ME 605 Computational Fluid Dynamics
Fall 2021-22

Instructor Prof. Dilip Sundaram

Project-2

Name - Devanshu Thakar

Roll no. – 1811017

Table of Contents

Problem Statement

Part A : 2D Steady-State Equation

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

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

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

$$\phi(0, y) = 1000 \left[\frac{1}{4} \sinh\left(-\frac{1}{2}\right) + \left(y - \frac{1}{2}\right)^2 \sinh\left(y - \frac{1}{2}\right) \right]$$

$$\phi(1, y) = 1000 \left[\frac{1}{4} \sinh\left(\frac{1}{2}\right) + \left(y - \frac{1}{2}\right)^2 \sinh\left(y - \frac{1}{2}\right) \right]$$

$$\phi(x, 0) = 1000 \left[\left(x - \frac{1}{2}\right)^2 \sinh\left(x - \frac{1}{2}\right) + \frac{1}{4} \sinh\left(-\frac{1}{2}\right) \right]$$

$$\phi(x, 1) = 1000 \left[\left(x - \frac{1}{2}\right)^2 \sinh\left(x - \frac{1}{2}\right) + \frac{1}{4} \sinh\left(\frac{1}{2}\right) \right]$$

The source term is given by :

$$S_\phi = 1000 \left[2 \sinh\left(x - \frac{1}{2}\right) + 4\left(x - \frac{1}{2}\right) \cosh\left(x - \frac{1}{2}\right) + \left(x - \frac{1}{2}\right)^2 \sinh\left(x - \frac{1}{2}\right) \right] + 1000 \left[2 \sinh\left(y - \frac{1}{2}\right) + 4\left(y - \frac{1}{2}\right) \cosh\left(y - \frac{1}{2}\right) + \left(y - \frac{1}{2}\right)^2 \sinh\left(y - \frac{1}{2}\right) \right]$$

For reference , the analytical solution is given below :

$$\phi(x, y) = 1000 \left[\left(x - \frac{1}{2}\right)^2 \sinh\left(x - \frac{1}{2}\right) + \left(y - \frac{1}{2}\right)^2 \sinh\left(y - \frac{1}{2}\right) \right]$$

1) Solve the 2D steady-state equation using Gaussian elimination method for the following grid sizes: 20, 40, 80, and 160 in each direction. Show the computed ϕ field as a contour plot for the finest grid. Plot the CPU time vs grid size and comment on the efficiency of the Gauss elimination method.

2) Solve the same equation using Jacobi and Gauss-Seidel iterative processes for the following three grid sizes: 40, 80, and 160 in each direction. Show the computed ϕ field as a contour plot for the finest grid. Plot the residual vs number of iterations for each of these cases in the same plot. Comment on convergence rates of Jacobi and Gauss-Seidel methods

3) Solve the same equation using line-by-line method for 80 × 80 grid for both row-wise sweep and column-wise sweep. Show the computed ϕ field as a contour plot for one of the sweeps. Plot residual vs number of iterations for the two cases in the same plot.

Repeat the same computations for 40×80 grid and plot the residual vs number of iterations for the two cases in the same plot. Discuss the trends.

4) Solve the same equation using ADI method for the following three grid sizes: 40, 80, and 160 in each direction. Show the computed ϕ field as a contour plot for the finest grid. Plot residual vs number of iterations for each of these cases in the same plot. Repeat the same computations for 40×80 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.

Part B : 2D Unsteady-State Equation

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

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

The initial condition is uniform $\phi(x, y)$= 0. The boundary conditions and source term remain the same.

(1) Solve the 2D unsteady state equation using the explicit scheme for a 40 ×40 grid. Investigate the effect of time step on the results.

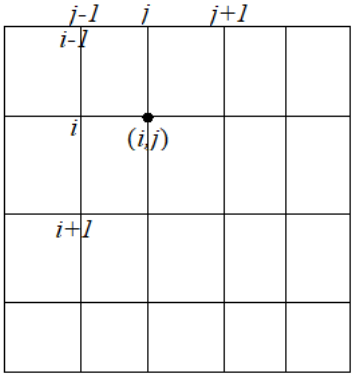
(2) Solve the 2D unsteady state equation using the implicit scheme for a 40 ×40 grid. How does the time step for the implicit scheme compare with the maximum possible time step in an explicit scheme?

(3) Solve the 2D unsteady state equation using the Crank-Nicolson scheme for a 40 × 40 grid.

(4) For one of the above three cases, show the evolution of ϕ field by presenting multiple contour plots corresponding to different times. Discuss the trends with physics based reasoning.

Part A : 2D Steady-State Equation

Let us take a grid in x and y direction. Suppose index i denote the i-th row in mesh and index j denote the j-th column, as shown in the grid below :



Approximating the second order partial derivative terms with a central difference scheme (CDS)

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

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

Since the grid is uniform $\Delta x = \Delta y = \Delta$ (say). Substituting the second order partial derivatives into the 2D-steady state equation

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

For a given i, j (i.e. the x, y coordinates from the grid) $S_{\phi_{i,j}}$ can be computed from the expression of the source term.

1. Direct Method : Gauss Elimination

1.1 Discretized equation

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

The discretized equation can be represented in Matrix for as

$$\boldsymbol{A\phi} = \boldsymbol{B}$$

Where $\boldsymbol{\phi}$ and $\boldsymbol{B}$ are a vector of length $(N-2)^2$ where N=number of grid points.

$\boldsymbol{A}$ is a Matrix of dimension $(N-2)^2 \times (N-2)^2$

1.2 Solution methodology

Each point in the grid will be an unknown in the vector $\boldsymbol{\phi}$**.** Therefore, the unknowns in 2D needs to be translated in 1D. All these is performed using the MATALB code PartA_Q1_GaussEli.m. This MATALB code has function translate, which performs the translation of unknown in 2D to unknown in 1D. The coefficients on the left side of equation-**(1)** are also translated similarly to form the matrix $\boldsymbol{A}$. After generating the Matrix $\boldsymbol{A}$, vector $\boldsymbol{B}$ and unknown vector $\boldsymbol{\phi}$ they are passed to a generic function *GaussEli.m* which performs the Gauss Elimination method.

1.3 Results

To compute the CPU time, the time to form the matrix and vectors is not considered also the time to transform the system from 2D to 1D and back to 2D from 1D is not considered. Only the time to solve the system of equation using Gauss Elimination is taken into account. The CPU time computed for different grid sizes is tabulated below :

Grid size	20	40	80	160
CPU time (s)	0.2656250	9.875000	1297.40625	61,227 .7737

Table 1 CPU versus grid size

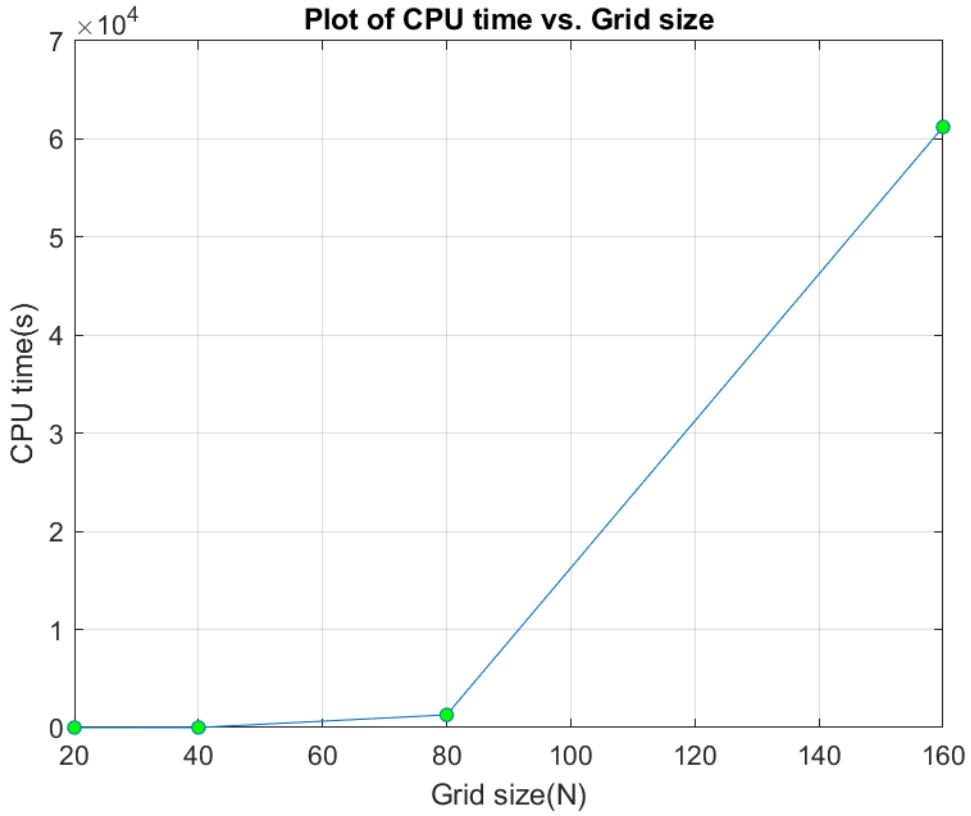


Figure 1 Plot of CPU time vs. grid size

The plot of ϕ for the finest grid using Gauss Elimination method is shown below :

