Solution of Prandtl-Glauert equation for an inviscid, non-heat conducting perfect gas free flow with small perturbation

Sagar Bhatt
Person Number: 50170651
Department of Mechanical and Aerospace Engineering,
University at Buffalo

1. Introduction:

1.1 Problem:

A uniform inviscid subsonic flow $U_\infty (M_\infty < 1)$ hits a one period sign wave wrinkle in the metal skin of the wind tunnel. The geometry of the configuration is given below. The maximum amplitude of the sign wave $\varepsilon / L \ll 1$ is small. Therefore, the Prandtl-Glauert equation for an inviscid, non-heat conducting perfect gas free flow, which is only slightly perturbed by a thin body such that fluid motion satisfies the small perturbation assumption, can be considered as the governing equation:

$$(1-M^2)\varphi_{xx}+\varphi_{yy}=0$$

where M_∞ is the free stream Mach number, and the perturbation potential is denoted by φ. The perturbation velocity components can be recovered by $u=\frac{\partial \varphi}{\partial x}$ and $v=\frac{\partial \varphi}{\partial y}$.

The Boundary conditions are Neumann type, which are: $\frac{\partial \varphi}{\partial x}(x,0)=U_\infty \frac{dy}{dx}=U_\infty \varepsilon \cos(x)$ at $y=0, \frac{\partial \varphi}{\partial n}=0$ on all the others.

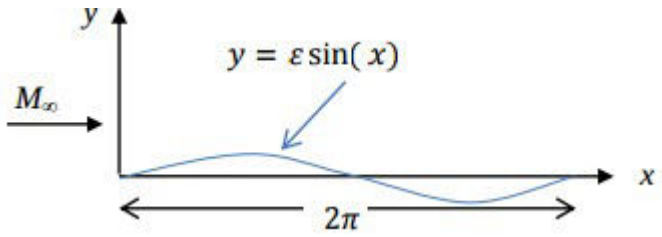

Figure 1

1.2 Background Information:

The given governing equation is a 2-D Elliptic Equation. Such equations do not have any real characteristic curves. Any disturbance propagates instantly in all directions. They are also called '*Boundary Value'* problems because such equations only need boundary conditions and no initial condition is required.

1.2.1 Successive Over Relaxation Method

If during the solution process, a trend in computed values of dependent variable is noticed, then the direction change can be used to extrapolate to extrapolate for the next iteration and ,thereby, accelerate the solution procedure. This method is known as '*Successive Over Relaxation'* (SOR).

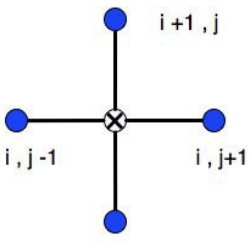

Figure 2[3]

To accelerate the solution procedure, the relaxation parameter, ω , is introduced. For the solution to converge, it is necessary that $o<\omega<2$. If $0<\omega<1$, it is called under-relaxation.

In general, ω_{opt} cannot be determined easily. Therefore, for most cases, numerical experimentation is performed.[1]

1.2.2 Alternating Direction Implicit Method

When we form the coefficient matrix for the discretized form of the governing equation, we end up with a pentadiagonal matrix. Pentadiagonal matrices are very time-consuming to solve and also increase our computational cost. One way to overcome this problem and inefficiency of the solution is to use a splitting method. This method is known as *'Alternating Direction Implicit Method'* or ADI.

In this method, we first solve the problem implicitly in x-direction, the solution at this stage in known as *'x-sweep'*. These vales are then used as starting points to solve the same equation implicitly in y-direction, it is referred to as *'y-sweep'* at this point.

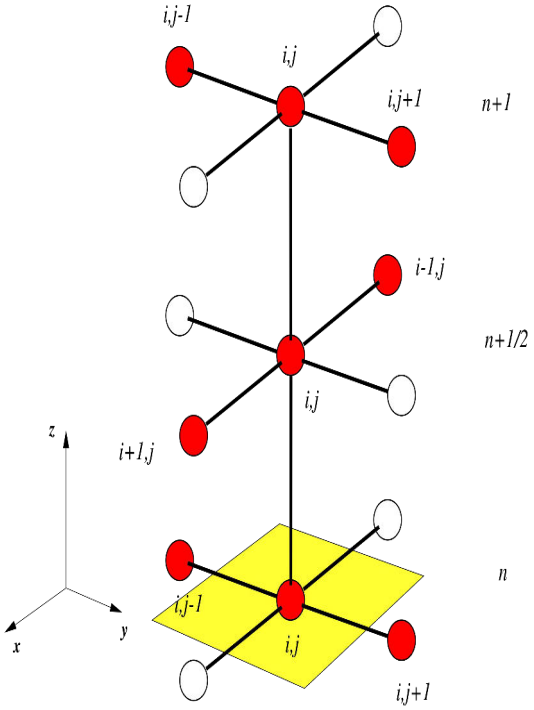

Figure 3[2]

In both x-sweep and y-sweep, we encounter a tridiagonal system. This system is much easily solved , using a tridiagonal solver that employs Thomas Algorithm, than a pentadiagonal system.

The method is of the order $(\Delta x^{2,} \Delta Y^2)$, and is unconditionally stable.[1]

2. Method of Solution:

2.1 Compatibility condition:

Given equation: $(1-M^2)\varphi_{xx}+\varphi_{yy}=0$

$$\Rightarrow(1-M^2)\frac{\partial^2\varphi}{\partial x^2}+\frac{\partial^2\varphi}{\partial y^2}$$

$$\Rightarrow \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{(1-M^2)\partial y^2}$$

let $\bar{y} = (1-M^2)\, y$

$$\Rightarrow \overline{dy} = (1-M^2)\, dy$$

$$\Rightarrow f = \nabla^2 \varphi$$

$$\Rightarrow \iiint (f)\, dV = 0$$

Now, $g = \frac{d\varphi}{dn}$

As per given boundary condition, $g = 0$ on three sides . And on the botton boundary,

$$\frac{d\varphi}{dy} = U_\infty \varepsilon \cos(x) = 0.1\cos(x)$$

Now,

$$\iint g dS = \int_0^{2\pi} 0.1\cos(x)\, dx$$

$$\Rightarrow \iint g dS = 0 = \iiint f dV$$

Hence, the boundary condition satisfies the compatibility solution.

2.2 Analytical Solution

First the analytical solution was evaluated,to get an idea of what the solution looks like and what to expect from the results. For this purpose,the following analytical solution was used, as given in the problem:

$$\varphi(x,y) = \frac{-U_\infty \varepsilon}{\sqrt{(1-M_\infty^2)}} \left(e^{\left(-\left(\sqrt{1-M_\infty^2}\right) y\right)} \right) \cos(x)$$

The perturbation potential was plotted and velocities were plotted by evaluating $u = \frac{\partial \varphi}{\partial x}$ and $v = \frac{\partial \varphi}{\partial y}$.

The code for analytical solution can be found in Appendix A.

