Computational study to solve the Pressure-Poisson equation using Gauss-Seidel & ADI iterative methods by obtaining the Pressure field given the velocity field for Taylor-Green vortex with Dirichlet & Neumann boundary conditions

Name: Chinmay Rajesh Chavan

UIN: 633002377

Course: MEEN 689 – Computational Fluid Dynamics

Mid-term Project

Date: 11-09-2022

Introduction

To find the pressure field, generally we use the Pressure-Poisson equation for a given velocity field. The Pressure-Poisson equation, obtained by taking the divergence of the Navier-Stokes equations is given by,

$$\nabla^2 p = -\nabla.(\vec{u}.\nabla\vec{u}) \quad \dots\dots\dots\dots\dots Eq.(1)$$

Equation (1) can also be written as,

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -f \quad \dots\dots\dots\dots\dots Eq.(2)$$

where f is given by,

$$f = \frac{\partial}{\partial x}\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial y}\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) \dots\dots\dots\dots\dots Eq.(3)$$

Equation (2) is an **elliptic** PDE as here we have A=1, B=0 and C=1, which gives B^2-4AC= -4 < 0.

For elliptic type of PDEs we only need the boundary conditions to compute the solution.

In this study we are considering a known solution, which is the Taylor-Green vortex, for implementation and testing the developed code. The velocity field for this flow is given by,

$$u(x,y,t) = -e^{-\frac{2}{Re}t}\cos(x)\sin(y) \text{ and } v(x,y,t) = e^{-\frac{2}{Re}t}\sin(x)\cos(y) \dots\dots\dots\dots\dots Eq.(4)$$

We obtain analytical solution for pressure field by substituting the velocity field in the Navier-Stokes equation. The analytical solution for pressure is given by,

$$p(x,y,t) = -\frac{1}{4}e^{-\frac{4}{Re}t}[\cos(2x) + \cos(2y)] \quad \dots\dots\dots\dots Eq.(5)$$

To find analytical value of f, we need to take partial derivatives of the velocity field components given in Eq. (4) in both x and y directions, and then substitute them in Eq. (3). We get the derivatives of u and v as follows:

$$\frac{\partial u}{\partial x} = e^{-\frac{2}{Re}t}\sin(x)\sin(y) \text{ and } \frac{\partial u}{\partial y} = -e^{-\frac{2}{Re}t}\cos(x)\cos(y) \quad \dots\dots\dots\dots Eq.(6)$$

$$\frac{\partial v}{\partial x} = e^{-\frac{2}{Re}t}\cos(x)\cos(y) \text{ and } \frac{\partial v}{\partial y} = -e^{-\frac{2}{Re}t}\sin(x)\sin(y) \quad \dots\dots\dots\dots Eq.(7)$$

Substituting Eq. (6) and Eq. (7) into Eq. (3), we get **analytical value of f** as,

$$f = -e^{-\frac{4}{Re}t}[\cos^2(x) - \sin^2(x) + \cos^2(y) - \sin^2(y)] \quad \dots\dots\dots\dots Eq.(8)$$

which can be simplified as,

$$f = -e^{-\frac{4}{Re}t}[\cos(2x) + \cos(2y)] \quad \dots\dots\dots\dots Eq.(9)$$

We have considered the domain from $-\pi < x < \pi$ and $-{}^{\pi}/_{2} < y < {}^{\pi}/_{2}$ for this study as depicted in Fig. (1).

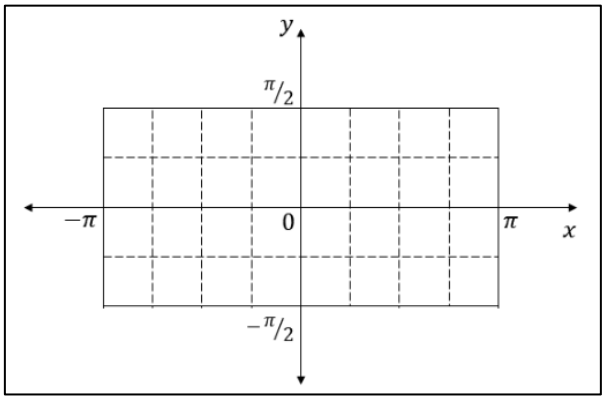

Fig.1. *Domain under consideration for this problem.*

In this study, several cases are considered for both Gauss-Seidel and ADI methods as shown in Table 1 and Table 2 for Dirichlet and Neumann boundary conditions respectively.

Case no.	No. of points in x-direction (NX)	Δx	No. of points in y-direction (NY)	Δy	Gauss-Seidel method		ADI method	
					No. of Iteration for convergence	Error value	No. of Iteration for convergence	Error value
1	101	0.062832	101	0.031416	3120	2.42E-05	800	4.03E-06
2	301	0.020944	151	0.020944	6328	4.45E-05	2896	1.04E-05
3	151	0.041888	301	0.010472	7377	1.01E-04	1689	5.03E-06
4	501	0.012566	251	0.012566	7938	7.34E-05	5376	1.82E-05
5	501	0.012566	501	0.006283	12819	7.49E-05	6515	2.17E-05
6	1001	0.006283	501	0.006283	16894	6.15E-05	7915	3.69E-05
7	1001	0.006283	1001	0.003142	27856	6.03E-05	9898	3.23E-05

Table 1. *Cases considered for evaluating with Dirichlet boundary conditions.*

Case no.	No. of points in x-direction (NX)	Δx	No. of points in y-direction (NY)	Δy	No. of Iteration for convergence	
					Gauss-Seidel method	ADI method
1	101	0.062832	101	0.031416	4638	1041
2	301	0.020944	151	0.020944	12082	4271
3	501	0.012566	251	0.012566	23549	9298
4	1001	0.006283	501	0.006283	42363	23549

Table 2. *Cases considered for evaluating with Neumann boundary conditions.*

Method of Solution

We discretize the Pressure-Poisson equation using the central difference scheme. The stencil selected has three points each in both x and y direction, giving an overall five point stencil in 2D as shown in Fig. (2). The resulting finite difference equation is expected to have accuracy of 2nd order in both x and y spatial directions. We are computing our solution for $t = 0$. After applying central difference approximation to Eq. (2), we get the discretized equation as:

$$\frac{p_{(i-1,j)} - 2p_{(i,j)} + p_{(i+1,j)}}{\Delta x^2} + \frac{p_{(i,j-1)} - 2p_{(i,j)} + p_{(i,j+1)}}{\Delta y^2} = -f \quad \dots\dots\dots\dots Eq.(10)$$

Multiplying Eq. (10) by Δx^2 and substituting $\beta = {}^{\Delta x}/_{\Delta y}$ we get FDE as,

$$p_{(i-1,j)} + p_{(i+1,j)} + \beta^2\left(p_{(i,j-1)} + p_{(i,j+1)}\right) - 2(1+\beta^2)p_{(i,j)} = -f\Delta x^2 \quad \dots\dots\dots\dots Eq.(11)$$

where f is given by Eq. (8) or Eq. (9). At $t = 0$, f is given as,

$$f = -[\cos^2(x) - \sin^2(x) + \cos^2(y) - \sin^2(y)] = -[\cos(2x) + \cos(2y)] \quad \dots\dots\dots Eq.(12)$$

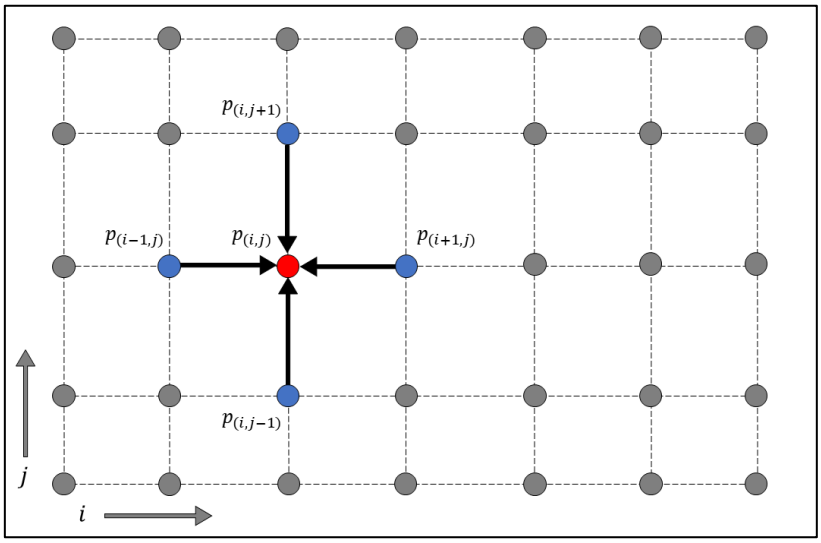

Fig.2. *Sketch of 5-point stencil used for this study.*

To compute the solution for FDE we can use various iterative methods. Since, these are iterative methods we need to define a criterion for a converged solution. We will assume solution to be converged if,

$$\sum_{j=2}^{NY-1}\sum_{i=2}^{NX-1} ABS\left(p_{(i,j)}^{k+1} - p_{(i,j)}^{k}\right) / \sum_{j=2}^{NY-1}\sum_{i=2}^{NX-1} ABS\left(p_{(i,j)}^{k}\right) < 10^{-5} \dots\dots Eq.(13)$$

where NX & NY are number of points in x & y direction respectively. For this study, we will be computing solution for pressure field using Gauss-Seidel and ADI iterative methods.

