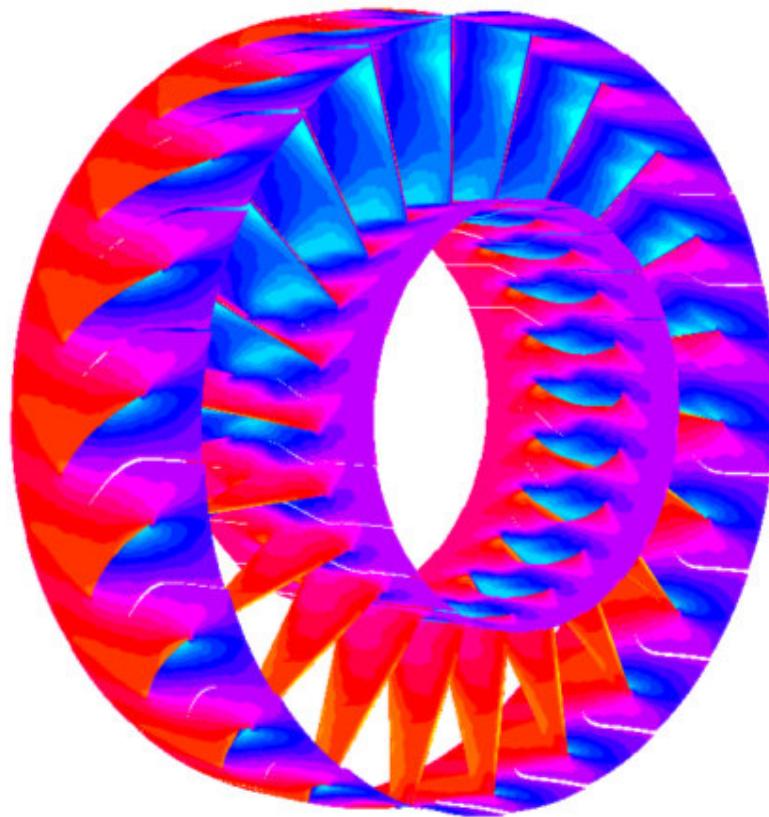


# 3D Design and Simulations of NASA Rotor 67

Author: Abdul Ahad Narejo



MASTER'S THESIS  
Mechanical Engineering, 2008  
Department of Technology, Mathematics and Computer Science

# **MASTER'S THESIS**

## **3D Design and Simulations of NASA Rotor 67**

### **Summary**

In this master's thesis work, research has been carried out to develop an automated and parameterized programming model in Matlab to generate a standard journal file, which can read by Gambit and produce a meshed 2D and 3D blade. This file then can be exported into mesh-formatted file for fluent for further simulations and numerical results.

<b>Author:</b>	Abdul Ahad Narejo
<b>Examiner:</b>	Dr. Tomas Beno
<b>Advisor:</b>	Dr. David Lindström Prof. Kenneth Eriksson
<b>Programme:</b>	Master's Programme Simulation of Manufacturing Processes
<b>Subject:</b>	Mechanical Engineering
<b>Date:</b>	June 13, 2008
<b>Keywords</b>	Gambit, Fluent, Matlab, residuals, priesto, segregated, coupled
<b>Publisher:</b>	University West, Department of Technology, Mathematics and Computer Science, S-461 86 Trollhättan, SWEDEN Phone: + 46 520 22 30 00 Fax: + 46 520 22 32 99 Web: <a href="http://www.hv.se">www.hv.se</a>
<b>Level:</b>	Master
<b>Report Number:</b>	2006:SM01

## **Preface**

I am highly thankful to Dr. David Lindström (Doctor at University West, Sweden) and Prof. Kenneth Eriksson (Professor at University West, Sweden) for their contributions and discussions from time to time, to complete this thesis work. So finally I dedicate this work to both of them.

## Contents

<b>Summary.....</b>	<b>ii</b>
<b>Preface .....</b>	<b>iii</b>
<b>Contents.....</b>	<b>iv</b>
<b>List of symbols.....</b>	<b>vi</b>
<b>Chapter 1.....</b>	<b>1</b>
<b>    1 Introduction .....</b>	<b>1</b>
1.1 Introduction and Back Ground .....	1
1.2 Objective .....	1
1.3 Limitations .....	1
1.4 Delimitations.....	2
<b>    Chapter 2.....</b>	<b>3</b>
<b>    2 Fundamental Concepts.....</b>	<b>3</b>
2.1 Atmospheric Fundamentals.....	3
2.2 Viscosity .....	3
2.3 Rotor Theory.....	3
2.4 Rotor Geometry.....	4
2.5 Aerodynamic forces and moments on a rotor.....	5
2.6 Basic aerodynamic principles and modelling.....	6
2.6.1 The Continuity equation.....	6
2.6.2 The incompressible Bernoulli's equation.....	6
2.6.3 The isentropic equation of state.....	7
2.6.4 The compressible Bernoulli's equation .....	7
2.6.5 Incompressible flow vs. compressible flow.....	8
<b>    Chapter 3.....</b>	<b>9</b>
<b>    3 Gambit Modelling.....</b>	<b>9</b>
3.1 2D design of airfoil.....	9
3.2 Boundary Conditions.....	11
3.3 3D design of airfoil.....	12
3.4 Boundary Conditions for 3D domain.....	19
<b>    Chapter 4.....</b>	<b>20</b>
<b>    4 Modeling and Boundary conditions.....</b>	<b>20</b>
4.1 Modelling in Fluent.....	20
4.1.1 Solver.....	20
4.1.2 Energy Equation.....	21
4.1.3 Viscous Model .....	21
4.2 Materials .....	22
4.3 Operating conditions.....	22
4.4 Boundary conditions.....	23
4.4.1 Air.....	23
4.4.2 Airfoil.....	24
4.4.3 Outer and inner wall .....	24
4.4.4 Periodic .....	24
4.4.5 Inlet pressure.....	25
4.4.6 Outlet pressure .....	26

4.4.7	Default interior .....	26
4.4.8	Solutions controls.....	27
4.5	Solution Initialisation.....	28
<b>Chapter 5.....</b>		<b>29</b>
<b>5</b>	<b>Simulations and numerics .....</b>	<b>29</b>
5.1	Simulations and numerical Results.....	29
<b>6</b>	<b>Conclusions.....</b>	<b>36</b>
<b>References .....</b>		<b>37</b>
<b>Appendices</b>		
A.	Appendix 1	
B.	Appendix 2	
C.	Appendix 3	
D.	Appendix 4	

## **List of symbols**

It will help your readers if you provide a list of new symbols, abbreviations, and definitions used in the report

P	Atmosphere pressure
$\rho$	Air density
g	Gravitational constant
T	Absolute temperature
R	Gas constant
$\gamma$	Ratio of specific heats
C <sub>p</sub>	Specific heat constant at constant pressure
C <sub>v</sub>	Specific heat constant at constant volume
A	Area
m	Mass
q	Amount of Heat
e	Internal energy

# **Chapter 1**

## **1 Introduction**

### **1.1 *Introduction and Back Ground***

Several CAD and CFD software are available today for turbo machinery design but most software are quite expansive and are not as flexible as the work proposed in this work. In general, when one have to design a rotor in Gambit, the design could be started from scratch, if many designs are carried out manually, it will cost a lot of time to save time, as time is money this research work will be carried out on a fundamental design and simulation of NASA 67 rotor and to build a link/program to handshake between CAD and CFD software. The program could be written in any good programming language but in this work Matlab has been selected, as it is strong mathematical language and tool. This customize program will help to save the time as well as to produce a customized airfoil geometry and domain. The improved/customized model is utmost important to predict the performance of an airfoil. The performance of the rotor depends of properties of atmosphere, shape and orientation. In CFD analysis, to simulate certain parameters of the fluid flow for the design of turbo-machinery in order to increase efficiency has become a vital to use CFD tools.

### **1.2 *Objective***

To develop a parameterized 3D CAD and meshed model with boundary conditions in Matlab to produce a Gambit journal file format for NASA 67 rotor blade and simulates a 3D rotor and a blade in fluent.

### **1.3 *Limitations***

1. The simulations have only been carried out on one stage rotor.
2. To create an automatic airfoil loop, one must need 47 vertices in case of more vertices than 47 the Matlab programme selects 47 vertices automatically.
3. The 3D-CAD geometry can be produced with any inner constant radius and outer variable radius in longitudinal direction. As well as number of blades can be defined as per requirement. The inlet and outlet flow angle can be defined as per requirement, for further more flexible parameters' details read section 2D design of airfoil in chapter 3.
4. The following software have been used for all work.
5. Matlab ver. R2000b
6. Gambit ver. 2.2.30
7. Fluent ver. 6.2.16

#### **1.4 Delimitations**

1. The built in, Gambit Turbo machinery module is not used because of the non-availability of license.
2. The optimization techniques for turbo-machinery rotor those are very useful to produce optimal aerodynamic shape and mesh for a domain is not considered in this thesis work.

## **Chapter 2**

### **2 Fundamental Concepts**

#### **2.1 Atmospheric Fundamentals**

The composition of atmosphere varies with altitude, generally in most cases the atmosphere is considered as a homogeneous gas, therefore in most cases air can be regarded as an ideal gas and follows this ideal gas law.

$$p = \rho g RT$$

Where

$P$ = atmosphere pressure

$\rho$ =air density

$g$ = gravitational constant

$T$ =absolute temperature

#### **2.2 Viscosity**

Apart from pressure and temperature, another important property is viscosity that effect on performance of rotor, because there is a close relation ship between viscosity of air and boundary layer of rotor.

#### **2.3 Rotor Theory**

The aerodynamic characteristics of rotors are highly influenced by the compressibility/Mach number and viscous effects/Reynolds number. The total force (lift and drag) and moment coefficient depend on pressure distribution over the rotor.

## 2.4 Rotor Geometry

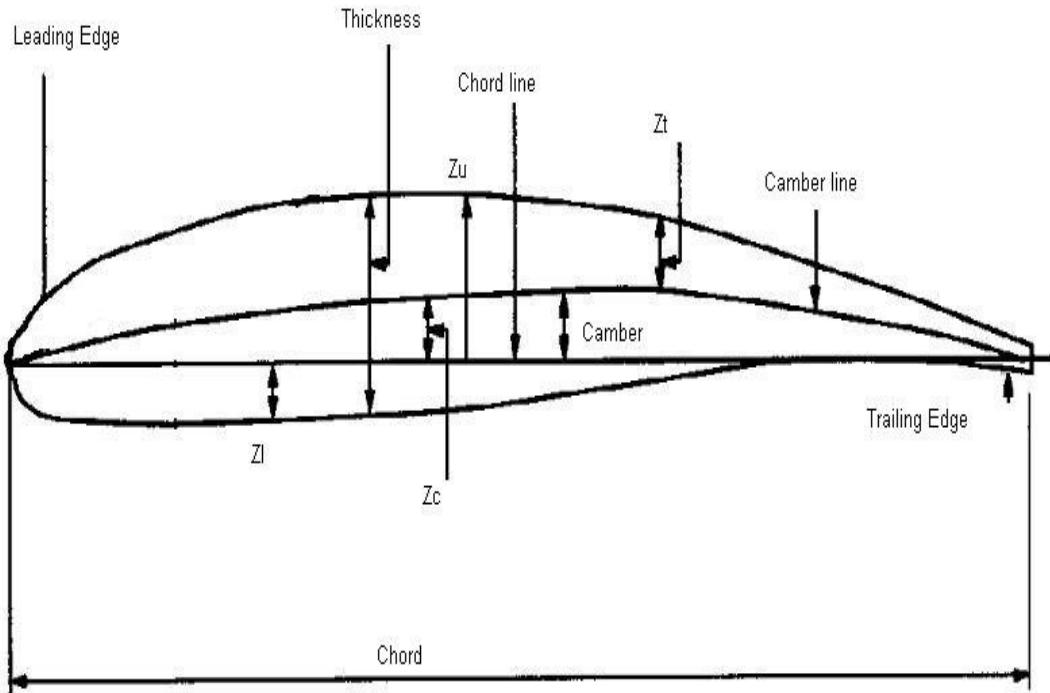


Fig. 2.1. Schematic Diagram of Airfoil

Rotor is a body that is formulated either by polynomials or certain set of data. The NACA/NASA terminologies have been explained below.

### **Leading Edge**

It is an edge at the front of rotor.

### **Trailing edge**

It is an edge at the rear of rotor.

### **Thickness**

It is the perpendicular distance over the chord line.

### **Chord line**

It is the line that connects left most and right most point of mean camber line.

### **Camber line**

The camber line is the line that is commonly divided between upper and lower line of the rotor. It is also called mean camber line.

### **Camber**

It is the normal distance between chord line and camber line.

The constants like  $Z_c$ ,  $Z_t$ ,  $Z_u$  and  $Z_l$  can be calculated by following formulas.

1.  $Z_c = (Z_u + Z_l) / 2$
2.  $Z_t = (Z_u - Z_l) / 2$
3.  $Z_u = (Z_c + Z_t) / 2$
4.  $Z_l = (Z_c - Z_t) / 2$

## **2.5 Aerodynamic forces and moments on a rotor**

When the torque is applied on rotor's shaft then the rotor will create aerodynamics forces. These aerodynamic forces depend on the eight variables.