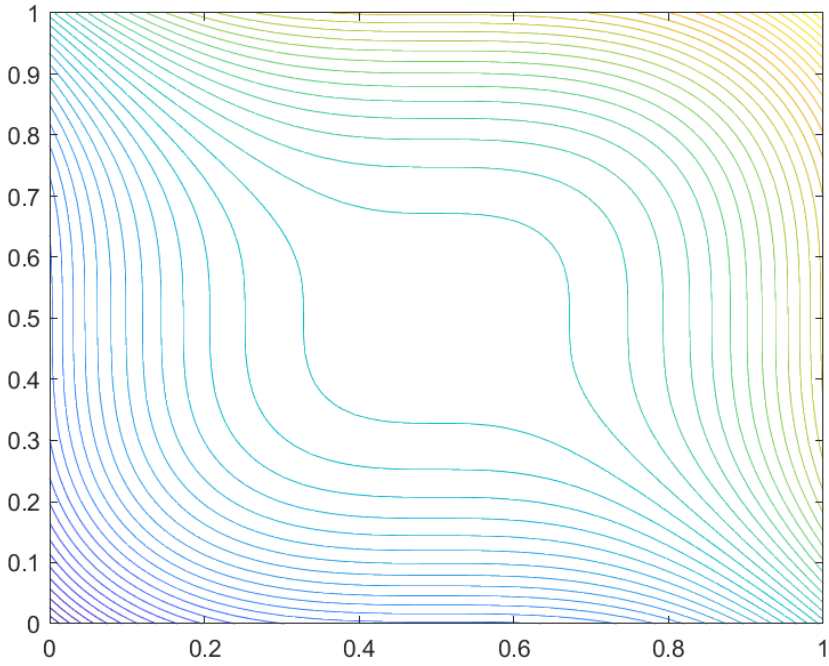


Figure 2 Plot of ϕ by Gauss Elimination for the Grid of 160×160

1.4 Discussion

For a grid size of N the totat number of unknowns and equations that needs to be solved are in the order of $O(N^2)$. The Gauss Elimination method scales as $O(n^3)$ where n is the number of unknowns. Therefore in a 2D-problem for a grid with N grid points in both x and y direction the Gauss Elimination methods scales as $O((N^2)^3) = O(N^6)$. Based on the assumption that the CPU of the computer is able to perform 10^8 operations per second, an approximate calculation for the time required for Gauss Elimination with grid size is carried out as follows.

For $N = 20$ the time taken will be approximately $\frac{20^6}{10^8} = 0.64$ s

For $N = 40$ the time $\frac{40^6}{10^8} = 40.96$ s

For $N = 80$ the time $\frac{80^6}{10^8} = 2621.44$ s

For $N = 160$ the time $\frac{160^6}{10^8} = 167772.16$ s $= 46.6$ hrs

From the above analysis it can be concluded that Gauss Elimination is applicable only for a small number of grid points. The scaling of Gauss Elimination method with the number of grid points is very inefficient. Gauss Elimination becomes unpractical as the number of gird points is increased. Therefore Gauss Elimination proves to be a highly inefficient method.

2. Iterative Methods

2.1 Jacobi Method

After the approximating the second order derivatives with a central Difference Scheme (CDS) the discretized expression is :

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

In Jacobi method, initially a guess is made for value of ϕ at all node points. Using that guess value, ϕ value are evaluated for next iteration.

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

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

2.2 Gauss Seidel

Gauss Seidel method is an iterative method which is very similar to Jacobi method. In Gauss Seidel, the new value of ϕ is used to compute ϕ at other points as soon as a value at that node is computed for the new iteration. The discretized equation for Gauss-Seidel method is similar to Jacobi method.

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

2.3 Solution methodology

The MATLAB code PartA_Q2_Jacobi.m is developed to perform the Jacobi method. The MATLAB code PartA_Q2_GaussSeidel.m is coded to compute ϕ using Gauss Seidel method. Both the code make use of two values of ϕ the current value and the previous value. The code keeps on switching between the current and previous values of ϕ as the iteration proceeds.

2.4 Computing the residual and stopping criteria

By using an iterative solution, the numerical solution will not be exactly equal to the analytical solution. In general if $\boldsymbol{\phi}$ is analytical solution and after n iterations we have computed the numerical solution say $\boldsymbol{\phi^n}$. The residual matrix $\boldsymbol{\rho}$ can be defined using the Matrix notation of the discretized equation as :

$$\boldsymbol{A\phi^n - A\phi = \rho}$$

$$\boldsymbol{A\phi^n - B = \rho} \qquad (\because A\phi = B)$$

In other words residual can be computed as difference between the left side of the steady state equation (the diffusion terms) and the right side terms (source term).

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j} - S_{\phi_{i,j}}(\Delta^2) = \rho \qquad \mathbf{(2)}$$

In the MATALB code the residual was computed as difference between the left hand side and right hand side of equation-**(1)** (i.e. between the diffusion and source term) over all node points. Therefore from equation –**(2)** we get a residual matrix named *resi* in the matlab code. The final residual is computed as the Euclidian norm of the residual matrix. The value of norm of residual

matrix was chosen as **stopping criteria** for both the iterative methods. The iterative method would terminate when the (norm of residual) $|\boldsymbol{\rho}| < \mathbf{10^{-5}}$.

2.5 Results

The contour plots of computed ϕ for the finest grid of 160 points is shown below :

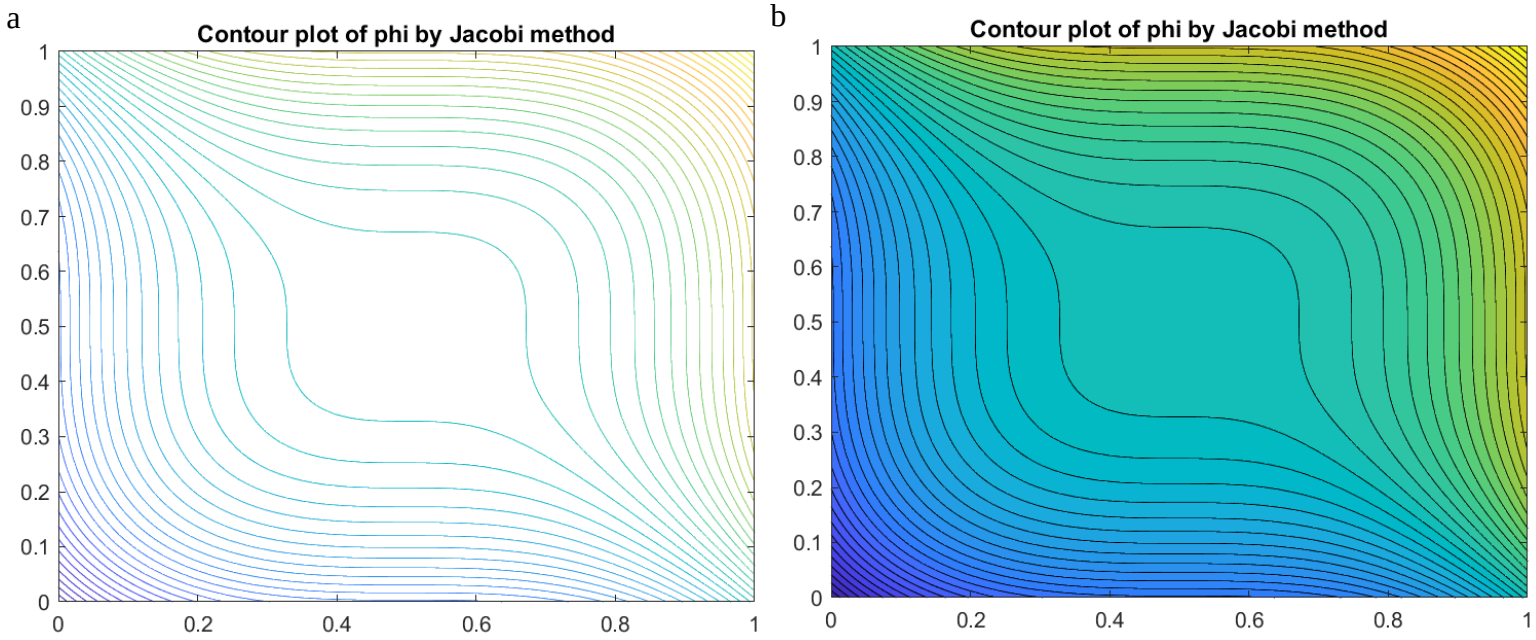


Figure 3 Contour plot of ϕ by Jacobi iterative method for grid size of 160 points (a) contour plot (b) filled contour plot

