Computational Fluid Dynamcis: Project 1

Hari Balaji (22110092)

Department of Mechanical Engineering

Prof. Dilip Srinivas Sundaram

1. Problem Statement

To write a computer program and solve the following 2D steady-state diffusion equation using the finite difference method on a square domain of unit length.

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=S_\phi \tag{1}$$

Given boundary conditions:

$$\phi(0,y)=500\exp(-50[1+y^2])$$
$$\phi(1,y)=100(1-y)+500\exp(-50y^2)$$
$$\phi(x,0)=100x+500\exp(-50[1-x]^2)$$
$$\phi(x,1)=500\exp(-50\{[1-x]^2+1\})$$

Source term:

$$S_\phi=50000\exp(-50\{[1-x]^2+y^2\})(100\{[1-x]^2+y^2\}-2)$$

Given analytical solution:

$$\phi(x,y)=500\exp(-50\{[1-x]^2+y^2\})+100x(1-y)$$

The solution needs to be obtained in four different ways:

1. **Part 1:** Using Gauss Elimination
2. **Part 2:** Using the Gauss Seidel Iterative Method
3. **Part 3:** Using the line-by-line method and the tridiagonal matrix decomposition algorithm
4. **Part 4:** Using the Alternating Direction Implicit method

```
\newcommand{\abslangague}{
    \iflanguage{spanish}{
        {\bfseries\itshape Resumen-}%
    }{%
        {\bfseries\itshape Abstract-}%
    }%
}
```

Code 1. *Abstract language code*

2. Mesh Details and Discretisation Approach

2.1. Mesh Details

1. **21×21 Grid:**
   - Used for Part 1 only.
   - A unit square domain was divided into 20 equal rows and columns for a total of 441 grid points.
   - It requires a relatively small amount of computing power.
2. **41×41 Grid:**
   - Used for Parts 1, 2 and 3.
   - A unit square domain was divided into 40 equal rows and columns for a total of 1681 grid points.
   - It is a more refined grid than the 21×21 grid.
3. **81×81 Grid:**
   - Used for Parts 1, 2 and 3.
   - A unit square domain was divided into 80 equal rows and columns for a total of 6561 grid points.
   - This grid is even more refined than the 41×41 grid.
4. **161×161 Grid:**
   - Used for Parts 2 and 3.
   - A unit square domain was divided into 160 equal rows and columns for a total of 25921 grid points.
   - This grid is very refined, and requires a significant amount of computing power.
5. **41×81 Grid:**
   - Used for Part 4.
   - A unit square domain was divided into 40 rows and 80 columns for a total of 3321 grid points.
   - Its objective is to compare the ADI method and the row-sweep and column sweep methods.

2.2. Approach for Discretising the Equations

3. Discretised Equations

Consider equation 1:

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=S_\phi$$

We first consider a grid having finite differences Δx and Δyin the x and y directions respectively.

$$x_i=i\cdot\Delta x \text{ and } y_j=j\cdot\Delta y$$

Where:

$$i=0,1,2,...,N_x-1 \text{ and } j=0,1,2,...,N_y-1$$

$$\Delta x=\frac{1}{N_x-1} \text{ and } \Delta y=\frac{1}{N_y-1}$$

Next, we obtain the second-order partial derivatives of ϕ with respect to x and y using the finite difference approximation as shown:

$$\frac{\partial^2\phi}{\partial x^2}\approx\frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{(\Delta x)^2}$$
$$\frac{\partial^2\phi}{\partial y^2}\approx\frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{(\Delta y)^2}$$

Next, we substitute the finite difference approximations into equation 1 considerig $S_{i,j}$ to be the source term at a grid point (i, j):

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{(\Delta x)^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{(\Delta y)^2} = S_{i,j} \quad (2)$$

This is the required discretised equation. If we further assume that $\Delta x = \Delta y = \Delta$, then the discretisation equation can be further simplified to:

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j} = S_{i,j}\Delta^2 \quad (3)$$

Final form of the discretised equations for each part:

Part 1 (Gauss Elimination):
This is the same equation as equation 3.

Part 2 (Gauss-Seidel Iterative Method):
This is the same equation as equation 3.