### **Air Velocity**

It is velocity of air at inlet.

### **Air density**

The air density can be calculated by using ideal gas law.

### **Coefficient of viscosity**

It is property of a fluid that shows the degree to which a fluid resists flow under the action of an applied force.

### **Numbers of blades**

Numbers of blades are very important to design optimal performance of turbo machinery. For example in NASA 67 rotor, 22 blades are used.

### **Speed of sound**

The speed of sound also varies with temperature. For example at a temperature of 20°C, the speed of sound is 345 m/s about. Practically air is compressible and viscous. The behaviour of air depends of Mach number. Mach number is equal to 1 when air velocity is equal to speed of sound relative to those following terminologies have been developed based on flow regimes.

Subsonic—Mach numbers below 0.75

Transonic—Mach numbers from 0.75 to 1.20

Supersonic—Mach numbers from 1.20 to 5.00

Hypersonic—Mach numbers above 5.00

When Mach number is near to 1 or more than 1, shock waves are produced, these are large amplitude compression waves and these waves cause separation of flow.

### **Flow angle**

The acute angle between the chord line of an airfoil and a line represents the airflow streams. It is also known as angle of attack.

### **Meridional angle**

It is the flow angle from apex to base

### **Tangential angle**

It is the flow angle on tip surface of the blade.

## **2.6 Basic aerodynamic principles and modelling**

The aerodynamic modelling is based on the following equations/principles.

### **2.6.1 The Continuity equation.**

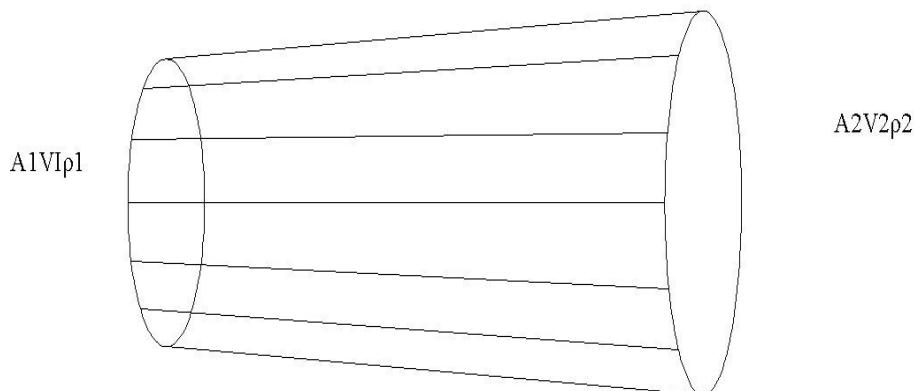


Fig. 2.2. Diagram for continuity equation

If flow is compressible then,

$$A_1 V_1 \rho_1 = A_2 V_2 \rho_2$$

Where A is area, V is velocity and  $\rho$  is density.

This equation is called the compressible continuity equation.

The subscript 1 is at the inlet and subscript 2 is at the outlet.

If the flow is incompressible then,

$$A_1 V_1 = A_2 V_2 \quad \text{i.e. } \rho_1 = \rho_2$$

This equation is called the incompressible continuity equation.

### **2.6.2 The incompressible Bernoulli's equation.**

It states that, when velocity of fluid is increased then pressure is decreased and when velocity is decreased then the pressure is increased. This is called Bernoulli's Principle.

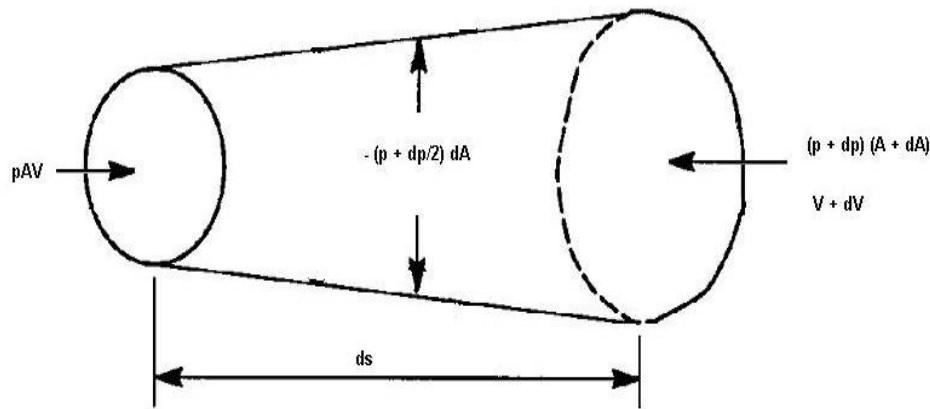


Fig. 2.3. Diagram for Bernoulli's equation

$$p + 0.5 \rho V^2 = \text{Constant.}$$

This equation is called incompressible Bernoulli's Equation.

### 2.6.3 The isentropic equation of state.

The internal energy of this system can be calculated by using the first law of thermodynamics; an amount of heat  $dq$  added to a unit mass of gas will result in a differential change in energy  $de$  and expansion of volume  $dv$ .

$$dq=de+pdv$$

If gas is perfect, the equation of state can be written as

$$pV=mRT$$

The equation of state can also be written as

$$pV^\gamma = \text{Constant, where } \gamma = Cp/Cv$$

$\gamma$  is a ratio of specific heat of the gas.

### 2.6.4 The compressible Bernoulli's equation

If flow is compressible, then the isentropic equation can be sum up in a following way.

$$(\gamma / (\gamma - 1)) p / \rho + 0.5 V^2 = \text{Constant}$$

This equation applies to isentropic flow means no heat transfer between streamline and this equation is also valid when the shock waves occurs.

## 2.6.5 Incompressible flow vs. compressible flow

### I. Incompressible flow

Generally at low speeds, the flow is to be regarded as an incompressible flow.

Firstly we consider incompressible Bernoulli's equation i.e.

$$p + 0.5 \rho V^2 = p_t$$

Where

$p$  is static pressure

$0.5 \rho V^2$  is the dynamic pressure.

$p_t$  is the total pressure or the stagnation pressure.

### II. Compressible flow

Generally speaking, the flow is compressible when  $Ma$  is greater than 0.3, the compressible Bernoulli's equation can be written as

$$(\gamma / (\gamma - 1))p/\rho + 0.5 V^2 = (\gamma / (\gamma - 1))p_t/\rho_t$$

## **Chapter 3**

### **3 Gambit Modelling**

#### **3.1 2D design of airfoil**

Initially, a 2D design of an airfoil was made by using set of data in Gambit software, a journal file has been saved, that journal file has been programmed in matlab by setting variables and functions as per requirement. The matlab program will generate customized journal file with certain variable and function. This journal file can be read in Gambit and generate a meshed file automatically, that can be used in fluent for calculations and simulations, so there is no need to design and mesh an airfoil again in gambit. The figure 3.1 will give you some idea about the basic conceptual work in matlab.

It is necessary to find out the lowest coordinate in the airfoil data to generate a good mesh in a domain. For that purpose edgeindex(filename) has been designed in matlab where the output of that function is a index number, that have the lowest x-value. filename is the file that contains geometrical data of the airfoil and the output will be the index that have the lowest x-ordinate value in the blade coordinate array.

The function is explained below.

e.g. it can be used at matlab prompt as,

```
edgeindex('airfoil.dat')
```

After receiving the index, it is required to read the coordinates of a certain index for that purpose the function coxy=edgeindexc(filename,x) can be used.

where

filename is the file that contains the geometrical data of an airfoil and the output will be the index of the lowest x-ordinate value in blade coordinate array.

if x=1 calculate minimum values in a data file and x=2 calculate maximum values in data file.

It can be used in the matlab prompt as,

e.g.

```
edgeindexc('airfoil.dat',1)
```

the last functions for a 2D airfoil is automatically designed and the meshing function is nodeco(filename,cno)

The function reads the co-ordinates of a certain node from the data file and cno is a index number in the file.

e.g.

it can be used at matlab prompt as,

`edgeindexc('airfoil.dat',27)`

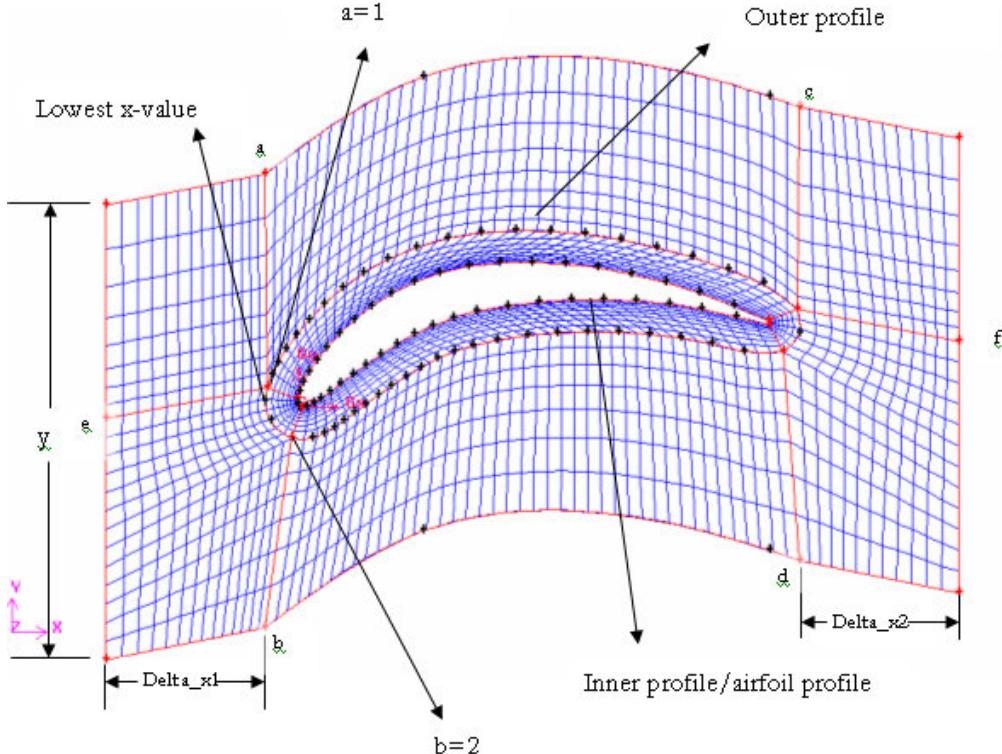


Fig. 3.1. Schematic diagram for basic parameters to design 2D mesh.

$$\text{delta\_y1} = \text{delta\_x1} * \tan(\alpha);$$

$$\text{delta\_y2} = \text{delta\_x2} * \tan(\beta);$$

In the `make_2dgambit_journal.m` program, many parameters and code lines can be edited easily to receive desired journal file. The profile data is loaded from the data file and a computer program generates the outer profile data and the inner profile data, to get a good mesh around the inner profile a program is used. The program called `in_out_prf_gen(filename,nr)`, where the file name is a data file, that contains airfoil data e.g `in_out_gen('airfoil.01',1)` and nr shows number of profile, calculates the mean distance between two inner data file locations and generates a point perpendicular on it. The perpendicular distance is variable in the program, because in some cases when curve is sharp for the airfoil, it might get a bad outer profile, that error can be controlled by defining short perpendicular distance in the program. The program also calculates three extra points outlet (trailing edge) of this profile. Finally program writes `naca_nr.outer` (`naca_1.outer`) and `naca_nr.inner` (`naca_1.inner`) the data files in the logged directory. The extra edges have been created on e and f respectively

to get real faces in gambit software and to produce a good mesh on radial profile in 3D domain.

A Few parameters/variables are discussed here to get a customized mesh and domain.

$a=1$ ; a represents is a difference between minimum index and index on above side.

$b=2$ ; b represents is a difference between minimum index and index on below side.

$\delta_x1=2$ ; certain distance from leading edge as shown in above figure.

$\delta_x2=2$ ; certain distance from trailing edge as shown in above figure.

$\alpha=.1$ ; alpha is a inflow angle in rad.

$\beta=.1$ ; beta is a outflow angel in rad.

$\alpha$  and  $\beta$  can be formulated as,

$$\tan(\alpha) = \delta_y1 / \delta_x1 \text{ and } \tan(\beta) = \delta_y2 / \delta_x2$$

$y = 2\pi r / nob$ ;  $y$  is the ratio of the circumference to the number of blades and where the nob represents the number of blades.

The nodes locations depend on data given as input to the program and the program generates all vertices mathematically e.g. vertices a and b have been calculated by using function `edgeindexc(['naca_',num2str(nr),'outer'],1)`, that gives minimum x-ordinate, after that half of vertical distance added and subtracted in y-ordinate respectively, similarly, vertices c and d have been calculated by using function `edgeindexc(['naca_',num2str(nr),'outer'],2)`, that gives the maximum x-ordinate, after that half of the vertical distance added and subtracted in the y-ordinate respectively.

### 3.2 Boundary Conditions.

The boundary conditions have been set in matlab program as the left most side is considered as inlet pressure, opposite side is considered as outlet pressure, sides are considered as periodic and domain contains air as fluid and airfoil is considered as wall.