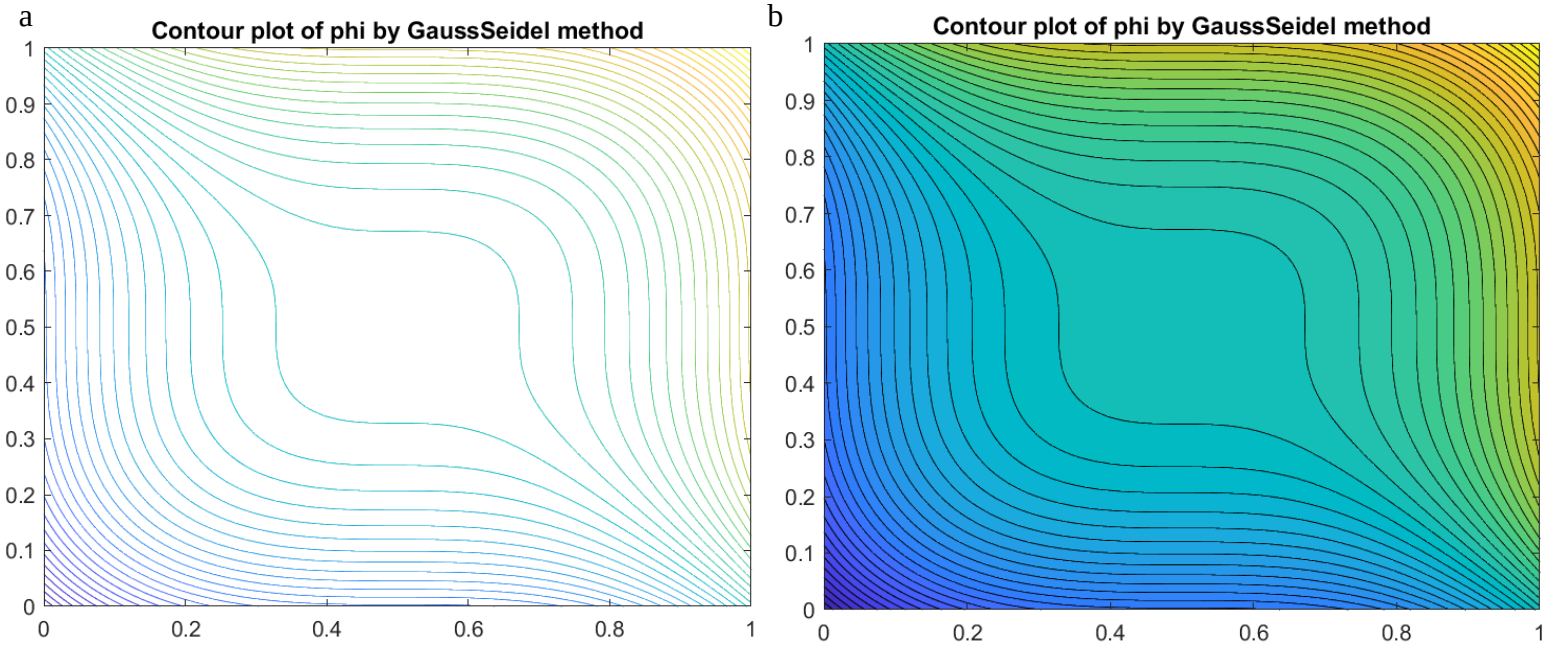


Figure 4 Contour plot of ϕ by Gauss Seidel iterative method for grid size of 160 points (a) contour plot (b) filled contour plot

The plot of residual versus the number of iterations for each of the grid sizes is shown below :

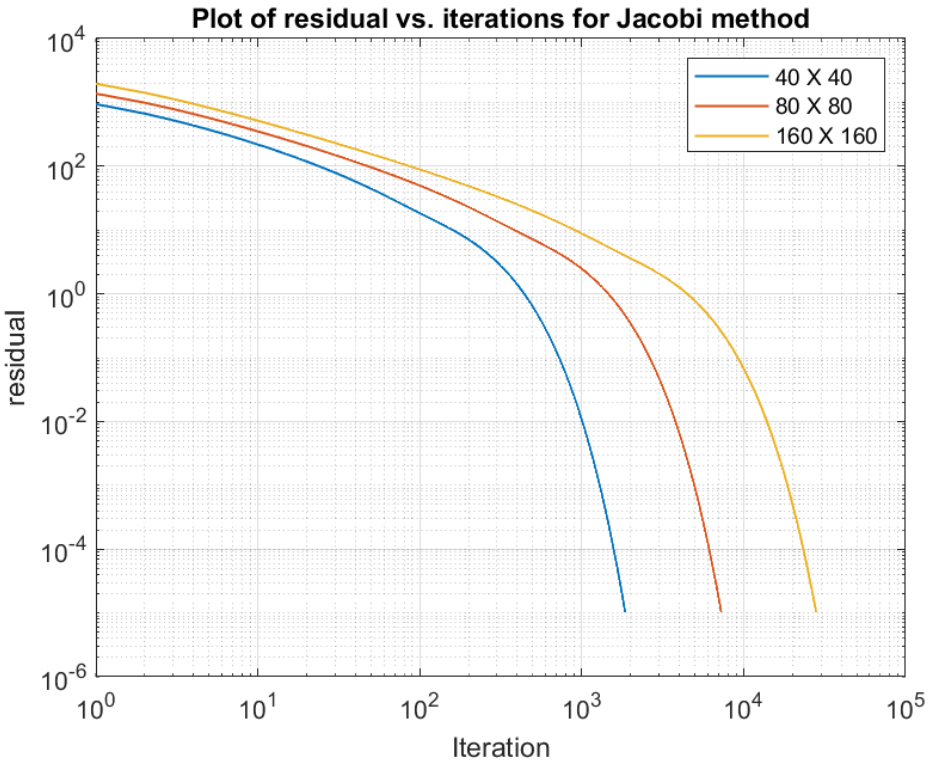


Figure 5 Plot of residual versus number of iterations computed by **Jacobi method**

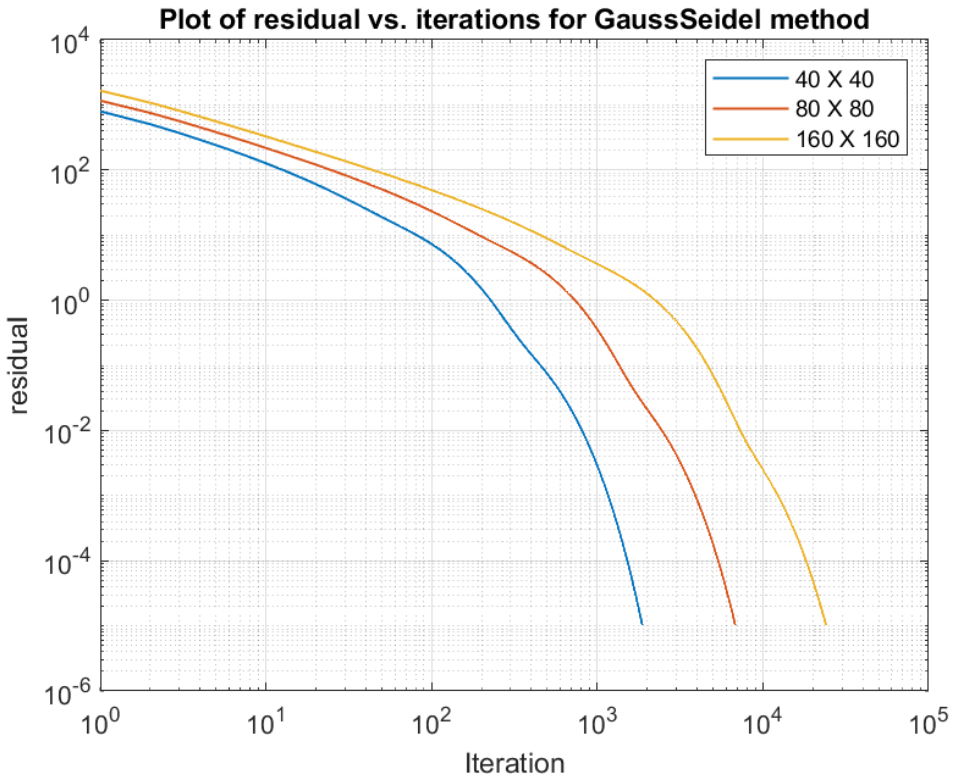


Figure 6 Plot of residual versus number of iterations computed by **Gauss Seidel method**

The following table tabulates the exact number of iterations taken by Jacobi and Gauss Seidel method for increasing grid sizes.

	Number of iterations and Method	
Grid Size	Jacobi	Gauss Seidel
40	1858	1872
80	7280	6818
160	28070	24089

Table 2 Number of iterations taken by Jacobi and Gauss Seidel method to converge a residual of $|\rho| < 10^5$

2.6 Discussion

Form the contour plots for the finest grid size of 160 for both methods Jacobi and Gauss Seidel, the solution converges to the desired tolerance. Contour plots from the both the methods are very similar, as expected.

