HUST

Scientific Computing Report

Lecturer: Dr. Vu Van Thieu

Class ID: 147831

Successive Over Relaxation Method on applying for 2D-Steady heat distribution

Members:
Le Dai Lam - 20225982
Nguyen Minh Khoi - 20226050
Dinh Nguyen Son - 20225997
Vu Hai Dang - 20225962
Bui Hoang Viet - 20226073

Contents

Abstract

This study focuses on the application of the Successive Over-Relaxation (SOR) method for solving the 2D-Steady heat distribution problem. We compare the performance of three established algorithms—Jacobi, Gauss-Seidel, and SOR 4-point—with an additional 9-point SOR algorithm that we have derived and implemented. Through a series of rigorous experiments, we evaluate these methods under various conditions, including different relaxation factors, grid dimensions, and convergence speeds. Our results demonstrate the superior performance of the SOR methods, particularly the 4-point and 9-point variations, in terms of both convergence rate and computational efficiency. By providing a comprehensive analysis of these algorithms, we highlight the advantages of SOR over traditional methods, making a compelling case for its adoption in computational heat transfer problems. Our findings offer valuable insights into the practical application and optimization of iterative solvers in engineering simulations.

1 Introduction

1.1 Introduction to 2D-Steady Heat Distribution problem

The 2D-Steady Heat Distribution problem is a fundamental issue in the field of heat transfer and thermodynamics, with significant applications in engineering and physical sciences. The problem involves determining the temperature distribution within a given domain over time until a steady state is reached, where the temperature at any point in the domain no longer changes. Historically, the study of heat distribution began with Joseph Fourier's pioneering work in the early 19th century, which led to the formulation of Fourier's law of heat conduction. Fourier's work laid the foundation for understanding heat transfer in solid materials, which has since evolved into a crucial area of research with wide-ranging applications.

Figure 1: French mathematician Jean Baptiste Joseph Fourier (1768 - 1830), who introduced the problem of 2D-steady heat distribution.

Applications of the 2D-Steady Heat Distribution problem are numerous, including the design of thermal systems in mechanical engineering, the analysis of heat dissipation in electronic components, and the thermal management of buildings. Accurate solutions to this problem are essential for optimizing energy efficiency and ensuring the structural integrity of materials subjected to thermal loads.

Traditional numerical methods like the Jacobi and Gauss-Seidel methods have been widely used to solve this problem. The Jacobi method updates the temperature grid iteratively, while the Gauss-Seidel method enhances convergence by using the latest available updates. Despite their utility, these methods often suffer from slow convergence, especially in large-scale problems. Similarly, the Successive Over-Relaxation (SOR) method was developed to address this issue. SOR improves upon the Gauss-Seidel method by introducing a relaxation factor, significantly speeding up convergence. In this study, we compare the performance of the Jacobi, Gauss-Seidel, SOR 4-point, and a newly implemented 9-point SOR method. Through various experiments analyzing relaxation factors, grid dimensions, and convergence rates, we demonstrate the superior efficiency and effectiveness of SOR methods in solving the 2D-Steady Heat Distribution problem.

1.2 Fourier's Law of Heat Conduction

Fourier's law describes how heat conduction occurs through a material:

$$\mathbf{q} = -k\nabla T$$

Where:

- $\mathbf{q}$ is the heat flux vector, representing the amount of heat flowing through a unit area per unit time.
- k is the thermal conductivity of the material.
- ∇T is the temperature gradient.

1.3 Principle of Energy Conservation

In the steady-state condition, the balance of heat at any point in the material can be described by the equation:

$$\nabla \cdot \mathbf{q} + Q = 0$$

where Q is a heat source or sink. In the absence of any heat sources or sinks, this simplifies to:

$$\nabla \cdot \mathbf{q} = 0$$

1.4 2D steady-state heat equation

- Combining Fourier's law with the heat balance equation, we get:

$$\nabla \cdot (-k\nabla T) = 0$$

- If the thermal conductivity k is constant (does not vary with space), the equation simplifies to:

$$k\nabla^2 T = 0$$

- Since k is a non-zero constant, we obtain:

$$\nabla^2 T = 0$$

- This is 2D steady-state heat equation and it is also the Laplace equation in two dimensions, which can be written as:

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

2 Ideas of Iterative method & Successive Over Relaxation Method

2.1 Ideas of Iterative method

- An initial solution is guessed and new values are computed.
- Based on the newly computed values, answer solution is sought.
- The procedure is repeated until a specified convergence criterion has been reached.
- The various formulations of the iterative method can be divided into two categories.