Part 3 (Line-by-Line Method):
Rearranging equation 3

$$\phi^{n+1}_{i-1,j} - 4\phi^{n+1}_{i,j} + \phi^{n+1}_{i+1,j} = S^{n}_{i,j}\Delta^2 - \phi^{n}_{i,j-1} - \phi^{n}_{i,j+1} \quad (4)$$

This is the discretised equation which is evaluated for every row (row-sweep) and then solved using TDMA for the nth iteration. The iterations continue until convergence is achieved.

Part 4 (Alternating Direction Implicit or ADI Method):
Rearranging equation 2, we get:

$$\phi^{n+1}_{i+1,j} + \phi^{n+1}_{i-1,j} - 2\phi^{n+1}_{i,j}\left(1 + \frac{(\Delta x)^2}{(\Delta y)^2}\right) = S^{n}_{i,j}(\Delta x)^2 - \left(\frac{(\Delta x)^2}{(\Delta y)^2}\right)(\phi^{n}_{i,j+1} + \phi^{n}_{i,j-1}) \quad (5)$$

and

$$\phi^{n+1}_{i,j+1} + \phi^{n+1}_{i,j-1} - 2\phi^{n+1}_{i,j}\left(1 + \frac{(\Delta y)^2}{(\Delta x)^2}\right) = S^{n}_{i,j}(\Delta y)^2 - \left(\frac{(\Delta y)^2}{(\Delta x)^2}\right)(\phi^{n}_{i+1,j} + \phi^{n}_{i-1,j}) \quad (6)$$

These are the discretised equations obtained for the ADI method. Equation 5 is the row-sweep discretisation and equation 6 is the column-sweep discretisation. The equation sets are evaluated in the same way as in *Part 3.*

4. Solution Methodology

4.1. Part 1 (Gauss Elimination)

1. Discretise the given differential equation (explanation in *section 3*).
2. Define the grid sizes, functions for calculating the source term S_ϕ and the boundary conditions.
3. Initialise the main loop which loops over the different grid sizes.
4. Initialise the matrices A and b to solve the matrix equation $A\phi = b$.
5. Set the grid spacing and divide the grid into the required number of points and define a function to flatten the indexes into a single dimension so that vector operations can be performed for improving speed and efficiency.
6. Fill matrices A and b point by point. First, fill the boundary conditions, and then fill the interior points.
7. Define a function which performs Gauss Elimination such that it takes A and b as input and returns ϕ. Gauss elimination works as follows:
   - **Step 1 - Forward Elimination:** Convert the augmented matrix (A) into an upper triangular form using row operations.
   - **Step 2 - Back Substitution:** Eliminate the element in the last row, then eliminate the element in the second last row, and go on until all the elements upto the top row are eliminated.
8. Perform Gauss Elimination on the filled matrices A and b and reshape the obtained 1D vector into a 2D matrix, and plot the solution as a contour plot. Also plot the graph of CPU run time versus the total number of grid points,

4.2. Part 2 (Gauss-Seidel Iterative Method)

1. Perform steps 1 to 3 as mentioned in *section 4.1*, and then define a function which executes the Gauss-Seidel Iterative Method and call it. The Gauss-Seidel Iterative Elimination works as follows:
   - Make an initial guess for the solution vector ϕ.
   - Use the guess to calculate the next value of the solution vector, and check if the difference between the two satisfies the given tolerance.
   - Repeat the previous step until the tolerance is satisfied and the solution is converged or until the maximum number of iterations.
2. Reshape the obtained 1D vector into a 2D matrix, and plot the solution as a contour plot and also plot the other graphs (residuals vs. number of iterations, CPU run time vs. total number of grid points).

4.3. Part 3 (Line-by-Line Method with TDMA)

1. Discretise the given differential equation (explanation in *section 3*).
2. Define the grid sizes, functions for calculating the source term S_ϕ and the boundary conditions.
3. Initialise the main loop which loops over the different grid sizes.
4. Make the initial guess for ϕ and apply the boundary conditions to it.
5. Set the tolerance, maximum number of iterations and initialise the diagonal arrays (sub, main and super diagonals) along with the residual array.

6. Define a function which executes the tri-diagonal matrix algorithm (TDMA). The TDMA algorithm is a special case of Gauss Elimination which only works for tri-diagonal matrices.

7. Start the main iteration loop and store the old value of ϕ to check for convergence.

8. Define another loop to loop through the rows of the grid and initialise the right hand side vector (based on equation 5). This loop performs the row-sweep.