From Figure 5 and Figure 6 it can observed that with an increase in the number of grid points the rate of convergence is decreasing for both Gauss-Seidel and Jacobi methods. In other words more number of iterations are required to achieve the same convergence criteria. From Figure 5 and Figure 6 and also from the Table 2, many interesting results are seen on comparing Jacobi and Gauss Seidel method. In Gauss Seidel method the value of the current iteration is used as soon as it is available, therefore it is expected that Gauss-Seidel would have faster convergence rate. For the grid size of 80 and 160, it is indeed true that Gauss Seidel has faster convergence rate. The case with grid size of 40 is interesting as Gauss Seidel method take slightly more number iterations than Jacobi method. One possible reason could be less number of grid points in grid size of 40. As the number of grid points are increased the performance of Gauss-Seidel method also increases and it takes lesser iteration to achieve convergence than Jacobi method.

3. Row-wise and Column-wise sweep

3.1 Row wise sweep for 80 X 80 grid

Index notations : $\phi_{i,j}$ indicates ϕ at i-th row and j-th column.

The discretization of the second derivatives is the same as shown in previous section. In row wise sweep the values of the previous iterations are used for the node points not belonging in the particular row. Doing a row-wise sweep we solve for particular row (i.e. fixed ith index) and march in direction of next columns to get the tri-diagonal system of equations :

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

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

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

The above system of equation can be solved using TDMA.

3.2 Column-wise sweep

The discretization of the second derivatives is the same as shown in previous section. In column wise sweep the values of the previous iterations are used for the node points not belonging to a particular column. Doing a column-wise sweep we solve for particular column(i.e. fixed jth index) and march in direction of next rows to get a tri-diagonal system of equations. For example for jth column the equations are :

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

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

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

The left hand side of the above equation produces a tridiagonal system of equation which can be solved by TDMA.

3.3 Results

The **convergence criteria** is kept the same as norm of residual matrix $|\rho| < 10^{-5}$

The contour plot of ϕ for 80 × 80 grid computed using a row sweep is shown below :

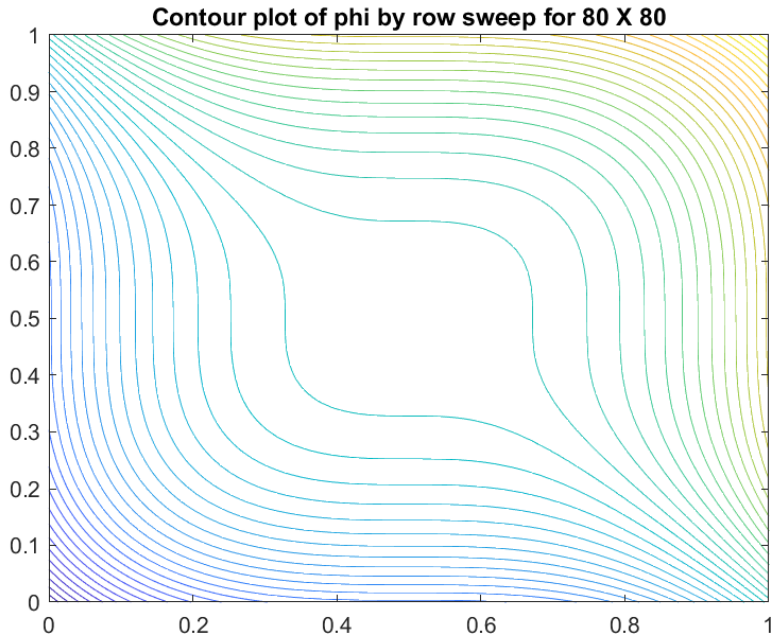


Figure 7 Contour plot of ϕ by row sweep for a grid size of 80 × 80

The plot of residual with number of iteration for both row and column sweep is shown in Figure 8.

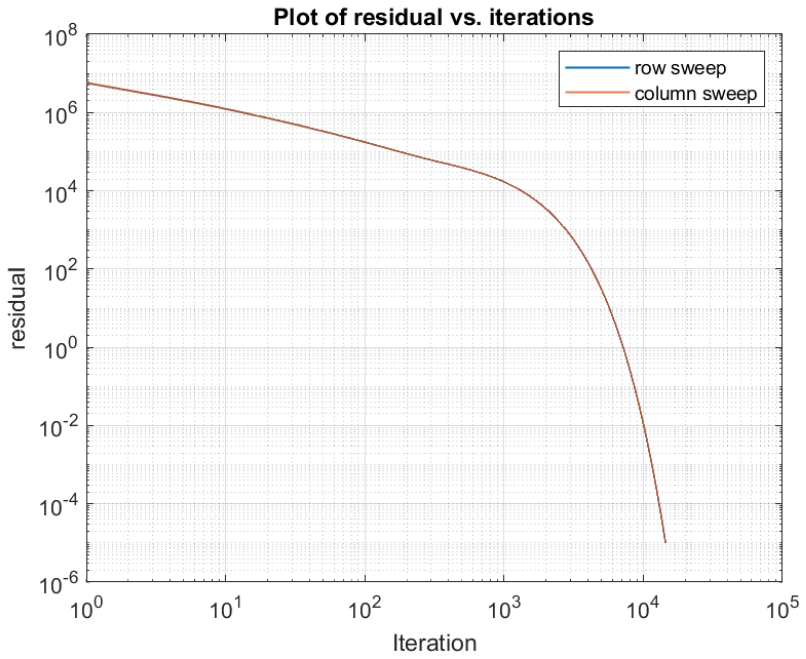


Figure 8 Plot of residual versus number of iteration for 80 × 80 grid size

The same plots for 40 × 80 grid are shown below :

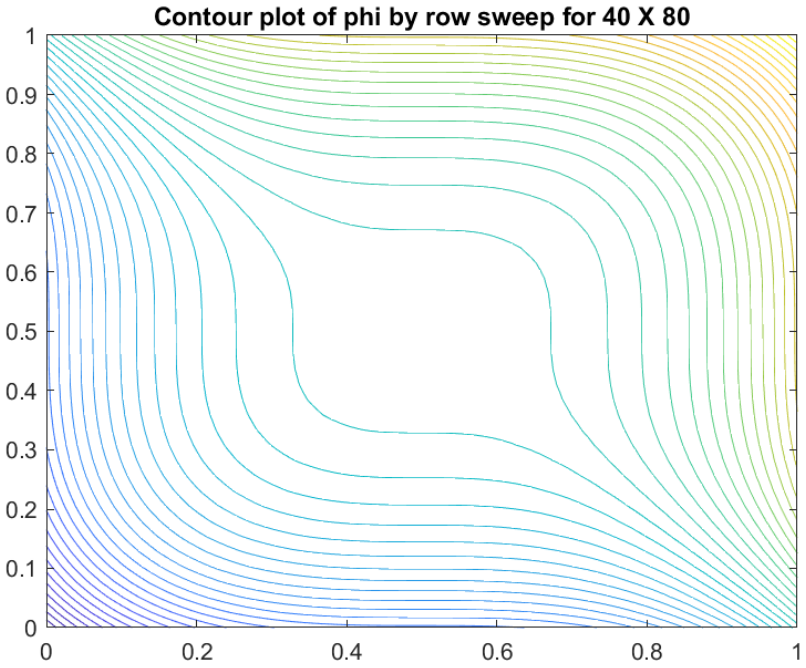


Figure 9 Contour plot of ϕ by **row sweep** for a grid size of 40 × 80

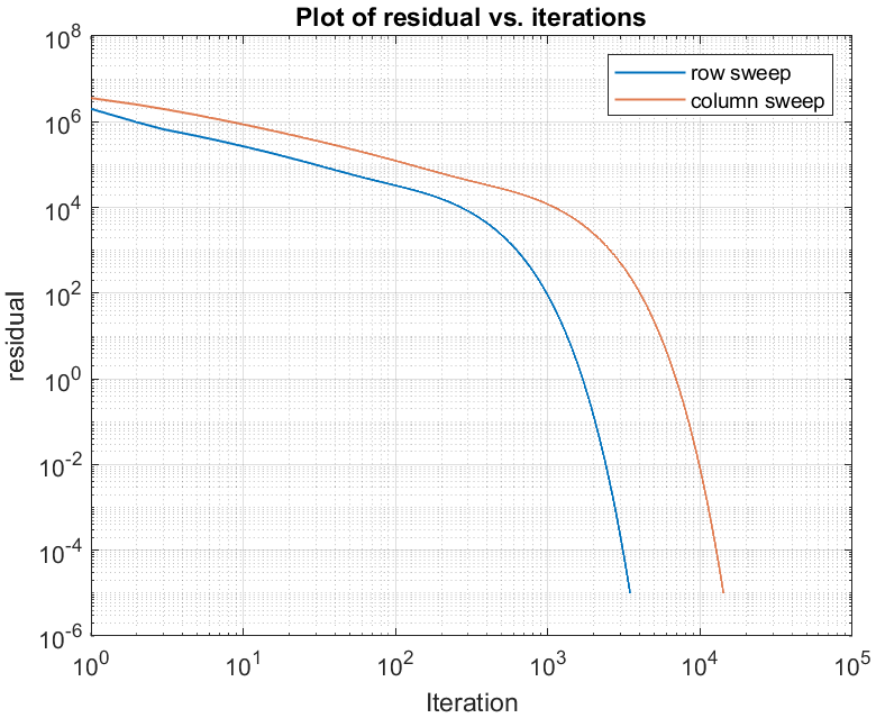


Figure 10 Plot of residual versus number of iteration for 40 × 80 grid size

3.4 Discussion

From the Figure 8 it is observed that the rate of convergence is same for row wise and column wise sweep for a grid size of 40×40. This is expected as the arrangement of grid points is symmetric in x and y direction.