2.2 Jacobi Iteration Method

- The dependent variable at each grid point is solved (at the new iteration $k+1$ level), using initial guessed values or previously computed values (at the new iteration k level).
- The computation is carried out until a specified convergence criterion is met.
- We start with the 2D steady-state heat equation:

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

- Discretization this equation, we have:

$$T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + \left(\frac{\Delta x}{\Delta y}\right)^2 (T_{i,j-1} - 2T_{i,j} + T_{i,j+1}) = 0$$

- Next, we replace $\frac{\Delta x}{\Delta y} = \beta$ to achieve:

$$T_{i+1,j} + T_{i-1,j} + \beta^2 T_{i,j+1} + \beta^2 T_{i,j-1} - 2(1+\beta^2)T_{i,j} = 0$$

- Rearranging the terms of the equation, we get:

$$T_{i,j}^{k+1} = \frac{1}{2(1+\beta^2)} \left[T_{i+1,j}^k + T_{i-1,j}^k + \beta^2 (T_{i,j+1}^k + T_{i,j-1}^k)\right]$$

- Now let $\Delta x = \Delta y$, then $\beta = 1$ and the Jacobi equation reduces to:

$$T_{i,j}^{k+1} = \frac{1}{4} \left[T_{i+1,j}^k + T_{i-1,j}^k + T_{i,j+1}^k + T_{i,j-1}^k\right]$$

2.3 Gauss-Seidel Iteration Method

- It is an improve of the Jacobi method by using the newly computed values of the dependent variable
- Increases the convergence rate
- Lead to less computation time

$$T_{i,j}^{k+1} = \frac{1}{2(1+\beta^2)} \left[T_{i+1,j}^k + T_{i-1,j}^{k+1} + \beta^2 \left(T_{i,j+1}^k + T_{i,j-1}^{k+1}\right)\right]$$

- For the computation of the first point:

$$T_{2,2}^{k+1} = \frac{1}{2(1+\beta^2)} \left[T_{3,2}^k + T_{1,2} + \beta^2 \left(T_{2,3}^k + T_{2,1}\right)\right]$$

- For point (3,2):

$$T_{3,2}^{k+1} = \frac{1}{2(1+\beta^2)} \left[T_{4,2} + T_{2,2} + \beta^2 \left(T_{3,3} + T_{3,1}\right)\right]$$

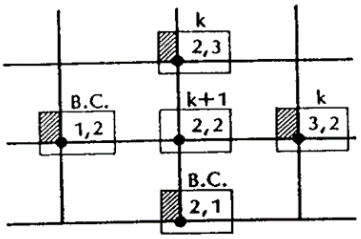

Figure 2: First point

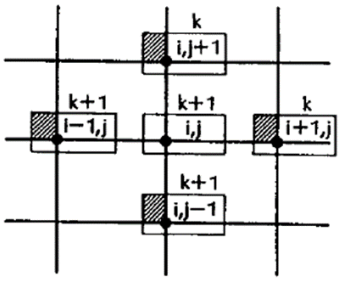

Figure 3: Normal point

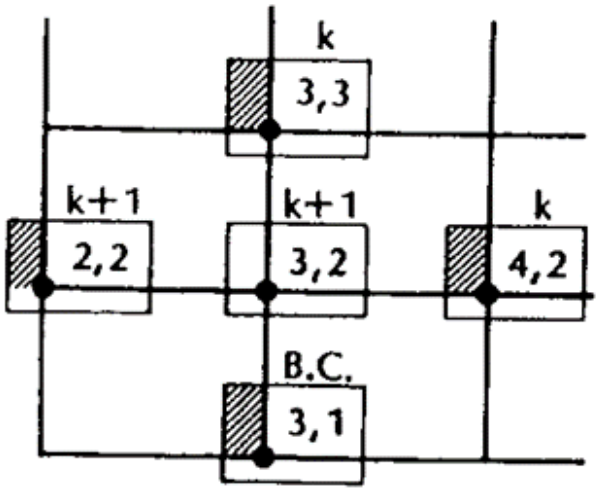

Figure 4: Point (3,2)

In this equation $T_{3,1}$ is provided by the boundary condition $T_{4,2}; T_{3,3}$ are taken from the previous computation. But $T_{2,2}$ is given by the previous equation. Thus we have:

$$T_{3,2}^{k+1} = \frac{1}{2(1+\beta^2)} \left[T_{4,2}^{k} + T_{2,2}^{k+1} + \beta^2 \left(T_{3,3}^{k} + T_{3,1} \right) \right]$$

Finally, the general formulation provides the equation

$$T_{i,j}^{k+1} = \frac{1}{2(1+\beta^2)} \left[T_{i+1,j}^{k} + T_{i-1,j}^{k+1} + \beta^2 \left(T_{i,j+1}^{k} + T_{i,j-1}^{k+1} \right) \right]$$

This is a point iteration method, since only one unknown is being required

2.4 Successive Over Relaxation Method

1. First, consider the point Gauss-Seidel iteration method:

$$T_{i,j}^{k+1} = \frac{1}{2(1+\beta^2)}\left[T_{i+1,j}^{k} + T_{i-1,j}^{k+1} + \beta^2\left(T_{i,j+1}^{k} + T_{i,j-1}^{k+1}\right)\right]$$

