Indian Institute of Technology, Gandhinagar

ME 605: Computational Fluid Dynamics

Project 1: Computer Simulation of Elliptic PDEs

Instructor: Prof. Dilip Sundaram

Abhinab Sharma
Roll No: 24250005

Abhiram Ramachandran
Roll No: 24250006

Department of Mechanical Engineering

Contents

1. Problem Statement

You are required to write a computer program and solve the following 2D steady-state diffusion equation using the finite difference method:

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

on a square domain of unit length. The boundary conditions are as follows:

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

The source term is given by:

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

For reference, the analytical solution is given below:

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

1. Solve the 2D steady-state diffusion equation using Gaussian elimination method for the following grids: 21, 41, and 81 grid points in each direction. Show the computed ∅ field as a contour plot for the finest grid. Plot the CPU run time vs total number of grid points and discuss the trend and comment on the computational efficiency of the Gauss elimination method.

2. Solve the same 2D steady-state diffusion equation using the Gauss-Seidel iterative method for the following three grids: 41, 81, and 161 grid points in each direction. Show the computed ϕ field as a contour plot for the finest grid. Plot the residual vs number of iterations for the three grids in the same plot. Compare and discuss the dependence of the convergence rate on the total number of grid points. Plot the variation of CPU run time with total number of grid points for the Gauss-Seidel iterative method and discuss the trend. How do the CPU run times of the Gauss-Seidel iterative method compare with those of Gauss elimination method?

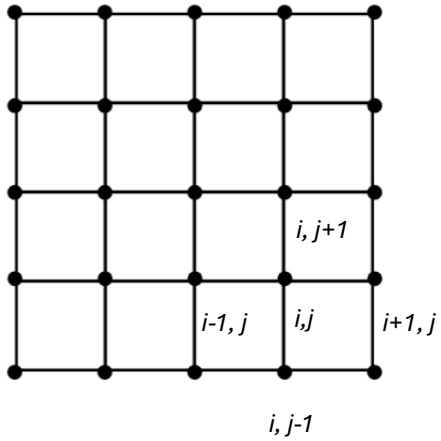
3. Solve the same 2D steady-state diffusion equation using the line-by-line method (row sweep) for the following three grids: 41, 81, and 161 grid points in each direction. Please note that you will need to use the TDMA solver to solve the resulting system of linear equations because of the tridiagonal structure of the resulting coefficient matrices. Show the computed ϕ field as a contour plot for the finest grid. Plot residual vs number of

iterations for the line-by-line method and for the Gauss-Seidel method for the 161 x 161 grid (both in the same plot). Compare and discuss the trends. Plot the variation of CPU run time with total number of grid points for the line-by-line method. How does the CPU run times and the obtained scaling compare with those obtained for the Gauss elimination method and point-wise iterative method?

4. Solve the same 2D steady-state diffusion equation using the Alternating Direction Implicit (ADI) method for the 41x81 grid and plot the residual vs number of iterations for row-wise sweep, column-wise sweep, and ADI method in the same plot. Discuss the trends.

1. DISCRETIZING THE EQUATION

Consider the equation that is given below. The mesh that is given below is an approximate representation of the mesh that has been used in solving the 2D Equation.



The partial differential equations can be discretized into a set of linear algebraic equations using the central difference scheme (CDS),

$$\left(\frac{\partial^2\phi}{\partial x^2}\right)_{i,j} = \phi_{i+1,j} + \phi_{i-1,j} - 2\phi_{i,j}$$

$$\left(\frac{\partial^2\phi}{\partial y^2}\right)_{i,j} = \phi_{i,j+1} + \phi_{i,j-1} - 2\phi_{i,j}$$

Substituting these discretized equations in the 2D steady state equation with source term while considering that Δx and Δy are equal since the domain is of unit length and the source term is computed at the corresponding $(i,j)^{th}$ nodal point,

$$\frac{\phi_{i+1,j} + \phi_{i-1,j} - 2\phi_{i,j}}{(\Delta)^2} + \frac{\phi_{i,j+1} + \phi_{i,j-1} - 2\phi_{i,j}}{(\Delta)^2} = S\phi_{i,j}$$

2. GAUSS ELIMINATION

2.1. Discretized Equation

The basic method of solving linear algebraic equations is the Gauss Elimination method. It's used widely for reducing a large number of equations into a smaller number. The matrix elements are modified by forward elimination, and further, the values of each unknown are found using back-substitution.

