Project 3: Murman-Cole Scheme for the Transonic Small Disturbance Equation

Name – Swadesh Suman

McGill ID – 261097252

Solve the transonic small disturbance (TSD) theory over a circular arc airfoil at various Mach numbers using the Murman-Cole method.

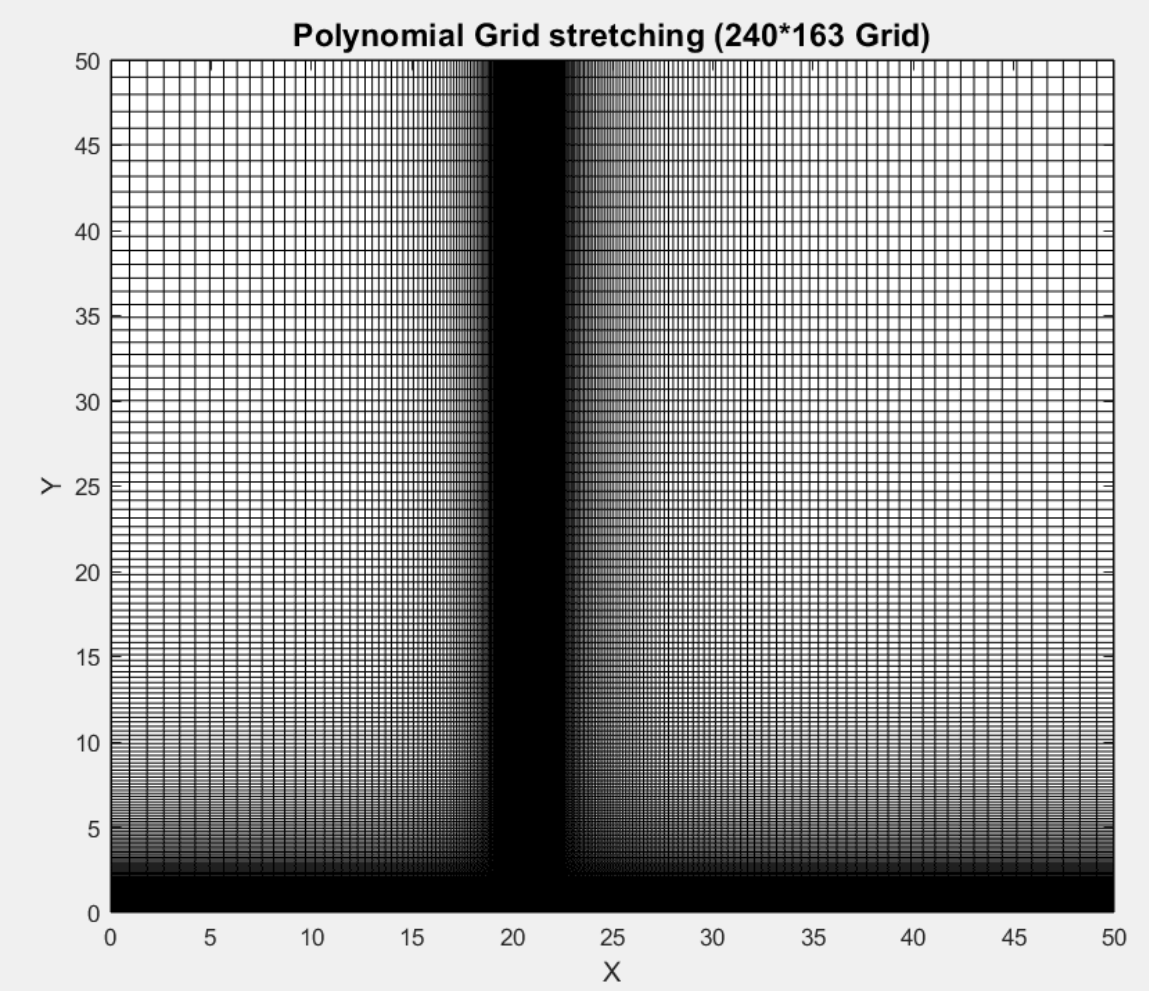

*Fig:1 – The Computational domain (50*50)*

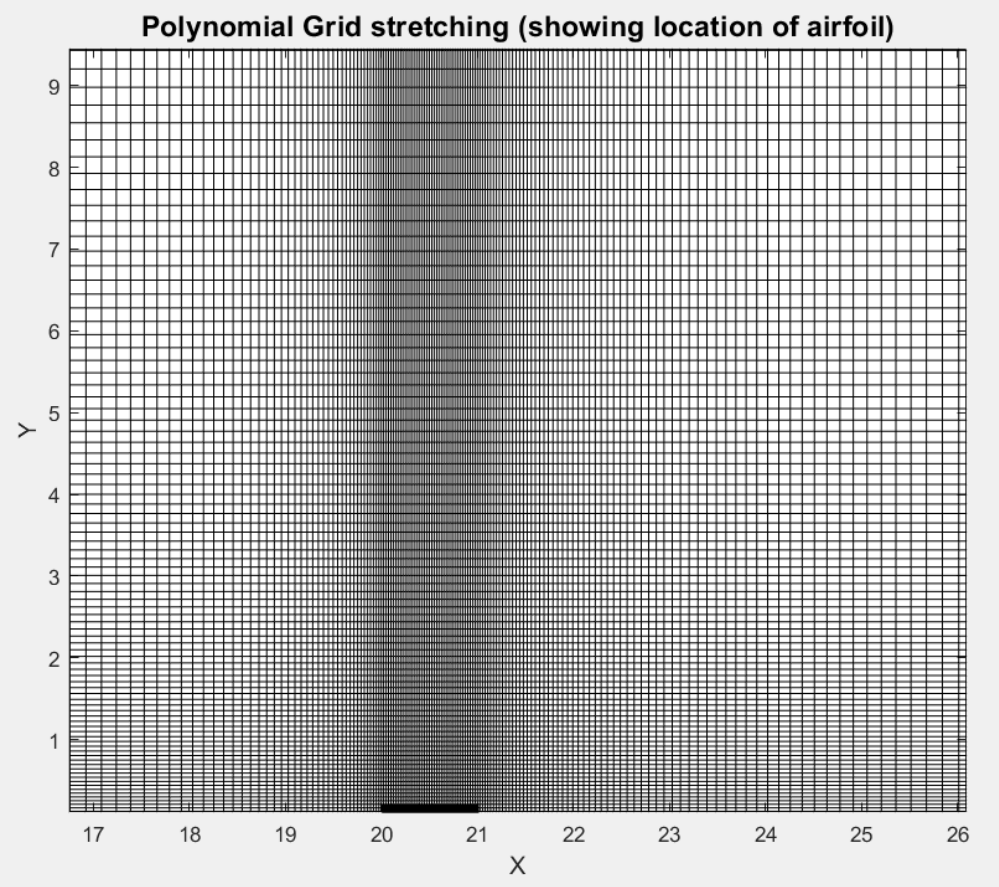

Fig:2 – The computational domain showing the location of the airfoil

Fig 1 shows the computational domain. Polynomial stretching is used to refine the mesh in the near-airfoil zones. Eq 1 shows the equation used to mesh the computational domain after the airfoil's trailing edge. Eq 2 shows the equation used to mesh the computational domain before the airfoil's leading edge. Eq 3 shows the equation used to mesh the computational domain in the y – direction.

$$x(i) = x(i-1) + \{x(i-1) - x(i-2)\} * 1.0286\text{^}1.05 \quad - \quad \text{eq 1}$$

$$x(i-1) = x(i) + \{x(i+1) - x(i)\} * 1.0444\text{^}1.05 \quad - \quad \text{eq 2}$$

$$y(i) = y(i-1) + \{y(i-1) - y(i-2)\} * 1.0186\text{^}1.1 \quad - \quad \text{eq 3}$$

Problem 2 – Provide a plot of the pressure coefficient along the airfoil surface with the negative pointing upwards. Vary the freestream Mach number between [0.80, 0.90] with 0.02 increments. For each case, provide a convergence plot of the L∞-norm, surface pressure coefficient as a function of x, and pressure contour for x ∈ [20, 21] and y ∈ [0, 1]. Discuss your findings. A four-order reduction in the residual is sufficient for the Gauss-Seidel method. Ensure that the residual reaches at least $1 * 10^{-4}$.

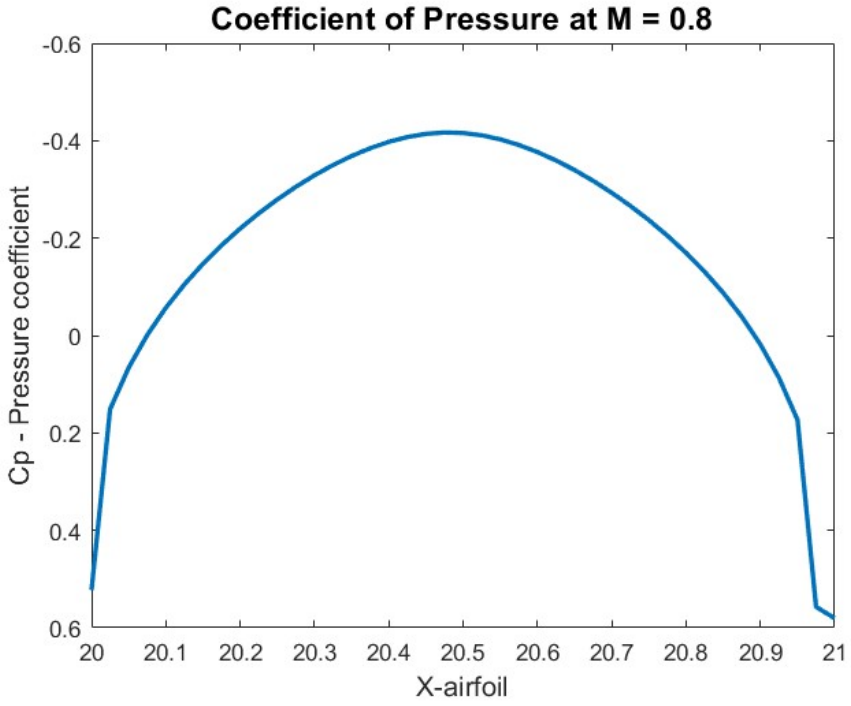

Fig:3 – Pressure Coefficient curve over the airfoil at M – 0.8

A mesh grid 240*163 is used to solve the TSD equation in the computational domain. Fig 3, 6, 9, 12, 15, & 18 show the Pressure Coefficient curve for Mach from [0.80,0.90] with 0.02 increments. Fig 4, 7, 13, 16, 19 shows the log10(error) vs iterations for Mach from [0.80 ,0.90] with 0.2 increments. Fig 5, 8, 14, 17, 20 shows the pressure contour for Mach from [0.80 ,0.90] with 0.2 increments. The shock appears on the airfoil at Mach 0.84 at 20.6 x-airfoil location and continues to shift towards the tailing edge of the airfoil with increasing Mach no. This can be seen clearly in the pressure coefficient and pressure contour plot. There is no shock presence over the airfoil for Mach 0.8 and 0.82. The total iterations takes to reach tolerance of E-4 also increases as the Mach no increases.