For a non-uniform grid of size 40 × 80 the plot of residual with iteration is shown in Figure 10. The convergence rate row-wise sweep is higher than column-wise sweep. This is because in row sweep one needs to iterate for total of 40 rows while simultaneously solving for 80 unknows in each row sweep. On the other hand in column sweep there are total 80 columns needs to be iterated while simultaneously solving 40 unknowns in each column sweep. The distance need to be covered for one iteration is higher in case of column sweep and lower for the case of row sweep. Row sweep solves more unknowns than the column sweep in a single sweep or iteration. Therefore the row sweep method achieves faster convergence.

It is interesting to note that when there is a non-uniform grid one should choose a sweep in that direction in which more unknowns are solved during a single sweep. This would help the method to converge faster.

4. ADI

4.1 ADI Methodology

ADI stands for Alternating Direction Implicit (ADI) in which there is alternate row sweep and column sweep. In implementing the ADI method each sweep either row or column is considered to be one iteration. Assuming that the first iteration is a row sweep followed by a column sweep and alternating between row and column sweep with iterations. The ADI method was implemented in the MATLAB code PartA_Q4_ADI.m.

The discretization of the uniform equation is similar to that in previous section using a CDS scheme for second order derivatives.

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

Row sweep :

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

The above row sweep equations forms a tridiagonal matrix, which can be efficiently solved by a TDMA solver.

Column sweep :

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

The above column sweep equations forms a tridiagonal matrix, which can be efficiently solved by a TDMA solver.

4.2 Solution methodology

The MATLAB code PartA_Q4_ADI.m was coded to implement the ADI method. The code uses a function *ThomasAlgo* to solve the system of tridiagonal equation formed during the row and column sweep. Each full sweep either row or column is consider to be one iteration.

4.3 Results

The plot of residual with the number of iteration for the grid sizes of 40, 80 and 160 by ADI method is shown below :

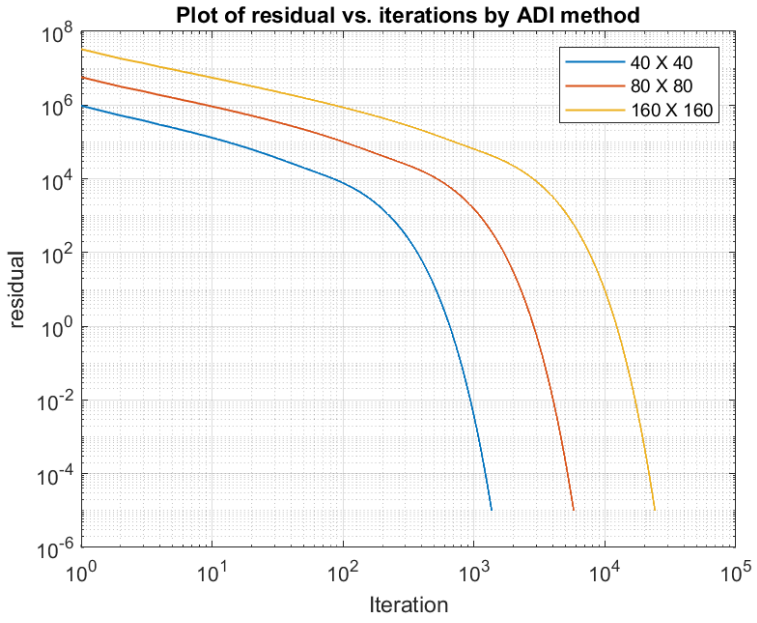


Figure 11 Plot of residual versus number of iterations computed by **ADI** method

The contour plot for the finest grid computed using ADI method is shown in Figure 12.

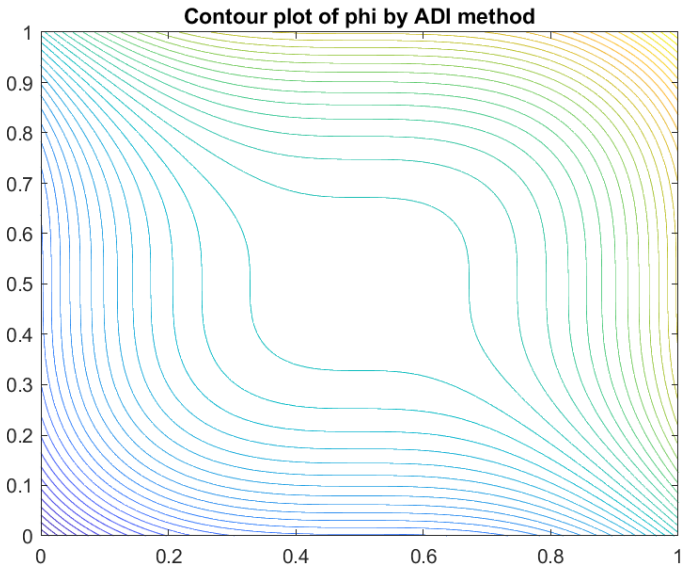


Figure 12 Contour plot of ϕ by **ADI** method for a grid size of 160 × 160

4.3 For 40×80 grid by row sweep, column sweep and ADI

The plot of residual versus number of iteration for a grid of 40 × 80 by a row-wise sweep, a column-wise sweep and by ADI method is shown in Figure13.

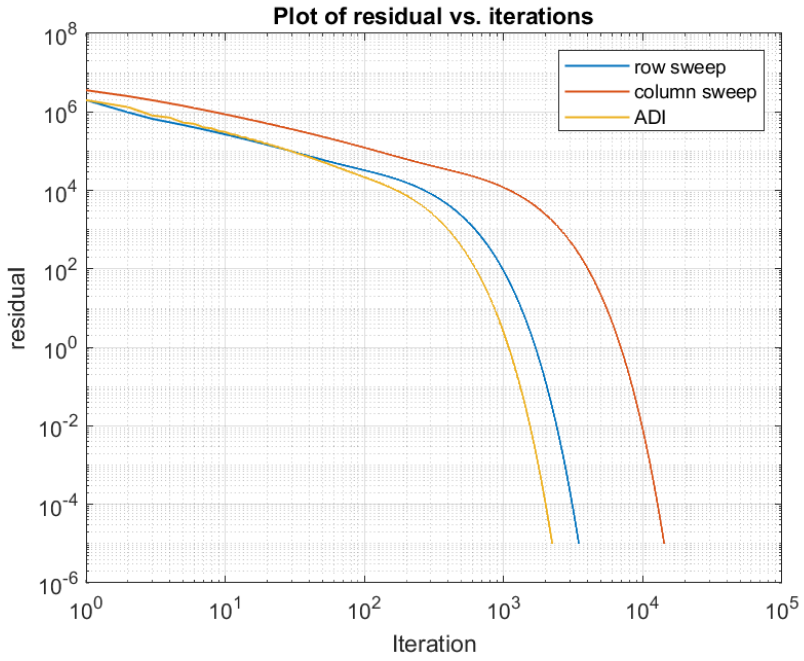


Figure 13 Plot of residual with iterations by different methods for 40×80 grid

Here the ADI method and row sweep are giving quite similar convergence rate. To get further insights on the comparison of ADI and row sweep method, ϕ was computed using row-wise sweep and ADI method for a grid of size 40×160. The plot of residual with number of iteration is shown in Figure 13.

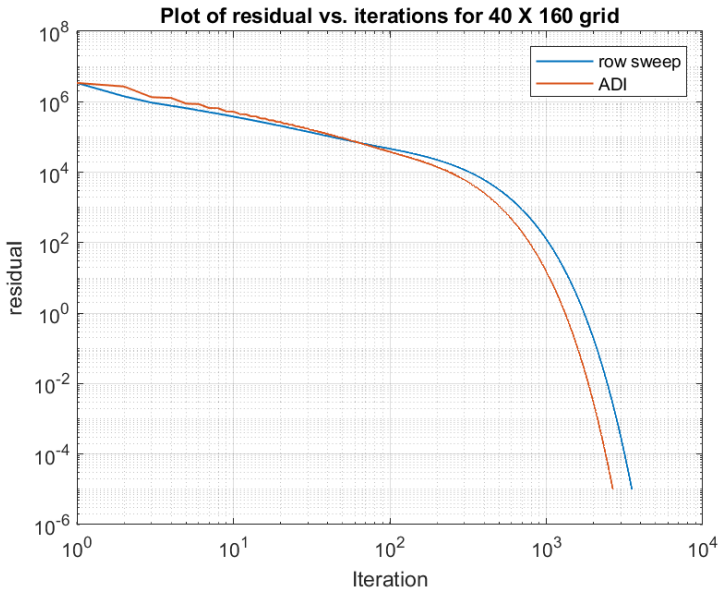


Figure 14 Plot of residual with iterations for 40×160 grid

4.4 Discussion

From the Figure 11 it can concluded that as the number of grid points is increased the rate of convergence decreases. In other words more number of iteration are required to achieve the same level of accuracy with increasing grid size.