The discretized equation is given as:

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

$$\text{where,} \qquad \Delta x = \Delta y = \Delta$$

This discretized equation is then represented in the form of a matrix given by:

$$A\phi = b$$

where A is the coefficient matrix, and ϕ and b are the vectors.

2.2. Solution Methodology

The Gauss Elimination involves two main processes- forward elimination and backward substitution. In forward elimination, the coefficient matrix is converted into an upper triangular matrix, by pivoting a row and then carrying out the forward elimination with respect to that row. This takes place in the function *gaussian_elimination* in the code.

Following forward elimination, back substitution is done by finding out value for each unknown from the lowermost row, and then finding the remaining values in the upward direction.

Here, at each point, there is an unknown in the ϕ vector. The grid points which are in 2D dimensions undergoes a translation into 1D. The matrix A and vector b are constructed and are passed through a function *gauss_elim* that contains the code to run the Gauss Elimination while setting up the boundary conditions, and the unknowns are calculated.

2.3. Results

The contour plots obtained after solving the equations by Gauss Elimination method as well as the CPU run time for the method is as shown below:

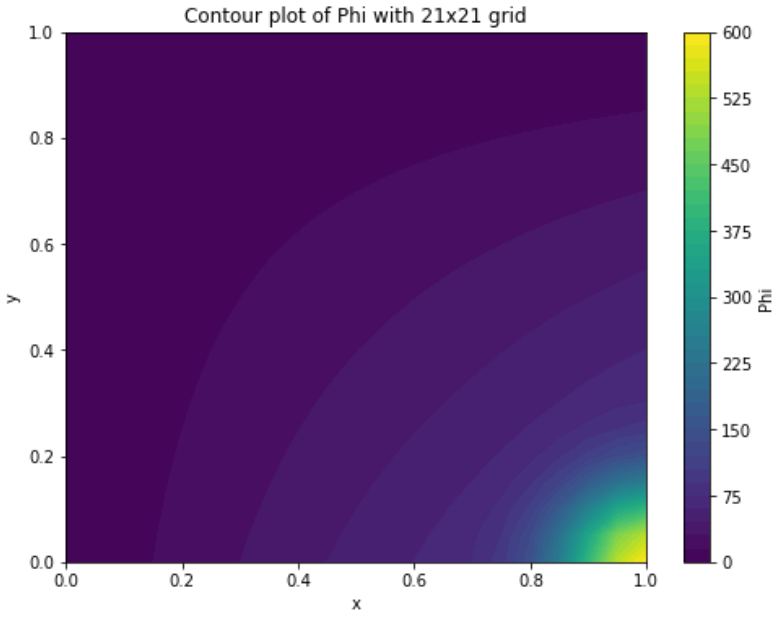


Figure 1: Contour plot of φ for 21 x 21 grid

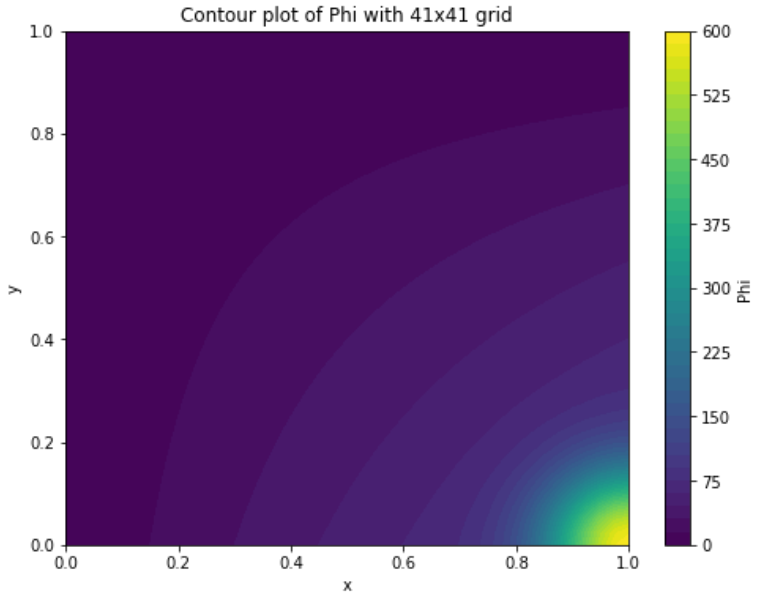


Figure 2: Contour plot of φ for 41 x 41 grid

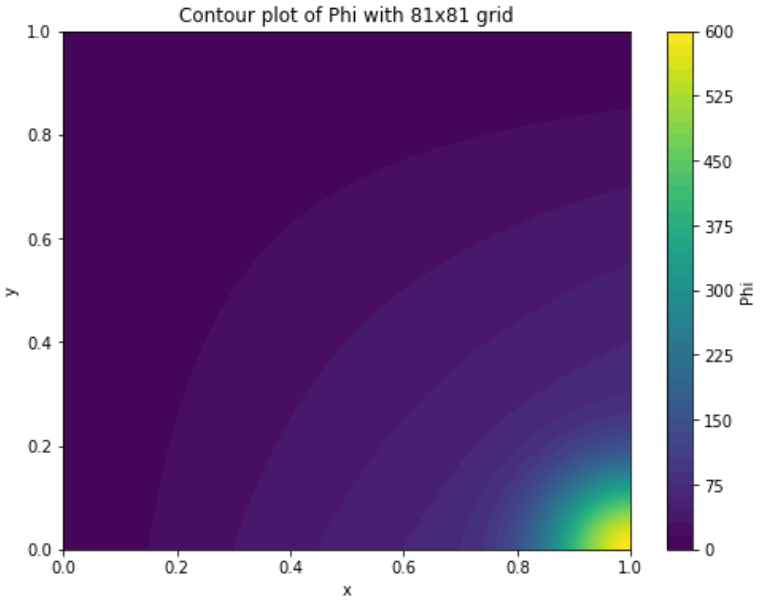


Figure 3: Contour plot of φ for 81 x 81 grid

The CPU run time for the Gauss Elimination method is given below:

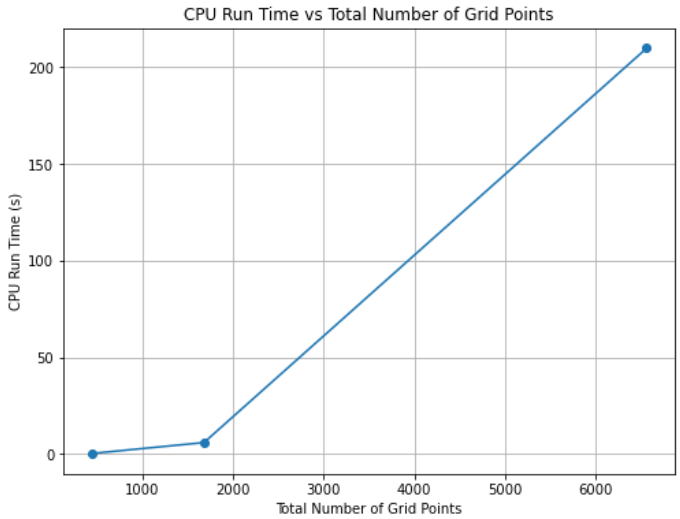


Figure 4: CPU Run time vs Total Number of Grid Points

2.4. Discussion

Gauss elimination usually scales as $O(n^3)$ where n is the number of unknowns. The problem that is at hand is of two-dimensional, i.e. that extends in both x and y directions, with N grid points in each direction. As a result, the complexity of the method increases even further.