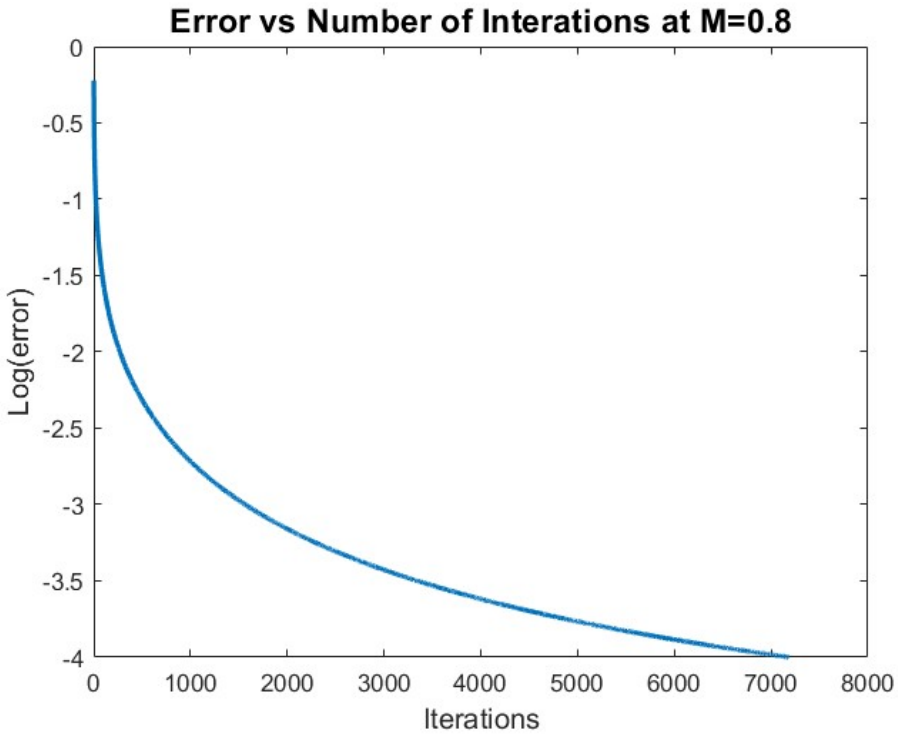

Fig:4 – Error vs Number of Iterations plot at M – 0.8

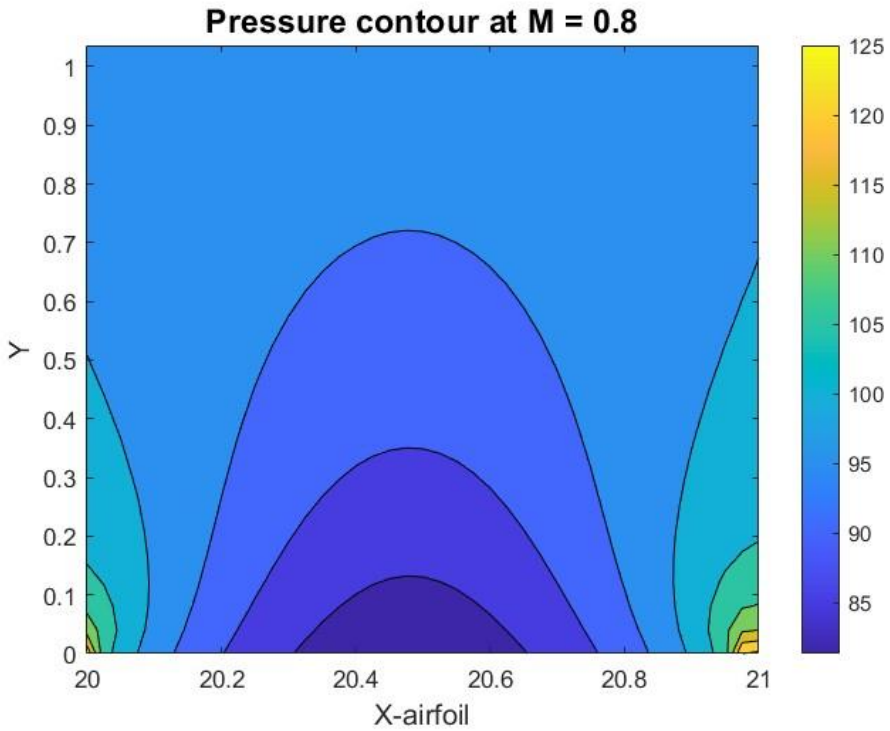

Fig:5 – Pressure Contour over an airfoil at M – 0.8

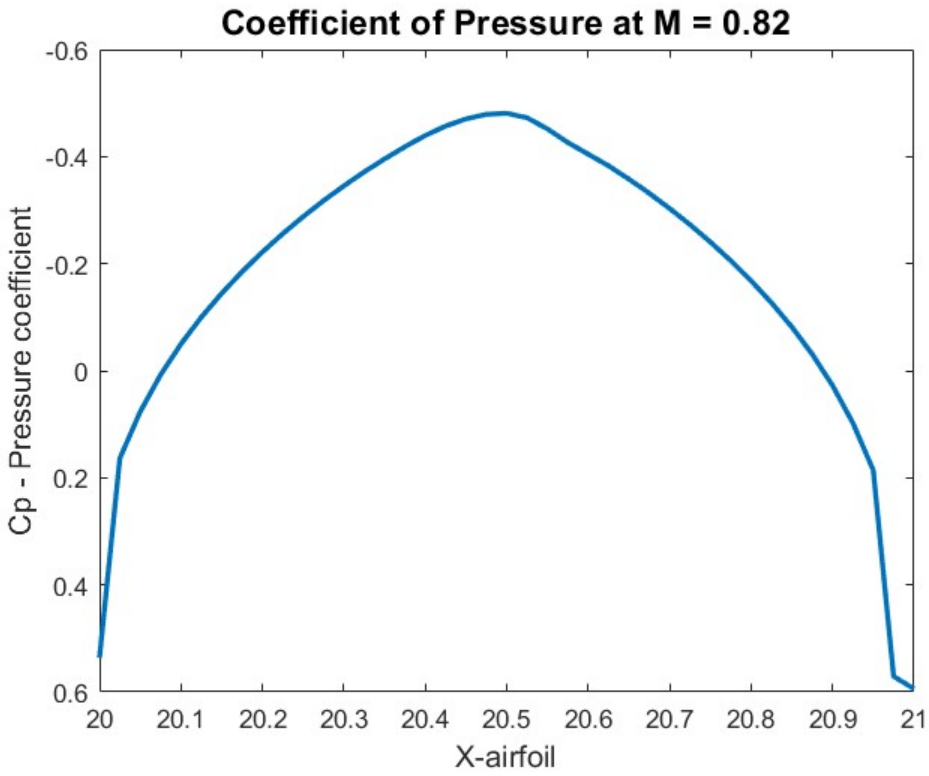

Fig:6 – Pressure Coefficient curve over the airfoil at M – 0.82

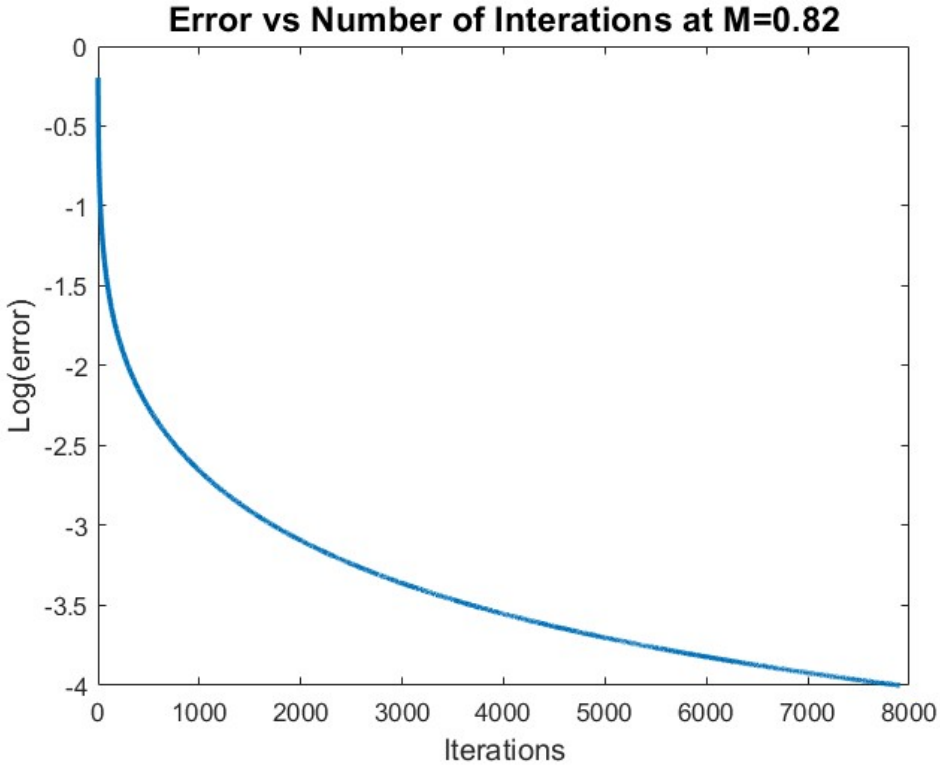

Fig:7 – Error vs Number of Iterations plot at M – 0.82

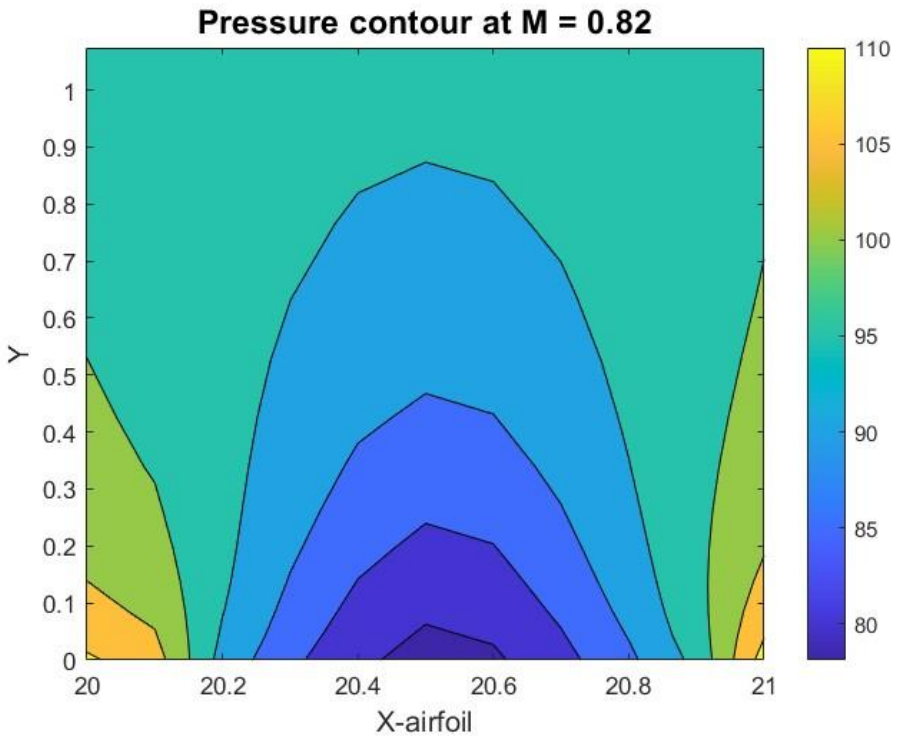

Fig:8 – Pressure Contour over an airfoil at M – 0.82