9. Define another loop to loop through the individual points inside a row. There is now a triple nested for loop system.

10. Calculate the right hand side term for a point. Make sure to check if the point is at the start or end of a row, and if it is apply the boundary conditions to it.

11. Repeat step 9 until the right hand side terms for all the points in a row are calculated and the right hand side vector is filled. Then, execute the TDMA on the right hand side vector and the diagonal vectors.

12. Repeat steps 7 to 10 until all rows are accounted for. This is one iteration. Now, calculate the residual and check for convergence. The residuals can be calculated using the following formula:

$$R^n = [A]\phi^n - B$$

where R^n is the residual of dimension n. We can take the L2-norm of R^n to check for convergence, i.e. convergence is achieved when $\frac{R_2^n}{R_2^{n-1}} < \varepsilon_{tol}$. If convergence is not achieved, move on to the next iteration and repeat the process until convergence is achieved or the maximum iteration limit is reached.

13. Plot the required contour plot for the finest grid. Also plot the other graphs required (residuals vs. number of iterations, CPU run time vs. gridsize).

4.4. Part 4 (Alternating Direction Implicit or ADI Method)

This method is similar to the row-sweep method mentioned in the previous subsection. The change is that after the loop which performs the row sweep (step 7), another loop which performs column sweep should be defined. This introduces the element of alternating direction in this algorithm. Based on the given problem statement, we need to use the equations 5 and 6 since the grid size is non-uniform.

5. Results and Discussions

5.1. Part 1 (Gauss Elimination)

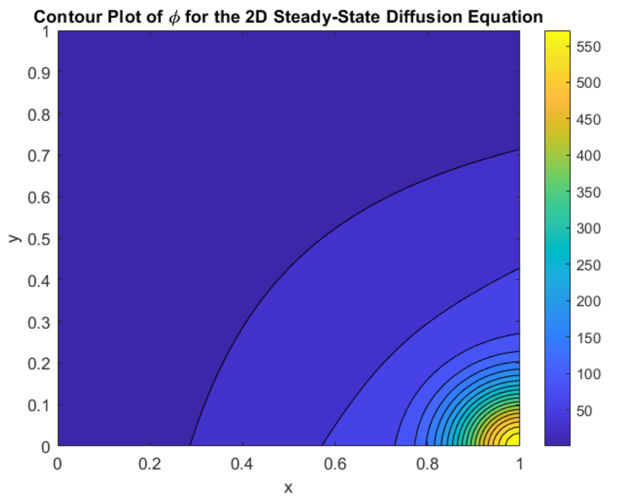

Figure 1. *Contour plot of the finest grid (81x×81) for Part 1*

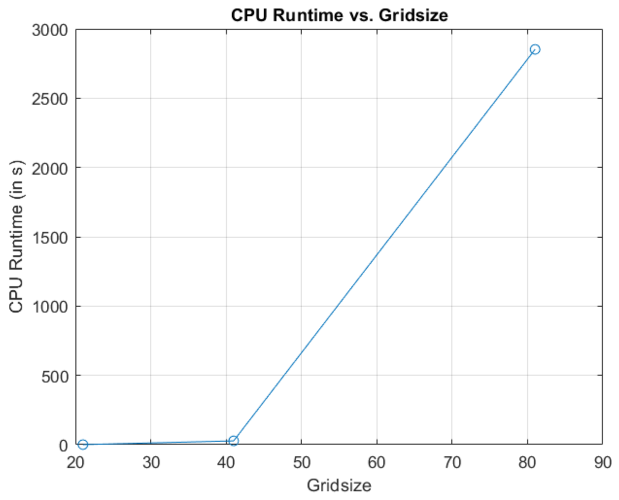

Figure 2. *CPU run time vs. the total number of grid points for Part 1*

Figure 1 shows the contour plot of the finest grid (81×81) for the solution of the two dimensional heat diffusion equation solved using Gauss Elimination. It provides a clear visualisation, but the amount of computation time taken (almost fifty minutes) makes it unfavourable (as seen from figure 2).
Using Gauss Elimination for even larger gridsizes is not computationally efficient since it will take a very large amount of time due to the point-by-point nature of its method.