In the CPU run time vs total number grid points plot, there is a sharp increase in the run time with increase in the number of grid points. As the number of grid points increases, larger number of equations needs to be solved simultaneously at each grid point until the solution converges, which increases the computational cost. This indicates that the performance of Gauss Elimination method is sensitive to the problem size.

Based on the above results and the discussion, we can easily conclude that the Gauss Elimination works best only for a small number of grid points (or rather, a coarse mesh). The time complexity increases as one increases the number of grid points considered and the method becomes highly inefficient.

Final Thoughts: *Gauss Elimination works best only for coarse mesh. In practical applications of CFD, finer grids are used for higher accuracy. In such scenarios, one may have to look into other methods with which the computation is time-efficient and accurate.*

3. GAUSS-SEIDEL METHOD

3.1. Discretization

Gauss-Seidel method is an iterative method in which the updated value of ϕ is used to compute the values of ϕ at the other nodes from the previous iteration as soon as they are available. The discretized equation for the Gauss-Seidel method is given as shown:

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

3.2. Solution Methodology

In the Python code, there are two values of ϕ, the values from the current and the previous iterations. As the iteration proceeds, the new value of ϕ is stored in a variable to be used for the computation in the following iteration.

A main characteristic of an iterative solution is that the numerical solution will not be exactly same as that of the analytical solution. So, after *n* iterations, the relative change between the numerical and the analytical solution is computed, and the difference in value is obtained. This difference in value is called the residual. One convenient condition to terminate the iteration is to ensure that the maximum difference in the ϕThe value of the current and the previous iteration falls below some predetermined acceptable error value. The smaller the acceptable error, the more accurate the numerical solution will be. But this condition is obtained at the expense of higher number of iterations.

3.3. Results

The contour plot of ϕ for the finest grid is as shown:

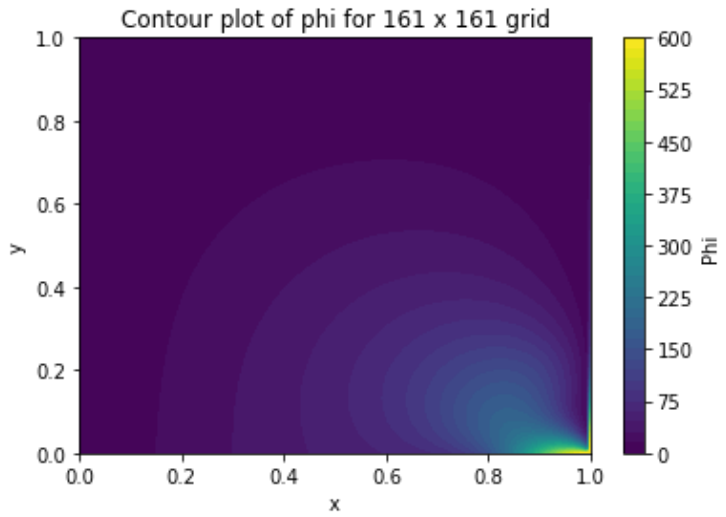
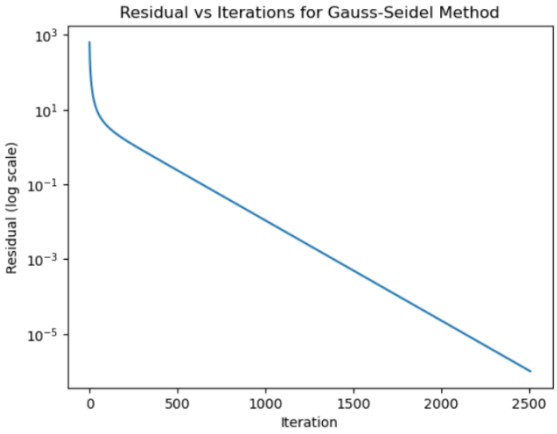
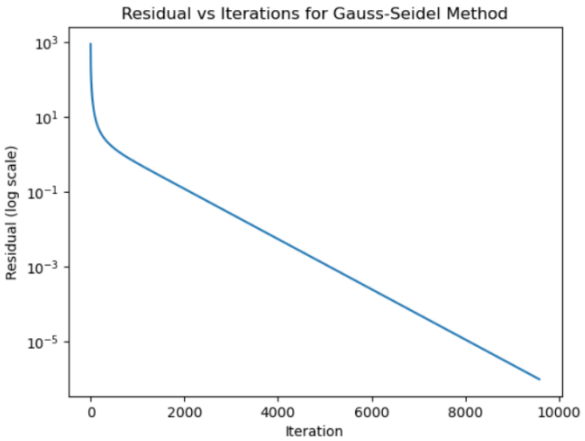