2. Adding to the right-hand side $T_{i,j}^{k} - T_{i,j}^{k+1}$ and collecting terms:

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \frac{1}{2(1+\beta^2)}\left[T_{i+1,j}^{k} + T_{i-1,j}^{k+1} + \beta^2\left(T_{i,j+1}^{k} + T_{i,j-1}^{k+1}\right) - 2(1+\beta^2)T_{i,j}^{k}\right]$$

3. To accelerate the solution the bracket term is multiplied by ω, the relaxation parameter:

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \frac{\omega}{2(1+\beta^2)}\left[T_{i+1,j}^{k} + T_{i-1,j}^{k+1} + \beta^2\left(T_{i,j+1}^{k} + T_{i,j-1}^{k+1}\right) - 2(1+\beta^2)T_{i,j}^{k}\right]$$

4. The formulation is rearranged into Successive Over Relaxation equation :

$$T_{i,j}^{k+1} = (1-\omega)T_{i,j}^{k} + \frac{\omega}{2(1+\beta^2)}\left[T_{i+1,j}^{k} + T_{i-1,j}^{k+1} + \beta^2\left(T_{i,j+1}^{k} + T_{i,j-1}^{k+1}\right)\right]$$

Theorems can show that $0 < \omega < 2$ is a condition for a scheme to converge:

- If we choose $0 < \omega < 1$, the solution is called under-relaxation, and the convergence will be slower.
- If $\omega = 1$, we just have the Gauss-Seidel iteration method.
- If we choose $1 < \omega < 2$, it is called successive over-relaxation.

For the solution of equations in rectangular domain subject with constant step sizes, we have:

$$\omega_{opt} = \frac{2 - 2\sqrt{1-a}}{a}$$

$$a = \left[\frac{\cos\left(\frac{\pi}{IM-1}\right) + \beta^2\cos\left(\frac{\pi}{JM-1}\right)}{1+\beta^2}\right]^2$$

- In general, ω_{opt} cannot be determined easily.
- Therefore, for most cases, numerical experiment is performed.

2.5 Successive Over Relaxation Method using nine points Laplacian

SOR method for the nine-point Laplacian:

$$T_{i,j}^{(k+1)} = \omega \left(\frac{T_{i-1,j}^{(k+1)} + T_{i+1,j}^{(k)} + T_{i,j-1}^{(k+1)} + T_{i,j+1}^{(k)}}{5} + \frac{T_{i-1,j+1}^{(k)} + T_{i+1,j+1}^{(k)} + T_{i-1,j-1}^{(k+1)} + T_{i+1,j-1}^{(k+1)}}{20} \right) + (1-\omega) T_{i,j}^{(k)}$$

This formula is used to approximate the Laplacian of a function $f(x, y)$ using finite differences with a higher order of accuracy by considering both adjacent and diagonal points around (x, y). The formula for the finite difference approximation of the Laplacian is given by:

$$\begin{aligned}\nabla^2 f(x,y) = \frac{1}{6h^2} [&4T(x-h,y) + 4T(x+h,y) + 4T(x,y-h) + 4T(x,y+h) \\ &+ T(x-h,y-h) + T(x+h,y-h) + T(x-h,y+h) \\ &+ T(x+h,y+h) - 20T(x,y)] = 0\end{aligned}$$

To prove this, we use the Taylor series expansions of the terms around (x, y): First, for the points $T(x \pm h, y)$:

$$T(x \pm h, y) = T(x,y) \pm h \frac{\partial T}{\partial x} + \frac{h^2}{2} \frac{\partial^2 T}{\partial x^2} + \mathcal{O}(h^3)$$

Similarly, for the points $T(x, y \pm h)$:

$$T(x, y \pm h) = T(x,y) \pm h \frac{\partial T}{\partial y} + \frac{h^2}{2} \frac{\partial^2 T}{\partial y^2} + \mathcal{O}(h^3)$$

And for the diagonal points $T(x \pm h, y \pm h)$:

$$\begin{aligned}T(x \pm h, y \pm h) = T(x,y) &\pm h \left(\frac{\partial T}{\partial x} + \frac{\partial T}{\partial y} \right) \\ &+ \frac{h^2}{2} \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial x \partial y} \right) + \mathcal{O}(h^3)\end{aligned}$$

Summing these expansions weighted as per the given formula:

$$\begin{aligned}4\left[T(x-h,y) + T(x+h,y) + T(x,y-h) + T(x,y+h)\right] = 16T(x,y) &+ 4h^2 \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \\ &+ \mathcal{O}(h^3)\end{aligned}$$

$$\begin{aligned}T(x-h,y-h) + T(x+h,y-h) + T(x-h,y+h) + T(x+h,y+h) = \\ 4T(x,y) + 2h^2 \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + \mathcal{O}(h^4)\end{aligned}$$

Combining these, we get:

$$\begin{aligned}\nabla^2 f(x,y) = \frac{1}{6h^2}\,[&4T(x-h,y)+4T(x+h,y)+4T(x,y-h)+4T(x,y+h)\\ &+T(x-h,y-h)+T(x+h,y-h)+T(x-h,y+h)]\\ &+T(x+h,y+h)-20T(x,y)] = 0\end{aligned}$$

