AM5630 Foundations of Computational Fluid Dynamics

Computer Assignment 2

Ruthwik Chivukula

ME21B166

Contents

1 Problem Statement

Consider a rectangular plate of unit thickness with dimensions 1 x 2 units (2-D steady-state diffusion problem, Boundary value problem). Plot the steady-state temperature contour of the rectangular plate for the following boundary conditions.

1.1 Boundary Conditions

$\mathbf{T = 100}$ at the top wall and $\mathbf{T = 0}$ at all other walls

1.2 Governing equation

We will be using the steady-state temperature equation for this problem statement.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

1.3 Numerical Formulation

We are supposed to plot the steady-state temperature contour of the rectangular plate obtained from different schemes, which are as follows:

- Point Gauss-Seidel
- PSOR.
- Line Gauss-Seidel
- LSOR
- ADI
- ADI with relaxation

1.3.1 Point Gauss Seidel

The discretized equation that we will be using here is:

$$T^{k+1}(i,j) = \frac{T^k(i+1,j) + T^{k+1}(i-1,j) + \beta^2 T^k(i,j+1) + \beta^2 T^{k+1}(i,j-1)}{2(1+\beta^2)}$$

This is an explicit scheme and hence can be solved directly by iterating over two for-loops.

Pseudo Code:

- initialize T^k as per the given boundary conditions
- iterate over x and y via for-loops to calculate T^{k+1} using the above discretized equation
- compute the absolute difference difference between T^k and T^{k+1} after every iteration
- copy values of T^{k+1} into T^k after every iteration
- continue until the absolute difference is below a set threshold

1.3.2 Line Gauss Seidel

The discretized equation that we will be using here is:

$$T^{k+1}(i,j) = \frac{T^{k+1}(i+1,j) + T^{k+1}(i-1,j) + \beta^2 T^k(i,j+1) + \beta^2 T^{k+1}(i,j-1)}{2(1+\beta^2)}$$

This is an implicit scheme, and in order to solve it, we need to formulate a tridiagonal matrix. As this is the Line Gauss-Seidel method, the values along every row can be constructed into a tridiagonal matrix which can be solved using TDMA.

Pseudo Code:

- initialize T^k as per the given boundary conditions
- iterate over the rows and formulate a tridiagonal matrix to calculate T^{k+1} using the above discretized equation for each of the rows
- after spanning the entire 2D matrix, compute the absolute difference difference between T^k and T^{k+1}
- copy values of T^{k+1} into T^k after one entire update step
- continue until the absolute difference is below a set threshold

1.3.3 PSOR

The discretized equation that we will be using here is:

$$T^{k+1}(i,j) = (1-w)T^k(i,j) + \frac{w(T^k(i+1,j) + T^{k+1}(i-1,j) + \beta^2 T^k(i,j+1) + \beta^2 T^{k+1}(i,j-1))}{2(1+\beta^2)}$$

This is a relaxed version of the Point Gauss-Seidel scheme. If you set w = 1 in the above equation you obtain back the point Gauss-Seidel result. Setting $w > 1$ will lead to over-relaxation (lesser iterations for convergence). By trial and error it has been found that w = 1.80 gives the optimal results.

Pseudo Code:

- initialize T^k as per the given boundary conditions
- iterate over x and y via for-loops to calculate T^{k+1} using the above discretized equation
- compute the absolute difference difference between T^k and T^{k+1} after every iteration

- copy values of T^{k+1} into T^k after every iteration
- continue until the absolute difference is below a set threshold

1.3.4 LSOR

The discretized equation that we will be using here is:

$$T^{k+1}(i,j) = (1-w)T^k(i,j) + \frac{w(T^{k+1}(i+1,j) + T^{k+1}(i-1,j) + \beta^2 T^k(i,j+1) + \beta^2 T^{k+1}(i,j-1))}{2(1+\beta^2)}$$

This is a relaxed version of the Line Gauss-Seidel scheme. If you set w = 1 in the above equation, you obtain back the line Gauss-Seidel result. Setting $w > 1$ will lead to over-relaxation (lesser iterations for convergence). By trial and error it has been found that w = 1.28 gives the optimal results.

