COMPUTATIONAL AERODYNAMICS

LABORATORY MANUAL

B.TECH
(IV YEAR – I SEM)
(2017-18)

Prepared by:
Mr. J. Sandeep,
Assistant Professor

Department of Aeronautical Engineering

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956
Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – A Grade - ISO 9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India.

DEPARTMENT OF AERONAUTICAL ENGINEERING

VISION

Department of Aeronautical Engineering aims to be indispensable source in Aeronautical Engineering which has a zeal to provide the value driven platform for the students to acquire knowledge and empower themselves to shoulder higher responsibility in building a strong nation.

MISSION

a) The primary mission of the department is to promote engineering education and research.

(b) To strive consistently to provide quality education, keeping in pace with time and technology.

(c) Department passions to integrate the intellectual, spiritual, ethical and social development of the students for shaping them into dynamic engineers.

PROGRAMME EDUCATIONAL OBJECTIVES (PEO'S)

PEO1: PROFESSIONALISM & CITIZENSHIP To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.
PEO2: TECHNICAL ACCOMPLISHMENTS To provide knowledge based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field.
PEO3: INVENTION, INNOVATION AND CREATIVITY To make the students to design, experiment, analyze, interpret in the core field with the help of other multi disciplinary concepts wherever applicable.
PEO4: PROFESSIONAL DEVELOPMENT To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.
PEO5: HUMAN RESOURCE DEVELOPMENT To graduate the students in building national capabilities in technology, education and research.

PROGRAM SPECIFIC OBJECTIVES (PSO's)

1. To mould students to become a professional with all necessary skills, personality and sound knowledge in basic and advance technological areas.
2. To promote understanding of concepts and develop ability in design manufacture and maintenance of aircraft, aerospace vehicles and associated equipment and develop application capability of the concepts sciences to engineering design and processes.
3. Understanding the current scenario in the field of aeronautics and acquire ability to apply knowledge of engineering, science and mathematics to design and conduct experiments in the field of Aeronautical Engineering.
4. To develop leadership skills in our students necessary to shape the social, intellectual, business and technical worlds.

PROGRAM OBJECTIVES (PO'S)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

IV Year B. Tech. A. E – I Sem

L	T/P/D	C
-	-/3/-	2

(A72186)COMPUTATIONAL AERODYNAMICS LAB

LIST OF EXPERIMENT:

1. Introduction to any one of the suitable software employed in modeling and simulation of aerodynamic problems.

2,3. Solution for the following equations using finite difference method(code development).

i) One dimensional wave equations using explicit method of lax

ii) One dimensional heat conduction equation using explicit method

4,5. Generation of the following grids(code development).

i) Algebraic Grid

iii) Elliptic Grids.

6,7,8,9,10. Numerical simulation of the following flow problems using commercial software packages:

i) Flow over an airfoil.

ii) Supersonic flow over a wedge.

iii) Flat plate boundary layer.

iv) Laminar flow through pipe.

v) Flow past a cylinder.

Suggested software:

1. ANSYS FLUENT and CFX
2. MATLAB

CONTENTS

CODE OF CONDUCT FOR THE LABORATORIES

- All students must observe the Dress Code while in the laboratory.
- Sandals or open-toed shoes are NOT allowed.
- Foods, drinks and smoking are NOT allowed.
- All bags must be left at the indicated place.
- The lab timetable must be strictly followed.
- Be PUNCTUAL for your laboratory session.
- Program must be executed within the given time.
- Noise must be kept to a minimum.
- Workspace must be kept clean and tidy at all time.
- Handle the systems and interfacing kits with care.
- All students are liable for any damage to the accessories due to their own negligence.
- All interfacing kits connecting cables must be RETURNED if you taken from the lab supervisor.
- Students are strictly PROHIBITED from taking out any items from the laboratory.
- Students are NOT allowed to work alone in the laboratory without the Lab Supervisor
- USB Ports have been disabled if you want to use USB drive consult lab supervisor.
- Report immediately to the Lab Supervisor if any malfunction of the accessories, is there.

Before leaving the lab

- Place the chairs properly.
- Turn off the system properly
- Turn off the monitor.
- Please check the laboratory notice board regularly for updates.

1. INTRODUCTION TO MODELING AND SIMULATION SOFTWARE TO AERODYNAMIC PROBLEMS

A **model** is a mathematical object that has the ability to predict the behavior of a real system under a set of defined operating conditions and simplifying assumptions.The term *modeling refers to the development of a mathematical representation of a physical situation*

WHAT IS MODELING?

- Modeling is the process of producing a model.
- A model is a representation of the construction and working of some system of interest.
- A model is similar to but simpler than the system it represents.
- One purpose of a model is to enable the analyst to predict the effect of changes to the system. Generally, a model intended for a simulation study is a mathematical model developed with the help of simulation software.
- Mathematical model classifications include
- Deterministic (input and output variables are fixed values) or Stochastic (at least one of the input or output variables is probabilistic);
- Static (time is not taken into account) or
- Dynamic (time-varying interactions among variables are taken into account).
- Typically, simulation models are stochastic and dynamic.

Simulation is the process of exercising a model for a particular instantiation of the system and specific set of inputs in order to predict the system response.

simulation refers to the procedure of solving the equations that resulted from model development

WHAT IS SIMULATION?

- A simulation of a system is the operation of a model of the system.
- The operation of the model can be studied, and hence, properties concerning the behavior of the actual system or its subsystem can be inferred.
- In its broadest sense, simulation is a tool to evaluate the performance of a system, existing or proposed, under different configurations of interest and over long periods of real time.
- Simulation is used
- before an existing system is altered or a new system built,
- to reduce the chances of failure to meet specifications,
- to eliminate unforeseen bottlenecks,
- to prevent under or over-utilization of resources,
- to optimize system performance.

INTRODUCTION TO AERODYNAMICS

The study of the dynamics of air is known as aerodynamics. It includes the intermolecular forces, the motion of molecules due to variation of flow field variables like pressure, velocity, temperature etc. Aerodynamics is an applied science with many practical applications in engineering. No matter how elegant an aerodynamic theory may be, or how mathematically complex a numerical solution may be, or how sophisticated an aerodynamic experiment may be, all such efforts are usually aimed at one or more of the following practical objectives:

1. The prediction of forces and moments on, and heat transfer to, bodies moving through air. For example:
    - Estimation of lift, drag and moments of airfoils, wings, fuselages, engine nacelles and whole airplane.
    - Aerodynamic heating of flight vehicles ranging from the supersonic transport to planetary probe entering the atmosphere of Jupiter.

2. Determination of flows moving internally through ducts. For example:
    - To calculate and measure the flow properties in compressors, combustion chamber, nozzle of rockets and air breathing jet engines and to calculate engine thrust.
    - To know the flow conditions in the test section of a wind tunnel.
    - To know how much fluid can flow through pipes under different conditions.
    - A very interesting application of aerodynamics is high-energy chemical and gas dynamic lasers.

These types of problems are solved in Computational Aerodynamics lab in two ways as follows:

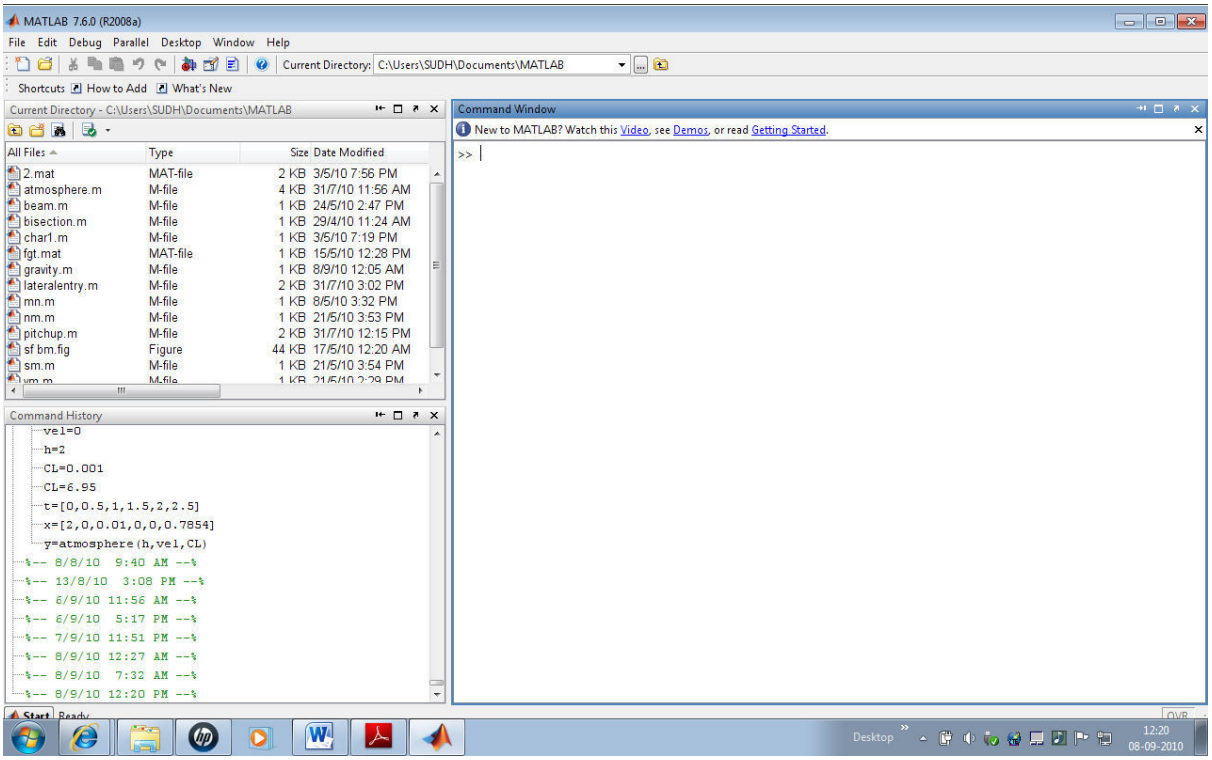
(i) Code development like MATLAB
(ii) Commercial CFD Software like ANYS

INTRODUCTION TO MATLAB

MATLAB – The language of Technical computing

MATLAB (*mat*rix *lab*oratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, and Fortran.

GETTING STARTED



The Matlab window consists of:

- COMMAND WINDOW
- COMMAND HISTORY
- CURRENT DIRECTORY

In the first command window the file called ***.m file*** which is done in a separate window in matlab is saved and made it run for execution.

In the command history the time is noted and displayed in this window from the staring of the program till the execution step of the program, also the list of steps or data which is given in command window is noted that gives us a better idea of how and what we

are doing when we get stuck up with the program as we will be able to read again the program when we miss any data regarding that program so that we can get back to the program, this is a simple way for execution.

In the command directory all the saved files which are executed in matlab are displayed and stored for easy opening of the program file.

SYNTAX

MATLAB, the application, is built around the MATLAB language. The simplest way to execute MATLAB code is to type it in at the prompt, >>, in the Command Window, one of the elements of the MATLAB Desktop. In this way, MATLAB can be used as an interactive mathematical shell. Sequences of commands can be saved in a text file, typically using the MATLAB Editor, as a script or encapsulated into a function, extending the commands available.

VARIABLES

Variables are defined with the assignment operator, =. MATLAB is a weakly dynamically programming language. It is a weakly typed language because types are implicitly converted. It is a dynamically typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function.

For example:

```
>> x = 17
x =
 17

>> x = 'hat'
x =
hat

>> y = x + 0
y =
     104       97      116

>> x = [3*4, pi/2]
x =
```

```
   12.0000    1.5708

>> y = 3*sin(x)
y =

          -1.6097    3.0000
```

MATLAB has several functions for rounding fractional values to integers:

- Round(X): round to nearest integer, trailing 5 rounds to the nearest integer away from zero. For example, round (2.5) returns 3; round(-2.5) returns -3.
- Fix(X): round to nearest integer toward zero (truncate). For example, fix(2.7) returns 2; fix(-2.7) returns -2
- Floor(X): round to the nearest integer toward minus infinity (round to the nearest integer less than or equal to X). For example, floor(2.7) returns 2; floor(-2.3) returns -3.
- ceil(X): round to the nearest integer toward positive infinity (round to the nearest integer greater than or equal to X); for example, ceil(2.3) returns 3; ceil(-2.7) returns –

VECTORS/MATRICES

MATLAB is a "Matrix Laboratory", and as such it provides many convenient ways for creating vectors, matrices, and multi-dimensional arrays. In the MATLAB vernacular, a *vector* refers to a one dimensional (1×*N* or *N*×1) matrix, commonly referred to as an array in other programming languages. A *matrix* generally refers to a 2-dimensional array, i.e. an *m*×*n* array where *m* and *n* are greater than or equal to 1. Arrays with more than two dimensions are referred to as multidimensional arrays.

MATLAB provides a simple way to define simple arrays using the syntax: init: increment: terminator. For instance:

```
>>array = 1:2:9
array =
 1 3 5 7 9
```

Defines a variable named array (or assigns a new value to an existing variable with the name array) which is an array consisting of the values 1, 3, 5, 7, and 9. That is, the array starts at 1 (the init value), increments with each step from the previous value by 2 (the increment value), and stops once it reaches (or to avoid exceeding) 9 (the terminator value).

The *increment* value can actually be left out of this syntax (along with one of the colons), to use a default value of 1.

```
>>ari = 1:5
ari =

 1 2 3 4 5
```

Indexing is one-based, which is the usual convention for matrices in mathematics, although not for some programming languages.

Matrices can be defined by separating the elements of a row with blank space or comma and using a semicolon to terminate each row. The list of elements should be surrounded by square brackets: []. Parentheses: () are used to access elements and sub arrays (they are also used to denote a function argument list).

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
16  3  2 13
 5 10 11  8
 9  6  7 12
 4 15 14  1

>>A(2,3)
ans =

 11
```

Sets of indices can be specified by expressions such as "2:4", which evaluates to [2, 3, 4]. For example, a submatrix taken from rows 2 through 4 and columns 3 through 4 can be written as:

```
>>A(2:4,3:4)
ans =
 11 8
 7 12
 14 1
```

A square identity matrix of size n can be generated using the function eye, and matrices of any size with zeros or ones can be generated with the functions zeros and ones, respectively.

```
>>eye(3)
ans =
 1 0 0
 0 1 0
```

```
 0 0 1

>>zeros(2,3)
ans =
 0 0 0
 0 0 0

>>ones(2,3)
ans =
 1 1 1
 1 1 1
```

Most MATLAB functions can accept matrices and will apply themselves to each element. For example, mod (2*J,n) will multiply every element in "J" by 2, and then reduce each element modulo "n". MATLAB does include standard "for" and "while" loops, but using MATLAB's vectorized notation often produces code that is easier to read and faster to execute. This code, excerpted from the function *magic.m*, creates a magic square *M* for odd values of *n* (MATLAB function *meshgrid* is used here to generate square matrices I and J containing 1: n).