Thus, we have shown that the given finite difference formula approximates the Laplacian. With $h = 1$ and we apply the same method as Gauss Seidel Method , we will have:

$$\begin{aligned}T_{i,j}^{(k+1)} =&\frac{T_{i-1,j}^{(k+1)}+T_{i+1,j}^{(k)}+T_{i,j-1}^{(k+1)}+T_{i,j+1}^{(k)}}{5}.\\ &+\frac{T_{i-1,j+1}^{(k)}+T_{i+1,j+1}^{(k)}+T_{i-1,j-1}^{(k+1)}+T_{i+1,j-1}^{(k+1)}}{20}\end{aligned}$$

SOR method for the nine-point Laplacian:

$$\begin{aligned}T_{i,j}^{(k+1)} = \omega\Bigg(&\frac{T_{i-1,j}^{(k+1)}+T_{i+1,j}^{(k)}+T_{i,j-1}^{(k+1)}+T_{i,j+1}^{(k)}}{5}\\ &+\frac{T_{i-1,j+1}^{(k)}+T_{i+1,j+1}^{(k)}+T_{i-1,j-1}^{(k+1)}+T_{i+1,j-1}^{(k+1)}}{20}\Bigg)\\ &+(1-\omega)T_{i,j}^{(k)}\end{aligned}$$

3 Implementation of solving problem in MatLab

3.1 Initialize part

```
l = 1.0;   % length of plate in x-direction
w = 1.0;   % length of plate in y-direction
nx = 200;
ny = nx;

x = linspace(0, l, nx); % Dividing Length in equal steps using user input
y = linspace(0, w, ny); % Dividing width in equal steps using user input

dx = l / nx;   % step size in x direction
dy = w / ny;   % step size in y direction

T_B = 400;
T_T = 200;
T_L = 300;
T_R = 300;

error_req = 1e-4;


% 4 point Initialization
T = ((T_B + T_T + T_L + T_R) * 0.25) * ones(nx, ny);  % Creating nx*ny
T(1, :) = T_B;           % Bottom side temperature
T(ny, :) = T_T;          % Top side temperature
T(2:end-1, 1) = T_L;     % Left side temperature
T(2:end-1, nx) = T_R;    % Right side temperature
T(1, 1) = (T_B + T_L) / 2;
T(nx, ny) = (T_R + T_T) / 2;
T(ny, 1) = (T_T + T_L) / 2;
T(1, nx) = (T_R + T_B) / 2;

errors = [];
error = 1e9;
k = 0;
```

Figure 5: Initialization of Successive Over Relaxation Method using 4 point

```
% 9 point Gauss-Seidel with SOR
T = ((T_B + T_T + T_L + T_R) * 0.25) * ones(nx, ny);
T(1, :) = T_B;           % Bottom side temperature
T(ny, :) = T_T;          % Top side temperature
T(2:end-1, 1) = T_L;     % Left side temperature
T(2:end-1, nx) = T_R;    % Right side temperature
T(1, 1) = (T_B + T_L) / 2;
T(nx, ny) = (T_R + T_T) / 2;
T(ny, 1) = (T_T + T_L) / 2;
T(1, nx) = (T_R + T_B) / 2;

omega = 1.97;  % Relaxation factor
errors3 = [];
k = 0;
error = 1e9;

tic;
```

Figure 6: Initialization of Successive Over Relaxation Method using Nine points

3.2 Iteration and Output Part

```
while error > error_req
    T0 = T;
    for i = 2:ny-1
        for j = 2:nx-1
            T(i, j) = 0.25 * (T0(i + 1, j) + T0(i - 1, j) + T0(i, j + 1) + T0(i, j - 1));
        end
    end
    error = max(max(abs(T0 - T)));
    errors = [errors, error];  % Store the error at each iteration
    k = k + 1;
end
F = toc;
Toutput = T;
fprintf("Jacobi Iterative Method\n");
fprintf('Computation time: %f\n',F);
fprintf("Number of iterations are: %d \n\n\n", k);

% Plot Jacobi
figure(1);
[M, N] = contourf(x, y, Toutput);
clabel(M, N);
colormap(jet);
colorbar;
title('Temperature Variation on Plate (Steady State) By Jacobi Iterative Method');
xlabel('X-axis');
ylabel('Y-axis');
```

Figure 7: Jacobi Iteration Method

```
        while error>error_req
            T0=T;
            for i=2:ny-1
                for j=2:nx-1
                    T(i,j)= 0.25 * (T0(i+1,j) + T(i-1,j) + T0(i,j+1) + T(i,j-1));
                end
            end
            error = max(max(abs(T0-T)));
            errors1 = [errors1, error];  % Store the error at each iteration
            k=k+1;
        end
F = toc;
Toutput2 = T;
fprintf("Gauss-Seidel Iterative Method\n");
fprintf('Computation time: %f\n',F);
fprintf("Number of iterations are: %d \n\n\n",k);

% Plot 2D Gauss-Seidel
figure(2);
[M, N] = contourf(x, y, Toutput2);
clabel(M, N);
colormap(jet);
colorbar;
title('Temperature Variation on Plate (Steady State) By Gauss-Seidel');
xlabel('X-axis');
ylabel('Y-axis');
```