Figure 5: Contour plot of ϕ for 161 x 161 grid

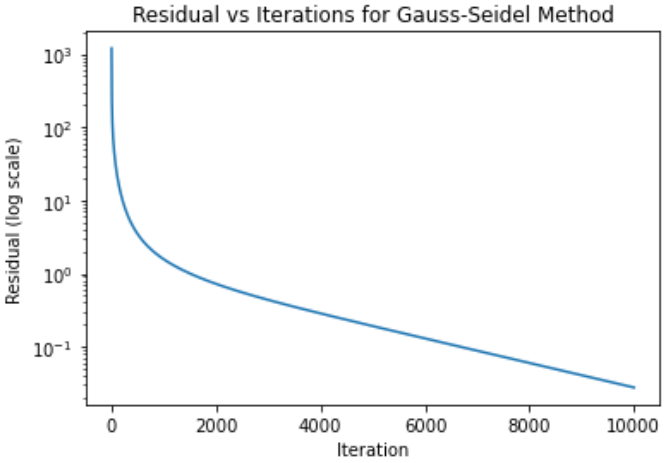
The residual versus number of iterations for the grids using Gauss-Seidel are as shown:



(a)



(b)



(c)

Figure 6: Residual vs Iterations for Gauss-Seidel Method for (a) 41 x 41 (b) 81 x 81 and (c) 161 x 161 grids

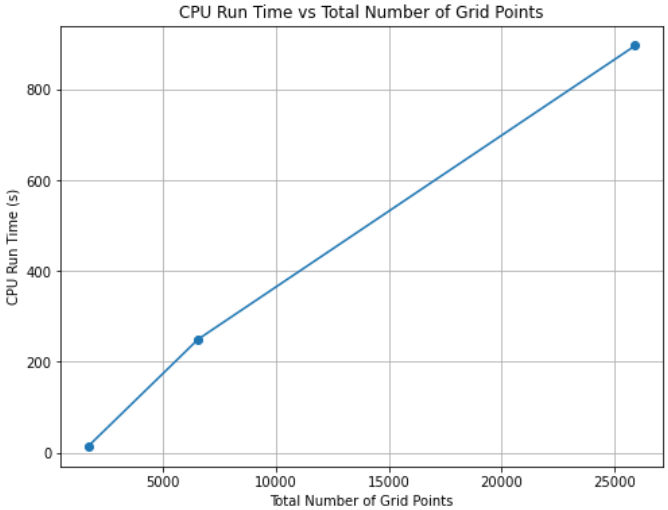


Figure 7: CPU Run time vs Total Number of Grid Points for Gauss Seidel Method

3.4. Discussion

In general, by increasing the number of grid points, the accuracy of the solution tends to increase. From the residual versus number of iterations(as shown in figure 6), the rate of convergence is decreasing with an increase in the number of grid points. There is an initial steep improvement in convergence with an increase in the number of grid points, which is then followed by a slower rate of convergence. As the distance between the grids is reduced, which is done by increasing the number of grid points, one can capture finer details of the physical process, and the solution becomes more accurate. There should, however, be a trade-off on the number of required grid points, as increasing the grid points for the sake of accuracy can be computationally expensive (in terms of memory and time).