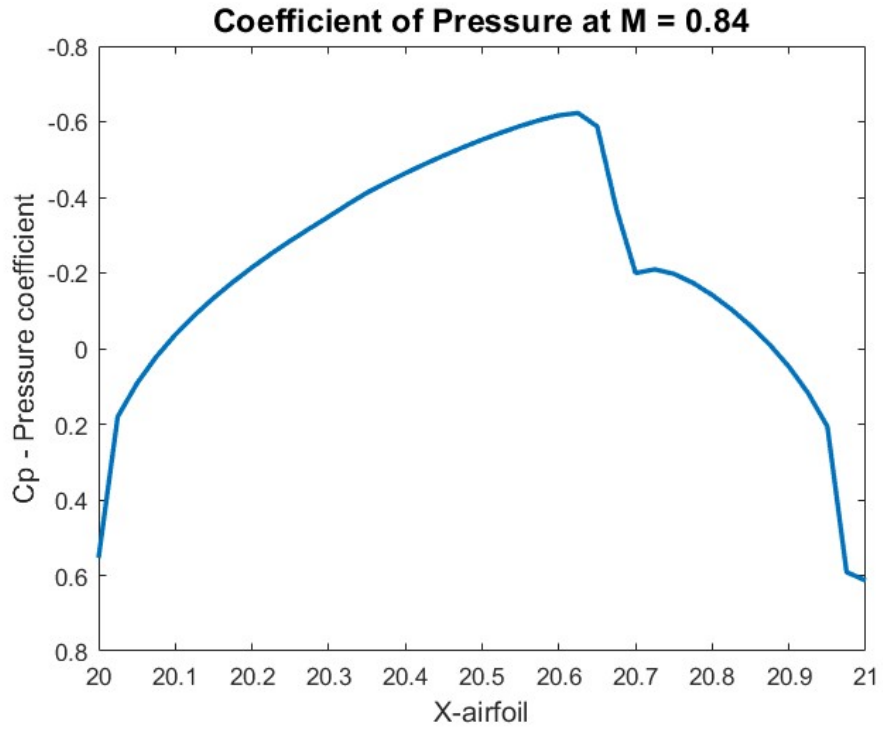

Fig:9 – Pressure Coefficient curve over the airfoil at M – 0.84

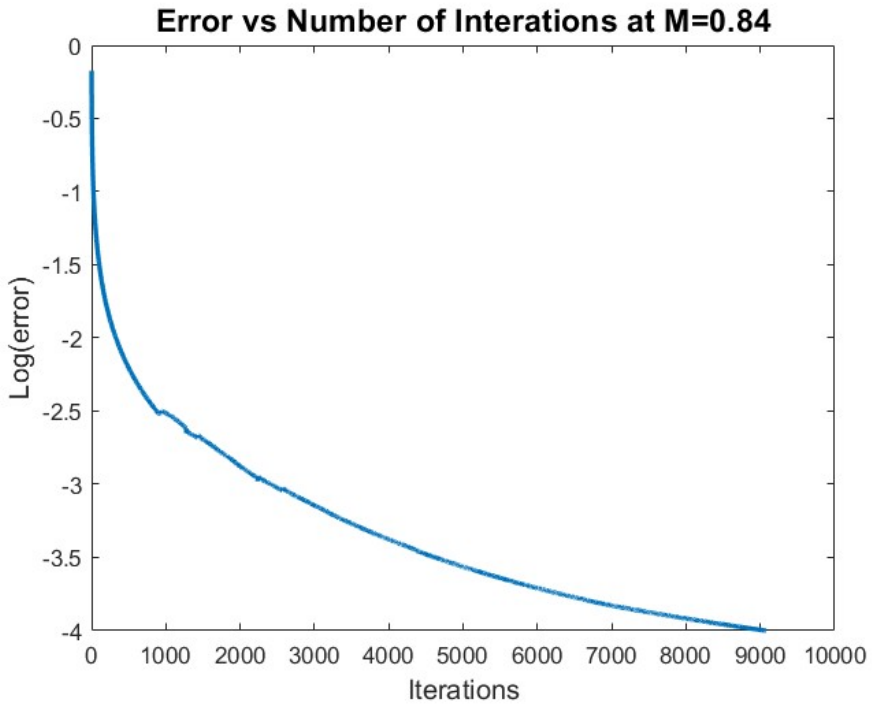

Fig:10 – Error vs Number of Iterations plot at M – 0.84

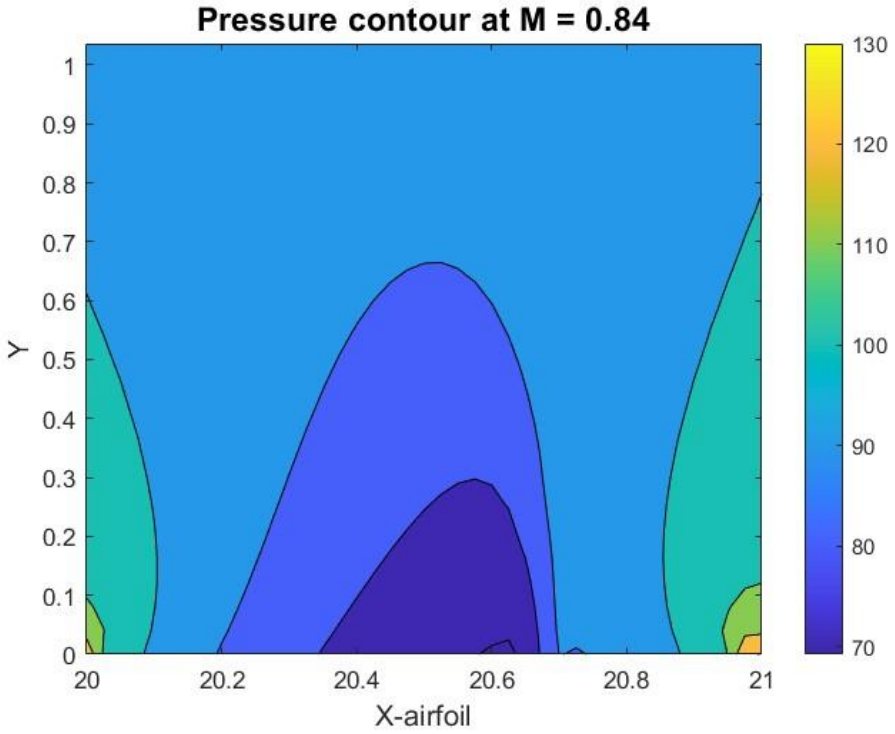

Fig:11 – Pressure Contour over an airfoil at M – 0.84

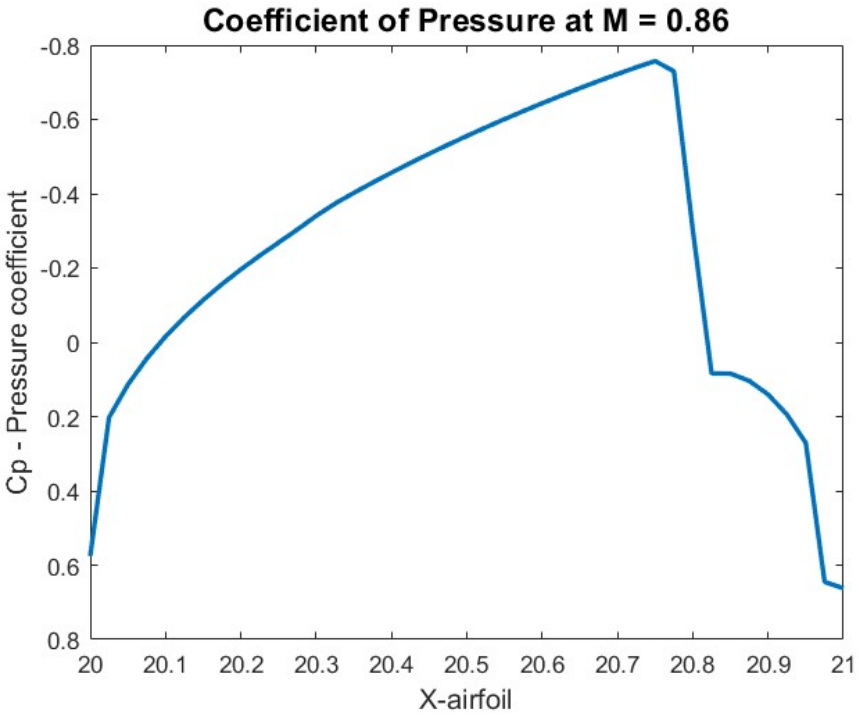

Fig:12 – Pressure Coefficient curve over the airfoil at M – 0.86

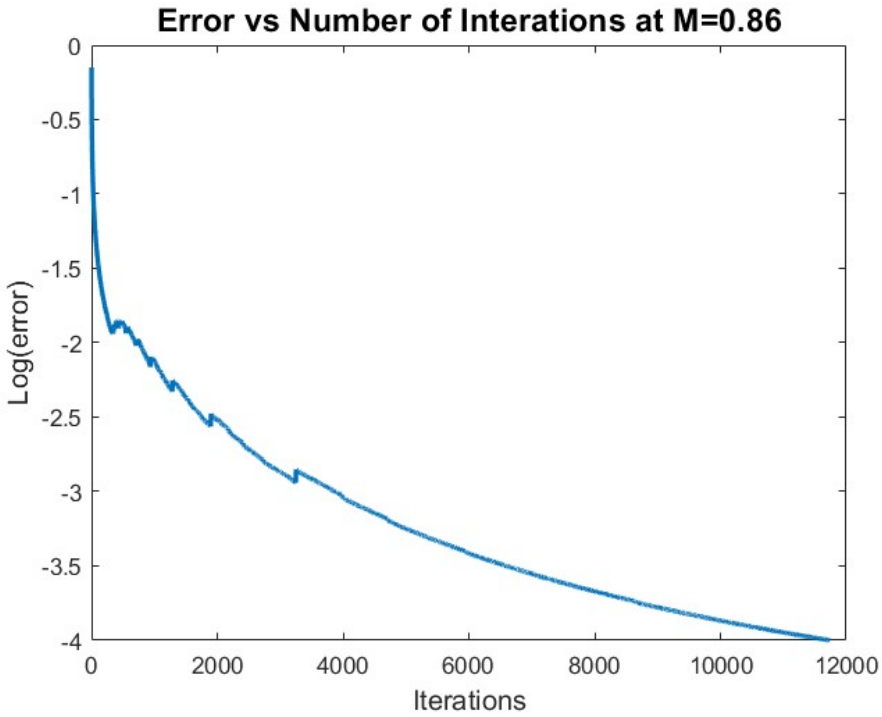

Fig:13 – Error vs Number of Iterations plot at M – 0.86

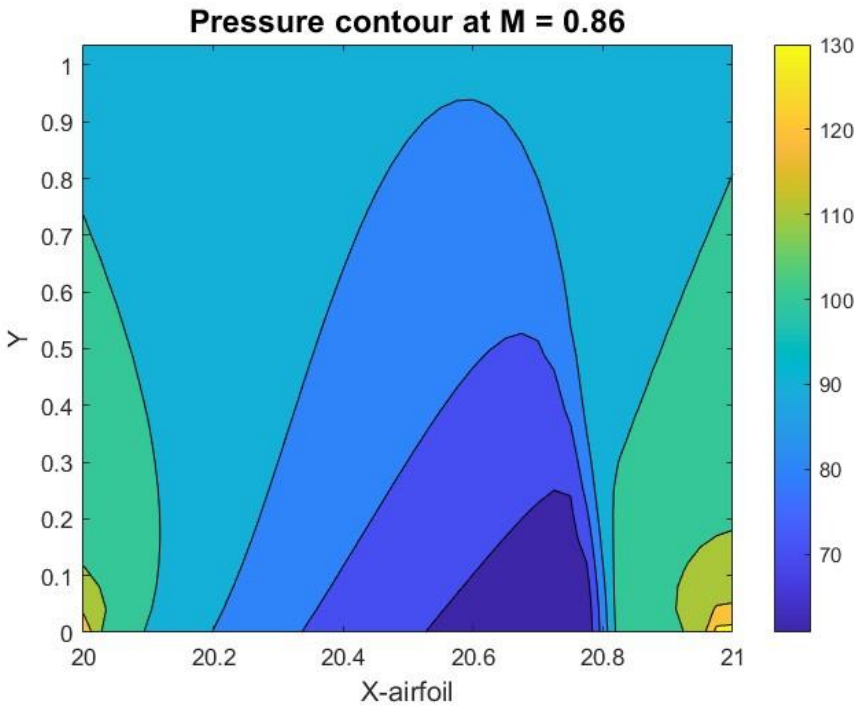