Figure 8: Gauss-Seidel Iteration Method

```
while error > error_req
    T0 = T;
    for i = 2:ny-1
        for j = 2:nx-1
            T(i, j) = (1 - a) * T0(i, j) + a * 0.25 * (T(i, j - 1) + T0(i, j + 1) + T(i - 1, j) + T0(i + 1, j));
        end
    end
    error = max(max(abs(T0 - T)));
    errors2 = [errors2, error];
    k = k + 1;
end
F = toc;
Toutput3 = T;
fprintf("Gauss-Seidel Iterative Method with SOR\n");
fprintf('Computation time: %f\n',F);
fprintf("Number of iterations are: %d \n\n\n",k);

% Plot 2D Gauss-Seidel with SOR
figure(3);
[M, N] = contourf(x, y, Toutput3);
clabel(M, N);
colormap(jet);
colorbar;
title('Temperature Variation on Plate (Steady State) By Gauss-Seidel with SOR');
xlabel('X-axis');
ylabel('Y-axis');
```

Figure 9: Successive Over Relaxation Method

```
while (error > error_req)
    Told = T;
    for i = 2:nx-1
        for j = 2:ny-1
            T(i, j) = (omega / 5) * (T(i-1, j) + Told(i+1, j) + T(i, j-1) + Told(i, j+1)) + ...
                      (omega / 20) * (T(i-1, j-1) + Told(i+1, j-1) + T(i-1, j+1) + Told(i+1, j+1)) + ...
                      (1 - omega) * Told(i, j);
        end
    end
    error = max(max(abs(Told - T)));
    errors3 = [errors3; error];
    k = k + 1;
end

F = toc;
Toutput1 = T;
fprintf("Gauss-Seidel Iterative Method with SOR 9 point\n");
fprintf('Computation time: %f\n',F);
fprintf("Number of iterations are: %d \n\n\n",k);
```

Figure 10: Successive Over Relaxation Method using Nine points Laplacian

4 Comparison

In this section of the study, we embark on a comparative analysis of four distinct iterative algorithms: Jacobi, Gauss-Seidel, SOR-Gauss 4-point, and SOR-Gauss 9-point. These methods were implemented in MATLAB to address the problem of Steady 2D heat conduction. The objective of this comparison is to evaluate each algorithm's effectiveness in terms of complexity, convergence rate, computational time, and loss reduction across epochs.

To facilitate a thorough assessment, we initialized the algorithms with three different sets of initial values. This approach allows us to observe the sensitivity of each algorithm to initial conditions, a critical factor in real-world applications where starting estimates can vary significantly. By comparing these methods under identical conditions, we aim to highlight their respective strengths and limitations.

Using FEATool, a specialized tool for finite element analysis, as a benchmark, we further enhance the robustness of our comparison. FEATool provides a standard against which the custom-implemented algorithms in MATLAB can be measured. This comparative analysis not only underscores the practical applications of these numerical methods in engineering problems but also offers insights into their computational demands and accuracy.

Throughout the experiment, key performance metrics such as the number of iterations (epochs) to convergence, the computational time required for each run, and the evolution of the loss function over the epochs are meticulously recorded and analyzed. These metrics serve as the basis for evaluating the efficiency and reliability of each algorithm in solving the Steady 2D heat conduction problem. By presenting these findings, we contribute valuable knowledge to the field of numerical heat transfer simulations, providing a data-driven foundation for selecting appropriate iterative methods based on specific performance criteria and computational constraints.

4.1 Experiment 1

Initialization and Verification

To assess the accuracy and reliability of our custom implementation for solving the Steady 2D heat conduction problem, we initialized the boundary values as specified in Table 1. This initial setup was tested using our MATLAB code, and the resulting temperature distribution is depicted in Figure 11. To validate our implementation, we compared these results with those obtained using the FEATool software, shown in Figure 12. The congruence between these figures indicates that our program operates correctly, as it produces results consistent with those from a well-established tool.