In the CPU run time plot, as shown in figure 7, the run time increases almost linearly initially, but a noticeable variation is observed in the plot during the computation of 81x81 grid. This can be an indication of the increase in the complexity.

Gauss-Seidel scales more gradually as compared to the Gauss Elimination method, which suggests that it is better suited for systems involving larger number of equations. However, in certain specialized problems involving smaller system, it may be convenient to use Gauss Elimination than Gauss-Seidel, because of its direct nature. The gauss-Seidal method is more efficient than the Gauss Elimination method, and for larger systems, it is often used for obtaining numerical solutions due to its iterative nature. However, there are other methods which are even more efficient, that are discussed below.

Final Thoughts: *Gauss-Seidel is more efficient as compared to Gauss Elimination and being an iterative solver, it works best for systems involving larger number of equations to be solved at each grid point. However, it is still computationally inefficient when compared to ADI and Row Sweep methods.*

4. LINE-BY-LINE METHOD (ROW SWEEP)

4.1. Discretization of equations

The row sweep method utilizes the idea of processing each grid points row by row, solving the system of equations to get the tri-diagonal system of equations. This method is particularly advantageous when solving elliptic PDEs, where the solution at a particular grid point depends on the neighbouring grid points. This iterative method helps in efficiently propagating information across the solution domain, ensuring stability and convergence.

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

4.2. Solution Methodology

The coefficient matrix consists of a tri-diagonal matrix, with the main diagonal having values of -4 and the two off-diagonals on either side being 1. The coefficient matrix as well as the vector containing the source term is solved by using *tdma* function in the code.

4.3. Results

The contour plot of ϕ for the finest grid is shown below:

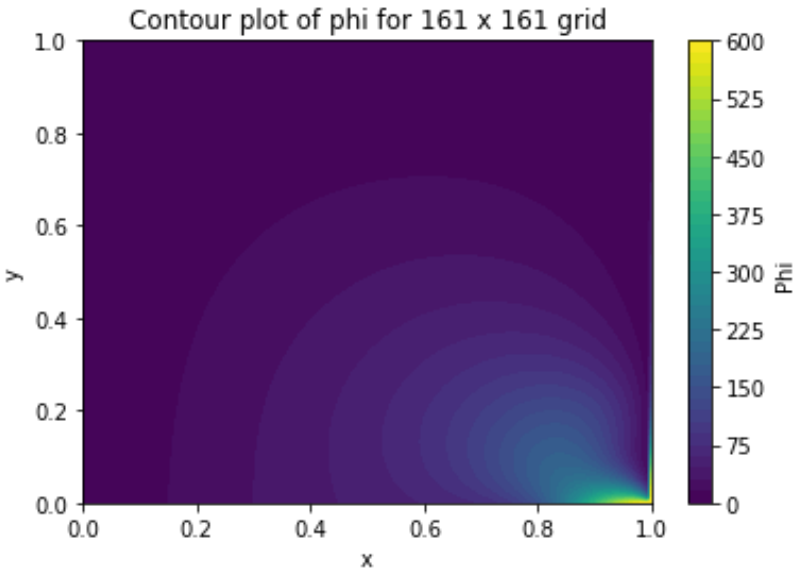
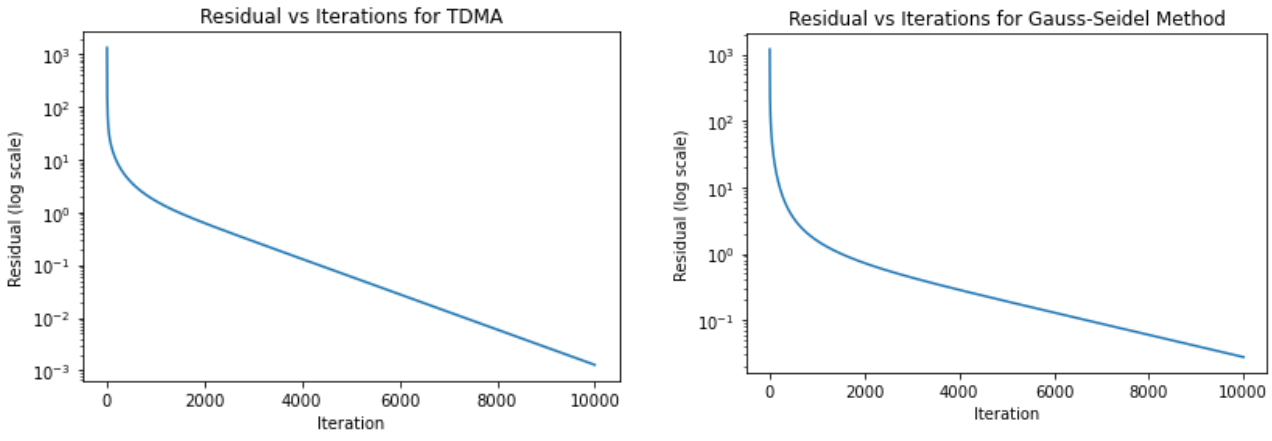


