MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS Institution – UGC Govt of India)
Permanently Affiliated to JNTUH, ACCREDITED by AICTE-NBA, NAAC
Maisammaguda, Dhulapally post, Secunderabad – 500014 (A. P.)

CERTIFICATE

Department of Aeronautical Engineering certified that this is a bonafide record of work done by ________________________________ Roll .No: ______________ of M.Tech (AEROSPACE ENGINEERING) First Year ______ Semester in the __________________________- Laboratory during the year __________.

Staff Incharge HOD/ANE

Internal Examiner External Examiner

INDEX

Experiment -1

INTRODUCTION TO MATLAB SOFTWARE

Aim:

To study the introduction of MATLAB software.

Equipment needed:

Core 2 duo processor with 1GB RAM and MATLAB software.

Brief introduction about MATLAB and algorithm:-

You should experiment as you proceed, keeping the following points in mind. Upper and lower case characters are not equivalent (MATLAB is case sensitive). Typing the name of a variable will cause MATLAB to display its current value. A semicolon at the end of a command suppresses the screen output.

MATLAB uses both parentheses, (), and square brackets, [], and these are not interchangeable. The up arrow and down arrow keys can be used to scroll through your previous commands. Also, an old command can be recalled by typing the first few characters followed by up arrow. You can type help topic to access online help on the command, function or symbol topic.

You can quit MATLAB by typing exit or quit. Having entered MATLAB, you should work through this tutorial by typing in the text that appears after the MATLAB prompt, >>, in the Command Window. After showing you what to type, we display the output that is produced. We begin with

```
>> a = [1 2 3]

    a =

    1 2 3
```

This means that you are to type \a = [1 2 3]", after which you will see MATLAB's output \a =" and \1 2 3" on separate lines separated by a blank line.

In the next example, semicolons separate the entries:

```
    >> c = [4; 5; 6]

    c =

    4

    5

    6
```

A semicolon tells MATLAB to start a new row, so c is 3-by-1 (a column vector). Now

you can multiply the arrays a and c:

```
>>a*c

ans =

32
```

Here, you performed an inner product: a 1-by-3 array multiplied into a 3-by-1 array. MATLAB automatically assigned the result to the variable ans, which is short for answer. An alternative way to compute an inner product is with the dot function:

```
>>dot(a,c)

ans =

32
```

Inputs to MATLAB functions are specified after the function name and within parentheses.

You may also form the outer product:

```
>> A = c*a

A =

4 8 12

5 10 15

6 12 18
```

Here, the answer is a 3-by-3 matrix that has been assigned to A.

The product a*a is not defined, since the dimensions are incompatible for matrix multiplication:

```
>>a*a

??? Error using ==> *

Inner matrix dimensions must agree.
```

Arithmetic operations on matrices and vectors come in two distinct forms. Matrix sense operations are based on the normal rules of linear algebra and are obtained with the usual symbols +, -, *, / and ^. Array sense operations are defined to act elementwise and are generally obtained by preceding the symbol with a dot. Thus if you want to square each element of a you can write

```
>> b = a.^2
    b =
    1 4 9
```

Since the new vector b is 1-by-3, like a, you can form the array product of it with a:

```
>> a.*b
ans =
1 8 27
```

Result:-Basics of MATLAB software

Experiment -2

SCALAR, VECTOR, MATRIX FUNCTIONS AND OPERATING WITH MATRICES

Aim:

To study the programs related to scalar, vector, matrix functions and operating with matrices.

Equipment needed:

Core 2 duo processor with 1GB RAM and MATLAB software.

Steps and algorithms:-

You may set up a two-dimensional array by using spaces to separate entries within a row and semicolons to separate rows:

```
>> B = [-3 0 1; 2 5 -7; -1 4 8]

B =

-3 0 1

2 5 -7

-1 4 8
```

The eigenvalues of B can be found using eig:

```
>> e = eig(B)

e =

-2.8601

6.4300 + 5.0434i

6.4300 - 5.0434i
```

Here, i is the imaginary unit, p

1. You may also specify two output arguments for

the function eig:

```
>> [V,D] = eig(B)

V =

0.9823 -0.0400 - 0.0404i -0.0400 + 0.0404i

-0.1275 0.7922 0.7922

0.1374 -0.1733 - 0.5823i -0.1733 + 0.5823i

D =

-2.8601 0 0

0 6.4300 + 5.0434i 0

0 0 6.4300 - 5.0434i
```

In this case the columns of V are eigenvectors of B and the diagonal elements of D are the corresponding eigenvalues.

Result:-

Basics of scalar, vector and matrix operation

Experiment -3

PROGRAMS FOR GRAPHICS -2D & 3D PLOTS

Aim:

To study the programs for 2D & 3D PLOTS

Equipment needed:

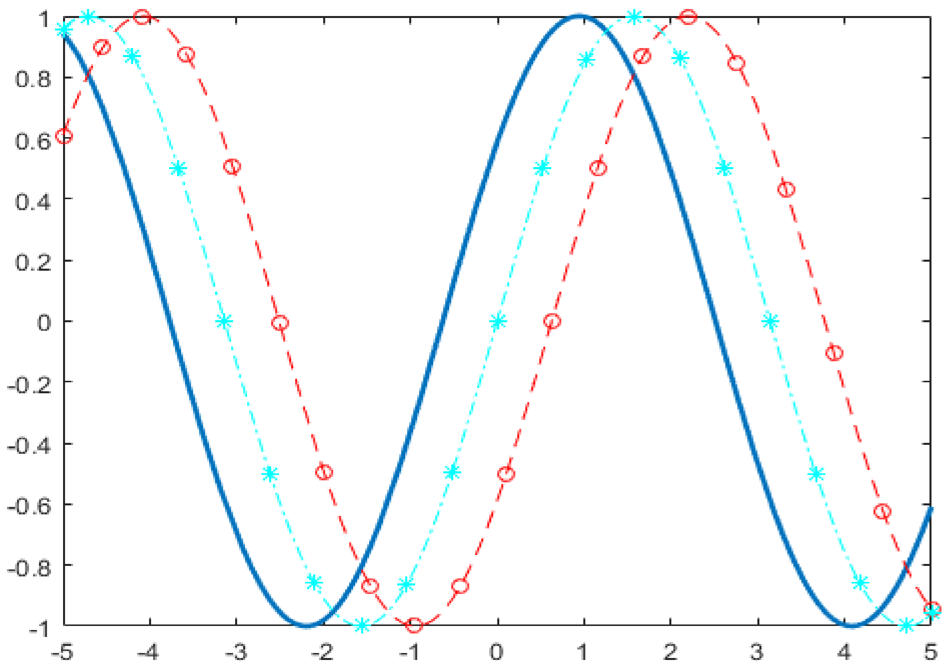
Core 2 duo processor with 1GB RAM and MATLAB software.

Steps and algorithms:-

Plot three sine waves with different phases. For the first, use a line width of 2 points. For the second, specify a dashed red line style with circle markers. For the third, specify a cyan, dash-dotted line style with asterisk markers.

Syntex and Description:

```
fplot(@(x) sin(x+pi/5),'Linewidth',2);
holdon
fplot(@(x) sin(x-pi/5),'--or');
fplot(@(x) sin(x),'-.*c')
holdoff
```



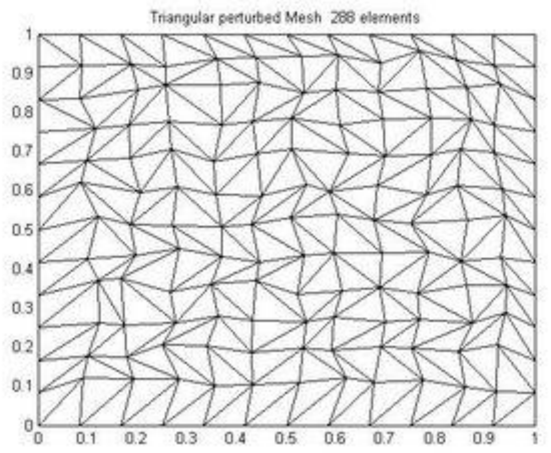
Result:-

Graph 2D and 3D plots

<u>EXPERIMENT 4:</u> Write a program to generate unstructured grid using Delaunay Triangulation method.