Figure 13 compares the residual versus the number of iterations for three methods row-wise sweep, column-wise sweep and ADI method for a grid of size 40×80. It can be concluded that ADI method gives faster convergence than row-wise sweep, both of them are significantly faster than the column sweep. The ADI and row-wise sweep methods are expected to have faster convergence, since they employ row wise sweeping and for 40×80 grid size more unknowns are solved simultaneously in particular row sweep. In other words less distance (number of rows) has to be travelled to solve for all the unknows.

There are two effects that leads to the convergence of methods. One is how many points are getting solved simultaneously or what is the path length that has to be covered to solve for the whole grid. The second one is how nodes contribute to the information propagation from the boundary nodes to interior nodes. ADI method uses column sweep half the number of times whereas row-wise sweep method uses only row sweep. Although the row sweep method solves more number of unknowns simultaneously, the information propagation from the boundary nodes happens only in one particular direction. Whereas in ADI though it uses column sweep in its iteration, the information from boundary travels from both the directions, which helps ADI method to achieve faster convergence than row-wise sweep. To further consolidate this observation the plot of residual versus iteration for 40×160 grid by row-wise sweep and ADI method is shown in Figure 14. In this case also the ADI methods achieves slightly faster convergence that row-wise sweep.

Part B : 2D Unsteady-State Equation

The unsteady 2D diffusion equation is give as :

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

The initial condition is uniform $\phi(x, y)$= 0. The boundary conditions and source term remain the same.

Since a range of time is not specified in the problem the unsteady-state equation was solved from t=0 s to t=10 s, with a chosen time step Δt.

1. Explicit scheme

1.1 Discretized equations

This is a time dependent problem and ϕ is now a function of three variables x, y and t $\phi(t, x, y)$. On integrating the right side of the unsteady state equation for fixed x and y

$$\int_{t_n}^{t_{n+1}} \frac{\partial \phi}{\partial t} dt = \int_{t_n}^{t_{n+1}} f(\phi, t)\, dt$$

Where $f(\phi, t)$= $\frac{\partial \phi}{\partial t}$ = $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y} - S_\phi$

$$\therefore \phi^{n+1} = \phi^n + \int_{t_n}^{t_{n+1}} f(\phi, t)\, dt$$

Using Taylor series expansion of ϕ^{n+1}

$$\phi^{n+1} = \phi^n + \left(\frac{d\phi}{dt}\right)_n (t_{n+1} - t_n) + O(\Delta t^2)$$

$$\phi^{n+1} = \phi^n + f(\phi^n, t_n)\Delta t + O(\Delta t^2)$$

Comparing equating (…) and (…) , we get the explicit Euler scheme

$$\phi^{n+1} \cong \phi^n + f(\phi^n, t_n)\Delta t + O(\Delta t^2)$$

$$\therefore \ \phi^{n+1} = \phi^n + f(\phi^n, t_n)\Delta t$$

Now $f(\phi^n, t_n)$ = $\frac{\partial \phi}{\partial t}$ can be expressed using the Central difference scheme in the second order partial derivatives of ϕ with x and y

$$f(\phi^n, t_n) = \frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y} - S_\phi$$

$$\therefore \ f(\phi^n, t_n) = \frac{\phi^n_{i+1,j} + \phi^n_{i-1,j} - 2\phi^n_{i,j}}{(\Delta \mathrm{x})^2} + \frac{\phi^n_{i,j+1} + \phi^n_{i,j-1} - 2\phi^n_{i,j}}{(\Delta \mathrm{y})^2} - S_\phi$$

On substituting the above expression in explicit Euler scheme we get

$$\phi^{n+1} = \phi^n + \frac{\phi^n_{i+1,j} + \phi^n_{i-1,j} - 2\phi^n_{i,j}}{(\Delta \mathrm{x})^2}\Delta t + \frac{\phi^n_{i,j+1} + \phi^n_{i,j-1} - 2\phi^n_{i,j}}{(\Delta \mathrm{y})^2}\Delta t - S_\phi \Delta t$$

The ϕ^{n+1} and ϕ^{n} are for a particular grid point (i, j)

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

1.2 Solution methodology and results

The MATLAB program PArtB_Q1.m was coded to implement the explicit scheme. The time dependent problem was solved for initial time t=0 s to t = 10 s. The problem was experimented with different values of time step Δt as tabulated in Table 3. The Error is the norm of the error matrix obtained by subtracting the ϕ at current time step and the previous time step.

Δt	Error	
0.01	∞	Diverged
0.001	∞	Diverged
0.0001	8.8×10^{-14}	Converged
0.00016	2.0×10^{-13}	Converged
0.000163	2.26×10^{-13}	Converged
0.00017	∞	Diverged

Table 3 Effect of changing Δt on explicit method

The maximum possible time step possible to ensure the convergence of explicit method comes out approximately as Δt=0.000163 s.

1.3 Stability Analysis

By using the stability analysis for 1D-covection diffusion problem, a similar analysis can be done for 2D problem.

$$G = 1 + 2D[\cos(K_m \Delta x) - 1] + 2D[\cos(K_m \Delta y) - 1]$$

$$G = 1 - 2D\left[2\sin^2\frac{K_m \Delta x}{2}\right] - 2D\left[2\sin^2\frac{K_m \Delta y}{2}\right]$$

$$G = 1 - 8D\left[\sin^2\frac{K_m \Delta}{2}\right]$$

$$8D < 2$$

$$D < \frac{1}{4} \Rightarrow \frac{\Delta t}{\Delta x^2} < \frac{1}{4}$$

$$\therefore \Delta t < \frac{\Delta^2}{4} = \frac{0.025^2}{4} = 0.000164$$

1.4 Discussion

From the above analysis it becomes clear that explicit method becomes unstable for larger time steps. A very small time step is needed for the explicit method to converge. This will lead to a high computational time for the explicit methods. The results concludes that explicit method is conditionally stable for convection-diffusion equation.

2. Implicit scheme

2.1 Discretized equations

From the above section the expression of ϕ^{n+1} can be written as :

$$\phi^{n+1} = \phi^n + \int_{t_n}^{t_{n+1}} f(\phi, t)\, dt$$

where $f(\phi, t) = \frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y} - S_\phi$

Taylor series expansion of $f(\phi, t)$ at t_{n+1}

$$f(\phi, t) = f(\phi, t_{n+1}) + f'(\phi, t_{n+1})(t - t_{n+1}) + \cdots$$

On integrating from t_n to t_{n+1}

$$\int_{t_n}^{t_{n+1}} f(\phi, t)\, dt = f(\phi, t_{n+1})\Delta t + f'(\phi, t_{n+1})\frac{\Delta t^2}{2} + \cdots$$

$$\int_{t_n}^{t_{n+1}} f(\phi, t)\, dt \cong f(\phi, t_{n+1})\Delta t + O(\Delta t^2)$$

$$\therefore\ \phi^{n+1} = \phi^n + f(\phi^{n+1}, t_{n+1})\Delta t$$

Now $f(\phi^{n+1}, t_{n+1}) = \frac{\partial \phi}{\partial t}$ can be expressed using the Central difference scheme in the second order partial derivatives of ϕ with x and y

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

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

For a 40 × 40 grid $\Delta x = \Delta y = \Delta$. On rearranging the above equation becomes :

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} = \frac{\phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} - 2\phi_{i,j}^{n+1}}{(\Delta)^2} + \frac{\phi_{i,j+1}^{n+1} + \phi_{i,j-1}^{n+1} - 2\phi_{i,j}^{n+1}}{(\Delta)^2} - S_\phi$$

2.2 Solution methodology

The above equation can be solved using a time dependent ADI scheme. Doing a row sweep from time t_n to $t_{n+1/2}$ and a column sweep from $t_{n+1/2}$ to t_{n+1}

Row sweep

$$\frac{\phi_{i,j}^{n+1/2} - \phi_{i,j}^n}{\Delta t/2} = \frac{\phi_{i+1,j}^n + \phi_{i-1,j}^n - 2\phi_{i,j}^{n+1/2}}{(\Delta)^2} + \frac{\phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2} - 2\phi_{i,j}^{n+1/2}}{(\Delta)^2} - S_\phi$$

$$\therefore\ \phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2} - \left(4 + \frac{\Delta^2}{\Delta t/2}\right)\phi_{i,j}^{n+1/2} = S_\phi \Delta^2 - \frac{\Delta^2}{\Delta t/2}\phi_{i,j}^n - \phi_{i+1,j}^n - \phi_{i-1,j}^n$$

The above equation system of equations forms a tri-diagonal matrix which can be efficiently solved by TDMA algorithm.