Then, the governing equation was discretized as follows:

$$(1-M^2)\varphi_{xx} + \varphi_{yy} = 0$$

$$\Rightarrow (1-M^2) \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\Delta x^2} + \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\Delta y^2} = 0$$

In both x-sweep and y-sweep, we encounter a tridiagonal system. This system is much easily solved , using a tridiagonal solver that employs Thomas Algorithm, than a pentadiagonal system.

2.3 Successive Over Relaxation (SOR):

The next step was to solve the equation with Successive Over Relaxation Method (SOR). For this method, Gauss Seidel Method was used to evaluate $\varphi_{i,j}$. Then a relaxation of ω was introduced. The value of ω was taken as 1.25 for optimum result.

Gauss Seidel: $$\varphi_{i-1,j}^{n+1}-2(1+\beta^2)\varphi_{i,j}^{n+1}+\varphi_{i+1,j}^{n+1}=-\beta^2(\varphi_{i,j+1}^{n}\varphi_{i,j-1}^{n+1})$$

$$\Rightarrow\varphi_{i,j}^{n+1}=\left(\frac{1}{1+\beta^2}\right)(\varphi_{i+1,j}^{n}+\varphi_{i-1,j}^{n+1}+\beta^2(\varphi_{i,j+1}^{n}+\varphi_{i,j-1}^{n+1}))$$

Relaxation of point (i,j),

$$\varphi_{i,j}^{n+1}=(1-\omega)\varphi_{i,j}^{n}+\omega\,\overline{\varphi}_{i,j}^{n+1} \quad \text{, where} \quad \omega=1.25$$

The following convergence criteria for the solution was used as per the problem:

$$\frac{\sum_{j=1}^{JM}\sum_{i=1}^{IM}\left|\varphi_{i,j}^{k+1}-\varphi_{i,j}^{k}\right|}{\sum_{j=1}^{JM}\sum_{i=1}^{IM}\left|\varphi_{i,j}^{k}\right|}<\epsilon \quad \text{, where} \quad \epsilon \quad \text{is a small number, e.g.} \quad 10^{-5}$$

The perturbation potential as well as velocity contours were plotted.

A log-log plot of error function is obtained in order to ascertain the order of accuracy.

The equation used to evaluate error was:

$$error=\frac{1}{JMxIM}\sqrt{\sum_{j=1}^{JM}\sum_{i=1}^{IM}(\varphi_{i,j}^{computed}-\varphi_{i,j}^{analytical})^2}$$

Similar plots were obtained for finer grids in order to show that the method converges to a grid independent solution.

The code for SOR method can be found in Appendix B.

2.4 Alternating Direction Implicit (ADI):

In this method the following equation was used:

$$\varphi_{i+1,j}-2\varphi_{i,j}+\varphi_{i-1,j}+\frac{\beta^2}{(1-M^2)}\varphi_{i,j+1}-2\varphi_{i,j}+\varphi_{i,j-1}=0 \quad \text{, where} \quad \beta=\frac{\Delta x}{\Delta y}$$

hence, X-Sweep:

$$\varphi_{i+1,j}^{n+1/2}-2\left(1+\frac{\beta^2}{(1-M^2)}\right)\varphi_{i,j}^{n+1/2}+\varphi_{i-1,j}^{n+1/2}=-\frac{\beta^2}{(1-M^2)}(\varphi_{i,j+1}^{n}+\varphi_{i,j-1}^{n+1/2})$$

and, Y-Sweep:

$$\left(\frac{\beta^2}{(1-M^2)}\right)\varphi_{i+1,j}^{n+1}-2\left(1+\frac{\beta^2}{(1-M^2)}\right)\varphi_{i,j}^{n+1}+\left(\frac{\beta^2}{(1-M^2)}\right)\varphi_{i-1,j}^{n+1}=-(\varphi_{i+1,j}^{n+1/2}+\varphi_{i-1,j}^{n})$$

The matrices for this method are lengthy and for further information regarding the coefficient matrices

generated during this solution, please refer the code for ADI in Appendix C.

Since the given boundary conditions were Neumann type, we didn't know the exact boundary values, hence to employ the boundary conditions in this method, 'ghost-points' were used.

In each sweep, the two tridiagonal matrices were generated. To solve these matrices, a tridiagonal solver function was written. The code for this tridiagonal solver can be found in Appendix D.

The some convergence criteria for the solution was used as discussed above:

The perturbation potential as well as velocity contours were plotted.

A log-log plot of error function is obtained in order to ascertain the order of accuracy. The equation to evaluate error was the same as used above

Similar plots were obtained for finer grids in order to show that the method converges to a grid independent solution.

3. Discussion of Results:

3.1 Analytical Solution:

Consider the following plots of Perturbation potential, u and v (i.e. x and y components of velocities):