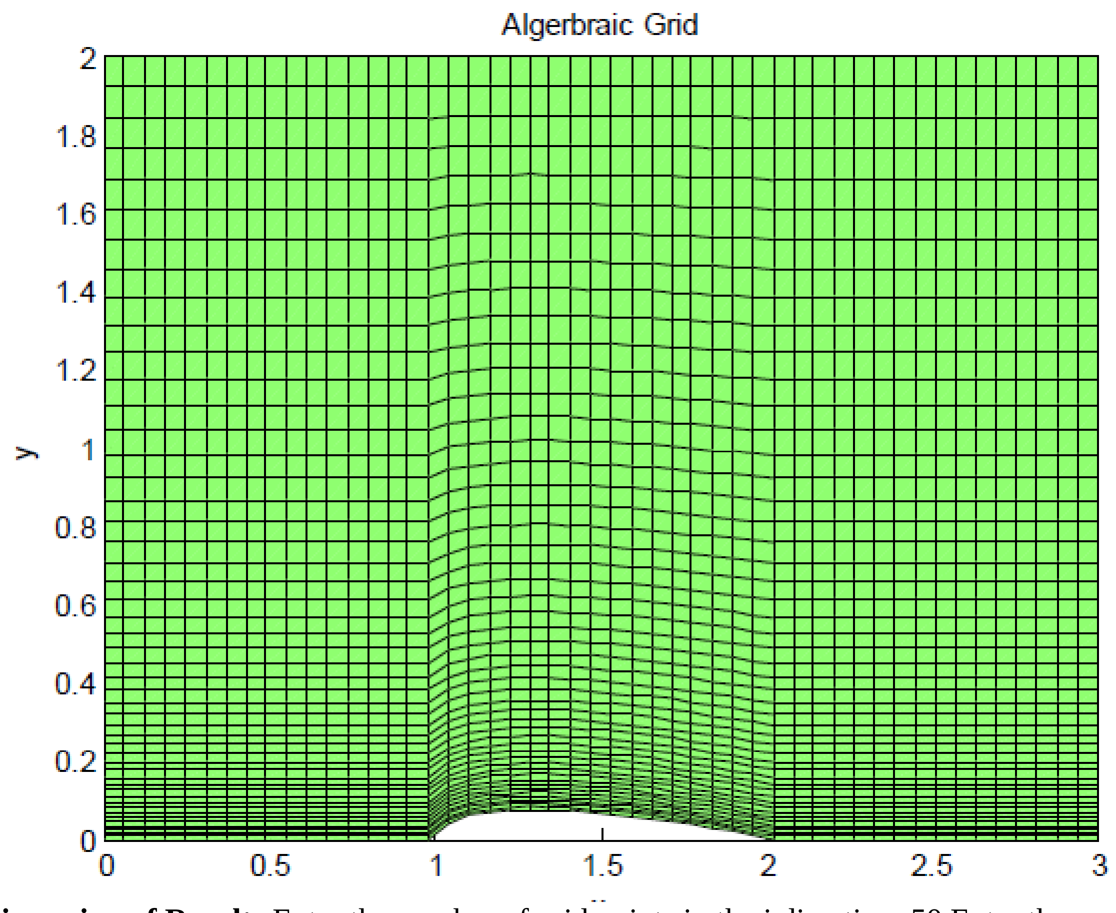
```
function tri = triangulate(xnod,ynod,nodes)
% tri = triangulate(xnod,ynod,nodes)
% triangulate the quadrilateral mesh

nele = size(nodes,1);
tri = zeros(3,2*nele)';
iv = [];
ii1 = [2 3 1];
jj1 = [4 1 3];
ii2 = [1 2 4];
jj2 = [2 3 4];
nrtri = 0;
for iel = 1:nele
iv = nodes(iel,:);
d1 = norm([xnod(iv(1))-xnod(iv(3));ynod(iv(1))-ynod(iv(3))]);
d2 = norm([xnod(iv(2))-xnod(iv(4));ynod(iv(2))-ynod(iv(4))]);
if d1 <= d2
nrtri = nrtri+1;
tri(nrtri,:) = iv(ii1);
nrtri = nrtri+1;
tri(nrtri,:) = iv(jj1);
else
nrtri = nrtri+1;
tri(nrtri,:) = iv(ii2);
nrtri = nrtri+1;
tri(nrtri,:) = iv(jj2);
end
end

x=[0:1/(numx):1];

y=[0:1/(numy):1]; %Matlab's meshgrid is used to create 2D grid from specified divisons above
[X,Y] = meshgrid(x,y);


X1=reshape(X',length(x)*length(y),1);

Y1=reshape(Y',length(x)*length(y),1); %Coordinates of the node
```

```
node=[X1 Y1]; % Node
tri = triangulate(X1,Y1,element); % element connectivity

nele = size(tri,1);

Z= zeros(length(y)-2,length(x)-2);

nn = tri; % nodes in a long list

xx = X1(nn); yy = Y1(nn); % coordinates corresponding to nodes

xplot = reshape(xx,size(tri));

yplot = reshape(yy,size(tri));figure(1);

clf;

fill(xplot',yplot','w');

title(['Triangular Mesh ', num2str(length(nn)),' elements'])

%%%%%%Subfunction % triangulate
```



<u>EXPERIMENT 5:</u> Write a program for 2D Strucutured grid over an airfoil.

```
%Algebraic Grid Generation
clear;
clc; %Assign values for t and beta
t=0.15;
beta=1.05; %Prompt user for number of grid points
n=input('Enter the number of grid points in the i direction: '); m=input('Enter the number of grid
points in the j direction: ');
%Create zeroes matrix for surface plots
z=zeros(n,m); %Assign lengths and values for eta and xi
L=3;
eta=linspace(0,1,m);
xi=linspace(0,L,n); %x is equal to xi
X=xi; %Find height
ytop=2;
for i=1:n
if X(i) < 1
        ybottom(i)=0;
elseif X(i) > 2
        ybottom(i)=0;
else
        x2(i)=X(i)-1;
        ybottom(i)=(t/.2)*(0.2969*x2(i)^.5-0.126*x2(i)-0.3516*x2(i)^2+0.2843*x2(i)^3-
        0.1015*x2(i)^4);
end
        H(i)=ytop-ybottom(i);
end %Loop to calculate coordinates
zeta=beta+1;
gamma=beta-1;
alpha=zeta/gamma;
for i=1:n
for j=1:m
        chi=1-eta(j);
        y(i,j)=H(i)*(zeta-gamma*alpha^chi)/(alpha^chi+1)+ybottom(i);
        x(i,j)=X(i);
end
end surface(x,y,z);
xlabel ('x');
ylabel ('y');
title ('Algerbraic Grid');
```



Discussion of Results Enter the number of grid points in the i direction: 50 Enter the number of grid points in the j direction: 50 Figure shows the algebraic grid generation with the growth rate β=1.05 the grids are very fine at y=0 and it gets coarser as the y increases. The value of growth rate β can be varied and you can see the difference in the growth rate of the grid. 0 0.5 1 1.5 2 2.5 3 0 0.2 0.4 0.6 0.81 1.2 1.4 1.6 1.82 x y Algerbraic Grid.

EXPERIMENT 6: Write a program to get the exact solution of Burgers equation.