Gauss-Seidel method:

Here, we use the latest available value for $(i-1,j),(i+1,j),(i,j-1)$ & $(i,j+1)$ to get next iterative value for (i,j). We use following equation to implement Gauss-seidel method for this problem,

$$p_{(i,j)}^{(n+1)} = \frac{1}{2(1+\beta^2)}\left[p_{(i-1,j)}^{(n+1)} + p_{(i+1,j)}^{(n)} + \beta^2\left(p_{(i,j-1)}^{(n+1)} + p_{(i,j+1)}^{(n)}\right) + f\Delta x^2\right] \dots\dots\dots Eq.(14)$$

ADI method:

In this method, we will use 2-step approach. First, we will compute solution using x-sweep for half step and then compute solution using y-sweep for the remaining half step to complete one iteration as illustrated in Fig. (3). To implement this method we use following equations,

For X-sweep,

$$p_{(i-1,j)}^{(n+1/2)} - 2(1+\beta^2)p_{(i,j)}^{(n+1/2)} + p_{(i+1,j)}^{(n+1/2)} = -\beta^2\left(p_{(i,j+1)}^{(n)} + p_{(i,j-1)}^{(n+1/2)}\right) - f\Delta x^2 \dots\dots Eq.(15)$$

For Y-sweep,

$$\beta^2 p_{(i,j-1)}^{(n+1)} - 2(1+\beta^2)p_{(i,j)}^{(n+1)} + \beta^2 p_{(i,j+1)}^{(n+1)} = -\left(p_{(i+1,j)}^{(n+1/2)} + p_{(i-1,j)}^{(n+1)}\right) - f\Delta x^2 \dots\dots Eq.(16)$$

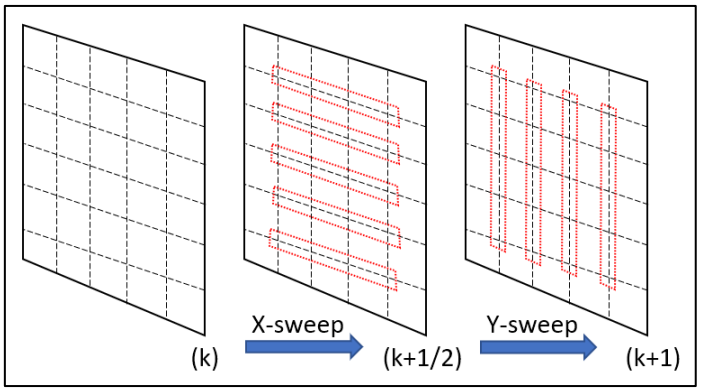

Fig.3. *Illustration for one iteration from k^{th} to $(k+1)^{th}$ step using ADI method.*

To compute the solution using both the above methods we will be applying Dirichlet and Neumann boundary conditions for this study.

Dirichlet boundary conditions:

For Dirichlet boundary condition, we need the value for boundary points which needs to be substituted for each iteration. Here, as we know the analytical solution for pressure field from Eq. (5), we will use this on the boundaries defined for our domain in Fig. (1).

At $t = 0$, analytical solution for pressure field is given by,

$$p = -\frac{1}{4}[\cos(2x) + \cos(2y)] \dots\dots\dots Eq.(17)$$

Neumann boundary conditions:

In Neumann boundary conditions, we need to know the value for normal (i.e., $\partial p/\partial n$) to the boundary. Here, as the domain is rectangular, we will need boundary values for two normal perpendicular to x and y directions. So, by taking partial derivatives of Eq. (5) w.r.t x and y separately, we get,

$$\frac{\partial p}{\partial x} = \frac{1}{2} e^{-\frac{4}{Re}t} \sin(2x) \quad \& \quad \frac{\partial p}{\partial y} = \frac{1}{2} e^{-\frac{4}{Re}t} \sin(2y) \dots\dots\dots\dots Eq.(18)$$

We will compute values for $\partial p/\partial x$ and $\partial p/\partial y$ at the domain boundaries as shown in Fig.1.
At $x = -\pi$ and $x = \pi$, we get $\partial p/\partial x = 0$ $\dots\dots\dots\dots\dots Eq.(19)$
Similarly, at $y = -\pi/2$ and $y = \pi/2$, we get $\partial p/\partial y = 0$ $\dots\dots\dots\dots\dots Eq.(20)$

To implement Neumann boundary conditions in the computer code, we need to update values for points on boundaries. We have considered ghost nodes outside the boundary for this implementation at $i = 0$, $i = (NX + 1)$, $j = 0$ and $j = (NY + 1)$.

On **left edge** boundary (i.e., $x = -\pi$), we have $i = 1$ and $j = 1{:}\,NY$. From Eq. (19) we know that $\partial p/\partial x = 0$. Using central difference scheme for points on this boundary we get,

$$\frac{p_{(2,j)} - p_{(0,j)}}{2\Delta x} = 0 \quad \xrightarrow{gives} \quad p_{(0,j)} = p_{(2,j)} \dots\dots\dots\dots Eq.(21)$$

The FDE for the points on left boundary is given by,

$$p_{(1,j)} = \frac{1}{2(1+\beta^2)}\left[p_{(0,j)} + p_{(2,j)} + \beta^2\left(p_{(1,j-1)} + p_{(1,j+1)}\right) + f\Delta x^2\right]$$

Substituting value for ghost node $p_{(0,j)}$ from Eq. (21) we get equation for left boundary points as,

$$p_{(1,j)} = \frac{1}{2(1+\beta^2)}\left[2p_{(2,j)} + \beta^2\left(p_{(1,j-1)} + p_{(1,j+1)}\right) + f\Delta x^2\right] \dots\dots\dots\dots Eq.(22)$$

On **right edge** boundary (i.e., $x = \pi$), we have $i = NX$ and $j = 1{:}\,NY$. From Eq. (19) we know that $\partial p/\partial x = 0$. Using central difference scheme for points on this boundary we get,

$$\frac{p_{(NX+1,j)} - p_{(NX-1,j)}}{2\Delta x} = 0 \quad \xrightarrow{gives} \quad p_{(NX+1,j)} = p_{(NX-1,j)} \dots\dots\dots\dots Eq.(23)$$

The FDE for the points on right boundary is given by,

$$p_{(NX,j)} = \frac{1}{2(1+\beta^2)}\left[p_{(NX-1,j)} + p_{(NX+1,j)} + \beta^2\left(p_{(NX,j-1)} + p_{(NX,j+1)}\right) + f\Delta x^2\right]$$

Substituting value for ghost node $p_{(NX+1,j)}$ from Eq. (23) we get equation for right boundary points as,

$$p_{(NX,j)} = \frac{1}{2(1+\beta^2)}\left[2p_{(NX-1,j)} + \beta^2\left(p_{(NX,j-1)} + p_{(NX,j+1)}\right) + f\Delta x^2\right] \dots\dots\dots\dots Eq.(24)$$

On **bottom edge** boundary (i.e., $y = -\pi/2$), we have $j = 1$ and $i = 1{:}\,NX$. From Eq. (20) we know that $\partial p/\partial y = 0$. Using central difference scheme for points on this boundary we get,

$$\frac{p_{(i,2)} - p_{(i,0)}}{2\Delta y} = 0 \quad \xrightarrow{gives} \quad p_{(i,0)} = p_{(i,2)} \dots\dots\dots\dots Eq.(25)$$

The FDE for the points on bottom boundary is given by,

$$p_{(i,1)} = \frac{1}{2(1+\beta^2)}\left[p_{(i-1,1)} + p_{(i+1,1)} + \beta^2\left(p_{(i,0)} + p_{(i,2)}\right) + f\Delta x^2\right]$$

Substituting value for ghost node $p_{(i,0)}$ from Eq. (25) we get equation for bottom boundary points as,

$$p_{(i,1)} = \frac{1}{2(1+\beta^2)}\left[p_{(i-1,1)} + p_{(i+1,1)} + \beta^2\left(2p_{(i,2)}\right) + f\Delta x^2\right] \dots\dots\dots\dots Eq.(26)$$

On **top edge** boundary (i.e., $y = \pi/2$), we have $j = NY$ and $i = 1{:}\,NX$. From Eq. (20) we know that $\partial p/\partial y = 0$. Using central difference scheme for points on this boundary we get,

$$\frac{p_{(i,NY+1)} - p_{(i,NY-1)}}{2\Delta y} = 0 \quad \xrightarrow{gives} \quad p_{(i,NY+1)} = p_{(i,NY-1)} \dots\dots\dots\dots Eq.(27)$$

The FDE for the points on top boundary is given by,

$$p_{(i,NY)} = \frac{1}{2(1+\beta^2)}\left[p_{(i-1,NY)} + p_{(i+1,NY)} + \beta^2\left(p_{(i,NY-1)} + p_{(i,NY+1)}\right) + f\Delta x^2\right]$$

Substituting value for ghost node $p_{(i,NY+1)}$ from Eq. (27) we get equation for top boundary points as,

$$p_{(i,NY)} = \frac{1}{2(1+\beta^2)}\left[p_{(i-1,NY)} + p_{(i+1,NY)} + \beta^2\left(2p_{(i,NY-1)}\right) + f\Delta x^2\right] \dots\dots\dots\dots Eq.(28)$$

Computing error:

After applying boundary conditions for both Gauss-Seidel and ADI methods we can compute error for computed solution w.r.t analytical solution using the L2 norm as follows:

$$error = \frac{1}{NY * NX}\sqrt{\sum_{j=2}^{NY-1}\sum_{i=2}^{NX-1}\left(p_{(i,j)}^{computed} - p_{(i,j)}^{analytical}\right)^2} \dots\dots\dots\dots Eq.(29)$$

Discussion of Results

A.) Solution using Dirichlet Boundary Condition:

i. Pressure plots:

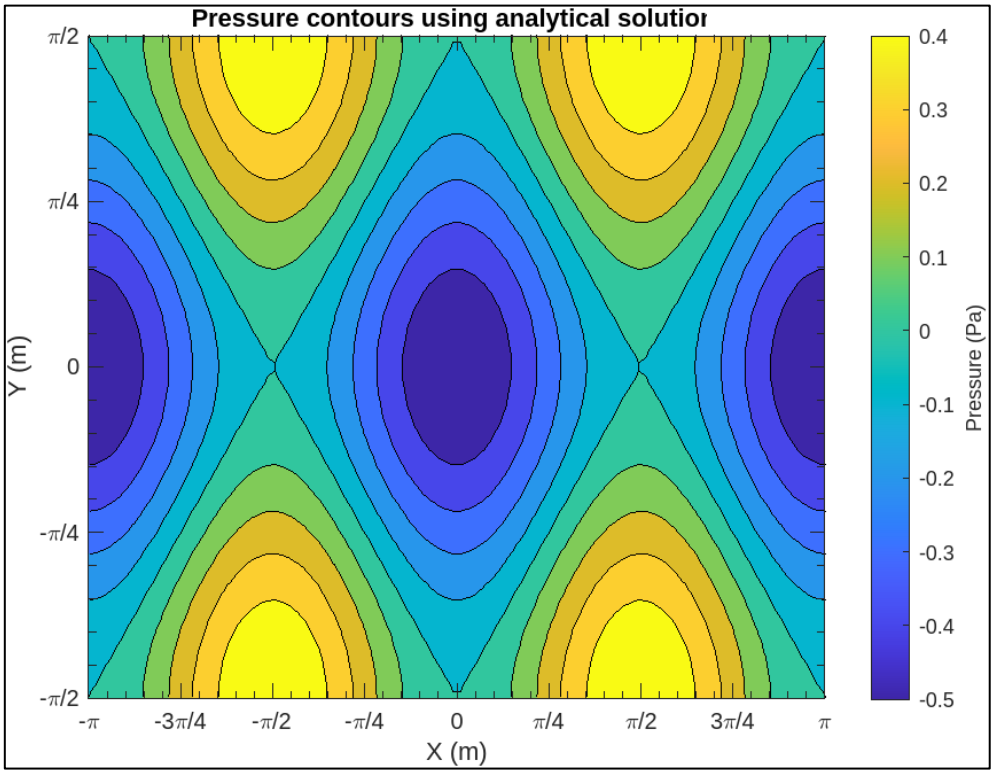

Fig.4. *Pressure contours using the analytical solution of Pressure-Poisson equation for Taylor-Green vortex at t=0.*