Figure 8: Contour plot of ϕ for 161 x 161 grid

The residual versus iterations for TDMA and for Gauss-Seidal is as shown:



(a) *(b)*

Figure 9: Residual vs Iterations for (a) TDMA and (b) Gauss Seidal Method for 161 x 161 grid

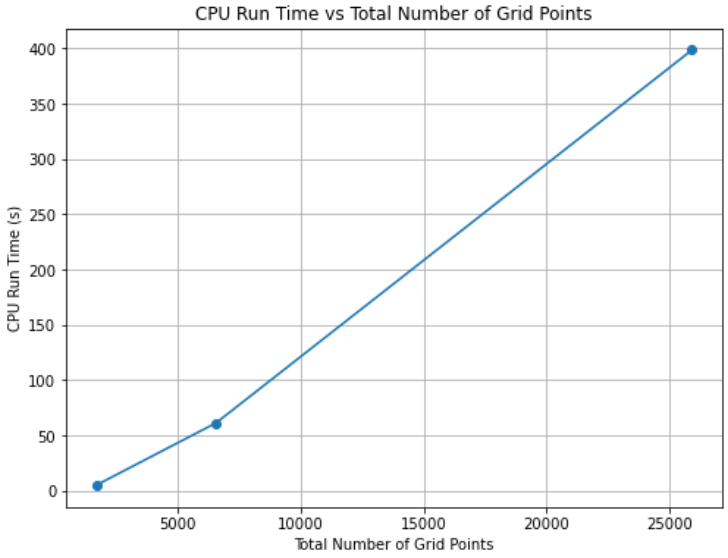


Figure 10: CPU Run Time vs Total Number of Grid Points for TDMA

4.4. Discussion

From the plots of residual versus number of iterations for the TDMA and the Gauss-Seidal method, it is evident that TDMA converges almost instantaneously which is evident from the steep drop in the plot. This makes it efficient for problems utilizing tri-diagonal structure, where it can make use of matrix characteristics to reduce the computation cost. After a certain point however, the slope diminishes with the residual per iteration becomes negligible. This highlights the fact that the method approaches a solution quickly, but subsequent iterations may not improve the accuracy of the solution. Gauss-Seidal, however, has a slower convergence rate as the number of iterations increases. The residuals as well undergoes reduction gradually, indicating that the solution may not converge quickly.

Final thoughts: *The Tri-Diagonal Matrix Algorithm is useful in solving systems having a large number of linear equations, with the computational efficiency being its advantage and converges at a faster rate than the Gauss Elimination and Gauss Seidal methods.*

5. ALTERNATING DIRECTION IMPLICIT (ADI) SCHEME

5.1. Discretization of equations

The Alternating Direction Implicit (ADI) scheme is a common methodology of solving elliptic PDEs adopted for solving linear system of equations in which there is an alternate row sweep, and a column sweep in each iteration. The discretization of the equations is similar to the above discretization that has been done earlier, using CDS scheme for second order derivatives, and is as follows:

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

5.2. Solution Methodology

Each sweep, both row and column, is carried out in one iteration. The coefficient matrix formed is then solved using the TDMA, using the *tdma* function in the code. The difference between the numerical and the analytical solution is compared in each loop iteration to check for convergence of solution.