Fig:14 – Pressure Contour over an airfoil at M – 0.86

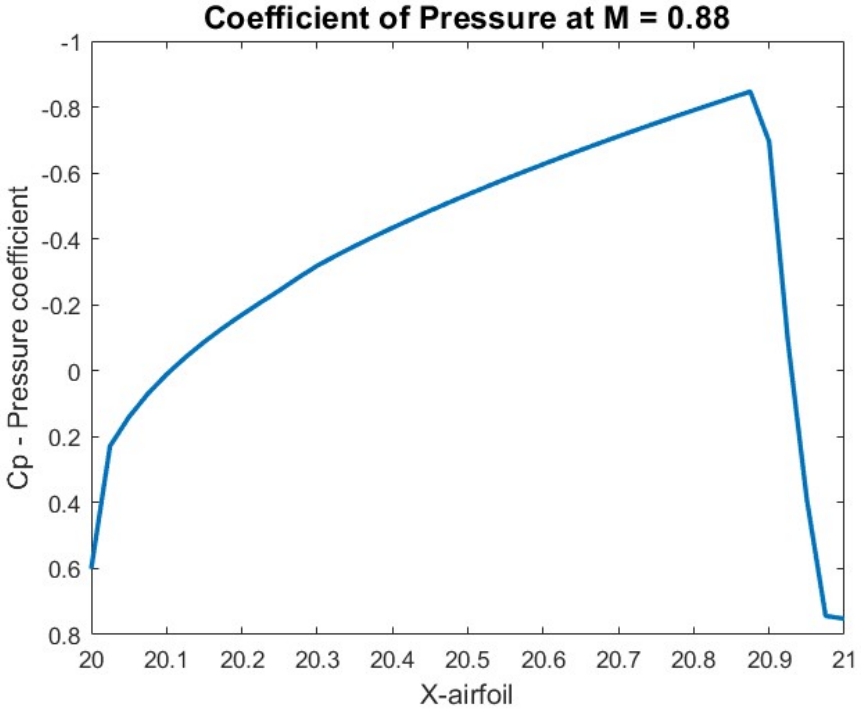

Fig:15 – Pressure Coefficient curve over the airfoil at M – 0.88

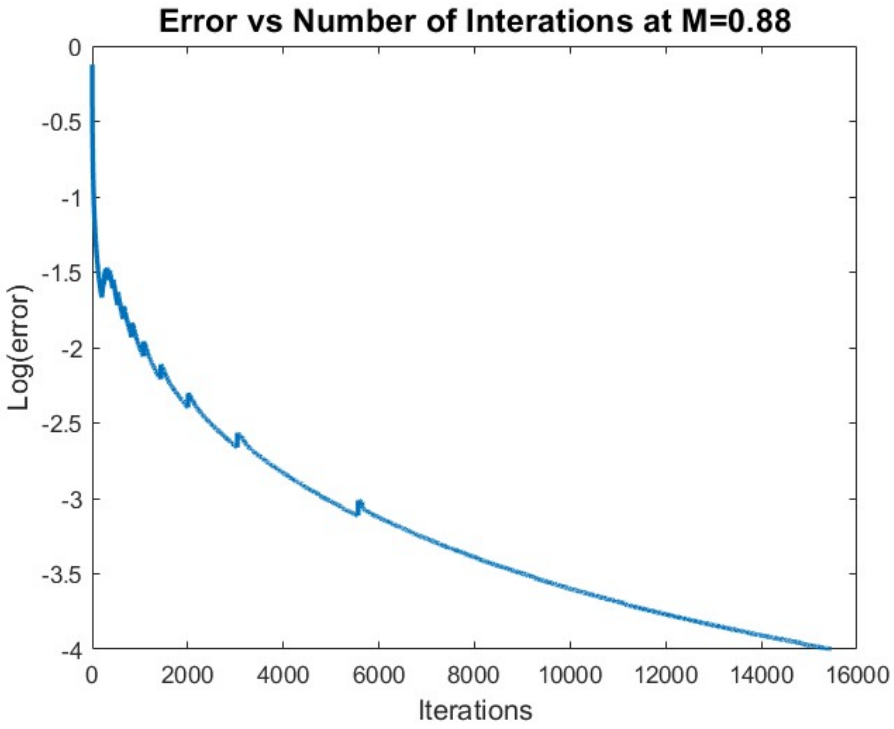

Fig:16 – Error vs Number of Iterations plot at M – 0.88

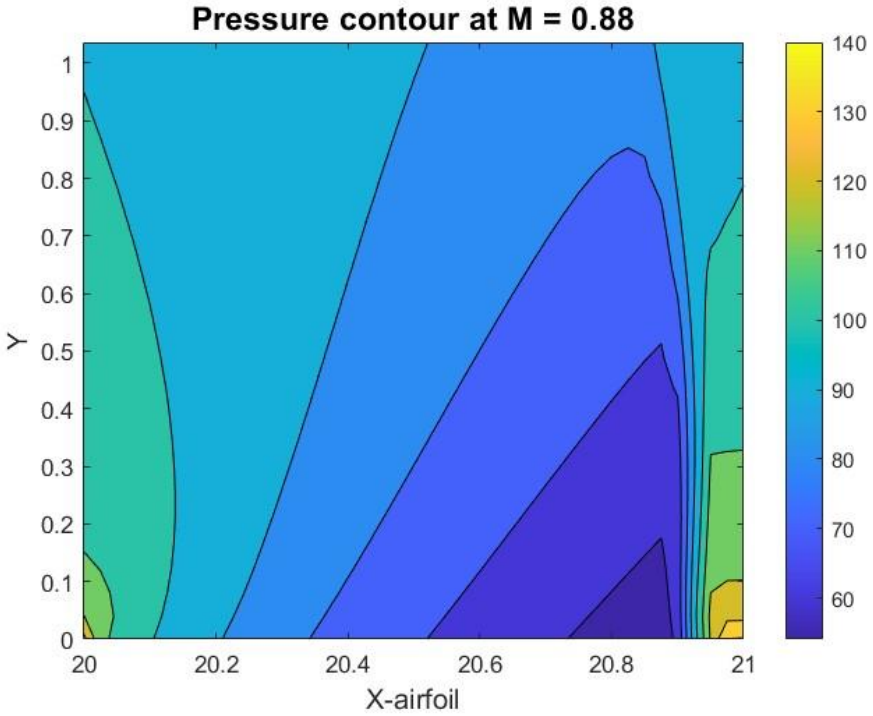

Fig:17 – Pressure Contour over an airfoil at M – 0.88

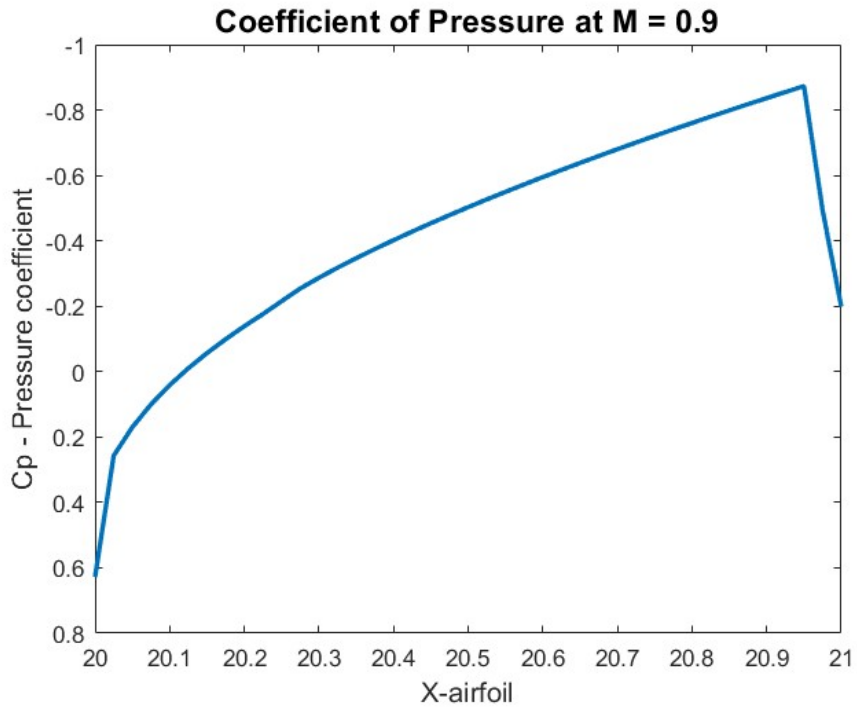

Fig:18– Pressure Coefficient curve over the airfoil at M – 0.9

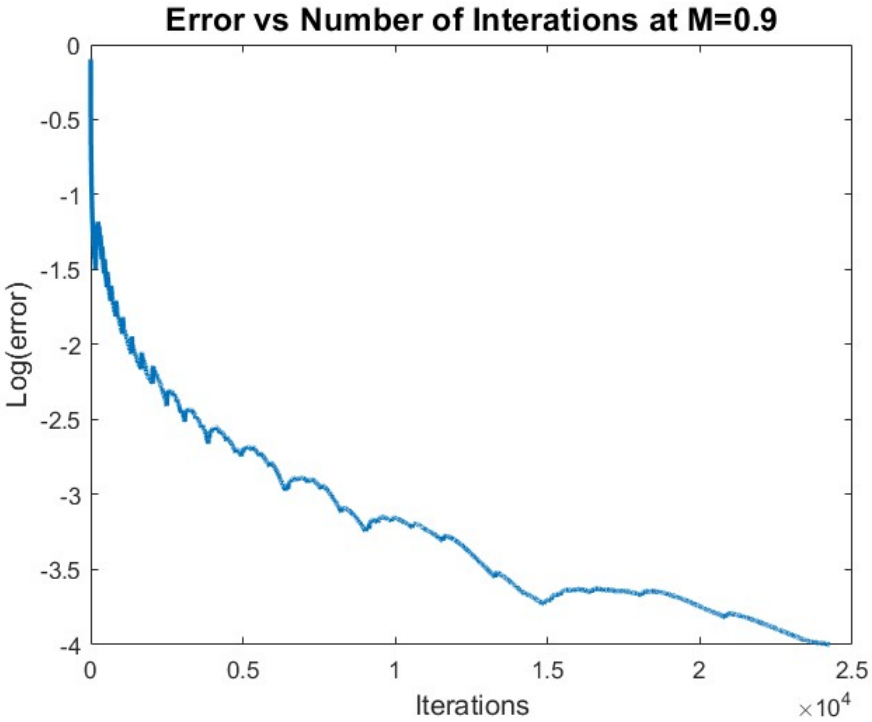