Program for function f

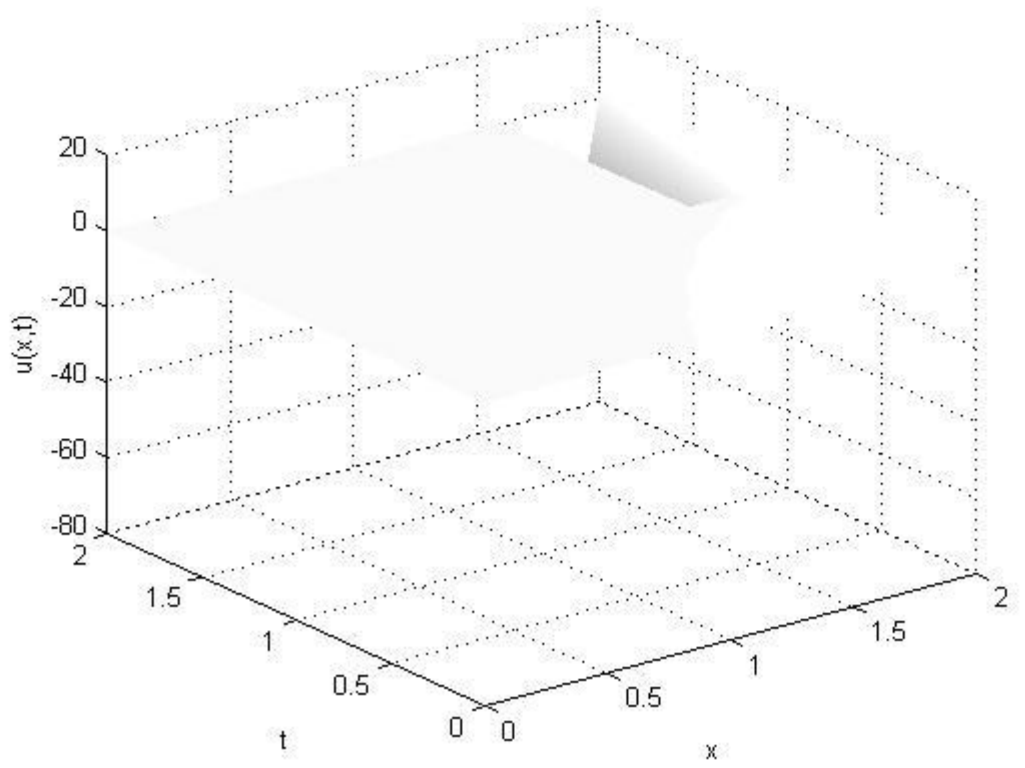
```
function ret = f( u )
ret = 0.5 * u.^2;
function ret = df( u )
ret = u;
function ret = nf( u, v )
for i = 1:length(u)
if (u(i) >= v(i))
if ((u(i)+v(i))/2 > 0)
ustar(i)=u(i);
else
ustar(i)=v(i);
end
else
if (u(i)>0)
ustar(i)=u(i);
elseif (v(i)<0)
ustar(i)=v(i);
else
ustar(i)=0;
end
end
end
ret =f(ustar);
```

Burgers Programm

```
clear all;
%Selection of equation and method.
xend = 2; % x-axis size.
tend = 2; % t-axis size.
N = input('Enter the number of grid points: ');
dx = xend/N; % Grid spacing
dt = input('Enter time step dt: ');
x = 0:dx:xend;
nt = floor(tend/dt);
dt = tend / nt;
%Set up the initial solution values.
u0=(0:0.2:2);
u=u0;
unew = 0*u;
%Implementation of the numerical methods.
for i = 1 : nt
    us = u(1:end-1) - dt/dx * (f(u(2:end)) - f(u(1:end-1)));
unew(2:end-1)= 0.5*(u(2:end-1) + us(2:end)) - ...
0.5*dt/dx* (f(us(2:end)) - f(us(1:end-1)));
unew(1) = u(1);
unew(end) = u(end);
u = unew;
U(i,:) = u(:);
end
U=[u0;U];
T=0:dt:tend;
%Plot of the solutions.
figure(1)
```

```
surf(x,T,U)
shading interp
xlabel('x'), ylabel('t'), zlabel ('u(x,t)');
grid on
colormap('Gray');
```

```
Enter the number of grid points: 10
Enter time step dt: 1
```



EXPERIMENT 7: Write a Program for Blasius solution for laminar boundary layer over a flat plate.

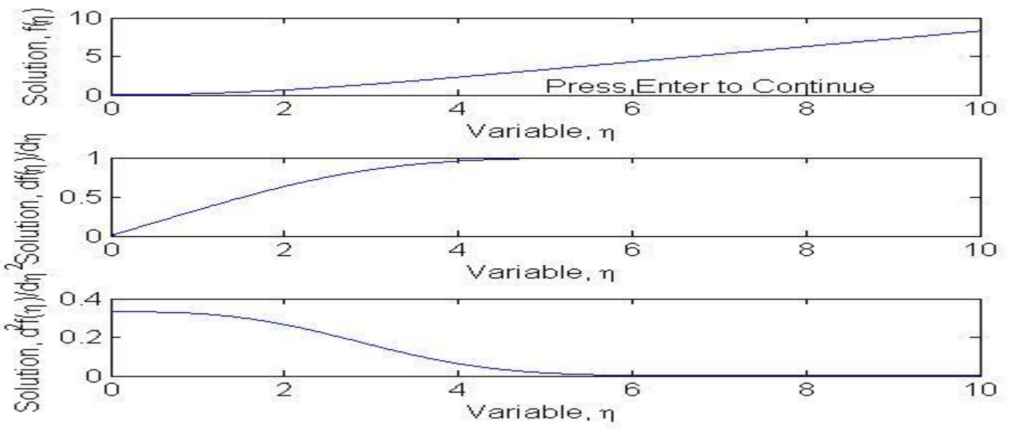
```
% This script animates the solution to the Balsius' model for a boundary
% layer flow over a flat plate. This model assumes steady flow, constant
% density and viscosity. First one must solve the Blasius differential
% equation f'''+0.5*f*f''= 0 with f(0)=0, f'(0)=0 AND f'(infinity)=0 where
% the prime denotes differentiation wuth respect to eta=y*(U*rho/mu)^.5.

clear all
set(0,'DefaultAxesFontSize',12);
set(0,'DefaultTextFontSize',12);
eta=linspace(0,10,2001);
deta=.005;
z1=0;
z2=0;
z3=.33193;%input('The second derivative at eta = 0 is');
% Found by trial and error assuming various values until the final
% condition was satisfied.
X=[z1;z2;z3];
for i=1:2000
z1=z1+z2*deta;
z2=z2+z3*deta;
z3=z3-0.5*z1*z3*deta;
X=[X [z1;z2;z3]];
end
figure(1);clf;
subplot(3,1,1)
plot(eta,X(1,:))
xlabel('Variable, \eta')
ylabel('Solution, f(\eta)')
subplot(3,1,2)
plot(eta,X(2,:))
xlabel('Variable, \eta')
ylabel('Solution, df(\eta)/d\eta')
subplot(3,1,3)
plot(eta,X(3,:))
xlabel('Variable, \eta')
ylabel('Solution, d^2f(\eta)/d\eta^2')
F=eta.*X(2,:)-X(1,:);
text(5,1.5,'Press Enter to Continue')
pause

figure(2);clf;
plot(eta,F)
xlabel('Variable, \eta')
ylabel('Solution, \etadf(\eta)/d\eta-f(\eta)')
text(4,.8,'Press Enter to Continue')
pause
y=linspace(0,.003,51);
x=[0 .05 .1 .15 .2 .25 .3 .35 .4 .45 .5]*.1;

figure(3);clf;
```