Pseudo Code:

- initialize T^k as per the given boundary conditions
- iterate over the rows and formulate a tridiagonal matrix to calculate T^{k+1} using the above discretized equation for each of the rows
- after spanning the entire 2D matrix, compute the absolute difference difference between T^k and T^{k+1}
- copy values of T^{k+1} into T^k after one entire update step
- continue until the absolute difference is below a set threshold

1.3.5 ADI

ADI is an example of an implicit scheme. It has two steps, in the first step, we sweep the X direction, and in the next step, we sweep the Y direction. We construct a tridiagonal matrix for the first step and find the unknowns which are used to construct the tridiagonal matrix for the second step. The equations are as follows:

$$-T^{k+1/2}(i+1,j)+2(1+\beta^2)T^{k+1/2}(i,j)-T^{k+1/2}(i-1,j)=\beta^2T^k(i,j+1)+\beta^2T^{k+1/2}(i,j-1)$$

$$-\beta^2T^{k+1}(i,j+1)+2(1+\beta^2)T^{k+1}(i,j)-\beta^2T^{k+1}(i,j-1)=T^{k+1}(i-1,j)+T^{k+1/2}(i+1,j)$$

Pseudo Code:

- initialize T^k as per the given boundary conditions
- iterate over the rows and formulate a tridiagonal matrix to calculate $T^{k+1/2}$ using the above discretized equation for each of the rows (X sweep)
- After computing $T^{k+1/2}$ over the entire 2D matrix, use these values to find T^{k+1} which can be done by iterating over columns and formulating the corresponding tridiagonal matrices (Y sweep)
- after spanning the entire 2D matrix, compute the absolute difference difference between T^k and T^{k+1}
- copy values of T^{k+1} into T^k after one entire update step
- continue until the absolute difference is below a set threshold

1.3.6 ADI with relaxation

Relaxed version of the ADI scheme. If we set $w = 1$ we get the same result as regular ADI. For $w > 1$, the scheme will be relaxed and hence will take a lesser number of iterations. It is found that w = 1.32 gives the best relaxation. The equations are as follows:

$$T^{k+1/2}(i,j) = (1-w)T^{k}(i,j) + \frac{w(T^{k+1/2}(i+1,j) + T^{k+1/2}(i-1,j) + \beta^2(T^{k}(i,j+1) + T^{k+1/2}(i,j-1)))}{2(1+\beta^2)}$$