Left border	Right border	Bottom border	Top border
300	100	400	400

Table 1: Initial Borders

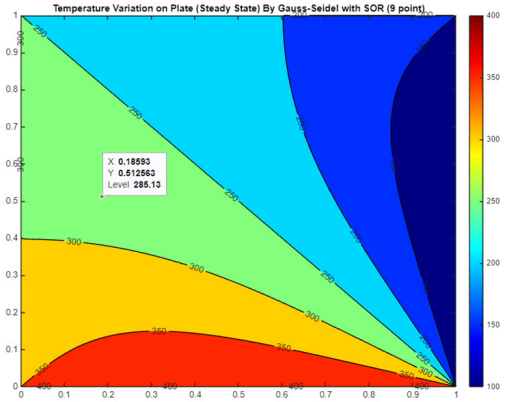

Figure 11: Implement by Mathlab

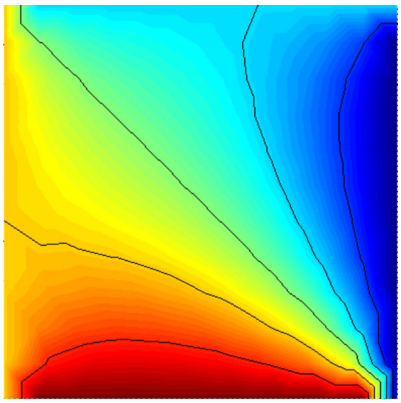

Figure 12: Using Featool

Method	Time (s)	Iterations
Jacobi	6.594078	18424
Gauss-Seidel	3.700655	10726
SOR	0.202659	498
9-point SOR	0.254031	526

Table 2: Comparison of Iterative Methods

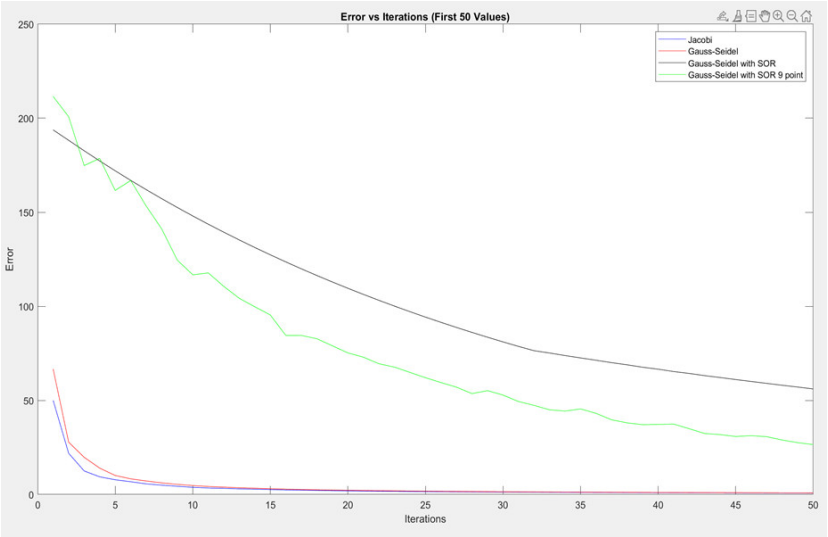

Figure 13: Plot error in the first 50 iterations

Convergence Analysis

When considering the convergence time and the number of iterations required for convergence, the results are summarized in Table 2. Additionally, the Error vs. Iterations plot (Figure 13) provides further insights into the performance of each method. From the data, it is evident that the SOR (Successive Over-Relaxation) methods, particularly the SOR-Gauss 4-point and 9-point algorithms, converge significantly faster than the Jacobi and Gauss-Seidel methods. Specifically, the SOR method achieved convergence in the least amount of time and with the fewest iterations, demonstrating its efficiency for this type of problem. The Error vs. Iterations plot corroborates these findings, showing a steeper decline in error for the SOR methods compared to Jacobi and Gauss-Seidel, which indicates quicker convergence.

4.2 Experiment 2

Initialization and Verification

In Experiment 2, we used the same setup as in Experiment 1, with boundary values specified in Table 3. The MATLAB results (Figure 14) matched well with those from FEATool (Figure 15), confirming the accuracy of our implementation.

Left border	Right border	Bottom border	Top border
300	300	400	200

Table 3: Initial Borders

Method	Time (s)	Iterations
Jacobi	6.088104	17038
Gauss-Seidel	3.481157	9875
SOR	0.200293	471
9-point SOR	0.250775	512