```
[J,I] = meshgrid(1:n);
A = mod(I+J-(n+3)/2,n);
B = mod(I+2*J-2,n);
M = n*A + B + 1
```

Semicolons

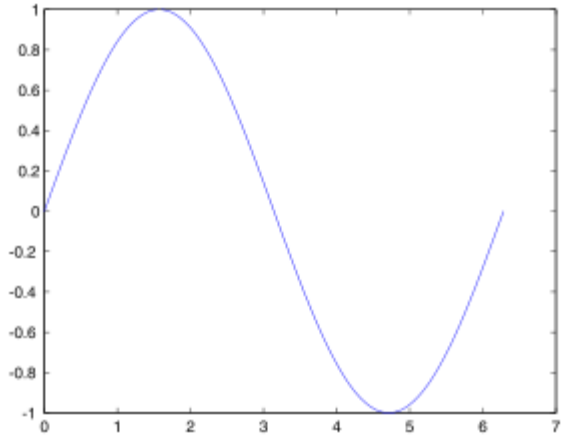
Unlike many other languages, where the semicolon is used to terminate commands, in MATLAB the semicolon serves to suppress the output of the line that it concludes (it serves a similar purpose in Mathematica.)

Graphics

Function *plot* can be used to produce a graph from two vectors *x* and *y*. The code:

```
x = 0: pi/100:2*pi;
y = sin(x);
Plot(x,y)
```

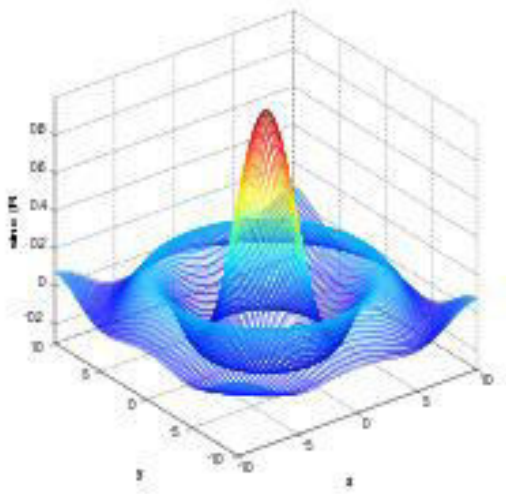
Produces the following figure of the sine function:



Three-dimensional graphics can be produced using the functions surf, plot3 or mesh.

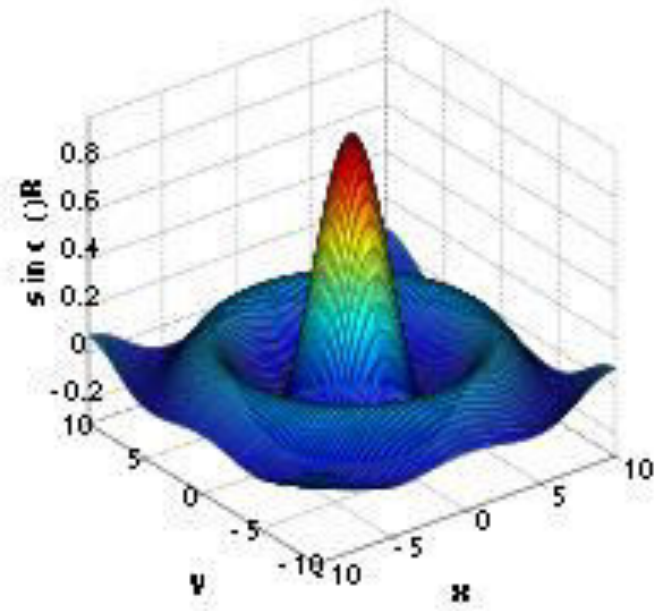
```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
mesh(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
hidden off
```

This code produces a wireframe 3D plot of the two-dimensional unnormalized sine function:



```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
surf(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
```

This code produces a surface 3D plot of the two-dimensional normalized sine function:



'if' statements, 'for' and 'while' Loops

'if' statements

Purpose: Tests an expression and only executes the code if the expression is true.

Format:

if *expression*

statement(s) to be executed (know as the body of the loop)

end

Rules:

- The variables in the expression to be tested must have values assigned prior to entering the IF statement.
- The block of code will only execute if the expression is true. If the expression is false, then the code is ignored.
- Values assigned to the variables used in the expression may be changed in the block of code inside the IF statement.

Examples:

```
x = input('Enter x')     % prompt to enter a value for x
if x < 4
  disp ('x is less than 4')
end
```

Enter several values of x, both greater than and less than 4 to see what happens. What do you expect if you entered 3? What about 10?

'for' loops

Purpose: To repeat a statement or a group of statements for a fixed number of times.

Format:

for *variable = expression*

statement(s) to be executed (know as the body of the loop)

end

Rules:

- FOR loops must end with the END statement (lower case).
- The values for the *loop variable* are controlled by the expression
- First time through the loop, the variable is assigned first value in the expression. For the second time, MATLAB *automatically* assigns to the variable second value in the expression. This continues for each value in the expression. The loop terminates after the body of the loop has been executed with the loop variable assigned to each value in the expression.
- The body of the loop is executed many times, but the loop is only executed *once* for each value in the expression.
- The expression in a FOR loop is an array of values. The number of times a loop executes equals the number of values in the expression array.
- After the loop is finished executing, the loop variable still exists in memory and its value is the last value used inside the loop.
- Any name can be used for a loop variable.
  - If the name of the loop variable was already used in the program prior to execution of the loop, old values of the variable are erased and the values of the variable are controlled by the loop.
  - i, j, and k are common loop variables; they should not be used if working with complex numbers.
- Loops can be nested.

Examples: Sequential numbers

```
for i=1:3                    %executes three times
        x=i^2
end
x =
        1
x =
        4
x =
        9
```

Nested (FOR loop inside a FOR loop)

```
for i=1:3                       %executes three times
        i
        for j = 10:10:30
                j
        end
end
i =
        1
j =
        10
j =
        20
j =
         30
i =
        2
j =
        10
j =
        20
j =
        30
i =
        3
j =
        10
j =
        20
j =
        30
```

'while' loops

Purpose: To execute a statement, or a group of statements, for an indefinite number of times until the condition specified by while is no longer satisfied.

Format:

while expression is true

statement(s) to be executed (known as the body of the loop)

end

Rules:

- Must have a variable defined BEFORE the 'while' loop, so you can use it to enter the loop.
- The variable in the while statement must change INSIDE the while loop, or you will never exit the loop.
- After the loop is finished executing, the loop variable(s) still exist in memory and its value is the last value the variable had in the while loop.

Examples: Basic Execution

```
x = 0;
while x <= 100
  x = x+30
end
% The while loop is executed 4 times.
% The values are:
x = 30
x = 60
x = 90
x = 120
% When x = 120, the test (x<=100) failed, so the loop was exited.
As a counter
x = 10;
y = 20;
i = 1;                  %initialize the counter
while(i<= 50)
        x=x+(y^2)               %calculate x
        i=i+1           %increment the counter
end

%This loop will execute exactly 50 times.
-OR-
x = 10;
y = 20;
i = 0;                  %initialize the counter; starting with
          %i=0

while(i < 50)           %test for i<50
        x=x+(y^2)               %calculate x
        i=i+1           %increment the counter
end

%This loop will execute exactly 50 times.
```

2. SOLUTION FOR THE ONE DIMENSIONAL WAVE EQUATION USING EXPLICIT METHOD OF LAX (CODE DEVELOPMENT).

The one dimensional scalar wave equation is given as

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

This equation represents a linear advection process with wave speed c = constant, which is the speed of the travelling wave or the speed of propagation. u(x,t) is the signal or wave information. The wave propagates at constant speed to the right if c > 0 and to the left if c <1. The spatial domain can vary from -∞ to ∞. Suppose the initial conditions are

$$u(x,0) = u_0(x)$$

where $u_0(x)$ is any function. The exact solution to the wave equation then is

$$u = u_o(x - ct)$$

$u_0(x)$ is called the wave shape of wave form. Travelling or propagation here means that the shape of the signal function with respect to x stays constant, however the function is translated left or right with time at the speed c.

Numerical Solution

Method of descretisation – finite difference form

Replace the spatial partial derivative with a central difference expression

$$\frac{\partial u}{\partial x} = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}$$

Where n is the temporal index and j is the spatial

index. Replace the time derivative with a forward

difference formula

$$\frac{\partial u}{\partial t} = \frac{u_j^{n+1} - u_j^n}{\Delta t}$$

We then have

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \qquad (1)$$

Now let us replace u_j^n by an average value between grid points j+1 and j-1 as

$$u_j^n = \frac{u_{j+1}^n + u_{j-1}^n}{2}$$

Substituting this in equation (1) we get the explicit method of Lax for the 1D scalar wave equations as,

$$u_j^{n+1} = \frac{u_{j+1}^n + u_{j-1}^n}{2} - c\frac{\Delta t}{\Delta x}\frac{u_{j+1}^n - u_{j-1}^n}{2}$$

Test Case for the numerical solution

Solve the one dimensional wave equation in the spatial domain of [0, 2*pi] with an initial

step function

condition given by

$U_0(x,0) = 1$ for $x \leq$

pi-1

= 0 otherwise

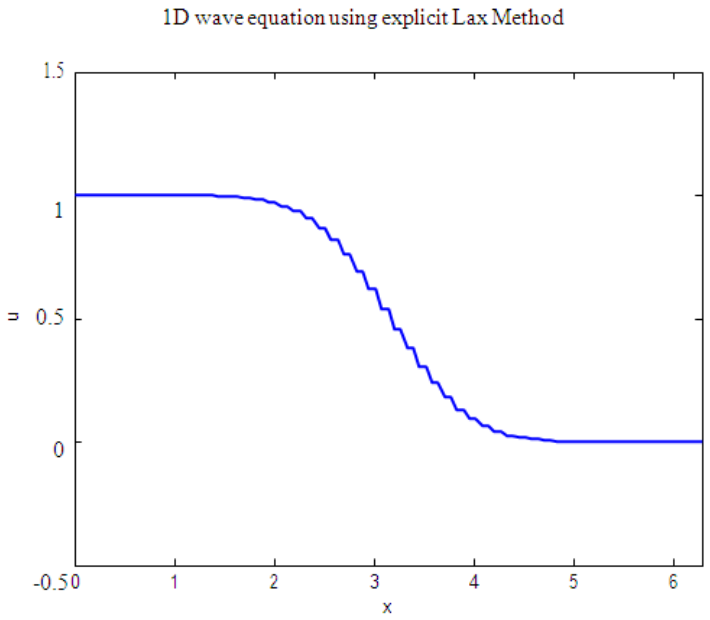
Choose 100 grid points and find the wave form at t = 0.2 s.

Matlab code for the one dimensional wave equation

```
% Solves the one dimensional scalar wave equation du/dt + du/dx = 0
[0,2*pi]
% Using LAX METHOD
clc;
clear all;
t0 = 0;
tf = 1;
M = 100; % number of points in x
direction
N = 100; % number of points in y
direction

% define the mesh in
space
dx =2*pi/M;
x = 0:dx:2*pi;

% define the mesh in
time
dt = (tf-t0)/N;
t = t0:dt:tf;

% calculate value for
lamda
c = 1;
lambda = c*dt/dx
display('lambda should be less than 1 for stability:')

% choose the wave number of the initial data and give its decay rate
 u0 = x<=(pi-1);
u = zeros(M+1,N+1);
u(:,1) = u0;
```

```
% Implement the time marching Lax scheme:
for n=1:N
 for i=2:M
u(i,n+1)=(u(i+1,n)+u(i-1,n))/2-(lambda/2)*(u(i+1,n)-u(i-1,n));
          end
% Introduce exact values at the endpoints.
          u(1,n+1)=1;
         u(M+1,n+1)=0;
end
% plot the result in 21 intervals
for  j=0:20
plot(x,u(:,1+5*j),'LineWidth',2);
axis([0,2*pi,-0.5,1.5]);
title('1D wave equation using explicit Lax Method','FontSize',12)
 xlabel('x');
ylabel('u');
pause(1)
end
%plot(x,u(:,101));
```

Results:



Exercise problems:

2.1 Write a Matlab Program to solve one dimensional wave equation using FTCS method.

2.2 Write a Matlab program to find wave propagation at t=0.5 secs

3. SOLUTION FOR THE ONE DIMENSIONAL TRANSIENT HEAT CONDUCTION EQUATION USING EXPLICIT METHOD (CODE DEVELOPMENT)

The one dimensional transient (unsteady) heat conduction equation is given as

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

Where α is the thermal diffusivity

This equation represents the conduction of heat energy in time and space. Transient nature of this equation is represented in the dependence of temperature with time as opposed to a steady state condition.

Numerical Solution

Method of discretization – finite difference form

Replace the time derivative with a forward difference expression

$$\frac{\partial T}{\partial t} = \frac{T_j^{n+1} - T_j^n}{\Delta t}$$

Where n is the temporal index and j is the spatial index.
Replace the second order spatial derivative on the RHS with a central difference formula

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{j+1}^n + T_{j-1}^n - 2T_j^n}{\Delta x^2}$$

We then have

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \alpha \frac{T_{j+1}^n + T_{j-1}^n - 2T_j^n}{\Delta x^2} \qquad (1)$$

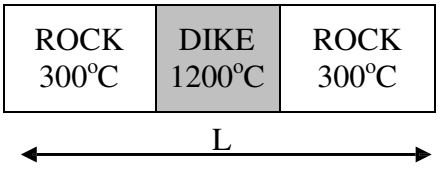
i.e. $$\boldsymbol{T_j^{n+1} = (1 - 2A)T_j^n + AT_{j+1}^n + AT_{j-1}^n} \qquad (2)$$

where $$A = \alpha \frac{\Delta t}{\Delta x^2}$$

Equation (2) is the final explicit update equation for the one dimensional transient heat conduction equation.

Test Case for the numerical solution

A country rock has a temperature of 300°C and the dike a width of 5m, with a magma temperature of 1200°C. Total length of the rock formation is 100m. Initial conditions are temperatures of 300°C and 1200°C for the rock and dike respectively. Boundary conditions at x = -L/2 and x = L/2 are at 300°C (see figure). Find the temperature distribution after 100 days. Use 200 grid points in the x direction with a 1 day time interval.

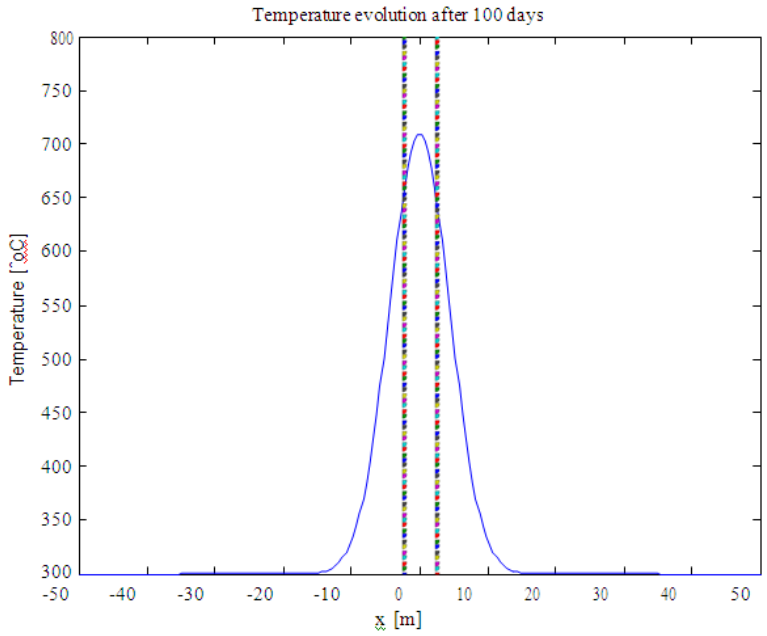


Matlab code for the one dimensional transient heat conduction equation