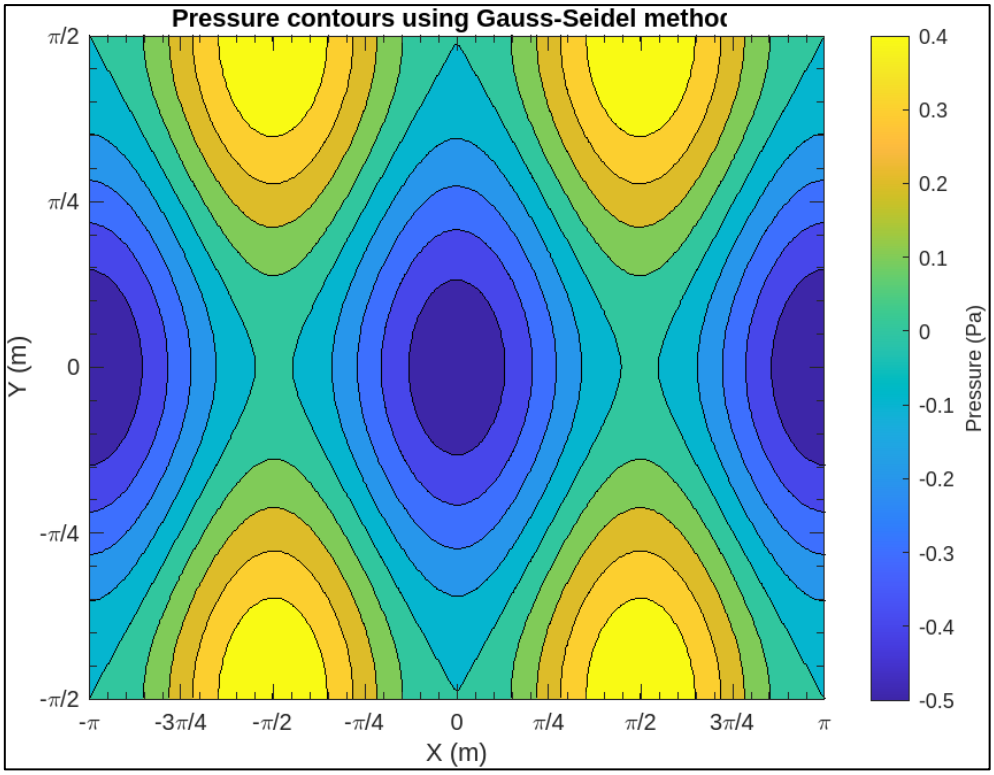

Fig.5. *Solution of the Pressure-Poisson equation using the Gauss-Seidel iterative method with Dirichlet boundary conditions for case 2 (i.e., Δx= 0.020944, Δy= 0.020944) at t=0.*

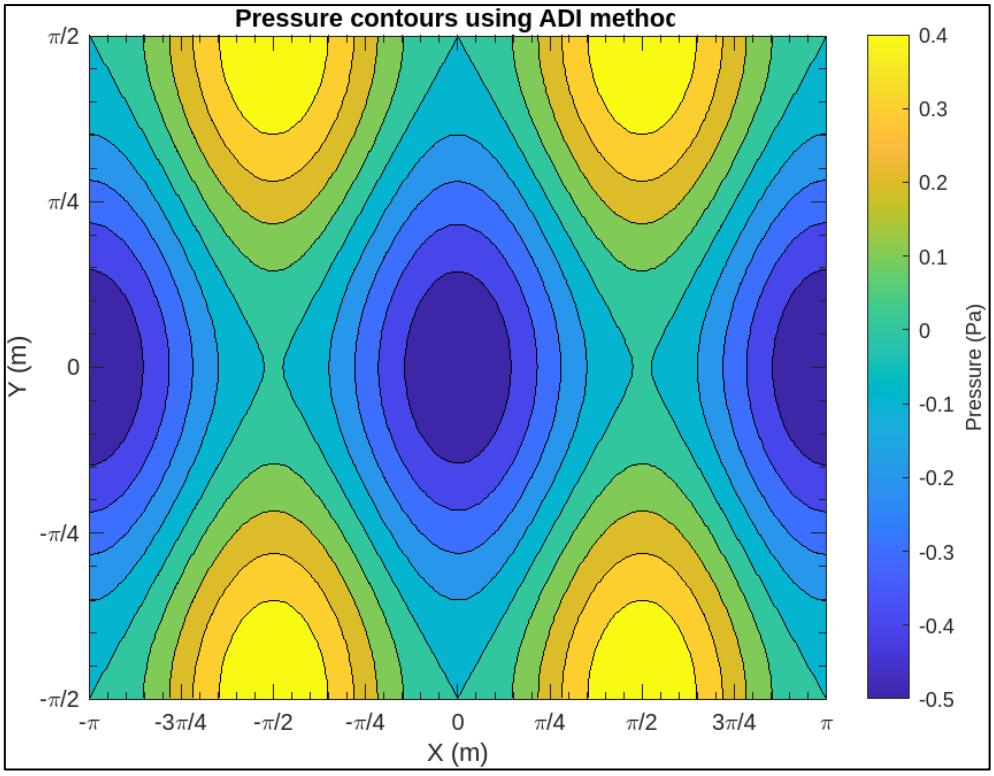

Fig.6. *Solution of the Pressure-Poisson equation using the ADI iterative method with Dirichlet boundary conditions for case 2 (i.e., Δx= 0.020944, Δy= 0.020944) at t=0.*

From the analytical solution plot in Fig. (4), we can see that the contours are well defined, has some contours as lines and most of them as elliptical curves and have sharp edges at certain points, specifically visible at $(\pm\pi/2, 0)$. Figures 5 & 6 show the solution obtained using Gauss-Seidel and ADI method respectively for case 2 using Dirichlet boundary conditions, and it does have well defined contours, but the edges do not have any sharp corners as seen in analytical solution at $(\pm\pi/2, 0)$. All the pressure plots are displaying pressure values in the same range of -0.5 Pa to 0.4 Pa to enable better comparison. We can observe that all the above plots have similar contours which shows that errors in the computed solutions are in acceptable limits and our convergence criterion is justified. When we look at the region in the vicinity of $(\pm\pi/2, 0)$, it is evident that in Fig. (5) for Gauss-Seidel method we have more error in the pressure contours than in Fig. (6) for ADI method when both are compared with analytical solution in Fig. (4).

ii. Order of accuracy:

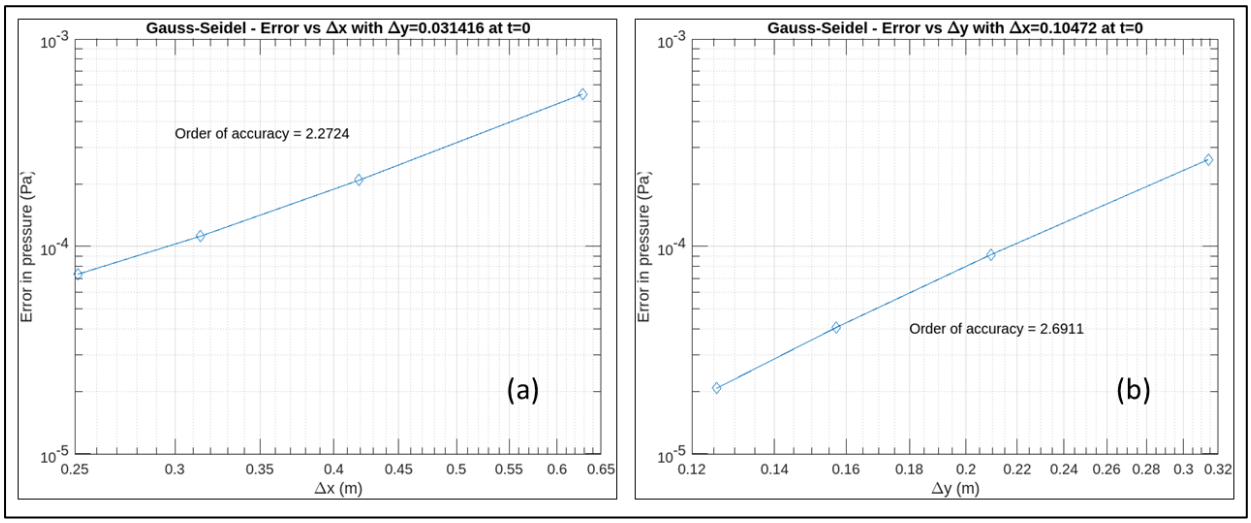

Fig.7. *Error plots in log-log scale (a) w.r.t. Δx with Δy=0.031416 at t=0 and (b) w.r.t. Δy with Δx=0.10472 at t=0 using Gauss-Seidel iterative method to obtain spatial order of accuracy.*

To get the spatial order of accuracy for Gauss-Seidel method, we have plotted error w.r.t. Δx & Δy as shown in Fig. (7a) & (7b) respectively. As these plots are in log-log scale, the slope will give us the order of accuracy and as we are plotting error against spatial step, the slope will give us spatial order of accuracy. From Fig. (7) we can say that using Gauss-Seidel method we are getting **second order accurate** results in **both x & y spatial directions**.