Fig:19 – Error vs Number of Iterations plot at M – 0.9

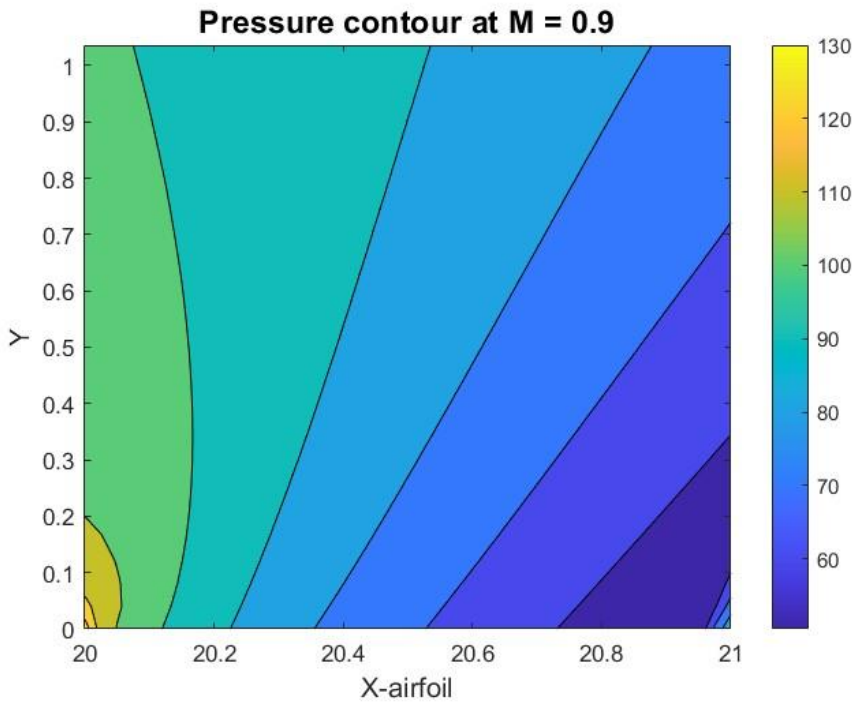

Fig:20 – Pressure Contour over an airfoil at M – 0.9

Problem 3 – Vary the grid size by doubling it in each direction as well as the number of points on the airfoil surface. Produce a coarse, medium, and fine grid. Plot the surface coefficient of pressure as a function of x on the airfoil surface for the three grids on the same plot at Mach 0.88. Discuss your findings. Does the shock location change.