```
% Solves the 1D heat equation with an explicit finite difference scheme
clear all
clc
%Physical parameters
L = 100; % Length of modeled domain [m]
Td = 1200; % Temperature of magma [^0C]
Tr = 300; % Temperature of country rock [^0C]
kappa = 1e-6; % Thermal diffusivity of rock [m^2/s]
W = 5; % Width of dike [m]
day = 3600*24; % # seconds per day
dt = 1*day; % Timestep [s]
% Numerical parameters
nx = 200; % Number of gridpoints in x-direction
nt = 100; % Number of timesteps to compute
dx = L/(nx-1); % Spacing of grid
 x = -L/2:dx:L/2;% Grid

% Setup initial temperature profile
T = ones(size(x))*Tr;
T(abs(x)<=W/2) = Td;
time = 0;
for n=1:nt % Timestep loop
% Compute new temperature
Tnew = zeros(1,nx);
for i=2:nx-1
Tnew(i) = T(i) + (kappa*dt/(dx)^2)*(T(i+1)-(2*T(i))+T(i-1));
end
% Set boundary conditions
Tnew(1) = T(1);
Tnew(nx) = T(nx);
% Update temperature and time
T = Tnew;
time = time+dt;
end
% Plot solution
plot(x,Tnew);
xlabel('x [m]')
ylabel('Temperature [^oC]')
title(['Temperature evolution after ',num2str(time/day),' days'])

% draw the dike boundaries
x1 = -2.5;
x2 = 2.5;
```

```
y = linspace(300,800);
% Plot the dike
boundaries
hold on
plot(x1,y, x2, y);
```

Results:



Exercise problems:

3.1 Write a Matlab Program to solve one dimensional heat conduction equation using Lax method.

3.2 Write a Matlab program to one dimensional heat conduction equation after 200 days.

4.GENERATION OF THE ALGEBRAIC GRIDS

Problem

Generate an algebraic grid about the upper surface of the airfoil. Points are clustered in j direction near the lower surface (using β=1.05 in algebraic grid). Make sure the number of points in i and j are flexible.

Introduction, Theory, & Formulations

A key component of grid generation is the conversion from the physical domain to the computational domain, in order to allow for equidistant grid lines in rectangular form. In considering a simple two dimensional case, physical coordinates x and y must be converted to computational coordinates ξ and η. These computational coordinates are furthermore known via the rectangular grid relations. As a result, they must be converted back into physical coordinates in order to be of use. For the particular case concerning an airfoil placed on the x axis, the following relationships exist:

$$x = \xi \qquad (1)$$

$$y = H.\frac{(\beta+1)-(\beta-1).(\frac{\beta+1}{\beta-1})^{1-\eta}}{(\frac{\beta+1}{\beta-1})^{1-\eta}+1} \qquad (2)$$

As can be seen, Eq. (1) simply states that the x coordinate is the ξ coordinate, as there exists no irregularities to alter that axis. The precise relationship in Eq. (2) is due to a required clustering near the bottom surface. Here, β represents the clustering parameter, which is given, and H represents the total height along the y axis. However, this does not account for the geometry of the airfoil, wherein its top surface coordinate is a function of the distance along the x axis. The exact equation is:

$$y = \frac{t}{0.2}.\left(0.2969x^{\frac{1}{2}} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \qquad (3)$$

Here, y represents the max height of the airfoil, which would thus be the correspond to y=0 in Eq.(2). Height is determined by subtracting this value from maximum height. This allows a total expression for the grid y coordinative can be obtained. Note that the x used in Eq. (3) assumes 0 at the nose of the airfoil and 1 at the tail. The previous equations effectively define all that is needed to generate an algebraic grid. However, this grid will simply be used as a starting point for the generation of an elliptic grid. Thus, once x and y are obtained algebraically, they will be set as initial conditions for the x and y values used in order to perform iterations of the developed finite difference equations.

MATLAB code for Algebraic Grid Generation

```
%Algebraic Grid Generation
clear all;
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 i direction: 50
Enter the number of grid points in the j direction: 50

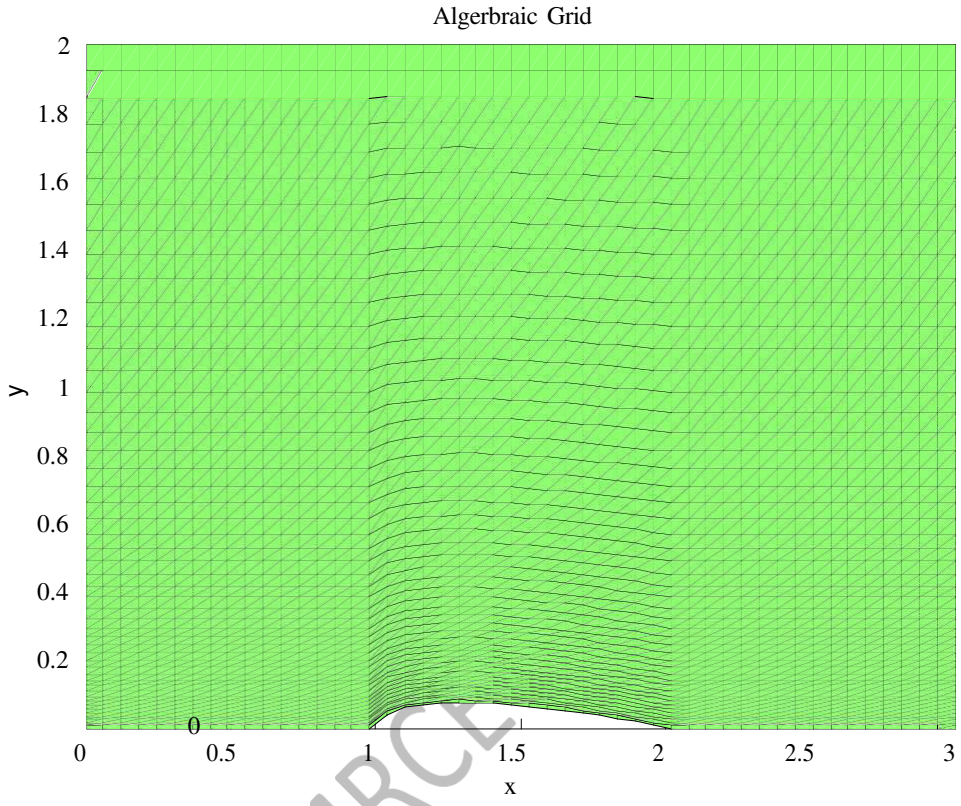


Figure shows the algebraic grid generation with the growth rate β=1.05 the grids are very fine at y=0 and it gets coarser as the y increases.
The value of growth rate β can be varied and you can see the difference in the growth rate of the grid.

Exercise problems:

4.1 Write a Matlab Program to generate algebraic grid over flat plate.

4.2 Write a Matlab program to generate algebraic grid over circle.

5. GENERATION OF THE ELLIPTIC GRIDS

Problem

Starting with an algebraic grid, generate an elliptic grid about the upper surface of the airfoil. Points are clustered in j direction near the lower surface (using β=1.05 in algebraic grid). Make sure the number of points in i and j are flexible.

Using a predetermined algebraic grid, an elliptic grid can be generated in order to fine tune the results for airfoil flow. Coding an algebraic grid necessitates an accounting for the geometry of the airfoil, as well as clustering via appropriate equations. Once these issues are addressed, partial differential equations can be utilized in order to generate an elliptic grid.

Introduction, Theory, & Formulations

A key component of grid generation is the conversion from the physical domain to the computational domain, in order to allow for equidistant grid lines in rectangular form. In considering a simple two dimensional case, physical coordinates x and y must be converted to computational coordinates ξ and η. These computational coordinates are furthermore known via the rectangular grid relations. As a result, they must be converted back into physical coordinates in order to be of use. For the particular case concerning an airfoil placed on the x axis, the following relationships exist:

$$x = \xi \qquad (1)$$

$$y = H.\frac{(\beta+1) - (\beta-1).(\frac{\beta+1}{\beta-1})^{1-\eta}}{(\frac{\beta+1}{\beta-1})^{1-\eta}+1} \qquad (2)$$

As can be seen, Eq. (1) simply states that the x coordinate is the ξ coordinate, as there exists no irregularities to alter that axis. The precise relationship in Eq. (2) is due to a required clustering near the bottom surface. Here, β represents the clustering parameter, which is given, and H represents the total height along the y axis. However, this does not account for the geometry of the airfoil, wherein its top surface coordinate is a function of the distance along the x axis. The exact equation is:

$$y = \frac{t}{0.2}.\left(0.2969x^{\frac{1}{2}} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \qquad (3)$$

Here, y represents the max height of the airfoil, which would thus be the correspond to y=0 in Eq.(2). Height is determined by subtracting this value from maximum height. This allows a total expression for the grid y coordinative can be obtained. Note that the x used in Eq. (3) assumes 0 at the nose of the airfoil and 1 at the tail. The previous equations effectively define all that is needed to generate an algebraic grid. However, this grid will simply be used as a starting point for the generation of an elliptic grid. Thus, once x and y are obtained algebraically, they will be set as initial conditions for the x and y values used in order to perform iterations of the developed finite difference equations.

Two elliptic partial differential equations must be solved in order to fully define the desired grid. In doing this, boundary conditions are required. For this case, x and y values

along the edges of the defined physical domain will be left in place. These being predefined allows all interior coordinates to be developed. The following system of elliptic partial differential equations can be used to define the domain:

$$\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = 0 \qquad (4)$$

$$\frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = 0 \qquad (5)$$

Here, the subscripts denote second order derivative of that variable. Notice that these equations do not express x and y as dependent variables. Rather, they are treated as the independent variables, requiring a transformation. When such a mathematical transformation is preformed Eqs. (4) And (5) become, respectively:

$$a.\frac{\partial^2 x}{\partial \xi^2} - 2.b.\frac{\partial^2 x}{\partial \xi \partial \eta} + c.\frac{\partial^2 x}{\partial \eta^2} = 0 \qquad (6)$$

$$a.\frac{\partial^2 y}{\partial \xi^2} - 2.b.\frac{\partial^2 y}{\partial \xi \partial \eta} + c.\frac{\partial^2 y}{\partial \eta^2} = 0 \qquad (7)$$

Where,

$$a = (\partial x/\partial \eta)^2 + (\partial y/\partial \eta)^2 \qquad (8)$$

$$b = \frac{\partial x}{\partial \xi}.\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi}.\frac{\partial y}{\partial \eta} \qquad (9)$$

$$c = (\partial x/\partial \xi)^2 + (\partial y/\partial \xi)^2 \qquad (10)$$

The previously stated equations must all be expressed in terms of finite differences. Once this is done, x and y at each grid point can be found through iterations. Expanding Equation (8) through (10) explicitly in central space yields:

$$a = \left[\frac{x^n_{i,j+1} - x^{n+1}_{i,j-1}}{2.\Delta\eta}\right]^2 + \left[\frac{y^n_{i,j+1} - y^{n+1}_{i,j-1}}{2.\Delta\eta}\right]^2 \qquad (11)$$

$$b = \left[\frac{x^n_{i+1,j} - x^{n+1}_{i-1,j}}{2.\Delta\xi}\right].\left[\frac{x^n_{i,j+1} - x^{n+1}_{i,j-1}}{2.\Delta\eta}\right] + \left[\frac{y^n_{i+1,j} - y^{n+1}_{i-1,j}}{2.\Delta\xi}\right].\left[\frac{y^n_{i,j+1} - y^{n+1}_{i,j-1}}{2.\Delta\eta}\right] \qquad (12)$$

$$c = \left[\frac{x^n_{i+1,j} - x^{n+1}_{i-1,j}}{2.\Delta\xi}\right]^2 + \left[\frac{y^n_{i,j+1} - y^{n+1}_{i,j-1}}{2.\Delta\xi}\right]^2 \qquad (13)$$

Here, the superscript, n, indexes the iteration, where n is the current iteration and n+1 is the following iteration. These equations are written this way due to the fact that points above and to the right of the point being evaluated are unknown, and, thus, old values must be used. The same procedure of finite differencing can be applied to Eqs. (6) and (7). However, results from these will be of the same form; that is, only the terms x and y will be different. Considering the expansion of Eq. (6) yields:

$$a.\left[\frac{x_{i+1,j}^{n}-2.x_{i,j}^{n+1}+x_{i-1,j}^{n+1}}{(\Delta\xi)^2}\right]-2.b\left[\frac{x_{i+1,j+1}^{n}-x_{i+1,j-1}^{n}-x_{i-1,j+1}^{n}-x_{i-1,j-1}^{n+1}}{4.\Delta\xi.\Delta\eta}\right]+c.\left[\frac{x_{i,j+1}^{n}-2.x_{i,j}^{n+1}+x_{i,j-1}^{n+1}}{\Delta\eta^2}\right]=0 \quad (14)$$

Considering,

$$\alpha=\frac{a}{(\Delta\xi)^2};\ \beta=\frac{b}{2.\Delta\xi.\Delta\eta};\ \gamma=\frac{c}{\Delta\eta^2}$$

This equation can then be explicitly solved for the value $x_{i,j}^{n+1}$ which is the coordinate of interest. Doing so yields:

$$x_{i,j}^{n+1}=\frac{\alpha.(x_{i+1,j}^{n}+x_{i-1,j}^{n+1})+\beta.(x_{i+1,j+1}^{n}-x_{i+1,j-1}^{n}-x_{i-1,j+1}^{n+1}-x_{i-1,j-1}^{n+1})+\gamma.(x_{i,j+1}^{n}+x_{i,j-1}^{n+1})}{2.(\alpha+\gamma)}$$

Similarly, Considering the expansion of Eq.(7) and solving it for value of $y_{i,j}^{n+1}$:

$$y_{i,j}^{n+1}=\frac{\alpha.(y_{i,j+1}^{n}+y_{i,j-1}^{n+1})+\beta.(y_{i+1,j+1}^{n}-y_{i+1,j-1}^{n}-y_{i-1,j+1}^{n+1}-y_{i-1,j-1}^{n+1})+\gamma.(y_{i+1,j}^{n}+y_{i-1,j}^{n+1})}{2.(\alpha+\gamma)}$$

This formula can then be implemented through coding in order to find all values of x. The formulation is exactly the same for the y value. Through code, multiple iterations will occur until convergence is reached; that is, the desired x values will be found once the difference between $x_{i,j}^{n+1}$ and $x_{i,j}^{n}$ is below tolerance and the desired y values will be found once difference between $y_{i,j}^{n}$ and $y_{i,j}^{n}$ falls below said tolerance. These values, when plotted, should produce an elliptic grid that can be utilized to determine flow within the domain containing the airfoil.