### **3.3 3D design of airfoil**

3D design of an airfoil is rather complicated work, to proceed to 3D work all 2D work considered as base. The 3D airfoil geometry is composed of several 2D layers projected on half cylinder with different radius, then projected layers must be connected together, and to form a 3D airfoil geometry.

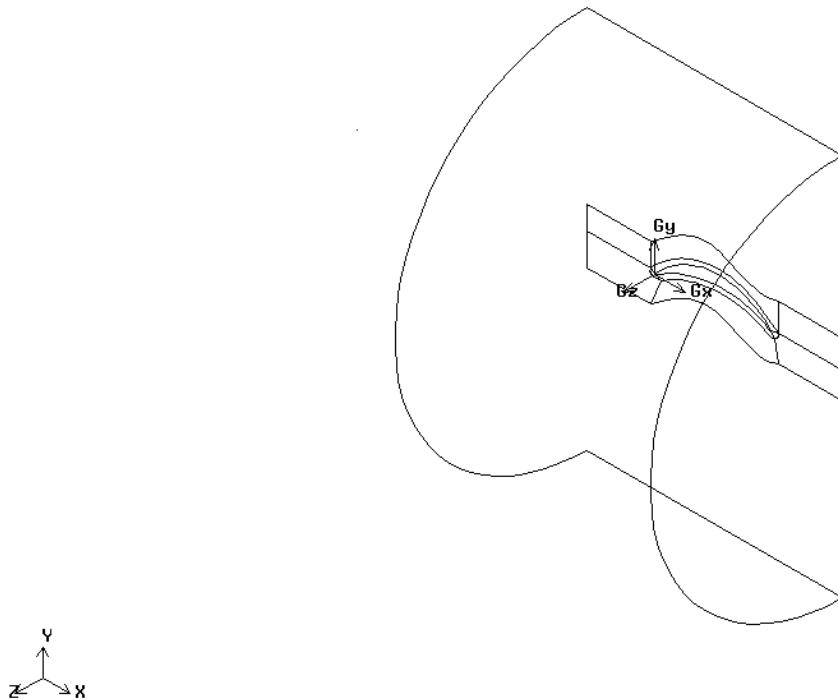


Fig. 3.2 Half cylinder and 2D airfoil with domain

The first step is to draw half cylinder with radius then the second step is to project the 2D domain on the cylinder. It is necessary to project the all edges of the 2D domain must be projected on the half cylinder step by step. That can be done by using project command in Gambit as shown in below figure.

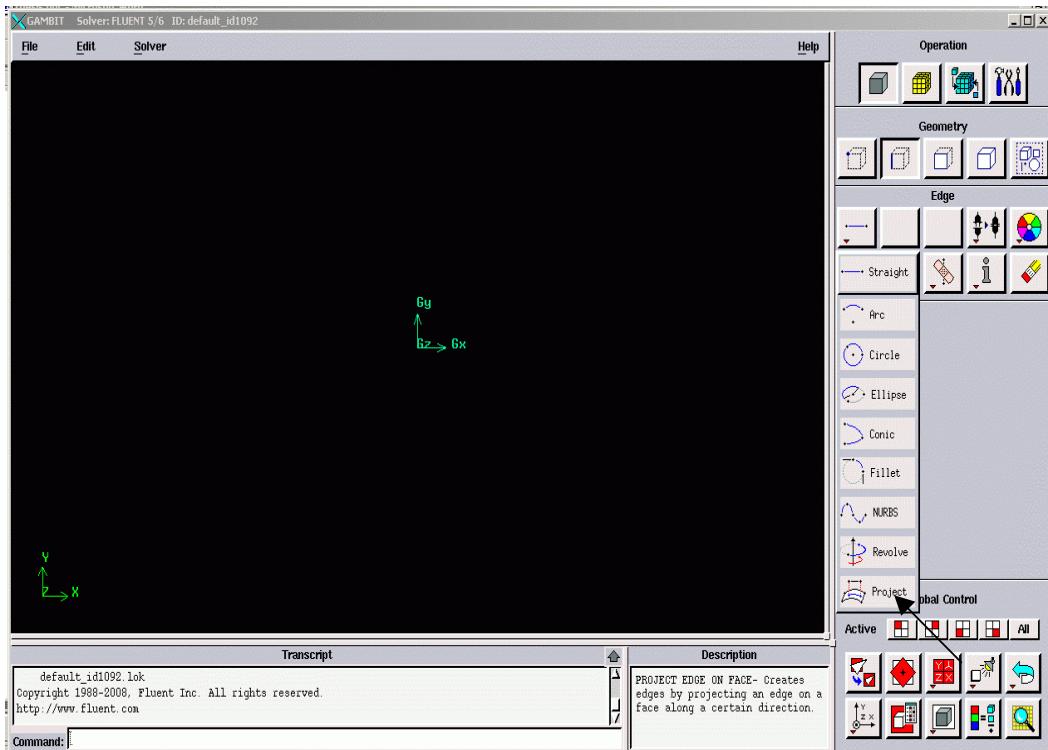


Fig. 3.3. Gambit screen shows project command.

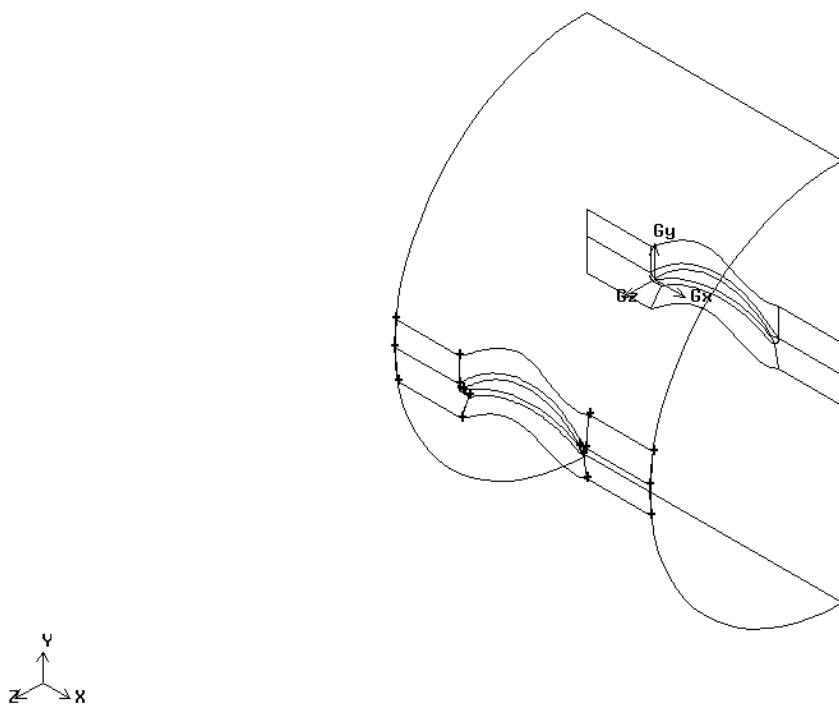


Fig. 3.4. Projection on the half cylinder.

When all edges will be projected on the cylinder, from the all projected edges the real faces must be composed as shown in fig. 3.4. after that the half cylinder can be deleted as shown in below fig. 3.5.

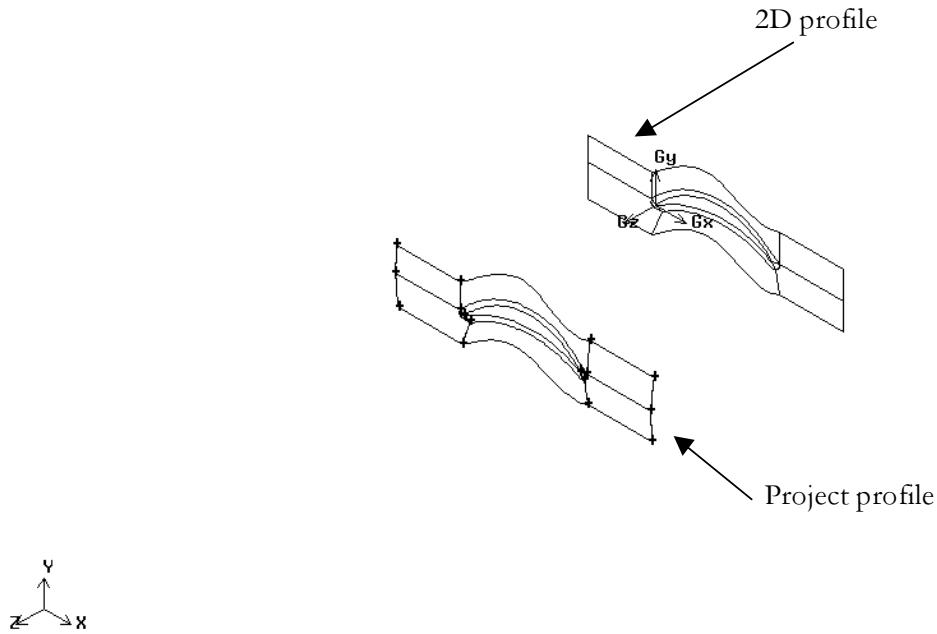


Fig. 3.5. Deletion of unused faces on half cylinder.

After that data for a second layer or profile can be loaded and all the above steps can be repeated. But a second projection on the second half cylinder as shown in fig. 3.6, can not be easy, for that purpose work could be carried out to find out the incremental loop constants, after finding and testing, the several layers can be projected automatically.

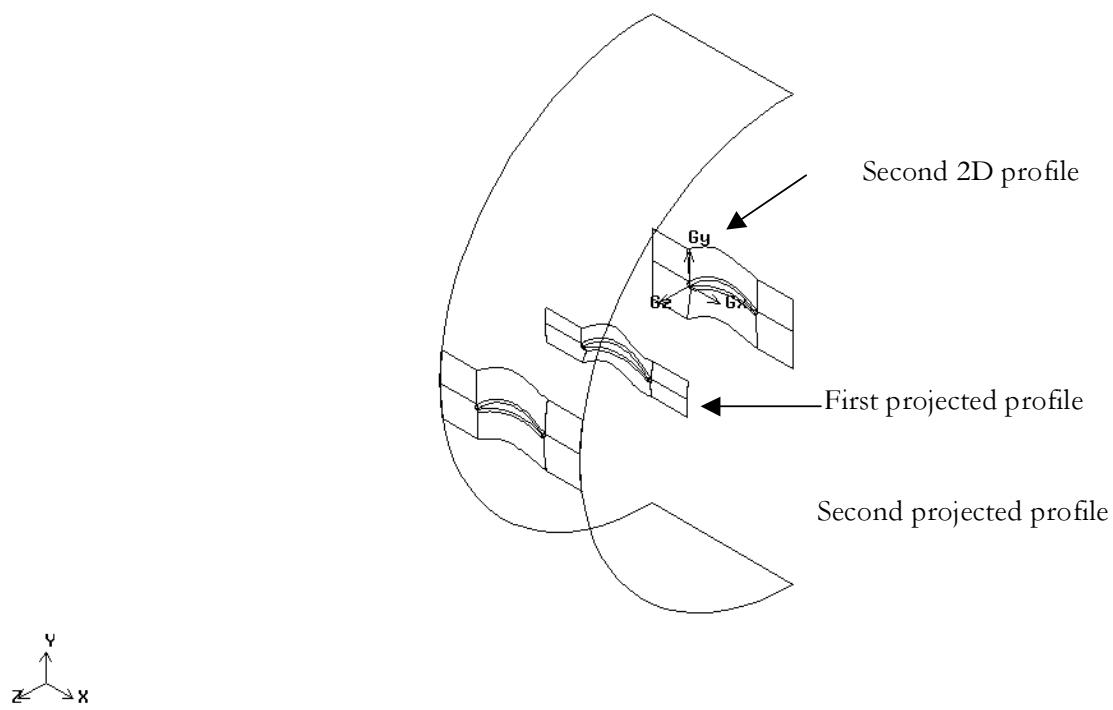


Fig. 3.6 Projection on Second layer.