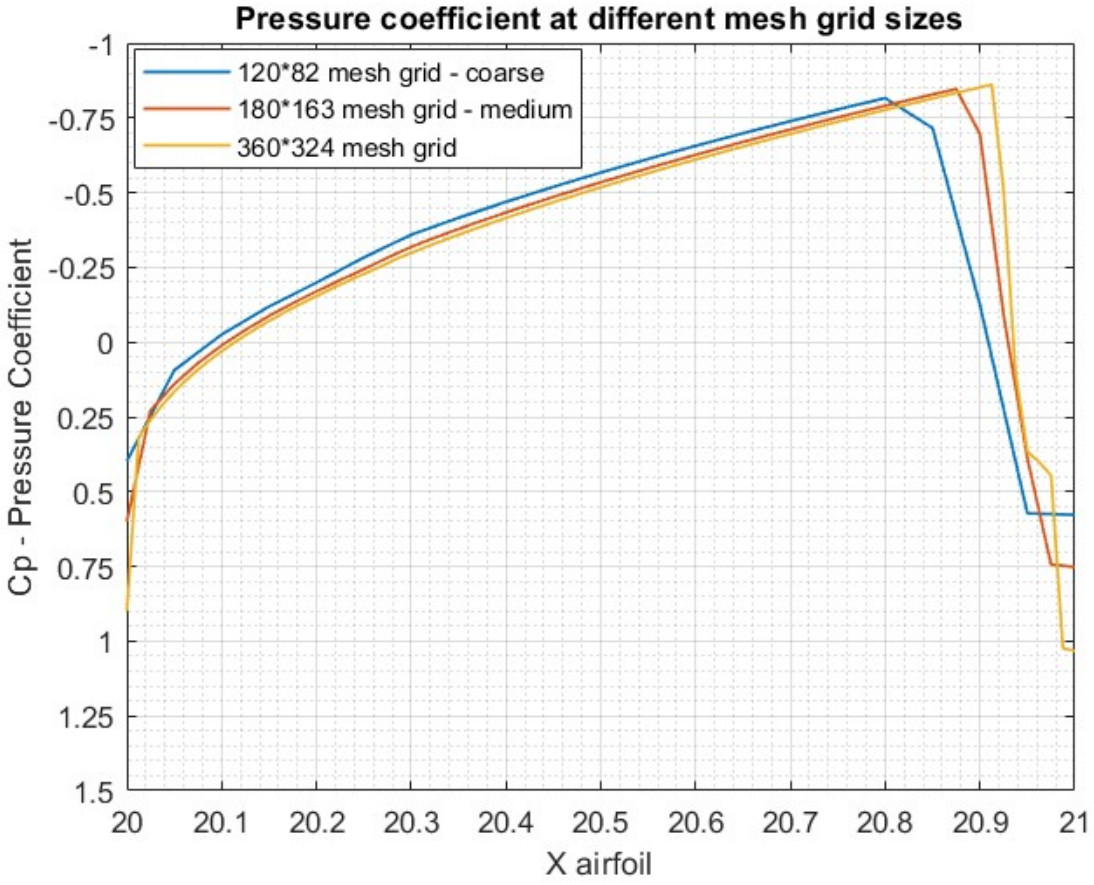

Fig:21 – Pressure Coefficient plot over the airfoil at M – 0.88 by varying computational grid sizes

- Fig 21 shows the Pressure Coefficient plot for three grids at M = 0.88. The overall profile of the Cp remains the same for all the three grids.
- The shock location for all the mesh grid does not change. It remains approximately at 20.9 (x airfoil position) for all the grids. However, we can see a slight shift in the Cp curve as the mesh is refined. It may be due to the grid refinement and the value of Cp shifting to the nearest x – position.

Problem 4-Now solves the equations using line implicit Gauss-Seidel at a Mach number of 0.86 and compares the convergence as a function of iterations and CPU time to that achieved by the standard Gauss-Seidel approach. Both approaches must reach machine accuracy.

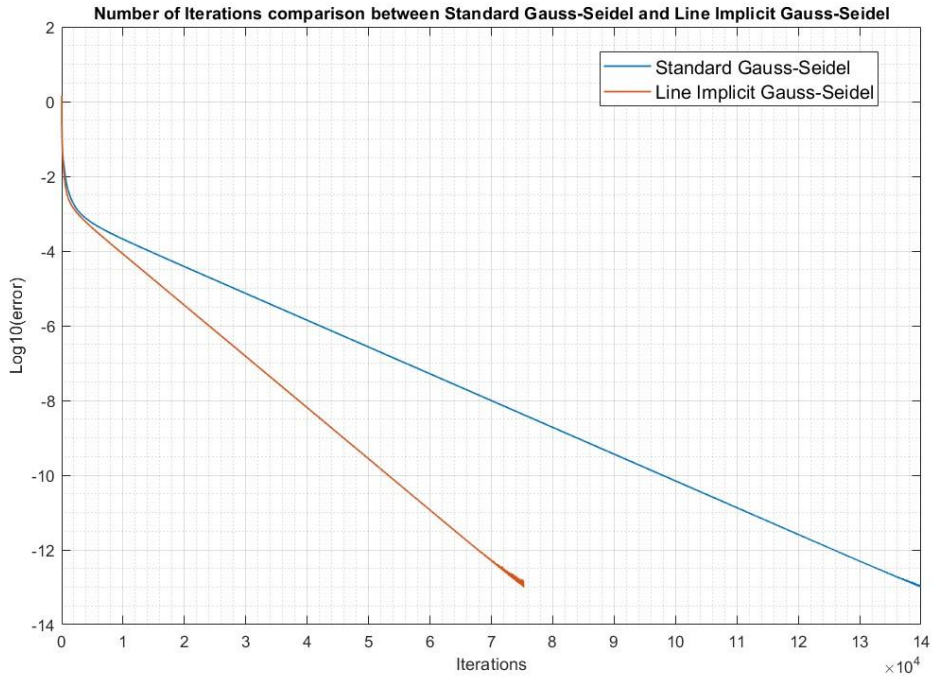

Fig:22 – Log(error) vs. Iterations plot at M – 0.86 for Standard GS and Line Implicit GS

- Fig 22 shows the Log10(error) vs. Iterations plot for Standard Gauss-Seidel (GS) and Line Implicit Gauss-Seidel. It can be seen from the plot that the standard GS takes more iterations to converge till 1*E-13 compared to Line Implicit GS.
- Since the problem needed to be solved till machine accuracy, the coarse grid of 60*41 is used instead of 240*163, as computationally the 240*163 grid is highly expensive.
- The final tolerance values are kept E-13 as the error stops dropping further and begins to stay constant.

- Fig 23 shows the Log10(error) vs CPU time plot for Standard Gauss-Seidel (GS) and Line Implicit Gauss-Seidel. The CPU time taken by the Line Implicit GS is slightly higher compared to Standard Gauss-Seidel.

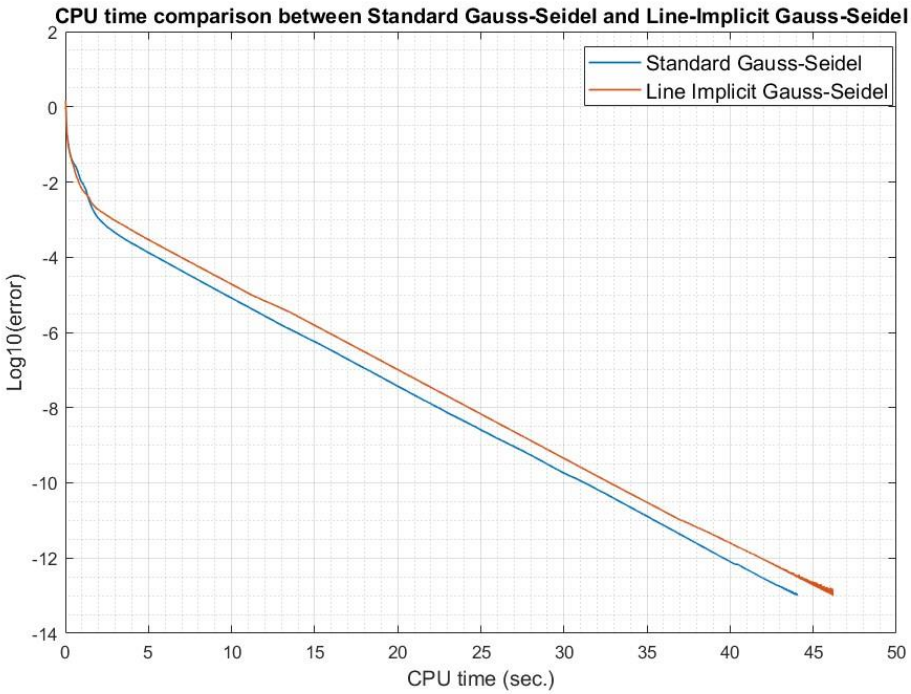

Fig:23 – Log(error) vs. CPU time plot at M – 0.86 for Standard GS and Line Implicit GS

Problem 5 - For a select grid size, plot the pressure coefficient along the airfoil surface for Mach number between [0.80, 0.90] with 0.02 increments on the same plot. Discuss your observations. What is the effect of increasing the Mach number?