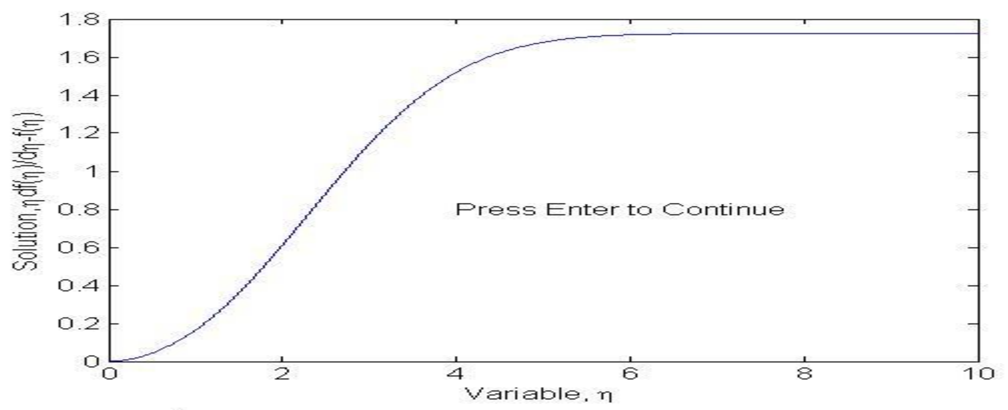
```
axis([0 1 -.0001 .003])
hold on
plot([0 1],[0 0])
hold on
u=zeros(51,11);
eta3=[];
% (U*rho/mu)=1e5:Corresponds to water at 20 C with U=0.1 m/s
for i=2:51 % y loop
for j=2:11 % x loop
eta1=sqrt(1e5/(x(1,j)))*y(1,i);
if y(1,i)==0
fprime=0;
fact=0;
else
fprime=interp1(eta,X(2,:),eta1);
fact=interp1(eta,F,eta1);
end

u(i,j)=fprime;
v(i,j)=sqrt(1/(1e5*x(1,j)))*fact;
eta3(i,j)=eta1;
end
end
u(:,1)=ones(51,1);
v(:,1)=zeros(51,1);
v(1,:)=zeros(1,11);
for i=39:51
u(i,2)=1;
end
for j=1:11
plot(u(:,j),y)
hold on
end
text(.8,.6e-3,'x = 0.005');
text(.7,.7e-3,'0.01');
text(.38,1e-3,'0.05');
box on
xlabel('Dimensionless Velocity, u(x,y)/U')
ylabel('Distance Above the Surface, y')
text(0.2,2e-3,['\rhoU/\mu = ' num2str(1e5) ' m^-^1'])
text(.2,2.5e-3,'Press Enter to Continue')
pause

 figure(4);clf;
axis([-.01 .06 -.0002 .006])
hold on
patch([0 .005 .062 .06 0],[0 -.0002 -.0002 0 0],'r')
box on
plot([0 .004 .004 0 0],[0 0 3e-3 3e-3 0])
patch([.004 .003 .003 .004],[3e-3 (3e-3)-.0001 (3e-3)+.0001 3e-3],'b')
hold on
for i=2:11
plot(x(1,i)*ones(1,51)+.004*u(:,i)',y(1,:))
xtip=x(1,i)+.004*u(51,i);
ytip=y(1,51);
```

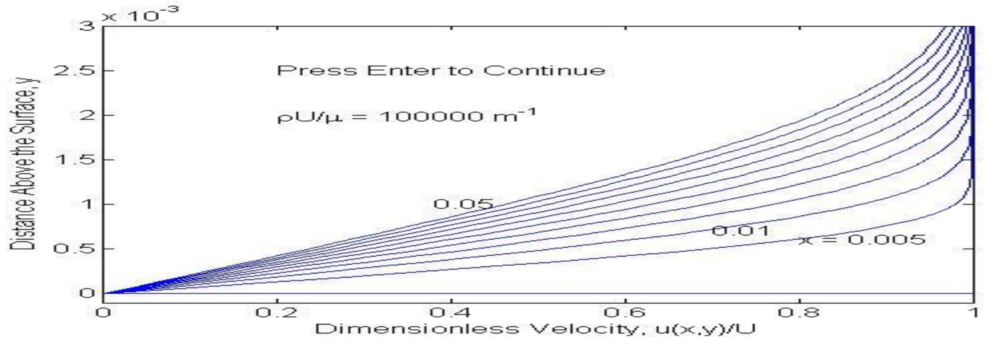
```
plot([x(1,i) x(1,i) xtip],[0 ytip ytip])
patch([xtip xtip-.001 xtip-.001 xtip ],[ytip ytip-.0001 ytip+.0001 ytip],'b')
hold on
end
x3=linspace(0,.06,201);
delta3=5*sqrt(x3/1e5);
plot(x3,delta3,'m')
hold on
plot([x3(1,150) .033],[delta3(1,150) 3.5e-3],'m');
text(.004,3.5e-3, 'Boundary Layer Thickness')
xlabel('Distance along the plate x, m')
ylabel('Distance above the plate y, m')
text(-.005,4.8e-3,'U\rho/\mu = 100000 m^-^1, Water at U = 0.1 m/s and 20 C')
text(-.005,5.5e-3,'Horizontal Velocity Profiles vs Distance along the Plate')
text(-.005, 4e-3,'Press Enter to Continue')
pause

 dt=.00005;
x1=-5*ones(8,1);
beginningx=-5;
y1=[.5e-5 .00001 .00002 .00003 .00004 .00005 .00006 .00007]'*2;
hold on
v1=zeros(8,1);
u1=ones(8,1);

figure(5);clf;%Do animation
axis([beginningx 200 -2 80]);
hold on
patch([0 4 200 200 0],[0 -2 -2 0 0],'r')
x1new=.000001*ones(8,1);
y1new=y1;
Udt=.1*dt;
plot([x1 x1new]'/Udt,[y1 y1new]'/Udt, 'k');
box on
xlabel('Dimensionless Location, x/U\Deltat')
ylabel('Dimensionless Location, y/U\Deltat')
text(10,70 ,['\rhoU/\mu = ' num2str(1e5) ' m^-^1, U = 0.1 m/s'])
T=text(100,5,'Press Enter to Animate');
text(10,65,'\Deltat = 0.00005 s')
pause
set(T,'String',' ');
x1=x1new;
y1=y1new;
for t=1:200%time loop
for i=1:8%streamline loop
eta3=sqrt(1e5/(x1(i,1)))*y1(i,1);
if eta3>=10
fprime=1;
F1=1.7233;
else
fprime=interp1(eta, X(2,:),eta3);
F1=interp1(eta,F,eta3);
end
u1(i,1)=fprime;
v1(i,1)=sqrt(1/(4e5*x1(i,1)))*F1;
```