5.2. Part 2 (Gauss-Seidel Iterative Method)

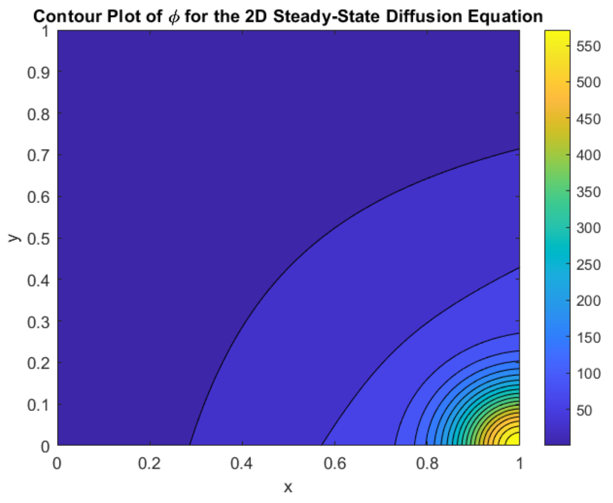

Figure 3. *Contour Plot of the finest grid (161×161) for Part 2*

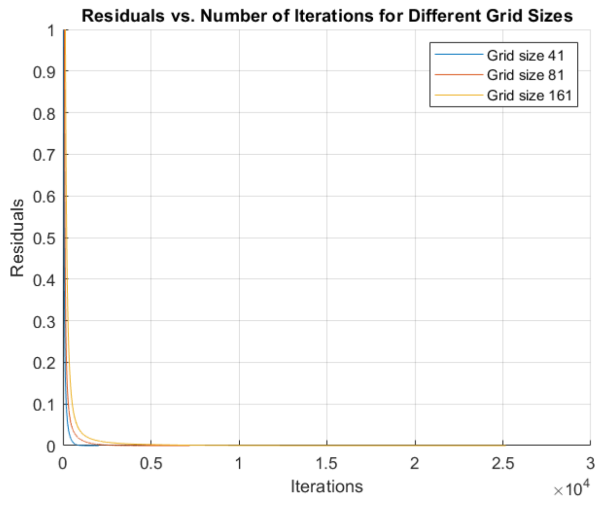

Figure 4. *Residuals vs. Number of Iterations for Part 2*

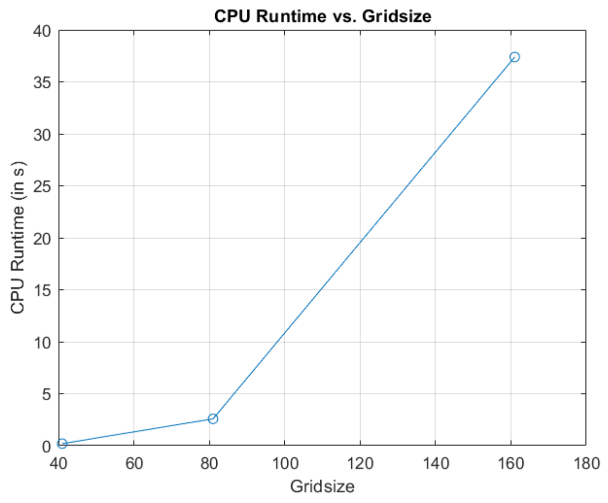

Figure 5. *CPU run time vs. the total number of grid points for Part 2*

Figure 3 shows the contour plot of the finest grid (161×161) for the solution of the two dimensional heat diffusion equation solved using the Gauss-Seidel Iterative Method. This method is much faster compared to Gauss Elimination since it proceeds from an initial guess and iterates over that to obtain the final solution. Figure 5 shows that the 81×81 grid only takes about three seconds compared to fifty minutes in the previous part. From figure4, we can see that smaller grid sizes converge quicker (i.e. in a lesser number of iterations) compared to larger ones.
This method performs reasonably well for 161 grid points, taking about thirty-eight seconds. However, this method will also take significant amounts of time to perform iterations for much larger grids (this shows me how inefficient the Gauss Elimination method actually is). Also, there is a possibility that this method might not converge for certain systems of equations, which makes it slightly risky (an additional step might be required to check if any of the diagonal elements of the system are non zero).