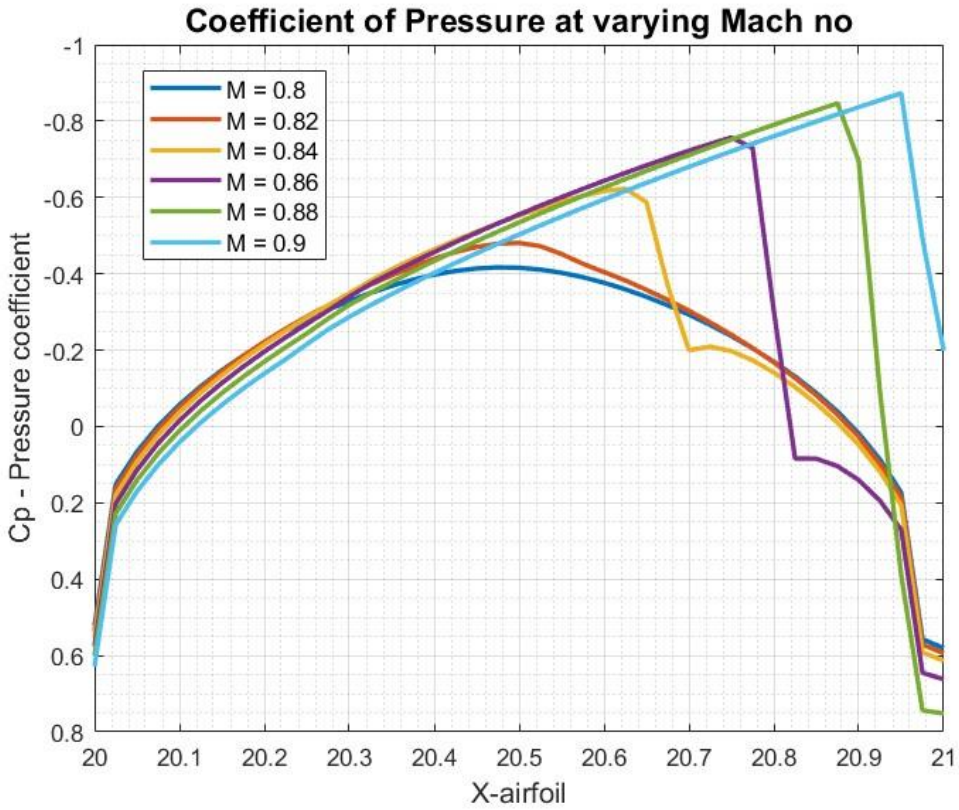

Fig:24 – Pressure Coefficient plot over airfoil at M = [0.80,0.90] with 0.02 increments

- Fig 24 shows the combined plot of pressure coefficient over the airfoil at varying Mach no.
- It can be seen from the plot that M = 0.8 and M = 0.82 produce no shock over the airfoil.
- The shock appears at M = 0.84 over the airfoil between 20.6 and 20.7. The shock location continuously changes over the airfoil as the Mach no increases. It shifts towards the trailing edge of the airfoil.

MATLAB CODE FOR STANDARD GAUSS SEIDEL

```
close
clear
clc

%% Defining the mesh
L = 50;
xf = 4; % factor by which mesh is refined compared to 10 grid points on airfoil
n1 = 80; % number of grid points before the leading edge
n2 = 40; % number of grid points for the airfoil
n3 = 120;  % number of grid points after the the trailing edge
le = n1; % leading edge x position
te = n1+n2; % trailing edge x position
tc = 0.08; % thickness ratio
nx = n1+n2+n3;
n4 = 163;
n = zeros(nx,1);
m = zeros(n4,1);
m(1) = 0;
m(2) = tc/2;
n(n1) = n1/xf;
dx = 1/n2;
for i = n1+1:n1+n2
    n(i) = n(i-1) + dx;
end
for i = n1+n2+1:nx
    n(i) = n(i-1) + (n(i-1)-n(i-2))*1.02858^1.05;
end
for i = n1:-1:2
    n(i-1) = n(i) - (n(i+1)-n(i))*1.04436^1.05;
end
% nt1 =zeros(nx,1);
% nt1(n1+n2+1:nx) = n(n1+n2+1:nx) - n1-1; nt2 = -flip(nt1);
% n(1:n1) = n1;
% n(1:n1) = n(1:n1)+(nt2(n1/2+1:n1+n2));
for i = 3:n4
    m(i) = m(i-1) + (m(i-1)-m(i-2))*1.01858^1.1;
end
% n = n/xf;

% n = n*L/n(nx,1);
n(1)=0;
m(end) = 50; n(end) = 50;
n_fine = n;
[x1,y1]= meshgrid(n,m);
x = x1'; y = y1';

%% Defining the boundary conditions and analysis parameters
mach = (0.8:0.02:0.9);
cp_store = zeros(length(mach),41);
for k = 1:length(mach)
yg = 1.4; % gamma for air
```

```
nx = n1+n2+n3;
ny = n4;
Ma = mach(k);   % Mach no of the flow
Ta = 293;   % Temp
pa = 100;   % Pressure
R = 287.058;  % gas constant
tc = 0.08;
p_a = 0;
p = zeros(nx,ny);

ca = sqrt(yg*R*Ta);
ua = ca*Ma; % calculation of u infinity
a = zeros(nx,ny);
b = zeros(nx,ny);
c = zeros(nx,ny);
d = zeros(nx,ny);
e = zeros(nx,ny);
g = zeros(nx,ny);

eps = 1E-4;
error = 1;   % initial error value
itr = 0;   % iteration
error_store = zeros(itr,1);   % storing the errors
itr_store = zeros(itr,1);    % storing the iteration
%% Boundary condition
for i = 2 : nx-1
    if i <= le
        p(i,2) = p(i,1);
    elseif i > te
        p(i,2) = p(i,1);
    else
         %x_pos = n1 + (i-le)/(te-le);
         x_pos = n(i);
         p(i,1)=p(2,1)-ua*pos(x_pos)*(m(2)-m(1));
    end
end

px = zeros(nx,ny);
for i = 2 : nx-1
    for j = 1: ny
        px(i,j) = (p(i+1,j)-p(i-1,j))/((n(i+1)-n(i-1)));
    end
end

A = zeros(nx,ny);
for i = 1 : nx
    for j = 1 : ny
         A(i,j) = (1-Ma^2)-((yg+1)*Ma^2/ua*px(i,j));
    end
end

q = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        if A(i,j)>0
```

```
            q(i,j) = 0;
        elseif A(i,j)<0
            q(i,j) = 1;
        end
    end
end

p_gauss = p;

while error>eps
    itr = itr+1;
    for i = 3:nx-1
        for j = 2:ny-1
            c(i,j) = 2/((m(j)-m(j-1))*(m(j+1)-m(j-1)));
            g(i,j) = 2*q(i-1,j)*A(i-1,j)/((n(i-1)-n(i-2))*(n(i)-n(i-2)));
            d(i,j) = (2*(1-q(i,j))*A(i,j))/((n(i)-n(i-1))*(n(i+1)-n(i-1)))+...
            (-2*q(i-1,j)*A(i-1,j))/((n(i)-n(i-1))*(n(i)-n(i-2)))+...
            ((-2*q(i-1,j)*A(i-1,j))/((n(i-1)-n(i-2))*(n(i)-n(i-2))));
            a(i,j) = (-2*(1-q(i,j))*A(i,j))/((n(i+1)-n(i))*(n(i+1)-n(i-1)))+...
            (-2*(1-q(i,j))*A(i,j))/((n(i)-n(i-1))*(n(i+1)-n(i-1)))+...
            2*q(i-1,j)*A(i-1,j)/((n(i)-n(i-1))*(n(i)-n(i-2)))+...
            (-2)/((m(j+1)-m(j))*(m(j+1)-m(j-1)))+...
            (-2)/((m(j)-m(j-1))*(m(j+1)-m(j-1)));
            e(i,j) = (2*(1-q(i,j))*A(i,j))/((n(i+1)-n(i))*(n(i+1)-n(i-1)));
            b(i,j) = 2/((m(j+1)-m(j))*(m(j+1)-m(j-1)));
            p_gauss(i,j) = (-c(i,j)*p_gauss(i,j-1)-d(i,j)*p_gauss(i-1,j)...
                -e(i,j)*p(i+1,j)-b(i,j)*p(i,j+1)-g(i,j)*p_gauss(i-2,j))...
                /a(i,j);
        end
    end
    error = max(abs(p_gauss-p),[],"all");
    error_store(itr,1) = error;
    itr_store(itr,1) = itr;
    fprintf('%d %d\n',itr,error);
    for i = 2 : nx-1
        if i <= le
            p_gauss(i,1) = p_gauss(i,2);
        elseif i > te
            p_gauss(i,1) = p_gauss(i,2);
        else
            %x_pos = n1 + (i-le)/(te-le);
            x_pos = n(i);
            p_gauss(i,1)=p_gauss(i,2)-ua*pos(x_pos)*(m(2)-m(1));
        end
    end
    p = p_gauss;
    % Boundary condition update
    for i = 2 : nx-1
        for j = 1: ny
            px(i,j)= (p_gauss(i+1,j)-p_gauss(i-1,j))/((n(i+1)-n(i-1)));
        end
    end
    for i = 1 : nx
        for j = 1 : ny
            A(i,j)= (1-Ma^2)-((yg+1)*Ma^2/ua*px(i,j));
```

```
        end
    end
    for i = 1:nx
        for j = 1:ny
            if A(i,j)>0
                q(i,j)= 0;
            elseif A(i,j)<0
                q(i,j)= 1;
            end
        end
    end

end
cp_fine = zeros(1,te-le);
pr = zeros(1,te-le);
x_airfoil_fi = zeros(1,te-le);
for i= 1 : te-le+1
    x_airfoil_fi(i) = n1/4 + (i-1)/(te-le);
    cp_fine(i) = -2*px(i+le,1)/ua;
end
cp_store(k,:)= cp_fine;
r = 1;
while m(r) <= 1
    r = r+1;
end

y_airfoil = m(1:r);
%pressure_contour = zeros(size(x_airfoil,2),size(y_airfoil,2));
pressure_contour = zeros(length(x_airfoil_fi),length(y_airfoil));

for i = 1 : length(x_airfoil_fi)
    for j = 1: length(y_airfoil)
        pressure_contour(i,j) = (1-(yg*Ma^2*px(i+le,j)/ua))*pa;
        %pressure_contour(i,j) = (-2*px(i+le,j)/ua);
    end
end
end
% for i= 1 : te-le+1
%     x_airfoil(i) = 20 + (i-1)/(te-le);
%     pr(i) = (1+ (yg*Ma^2*px(i+le,1)/ua))*pa;
% end
%%  Plotting the results
figure(1);
plot(x1,y1,'k-');
hold on;
plot(x,y,'k-')
% axis equal;
xlabel('X','FontSize',12);
ylabel('Y','FontSize',12);
title('Polynomial Grid stretching', 'FontSize',14);
figure(2);
plot(itr_store,log10(error_store),'LineWidth',2);
xlabel('Iterations','FontSize',12);
ylabel('Log(error)','FontSize',12);
title('Error vs Number of Interations at M=0.9', 'FontSize',14);
```

```
figure(3)
plot (x_airfoil_fi,cp_store(1,:),'LineWidth',2);
set(gca,'YDir','reverse');
hold on;
plot (x_airfoil_fi,cp_store(2,:),'LineWidth',2);
set(gca,'YDir','reverse');
hold on;
plot (x_airfoil_fi,cp_store(3,:),'LineWidth',2);
set(gca,'YDir','reverse');
hold on;
plot (x_airfoil_fi,cp_store(4,:),'LineWidth',2);
set(gca,'YDir','reverse');
hold on;
plot (x_airfoil_fi,cp_store(5,:),'LineWidth',2);
set(gca,'YDir','reverse');
hold on;
plot (x_airfoil_fi,cp_store(6,:),'LineWidth',2);
set(gca,'YDir','reverse');
hold on;
xlabel('X-airfoil','FontSize',12);
ylabel('Cp - Pressure coefficient','FontSize',12);
title('Coefficient of Pressure at varying mach no','FontSize',14);
legend('M = 0.8','M = 0.82','M = 0.84','M = 0.86','M = 0.88','M = 0.9',...
   'FontSize',10)

figure(4)
[x2,y2]= meshgrid(x_airfoil_fi,y_airfoil);
contourf(x2,y2,pressure_contour.');
xlabel('X-airfoil','FontSize',12);
ylabel('Y','FontSize',12);
zlabel('Pressure');
title ('Pressure contour at M = 0.9','FontSize',14)

function dydx = pos(x)
dydx = 0.08*((-4*x)+82);
end
```