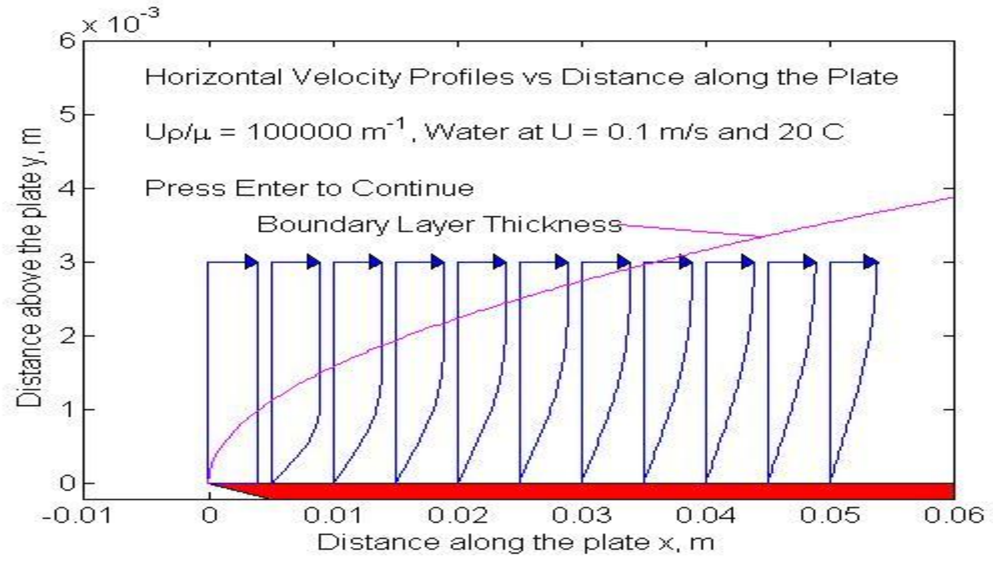
```
x1new(i,1)=x1(i,1)+dt*u1(i,1)*.1;
y1new(i,1)=y1(i,1)+dt*v1(i,1)*.1;
hold on
end

plot([x1 x1new]'/Udt,[y1 y1new]'/Udt, 'k');
x1=x1new;
y1=y1new;
pause(.01)
end
x4=linspace(0,200,201);
delta4=5*sqrt(x4/.50);
plot(x4,delta4,'m','Linewidth',2)
text(100,62,'Boundary Layer Thickness')
plot([x4(1,90),97],[delta4(1,90),63],'m','LineWidth',2);
set(T,'String','Press Enter to Continue')
pause

x1=-5*ones(8,1);
beginningx=-5;
y1=[.5e-5 .00001 .00002 .00003 .00004 .00005 .00006 .00007]'*5;
v1=zeros(8,1);
u1=ones(8,1);

figure(6);clf;%Do animation
axis([beginningx 200 -3 120]);
hold on
patch([0 4 200 200 0],[0 -3 -3 0 0],'r')
x1new=.000001*ones(8,1);
y1new=y1;
plot([x1 x1new]'/Udt,[y1 y1new]'/Udt, 'k');
box on
xlabel('Dimensionless Location, x/U\Deltat')
ylabel('Dimensionless Location, y/U\Deltat')
text(20,110 ,['\rhoU/\mu = ' num2str(1e5) ' m^-^1, U = 0.1 m/s'])
%text(20,110 ,['\rhoU/\mu = ' num2str(1e5) ' m^-^1'])
T=text(120,10,'Press Enter to Animate');
text(20,100,'\Deltat = 0.00005 s')
pause
set(T,'String',' ');
x1=x1new;
y1=y1new;
u8=[];
for t=1:200%time loop
for i=1:8%streamline loop
eta3=sqrt(1e5/(x1(i,1)))*y1(i,1);
if eta3>=10
fprime=1;
F1=1.7233;
else
fprime=interp1(eta, X(2,:),eta3);
F1=interp1(eta,F,eta3);
end
u1(i,1)=fprime;%actuallu u/U
v1(i,1)=sqrt(1/(4e5*x1(i,1)))*F1;%actually v/U
```

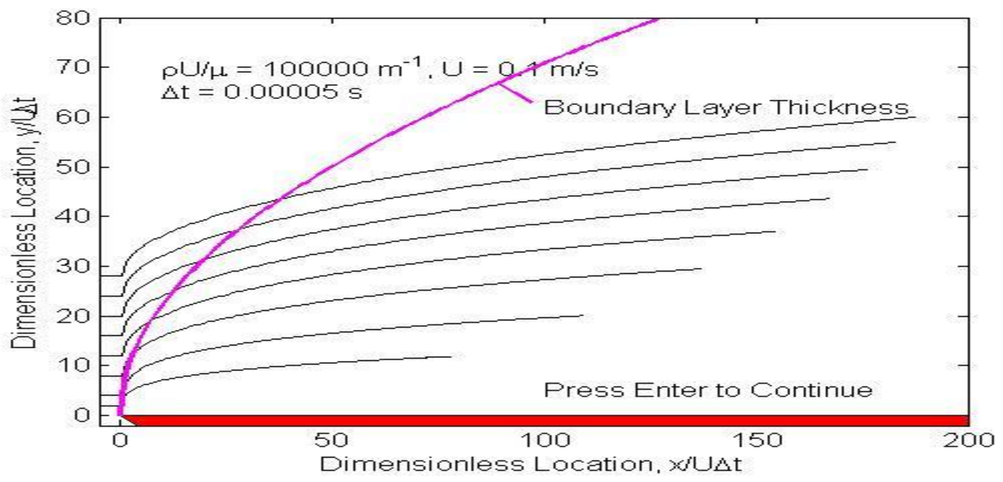
```
x1new(i,1)=x1(i,1)+dt*u1(i,1)*.1;
y1new(i,1)=y1(i,1)+dt*v1(i,1)*.1;
hold on
end
plot([x1 x1new]'/Udt,[y1 y1new]'/Udt, 'k');
hold on
x1=x1new;
y1=y1new;
pause(.01)
end
plot(x4,delta4,'m','Linewidth',2)
plot([x4(1,90),99],[delta4(1,90),19],'m','Linewidth',2);
text(100,18,'Boundary Layer Thickness')
set(T,'String','Press Enter to End')
pause
```

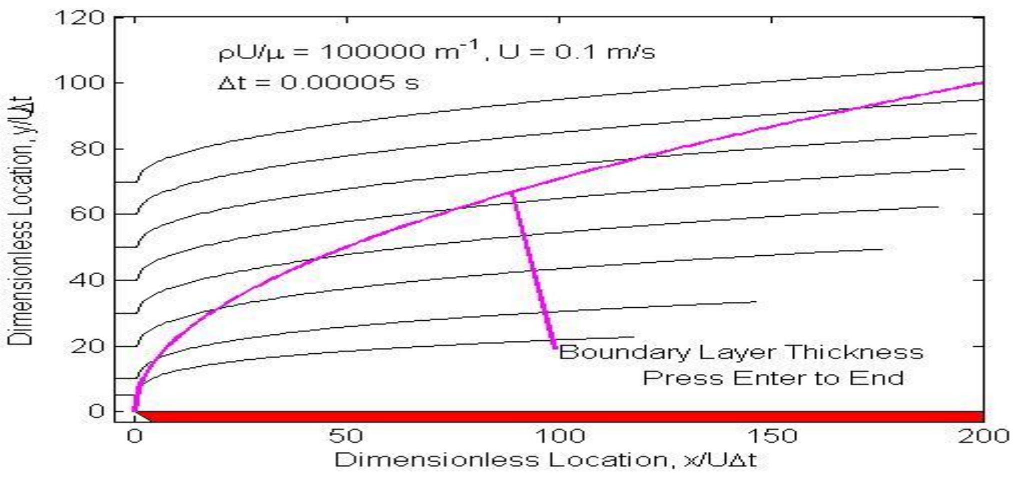







x 10^-3
Horizontal Velocity Profiles vs Distance along the Plate
Uρ/μ = 100000 m^-1, Water at U = 0.1 m/s and 20 C
Press Enter to Continue
Boundary Layer Thickness
Distance above the plate y, m
Distance along the plate x, m


ρU/μ = 100000 m^-1, U = 0.1 m/s
Δt = 0.00005 s
Boundary Layer Thickness
Press Enter to Continue
Dimensionless Location, y/UΔt
Dimensionless Location, x/UΔt


ρU/μ = 100000 m^-1, U = 0.1 m/s
Δt = 0.00005 s
Boundary Layer Thickness
Press Enter to End
Dimensionless Location, y/UΔt
Dimensionless Location, x/UΔt

Experiment - 8: Flow over Airfoil

AIM: To simulate flow over NACA 0012 airfoil

Problem description:

Consider air flowing over NACA 0012 airfoil. The free stream mach number is 1.2 and the angle of attack is 5°. Assume standard sea-level values for the free stream properties.