5.3. Part 3 (Line-by-Line Method with TDMA)

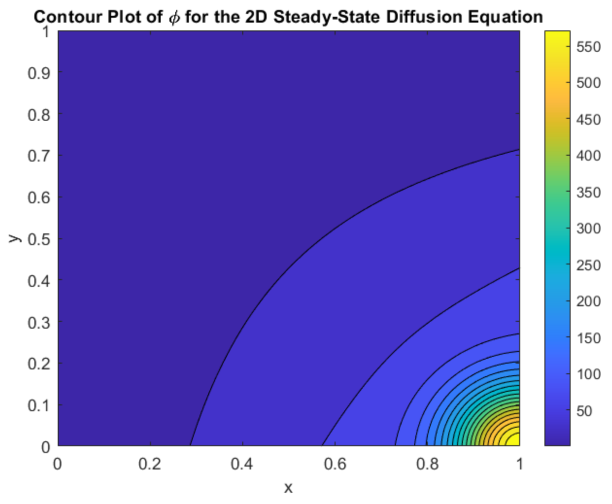

Figure 6. *Contour plot of the finest grid (161×161) for Part 3*

Figure 3 shows the contour plot of the finest grid (161×161) for the solution of the two dimensional heat diffusion equation solved using the Line-by-Line method. The sweeping was done across rows and solved using TDMA for this case. Figure 8 shows that the row sweep method is significantly faster than Gauss Elimination and the Gauss-Seidel Iterative Method, with the 161 point grid taking only about ten seconds to converge.
Figure 7 shows that for a higher grid size, a higher number of iterations are required for convergence. This method can be used for larger grids, since its convergence rate is very reasonable.
Figure 9 compares the residual for a grid size of 161 between the Gauss-Seidel Iterative Method and the Line-by-Line method. We can clearly see that the Line-by-Line method converges in a lesser number of iterations