Table 4: Comparison of Iterative Methods

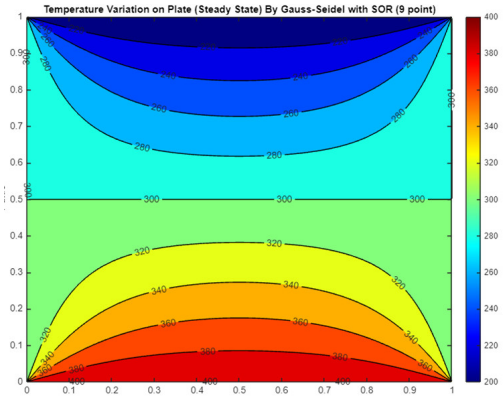

Figure 14: Implement by Mathlab

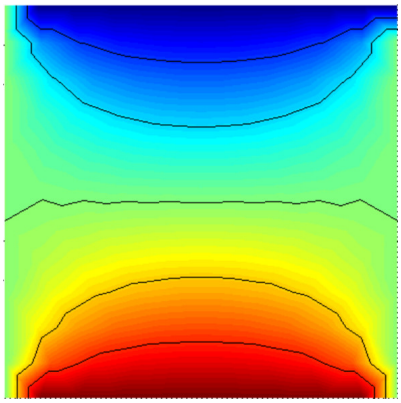

Figure 15: Using Featool

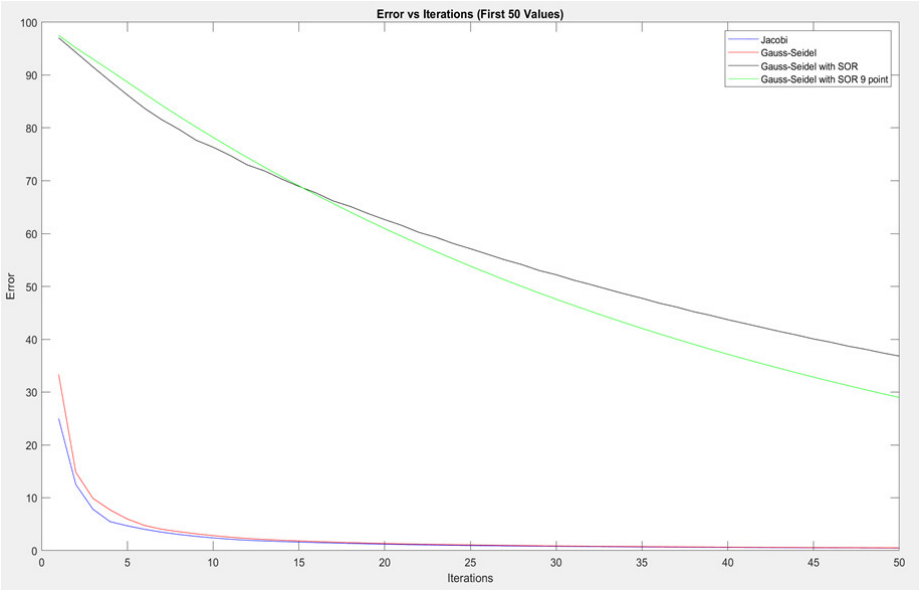

Figure 16: Plot error in the first 50 iterations

Convergence Analysis

Similar to Experiment 1, the convergence time and iterations (Table 4)) show that SOR methods converged faster than Jacobi and Gauss-Seidel methods. The Error vs. Iterations plot (Figure 16) supports these findings, indicating that SOR methods are more efficient.

4.3 Experiment 3

Initialization and Verification

In Experiment 3, we initialized the boundary values as shown in Table 5, following the same setup as in Experiments 1 and 2. The resulting temperature distribution from our MATLAB implementation, depicted in Figure 17, was validated against the results obtained using FEATool, as shown in Figure 18. The consistency between these figures confirms that our implementation is accurate and produces reliable results.

Left border	Right border	Bottom border	Top border
100	300	400	400

Table 5: Initial Borders

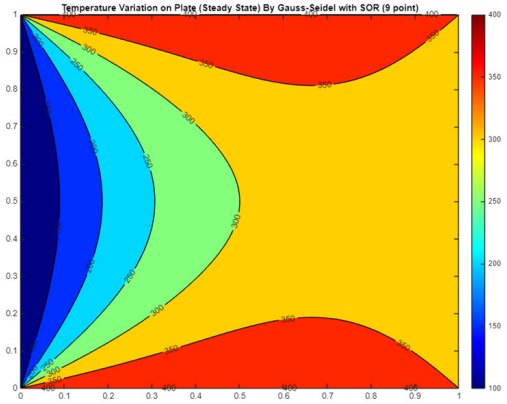

Figure 17: Implement by Mathlab

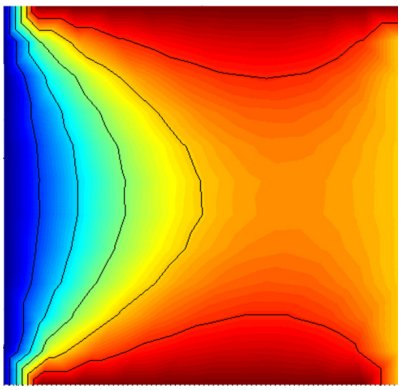

Figure 18: Using Featool

Method	Time (s)	Iterations
Jacobi	6.046164	17068
Gauss-Seidel	3.494897	9834
SOR	0.193138	470
9-point SOR	0.243434	512

Table 6: Comparison of Iterative Methods

Convergence Analysis

The convergence time and the number of iterations required, summarized in Table 6, indicate similar findings to Experiments 1 and 2. The Error vs. Iterations plot (Figure 19) further demonstrates that the SOR methods, particularly the SOR-Gauss 4-point and 9-point algorithms, converged faster than the Jacobi and Gauss-Seidel methods.