MATLAB code for Elliptic Grid Generation

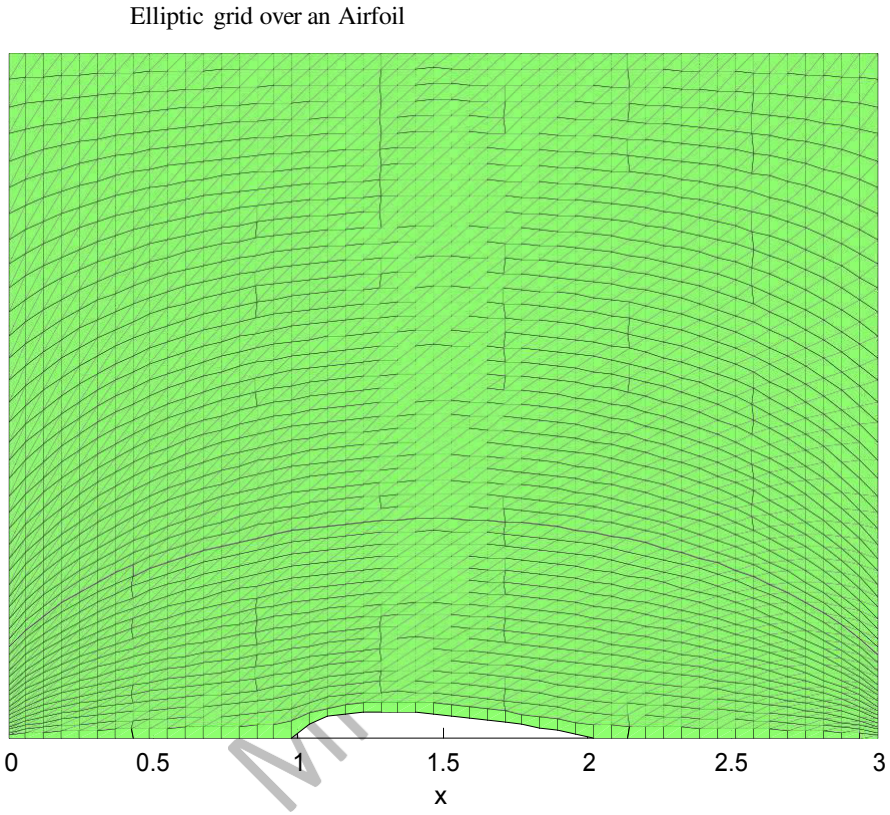
```
%Elliptic Grid Generation
Clear all;
clc;
%Assign values for t and beta
t=0.15;
beta=1.05;
%Prompt user for number of grid points
n=input('Enter the number of grid points in the i direction: ');
m=input('Enter the number of grid points in the j direction: ');
%Create zeroes matrix for surface
plots  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

 %Elliptic initial conditions
 xold=x;
 yold=y;
```

```
%Calculate computational step sizes
delta_eta=1/(m-1);
delta_xi=L/(n-1);
dx=1; %Conditions to start loop
dy=1; %Conditions to start loop
%Assign tolerance value
tol=.0001;
%Nested loop to determine elliptic grid
xdiff=0;
ydiff=0;
count=0;
while dy > tol || dx > tol
for i=2:n-1
for j=2:m-1
a1=(xold(i,j+1)-x(i,j-1))/(2*delta_eta);
a2=(yold(i,j+1)-y(i,j-1))/(2*delta_eta); a=a1^2+a2^2;
c1=(xold(i+1,j)-x(i-1,j))/(2*delta_xi);
c2=(yold(i+1,j)-y(i-1,j))/(2*delta_xi); c=c1^2+c2^2;
b=a1*c1+a2*c2;
alpha=a/delta_xi^2;
beta=-2*b/(4*delta_xi*delta_eta); gamma=c/delta_eta^2; theta=1/(2*alpha+2*gamma);
phi_1=beta*(xold(i+1,j+1)-xold(i+1,j-1)-xold(i-1,j+1)+x(i-1,j-1));
x(i,j)=theta*(alpha*(xold(i+1,j)+x(i-1,j))+gamma*(xold(i,j+1)+x(i,j-1))+phi_1);
xdiff=x(i,j)-xold(i,j)+xdiff;
phi_2=beta*(yold(i+1,j+1)-yold(i+1,j-1)-yold(i-1,j+1)+y(i-1,j-1));
y(i,j)=theta*(alpha*(yold(i+1,j)+y(i-1,j))+gamma*(yold(i,j+1)+y(i,j-1))+phi_2);
  ydiff=y(i,j)-yold(i,j)+ydiff;
  end
end
dx=xdiff; dy=ydiff; xdiff=0;
ydiff=0;
xold=x;
yold=y;
count=c
ount+1;
end
fprintf('The solution took %i iterations to converge. \n \n', count);
surface(x,y,z);
xlabel ('x');
ylabel ('y');
title ('Elliptic grid over an Airfoil');
```

Result and Discussion

In the plot of an Elliptical, the Grid lines have been smoothed out due to the elliptic equations, eliminating extreme jaggedness resulting from the algebraic grid. This would ensure a more accurate flow model.

Enter the number of grid points in the i direction: 50
Enter the number of grid points in the j direction: 50
The solution took 2434 iterations to converge.



Overall, an elliptic grid was shown to provide desired results for discretization. It succeeded in smoothing out otherwise rough edges created through algebraic grid generation. At the same time, the algebraic grid provided a suitable starting point for the generation of the elliptic grid.

INTRODUCTION TO ANSYS

ANSYS ICEM CFD meshing software starts with advanced CAD/geometry readers and repair tools to allow the user to quickly progress to a variety of geometry-tolerant meshers and produce high-quality volume or surface meshes with minimal effort. Advanced mesh diagnostics, interactive and automated mesh editing, output to a wide variety of computational fluid dynamics (CFD) and finite element analysis (FEA) solvers and multiphysics post-processing tools make ANSYS ICEM CFD a complete meshing solution. ANSYS endeavors to provide a variety of flexible tools that can take the model from any geometry to any solver in one modern and fully scriptable environment.

ANSYS ICEM CFD is a popular proprietary software package used for CAD and mesh generation. Some open source software includes OpenFOAM, HeatFlow, Open FVM etc. Present discussion is applicable to ANSYS ICEM CFD software.

It can create structured, unstructured, multi-block, and hybrid grids with different cell geometries.

GEOMETRY MODELLING

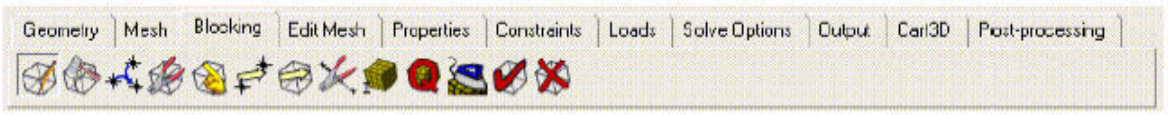
ANSYS ICEM CFD is meant to mesh a geometry already created using other dedicated CAD packages. Therefore, the geometry modelling features are primarily meant to 'clean-up' an imported CAD model. Nevertheless, there are some very powerful geometry creation, editing and repair (manual and automated) tools available in ANSYS ICEM CFD which assist in arriving at the meshing stage quickly. Unlike the concept of volume in tools like GAMBIT, ICEM CFD rather treats a collection of surfaces which encompass a closed region as BODY. Therefore, the typical topological issues encountered in GAMBIT (e.g. face cannot be deleted since it is referenced by higher topology) don't show up here. The emphasis in ICEM CFD to create a mesh is to have a 'water-tight' geometry. It means if there is a source of water inside a region, the water should be contained and not leak out of the BODY.

Geometry Menu

The **Geometry** tab contains the following options: Create Point
Create/Modify Curve
Create/Modify Surface
Create Body
Create/Modify Faceted
Repair Geometry
Transform Geometry
Restore Dormant Entities
Delete Point

Delete Curve Delete
Surface Delete Body
Delete Any Entity

Blocking Menu



The **Blocking** tab contains the following options to create blocking over any geometry.
Create Block
Split Block
Merge Vertices
Edit Block
Associate
Move Vertex
Transform Blocks
Edit Edge
Pre-Mesh Params
Pre-Mesh Quality
Pre-Mesh Smooth
Block Checks
Delete Block

Mesh Menu

The **Mesh** tab contains the following options:
Global Mesh Setup
Part Mesh Setup
Surface Mesh Setup
Curve Mesh Setup
Create Mesh Density
Define Connectors
Mesh Curve
Compute Mesh

Numerical simulation of the following flow problems using commercial software packages:

6. FLOW OVER AN AEROFOIL AIM: To simulate flow over NACA 0012 airfoil

Problem description:

Consider air flowing over NACA 0012 airfoil. The free stream velocity is 40 m/s .

Assume standard sea-level values for the free stream properties:

Pressure = 101,325 Pa

Density = 1.2250 kg/m^3

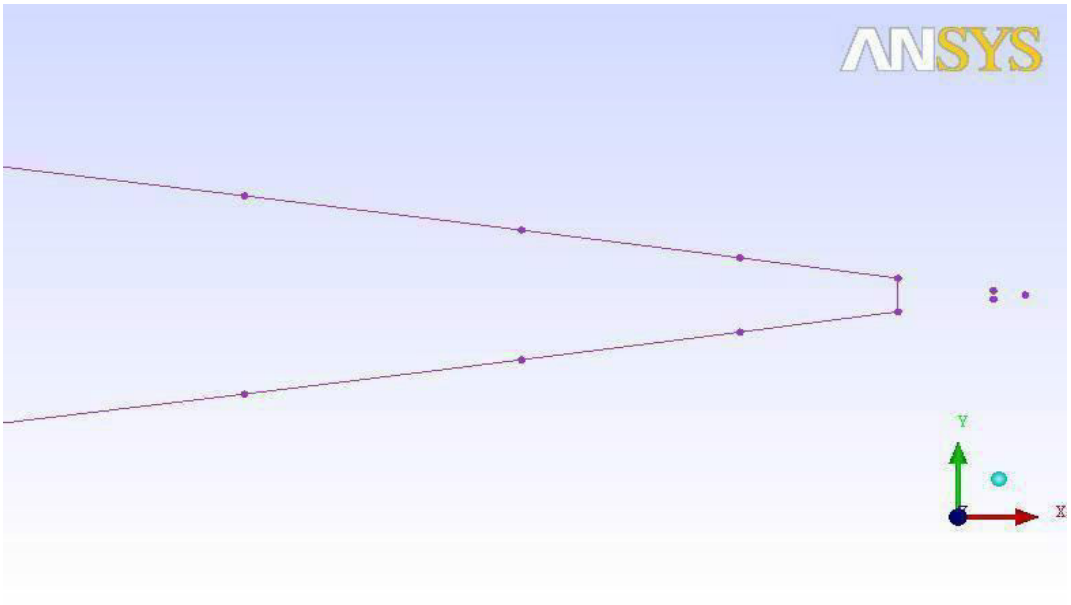
Temperature = 288.16 K

Kinematic viscosity v = 1.4607e-5 m^2/s

Steps Involved In ICEM

CFD: Creation of Geometry in ICEM CFD:

- 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:

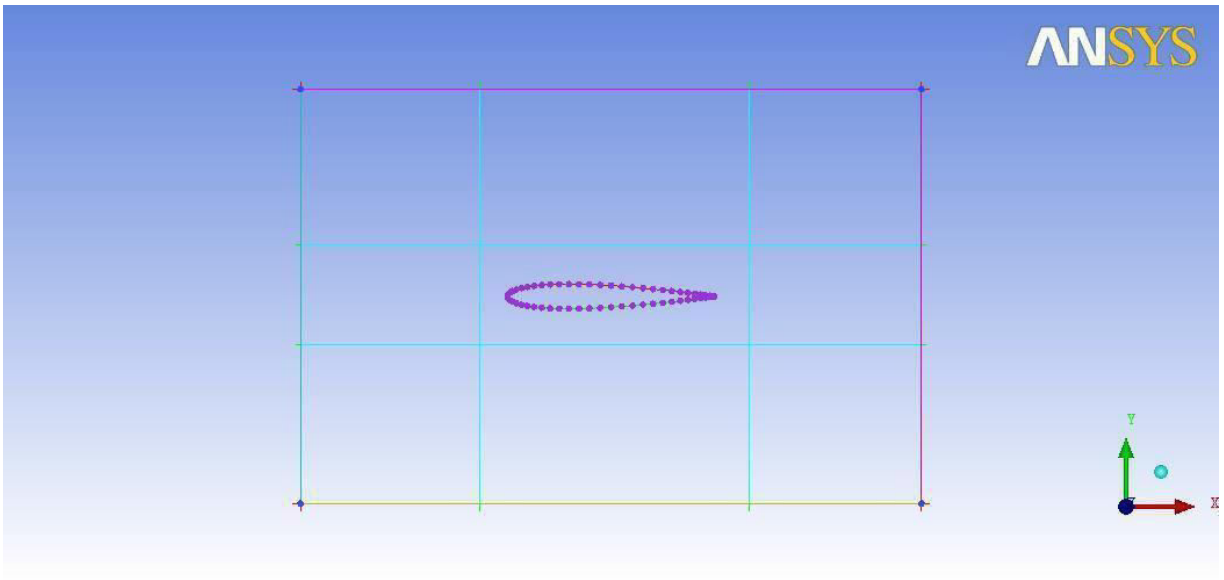
- Create points (-1,1),(-1,-1),(2,1),(2,-1)
- Join these points
- Create parts as Inlet, Outlet, Top & Bottom
- Geometry→Create/Modify surface→Simple surface→Select all the lines of domain→ok
- Create the new part as: Surface

3) Saving the Geometry:

- File→Change working directory→Choose the folder
- File→Geometry→Save Geometry as→Give the name.

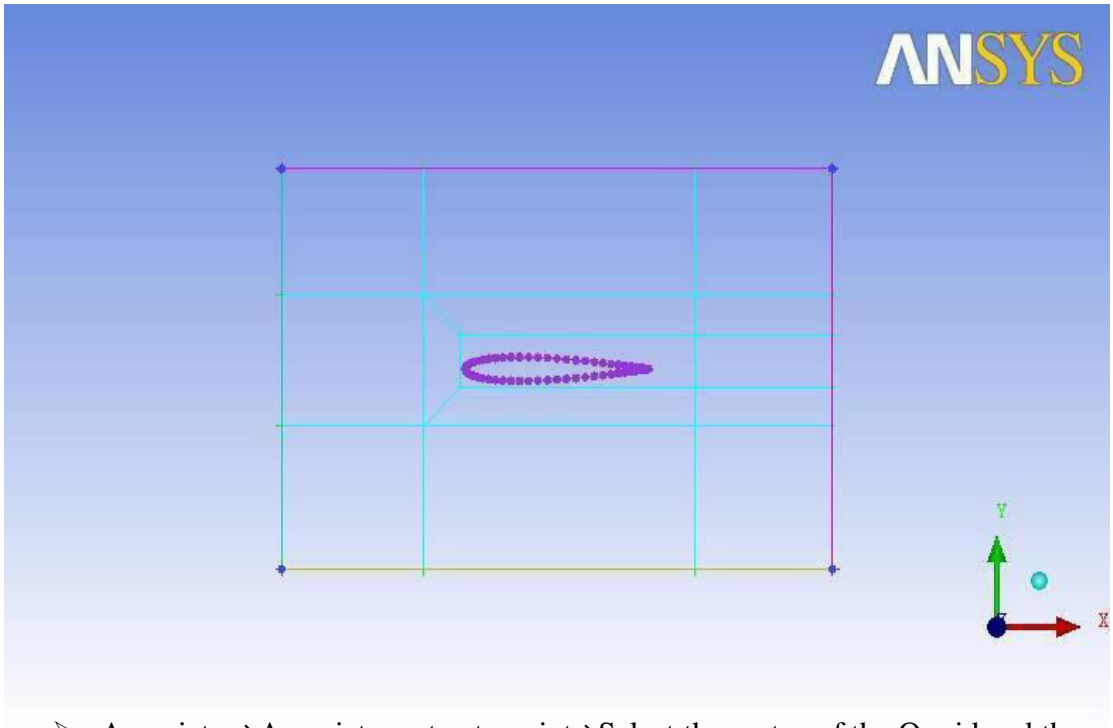
4) Creation of Blocking and Association:

- 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

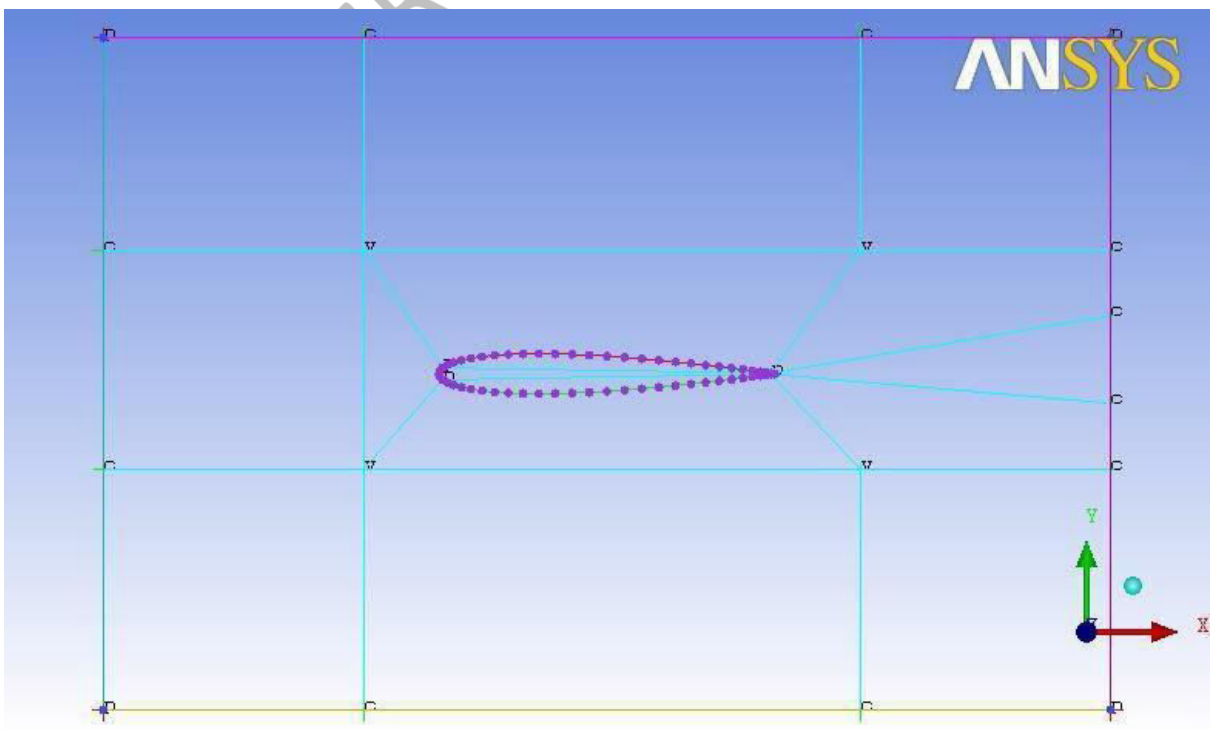


- Split block→O grid →Select edges→Select last 2 edges in the middle row→ok→Select blocks→Select last 2 blocks in the middle row→ok

Thus the O grid has been generated as shown in below fig.



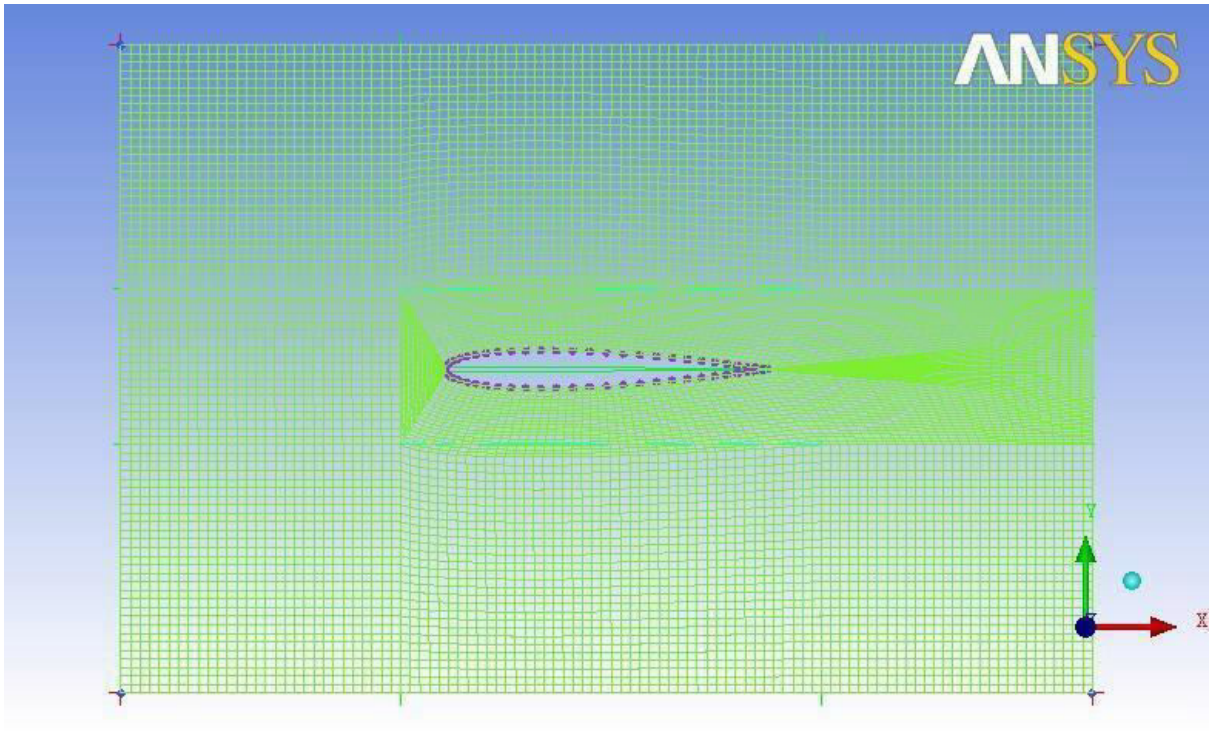
- Associate →Associate vertex to point→Select the vertex of the O grid and the 2nd point on the upper curve(suction) →ok
- Similarly associate remaining 3 vertices of the O grid to the points on the aerofoil as shown in the below fig.



- 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

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→ inviscid
- Materials→Air
- Cell zone conditions→Type as: fluid→ok
- Boundary conditions→Select inlet→Edit→Give velocity magnitude as: 40 m/s.
- Boundary conditions→Select outlet→Edit→Give gauge pressure as: 0 Pa
- Monitors →Drag and Lift → select suction, pressure, TE parts→plot.

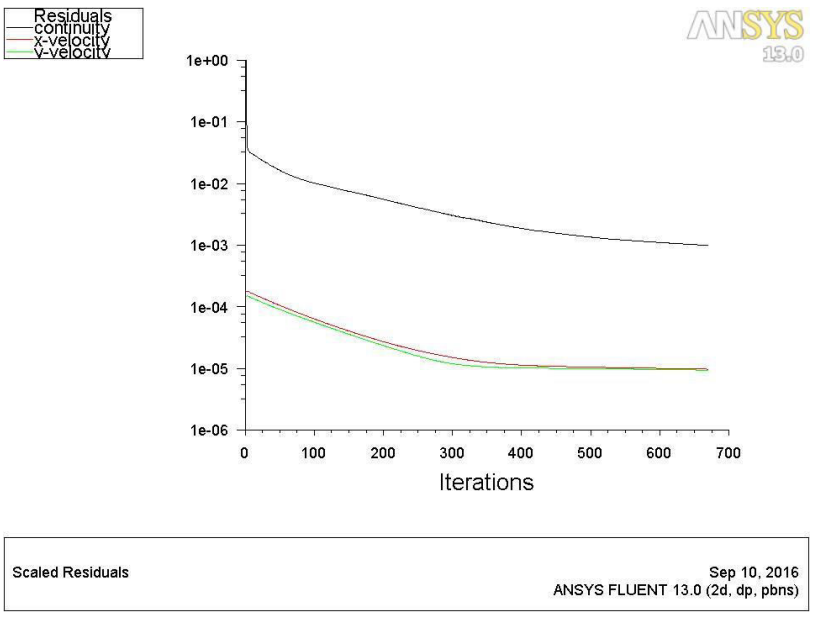
10) Solution:

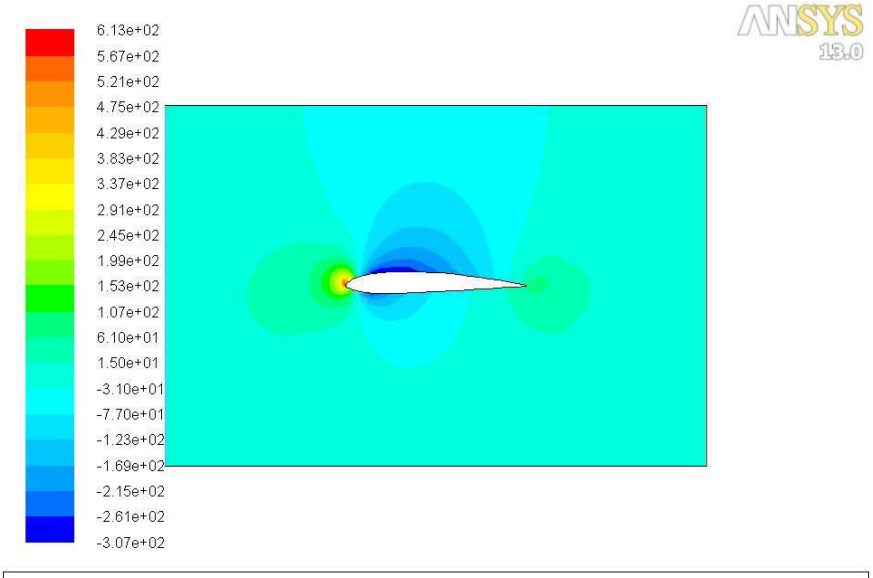
- Select the required monitors
- Solution initialization→Compute from: inlet→Initialize
- Run calculations→Enter the no. of iterations as: 1000→Calculate

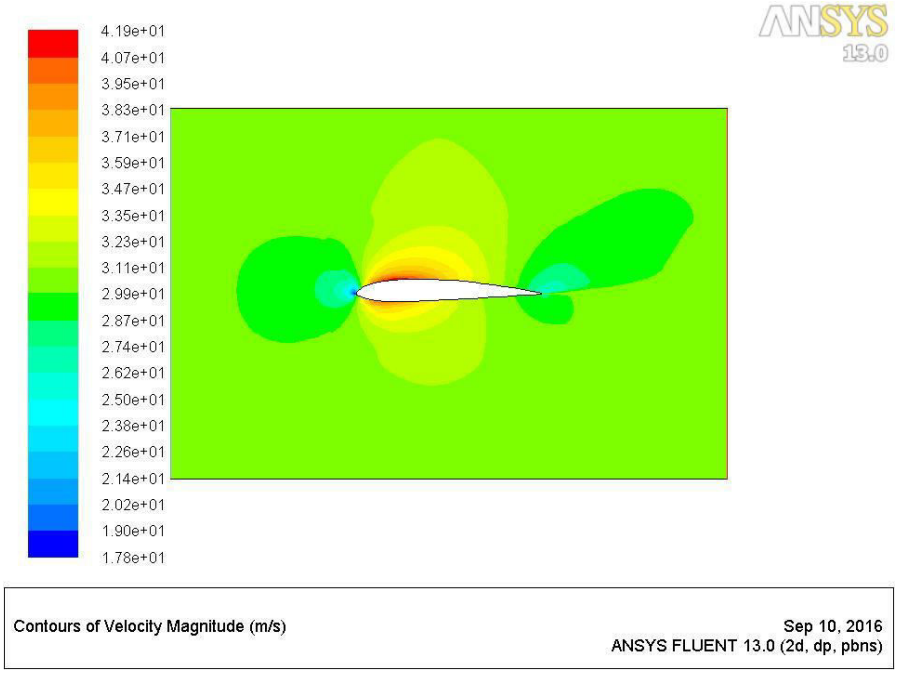
11) Results:

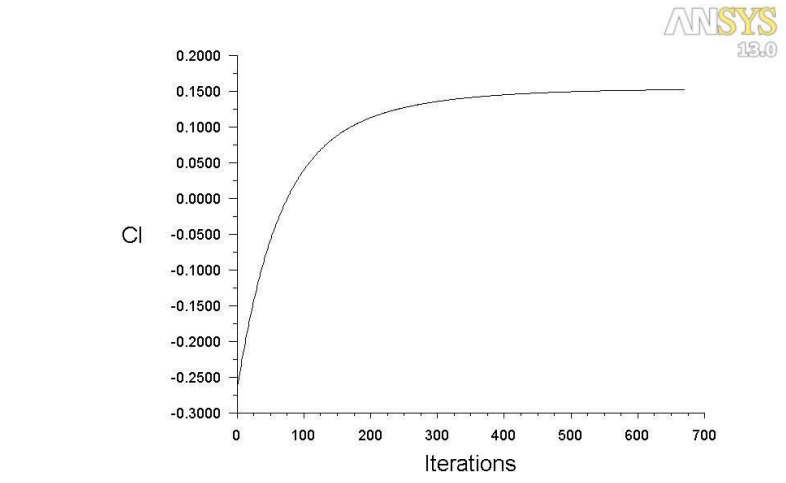
- Graphics and animations→select the required flow parameters in the contours and vectors.