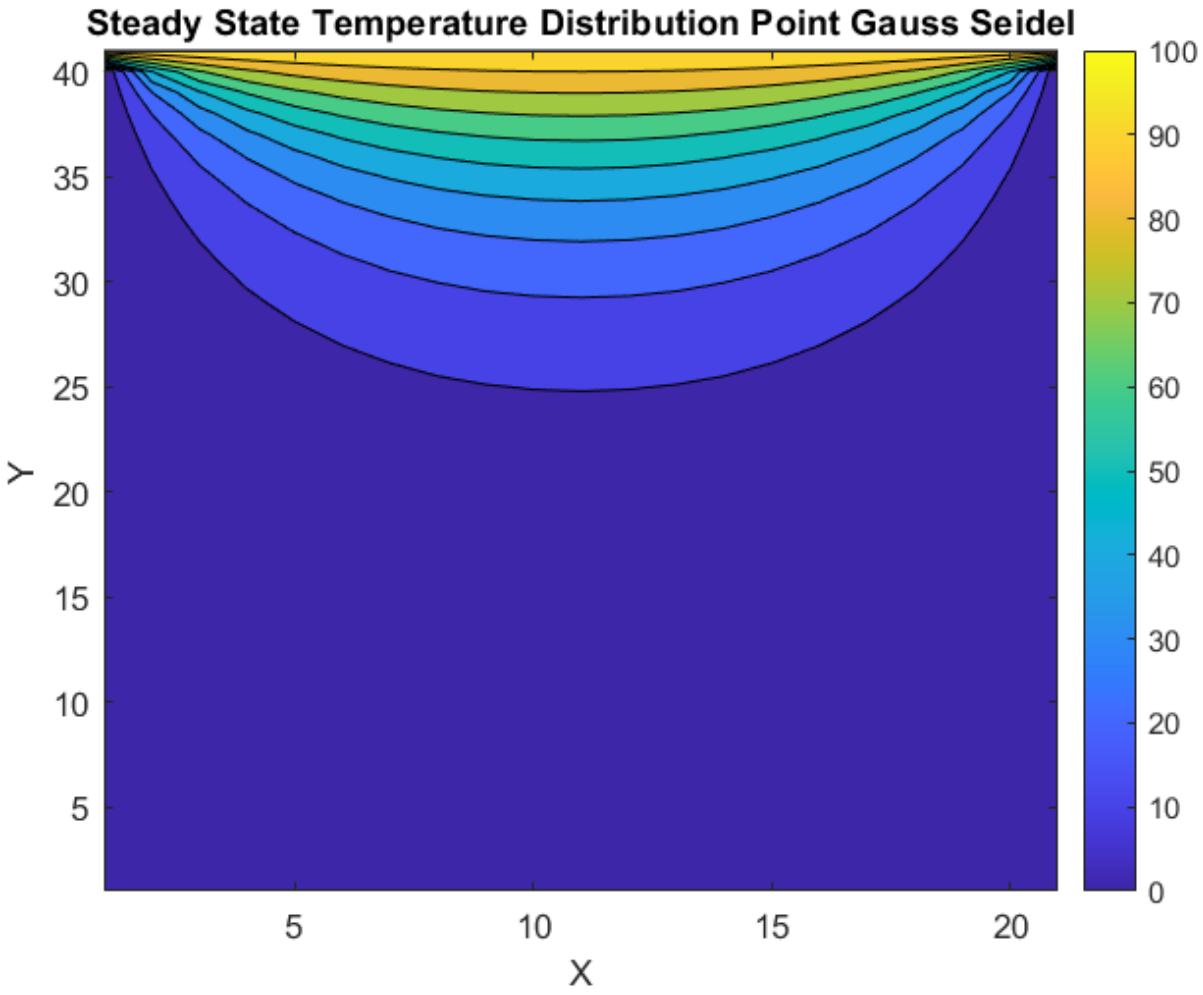
$$T^{k+1}(i,j) = (1-w)T^{k+1/2}(i,j) + \frac{w(T^{k+1/2}(i+1,j) + T^{k+1}(i-1,j) + \beta^2(T^{k+1}(i,j+1) + T^{k+1}(i,j-1)))}{2(1+\beta^2)}$$