Steps Involved In ICEMCFD:

Creation of Geometry in ICEMCFD:

- Importing the Aerofoil coordinates

File→Import Geometry→Formatted point data→Select the file of aerofoil coordinates which is in DAT format→ok. Now the coordinates will be displayed.

- Geometry→Create/modify curve→From points→Select above points and leave last 2 points→middle click
- Similarly on bottom side
- Join the end points of the curves

1) Creation of parts:

- Parts in the tree→Right click→Create part→

  Select Upper curve: Suction

  Select Lower curve: Pressure 3rd

  Line: TE

2) Creation of Domain:

  - Create points (-1,1),(-1,-1),(2,1),(2,-1)
- Join these points
- Create parts as Inlet, Outlet, Top & Bottom
- Geometry→Create/Modify surface→Simple surface→Select all the lines of domain→ok
- Create the new part as: Surface

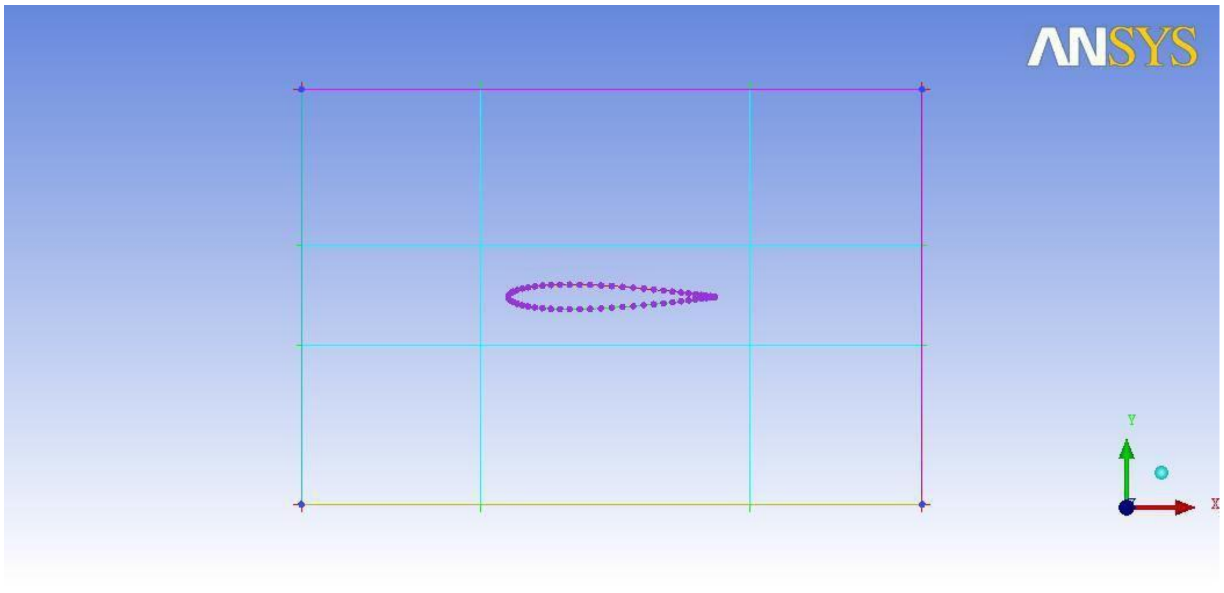
3) Saving the Geometry:

- File→Change working directory→Choose the folder

- File→Geometry→Save Geometry as→Give the name.

4) Creation of Blocking and Association:

- Blocking→Create block→Initialize blocks→Type as:2D Planar→ok
- Associate→Associate vertex to point→Select a vertex and a point→Apply→ Similarly associate remaining 3 vertices to points
- Associate→Associate edge to curve→Select a edge and a curve →Apply→ Similarly associate remaining 3 edges to curves
- Split block→Select the edges→Create the blocks as shown in figure

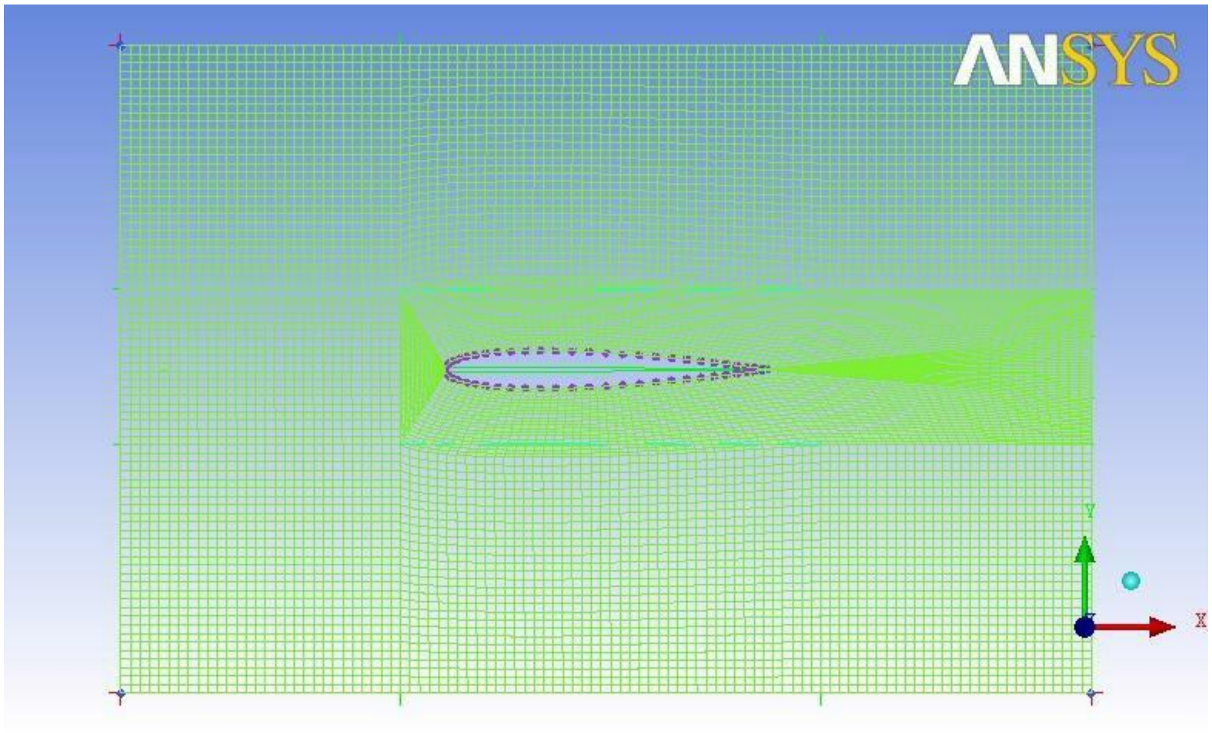


- Split block→O grid →Select edges→Select last 2 edges in the middle row→ok→Select blocks→Select last 2 blocks in the middle row→ok
    - Associate→ Associate vertex to point→Select the vertex of the O grid and the 2nd point on the upper curve(suction)→ok Similarly associate remaining 3 vertices of the O grid to the points on the aerofoil as shown in the below fig.
    - Associate→ Associate edge to curve→Select the 3 edges of block which is inside of the aerofoil and select the suction & pressure curves→ok Similarly associate the TE edge to TE curve.
    - Delete block→Select the block inside the aerofoil→ok

5) Generation of Mesh:

- Pre-mesh parameters→Edge parameters→Switch ON the Copy Parameters→ Select the edges and give desired no. of nodes→ok
- Switch ON Pre-mesh in the tree→click yes to compute the meshing
- Pre-mesh→Right click→Convert to unstructured mesh

  Now the required mesh has been generated as shown in below fig.