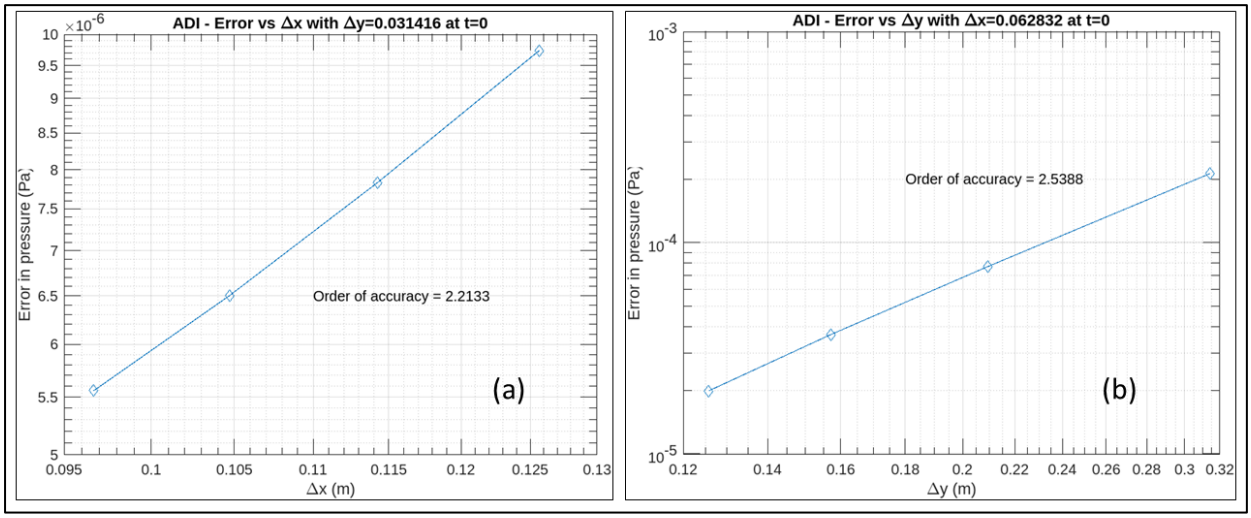

Fig.8. *Error plots in log-log scale (a) w.r.t. Δx with Δy=0.031416 at t=0 and (b) w.r.t. Δy with Δx=0.062832 at t=0 using ADI iterative method to obtain spatial order of accuracy.*

To get the spatial order of accuracy for ADI method, we have plotted error w.r.t. Δx & Δy as shown in Fig. (8a) & (8b) respectively. Similarly, as stated above the slope of these plots will give us spatial order of accuracy. From Fig. (8) we can say that using ADI method we are getting **second order accurate** results in **both x & y spatial directions**.

From Figs. (7) & (8), we have a common observation that the spatial order of accuracy is close to 2 w.r.t. Δx and farther from 2 w.r.t Δy. Also, we observe that in both methods the error value reduces as Δx & Δy value reduces and becomes closer to zero.

iii. Comparison between Gauss-Seidel and ADI methods:

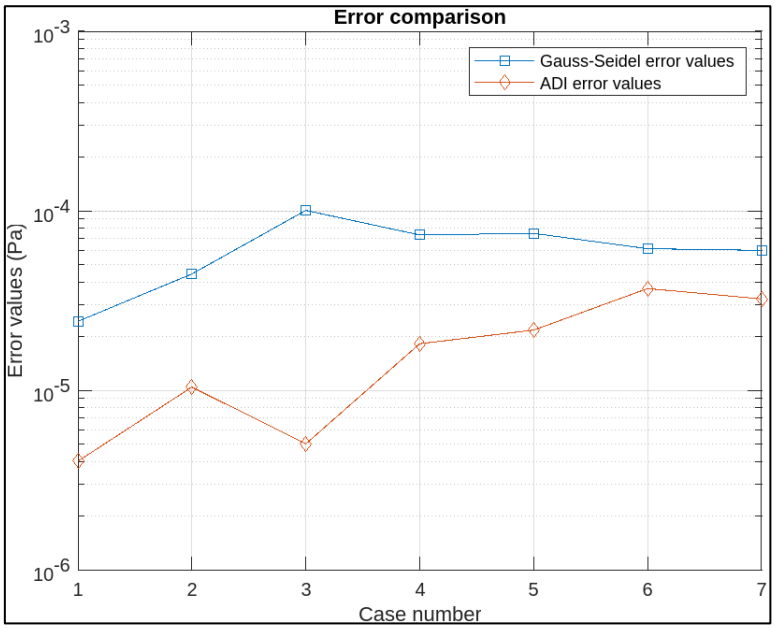

Fig.9. *Error comparison for each case with Dirichlet boundary condition using Gauss-Seidel and ADI methods.*

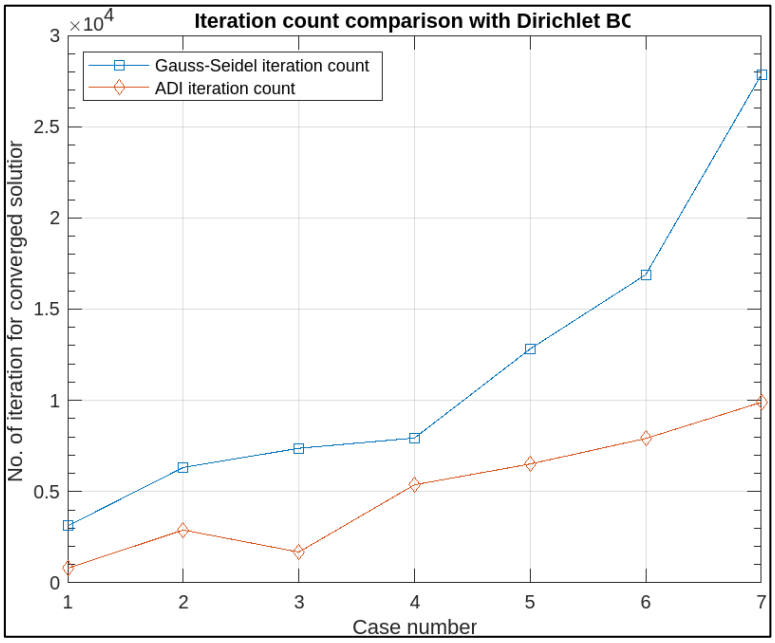

Fig.10. *Comparison between number of iterations required to achieve convergence criterion for each case with Dirichlet boundary condition using Gauss-Seidel and ADI methods.*

From Fig. (9), which shows the error comparison for each case with Dirichlet boundary condition using Gauss-Seidel & ADI methods, we can say that for all cases the error is lesser for ADI as compared to Gauss-Seidel method. We are observing minimum error reduction of 40% for case 6 using ADI method. From Fig. (10), which shows the number of iteration comparison for each case with Dirichlet boundary condition using Gauss-Seidel & ADI methods, we can say that for all cases ADI method required less iterations for converged solution as compared to Gauss-Seidel method. We are observing reduction in iteration count of at least 32% for case 4 using ADI method. Although ADI has two sweeps for one iteration, which would double the count of total iterative loops, still we get lower iteration

loops for ADI than for Gauss-Seidel scheme for most of the cases except case 4 & case 5. Thus, in general ADI method seems to be better than Gauss-Seidel method in terms of both computational cost and accuracy.

B.) Solution using Neumann Boundary Conditions:

i. Pressure Plots:

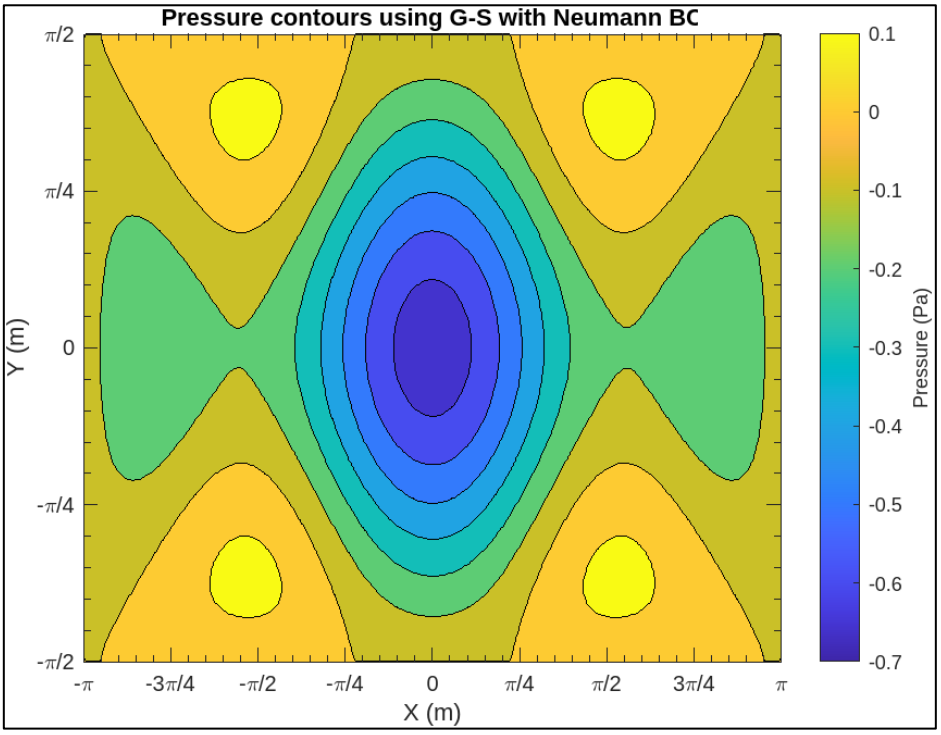

Fig.11. *Solution of the Pressure-Poisson equation using the Gauss-Seidel iterative method with Neumann boundary conditions for case 2 (i.e., Δx= 0.020944, Δy= 0.020944) at t=0.*