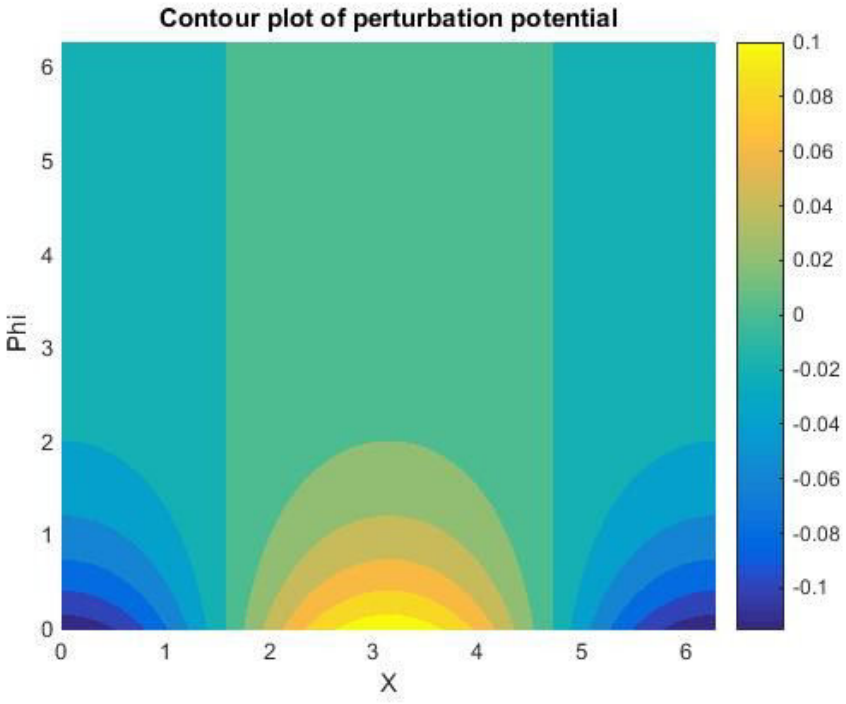

Figure 4

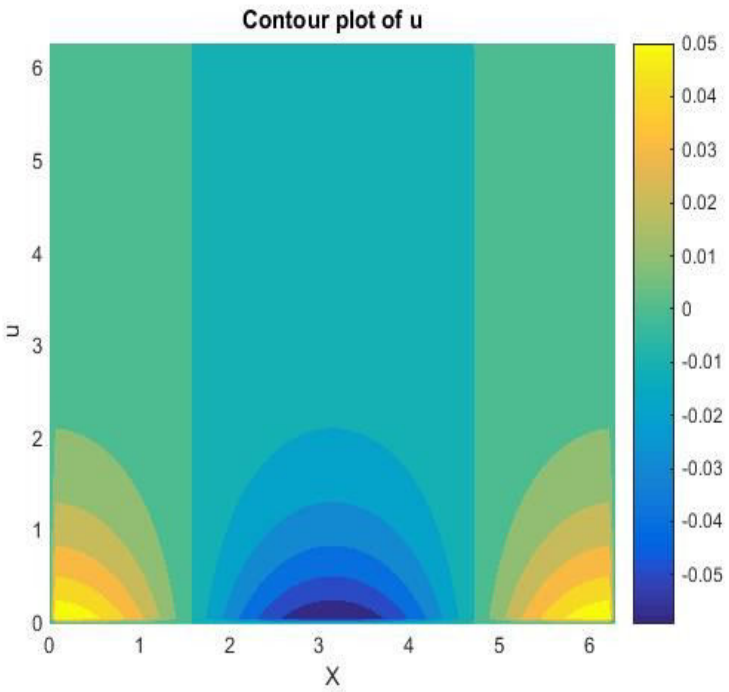

Figure 5

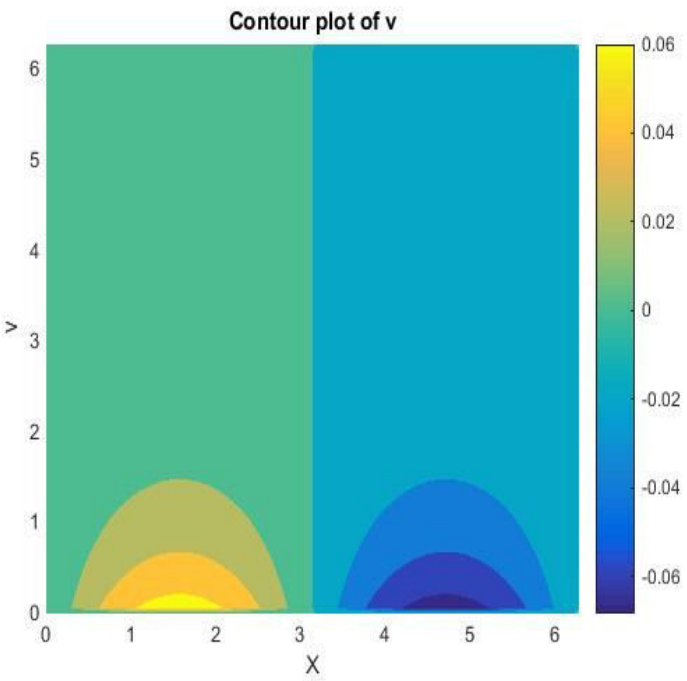

Figure 6

Figure 4, 5 and 6 represent the results obtained through analytical method. They will serve as a benchmark to compare our results.

3.2 SOR:

The following plots represent the solutions obtained through SOR method:

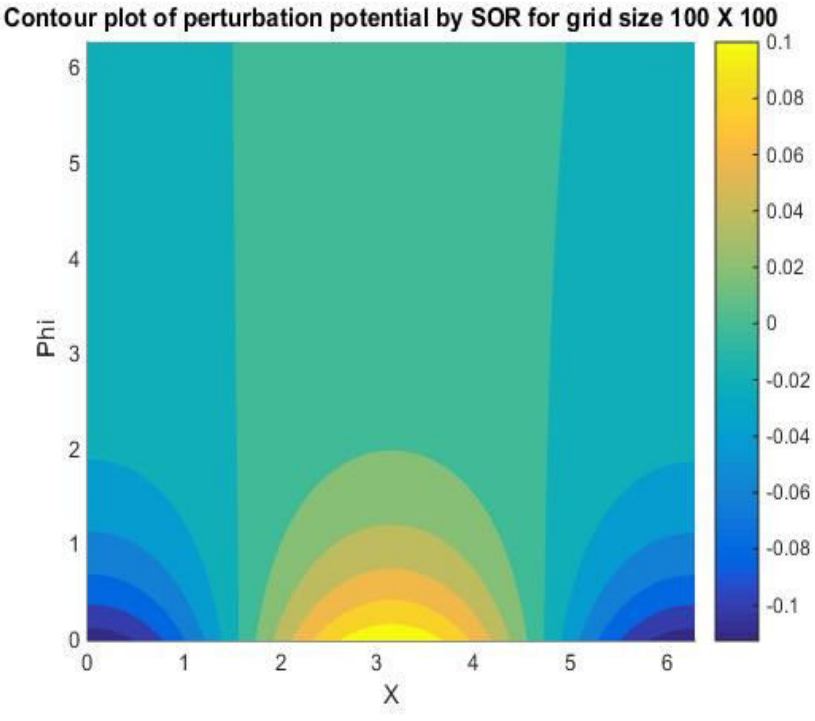

Figure 7

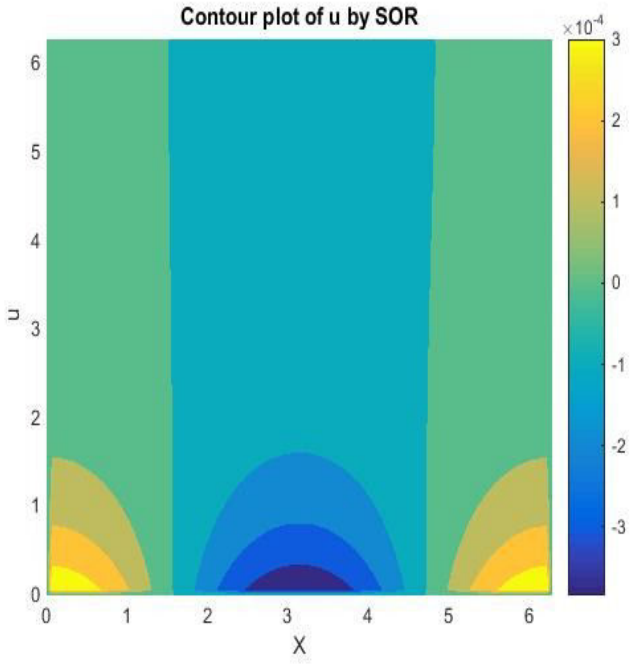

Figure 8

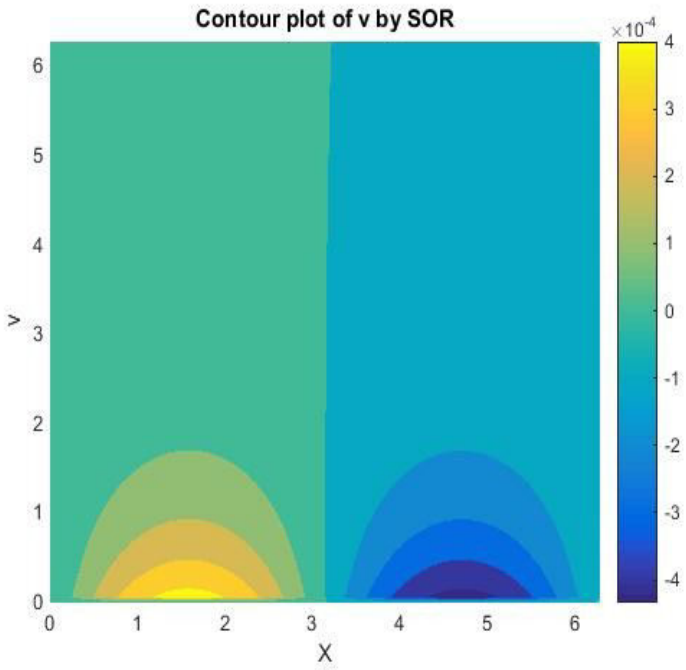

Figure 9

Comparing Fig. 7, Fig. 8 and Fig. 9 with the corresponding plots from analytical solution, we can say that our method for solving the problem using SOR is correct. To further investigate the accuracy of the method, we analyze the plots at finer grids- which will establish the grid independence of the method:

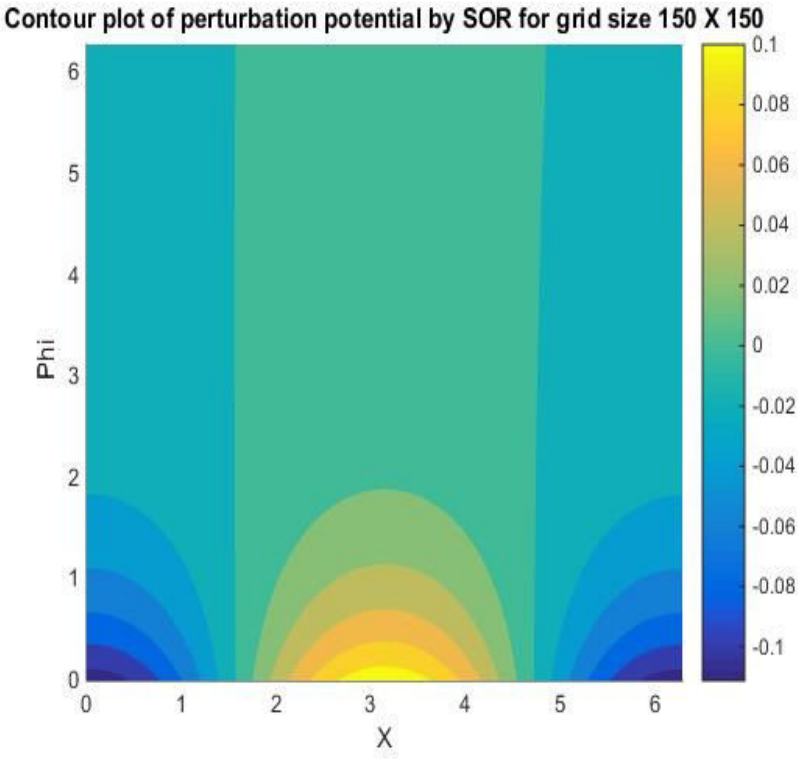

Figure 10

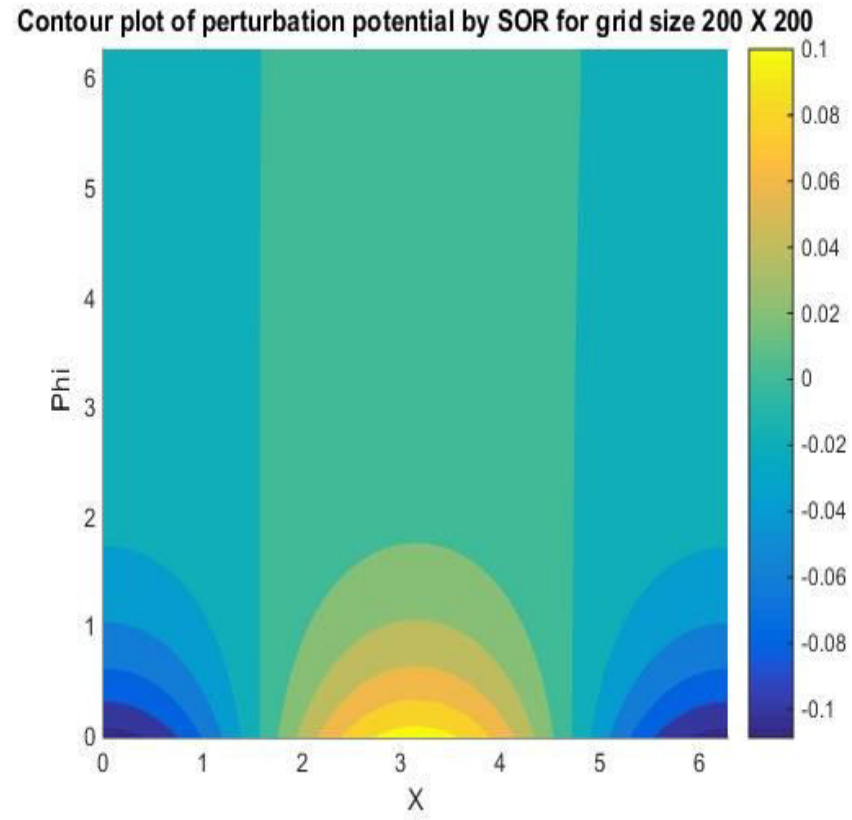

Figure 11

After comparing Fig. 10 and Fig. 11 with Fig. 7, we infer that the method is grid independent and the values do not change much . The exact change in values and the error therein can be found in the next plot which describes the error:

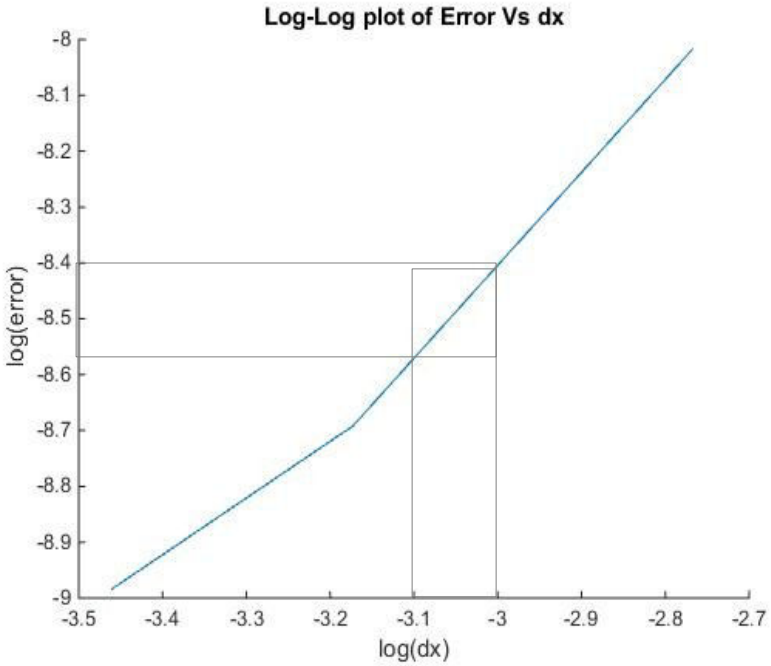

Figure 12

Figure 12 is a log-

log plot of error vs dx. By analyzing the above plot, we can infer that as dx (i.e. cell width) decreases, the error also decreases. The *slope* of the log-log curve in fig. 9 is nearly 2. This implies that the method is nearly *second order accurate in space.*

$$slope \approx \frac{-8.4-(-8.58)}{-3-(-3.1)} \approx 1.8$$

3.3 ADI:

Plots of solution obtained from ADI method are as follows:

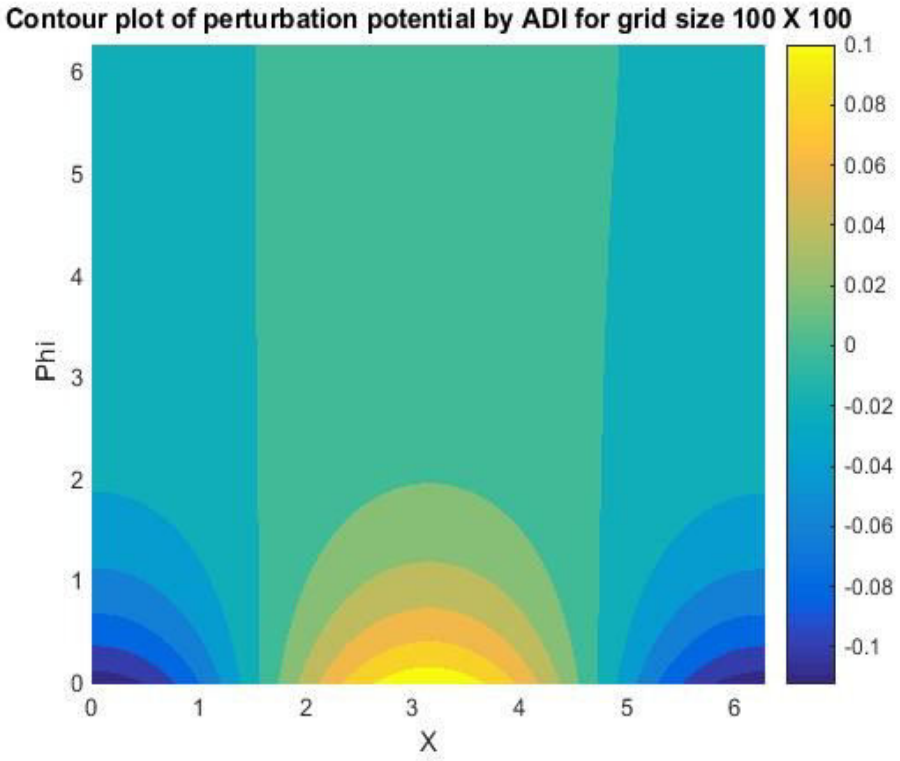

Figure 13

Comparing Fig. 10, 11 and 12, we observe that the results are in agreement with the analytical solution evaluated in section 3.1. From this, we can infer that our method to evaluate the solution by ADI is correct.

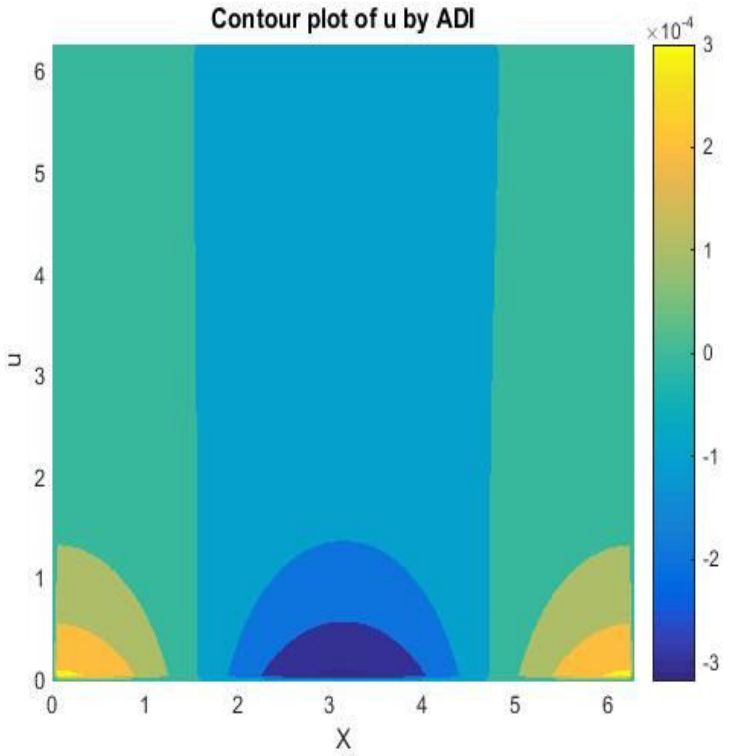

Figure 14

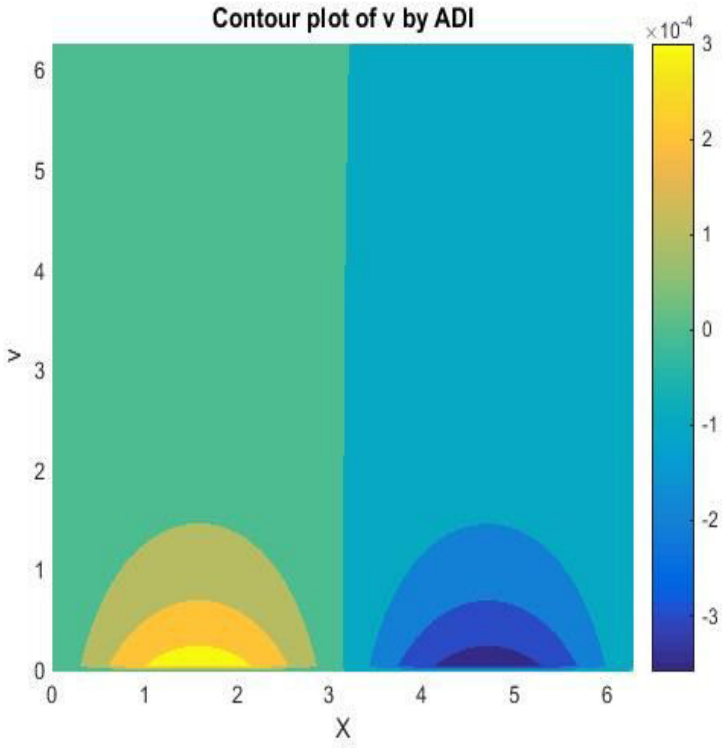

Figure 15

To further investigate the accuracy of the method, we analyze the plots at finer grids- which will establish the grid independence of the method:

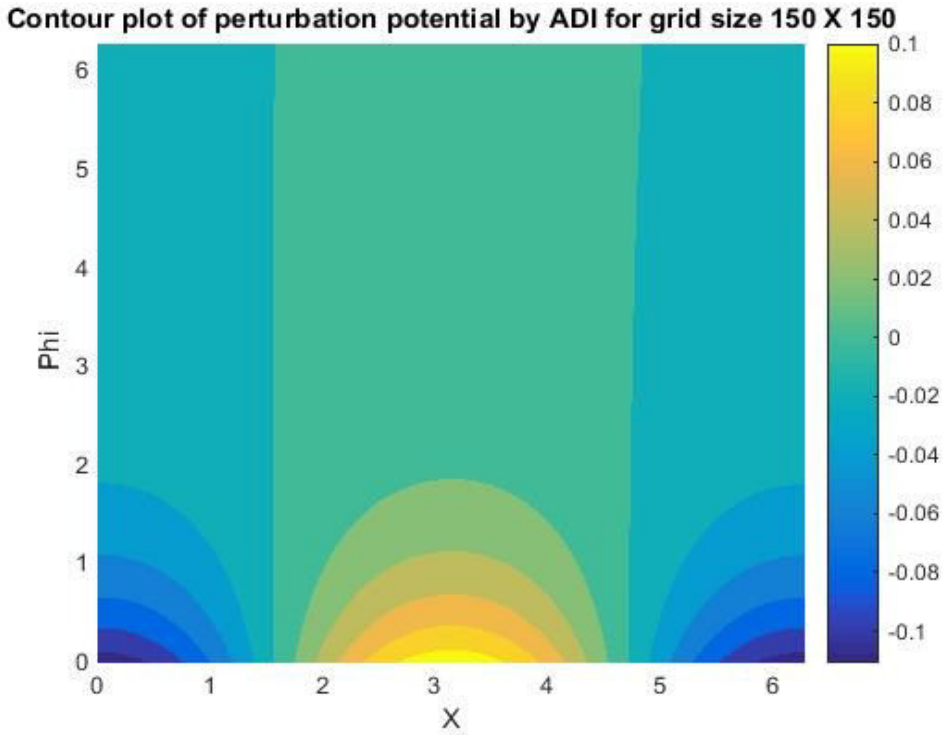

Figure 16

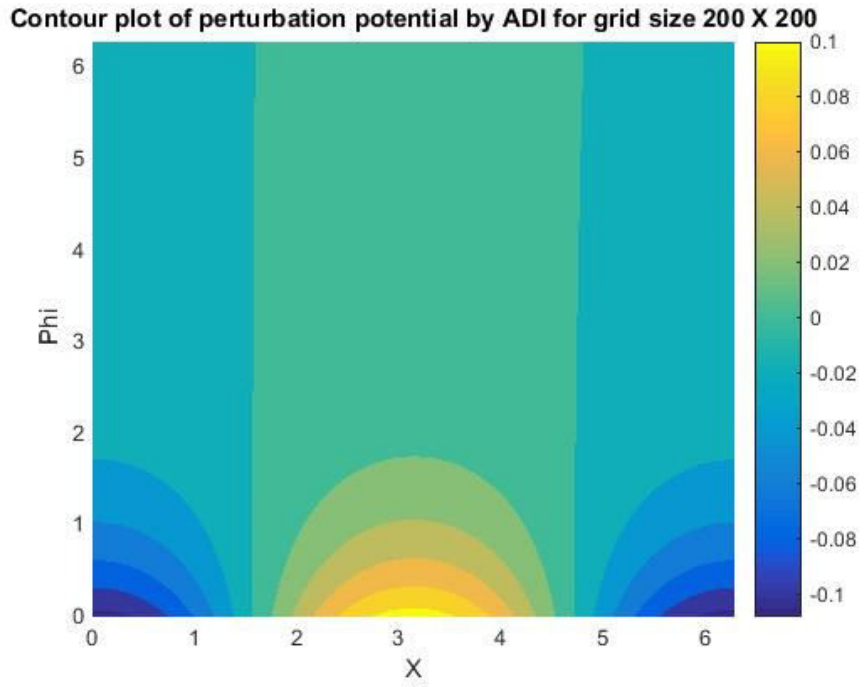

Figure 17

On comparing Fig. 13 and Fig. 14 with Fig. 10, we observe that the values are very slightly effected. Hence, we can conclude that the method converges to a grid independent solution. The exact change in values i.e. the error for different grid sizes is plotted next:

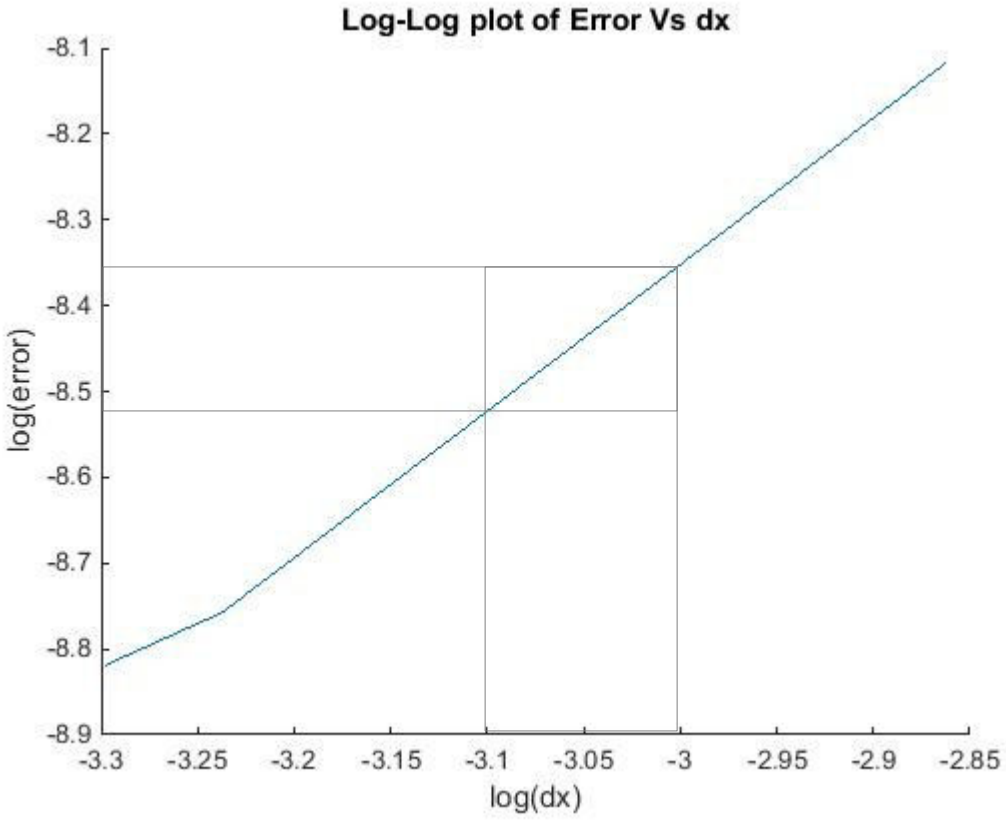

Figure 18

Figure 15 is a log-log plot of error vs dx. By analyzing the above plot, we can infer that as dx (i.e. cell width) decreases, the error also decreases. The *slope* of the log-log curve in fig. 15 is nearly 2. This implies that the method is nearly *second order accurate in space.*

$$slope \approx \frac{-8.52-(-8.35)}{-3-(-3.1)} \approx 1.7$$

This value of slope of log-log plot or order of accuracy is precisely what was expected as our method was second order accurate in space for internal points and first order accurate on the boundaries. Hence, we were expecting order of accuracy closer to 2 but not not exactly 2.

4. Summary and Conclusions:

From the solutions evaluated n this problem, it is apparent that the SOR method is much faster than ADI method since it reaches convergence faster due to presence of a relaxation parameter. The results, however, do not shed a light on which method would be more accurate since the results are nearly the same and so is the order of accuracy.

With regard to computational cost, ADI is definitely the more expensive method owing to the number of iterations and the time take by the method to achieve convergence. SOR achieves convergence faster and in lesser number of iterations, hence it is the cheaper of the two methods.

5. Appendix

5.1 Appendix A

```
% MAE 542 Midterm Project
%(1-M^2)phi_xx+phi_yy=0
%
%Analytical Solution
clear all;
clc;

%-------constants------
M=0.5;
U=1;
L=2*pi;
ep=0.1;
N=100;
dx=L/2*pi;
dy=L/2*pi;
dphi_dx=zeros(N);
```

```
dphi_dy=zeros(N);

%-------initializing grid-----------
x=linspace(0,2*pi);
y=linspace(0,2*pi);
[X,Y] = meshgrid(x,y);

%-------------analytical solution-------------
phi=((-U.*ep).*exp(-(sqrt(1-M^2).*Y)).*cos(X))./sqrt(1-M^2);

figure
hold on;
contourf(X,Y,phi,'edgecolor','none'); %contour plot

for j=2:N-1
    for i=2:N-1
    dphi_dx(i,j) = ((phi(i+1,j)-phi(i-1,j))/2*dx);
    dphi_dy(i,j)=((phi(i,j+1)-phi(i,j-1))/2*dy);
    end
 end
  figure
 hold on;
 contourf(X,Y,dphi_dx,'edgecolor','none'); %contour plot

  figure
 hold on;
 contourf(X,Y,dphi_dy,'edgecolor','none'); %contour plot
```

5.2 Appendix B

```
%------------SOR_New--------

clear all;
clc;

%-------constants------
M=0.5;
U=1;
ep=0.1;
E=10e-5;
```

```
EP=1;
N=[100 150 200];
for g=1:3
w=1.25;
dx(g)=2*pi/(N(g));
dy(g)=2*pi/(N(g));
p=(dx(g)^2)/((1-M^2)*(dy(g)^2));
phi_old=zeros(N(g));
phi=zeros(N(g));
dphi_dx=zeros(N(g));
dphi_dy=zeros(N(g));
%-------initializing grid-----------
x=linspace(0,2*pi,N(g));
y=linspace(0,2*pi,N(g));
[X,Y] = meshgrid(x,y);

for i=1:N(g)
phi_old(i,1)=phi_old(i,2)-dy(g)*U*ep*cos((x(i)));
end
for i=1:N(g)
phi(i,1)=phi(i,2)-dy(g)*U*ep*cos((x(i)));
end

while EP>E

    for i=2:N(g)-1
        for j=2:N(g)-1
            phi(i,j)=(1/(2*(1+p)))*(phi_old(i+1,j)+phi(i-1,j)
+p*(phi_old(i,j+1)+phi(i,j-1)));
        end
    end

phi(1,:)=phi(2,:);
phi(N(g),:)=phi(N(g)-1,:);
phi(:,N(g))=phi(:,N(g)-1);

for i=1:N(g)
phi(i,1)=phi(i,2)-dy(g)*U*ep*cos((x(i)));
end
```

```
for j=1:N(g)
 for i=1:N(g)
     phi(i,j)=(1-w)*phi_old(i,j)+w*phi(i,j);
%      phi_old(i,j)=phi(i,j);
 end
end

for j=1:N(g)
    for i=1:N(g)
        phi_dummy(i,j)=phi_old(i,j);
        phi_old(i,j)=phi(i,j);
    end
end
s=abs(phi-phi_dummy);
EP=(sum(s(:)))/sum(abs(phi_dummy(:)));



end


 %-------plot-----------------------
 figure
 hold on;
 contourf(Y,X,phi,'edgecolor','none'); %contour plot
 xlabel('X');
ylabel('Phi');
str=sprintf('Contour plot of perturbation potential by SOR for grid size %i X
%i',N(g),N(g));
title(str);



 for j=2:N(g)-1
    for i=2:N(g)-1
    dphi_dx(i,j) = ((phi(i+1,j)-phi(i-1,j))/2*dx(g));
    dphi_dy(i,j)=((phi(i,j+1)-phi(i,j-1))/2*dy(g));
    end
 end
  figure
 hold on;
 contourf(Y,X,dphi_dx,'edgecolor','none'); %contour plot
```

```
  xlabel('X');
ylabel('v');
title('Contour plot of v by SOR');

  figure
 hold on;
 contourf(Y,X,dphi_dy,'edgecolor','none'); %contour plot
 xlabel('X');
ylabel('u');
title('Contour plot of u by SOR');

 phi_comp=((-U.*ep).*exp(-(sqrt(1-M^2).*Y)).*cos(X))./sqrt(1-M^2);
 for i=1:N(g)
     for j=1:N(g)
 rms(i,j)=(phi(i,j)-phi_comp(i,j))^2;
     end
 end
 r=sum(rms(:));
 error(g)=(1/(N(g)*N(g)))*sqrt(r);
end
 figure;
 hold on;
 plot(log(dx),log(error));
 xlabel('log(dx)');
 ylabel('log(error)');
 title('Log-Log plot of Error Vs dx');
```