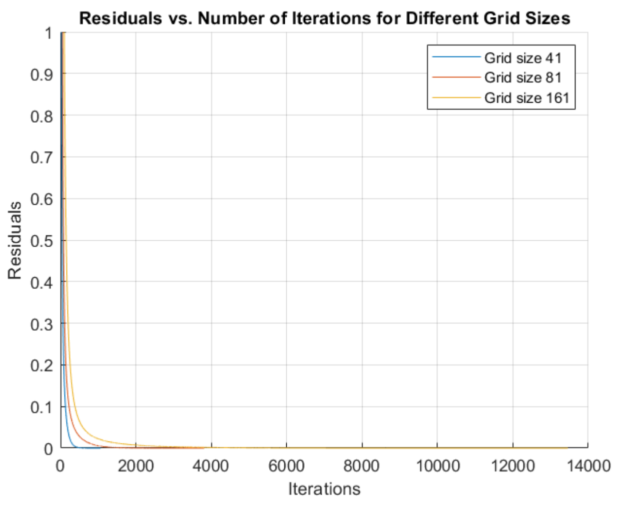

Figure 7. *Residuals vs. Number of Iterations for Part 3*

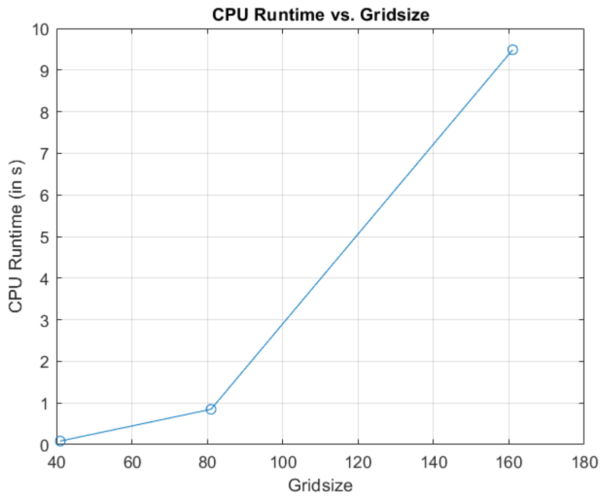

Figure 8. *CPU run time vs. the total number of grid points for Part 3*

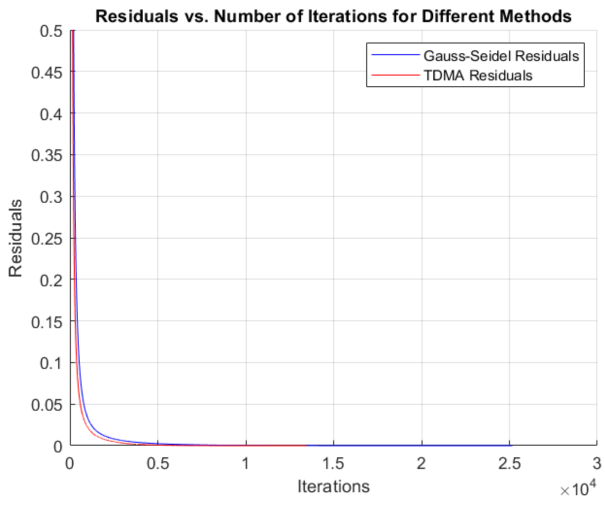

Figure 9. *Residuals vs. Number of Iterations compared for Part 2 and Part 3*

5.4. Part 4 (Alternating Direction Implicit or ADI Method)

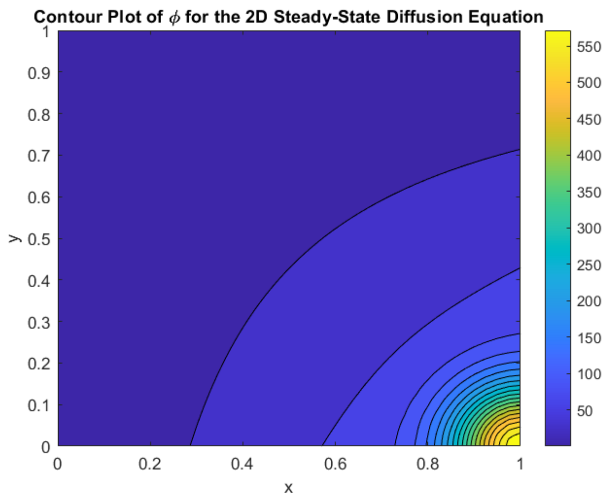

Figure 10. *Contour plot of the 41×81 grid for Part 4*

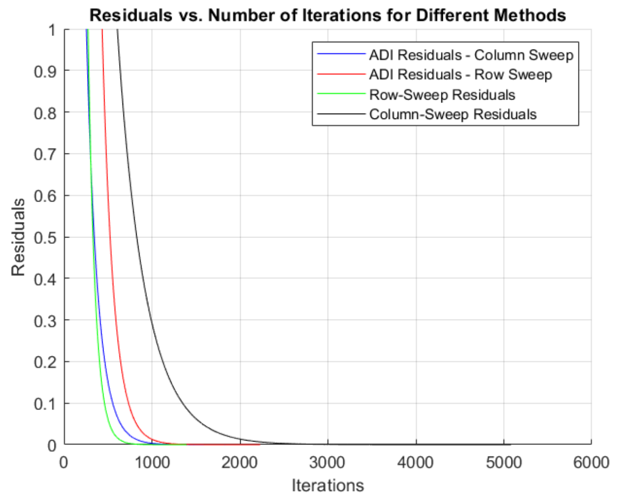

Figure 11. *Residuals of the row-sweep, column-sweep and the ADI method vs. Number of Iterations*

Figure 10 shows the contour plot of the given grid (41×81) for the solution of the two dimensional heat diffusion equation solved using the Alternating Direction Implicit (ADI) Method. This method is an optimal method which performs a row-sweep and a column-sweep in the same iteration, and then solves the system using TDMA for faster convergence. Figure 11 shows the residuals for the given 41×81 grid for row-sweep, column-sweep and ADI. We can see from the graph that the row-sweep converges in the least number of iterations, followed by the set for the ADI method and finally ending with column sweep (Note that the number of iterations in the graph). The ADI has two residuals (one for its row-sweep and one for its column-sweep) since it performs a row and column sweep in one iteration.

6. Conclusion

The Gauss Elimination method is the slowest of all and it is the least efficient method to solve the given equation.

The Gauss-Seidel Iterative Method is faster than Gauss Elimination but it is still quite efficient.

The Line-by-Line method is quite efficient and can be used to solve the given equation for a significant grid size without consuming a lot of time.

The ADI method works well for non-uniform domains and grid sizes since it performs both row and column sweeps alternately leading to faster convergence.

7. Acknowledgements

I would like to thank to Prof. Dilip S. Sundaram for teaching me the principles to enable me to successfully complete this project. His expertise and support greatly helped us.

I would also like to thank the TAs Malay Vyas and Pardha Sai for helping me understand the nuances of the code in this project. Their support invaluable to me.

I would also like to thank IIT Gandhinagar for providing me the platform and support necessary to complete this project.

Lastly, I would like to thank my peers, whose continued support was instrumental in completing this experiment successfully.