6) Saving the Project:

- File→ Save Project as→Give the name.

7) Writing output file:

- Output→Select solver→Output solver as: Fluent_V6→Common Structural solver as: ANSYS→ok
- Write input→Click NO→Open the file→Click 2D→ok

3 Steps Involved in Fluent:

8) Importing the mesh file:

- File→Read→mesh→Choose the output file written in ICEM CFD Now the mesh has imported into the fluent solver.

9) Problem setup:

- General→Type as: Pressure based
- Models→Energy ON→Viscous-laminar
- Materials→Air
- Cell zone conditions→ Type as: fluid→ok
- Boundary conditions→Select inlet→Edit→Give velocity magnitude as: 400m/s.
- Boundary conditions→Select outlet→Edit→Give gauge pressure as: 0 Pa

10) Solution:

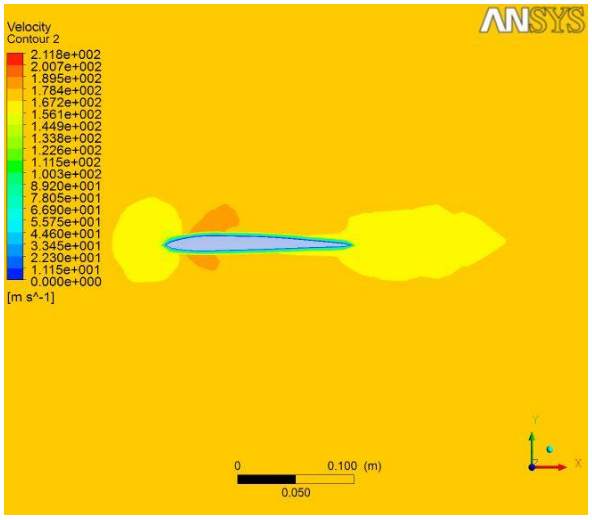
- Select the required monitors
- Solution initialization→Compute from: inlet→Initialize
- Run calculations→Enter the no. of iterations as: 1000→Calculate

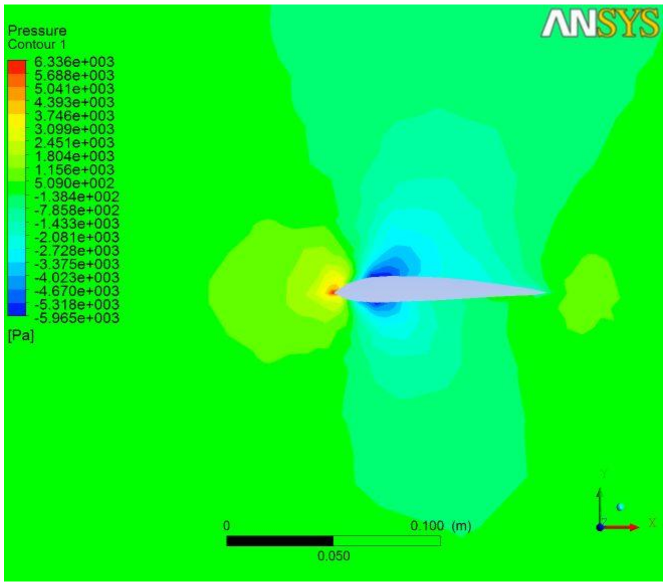
11) Results:

- Graphics and animations→elect the required flow parameters in the contours and vectors.

The results are shown below as:

Results:


Velocity
Contour 2
2.118e+002
2.007e+002
1.895e+002
1.784e+002
1.672e+002
1.561e+002
1.449e+002
1.338e+002
1.226e+002
1.115e+002
1.003e+002
8.920e+001
7.805e+001
6.690e+001
5.575e+001
4.460e+001
3.345e+001
2.230e+001
1.115e+001
0.000e+000
[m s^-1]
ANSYS
0
0.100 (m)
0.050
Y
X


Pressure
Contour 1
6.336e+003
5.688e+003
5.041e+003
4.393e+003
3.746e+003
3.099e+003
2.451e+003
1.804e+003
1.156e+003
5.090e+002
-1.384e+002
-7.858e+002
-1.433e+003
-2.081e+003
-2.728e+003
-3.375e+003
-4.023e+003
-4.670e+003
-5.318e+003
-5.965e+003
[Pa]
ANSYS
0
0.100 (m)
0.050
Y
X

Experiment -9: Flow over a wing

Aim: To obtain flow field over a finite rectangular wing with incoming flow conditions velocity 120 m/s, Pressure 1 atm and air at 25^0c.

Software: ICEM and CFX

Procedure: Pre processing:

1. **Create geometry in ICEM**
    - **Import vertex data of airfoil coordinates from File** → Import Geometry → Formatted point data.
    - Join the points by selecting geomtry tab, Create curve from points.
    - Create a line in Z direction to extrude airfoil.
    - Geometry → Create surface → Curve driven → select driving curve and then the remaining lines under the driven curve → Apply.
    - Geometry → Create surface → Simple surface → for upper and lower surface.
    - Create domain for the flow analysis.
    - Delete unwanted surfaces, lines and points.
    - From the model tree → right click on parts → create part → Name

      Inlet – in_1
      Outlet – out_1
      Top – top_1
      Side – sidewall_1
      Bottom – Bot _ 1
      Airfoil – Airfoil_1
      Apply
    - Geometry → Create body → Location → centroid of 2 points → select the two diagonally opposite points on the model such that point should not be within airfoil→ Apply.

2. **Mesh geometry:**
    - Click Mesh Function tab
    - Give global mesh parameters in global mesh setup like element scale factor and element max size = 0.1 → display on → apply.
    - Create partmesh setup for inlet, outlet and wall.
    - Compute mesh → volume mesh → mesh type → tetra/mixed → create prism layers → create hexa-core → mesh method → Robust[octree] → select geometry – compute.

- Edit mesh – check mesh – quality mesh – smooth mesh.

3. **Export mesh**
   - Output – output to cfx – save project – output type – output scale factor - .msh file created.
4. **CFX:**
   - CFX Pre – New file – general.
   - Mesh - Import mesh – ICEM CFD – open .msh file.
   - Default Domain Basic Settings Materials Air at 25^{0}c, Reference Pressure 1 atm

Domain - Default Domain Modified	
Type	Fluid
Location	FLUID
Materials	
Air Ideal Gas	
Fluid Definition	Material Library
Morphology	Continuous Fluid
Settings	
Buoyancy Model	Non Buoyant
Domain Motion	Stationary
Reference Pressure	1.0000e+00 [atm]
Heat Transfer Model	Isothermal
Fluid Temperature	2.5000e+01 [C]
Turbulence Model	SST
Turbulent Wall Functions	Scalable

Domain	**Boundaries**	
Default Domain Modified	**Boundary - inlet**	
	Type	INLET
	Location	IN_1
	Settings	
	Flow Regime	Subsonic
	Mass And Momentum	Normal Speed
	Normal Speed	1.3000e+02 [m s^-1]

	Turbulence	low Intensity = 1%
	Boundary - outlet	
	Type	OUTLET
	Location	OUT_1
	Settings	
	Flow Regime	Subsonic
	Mass And Momentum	Average Static Pressure
	Pressure Profile Blend	5.0000e-02
	Relative Pressure	1.0132e+05 [Pa]
	Pressure Averaging	Average Over Whole Outlet
	Boundary - WALL	
	Type	WALL
	Location	Airfoil_1, Top_1, Bot _1
	Settings	
	Mass And Momentum	No Slip Wall
	Wall Roughness	Smooth Wall

- Click ok and define run to start solution.

POST PROCESSING

5. **Analyze results in CFX POST:**
    - Create plane
    - Create contour – pressure, mach number
    - Create streamlines
    - Create chart for velocity along plate.