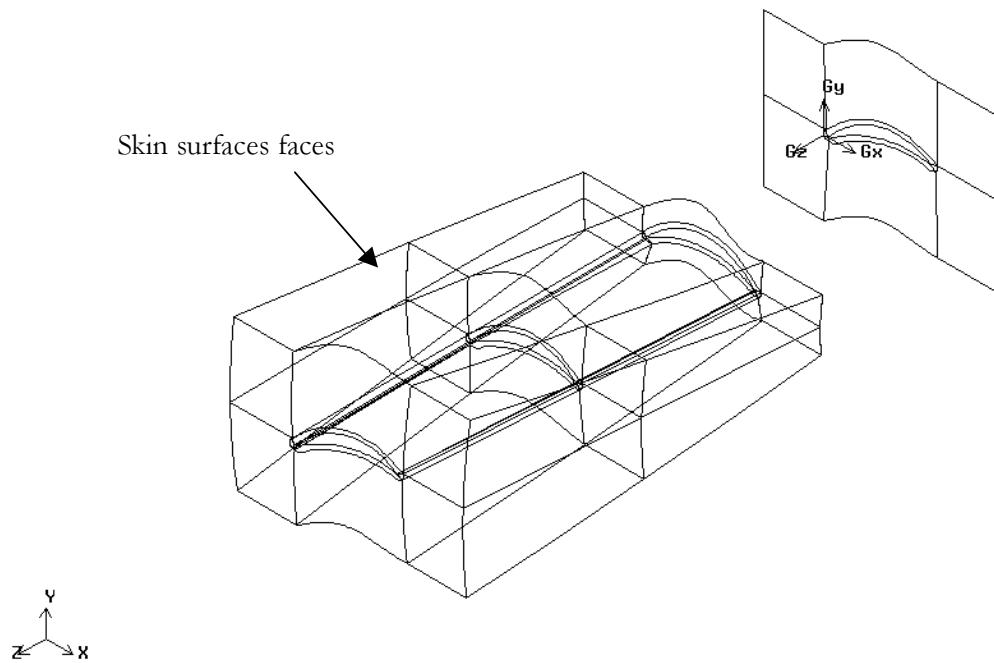


Fig. 3.7 Creation of skin surface faces.

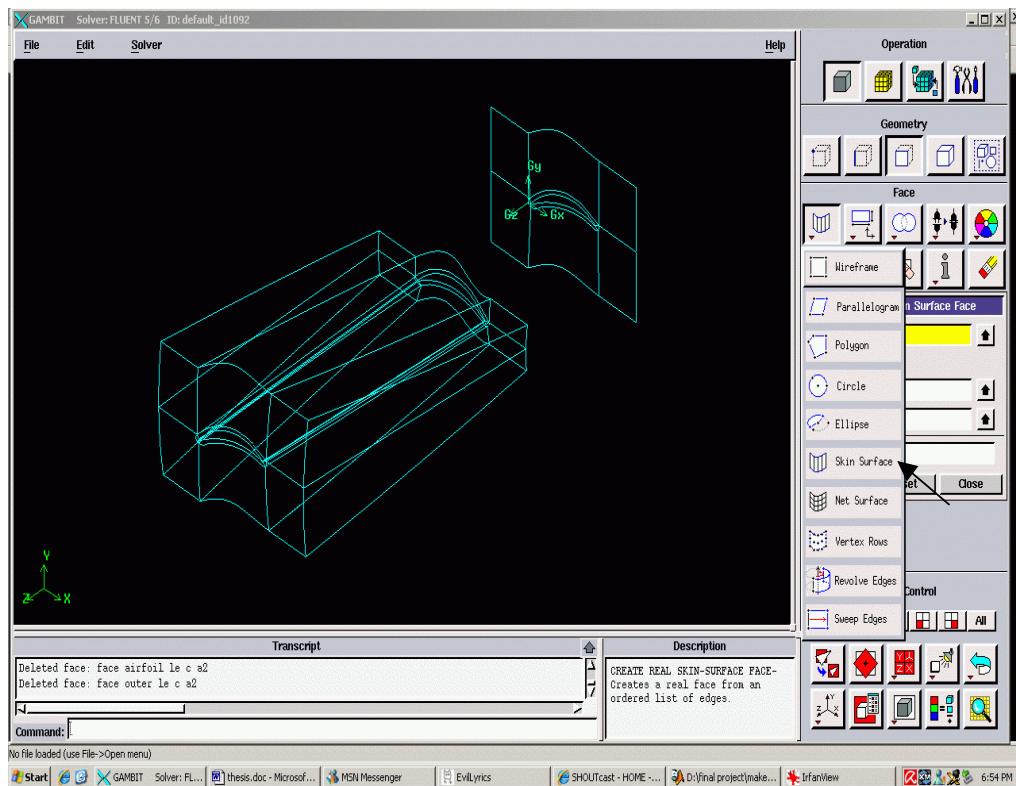


Fig. 3.8. Command of skin surface in gambit

To create the real skin surface faces, the all identical edges must be selected in the order. After that from every six consecutive faces a unit volume can be created. There are 10 units of volume in the fig. 3.8.

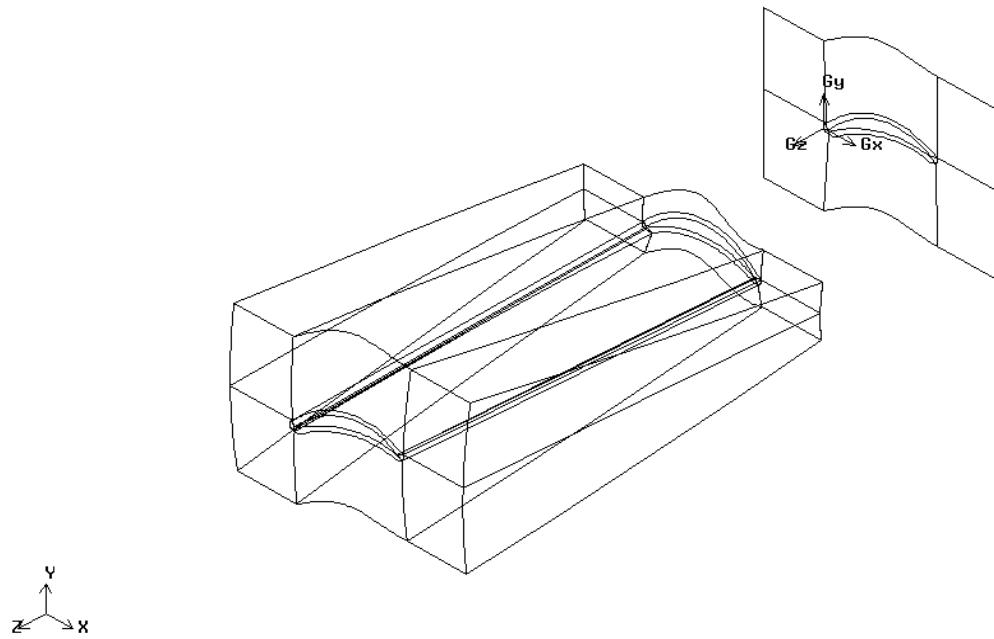


Fig 3.9. Creation of sub-volumes.

After that, data can be loaded, to create a cutting profile. The 3D airfoil can be cut at a certain radial distance as per requirement with the created cutting profile.

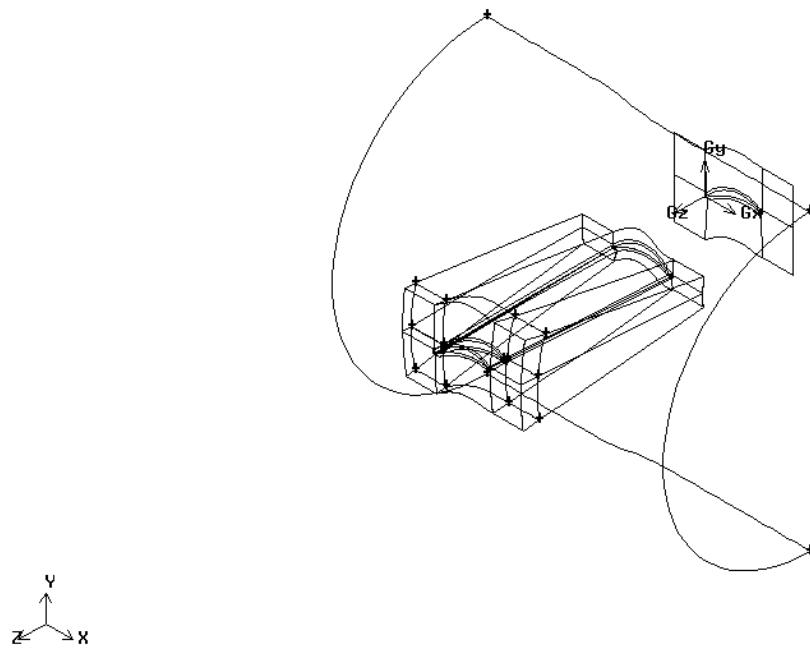


Fig. 3.10 3D airfoil have been shown with cutting profile.

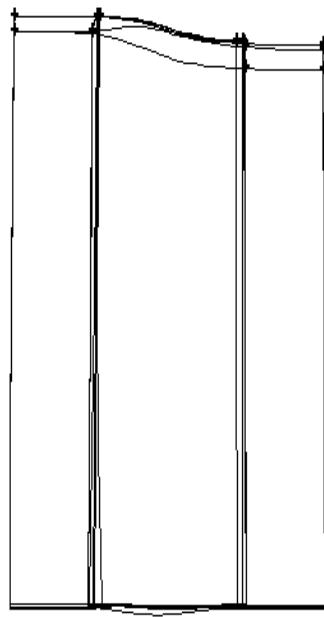


Fig. 3.11 3D view of airfoil after cutting with cutting profile.

After cutting with the cutting profile, the airfoil can be obtained as shown in fig. 3.11. and the cutting profile can be deleted.

Finally, composed volume is meshed with tetrahedral units as shown below, which later can be exported into a fluent mesh file.

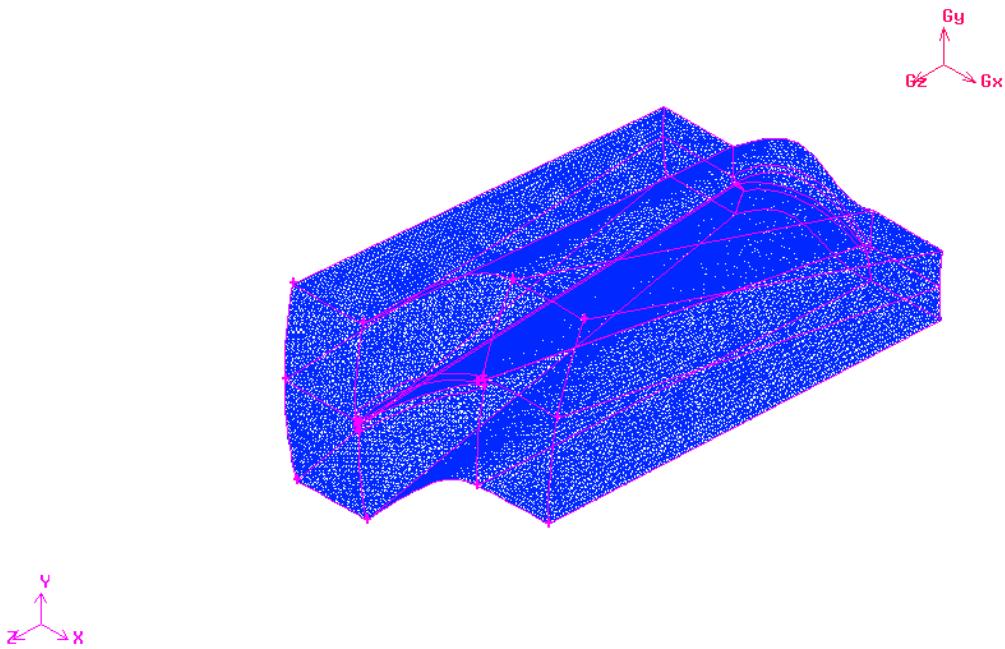


Fig. 3.12 Meshed Domain.

The all above tasks have been considered, customized program have been written i.e. make\_gambit\_journal.m in matlab to do all these tasks and generates gbt3dnr.jou automatically, where nr shows number of layers, that is variable, in the case of two layers the geneated file name will be gbt3d2.jou, so there is no need to work in Gambit, only one need to run gambit file in it.

### **3.4 Boundary Conditions for 3D domain**

The boundary conditions can be set following way

The inlet and outlet can be set as presurre inlet and presssure oulte respectively, the side profiles can be set as periodic, lower,upper wall and airfoil can be considered as wall.

## Chapter 4

### 4 Modeling and Boundary conditions

#### 4.1 Modelling in Fluent

The modelling and setting of parameters is crucial step in modelling, simulation and numerical solutions. Here are some crucial schemes and parameters discussed below.

##### 4.1.1 Solver

Fluent have two solvers; Segregated Solver and Coupled Solver.

Both solver schemes can be used for steady and unsteady flows. But in general segregated solver is more flexible in fluent, gives more optional settings than coupled solver. Both solvers are composed of momentum equations, pressure correction, (continuity) equation, energy, turbulence and other equations. The segregated solver solves equations in sequence while coupled solver solves equations simultaneously hence consumes more memory than segregated solver. The segregated model can be used up to 1.5 Mach number but at higher Mach number coupled solver is recommended.