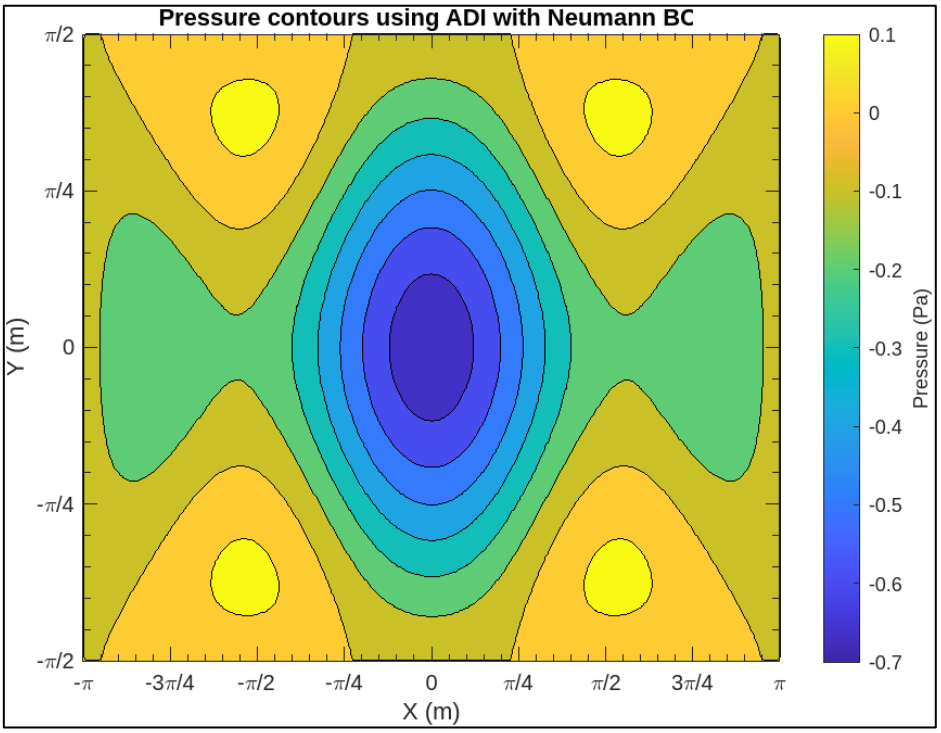

Fig.12. *Solution of the Pressure-Poisson equation using the ADI iterative method with Neumann boundary conditions for case 2 (i.e., Δx= 0.020944, Δy= 0.020944) at t=0.*

Figures 11 & 12 show the solution obtained using Gauss-Seidel and ADI method respectively for case 2 using Neumann boundary conditions, and we can see that the plots have well defined contours. Here, both the pressure plots are displaying pressure values in the same range of -0.7 Pa to 0.1 Pa to enable better comparison, and we can observe that they have similar contours. On the contrary, we observe larger deviation in the plots with Neumann boundary condition when compared to the analytical solution in Fig. 4. Although the overall shape is identical, especially in the center region, the higher error away from center is due to 3-point central difference stencil used for obtaining the discretized equation for the boundary points. This error could be mitigated if we use a stencil with more points while implementing the Neumann boundary condition.

ii. Comparison between Gauss-Seidel and ADI methods:

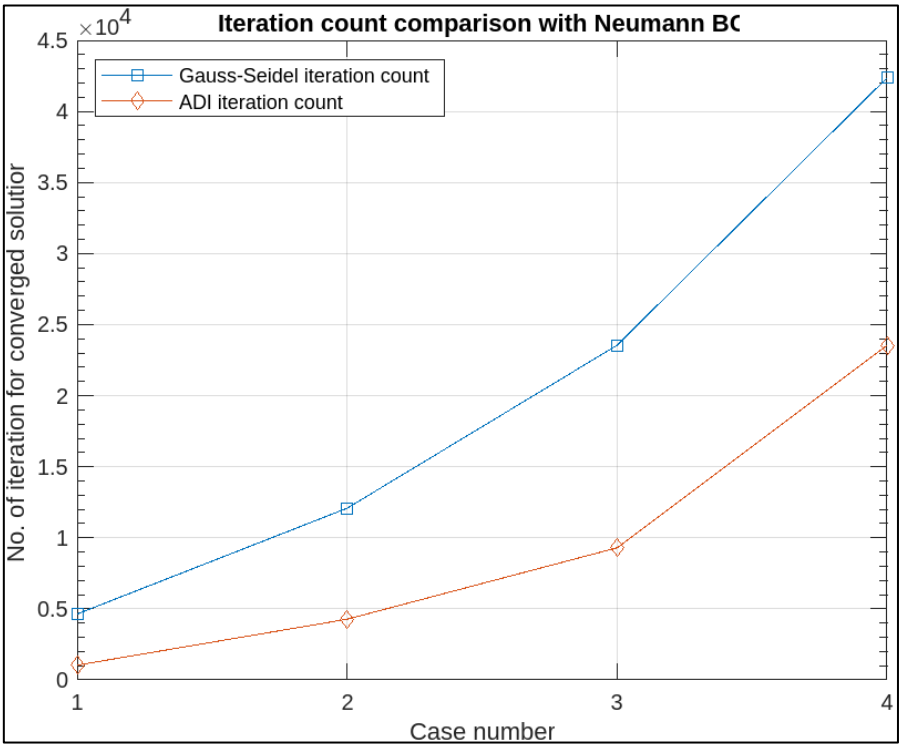

Fig.13. *Comparison between number of iterations required to achieve convergence criterion for each case with Neumann boundary condition using Gauss-Seidel and ADI methods.*

From Fig. (13), which shows the number of iteration comparison for each case with Neumann boundary condition using Gauss-Seidel & ADI methods, we can say that for all cases ADI method required less iterations for converged solution as compared to Gauss-Seidel method. We are observing reduction in iteration count of at least 44% for case 4 using ADI method. For rest of the cases, ADI has more than 60% reduction in the iteration count. So, even if it requires two iterative loops for one ADI iteration, it is still better than Gauss-Seidel method in terms of computational cost. This observation is similar to our previous inference with Dirichlet boundary condition.

Summary & Conclusions

- From this study we got good alignment between analytical solution pressure plot and computed solution plot using both Gauss-Seidel & ADI methods with Dirichlet boundary conditions.
- Plots with Neumann boundary condition deviate from the analytical solution, although the overall shape is identical. We can say that, to improve accuracy, a stencil with higher number of points needs to be considered for implementing Neumann boundary condition.
- We achieved good correlation between order of accuracy obtained using both Gauss-Seidel & ADI iterative methods and the theoretical values which indicate second order accuracy in spatial domain.
- We observed that both methods converge towards zero error when Δx & Δy are closer to zero.
- ADI method resulted in smaller error value as compared to Gauss-Seidel method, which was expected as we are doing 2 sweeps in one iteration of ADI.
- ADI method also resulted in smaller iteration count as compared to Gauss-Seidel method, which can be considered proportional to the computing cost.
- To conclude, it can be said that ADI method is better than Gauss-Seidel method in terms of both computational cost and accuracy.

APPENDIX

MATLAB codes used for CFD mid-term project:-

1.) Code to plot **Analytical** pressure contours:

```
clc;
figure;
x=linspace(-pi,pi);
y=linspace(-pi/2,pi/2);
[X,Y]=meshgrid(x,y);
P=-(cos(X.*2)+cos(Y.*2))/4;
contourf(X,Y,P);
clim([-0.5 0.4]);
colorbar;
title('Pressure contours using analytical solution ');
xlabel('X (m)');
ylabel('Y (m)');
axis([-pi pi -pi/2 pi/2]);
set(gca,'XTick',-pi:pi/4:pi);
set(gca,'XTickLabel',{'-\pi','-3\pi/4','-\pi/2','-
\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'});
set(gca,'YTick',-pi/2:pi/4:pi/2);
set(gca,'YTickLabel',{'-\pi/2','-\pi/4','0','\pi/4','\pi/2'});
a=colorbar;
a.Label.String = 'Pressure (Pa)';
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
```

2.) Code to plot pressure contours using **Gauss-Seidel method with Dirichlet boundary conditions** along with error value and iteration count for case study:

```
clc;
e=1;
x=-pi;
y=-pi/2;
NX=301;
NY=151;
dx=2*pi/(NX-1);
dy=pi/(NY-1);
b=dx/dy;
it=0;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
    u(1,j)=-(cos(2*x)+cos(2*y))/4;
    x=pi;
    u(NX,j)=-(cos(2*x)+cos(2*y))/4;
```

```
end
%% Gauss-Seidel iteration loop
while e>0.00001
    e=0;
    it=it+1;
    e1=0;
    e2=0;
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        for i=2:NX-1
            x=-pi+(i-1)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c=u(i,j);
            u(i,j)=(u(i-1,j)+u(i+1,j)+(b^2)*(u(i,j-
1)+u(i,j+1))+f*(dx^2))/(2*(1+(b^2)));
            e1=e1+abs(u(i,j)-c);
            e2=e2+abs(c);
        end
        for i=1:NX
            x=-pi+(i-1)*dx;
            y=-pi/2;
            u(i,1)=-(cos(2*x)+cos(2*y))/4;
            y=pi/2;
            u(i,NY)=-(cos(2*x)+cos(2*y))/4;
        end
        for j=1:NY
            y=-(pi/2)+(j-1)*dy;
            x=-pi;
            u(1,j)=-(cos(2*x)+cos(2*y))/4;
            x=pi;
            u(NX,j)=-(cos(2*x)+cos(2*y))/4;
        end
    end
    e=e1/e2;
end
%% Plot pressure contours
x=linspace(-pi,pi,NX);
y=linspace(-pi/2,pi/2,NY);
[X,Y]=meshgrid(x,y);
figure;
contourf(X,Y,u');
%clim([-0.5 0.4]);
colorbar;
title('Pressure contours using Gauss-Seidel method');
xlabel('X (m)');
ylabel('Y (m)');
axis([-pi pi -pi/2 pi/2]);
set(gca,'XTick',-pi:pi/4:pi);
set(gca,'XTickLabel',{'-\pi','-3\pi/4','-\pi/2','-
\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'});
set(gca,'YTick',-pi/2:pi/4:pi/2);
set(gca,'YTickLabel',{'-\pi/2','-\pi/4','0','\pi/4','\pi/2'});
a=colorbar;
a.Label.String = 'Pressure (Pa)';
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
%% Error value
e=0;
for j=2:NY-1
    y=-(pi/2)+(j-1)*dy;
```

```
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        P(i,j)=-(cos(x.*2)+cos(y.*2))/4;
        e=e+((u(i,j)-P(i,j))^2);
    end
end
Er=(sqrt(e))/(NX*NY) %display error
it %display iteration count
```