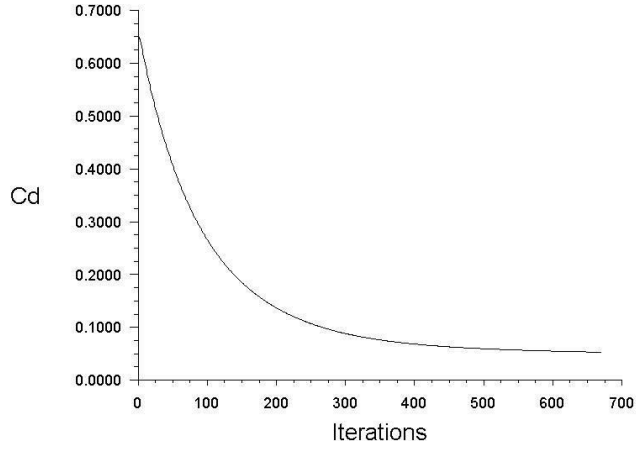
  The results are shown below as:

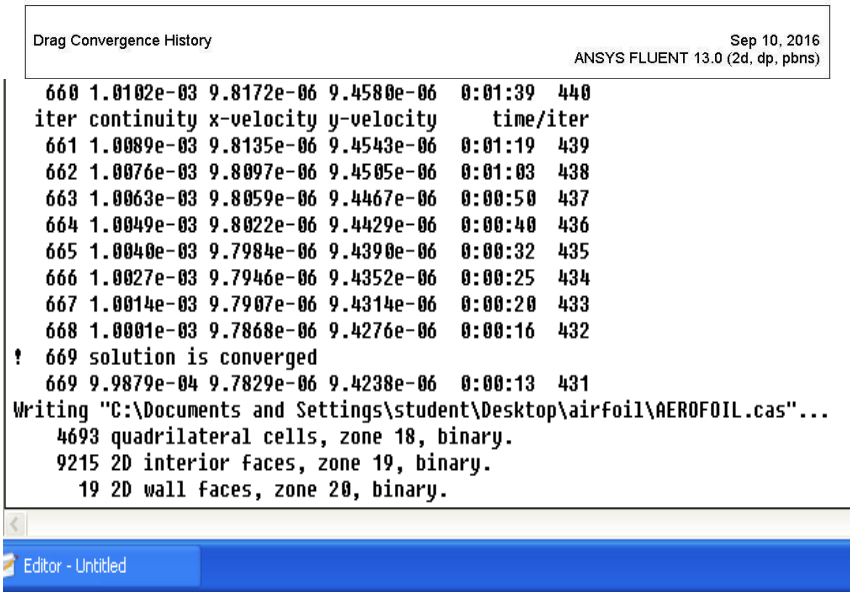



ANSYS
13.0
6.13e+02
5.67e+02
5.21e+02
4.75e+02
4.29e+02
3.83e+02
3.37e+02
2.91e+02
2.45e+02
1.99e+02
1.53e+02
1.07e+02
6.10e+01
1.50e+01
-3.10e+01
-7.70e+01
-1.23e+02
-1.69e+02
-2.15e+02
-2.61e+02
-3.07e+02
Contours of Static Pressure (pascal)
Sep 10, 2016
ANSYS FLUENT 13.0 (2d, dp, pbns)


ANSYS
13.0
4.19e+01
4.07e+01
3.95e+01
3.83e+01
3.71e+01
3.59e+01
3.47e+01
3.35e+01
3.23e+01
3.11e+01
2.99e+01
2.87e+01
2.74e+01
2.62e+01
2.50e+01
2.38e+01
2.26e+01
2.14e+01
2.02e+01
1.90e+01
1.78e+01
Contours of Velocity Magnitude (m/s)
Sep 10, 2016
ANSYS FLUENT 13.0 (2d, dp, pbns)


ANSYS
13.0
0.2000
0.1500
0.1000
0.0500
0.0000
-0.0500
-0.1000
-0.1500
-0.2000
-0.2500
-0.3000
Cl
0
100
200
300
400
500
600
700
Iterations
Lift Convergence History
Sep 10, 2016
ANSYS FLUENT 13.0 (2d, dp, pbns)





Exercise Problems:

6.1 Evaluate aerodynamic characteristics of aerofoil at 15 degrees angle of attack.

6.2 Perform the grid independence study over airfoil.

7. SUPERSONIC FLOW OVER A WEDGE

Problem description:

Consider air flowing over wedge. The free stream Mach number is 3 and the angle of attack is 5°. Assume standard sea-level values for the free stream properties:

Pressure =101,325Pa

Density =1.2250kg/m^3

Temperature =288.16K

Kinematic viscosity v = 1.4607e-5 m^2/s

Steps Involved In ICEM CFD:

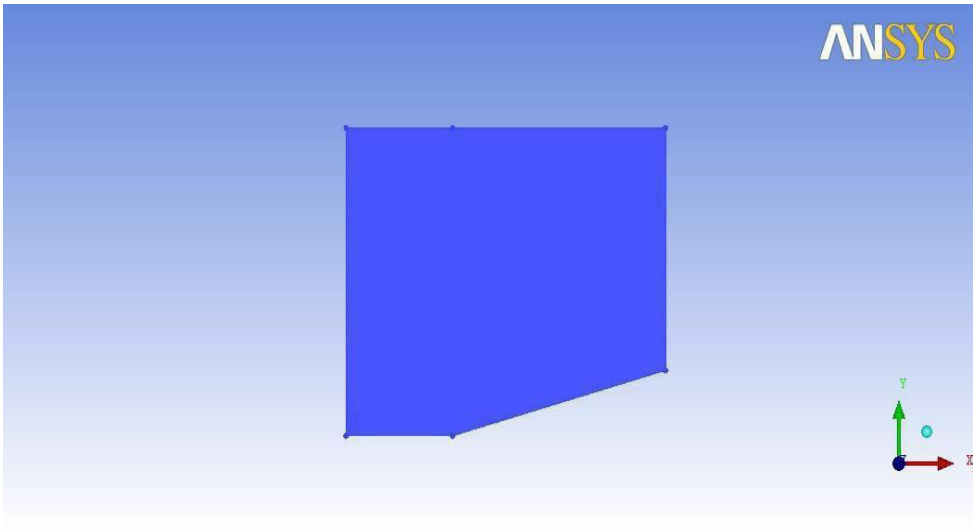
1) Creation of Geometry in ICEM CFD:

a. Geometry→Create point→Explicit coordinates→Enter the coordinates as given in table shown:

X	0	0	0.5	1.5	1.5	0.5
Y	0	1.259	1.259	1.259	0268	0
Z	0	0	0	0	0	0

Geometry→Create/modify curve→From points→Select any 2 points→ok→Similarly create the curves to all points

b. Geometry→Create/Modify surface→Simple surface→Select all the lines of domain→ok



2) **Creation of parts:**

   a. Parts in the tree→Right click→Create part→

      Select Left curve: **Inlet**

      Select Right curve: **Outlet**

      Select Top curve: **Top**

      Select inclined curve: **Wedge**

      Select bottom curve: **Front_wedge**

3) **Saving the Geometry:**

   a. File→Change working directory→Choose the folder

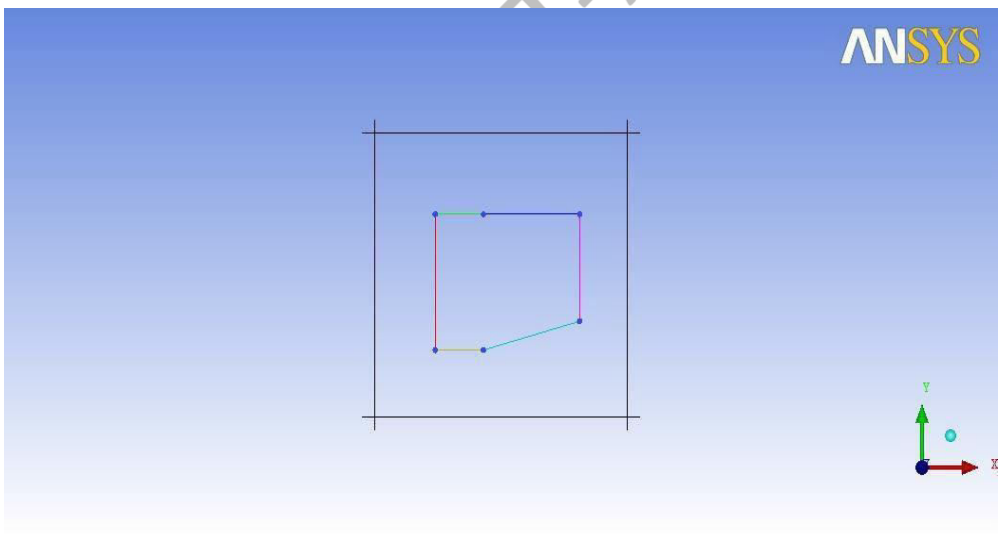
   b. File→Geometry→Save Geometry as→Give the name.

4) **Creation of Blocking and Association:**

   a. Blocking→Create block→Initialize blocks→Type as:2D Planar→ok

   b. Associate→Associate vertex to point→Select a vertex and a point→Apply→ Similarly associate remaining 3 vertices to points

   c. Associate→Associate edge to curve→Select a edge and a curve →Apply→ Similarly associate remaining 3 edges to 5 curves



5) **Generation of Mesh:**

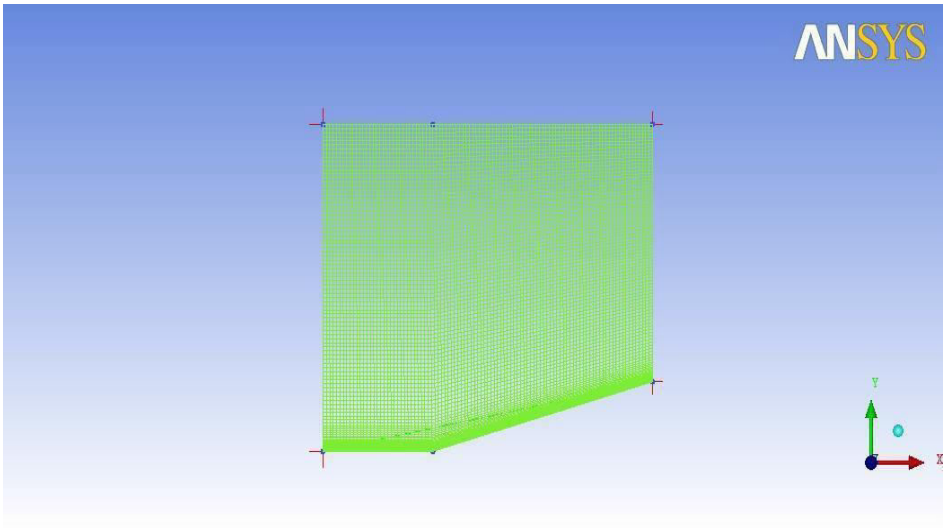
   a. Pre-mesh parameters→Edge parameters→Switch ON the Copy Parameters→ Select the Horizontal edge and give no. of nodes as: 100→ok

   b. Pre-mesh parameters→Edge parameters→Switch ON the Copy Parameters→ Select the Vertical edge and give no. of nodes as: 100→Spacing as: 0.001→Ratio as: 1.1→ok

c. Switch ON Pre-mesh in the tree→click yes to compute the meshing

d. Pre-mesh→Right click→Convert to unstructured mesh

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



6) Saving the Project:

a. File→Save Project as→Give the name.

7) Writing output file:

a. Output→Select solver→Output solver as: **Fluent_V6**→Common Structural solver as: ANSYS→ok

b. Write input→Click NO→Open the file→**Click 2D**→ok

Steps Involved in Fluent:

8) Importing the mesh file:

a. File→Read→mesh→Choose the output file written in ICEM CFD

Now the mesh has imported into the fluent solver.

9) Problem setup:

a. General→Type as: Density based

b. Models→Energy ON→Select Viscous-laminar→Edit→Set model as: k-omega(2 equ)

c. Materials→Air→Create/Edit→Set density as: Ideal-gas→Set viscosity as: Sutherland→Change

d. Cell zone conditions→Type as: fluid→Set operating conditions→Set operating pressure as: 0Pa

e. Boundary conditions→Select inlet→Give type as: pressure-far-field→Edit→Give Gauge pressure as: 101325Pa→Set Mach as: 3→ok

f. Boundary conditions→Select outlet→Edit→Give gauge pressure as: 0Pa

10) Solution:

a. Select Solution Controls→Set Courant number as: 1

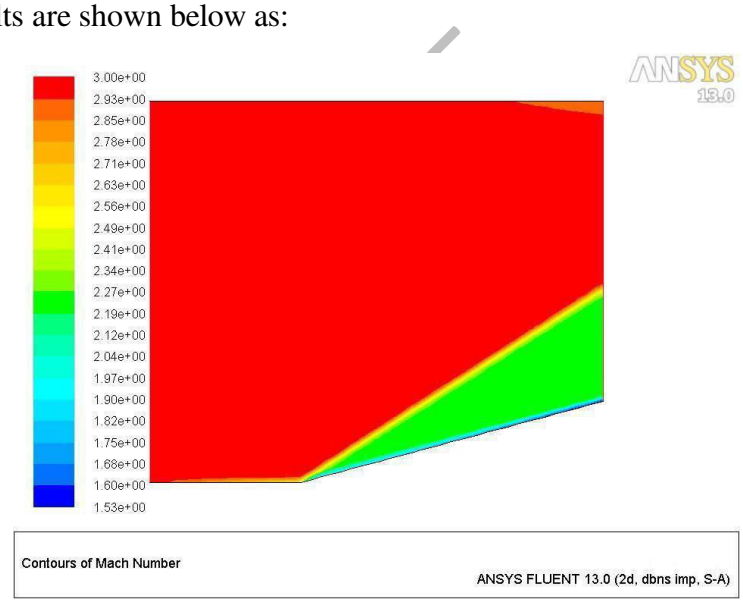
b. Select the required monitors

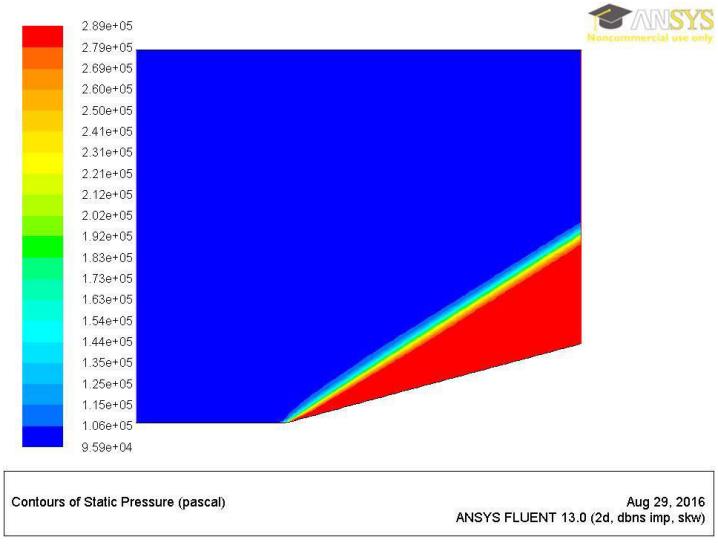
c. Solution initialization→Compute from: inlet→Initialize

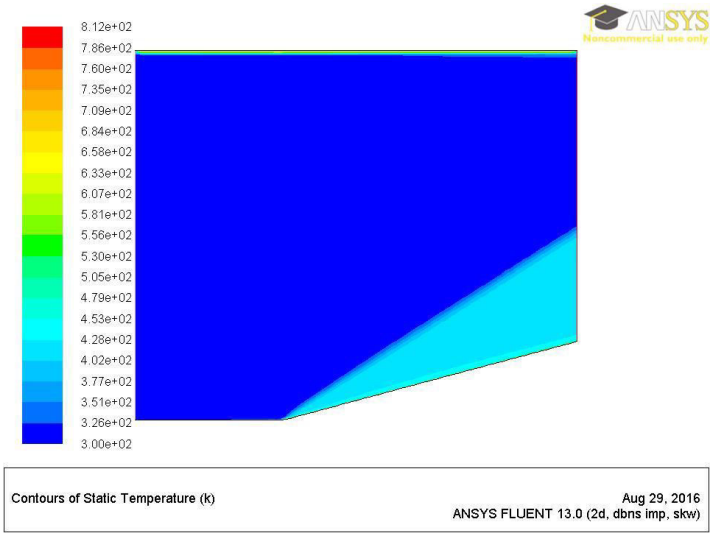
d. Run calculations→Enter the no. of iterations as: 1000→Calculate

11) Results:

a. Graphics and animations→Select the required flow parameters in the contours. The results are shown below as:







Exercise Problems:

7.1 Evaluate flow properties across oblique shock of wedge at 15 degrees angle of attack.

7.2 Compare the results with analytical formulas.

8. FLOW OVER A FLAT PLATE

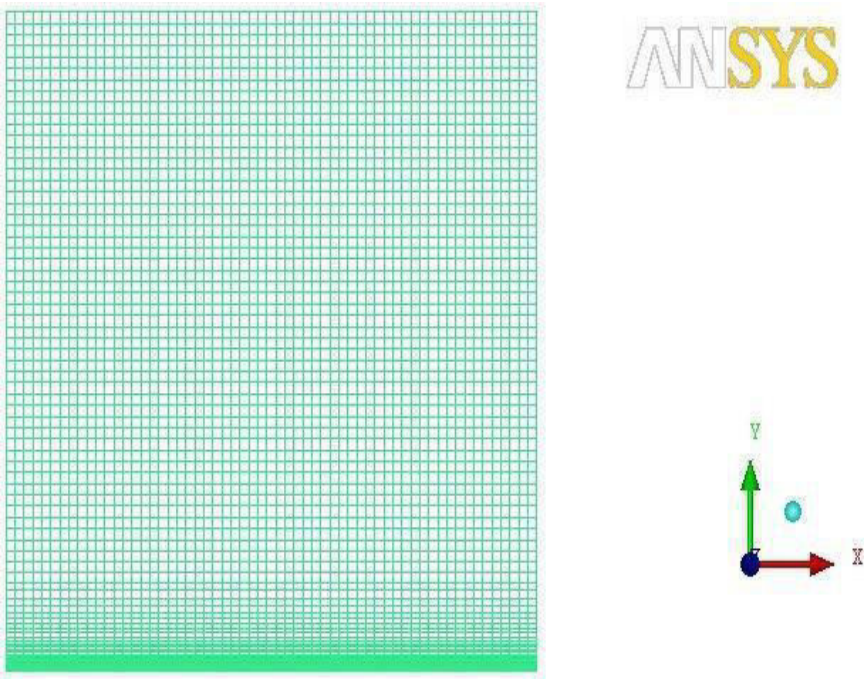
Aim: To study the characteristics of flow over a flat plate

Description: Consider a plate of 1m and the flow of air is 0.00133 m/s. The plate is an stationary solid wall having no slip as its boundary condition.