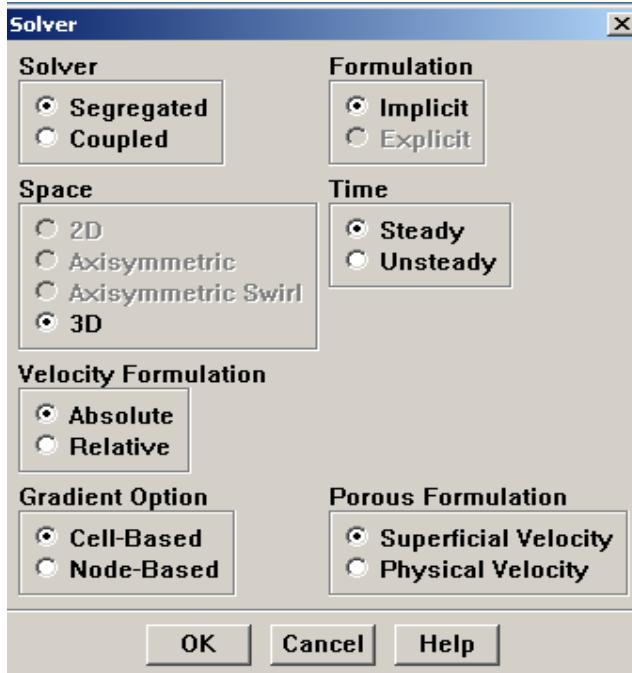


Fig. 4.1 Solver window

The study state option has been selected as per requirement in this research and further options can be selected as per requirement.

#### 4.1.2 Energy Equation

The energy equation gives you rights to set the energy related parameters or the heat related parameters. Panel allows to set parameters related to energy or heat transfer in your model. It also calculates viscous energy, pressure energy, kinetic energy and energy includes diffusion at inlets.

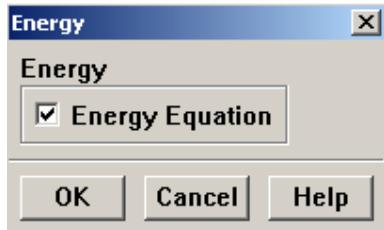


Fig. 4.2. Energy window

#### 4.1.3 Viscous Model

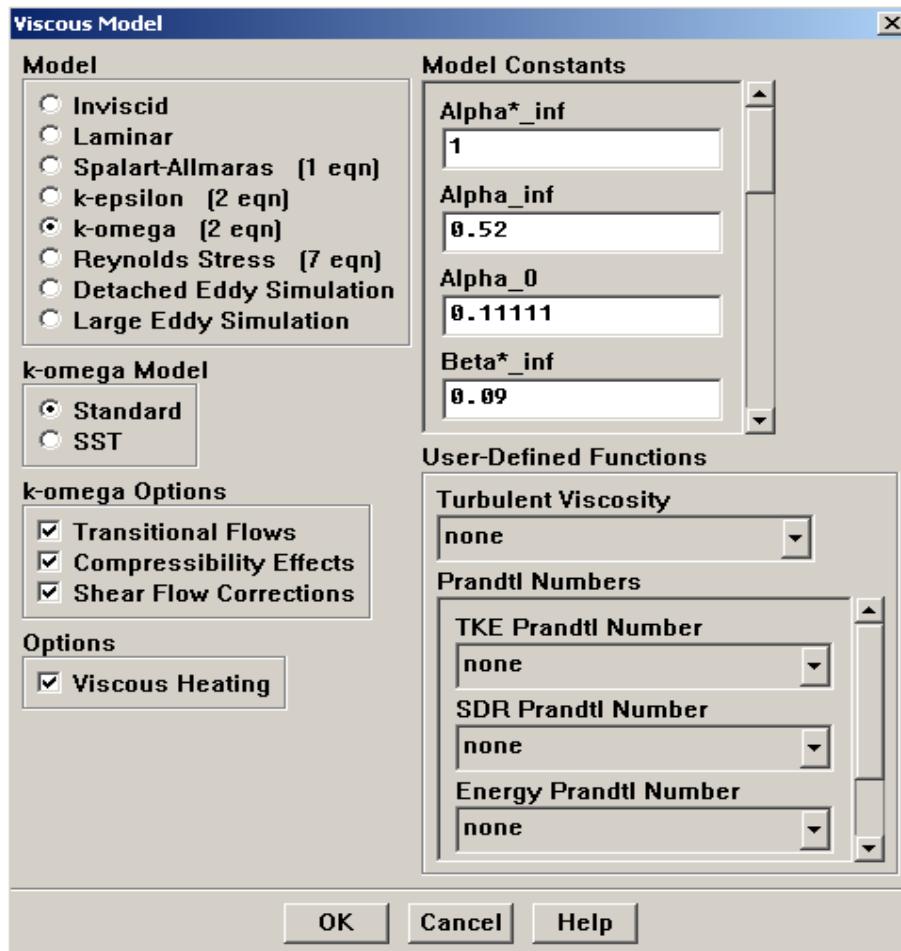


Fig. 4.3 Viscous Model Window

When the fluid flow is laminar, transitional and turbulent then k-omega model can be used. The k-omega model is used with default options.

## 4.2 Materials

The air has been considered as ideal gas.

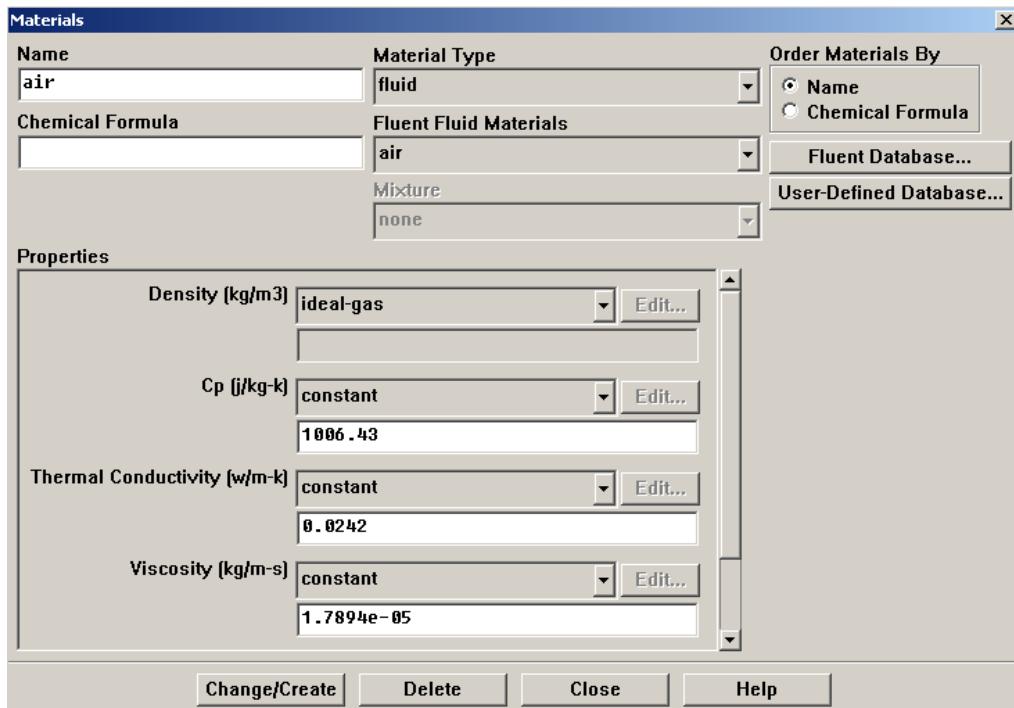


Fig. 4.4 Material properties Window

## 4.3 Operating conditions

The operating conditions have been set as show in below figure.

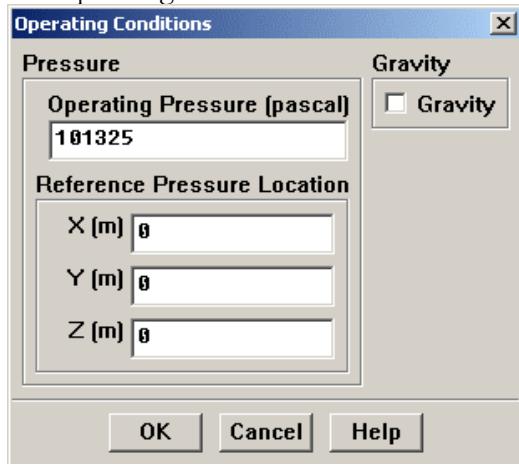


Fig. 4.5 Operating conditions Window

## 4.4 Boundary conditions

The most critical and crucial step is set to boundary conditions to get authentic results.

### 4.4.1 Air

The air has been selected as air, moving in longitudinal direction and rotation in x-axis.

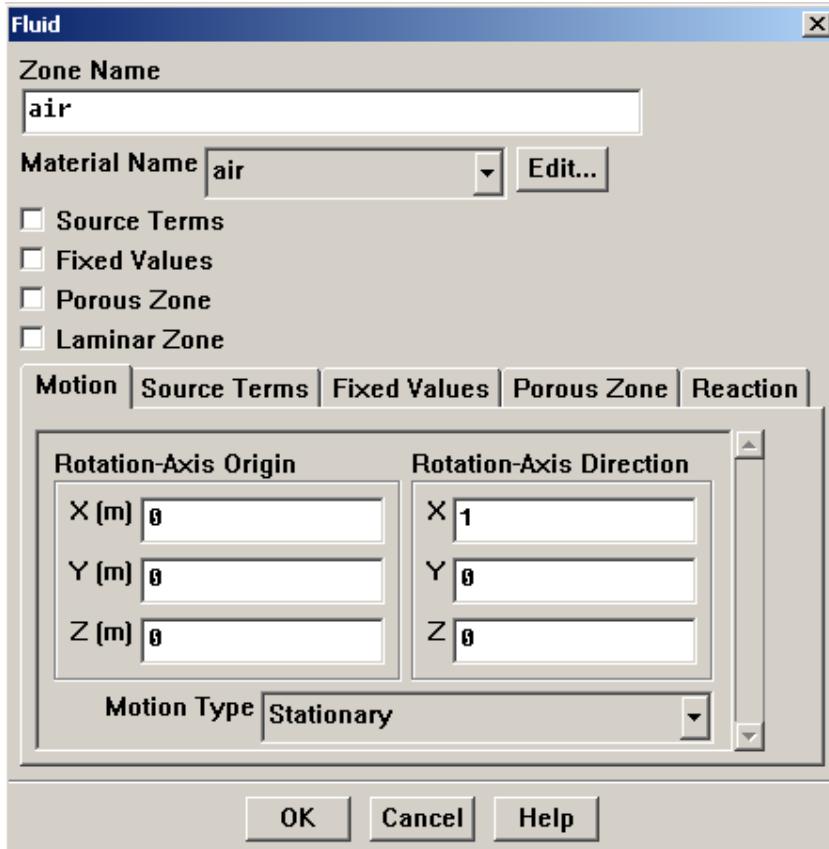


Fig. 4.6 Fluid properties window

#### 4.4.2 Airfoil

The airfoil has been considered as wall. The wall is considered moving and rotating in x-axis.

The rest options have been used as default.

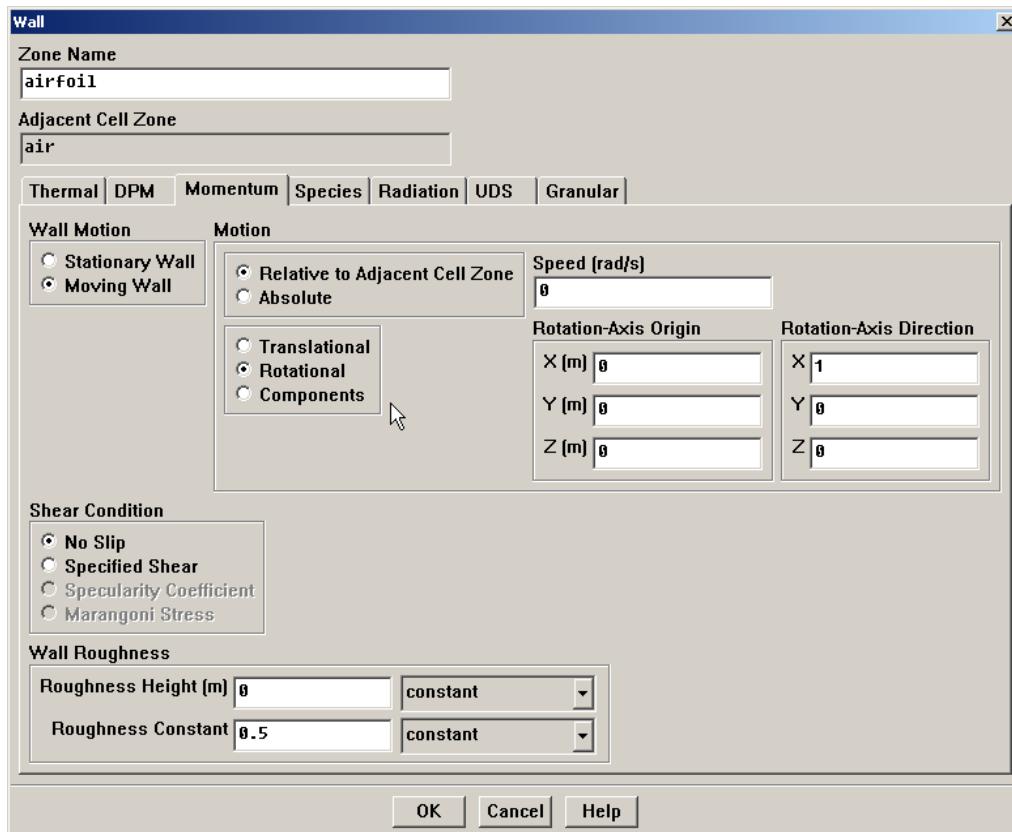


Fig. 4.7 Wall properties window

#### 4.4.3 Outer and inner wall

Outer and inner wall's boundary conditions have been set as airfoil because airfoil is connected with the walls.