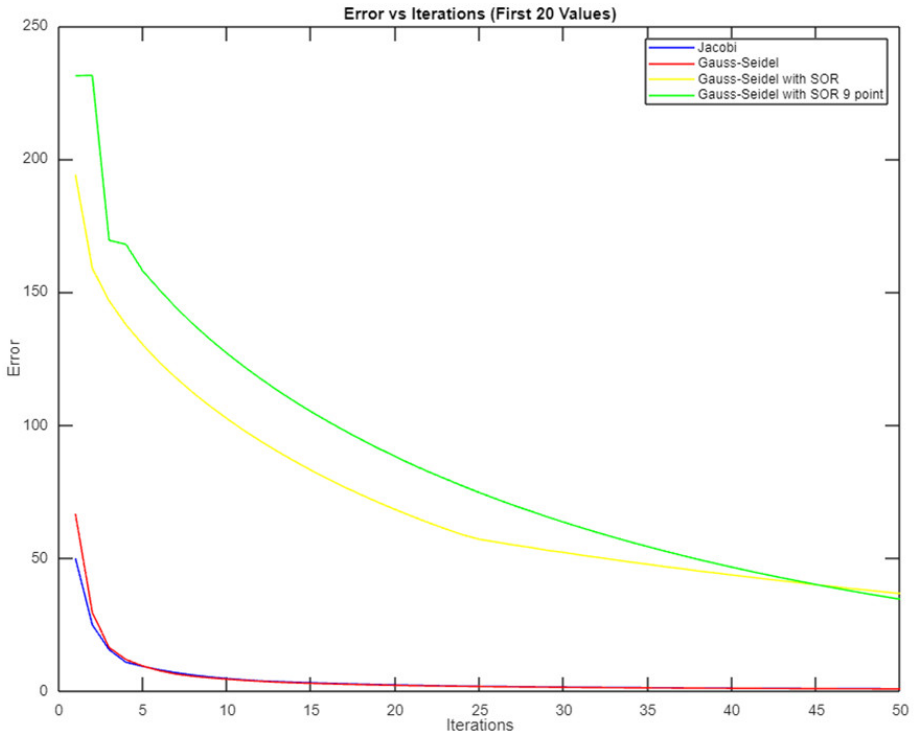

Figure 19: Plot error in the first 50 iterations

4.4 Insight

From the three experiments conducted, we derive the following key insights:

1. **Superior Performance of SOR**: The Successive Over-Relaxation (SOR) method significantly reduces both the number of iterations and computation time compared to Gauss-Seidel and Jacobi methods. This highlights the efficiency of SOR in solving the Steady 2D heat conduction problem.

2. **Comparison of 4-point vs 9-point SOR**: Both SOR variations perform similarly with marginal differences. The standard 4-point SOR is slightly more efficient, requiring fewer iterations and less time than the 9-point version. This suggests that the 4-point SOR is preferable for faster convergence with minimal computational overhead.

3. **Convergence Analysis**: Initially, SOR appears less stable than other methods but quickly surpasses them in convergence speed after initial iterations, effectively minimizing errors faster. This demonstrates that while SOR may start slower, its long-term efficiency is superior, making it a robust choice for rapid convergence.

5 Conclusion

- The Jacobi method is simple and easy to implement, but it has the longest convergence time.
- The Gauss method is an improved method that accelerates the convergence process of the Jacobi method.
- SOR stands out for rapid convergence and reduced computational demands, making it an optimal choice for solving high-precision and time-sensitive engineering problems. These findings confirm the effectiveness of SOR methods, particularly the 4-point variation, in practical applications requiring efficient and accurate solutions.
- The relaxation factor used does not depend on the temperature of the four boundaries but directly depends on the number of grid points divided on the disk.

References

[1] Han-Taw Chen, Shen-Yih Lin, and Lih-Chuan Fang. "Estimation of surface temperature in two-dimensional inverse heat conduction problems". In: *International Journal of Heat and Mass Transfer* 44.8 (2001), pp. 1455–1463. ISSN: 0017-9310. DOI: https://doi.org/10.1016/S0017-9310(00)00212-X. URL: https://www.sciencedirect.com/science/article/pii/S001793100000212X.

[2] Taloub Djedid, Abdelkarim Bouras, and Zied Driss. "Modeling and Numerical Solution of the Laplace Equation in 2D by the Finite Difference Method: Case of the Heat Equation - Study of Stability". In: *Research Square* (2023). PREPRINT (Version 1). URL: https://doi.org/10.21203/rs.3.rs-2440144/v1 (visited on 01/05/2023).

[3] T. N. Narasimhan. "Fourier's heat conduction equation: History, influence, and connections". In: *Reviews of Geophysics* 37.1 (1999), pp. 151–172. DOI: https://doi.org/10.1029/1998RG900006. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/1998RG900006. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/1998RG900006.

[4] Azali Saudi. *The Solutions to 2D Laplace's Equation using Iterative Methods*. Mar. 2022.

[5] E. R. Smith, P. J. Daivis, and B. D. Todd. "Measuring heat flux beyond Fourier's law". In: *The Journal of Chemical Physics* 150.6 (Feb. 2019), p. 064103. ISSN: 0021-9606. DOI: 10.1063/1.5079993. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.5079993/13569281/064103_1_online.pdf. URL: https://doi.org/10.1063/1.5079993.