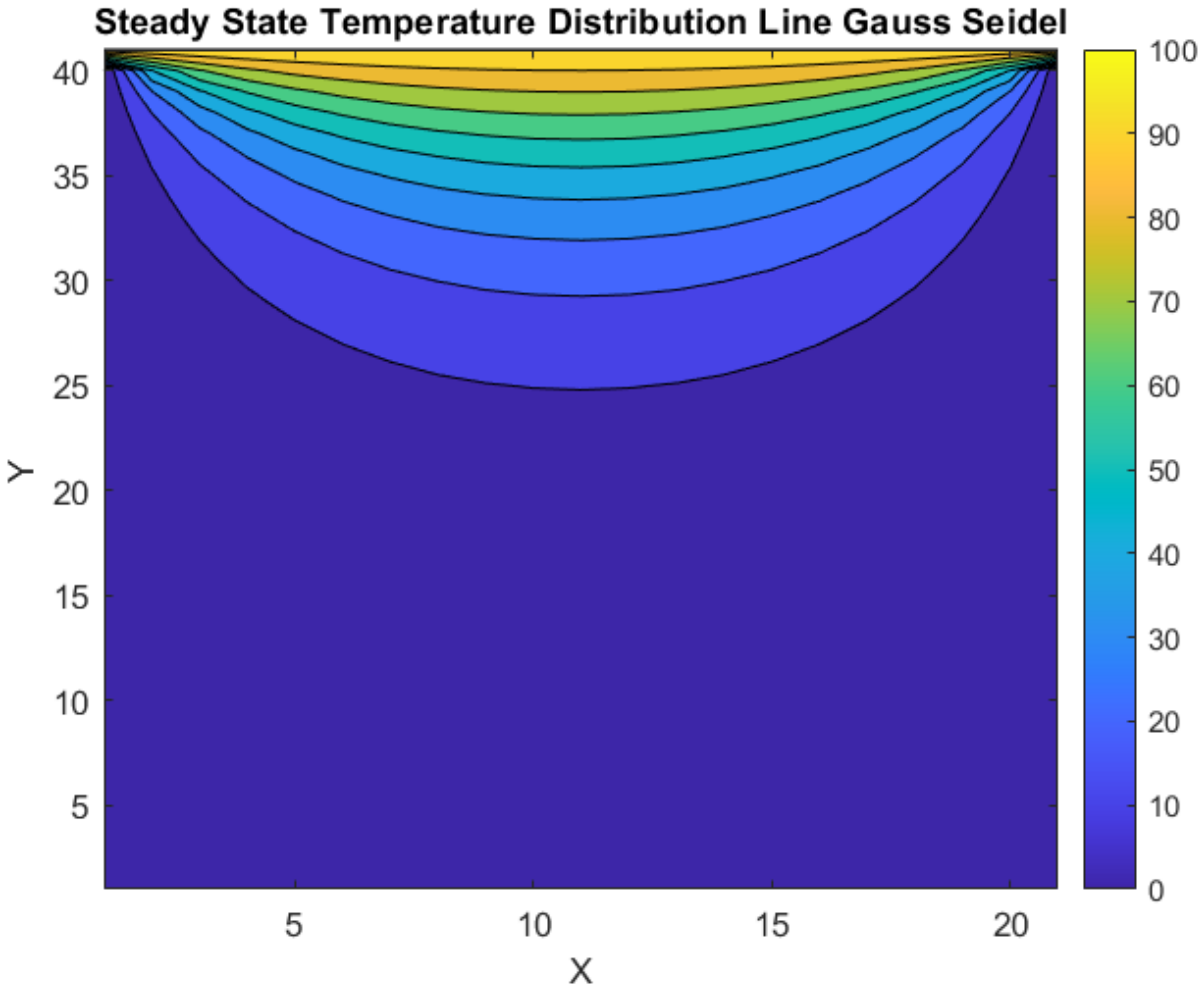
Pseudo Code:

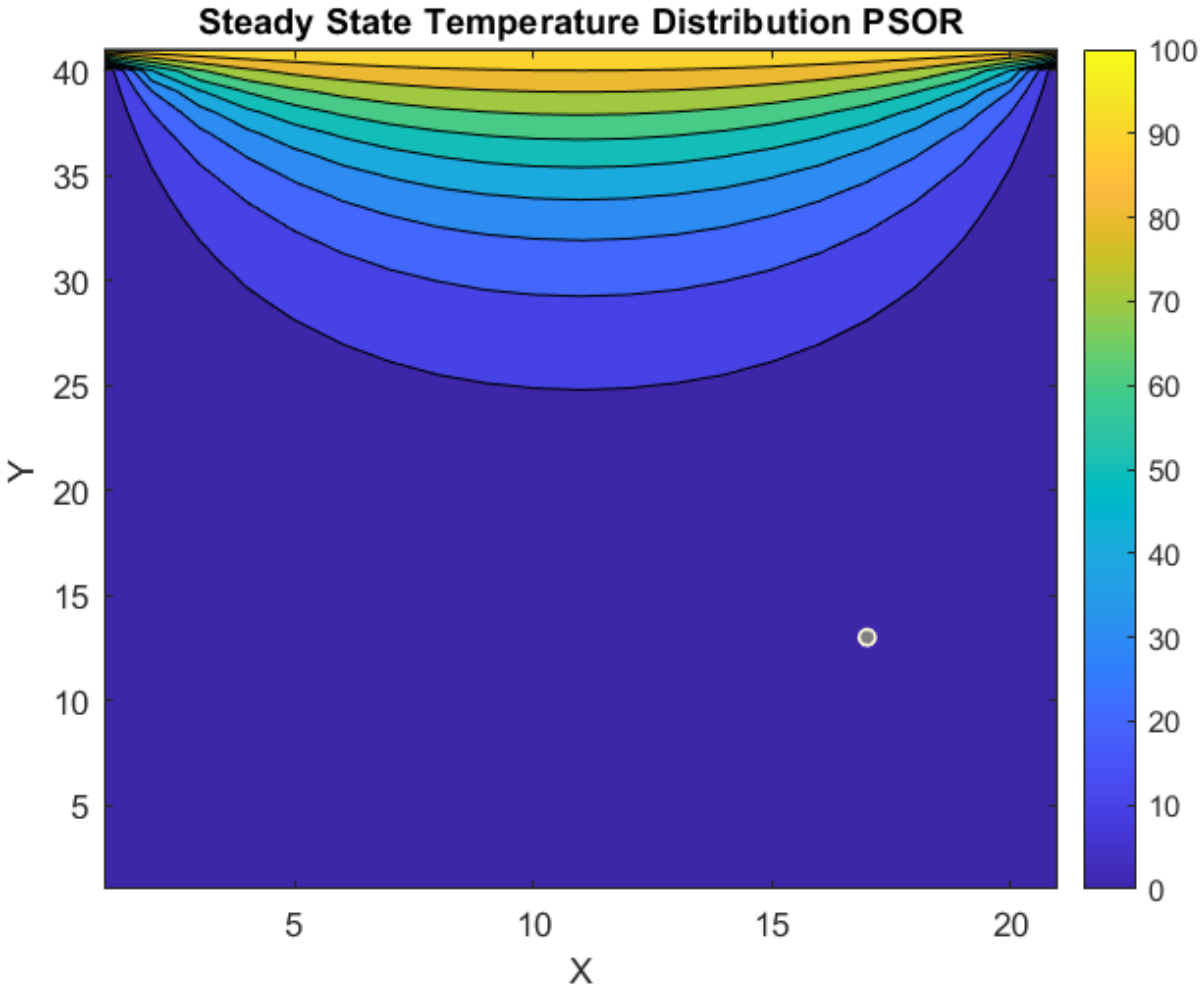
- initialize T^k as per the given boundary conditions
- iterate over the rows and formulate a tridiagonal matrix to calculate $T^{k+1/2}$ using the above discretized equation for each of the rows (X sweep)
- After computing $T^{k+1/2}$ over the entire 2D matrix, use these values to find T^{k+1} which can be done by iterating over columns and formulating the corresponding tridiagonal matrices (Y sweep)
- after spanning the entire 2D matrix, compute the absolute difference difference between T^k and T^{k+1}
- copy values of T^{k+1} into T^k after one entire update step
- continue until the absolute difference is below a set threshold