RESULTS:

EXPERIMENT 10: Simulation of compressible flow in a convergent-divergent nozzle

Aim: Consider the nozzle having a cross sectional area A varies with axial distance from the throat, according to the formula $A = 0.1+X^2$; where X varies from $-0.5<X<0.5$. Stagnation pressure P_o = 101325 pa; stagnation temperature T_o = 300K;

Software: ICEM and CFX

Procedure:

Pre processing:

1. **Create geometry in ICEM**
   - Create the vertex data of Nozzle contour variation along axis line with vertices (-0.5, 0) and (0.5, 0) and $A = 0.1+X^2$ in excel sheet.
   - Import vertex data into ICEM by creating .dat file.
   - Create edges by create/modify curves and face by create/modify faces of nozzle using geometry function tab.
   - Create body point.
   - Create parts inlet, outlet and wall.

2. **Mesh geometry:**
   - Click Mesh Function tab
   - Give global mesh parameters in global mesh setup like element scale factor and element max size.
   - Create partmesh setup for inlet, outlet and wall.
   - Compute mesh – volume mesh – mesh type – select geometry – compute.
   - Edit mesh – check mesh – quality mesh – smooth mesh.

3. **Export mesh**
   - Output – output to cfx – save project – output type – output scale factor - .msh file created.

4. **CFX:**
   - CFX Pre – New file – general.
   - Mesh - Import mesh – ICEM CFD – open .msh file.

Domain - Default Domain Modified	
Type	Fluid
Location	FLUID
Materials	
Air Ideal Gas	
Fluid Definition	Material Library
Morphology	Continuous Fluid
Settings	
Buoyancy Model	Non Buoyant
Domain Motion	Stationary
Reference Pressure	1.0000e+00 [atm]
Heat Transfer Model	Isothermal
Fluid Temperature	2.5000e+01 [C]
Turbulence Model	k epsilon
Turbulent Wall Functions	Scalable

Domain	**Boundaries**	
Default Domain Modified	**Boundary - inlet**	
	Type	INLET
	Location	IN
	Settings	
	Flow Regime	Subsonic
	Mass And Momentum	Normal Speed
	Normal Speed	2.8000e+02 [m s^-1]
	Turbulence	Medium Intensity and Eddy Viscosity Ratio
	Boundary - outlet	
	Type	OUTLET
	Location	OUT
	Settings	
	Flow Regime	Subsonic

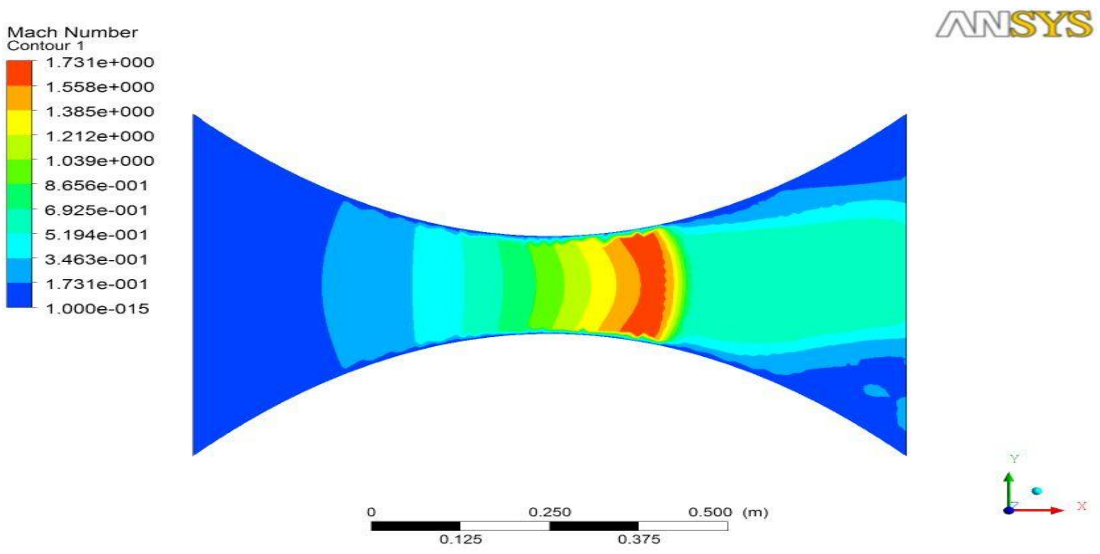
	Mass And Momentum	Average Static Pressure
	Pressure Profile Blend	5.0000e-02
	Relative Pressure	1.0132e+05 [Pa]
	Pressure Averaging	Average Over Whole Outlet
	Boundary - WALL	
	Type	WALL
	Location	Wall
	Settings	
	Mass And Momentum	No Slip Wall
	Wall Roughness	Smooth Wall

5. **Solve the problem:**

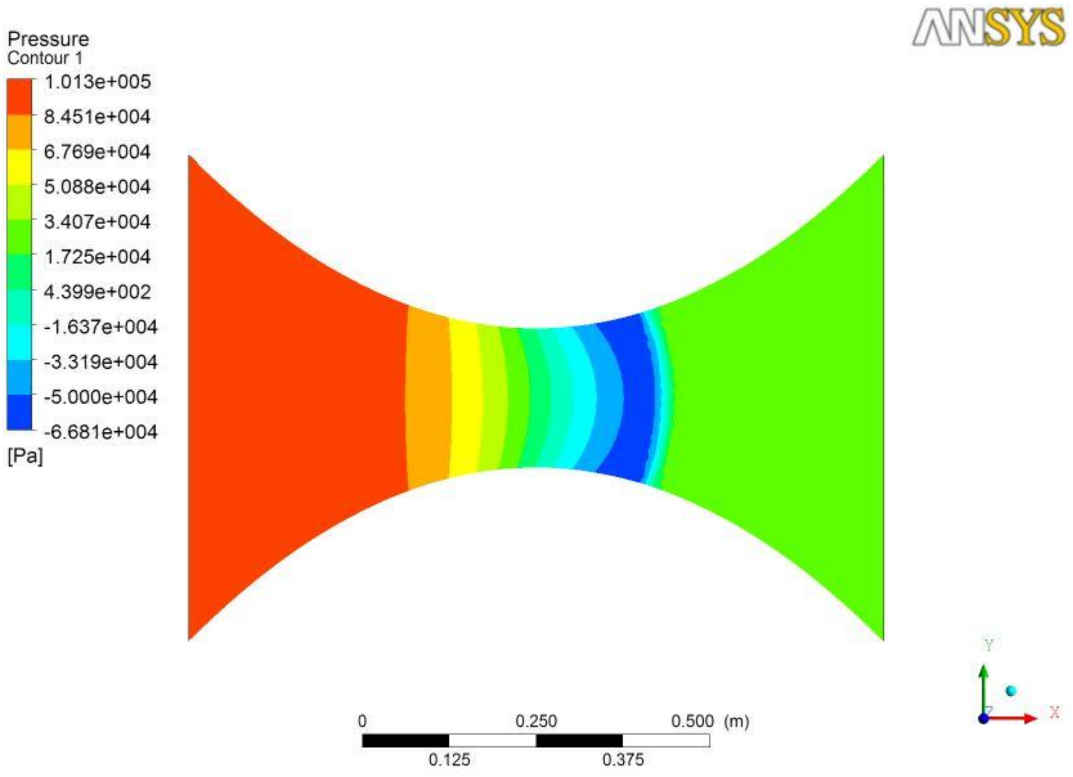
- Solver Control – Basic setting
- Advection scheme
- Turbulence Numerics
- Convergence controls
- Fluid timescale control
- Convergence criteria

POST PROCESSING

6. **Analyze results:**
- Create plane
- Create contour – pressure, mach number
- Create streamlines
- Create chart for temperature along nozzle axis.



Mach number contour



Pressure contour