5.3 Appendix C

```
% MAE 542 Midterm Project
%(1-M^2)phi_xx+phi_yy=0
%
%Solution Using ADI
clear all;
clc;


%-------constants------
M=0.5;
U=1;
L=2*pi;
```

```
ep=0.1;
EX=10e-5;
EP=1;
N=[100 150 200];
for g=1:3
dx(g)=2*pi/(N(g));
dy(g)=2*pi/(N(g));
p=(dx(g)^2)/((1-M^2)*(dy(g)^2));
phi_old=zeros(N(g));
phi=zeros(N(g));
phi_new=zeros(N(g));
%-------initializing grid-----------
x=linspace(0,2*pi,N(g));
y=linspace(0,2*pi,N(g));
[X,Y] = meshgrid(x,y);

while EP>EX

for j=1:N(g)
    A(1,j)=0;
    B(1,j)=-1;
    C(1,j)=1;
    D(1,j)=0;

end

for i=2:N(g)-1
    for j=2:N(g)-1

        A(i,j)=0.75/(dx(g)*dx(g));
        B(i,j)=-2*((0.75/(dx(g)*dx(g)))+1/(dy(g)*dy(g)));
        C(i,j)=0.75*(1/(dx(g)*dx(g)));
        D(i,j)=-(dy(g)^(-2))*(phi_old(i,j+1)+phi(i,j-1));

    end
end

for j=1:N(g)
    A(N(g),j)=-1;
    B(N(g),j)=1;
```

```
    C(N(g),j)=0;
    D(N(g),j)=0;
end

TRI_2D_X(1,N(g),2,N(g)-1,A,B,C,D);
for j=1:N(g)
for i=1:N(g)
     phi(i,j)=D(i,j);
     phi_old(i,j)=phi(i,j);
end
end
phi(1,1)=phi(2,1);
phi(N(g),1)=phi(N(g)-1,1);
phi(1,N(g))=phi(2,N(g));
phi(N(g),N(g))=phi(N(g)-1,N(g));
% ---------------------------------------------

for i=1:N(g)

    E(i,1)=0;
    F(i,1)=-1;
    G(i,1)=1;
    H(i,1)=2*dy(g)*0.1*cos(x(i));

end

for i=2:N(g)-1
    for j=2:N(g)-1

        E(i,j)=(dy(g)^(-2));
        F(i,j)=-2*((0.75*(1/(dx(g)*dx(g))))+1/(dy(g)*dy(g)));
        G(i,j)=(1/(dy(g)*dy(g)));
        H(i,j)=-(0.75*(1/(dx(g)*dx(g))))*(phi(i,j+1)+phi_new(i,j-1));

    end
end

for i=1:N(g)
    E(i,N(g))=-1;
    F(i,N(g))=1;
```

```
    G(i,N(g))=0;
    H(i,N(g))=0;
end

TRI_2D_Y(2,N(g)-1,1,N(g),E,F,G,H);
for j=1:N(g)
for i=1:N(g)
     phi_new(i,j)=H(i,j);
     phi(i,j)=phi_new(i,j);
end
end

phi_new(1,1)=phi_new(2,1);
phi_new(N(g),1)=phi_new(N(g)-1,1);
phi_new(1,N(g))=phi_new(2,N(g));
phi_new(N(g),N(g))=phi_new(N(g)-1,N(g));

for j=1:N(g)
    for i=1:N(g)
        phi_dummy(i,j)=phi_old(i,j);
        phi_old(i,j)=phi(i,j);
    end
end
s=abs(phi-phi_dummy);
EP=(sum(s(:)))/sum(abs(phi_dummy(:)));



end
```

```
 %-------plot-----------------------
 figure
 hold on;
 contourf(Y,X,phi,'edgecolor','none'); %contour plot
 xlabel('X');
ylabel('Phi');
str=sprintf('Contour plot of perturbation potential by ADI for grid size %i X
%i',N(g),N(g));
```

```
title(str);


 for j=2:N(g)-1
    for i=2:N(g)-1
    dphi_dx(i,j) = ((phi(i+1,j)-phi(i-1,j))/2*dx(g));
    dphi_dy(i,j)=((phi(i,j+1)-phi(i,j-1))/2*dy(g));
    end
 end
  figure
 hold on;
 contourf(Y,X,dphi_dx,'edgecolor','none'); %contour plot
  xlabel('X');
ylabel('v');
title('Contour plot of v by ADI');

  figure
 hold on;
 contourf(Y,X,dphi_dy,'edgecolor','none'); %contour plot
 xlabel('X');
ylabel('u');
title('Contour plot of u by ADI');

 phi_comp=((-U.*ep).*exp(-(sqrt(1-M^2).*Y)).*cos(X))./sqrt(1-M^2);
 for i=1:N(g)
     for j=1:N(g)
 rms(i,j)=(phi(i,j)-phi_comp(i,j))^2;
     end
 end
 r=sum(rms(:));
 error(g)=(1/(N(g)*N(g)))*sqrt(r);
end
 figure;
 hold on;
 plot(log(dx),log(error));
 xlabel('log(dx)');
 ylabel('log(error)');
 title('Log-Log plot of Error Vs dx');
```

5.4 Appendix D

```
function TRI_2D_X(ibeg,iend,jbeg,jend,aa,bb,cc,dd)
for j=jbeg:jend
for i=ibeg:iend
    if ibeg==1
        cc(i,j)=cc(i,j)/bb(i,j);
        dd(i,j)=dd(i,j)/bb(i,j);
    else
        cc(i,j)=cc(i,j)/(bb(i,j)-(aa(i,j)*cc(i-1,j)))
        dd(i,j)=(dd(i,j)-aa(i,j)*dd(i-1,j))/(bb(i,j)-aa(i,j)*cc(i-1,j));

    end
end

for i=iend-1:-1:ibeg
    dd(i,j)=(dd(i,j)-cc(i,j)*dd(i+1,j));
end
end

return
end

function TRI_2D_Y(ibeg,iend,jbeg,jend,aa,bb,cc,dd)
for i=ibeg:iend
for j=jbeg:jend
    if jbeg==1
        cc(i,j)=cc(i,j)/bb(i,j);
        dd(i,j)=dd(i,j)/bb(i,j);
    else
        cc(i,j)=cc(i,j)/(bb(i,j)-(aa(i,j)*cc(i,j-1)));
        dd(i,j)=(dd(i,j)-aa(i,j)*dd(i,j-1))/(bb(i,j)-aa(i,j)*cc(i,j-1));

    end
end
for j=jend-1:-1:jbeg
    dd(i,j)=(dd(i,j)-cc(i,j)*dd(i,j+1));
end
end
return
end
```

6. References

1. Hoffmann, K.A. & Chiang S.T., 2000, Computational Fluid Dynamics Volume 1, Fourth Edition, Engineering Education Systems, Kansas, USA.

2.https://upload.wikimedia.org/wikipedia/commons/thumb/8/82/ADI-stencil.svg/2000px-ADI-stencil.svg.png

3.http://i.stack.imgur.com/khIK0.png