5.3. Results

The contour plot for ϕ for 41 x 81 grids is as shown:

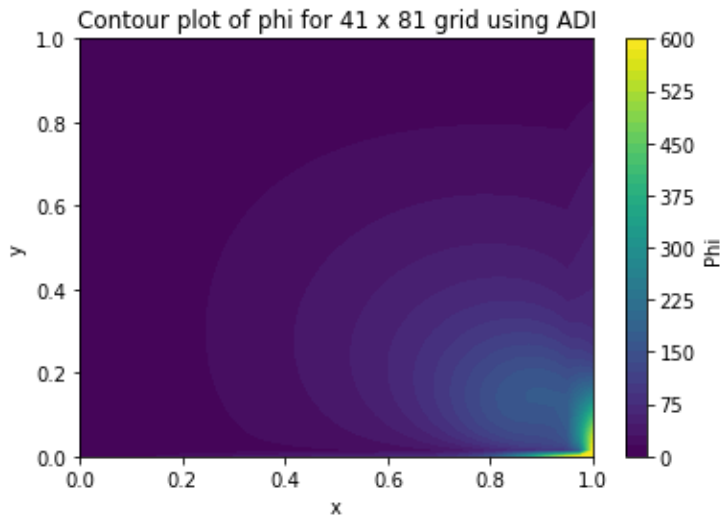


Figure 11: Contour Plot of ϕ for 41 x 81 grid

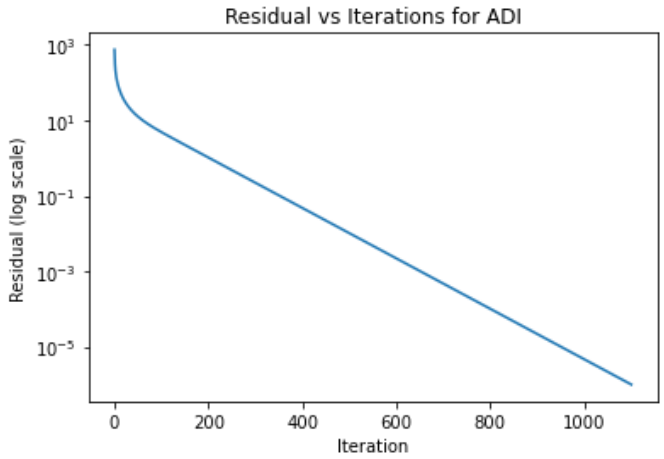


Figure 12: Residual vs Iterations for ADI

5.4. Discussions

In figure 12, the residual versus iterations for ADI is shown, which shows that initially, the solution starts to converge instantaneously but slows down at an earlier iteration than the TDMA. This can be attributed to the fact that there is a presence of high-frequency errors at higher iterations, which are difficult to eliminate. They cause the rate of convergence to slow down, but nonetheless, the residual error tends to decrease, but at a slower rate. The high frequency errors that creep up during the computation occurs as a result of the discretization process. These errors can be more significant at higher grid resolutions, when we require to capture physical process with greater accuracy. When TDMA is applied iteratively, the convergence behaviour tends to be more consistent. It can reduce these errors and can continue to converge at a steady rate after the initial high rate.

Final Thoughts: *Alternating Direction Implicit (ADI) methods are numerical techniques designed to work by breaking down PDEs into linear algebraic equations in a computationally efficient manner. The method balances between accuracy and efficiency by alternating across the row and column, converging to the solution at a faster rate. However, handling high-frequency errors may be an issue with this particular method, due to which the convergence rate decreases.*