3.) Code to plot pressure contours using **ADI method with Dirichlet boundary conditions** along with error value and iteration count for case study:

```
clc;
e=1;
x=-pi;
y=-pi/2;
NX=301;
NY=151;
dx=2*pi/(NX-1);
dy=pi/(NY-1);
b=dx/dy;
it=0;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
    u(1,j)=-(cos(2*x)+cos(2*y))/4;
    x=pi;
    u(NX,j)=-(cos(2*x)+cos(2*y))/4;
end
%% ADI iteration loop
while e>0.00001
    e=0;
    it=it+1;
    e1=0;
    e2=0;
    %% X-sweep
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        for i=2:NX-1
            x=-pi+(i-1)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c(i,j)=u(i,j);
            A(i)=1;
            B(i)=-2*(1+(b^2));
            C(i)=1;
            D(i)=-(b^2)*(u(i,j+1)+u(i,j-1))-(f*(dx^2));
```

```
    end
    B(1,1)=-2*(1+(b^2));
    B(1,NX)=-2*(1+(b^2));
    D(1,2)=-(b^2)*(u(2,j+1)+u(2,j-1))-(f*(dx^2))-u(1,j);
    D(1,NX-1)=-(b^2)*(u(NX-1,j+1)+u(NX-1,j-1))-(f*(dx^2))-u(NX,j);
    for i=2:(NX-1)
        R=A(1,i)/B(1,i-1);
        B(1,i)=B(1,i)-R*C(1,i-1);
        D(1,i)=D(1,i)-R*D(1,i-1);
    end
    D(1,NX-1)=D(1,NX-1)/B(1,NX-1);
    for i=(NX-2):-1:2
        D(1,i)=(D(1,i)-C(1,i)*D(1,i+1))/B(1,i);
    end
    for i=2:NX-1
        u(i,j)=D(1,i);
    end
end
%% Y-sweep
for i=2:NX-1
    x=-pi+(i-1)*dx;
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
        A1(j)=b^2;
        B1(j)=-2*(1+(b^2));
        C1(j)=b^2;
        D1(j)=-u(i+1,j)-u(i-1,j)-(f*(dx^2));
    end
    B1(1,1)=-2*(1+(b^2));
    B1(1,NY)=-2*(1+(b^2));
    D1(1,2)=-u(i+1,2)-u(i-1,2)-(f*(dx^2))-(b^2)*u(i,1);
    D1(1,NY-1)=-u(i+1,NY-1)-u(i-1,NY-1)-(f*(dx^2))-(b^2)*u(i,NY);
    for j=2:(NY-1)
        R1=A1(1,j)/B1(1,j-1);
        B1(1,j)=B1(1,j)-R1*C1(1,j-1);
        D1(1,j)=D1(1,j)-R1*D1(1,j-1);
    end
    D1(1,NY-1)=D1(1,NY-1)/B1(1,NY-1);
    for j=(NY-2):-1:2
        D1(1,j)=(D1(1,j)-C1(1,j)*D1(1,j+1))/B1(1,j);
    end
    for j=2:NY-1
        u(i,j)=D1(1,j);
        e1=e1+abs(u(i,j)-c(i,j));
        e2=e2+abs(c(i,j));
    end
end
% Update BC
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
```

```
        u(1,j)=-(cos(2*x)+cos(2*y))/4;
        x=pi;
        u(NX,j)=-(cos(2*x)+cos(2*y))/4;
    end
    e=e1/e2;
end
%% Plot pressure contours
x=linspace(-pi,pi,NX);
y=linspace(-pi/2,pi/2,NY);
[X,Y]=meshgrid(x,y);
figure;
contourf(X,Y,u');
%clim([-0.5 0.4]);
colorbar;
title('Pressure contours using ADI method');
xlabel('X (m)');
ylabel('Y (m)');
axis([-pi pi -pi/2 pi/2]);
set(gca,'XTick',-pi:pi/4:pi);
set(gca,'XTickLabel',{'-\pi','-3\pi/4','-\pi/2','-
\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'});
set(gca,'YTick',-pi/2:pi/4:pi/2);
set(gca,'YTickLabel',{'-\pi/2','-\pi/4','0','\pi/4','\pi/2'});
a=colorbar;
a.Label.String = 'Pressure (Pa)';
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
%% Error value
e=0;
for j=2:NY-1
    y=-(pi/2)+(j-1)*dy;
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        P(i,j)=-(cos(x.*2)+cos(y.*2))/4;
        e=e+((u(i,j)-P(i,j))^2);
    end
end
Er=(sqrt(e))/(NX*NY) %display error
it %display iteration count
```

4.) Code to get **Error plot w.r.t Δx** using Gauss-Seidel method with Dirichlet BC along with spatial order of accuracy in x-direction:

```
clc;
e=1;
x=-pi;
y=-pi/2;
it=0;
for NX = [11 16 21 26]
it=it+1;
NY=101;
dx=2*pi/(NX-1);
dy=pi/(NY-1);
b=dx/dy;


%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
```

```
    end
end
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
    u(1,j)=-(cos(2*x)+cos(2*y))/4;
    x=pi;
    u(NX,j)=-(cos(2*x)+cos(2*y))/4;
end
%% Gauss-Seidel iteration loop
while e>0.00001
    e=0;
    e1=0;
    e2=0;
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        for i=2:NX-1
            x=-pi+(i-1)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c=u(i,j);
            u(i,j)=(u(i-1,j)+u(i+1,j)+(b^2)*(u(i,j-
1)+u(i,j+1))+f*(dx^2))/(2*(1+(b^2)));
            e1=e1+abs(u(i,j)-c);
            e2=e2+abs(c);
        end
        for i=1:NX
            x=-pi+(i-1)*dx;
            y=-pi/2;
            u(i,1)=-(cos(2*x)+cos(2*y))/4;
            y=pi/2;
            u(i,NY)=-(cos(2*x)+cos(2*y))/4;
        end
        for j=1:NY
            y=-(pi/2)+(j-1)*dy;
            x=-pi;
            u(1,j)=-(cos(2*x)+cos(2*y))/4;
            x=pi;
            u(NX,j)=-(cos(2*x)+cos(2*y))/4;
        end
    end
    e=e1/e2;
end
%% Error plot
e=0;
for j=2:NY-1
    y=-(pi/2)+(j-1)*dy;
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        P(i,j)=-(cos(x.*2)+cos(y.*2))/4;
        e=e+((u(i,j)-P(i,j))^2);
    end
end
```

```
Er(1,it)=(sqrt(e))/(NX*NY);
DX(1,it)=dx;
end
figure;
lg=loglog(DX,Er,"Marker","diamond");
acry=0;
for k=2:it
    acry = acry+((log(Er(1,k))-log(Er(1,1)))/(log(DX(1,k))-log(DX(1,1))));
end
acry = acry/(it-1);
txt = "Order of accuracy = " + acry;
text(0.3,0.00035,txt);
title("Gauss-Seidel - Error vs \Deltax with \Deltay=" + dy + " at t=0");
xlabel('\Deltax (m)');
ylabel('Error in pressure (Pa)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.03, 0.005]);
xlim([0.25 0.65]);
ylim([0.00001 0.001]);
grid on;
```

5.) Code to get **Error plot w.r.t Δy** using Gauss-Seidel method with Dirichlet BC along with spatial order of accuracy in y-direction:

```
clc;
e=1;
x=-pi;
y=-pi/2;
it=0;
for NY = [11 16 21 26]
it=it+1;
NX=61;
dx=2*pi/(NX-1);
dy=pi/(NY-1);
b=dx/dy;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
    u(1,j)=-(cos(2*x)+cos(2*y))/4;
    x=pi;
    u(NX,j)=-(cos(2*x)+cos(2*y))/4;
end
%% Gauss-Seidel iteration loop
while e>0.00001
    e=0;
    e1=0;
```

```
    e2=0;
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        for i=2:NX-1
            x=-pi+(i-1)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c=u(i,j);
            u(i,j)=(u(i-1,j)+u(i+1,j)+(b^2)*(u(i,j-
1)+u(i,j+1))+f*(dx^2))/(2*(1+(b^2)));
            e1=e1+abs(u(i,j)-c);
            e2=e2+abs(c);
        end
        for i=1:NX
            x=-pi+(i-1)*dx;
            y=-pi/2;
            u(i,1)=-(cos(2*x)+cos(2*y))/4;
            y=pi/2;
            u(i,NY)=-(cos(2*x)+cos(2*y))/4;
        end
        for j=1:NY
            y=-(pi/2)+(j-1)*dy;
            x=-pi;
            u(1,j)=-(cos(2*x)+cos(2*y))/4;
            x=pi;
            u(NX,j)=-(cos(2*x)+cos(2*y))/4;
        end
    end
    e=e1/e2;
end
%% Error plot
e=0;
for j=2:NY-1
    y=-(pi/2)+(j-1)*dy;
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        P(i,j)=-(cos(x.*2)+cos(y.*2))/4;
        e=e+((u(i,j)-P(i,j))^2);
    end
end
Er(1,it)=(sqrt(e))/(NX*NY);
DY(1,it)=dy;
end
figure;
lg=loglog(DY,Er,"Marker","diamond");
acry=0;
for k=2:it
    acry = acry+((log(Er(1,k))-log(Er(1,1)))/(log(DY(1,k))-log(DY(1,1))));
end
acry = acry/(it-1);
txt = "Order of accuracy = " + acry;
text(0.18,0.00004,txt);
title("Gauss-Seidel - Error vs \Deltay with \Deltax=" + dx + " at t=0");
xlabel('\Deltay (m)');
ylabel('Error in pressure (Pa)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.03, 0.005]);
xlim([0.12 0.32]);
ylim([0.00001 0.001]);
grid on;
```