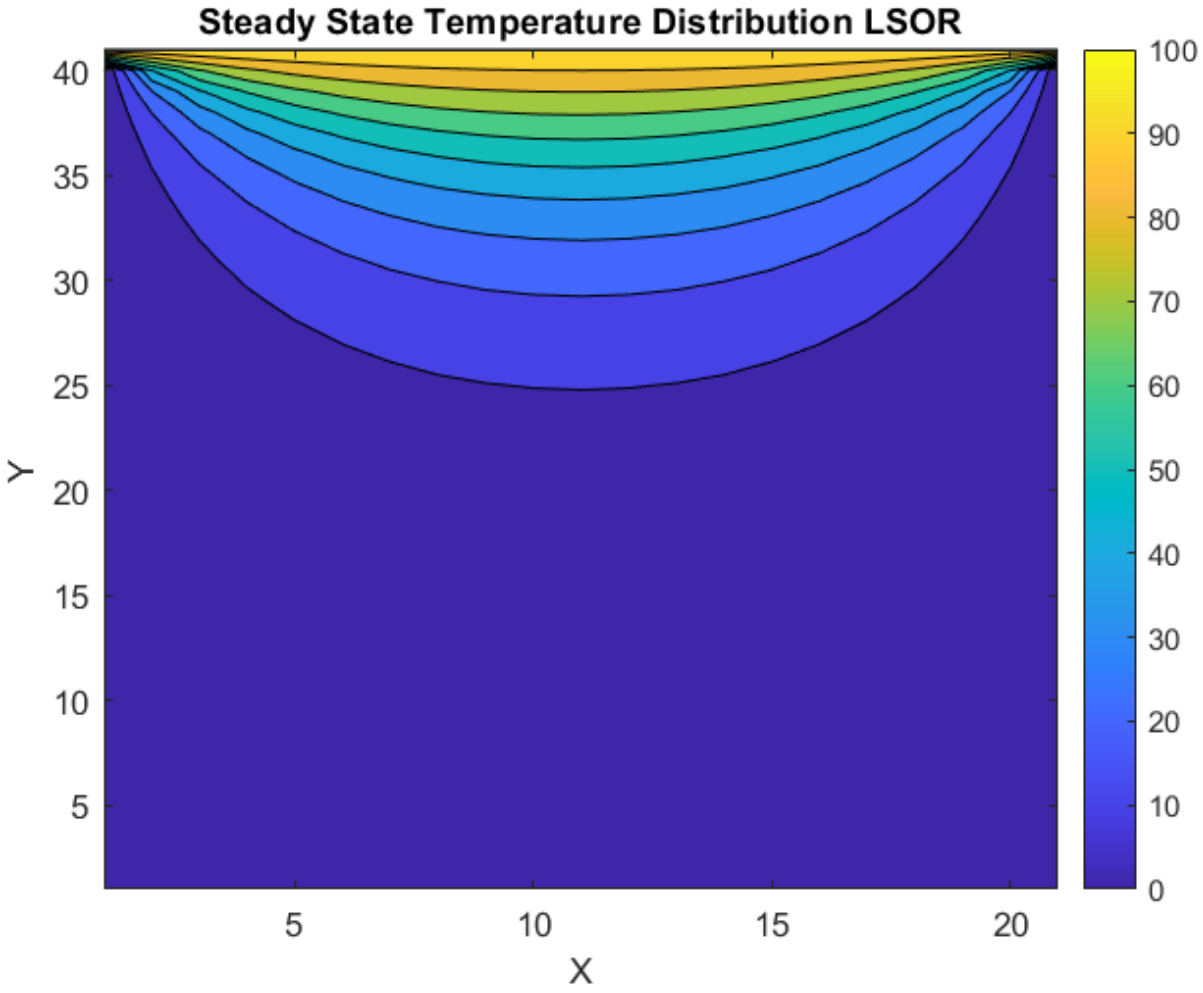
2 Plots

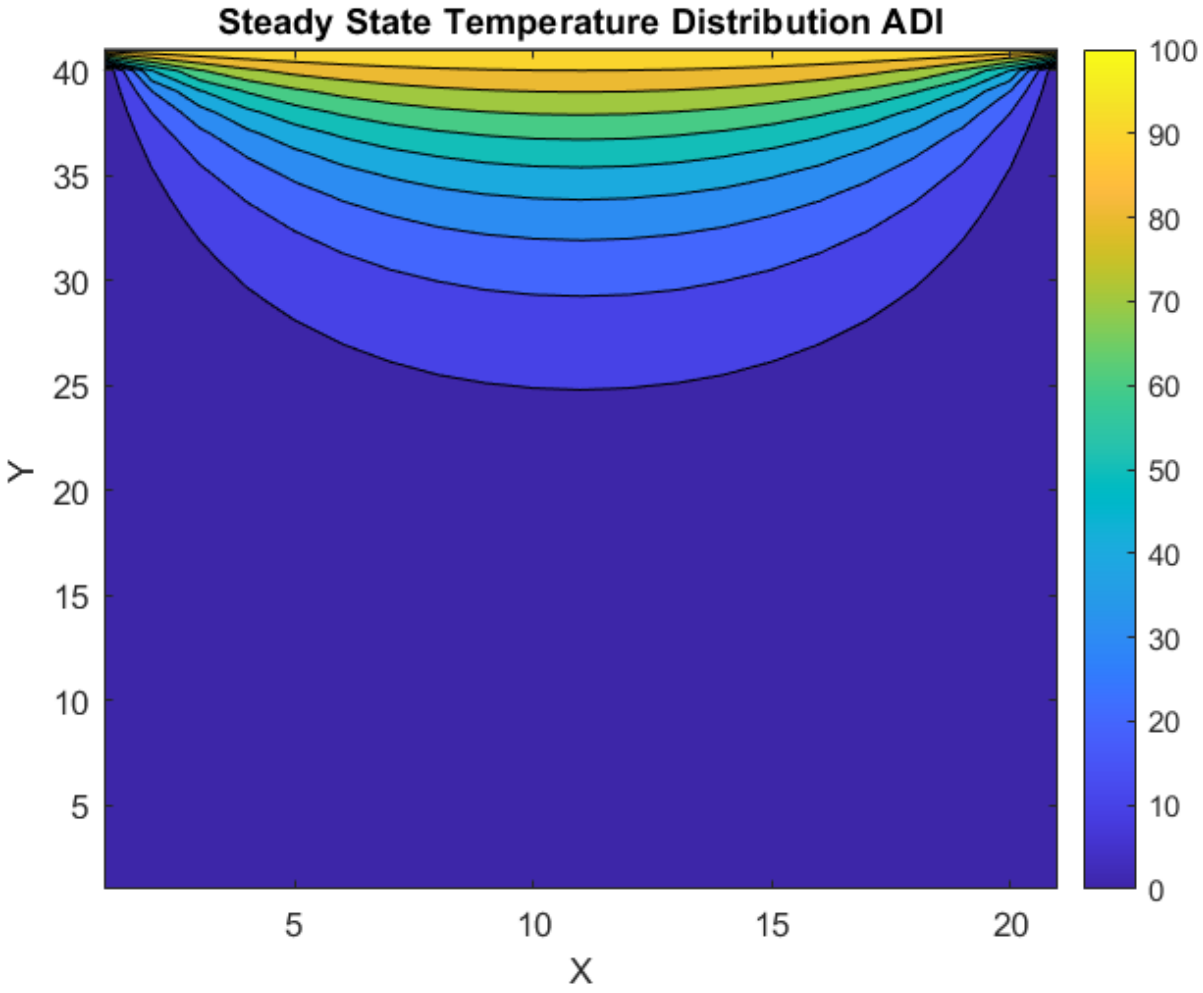
Below are the plots that have been obtained after implementing the above schemes.

