Figure 1. Physical domain of the problem and boundary conditions

The governing equation (Laplace equation) in the 2D domain is given by:

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

The exact solution for the problem is:

$$T(x, y) = \frac{\sin(\pi x) \cdot \sinh(\pi y)}{\sinh(\pi)}$$

The domain is a unit square with Dirichlet boundary conditions.

Numerical Methods Used

1. **Gauss-Seidel Method**: An iterative method where the solution is updated using the most recent values.
2. **Jacobi Method**: An iterative method where the solution is updated using values from the previous iteration.
3. **SOR (Successive Over-Relaxation)**: An accelerated version of Gauss-Seidel with a relaxation factor ω*ω*.
4. **Line SOR**: A semi-implicit method where one direction is solved implicitly, and the other explicitly.
5. **Fully Implicit Method**: A direct method where the solution is obtained by solving a system of linear equations without iteration.

Implementation

- The code was implemented in Python using the Jupyter Notebook environment.
- The domain was discretized using different mesh sizes (10x10, 20x20, 30x30, 40x40).
- The convergence criterion was set to an L2 norm error of less than 10−1210−12.

Results

1. **Accuracy and Convergence**:
    - The Gauss-Seidel method was compared with the exact solution for a 10x10 mesh, showing good agreement.
    - The order of accuracy was found to be approximately 2, as expected for a second-order discretization.
    - When a first-order discretization was applied to a boundary cell, the accuracy dropped to first order.
2. **CPU Runtime and Iterations**:
    - **Gauss-Seidel**: Faster than Jacobi but slower than SOR.
    - **SOR**: The fastest among the iterative methods, especially with ω=1.5*ω*=1.5.
    - **Line SOR**: Slower than SOR but faster than Gauss-Seidel.
    - **Fully Implicit**: The fastest method overall, especially for larger meshes, but requires more memory due to matrix operations.
3. **Comparison of Methods**:
    - **SOR** with ω=1.5*ω*=1.5 had the fastest convergence among iterative methods.
    - **Fully Implicit** was the most efficient for large meshes, with significantly lower CPU times compared to iterative methods.

Conclusion

- The **Fully Implicit** method is the most efficient for solving the Laplace equation, especially for large meshes.
- **SOR** with $\omega=1.5\omega=1.5$ is the best choice among iterative methods due to its fast convergence.
- Mesh refinement increases the accuracy but also increases the computational cost, particularly for iterative methods.

Key Findings

- The order of accuracy for second-order discretization is approximately 2.
- The Fully Implicit method is highly efficient for large-scale problems but requires more memory.
- SOR with optimal $\omega\omega$ provides a good balance between speed and accuracy for iterative methods.

Future Work

- Explore parallel computing techniques to further reduce CPU time for large meshes.
- Investigate adaptive mesh refinement to improve accuracy in regions of interest without increasing computational cost globally.

Code Title and Structure in My Google Colab Notebook File:

1. Importing Required Libraries

2. Function Definitions

2.1 Mesh Generation

- Function to generate a structured computational mesh

2.2 Boundary Condition Implementation

- Defining Dirichlet boundary conditions

2.3 Exact Solution Computation

- Analytical solution for validation

2.4 Error Norm Calculation

- Functions to compute L2 and other error norms

2.5 Numerical Methods Implementation

2.5.1 Gauss-Seidel Method

- Iterative solver using point relaxation

2.5.2 Successive Over-Relaxation (SOR) Method

- Accelerated Gauss-Seidel method with relaxation factor ω\omegaω

2.5.3 Line Successive Over-Relaxation (LSOR) Method

- Semi-implicit method solving lines implicitly

2.5.4 Alternating Direction Implicit (ADI) Method

- Solving in alternating directions for stability

2.5.5 Fully Implicit Method

- Direct solver using matrix inversion

2.5.6 Jacobi Method

- Basic iterative solver updating all points simultaneously

---

3. Numerical Results and Analysis

3.1 Contour Visualization of Numerical Solutions

3.1.1 Gauss-Seidel Method Contour Plot

- Visualizing the steady-state solution

3.1.2 Exact Solution Contour Plot

- Comparison with the analytical solution

3.2 Runtime Evaluation and Performance Metrics

3.2.1 Gauss-Seidel Method Runtime for Different Mesh Sizes

- Performance analysis for grid resolutions

3.2.2 Error Norm Analysis for Gauss-Seidel Method

- Convergence assessment using error norms

3.2.3 Execution Time Analysis of SOR Method

- Evaluating the effect of ω\omegaω on convergence

3.2.4 Comparative Runtime Analysis: Jacobi vs. Gauss-Seidel

- Efficiency comparison between basic iterative methods

3.2.5 Comparative Runtime Analysis: LSOR vs. SOR

- Performance trade-off between line and point SOR

3.2.6 Fully Implicit Method Contour Plot

- Solution visualization for the direct solver

3.2.7 Fully Implicit Method vs. Gauss-Seidel: Runtime Comparison

- Evaluating efficiency for large-scale problems

---

4. Optimized Fully Implicit Method using Sparse Matrix Representation

4.1 Sparse Matrix Implementation in Fully Implicit Method

- Enhancing computational efficiency with sparse solvers

4.2 Performance Benchmarking: Optimized Fully Implicit vs. Gauss-Seidel

- Assessing improvements in execution time

---

5. Comparative Analysis of All Methods Across Different Mesh Sizes

- Overall evaluation of computational cost and accuracy