6.) Code to get **Error plot w.r.t Δx** using ADI method with Dirichlet BC along with spatial order of accuracy in x-direction:

```
clc;
e=1;
x=-pi;
y=-pi/2;
it=0;
for NX = [51 56 61 66]
it=it+1;
NY=101;
dx=2*pi/(NX-1);
dy=pi/(NY-1);
b=dx/dy;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
    u(1,j)=-(cos(2*x)+cos(2*y))/4;
    x=pi;
    u(NX,j)=-(cos(2*x)+cos(2*y))/4;
end
%% ADI iteration loop
while e>0.00001
    e=0;
    e1=0;
    e2=0;
    %% X-sweep
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        for i=2:NX-1
            x=-pi+(i-1)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c(i,j)=u(i,j);
            A(i)=1;
            B(i)=-2*(1+(b^2));
            C(i)=1;
            D(i)=-(b^2)*(u(i,j+1)+u(i,j-1))-(f*(dx^2));
        end
        B(1,1)=-2*(1+(b^2));
        B(1,NX)=-2*(1+(b^2));
        D(1,2)=-(b^2)*(u(2,j+1)+u(2,j-1))-(f*(dx^2))-u(1,j);
        D(1,NX-1)=-(b^2)*(u(NX-1,j+1)+u(NX-1,j-1))-(f*(dx^2))-u(NX,j);
        for i=2:(NX-1)
            R=A(1,i)/B(1,i-1);
            B(1,i)=B(1,i)-R*C(1,i-1);
            D(1,i)=D(1,i)-R*D(1,i-1);
        end
```

```
        D(1,NX-1)=D(1,NX-1)/B(1,NX-1);
        for i=(NX-2):-1:2
            D(1,i)=(D(1,i)-C(1,i)*D(1,i+1))/B(1,i);
        end
        for i=2:NX-1
            u(i,j)=D(1,i);
        end
    end
    %% Y-sweep
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        for j=2:NY-1
            y=-(pi/2)+(j-1)*dy;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            A1(j)=b^2;
            B1(j)=-2*(1+(b^2));
            C1(j)=b^2;
            D1(j)=-u(i+1,j)-u(i-1,j)-(f*(dx^2));
        end
        B1(1,1)=-2*(1+(b^2));
        B1(1,NY)=-2*(1+(b^2));
        D1(1,2)=-u(i+1,2)-u(i-1,2)-(f*(dx^2))-(b^2)*u(i,1);
        D1(1,NY-1)=-u(i+1,NY-1)-u(i-1,NY-1)-(f*(dx^2))-(b^2)*u(i,NY);
        for j=2:(NY-1)
            R1=A1(1,j)/B1(1,j-1);
            B1(1,j)=B1(1,j)-R1*C1(1,j-1);
            D1(1,j)=D1(1,j)-R1*D1(1,j-1);
        end
        D1(1,NY-1)=D1(1,NY-1)/B1(1,NY-1);
        for j=(NY-2):-1:2
            D1(1,j)=(D1(1,j)-C1(1,j)*D1(1,j+1))/B1(1,j);
        end
        for j=2:NY-1
            u(i,j)=D1(1,j);
            e1=e1+abs(u(i,j)-c(i,j));
            e2=e2+abs(c(i,j));
        end
    end
    %% Updating BC
        for i=1:NX
            x=-pi+(i-1)*dx;
            y=-pi/2;
            u(i,1)=-(cos(2*x)+cos(2*y))/4;
            y=pi/2;
            u(i,NY)=-(cos(2*x)+cos(2*y))/4;
        end
        for j=1:NY
            y=-(pi/2)+(j-1)*dy;
            x=-pi;
            u(1,j)=-(cos(2*x)+cos(2*y))/4;
            x=pi;
            u(NX,j)=-(cos(2*x)+cos(2*y))/4;
        end
    e=e1/e2;
end
%% Error plot
e=0;
for j=2:NY-1
    y=-(pi/2)+(j-1)*dy;
```

x

```
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        P(i,j)=-(cos(x.*2)+cos(y.*2))/4;
        e=e+((u(i,j)-P(i,j))^2);
    end
end
Er(1,it)=(sqrt(e))/(NX*NY);
DX(1,it)=dx;
end
figure;
lg=loglog(DX,Er,"Marker","diamond");
acry=0;
for k=2:it
    acry = acry+((log(Er(1,k))-log(Er(1,1)))/(log(DX(1,k))-log(DX(1,1))));
end
acry = acry/(it-1);
txt = "Order of accuracy = " + acry;
text(0.11,0.0000065,txt);
title("ADI - Error vs \Deltax with \Deltay=" + dy + " at t=0");
xlabel('\Deltax (m)');
ylabel('Error in pressure (Pa)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.03, 0.005]);
xlim([0.095 0.13]);
ylim([0.000005 0.00001]);
grid on;
```

7.) Code to get **Error plot w.r.t Δy** using ADI method with Dirichlet BC along with spatial order of accuracy in y-direction:

```
clc;
e=1;
x=-pi;
y=-pi/2;
it=0;
for NY = [11 16 21 26]
it=it+1;
NX=101;
dx=2*pi/(NX-1);
dy=pi/(NY-1);
b=dx/dy;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
for i=1:NX
    x=-pi+(i-1)*dx;
    y=-pi/2;
    u(i,1)=-(cos(2*x)+cos(2*y))/4;
    y=pi/2;
    u(i,NY)=-(cos(2*x)+cos(2*y))/4;
end
for j=1:NY
    y=-(pi/2)+(j-1)*dy;
    x=-pi;
    u(1,j)=-(cos(2*x)+cos(2*y))/4;
    x=pi;
    u(NX,j)=-(cos(2*x)+cos(2*y))/4;
```

```
end
%% ADI iteration loop
while e>0.00001
    e=0;
    e1=0;
    e2=0;
    %% X-sweep
    for j=2:NY-1
        y=-(pi/2)+(j-1)*dy;
        for i=2:NX-1
            x=-pi+(i-1)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c(i,j)=u(i,j);
            A(i)=1;
            B(i)=-2*(1+(b^2));
            C(i)=1;
            D(i)=-(b^2)*(u(i,j+1)+u(i,j-1))-(f*(dx^2));
        end
        B(1,1)=-2*(1+(b^2));
        B(1,NX)=-2*(1+(b^2));
        D(1,2)=-(b^2)*(u(2,j+1)+u(2,j-1))-(f*(dx^2))-u(1,j);
        D(1,NX-1)=-(b^2)*(u(NX-1,j+1)+u(NX-1,j-1))-(f*(dx^2))-u(NX,j);
        for i=2:(NX-1)
            R=A(1,i)/B(1,i-1);
            B(1,i)=B(1,i)-R*C(1,i-1);
            D(1,i)=D(1,i)-R*D(1,i-1);
        end
        D(1,NX-1)=D(1,NX-1)/B(1,NX-1);
        for i=(NX-2):-1:2
            D(1,i)=(D(1,i)-C(1,i)*D(1,i+1))/B(1,i);
        end
        for i=2:NX-1
            u(i,j)=D(1,i);
        end
    end
    %% Y-sweep
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        for j=2:NY-1
            y=-(pi/2)+(j-1)*dy;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            A1(j)=b^2;
            B1(j)=-2*(1+(b^2));
            C1(j)=b^2;
            D1(j)=-u(i+1,j)-u(i-1,j)-(f*(dx^2));
        end
        B1(1,1)=-2*(1+(b^2));
        B1(1,NY)=-2*(1+(b^2));
        D1(1,2)=-u(i+1,2)-u(i-1,2)-(f*(dx^2))-(b^2)*u(i,1);
        D1(1,NY-1)=-u(i+1,NY-1)-u(i-1,NY-1)-(f*(dx^2))-(b^2)*u(i,NY);
        for j=2:(NY-1)
            R1=A1(1,j)/B1(1,j-1);
            B1(1,j)=B1(1,j)-R1*C1(1,j-1);
            D1(1,j)=D1(1,j)-R1*D1(1,j-1);
        end
        D1(1,NY-1)=D1(1,NY-1)/B1(1,NY-1);
        for j=(NY-2):-1:2
            D1(1,j)=(D1(1,j)-C1(1,j)*D1(1,j+1))/B1(1,j);
        end
```

```
        for j=2:NY-1
            u(i,j)=D1(1,j);
            e1=e1+abs(u(i,j)-c(i,j));
            e2=e2+abs(c(i,j));
        end
    end
    % Update BC
    for i=1:NX
        x=-pi+(i-1)*dx;
        y=-pi/2;
        u(i,1)=-(cos(2*x)+cos(2*y))/4;
        y=pi/2;
        u(i,NY)=-(cos(2*x)+cos(2*y))/4;
    end
    for j=1:NY
        y=-(pi/2)+(j-1)*dy;
        x=-pi;
        u(1,j)=-(cos(2*x)+cos(2*y))/4;
        x=pi;
        u(NX,j)=-(cos(2*x)+cos(2*y))/4;
    end
    e=e1/e2;
end
%% Error plot
e=0;
for j=2:NY-1
    y=-(pi/2)+(j-1)*dy;
    for i=2:NX-1
        x=-pi+(i-1)*dx;
        P(i,j)=-(cos(x.*2)+cos(y.*2))/4;
        e=e+((u(i,j)-P(i,j))^2);
    end
end
Er(1,it)=(sqrt(e))/(NX*NY);
DY(1,it)=dy;
end
figure;
lg=loglog(DY,Er,"Marker","diamond");
acry=0;
for k=2:it
    acry = acry+((log(Er(1,k))-log(Er(1,1)))/(log(DY(1,k))-log(DY(1,1))));
end
acry = acry/(it-1);
txt = "Order of accuracy = " + acry;
text(0.18,0.0002,txt);
title("ADI - Error vs \Deltay with \Deltax=" + dx + " at t=0");
xlabel('\Deltay (m)');
ylabel('Error in pressure (Pa)');
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.03, 0.005]);
xlim([0.12 0.32]);
ylim([0.00001 0.001]);
grid on;
```