Column sweep

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n+1/2}}{\Delta t/2} = \frac{\phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} - 2\phi_{i,j}^{n+1}}{(\Delta)^2} + \frac{\phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2} - 2\phi_{i,j}^{n+1}}{(\Delta)^2} - S_\phi$$

$$\therefore \phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} - \left(4 + \frac{\Delta^2}{\Delta t/2}\right)\phi_{i,j}^{n+1} = S_\phi \Delta^2 - \frac{\Delta^2}{\Delta t/2}\phi_{i,j}^{n+1/2} - \phi_{i,j+1}^{n+1/2} - \phi_{i,j-1}^{n+1/2}$$

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

The above equation system of equations forms a tri-diagonal matrix which can be efficiently solved by TDMA algorithm.

There is no issue of convergence in time-dependent problems, as we are not guessing values at any of the step. Iterations in ADI are needed for steady state problems. Iterations are needed when we are guessing some values at some nodes. In transient (time-dependent problem) iterations are not be required.

2.3 Results

The MATLAB code PartB_Q2.m is coded to implement the implicit method. The code uses ADI scheme to solve the implicit set of equations as described in the previous section. The time dependent problem was solved for initial time t=0 s to t = 10 s. The Error is computed as the norm of error matrix generated by taking a difference between the current value of ϕ and previous values of ϕ at all the grid points.

Δt	Error	
0.1	1.072	Converged
0.01	1.6015×10^{-12}	Converged
0.001	0	Converged
0.0001	0	Converged

Table 4 Effect of changing Δt on implicit method

2.4 Discussion

For the convection-diffusion equation the implicit method is unconditionally stable. Therefore even on increasing the time step the method converges. This is the significant advantage of implicit method. Though the programming of an implicit method is difficult it generally gives faster convergence than the explicit method.

The maximum time step possible in explicit method was 0.0001 s and for the implicit method the time step as large as 0.1 is found to give accurate results. Therefore, by using implicit method we could talk 1000 (0.1/0.0001) times larger time step than explicit method. Thus, implicit method are 1000 times faster than the explicit method in this problem.

3. Crank Nicolson scheme

3.1 Discretized equations

From the above section the expression of ϕ^{n+1} can be written as :

$$\phi^{n+1} = \phi^n + \int_{t_n}^{t_{n+1}} f(\phi, t)\, dt$$

Using a trapezoidal formula to approximate the integral

$$\phi^{n+1} = \phi^n + \frac{1}{2}[f(\phi^n, t_n) + f(\phi^{n+1}, t_{n+1})]\Delta t$$

Substituting the Central difference approximation for the diffusive terms in expression of $f(\phi, t) = \frac{\partial \phi}{\partial t} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y} - S_\phi$

$$\phi^{n+1} = \phi^n + \frac{1}{2}\left[\frac{\phi^n_{i+1,j} + \phi^n_{i-1,j} - 2\phi^n_{i,j}}{(\Delta)^2} + \frac{\phi^n_{i,j+1} + \phi^n_{i,j-1} - 2\phi^n_{i,j}}{(\Delta)^2} - S_\phi + \frac{\phi^{n+1}_{i+1,j} + \phi^{n+1}_{i-1,j} - 2\phi^{n+1}_{i,j}}{(\Delta)^2} + \frac{\phi^{n+1}_{i,j+1} + \phi^{n+1}_{i,j-1} - 2\phi^{n+1}_{i,j}}{(\Delta)^2} - S_\phi\right]\Delta t$$

$$\phi^{n+1}_{i,j} = \phi^n_{i,j} + \frac{\Delta t}{2\,\Delta^2}\left[\phi^n_{i+1,j} + \phi^n_{i-1,j} + \phi^n_{i,j+1} + \phi^n_{i,j-1} - 4\phi^n_{i,j} + \phi^{n+1}_{i+1,j} + \phi^{n+1}_{i-1,j} + \phi^{n+1}_{i,j+1} + \phi^{n+1}_{i,j-1} - 4\phi^{n+1}_{i,j}\right] - S_\phi \Delta t$$

3.2 Solution Methodology

The above equation can be solved using a time dependent ADI scheme. Doing a row sweep from time t_n to $t_{n+1/2}$ and a column sweep from $t_{n+1/2}$ to t_{n+1}

Row sweep

$$\frac{\phi^{n+1/2} - \phi^n}{\Delta t/2} = \frac{1}{2}\left[\frac{\phi^n_{i+1,j} + \phi^n_{i-1,j} - 2\phi^n_{i,j}}{(\Delta)^2} + \frac{\phi^n_{i,j+1} + \phi^n_{i,j-1} - 2\phi^n_{i,j}}{(\Delta)^2} - S_\phi + \frac{\phi^n_{i+1,j} + \phi^n_{i-1,j} - 2\phi^{n+1/2}_{i,j}}{(\Delta)^2} + \frac{\phi^{n+1/2}_{i,j+1} + \phi^{n+1/2}_{i,j-1} - 2\phi^{n+1/2}_{i,j}}{(\Delta)^2} - S_\phi\right]$$

$$\phi^{n+1/2}_{i,j} = \phi^n_{i,j} + \frac{\Delta t}{4\,\Delta^2}\left[\phi^n_{i+1,j} + \phi^n_{i-1,j} + \phi^n_{i,j+1} + \phi^n_{i,j-1} - 4\phi^n_{i,j} + \phi^n_{i+1,j} + \phi^n_{i-1,j} + \phi^{n+1/2}_{i,j+1} + \phi^{n+1/2}_{i,j-1} - 4\phi^{n+1/2}_{i,j}\right] - S_\phi \Delta t/2$$

$$\therefore \frac{\Delta t}{4\,\Delta^2}\phi^{n+1/2}_{i,j+1} + \frac{\Delta t}{4\,\Delta^2}\phi^{n+1/2}_{i,j-1} - \left(\frac{\Delta t}{\Delta^2} + 1\right)\phi^{n+1/2}_{i,j} = S_\phi \Delta t/2 - \phi^n_{i,j} - \frac{\Delta t}{4\,\Delta^2}\left[2\phi^n_{i+1,j} + 2\phi^n_{i-1,j} + \phi^n_{i,j+1} + \phi^n_{i,j-1} - 4\phi^n_{i,j}\right]$$

$$\phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2} - \left(4 + \frac{4\Delta^2}{\Delta t}\right)\phi_{i,j}^{n+1/2} = 2S_\phi\Delta^2 - \frac{4\Delta^2}{\Delta t}\phi_{i,j}^{n} - \left[2\phi_{i+1,j}^{n} + 2\phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 4\phi_{i,j}^{n}\right]$$

A tri-diagonal system of equation which can be solved by TDMA

Column sweep

$$\frac{\phi^{n+1} - \phi^{n+1/2}}{\Delta t/2} = \frac{1}{2}\left[\frac{\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} - 2\phi_{i,j}^{n}}{(\Delta)^2} + \frac{\phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 2\phi_{i,j}^{n}}{(\Delta)^2} - S_\phi + \frac{\phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} - 2\phi_{i,j}^{n+1}}{(\Delta)^2} + \frac{\phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2} - 2\phi_{i,j}^{n+1}}{(\Delta)^2} - S_\phi\right]$$

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n+1/2} + \frac{\Delta t}{4\,\Delta^2}\left[\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 4\phi_{i,j}^{n} + \phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2} - 4\phi_{i,j}^{n+1}\right] - S_\phi\Delta t/2$$

$$\therefore \frac{\Delta t}{4\,\Delta^2}\phi_{i-1,j}^{n+1} + \frac{\Delta t}{4\,\Delta^2}\phi_{i+1,j}^{n+1} - \left(\frac{\Delta t}{\Delta^2} + 1\right)\phi_{i,j}^{n+1} = S_\phi\Delta t/2 - \phi_{i,j}^{n+1/2} - \frac{\Delta t}{4\,\Delta^2}\left[\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 4\phi_{i,j}^{n} + \phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2}\right]$$

$$\therefore \phi_{i-1,j}^{n+1} + \phi_{i+1,j}^{n+1} - \left(4 + \frac{4\Delta^2}{\Delta t}\right)\phi_{i,j}^{n+1} = 2S_\phi\Delta^2 - \frac{4\Delta^2}{\Delta t}\phi_{i,j}^{n+1/2} - \left[\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 4\phi_{i,j}^{n} + \phi_{i,j+1}^{n+1/2} + \phi_{i,j-1}^{n+1/2}\right]$$

A tri-diagonal system of equation which can be solved by TDMA

3.3 Results

The MATLAB code PartB_Q3.m is coded to implement the Crank Nicolson method. The code uses ADI scheme to solve the implicit set equations as described in the previous section. The time dependent problem was solved for initial time t=0 s to t = 10 s. The Error is computed as the norm of error matrix generated by taking a difference between the current value of ϕ and previous values of ϕ at all the grid points. If the error tends to zero than it can be said that system has reached steady state.