Procedure:

- Geometry→ create point→ explicit coordinates→ 1(0,0,0), 2(1,0,0), 3(1,1,0) and 4(0,1,0) → ok
- Create/modify curve→ select 2 points→ middle click
- Select all points to make a rectangle

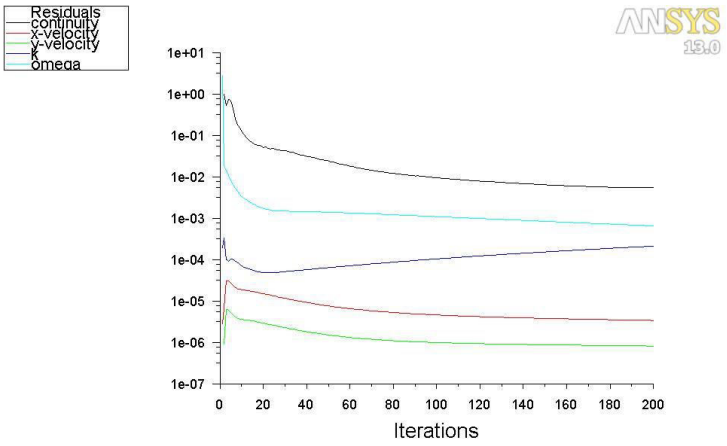
- Create/modify surface→ select all the lines→ surface is created
- Create part→ name inlet→ select the left edge→ middle click
- Similarly create outlet, top and bottom
- Switch off points and curves→ create part→ name surf→ click on surface→ ok
- Blocking→ create block→ select entities→ click spectacles→ middle click→ switch on points and curves
- Go to association→ associate vertex→ select the point→ double click on the point
- Associate→ edge to curve→ select the edge→ ok→ again select the edge→ ok
- Similarly for the remaining edges
- Premesh parameters→ edge parameters→ select any edge→ click on copy parameters→ nodes-60, spacing-0.01, ratio-1.1→ ok
- Blocking tree→ premesh→ right click→ convert structured to unstructured mesh

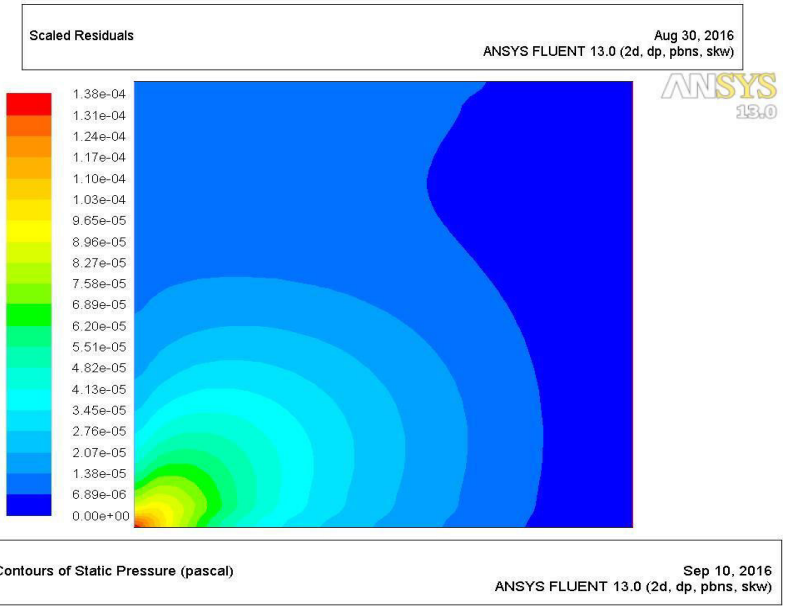


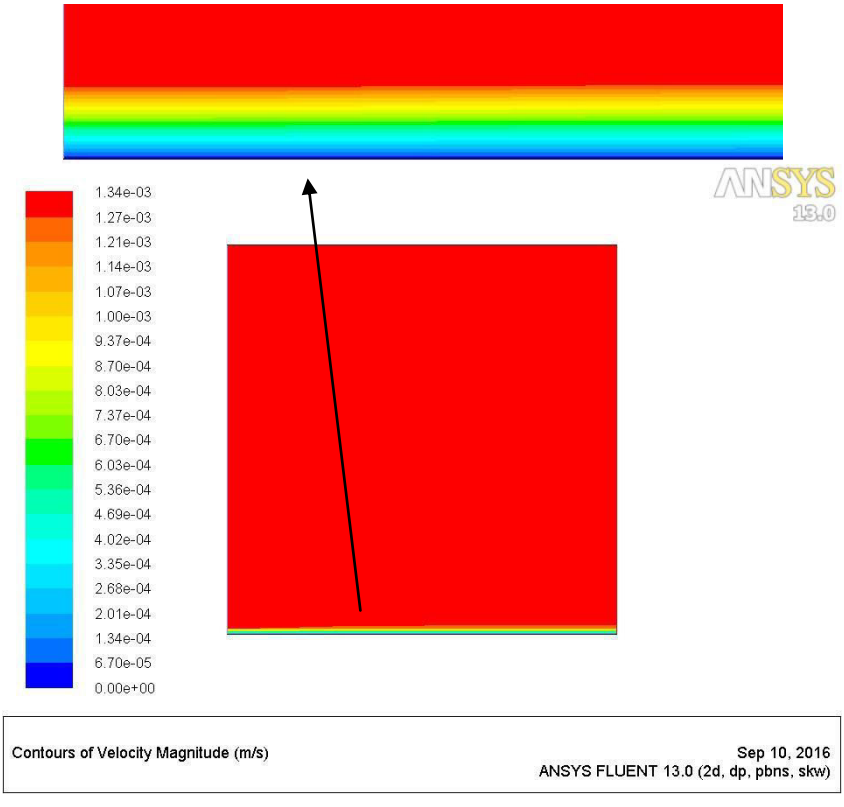
- Change the working directory
- output→ output solver→ fluent V6→ common-ansys→ ok

FLUENT:

- Folder→ general→ mesh→ fluent mesh→ ok
- Click on check→ done
- Models→ viscous laminar→ materials→ air
- Cell zone conditions→ solid→ ok
- Boundary conditions→ bottom→ edit→ stationary wall→ ok, inlet→ velocity-0.00133→ ok, outlet→ guage pressure-0→ ok, top→ edit→ moving wall→ ok
- Dynamic mesh→ solution→ solution method-simple, solution controls-0.3,1,0.3→ ok
- Monitor initializer→ compute from inlet→ x=0.00133→ initialize
- Calculation activities→ no of iterations-200→ run calculations→ click on calculate→ ok
- Results→ graphics and animations→ contour→ set up→ display options→ filled→ display
  Contour→ velocity→ display







8.1 Find out the effect of viscosity of water on flat plate.
8.2 Find the material of the plate on velocity profile.

9. LAMINAR FLOW THROUGH PIPE

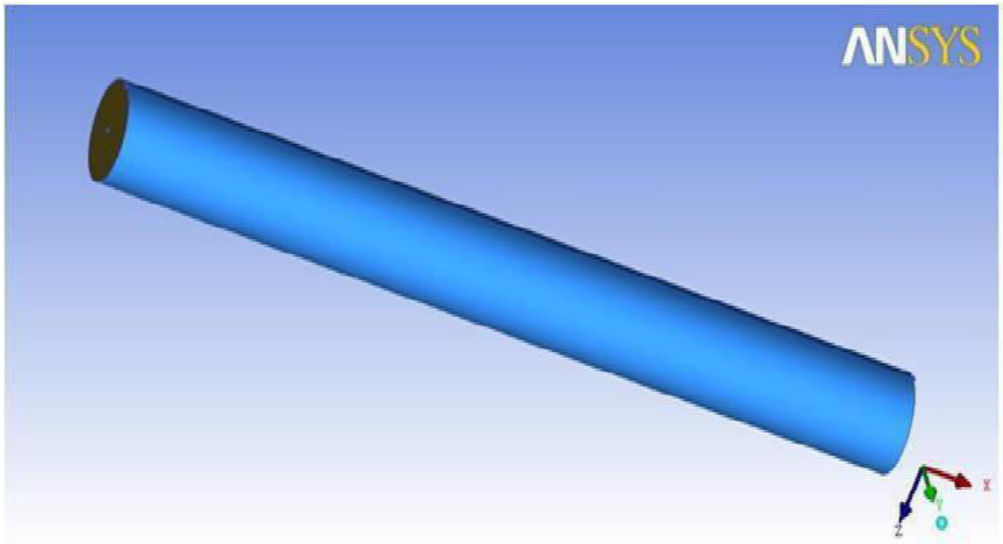
AIM: To study characteristics of laminar flow through a pipe.

DESCRIPTION: Consider a pipe of radius 0.05 and 1 mt length. The freestream velocity considered is 40m/s.

STEPS INVOLVED:

1) **Create A Geometry:**
   a) **Create a point:** Geometry →create point →explicit coordinates→(X, Y, Z) = (0, 0, 0) →apply→(X, Y, Z) = (1, 0, 0) →ok.

   b) **Create a pipe:** Geometry→Create/modify surfaces→Standard shapes→ cylinder→radius1=radius2 = 0.05→select the 2 points →ok



2) **Generation of parts:**

   - Part →create part→inlet→select inlet→ok.
   - Part →create part →outlet →select outlet →ok.
   - Part →create part →pipe →select pipe without inlet and outlet →ok.

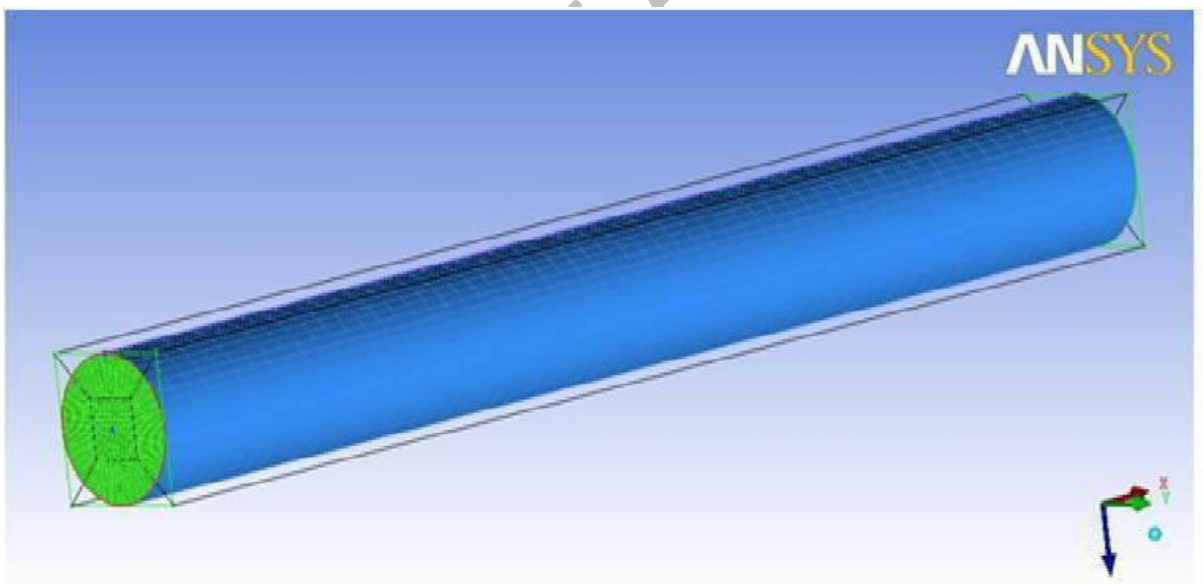
3) **Generation of blocking:**

   - Blocking →create block →solid →select pipe element with inlet and outlet→ apply→ok.
   - Blocking →associate →edge to curve →select the 4 edges of the blocking at inlet →apply →select the inlet curve→ok.

- Blocking→associate →edge to curve→select the 4 edges of the blocking at outlet → apply→select the outlet curve→ok.
- Associate→faces to surface→select inlet face → apply →select as inlet→accept →ok.
- Associate →faces to surface →select outlet face →apply →select as outlet →accept →ok.
- Associate →face to surface →select pipe faces →apply →select as pipe → accept →
- Blocking →split block →O grid block →select the 2 faces (inlet & outlet) →apply →ok.

4) Generation of Meshing:

- Blocking →pre-mesh parameters →edge parameters →switch on the copy parameters →select 1 edge →give no. of nodes =20→ok.
- Repeat the above steps to the remaining edges also and then apply.
- Blocking →pre-mesh →compute.



- Blocking →pre-mesh →convert to unstructured mesh.