#### 4.4.4 Periodic

The interfaces on both sides of airfoil have been considered rotational and periodic.

#### 4.4.5 Inlet pressure

The code has been written in matlab to create profiles for gauge total pressure, supersonic/initial gauge pressure, total pressure, radial, tangential and axial flow direction components from data file so fluent can read as boundary conditions. After that profiles have been imported in fluent and used as shown in below figure.

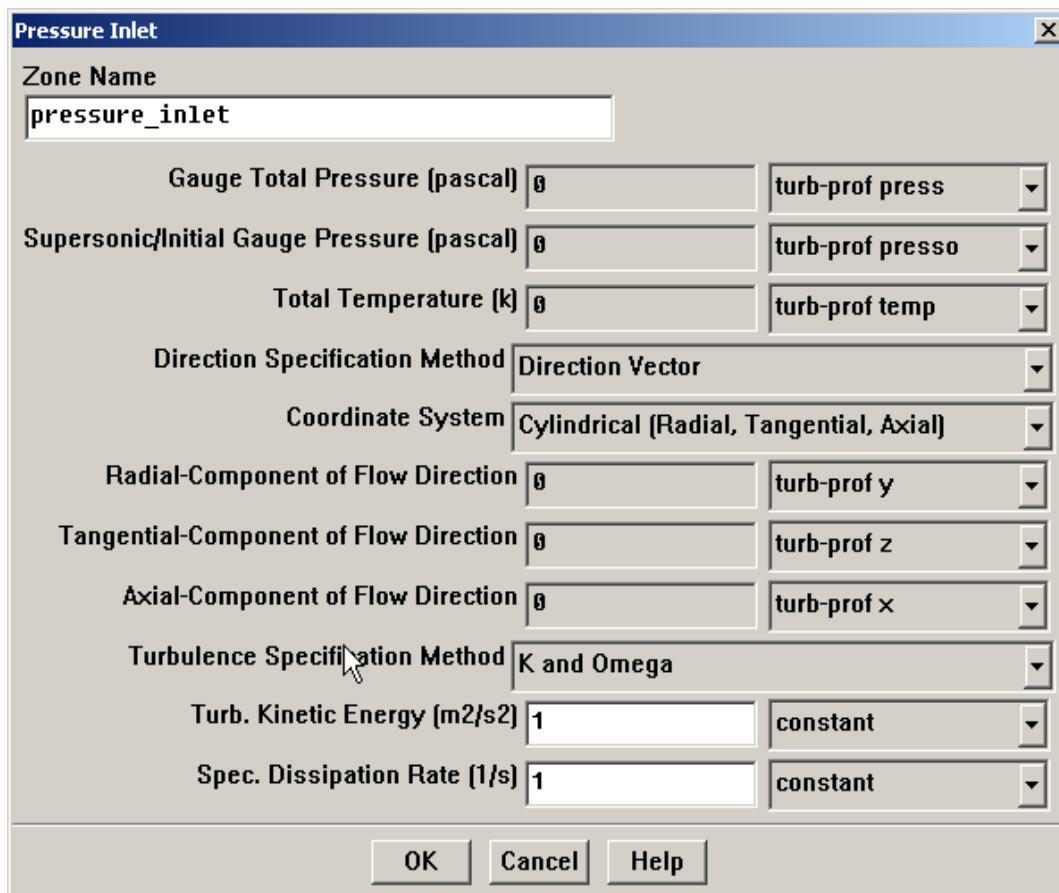


Fig. 4.8 Pressure inlet properties window

#### 4.4.6 Outlet pressure

Similar way, generated profiles for gauge pressure and back flow total temperature has been used for outer pressure boundary conditions.

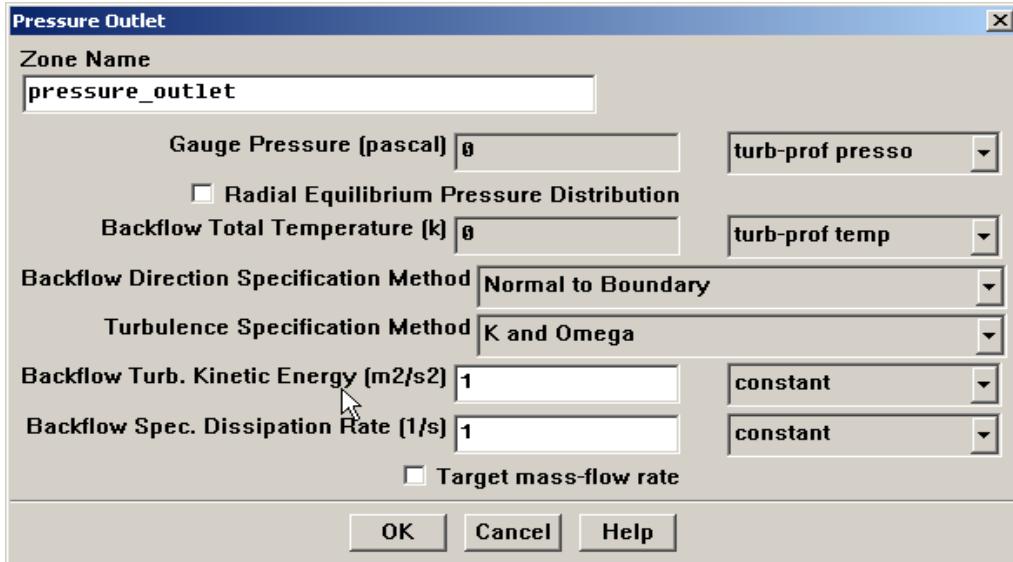


Fig. 4.9. Pressure outlet properties window

#### 4.4.7 Default interior

Default interior is considered as fan, swirl velocity is also considered in the case and further options have been set as show in figure and pressure jump is used on average conditions.

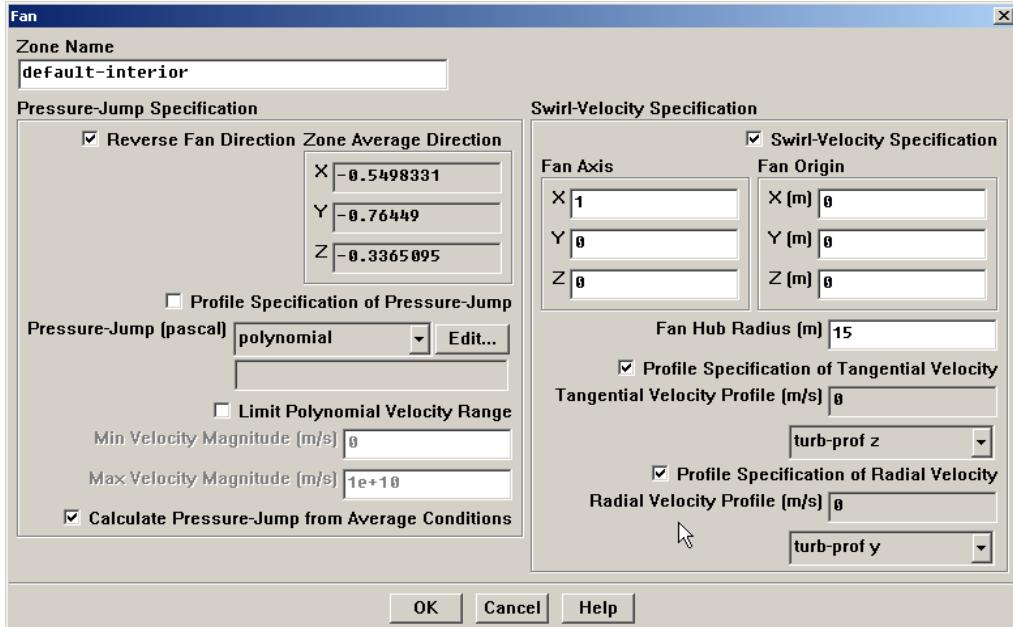


Fig. 4.10 Default interior properties windows

#### 4.4.8 Solutions controls

When a case is to be solved, solutions controls are very important, wrong selection of control schemes can lead adverse and not interesting results. There are several solutions schemes are available for segregated solver and coupled solver. After reading all sections about solutions controls facts, following equations set, relaxation factors, pressure velocity coupling. But here certain solutions controls will be discussed those are used in this case.

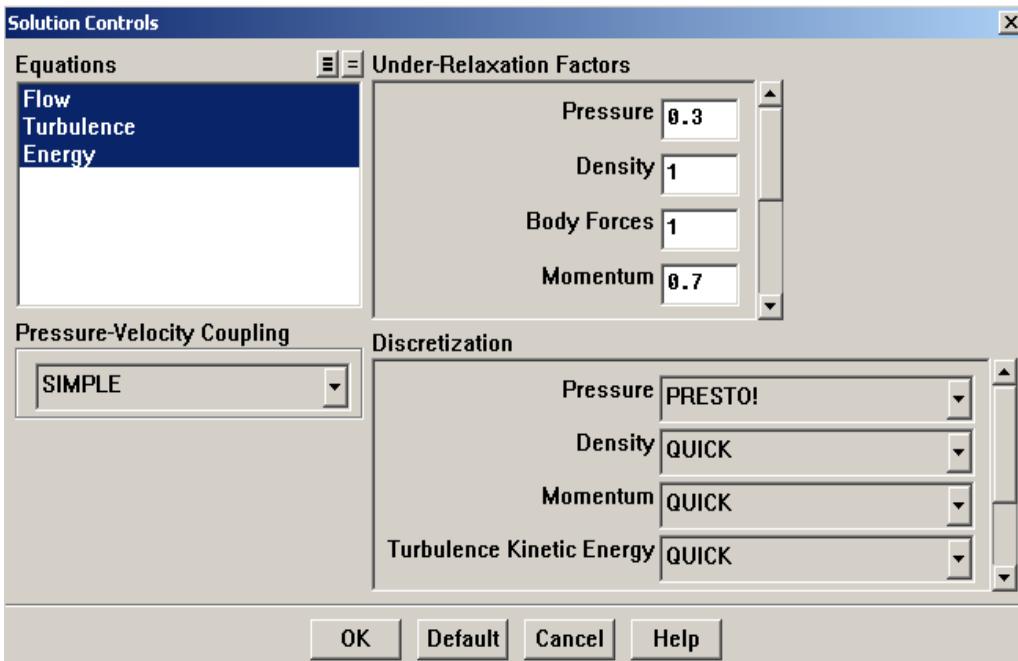


Fig. 4.11 Solutions controls properties window

##### 4.4.8.1 Under Relaxation Factors.

These factors control the calculations of computed variables during the each iteration. Either higher or lower values can give tiring result, might higher values give solver failures i.e. solution is diverged or etc., and might lower values can take long time to get good residuals. It is recommended to use the default under-relaxation factors. If the residuals continue to increase, relaxation factors can be decreased accordingly.

##### 4.4.8.2 Pressure-Velocity Coupling

Pressure-velocity coupling can be read for details by using section 26.3-5 from Fluent user guide for further details. FLUENT gives four options for pressure-velocity coupling algorithms i.e., SIMPLE, SIMPLEC, PISO, and (for unsteady flows using the non-iterative time advancement scheme (NITA)) Fractional Step (FSM).

#### 4.4.8.3 SIMPLE

SIMPLE is a fluent default scheme. The SIMPLE algorithm utilizes an interaction between velocity and pressure corrections to enforce mass conservation and to obtain the pressure field.

#### 4.4.8.4 Discretization

The PRESTO! (PRESto! STaggering Option) scheme can be used for all kind of meshes. This scheme is recommended for swirl flows, high-Raleigh-number natural convection, high-speed rotating flows, flows in strongly curved, and flows involving porous domains. This scheme is highly recommended for the steep pressure gradients with swirling flows.

The QUICK discretization scheme can give more accurate results than the second-order scheme for rotating or swirling flows solved on quadrilateral or hexahedral meshes. When a compressible flow with shocks is to be calculated, the first-order upwind scheme may tend to smooth the shocks; but use of the second-order-upwind or QUICK scheme for such flows is recommended. For compressible flows with shocks, using the QUICK scheme for all variables, including density, is highly useful for quadrilateral, hexahedral, or hybrid meshes.

### 4.5 Solution Initialisation

The solution was initialised with gauge pressure 153869.6 pa, x velocity 35 m/sec, temperature 306.0071 K and rest parameters were used as default. The case has been iterated up to 1000 steps and residuals can be seen in below figure.