8.) Code to plot pressure contours using **Gauss-Seidel method with Neumann boundary conditions** along with iteration count for case study:

```
clc;
e=1;
x=-pi;
y=-pi/2;
NX=301; % input NX
NY=151; % input NY
NX=NX+2;
NY=NY+2;
dx=2*pi/(NX-3);
dy=pi/(NY-3);
b=dx/dy;
it=0;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
% Neumann BC
for i=2:NX-1
    x=-pi+(i-2)*dx;
    y=-pi/2;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(i,1)=(u(i-1,1)+u(i+1,1)+(b^2)*(2*u(i,2))+(f*(dx^2)))/(2*(1+(b^2)));
    y=pi/2;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(i,NY)=(u(i-1,NY)+u(i+1,NY)+(b^2)*(2*u(i,NY-1))+(f*(dx^2)))/(2*(1+(b^2)));
end
for j=2:NY-1
    y=-(pi/2)+(j-2)*dy;
    x=-pi;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(1,j)=((2*u(2,j))+(b^2)*(u(1,j-1)+u(1,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
    x=pi;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(NX,j)=((2*u(NX-1,j))+(b^2)*(u(NX,j-1)+u(NX,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
end
%% Gauss-Seidel iteration loop
while e>0.00001
    e=0;
    it=it+1;
    e1=0;
    e2=0;
    for j=3:NY-2
        y=-(pi/2)+(j-2)*dy;
        for i=3:NX-2
            x=-pi+(i-2)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c=u(i,j);
            u(i,j)=(u(i-1,j)+u(i+1,j)+(b^2)*(u(i,j-
1)+u(i,j+1))+f*(dx^2))/(2*(1+(b^2)));
            e1=e1+abs(u(i,j)-c);
            e2=e2+abs(c);
        end
        % Neumann BC
        for i=2:NX-1
            x=-pi+(i-2)*dx;
```

```
            y=-pi/2;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            u(i,1)=(u(i-1,1)+u(i+1,1)+(b^2)*(2*u(i,2))+(f*(dx^2)))/(2*(1+(b^2)));
            y=pi/2;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            u(i,NY)=(u(i-1,NY)+u(i+1,NY)+(b^2)*(2*u(i,NY-
1))+(f*(dx^2)))/(2*(1+(b^2)));
        end
        for j=2:NY-1
            y=-(pi/2)+(j-2)*dy;
            x=-pi;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            u(1,j)=((2*u(2,j))+(b^2)*(u(1,j-
1)+u(1,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
            x=pi;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            u(NX,j)=((2*u(NX-1,j))+(b^2)*(u(NX,j-
1)+u(NX,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
        end
    end
    e=e1/e2;
end
%% Plot pressure contours
for i=1:NX-2
    for j=1:NY-2
        u1(i,j)=u(i+1,j+1);
    end
end
x=linspace(-pi,pi,NX-2);
y=linspace(-pi/2,pi/2,NY-2);
[X,Y]=meshgrid(x,y);
figure;
contourf(X,Y,u1');
%clim([-0.7 0.1]);
colorbar;
title('Pressure contours using G-S with Neumann BC');
xlabel('X (m)');
ylabel('Y (m)');
axis([-pi pi -pi/2 pi/2]);
set(gca,'XTick',-pi:pi/4:pi);
set(gca,'XTickLabel',{'-\pi','-3\pi/4','-\pi/2','-
\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'});
set(gca,'YTick',-pi/2:pi/4:pi/2);
set(gca,'YTickLabel',{'-\pi/2','-\pi/4','0','\pi/4','\pi/2'});
a=colorbar;
a.Label.String = 'Pressure (Pa)';
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
it %display iteration count
```

9.) Code to plot pressure contours using **ADI method with Neumann boundary conditions** along with iteration count for case study:

```
clc;
e=1;
x=-pi;
y=-pi/2;
NX=301; % input NX
NY=151; % input NY
NX=NX+2;
```

```
NY=NY+2;
dx=2*pi/(NX-3);
dy=pi/(NY-3);
b=dx/dy;
it=0;
%% Initialise
for j=1:NY
    for i=1:NX
        u(i,j)=0;
    end
end
% Neumann BC
for i=2:NX-1
    x=-pi+(i-2)*dx;
    y=-pi/2;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(i,1)=(u(i-1,1)+u(i+1,1)+(b^2)*(2*u(i,2))+(f*(dx^2)))/(2*(1+(b^2)));
    y=pi/2;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(i,NY)=(u(i-1,NY)+u(i+1,NY)+(b^2)*(2*u(i,NY-1))+(f*(dx^2)))/(2*(1+(b^2)));
end
for j=2:NY-1
    y=-(pi/2)+(j-2)*dy;
    x=-pi;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(1,j)=((2*u(2,j))+(b^2)*(u(1,j-1)+u(1,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
    x=pi;
    f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
    u(NX,j)=((2*u(NX-1,j))+(b^2)*(u(NX,j-1)+u(NX,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
end
%% ADI iteration loop
while e>0.00001
    e=0;
    it=it+1;
    e1=0;
    e2=0;
    %% X-sweep
    for j=3:NY-2
        y=-(pi/2)+(j-2)*dy;
        for i=3:NX-2
            x=-pi+(i-2)*dx;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            c(i,j)=u(i,j);
            A(i)=1;
            B(i)=-2*(1+(b^2));
            C(i)=1;
            D(i)=-(b^2)*(u(i,j+1)+u(i,j-1))-(f*(dx^2));
        end
        B(1,2)=-2*(1+(b^2));
        B(1,NX-1)=-2*(1+(b^2));
        D(1,3)=-(b^2)*(u(3,j+1)+u(3,j-1))-(f*(dx^2))-u(2,j);
        D(1,NX-2)=-(b^2)*(u(NX-2,j+1)+u(NX-2,j-1))-(f*(dx^2))-u(NX-1,j);
        for i=3:(NX-2)
            R=A(1,i)/B(1,i-1);
            B(1,i)=B(1,i)-R*C(1,i-1);
            D(1,i)=D(1,i)-R*D(1,i-1);
        end
        D(1,NX-2)=D(1,NX-2)/B(1,NX-2);
        for i=(NX-3):-1:3
```

```
            D(1,i)=(D(1,i)-C(1,i)*D(1,i+1))/B(1,i);
        end
        for i=3:NX-2
            u(i,j)=D(1,i);
        end
    end
    %% Y-sweep
    for i=3:NX-2
        x=-pi+(i-2)*dx;
        for j=3:NY-2
            y=-(pi/2)+(j-2)*dy;
            f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
            A1(j)=b^2;
            B1(j)=-2*(1+(b^2));
            C1(j)=b^2;
            D1(j)=-u(i+1,j)-u(i-1,j)-(f*(dx^2));
        end
        B1(1,2)=-2*(1+(b^2));
        B1(1,NY-1)=-2*(1+(b^2));
        D1(1,3)=-u(i+1,3)-u(i-1,3)-(f*(dx^2))-(b^2)*u(i,2);
        D1(1,NY-2)=-u(i+1,NY-2)-u(i-1,NY-2)-(f*(dx^2))-(b^2)*u(i,NY-1);
        for j=3:(NY-2)
            R1=A1(1,j)/B1(1,j-1);
            B1(1,j)=B1(1,j)-R1*C1(1,j-1);
            D1(1,j)=D1(1,j)-R1*D1(1,j-1);
        end
        D1(1,NY-2)=D1(1,NY-2)/B1(1,NY-2);
        for j=(NY-3):-1:3
            D1(1,j)=(D1(1,j)-C1(1,j)*D1(1,j+1))/B1(1,j);
        end
        for j=3:NY-2
            u(i,j)=D1(1,j);
            e1=e1+abs(u(i,j)-c(i,j));
            e2=e2+abs(c(i,j));
        end
    end
    % Neumann BC
    for i=2:NX-1
        x=-pi+(i-2)*dx;
        y=-pi/2;
        f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
        u(i,1)=(u(i-1,1)+u(i+1,1)+(b^2)*(2*u(i,2))+(f*(dx^2)))/(2*(1+(b^2)));
        y=pi/2;
        f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
        u(i,NY)=(u(i-1,NY)+u(i+1,NY)+(b^2)*(2*u(i,NY-
1))+(f*(dx^2)))/(2*(1+(b^2)));
    end
    for j=2:NY-1
        y=-(pi/2)+(j-2)*dy;
        x=-pi;
        f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
        u(1,j)=((2*u(2,j))+(b^2)*(u(1,j-1)+u(1,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
        x=pi;
        f=-(cos(x).^2)+(sin(x).^2)-(cos(y).^2)+(sin(y).^2);
        u(NX,j)=((2*u(NX-1,j))+(b^2)*(u(NX,j-
1)+u(NX,j+1))+(f*(dx^2)))/(2*(1+(b^2)));
    end
    e=e1/e2;
end
```

```
%% Plot pressure contours
for i=1:NX-2
    for j=1:NY-2
        u1(i,j)=u(i+1,j+1);
    end
end
x=linspace(-pi,pi,NX-2);
y=linspace(-pi/2,pi/2,NY-2);
[X,Y]=meshgrid(x,y);
figure;
contourf(X,Y,u1');
%clim([-0.7 0.1]);
colorbar;
title('Pressure contours using ADI with Neumann BC ');
xlabel('X (m)');
ylabel('Y (m)');
axis([-pi pi -pi/2 pi/2]);
set(gca,'XTick',-pi:pi/4:pi);
set(gca,'XTickLabel',{'-\pi','-3\pi/4','-\pi/2','-
\pi/4','0','\pi/4','\pi/2','3\pi/4','\pi'});
set(gca,'YTick',-pi/2:pi/4:pi/2);
set(gca,'YTickLabel',{'-\pi/2','-\pi/4','0','\pi/4','\pi/2'});
a=colorbar;
a.Label.String = 'Pressure (Pa)';
set(gca,'XMinorTick','on','YMinorTick','on','TickLength',[0.02,0.005]);
it %display iteration count
```

10.)Code to plot **comparison plots** for error values and iteration counts obtained using both Gauss-Seidel & ADI methods:

```
clc;
%% Error comparison
e1=[0.000024241 0.000044519 0.0001006 0.0000734 0.000074894 0.000061542
0.000060258];
e2=[0.0000040296 0.000010431 0.0000050339 0.000018191 0.000021662 0.000036915
0.000032332];
figure;
lg=semilogy(e1,"Marker","square");
hold on;
lg1=semilogy(e2,"Marker","diamond");
title('Error comparison');
ylabel('Error values (Pa)');
xlabel('Case number');
ylim([0.000001 0.001]);
legend('Gauss-Seidel error values','ADI error values');
set(gca,'YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
hold off;
%% Compare iterations for Dirichlet BC
it1=[3120 6328 7377 7938 12819 16894 27856];
it2=[800 2896 1689 5376 6515 7915 9898];
figure;
plot(it1,"Marker","square");
hold on;
plot(it2,"Marker","diamond");
title('Iteration count comparison with Dirichlet BC');
ylabel('No. of iteration for converged solution');
xlabel('Case number');
```

```
legend('Gauss-Seidel iteration count','ADI iteration
count','Location','northwest');
set(gca,'YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
hold off;
%% Compare iterations for Neumann BC
it3=[4638 12082 23549 42363];
it4=[1041 4271 9298 23549];
figure;
plot(it3,"Marker","square");
hold on;
plot(it4,"Marker","diamond");
title('Iteration count comparison with Neumann BC');
ylabel('No. of iteration for converged solution');
xlabel('Case number');
set(gca,'XTick',1:1:4);
legend('Gauss-Seidel iteration count','ADI iteration
count','Location','northwest');
set(gca,'YMinorTick','on','TickLength',[0.02,0.005]);
grid on;
hold off;
```