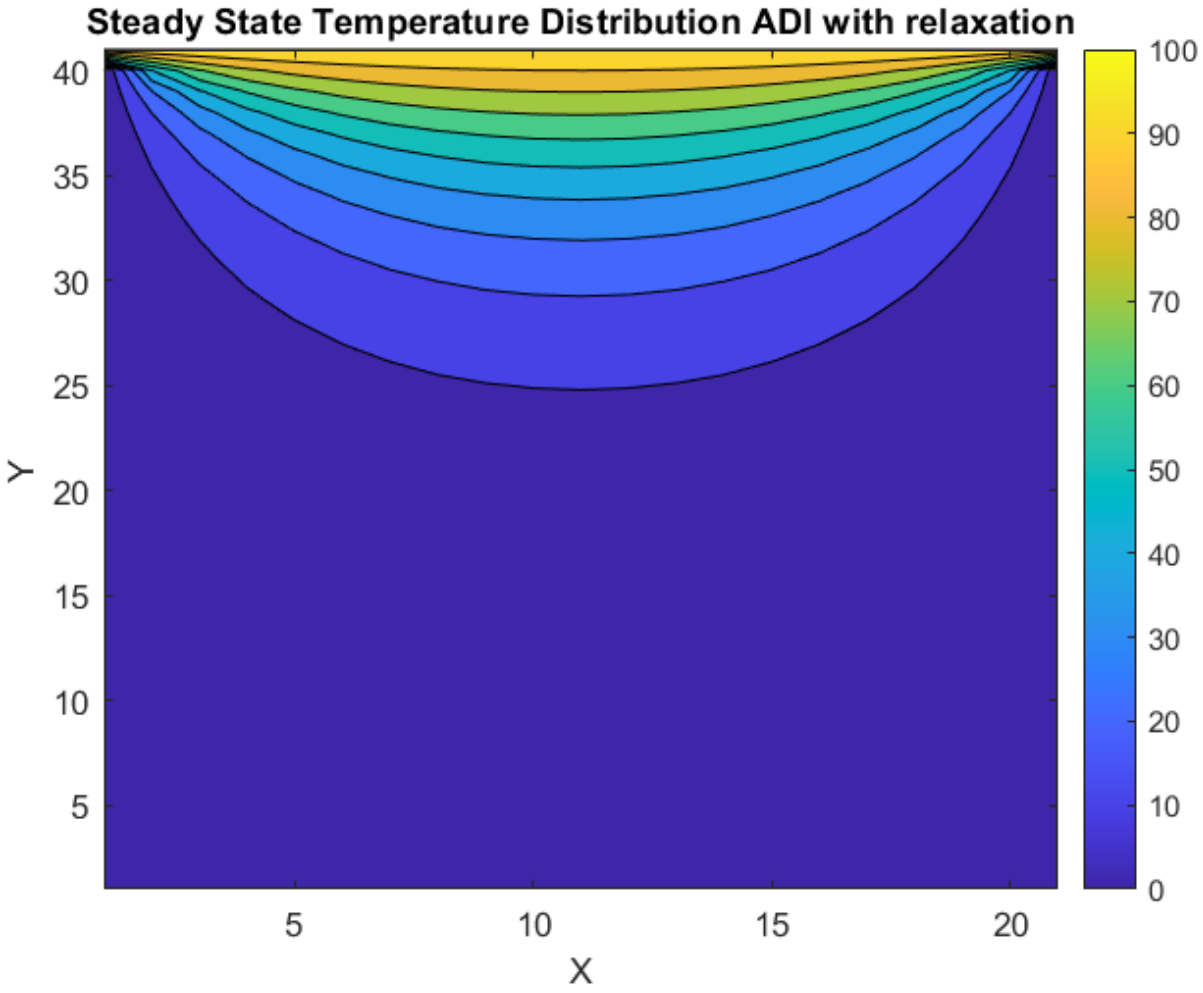



Steady State Temperature Distribution Line Gauss Seidel
Y
X
40
35
30
25
20
15
10
5
5
10
15
20
100
90
80
70
60
50
40
30
20
10
0


Steady State Temperature Distribution PSOR
40
35
30
25
20
15
10
5
Y
5
10
15
20
X
100
90
80
70
60
50
40
30
20
10
0


Steady State Temperature Distribution LSOR
40
35
30
25
20
15
10
5
Y
5
10
15
20
X
100
90
80
70
60
50
40
30
20
10
0


Steady State Temperature Distribution ADI
40
35
30
25
20
15
10
5
Y
5
10
15
20
X
100
90
80
70
60
50
40
30
20
10
0



3 Results

In this section, we will plot a table of the results obtained from each of the schemes. We will analyze the values and draw insightful conclusions. **Note:** There is no significant difference between CPU time and Total Computation Time because the problem set is not very huge. CPU time is the entire time taken for running the code whereas Total Computation Time accounts only for the time taken for running the algorithm code.

3.1 Table

Scheme Used	Gride size	Iterations	w	CPU time	Total Computation Time
Point GS	41 x 21	593	N/A	0.0156	0.0156
Line GS	41 x 21	327	N/A	0.3125	0.2969
PSOR	41 x 21	77	1.80	0.0469	0.0469
LSOR	41 x 21	58	1.28	0.1406	0.1406
ADI	41 x 21	174	N/A	0.5781	0.5781
ADI with relaxation	41 x 21	27	1.32	0.1875	0.1875

3.2 Conclusions

- Point Gauss-Seidel takes the most number of iterations
- ADI with relaxation takes the least number of iterations
- Implicit schemes converge faster in comparison to explicit schemes
- CPU time and total computation time are more or less the same value, which won't be the case if we deal with bigger problem sets.

3.3 Computer Code

code has been attached in the zip file. The zip file has a comprehensive report, all codes that were written for schemes and the corresponding plots