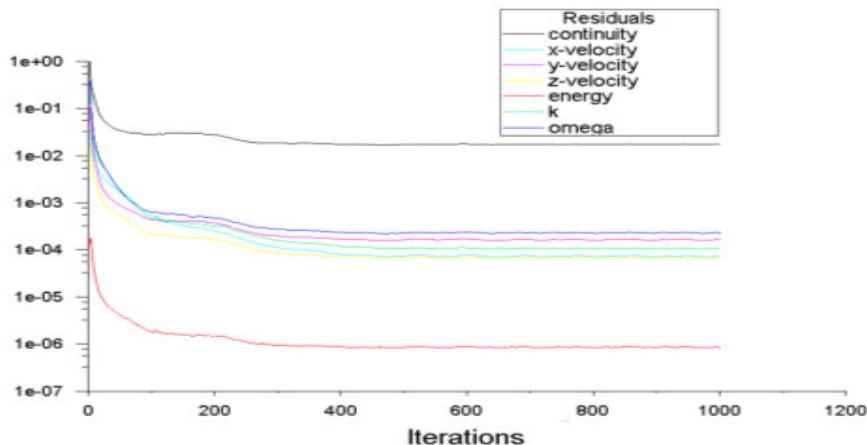


Fig.4.12 View of Residuals

Sometimes, solution can't be converged and it is believed that if all residuals are not changing versus iterations solution could be reliable, acceptable for further simulations and calculations.

## Chapter 5

### 5 Simulations and numerics

#### 5.1 Simulations and numerical Results

The simulation and numerical results for airflow over an airfoil is very important for aerospace applications and research and development department. For example below static pressure contours show the stagnation region near to leading edge of airfoil and as well as show the low-pressure region.

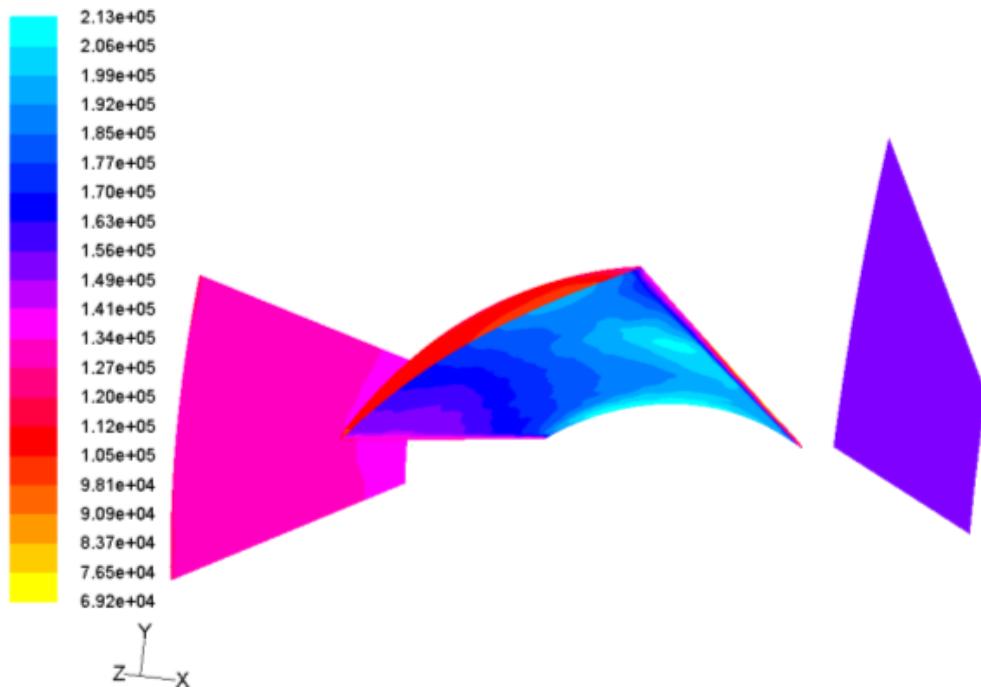


Fig. 5.1Contours of static pressure on airfoil of lower side of airfoil

The highest static pressure can be seen at airfoil near the hub, The pressure developed in region near hub, is to be released in region near to shroud can be seen in fig. 5.1.

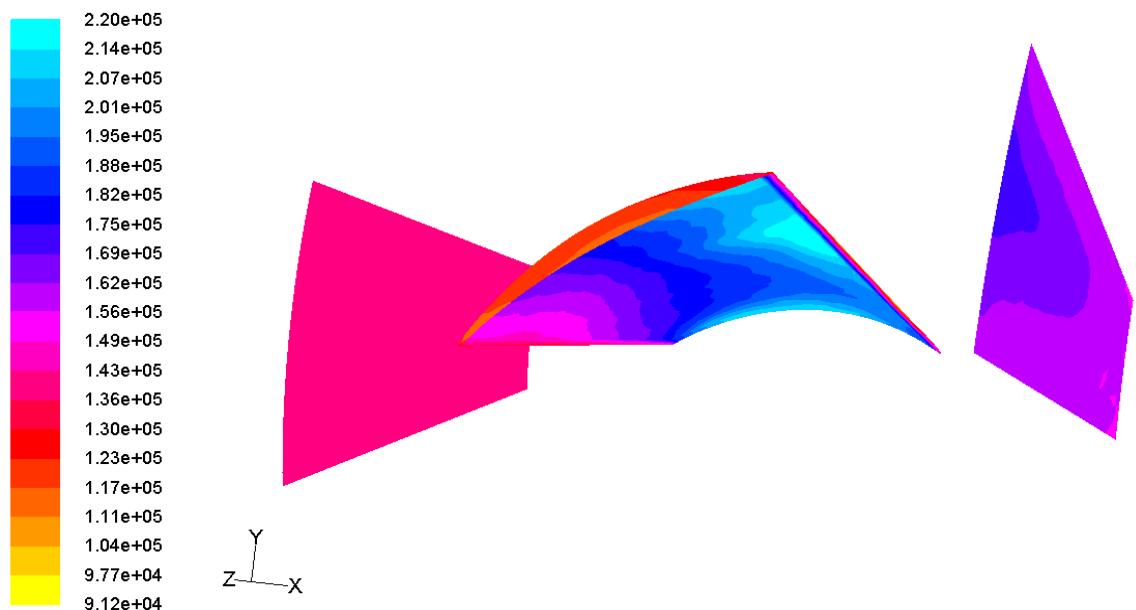


Fig. 5.2. Contours of total pressure on airfoil of inner side of airfoil

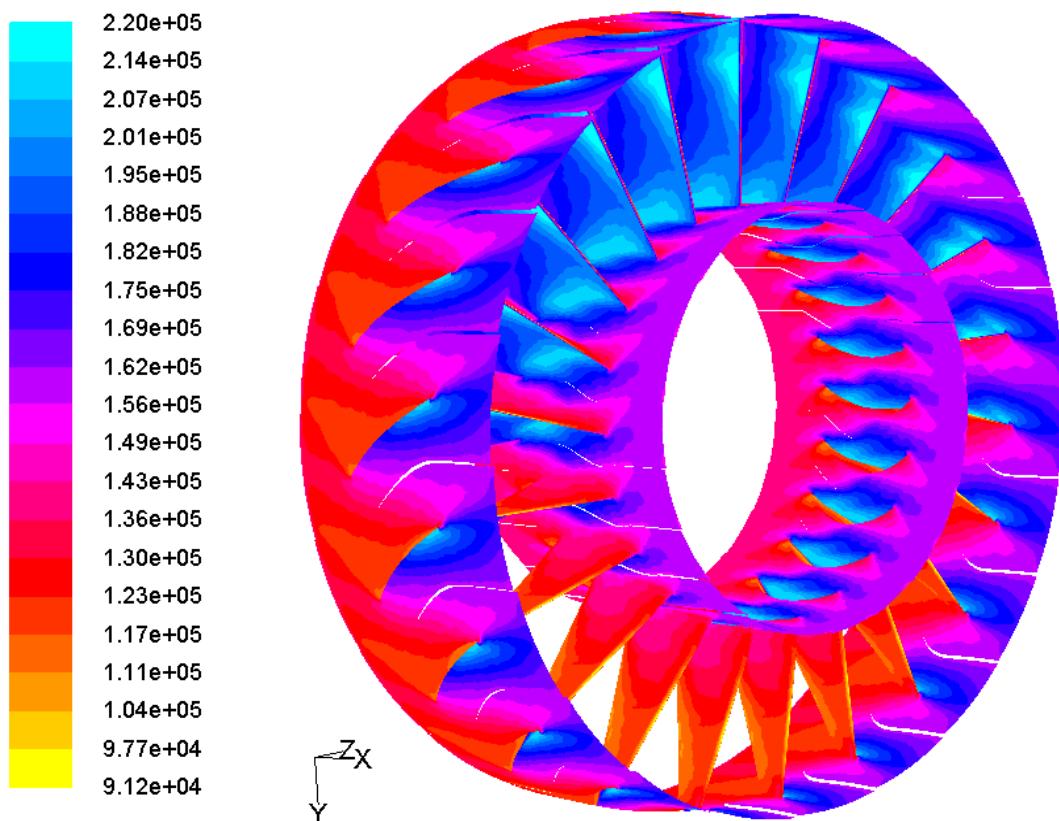


Fig. 5.3. Contours of total pressure on a rotor

The higher total pressure can be seen at the inner side of airfoil in the rotational direction and near the outlet of the shroud. The higher total pressure developed in the region near the hub, is to be transferred in the region near to the shroud and outlet as well as compression can be seen in fig. 5.2 and fig. 5.3

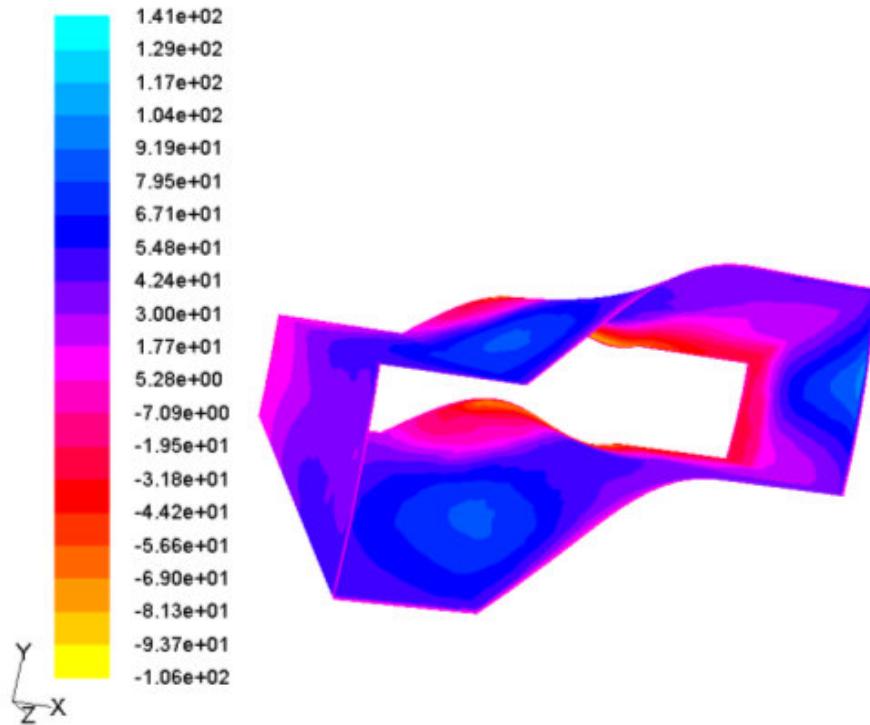


Fig. 5.4. Contours of axial velocity at inlet, outlet and periodic interface

The contours of axial velocity can be seen at different locations in above figure. The negative values near to the hub and the outlet region shows reverse flow cause due to pressure different at the two different locations as well as shape of airfoil section on the hub.

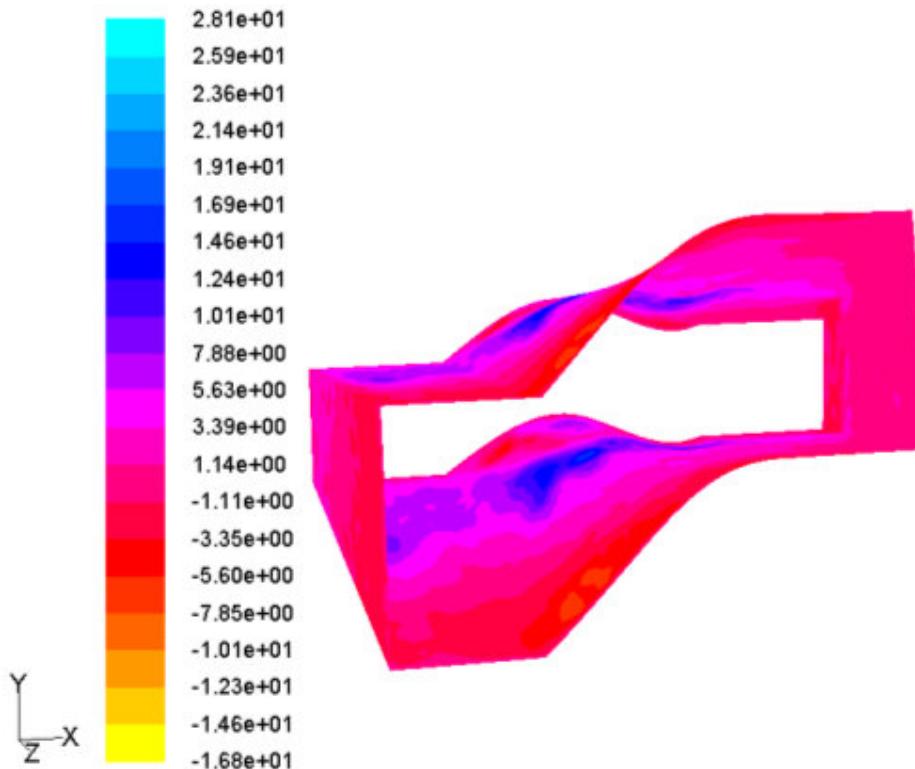


Fig. 5.5. Contours of radial velocity at inlet, outlet and periodic interface.