5) Generation of Solver file :

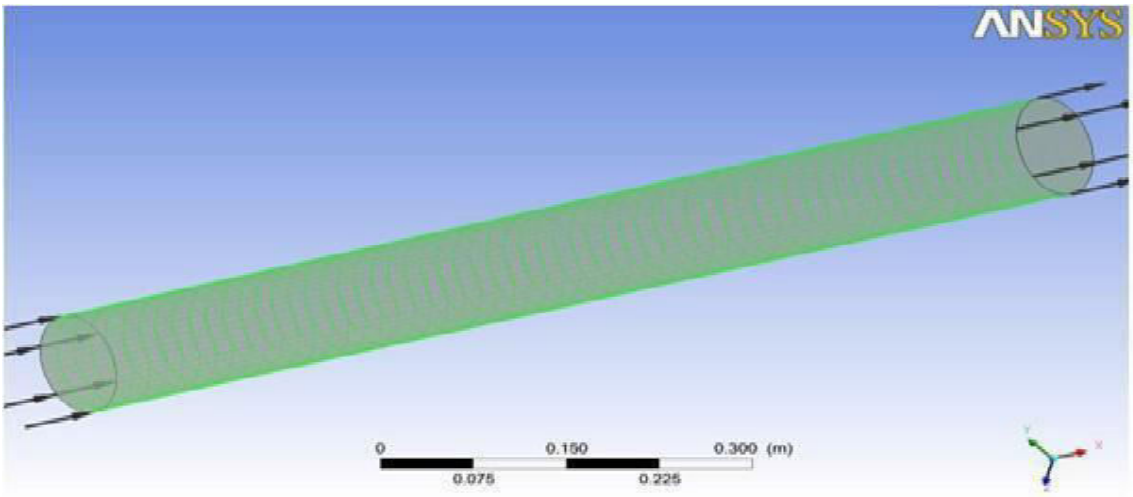
- Output→select solver→ansys.cfx→ANSYS→ok.
- Output →write input →done →check the file is saved folder →ok.

6) Solution in CFX Solver:

- Start →programs →ANSYS →fluid dynamics →CFX →ok.
- Change the working directory (where ICEM CFD mesh file was saved)→Click on CFX-PRE.

7) CFX-PRE:

- File→new case→general→ok.
- File→import→mesh→select the meshed filed→ok.
- Boundary→Boundary1→Boundary type as: inlet→location as: inlet→boundary conditions as: velocity=40m/s→ok.
- Boundary→Boundary2→boundary type as: outlet→location as: outlet→boundary conditions as: static pressure=0→ok.
- Boundary→Boundary3→boundary type as: wall→location as: pipe→boundary conditions as: no slip condition, smooth wall→ok.
- Domain→basic settings→location as: solid→domain type as: fluid domain→material as: air→ok.
- Solver control→basic settings→ max.itterationsas:1000→residual target as: 0.000000001→ok.
- Write solver input file→give the name of the file→ok.



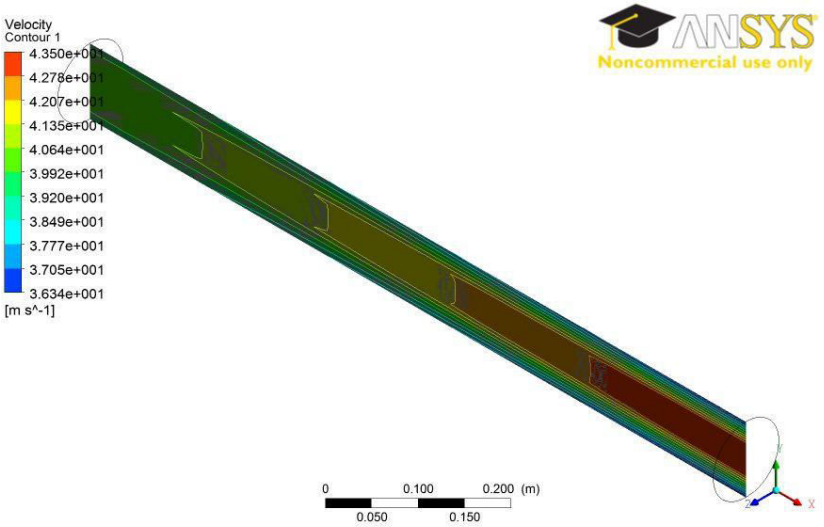
8) CFX-Solver Manager:

- File→Define run→Solver input file→Select the file→Start run.

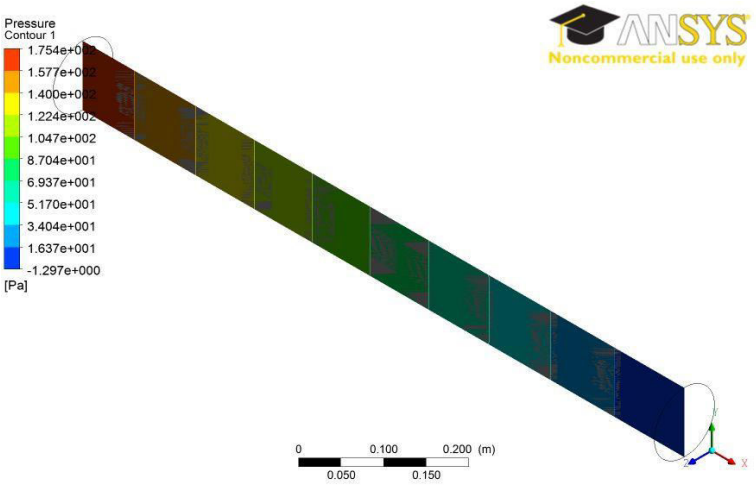
9) CFD POST

- Select →Location →Plane →XY Plane → Apply
- Contours →Location → plane→Variable→ Velocityand pressure →Apply.

Incoming flow through inlet



pressure contour



Exercise problems

9.1 Find out the effect convergent section on velocity.

9.2 Find the minor losses in a bend pipe.

10.FLOW PAST OVER A CYLINDER

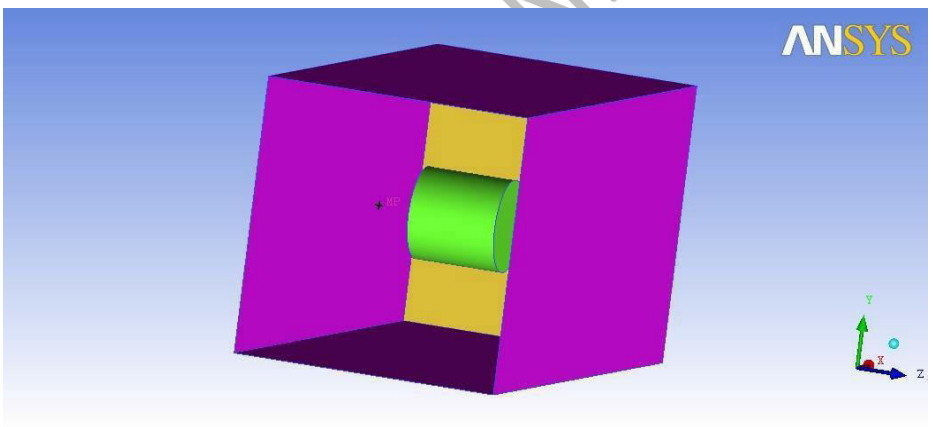
AIM : To study the characteristics of flow over a cylinder.

DESCRIPTION: Consider a cyclinder of 3m radius and 6m height. The free stream velocity considered 20m/s. the properties of air is ρ=1.18kg/m^3.

PROCEDURE:

CREATION OF GEOMETRY:

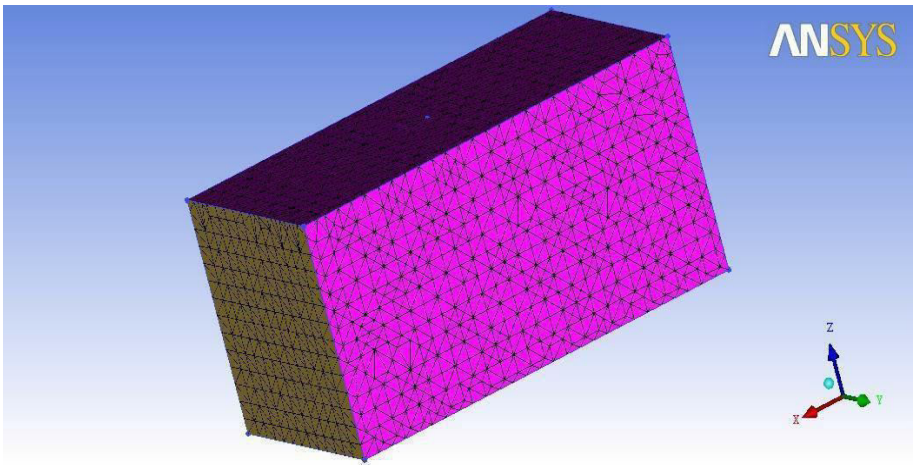
- Geometry → create point → explicit coordinates → (0,0,0)
- Geometry → create surface → standard shapes → box → (36 18 18) → apply → solid simple display
- Geometry → create point → based on 2 locations → select 2 diagonal points of face
- Geometry → transform geometry → copy → select point → Z-offset =6 → apply → z-offset=12→ ok.
- Geometry → surfaces → standard shapes → cylinder r1=3.r2=3 → select 2 points of cylinder → apply



CREATION OF PARTS AND MESH GENERATION:

- parts→ create parts → (part name) → select entities → middle click (create parts according to the problem i.e. inlet, outlet, cylinder & free slip wall)
- geometry → solid → part(mp) → select two points lying outside the cylinder → apply.
- Mesh → mesh parameters → cylinder -1.5, inlet-2.5, outlet-2.5, slipfree-0.7 Mesh →
- global mesh setup → global mesh size → max element size (3) → apply.

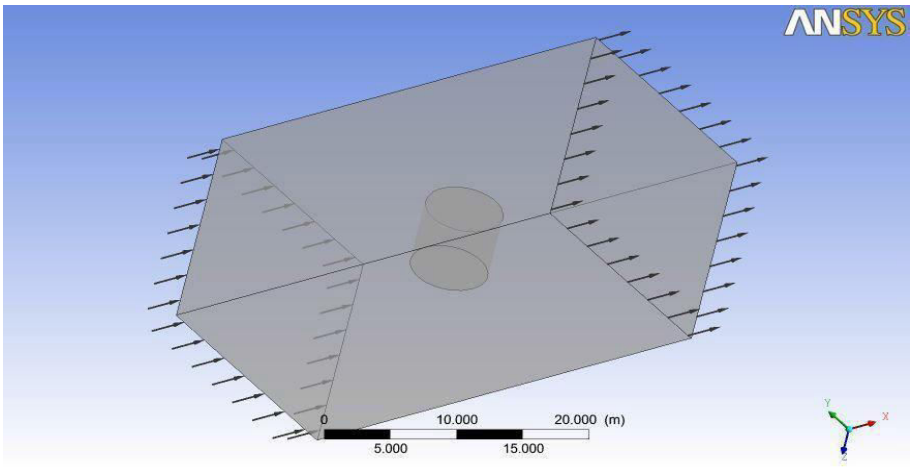
Mesh → compute mesh → compute.



- Output → outpur solver- ANSYS CFX → common solver → ANSYS →APPLY
- WRITE INPUT → OK

PROBLEM DEFINITION IN CFX-PRE:

- CFX → change the working directory → cfx-pre File → new case
- → general → apply.
- Mesh → import mesh → ICEM CFD → OK domain → fluid
- domain → air at 25c
- Boundary → inlet → domain: inlet → velocity=40m/s.
- Boundary → outlet → domain outlet → static pressure=0 Pa → apply Boundary → freeslip
- → domain free slip → free slip → ok.
- Solver settings → 1000 iterations → apply. Define solver →
- solver input file → ok

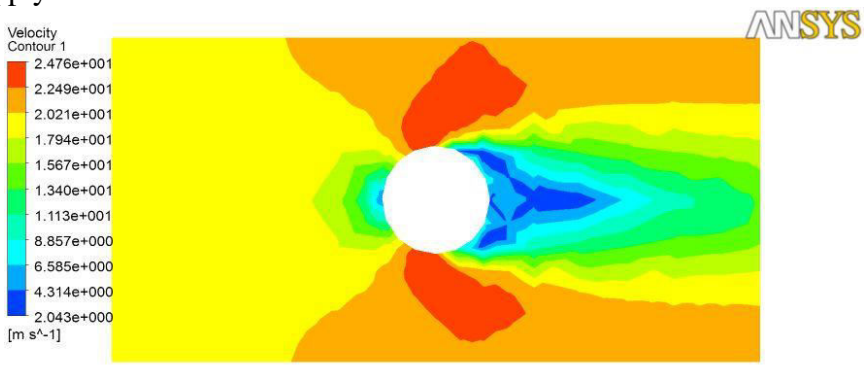


Solve:

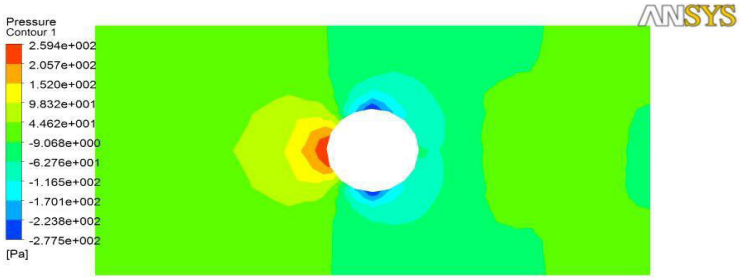
- CFD solver → open cfx file → define run → ok

Post processing:

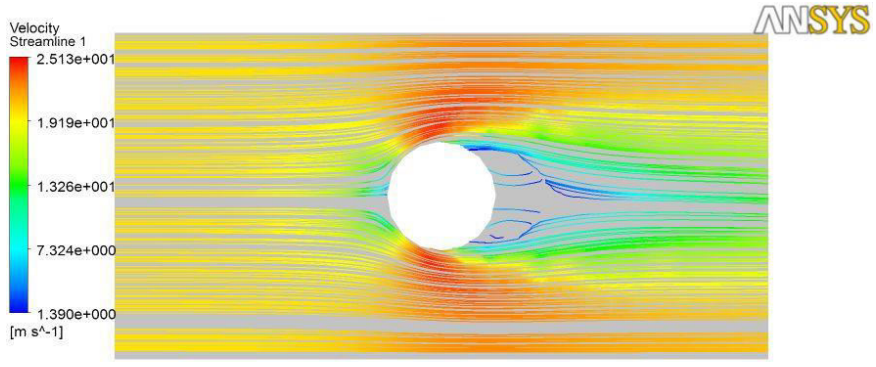
- CFD post → load result → select .res file
- Location → plane → Z=9 apply
- Contours → domain: plane1 → velocity → local → conservative → apply.



- Contours → domain: plane1 → pressure→ local → conservative → apply.



- Stream lines → domain : plane 1 → local → conservative → apply.



Exercise problems

10.1 Find out the effect of supersonic flow over cylinder .

10.2 Find the minor losses in a bend pipe

VIVA QUESTIONS:

1. Define CFD?
2. What are the three major steps of CFD?
3. What are the governing equations of CFD?
4. What is meant by Discretization?
5. Which type of Discretization is used in CFD?
6. Difference between forward and backward differencing scheme?
7. What is Explicit method and Implicit method?
8. What is LAX method?
9. What is a stability criterion?
10. What is thermal diffusivity?
11. Define Grid?
12. Difference between Structured and Unstructured grid?
13. What is meant by Grid Independence study?
14. What is linspace command in MATLAB?
15. How to give titles to X and Y axis of a graph?
16. How to create Hybrid mesh in ICEM?
17. How to create structured grid in ICEM?
18. What is the importance of Body point in ICEM?
19. How to define material properties in CFX or FLUENT?
20. What is meant by convergence criteria?
21. How to define supersonic inlet conditions in CFX?
22. What is Grid adaption technique?
23. What is meant by parallel and serial processing?
24. In how many ways CFD results can be presented?
25. How to define formulas in CFD Post?
26. What are the causes for reverse flow or diverged flow during CFD iterations?
27. What are the relaxations factors in FLUENT?
28. What is courant number and how does it affects the solution?
29. What are the different types of turbulence models in CFD?
30. Difference between free slip and no-slip conditions?