MATLAB CODE FOR LINE IMPLICIT GAUSS SEIDEL

```
close
clear
clc

%% Defining the mesh
%Ma = 0.88;   % Mach no of the flow
L = 50;
xf = 2;% Length of domain
n1 = 20; % number of grid points before the leading edge
n2 = 10; % number of grid points for the airfoil
n3 = 30;  % number of grid points after the the trailing edge
```

```
le = n1;
te = n1+n2;
tc = 0.08;
nx = n1+n2+n3;
n4 = 41;
n = zeros(nx,1);
m = zeros(n4,1);
m(1) = 0;
m(2) = tc/2;
n(n1) = n1/2;
dx = 1/n2;
for i = n1+1:n1+n2
    n(i) = n(i-1) + dx;
end
for i = n1+n2+1:nx
    n(i) = n(i-1) + (n(i-1)-n(i-2))*1.11728^1.05;
end
for i = n1:-1:2
    n(i-1) = n(i) - (n(i+1)-n(i))*1.1856^1.1;
end
for i = 3:n4
    m(i) = m(i-1) + (m(i-1)-m(i-2))*1.12425^1.1;
end

n(1)=0;
m(end) = 50; n(end) = 50;
n_coarse = n;
[x1,y1]= meshgrid(n,m);
x = x1'; y = y1';


% Defining the parameters
yg = 1.4; % gamma for air
nx = n1+n2+n3;
ny = n4;
Ma = 0.86;   % Mach no of the flow
Ta = 293;
pa = 100;
R = 287.058;
tc = 0.08;
p_a = 0;
p = zeros(nx,ny);

ca = sqrt(yg*R*Ta);
ua = ca*Ma; % calculation of u infinity
a = zeros(nx,ny);
b = zeros(nx,ny);
c = zeros(nx,ny);
d = zeros(nx,ny);
e = zeros(nx,ny);
g = zeros(nx,ny);

eps = 1E-13;
error = 1;   % initial error value
itr = 0;   % iteration
```

```
cpu_limp = zeros(itr,1);
error_limp_store = zeros(itr,1);   % storing the errors
itr_limp_store = zeros(itr,1);    % storing the iteration
%% Boundary condition
for i = 2 : nx-1
    if i <= le
        p(i,2) = p(i,1);
    elseif i > te
        p(i,2) = p(i,1);
    else
         x_pos = n1 + (i-le)/(te-le);
         %x_pos = n(i);
         p(i,1)=p(2,1)-ua*pos(x_pos)*(m(2)-m(1));
    end
end

px = zeros(nx,ny);
for i = 2 : nx-1
    for j = 1: ny
        px(i,j) = (p(i+1,j)-p(i-1,j))/((n(i+1)-n(i-1)));
    end
end

A = zeros(nx,ny);
for i = 1 : nx
    for j = 1 : ny
         A(i,j) = (1-Ma^2)-((yg+1)*Ma^2/ua*px(i,j));
    end
end

q = zeros(nx,ny);
for i = 1:nx
    for j = 1:ny
        if A(i,j)>0
            q(i,j) = 0;
        elseif A(i,j)<0
            q(i,j) = 1;
        end
    end
end
%% Analysis steps
p_gauss = p;
p_limp = zeros(ny-2,ny-2);
kn = zeros(ny-2,1);
tic
while error>eps
    itr = itr+1;
    for i = 3:nx-1
        for j = 2:ny-1
            c(i,j) = 2/((m(j)-m(j-1))*(m(j+1)-m(j-1)));
            g(i,j) = 2*q(i-1,j)*A(i-1,j)/((n(i-1)-n(i-2))*(n(i)-n(i-2)));
            d(i,j) = (2*(1-q(i,j))*A(i,j))/((n(i)-n(i-1))*(n(i+1)-n(i-1)))+...
            (-2*q(i-1,j)*A(i-1,j))/((n(i)-n(i-1))*(n(i)-n(i-2)))+...
            ((-2*q(i-1,j)*A(i-1,j))/((n(i-1)-n(i-2))*(n(i)-n(i-2))));
            a(i,j) = (-2*(1-q(i,j))*A(i,j))/((n(i+1)-n(i))*(n(i+1)-n(i-1)))+...
```

```
            (-2*(1-q(i,j))*A(i,j))/((n(i)-n(i-1))*(n(i+1)-n(i-1)))+...
            2*q(i-1,j)*A(i-1,j)/((n(i)-n(i-1))*(n(i)-n(i-2)))+...
            (-2)/((m(j+1)-m(j))*(m(j+1)-m(j-1)))+...
            (-2)/((m(j)-m(j-1))*(m(j+1)-m(j-1)));
            e(i,j) = (2*(1-q(i,j))*A(i,j))/((n(i+1)-n(i))*(n(i+1)-n(i-1)));
            b(i,j) = 2/((m(j+1)-m(j))*(m(j+1)-m(j-1)));
            if j == 2
                p_limp(j-1,j-1) = a(i,j);
                p_limp(j-1,j) =  b(i,j);
                kn(j-1) = (-c(i,j)*p(i,j-1)-d(i,j)*p(i-1,j)-e(i,j)*p(i+1,j)...
                    -g(i,j)*p(i-2,j));
            elseif j == ny-1
                p_limp(j-1,j-1) = a(i,j);
                p_limp(j-1,j-2) =  c(i,j);
                kn(j-1) = (-b(i,j)*p(i,j+1)-d(i,j)*p(i-1,j)-e(i,j)*p(i+1,j)...
                    -g(i,j)*p(i-2,j));
            else
                p_limp(j-1,j-1) = a(i,j);
                p_limp(j-1,j) =  b(i,j);
                p_limp(j-1,j-2) =  c(i,j);
                kn(j-1) = (-d(i,j)*p(i-1,j)-e(i,j)*p(i+1,j)-g(i,j)*p(i-2,j));
            end
        end
%         p_limp_inv= inv(p_limp);
        p_inv = p_limp\kn;
        for j = 2:ny-1
            p_gauss(i,j) = p_inv(j-1);
        end

    end
    error = max(abs(p_gauss-p),[],"all");
    error_limp_store(itr,1) = error;
    itr_limp_store(itr,1) = itr;
    cpu_limp(itr,1) = toc;
    fprintf('%d %d\n',itr,error);
    for i = 2 : nx-1
        if i <= le
            p_gauss(i,1) = p_gauss(i,2);
        elseif i > te
            p_gauss(i,1) = p_gauss(i,2);
        else
            x_pos = n1 + (i-le)/(te-le);
            %x_pos = n(i);
            p_gauss(i,1)=p_gauss(i,2)-ua*pos(x_pos)*(m(2)-m(1));
        end
    end
    p = p_gauss;
    % Boundary condition update
    for i = 2 : nx-1
        for j = 1: ny
            px(i,j)= (p_gauss(i+1,j)-p_gauss(i-1,j))/((n(i+1)-n(i-1)));
        end
    end
    for i = 1 : nx
        for j = 1 : ny
```

```
            A(i,j)= (1-Ma^2)-((yg+1)*Ma^2/ua*px(i,j));
        end
    end
    for i = 1:nx
        for j = 1:ny
            if A(i,j)>0
                q(i,j)= 0;
            elseif A(i,j)<0
                q(i,j)= 1;
            end
        end
    end
end
cp_limp = zeros(1,te-le);
pr = zeros(1,te-le);
x_airfoil_co= zeros(1,te-le);
for i= 1 : te-le+1
    x_airfoil_co(i) = n1 + (i-1)/(te-le);
    cp_limp(i) = -2*px(i+le,1)/ua;
end

%% Plotting the results
figure(1);
plot(itr_limp_store,log10(error_limp_store),'LineWidth',2);
xlabel('Iterations','FontSize',12);
ylabel('Log(error)','FontSize',12);
title('Error vs Number of Interations', 'FontSize',14);

figure(2);
plot(cpu_limp,log10(error_limp_store),'LineWidth',2);
xlabel('CPU time','FontSize',12);
ylabel('Log(error)','FontSize',12);
title('Error vs CPU time (sec.)', 'FontSize',14);


function dydx = pos(x)
dydx = 0.08*((-4*x)+82);
end
```