Δt	Error	
0.1	4×10^{11}	Diverges
0.01	2×10^{-4}	Converges

0.001	3.8×10^{-13}	Converges
0.0001	7.33×10^{-14}	Converges

Table 5 Effect of changing Δt on Crank Nicolson method

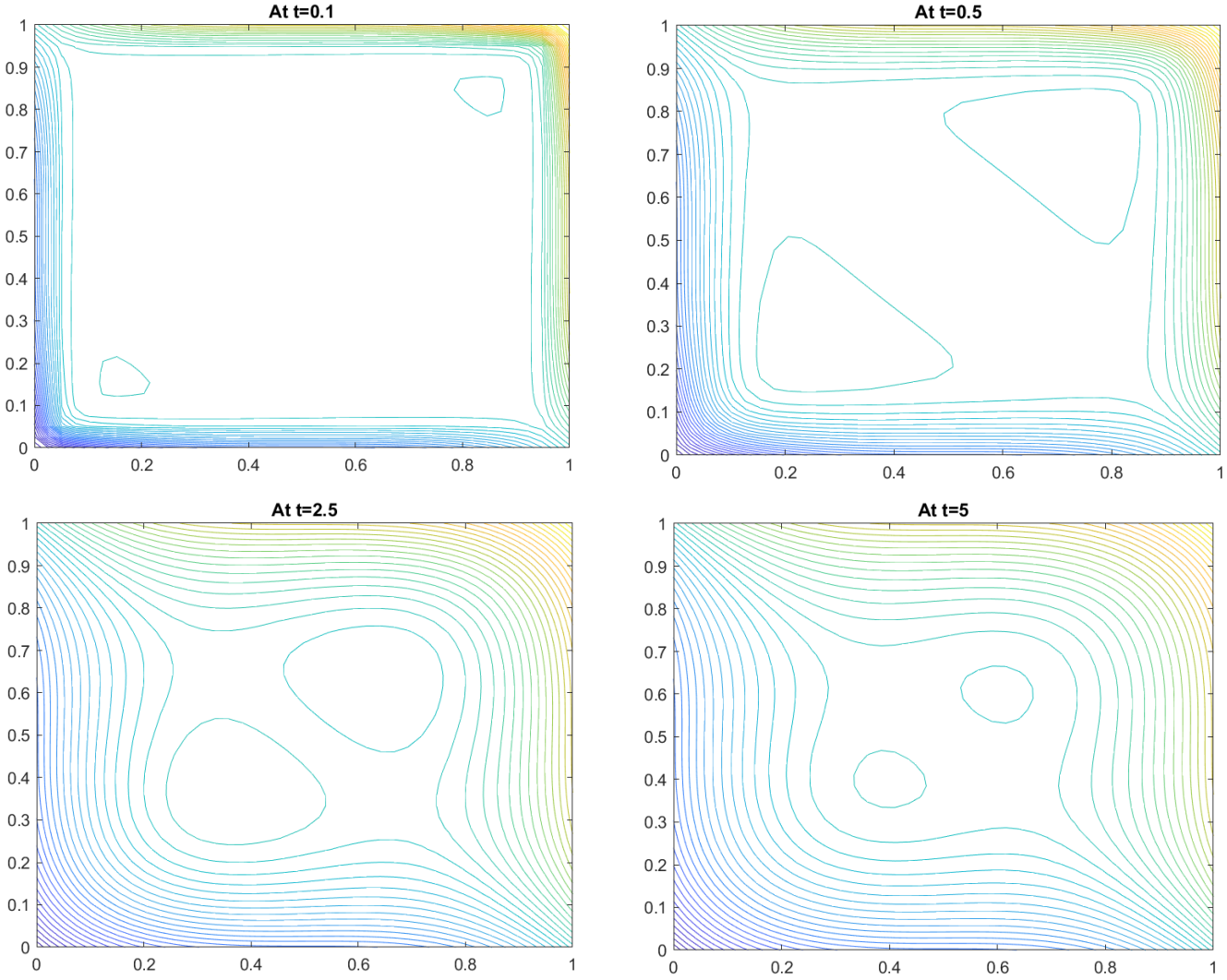
3.4 Discussion

Crank Nicolson method is intermediate between an explicit method and implicit method. Because Crank Nicolson method uses average of the previous and current time step to evaluate the integral. The previous value is used in computing the integral could introduce some explicit nature in the method, which would put a limit on how large a time step could be used. From Table 5 it can be seen that for the time step of 0.1 s Crank Nicolson method diverges. However, Crank Nicolson method allows large enough time step sufficient for many of practical problems.

4. Evolution ϕ with time

4.1 Plot of ϕ with time

For the unsteady-state equation, the equation was solved from t=0 s to t = 10 s by all the three methods – the explicit, implicit and Crank Nicolson method. The plot of ϕ at different time step computed using implicit method is shown below :



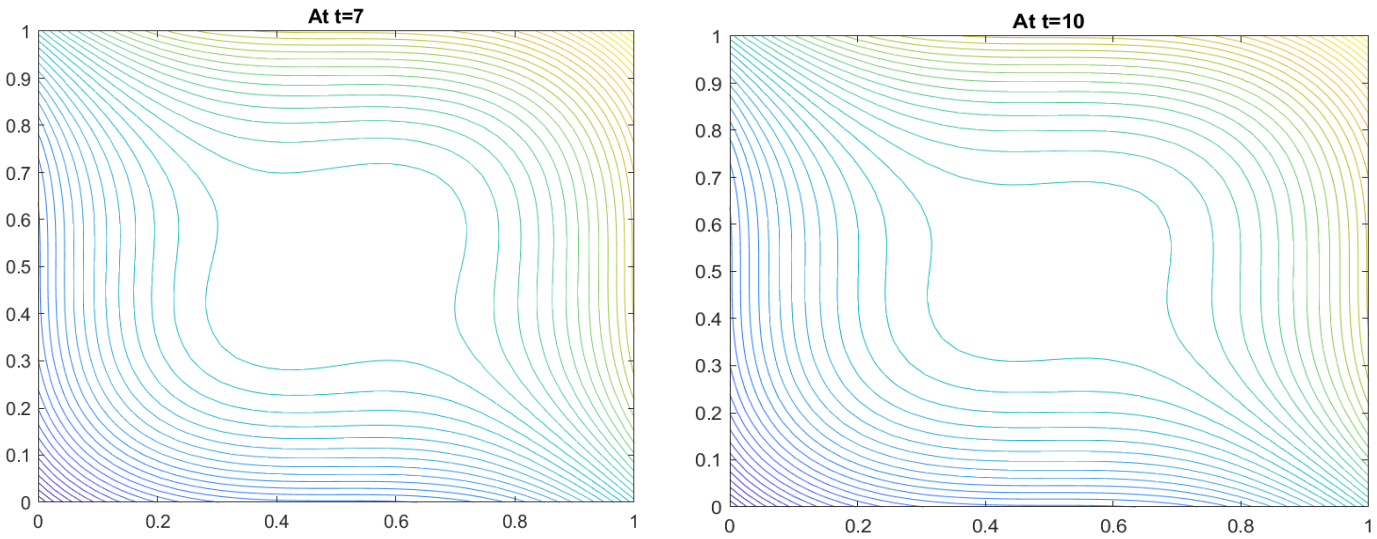


Figure 15 Plot of evolution of ϕ for the unsteady 2D-convection diffusion equation from t=0 s to t=10 s solved using implicit method

4.2 Discussion

The given unsteady 2D convection diffusion equation is essentially equating the bulk movement of ϕ with the diffusion and the source terms. From the first plot it can be seen that the value of ϕ at (1,1) is highest and at (0,0) it is lowest. The points (1,0) and (0,1) are symmetric and they have the same value of ϕ. One can imagine a line passing through (1,0) and (0,1) which divides the contour plot of ϕ in two components. The value of ϕ is greater in the upper half consisting the sides joining (0,1) with (1,1) and joining (1,0) with (1,1) whereas the value of ϕ is smaller in the lower half formed by the sides joining (0,0) with (0,1) and (0,0) with (1,0). Therefore, there exists a concentration gradient between the lower half and upper half. Because of this concentration gradient there is a diffusion of ϕ from the upper triangle to the lower triangles. This diffusion can be evident from the separation of contour lines with increasing time.

The source term also divides the 2D region into the similar triangular halves. The source term in the upper triangular half is positive whereas in the lower triangular half it is negative. Therefore even when the diffusion has become steady, the value of ϕ in node points of upper triangle is greater than the value of ϕ at node points of lower triangle. The contour seen at time 0.1, 0.5, 2.5 and 5 may due the local concentration gradients produced as a result of combined effects of source and diffusive terms.

The value of ϕ at the centre of plate i.e. the point $(\frac{1}{2},\frac{1}{2})$ asymptotically approaches zero. Moreover, there are no contour plots in the region near the centre point. This is because the concentration gradient keeps on decreasing as one moves towards the centre point.