The contours of the radial velocity can be seen at the different locations in above the figure. The negative values shows air streams are moving near to hub and positive values means air streams are moving away from the hub.

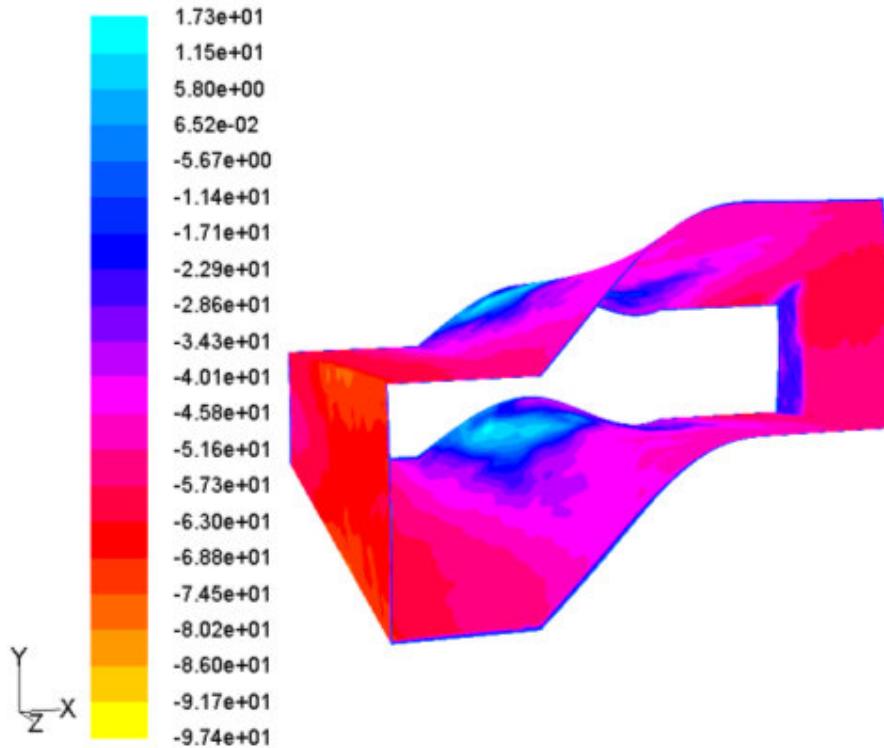


Fig. 5.6. Contours of tangential velocity at inlet, outlet and periodic interface

The contours of the tangential velocity can be seen at the different locations in above figure. The negative values shows air streams are moving near to hub and positive values means air streams are moving away from the hub

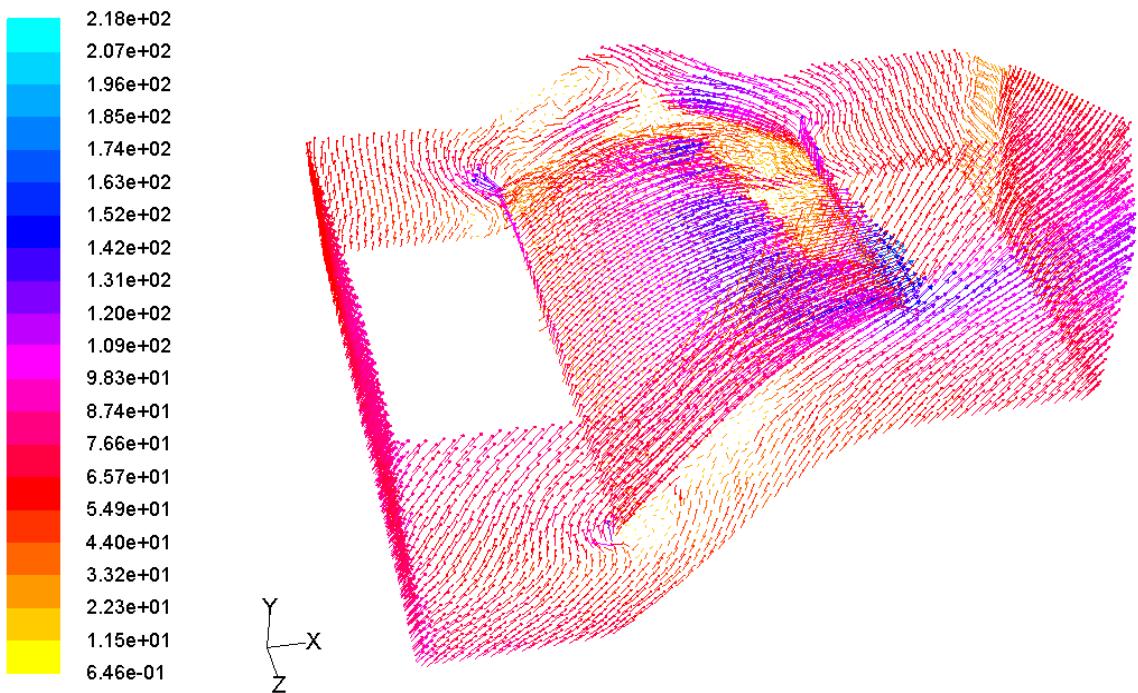


Fig. 5.6. Contours of magnitude of air velocity vectors

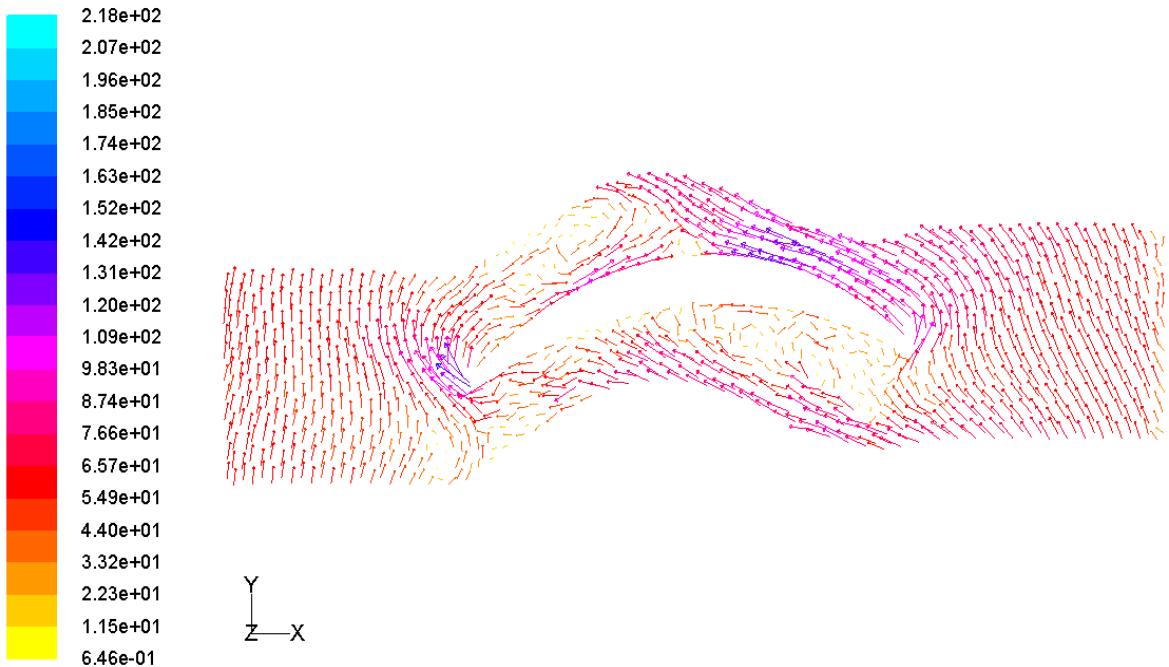


Fig. 5.7. Contours of magnitude of air velocity vectors.

The velocity vectors have been shown at the inlet, the outlet, the hub and the shroud as shown in Fig. 5.6. But in fig. 5.7 the effects of reversed flow can be seen. As an airfoil is revolving and compressing air at the inner side of airfoil causes the lower

pressure at the ends and the opposite sides in result the high velocity vectors can be seen due to the low pressure.

## **6 Conclusions and Discussions**

The link/program has been developed successfully to handshake between CAD and CFD software in Matlab and has been tested with several sets of data. A parameterised 3D model of a NASA rotor 67 has been produced and has been verified by using boundary conditions, the results were satisfactory but some drawbacks have been noticed. The program (make\_gambit\_journal.m) generates big journal file for Gambit for example when the rotor will be made from the two data files of blades, it will generate about 20 pages journal file, if rotor will be made for several layers data, it will generate large journal file could cross the limits to execute such a large file or extremely slow down the execution of journal file. It has been observed by this work, a 3D rotor with meshed domain composed from two or three blade data files is easily manageable. By considering more layers could complex the geometry or produce a bad geometry might crash the Gambit or domain can't be produced with mesh specially when blade is cut by variable radial profile. As well as rotor blade data and parameters are critically important to produce a 3D rotor with meshed domain, if some discrepancy found in the blade data or the radial distance is too near between consecutive layers after cutting the blade with variable radial profile Gambit may crash or does not produce a meshed domain. The issue could be solved looking the set of data and parameters or by setting faces with certain tolerance. Otherwise the simulations could be carried out by dividing the whole set of blade data into subsets.

These developed programs and approaches can be for further used for advanced level research (Masters' level or PhD level) i.e. shape and flow optimisation or any kind of blade simulations research.

## References

1. John S. Anagnostopoulos, “A numerical simulation methodology for hydraulic turbomachines”, 5th GRACM International Congress on Computational Mechanics Limassol, 29 June – 1 July, 2005.
2. Z.Q.Zhu, X.L.Wang, H.Y.Fu, and H.Liu, “Aerodynamic optimisation design of airfoil and wing”, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimisation Conference AIAA 2004-4367 30 August - 1 September 2004, Albany, New York.
3. A. Vicini, and D. Quagliarella “Airfoil and Wing Design through hybrid optimisation strategies”, AiAA 98-2729.
4. Michael Casey, Sulzer Innotec, Switzerland, “Best practice advice for cfd in turbo machinery design” QNET-CFD Network Newsletter, Volume 2, No. 3 – December 2003.
5. Miklos Gerendas and Ralph pfister “Development of a very small aero Engine”, Proceedings of the ASME Turbo Expo 2000, 45<sup>th</sup> ASME international gas turbine and aero engine Technical congress and exposition May 8-11, 2000, Munich, Germany.
6. Guido Dietz , Ralph Voß , and Roeland De Breuker, “Airfoil optimisation based on an evolution strategy With respect to aero-elasticity”, European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 24–28 July 2004.
7. Martin Rothand and Ronald Peikert, “Flow Visualization for Turbo-machinery Design”, Proceedings of the 7th IEEE Visualization Conference (VIS’96).
8. U. Schluter, X. Wu, S. Kim, J. J. Alonso, and H. Pitsch., “Integrated RANS-LES computations of turbo-machinery components Generic compressor/diffuser”, Center for Turbulence Research, Annual Research Briefs 2003.
9. Rolf Dornberger, Dirk Büche and Peter Stoll, “Multidisciplinary optimisation in turbo-machinery design”, European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2000, Barcelona, 11-14 September 2000.
10. Jean-Antoine, D'esid'eri and AlevsJanka, “Multilevel shape parameterisation for aerodynamic optimisation—application to drag and noise reduction of transonic/supersonic business jet”, European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2004 Jyvaskyla, 24–28July2004.
11. Ihor S. Diakunchak, Greg R. Gaul, Gerry McQuiggan and Leslie R. Southall, “Siemens westinghouse advanced turbine systems program final summary”, Proceedings of ASME TURBO EXPO 2002 June 3-6, 2002, Amsterdam, The Netherlands.
12. G. Grondin, V. Kelner, P. Ferrand, and S. Moreau, “ Robust Design and parametric performance study of an automotive fan blade by coupling multi-objective genetic optimisation and flow parameterisation”

13. S.Pierret, “Multi-objective and Multi-Disciplinary Optimisation of Three-dimensional Turbo-machinery Blades”, 6<sup>th</sup> World Congresses of Structural and Multi disciplinary Optimisation” Rio de Janeiro, 30 May – 03 June 2005, Brazil
14. Anthony J. Strazisar, Jerry R. Wood, Michael D. Hathaway, and Kenneth L. Suder, “laser anemometer measurement in a transonic axial flow fan rotor” NASA technical paper 2879, November 1989.
15. Hans Mårtensson, Jörgen Burman and Ulf Johansson, “Design of the high pressure ratio transonic 1.5 stage fan demonstration hulda”, Proceeding of GT 2007, ASME Turbo expo. Power for land, sea and air, May 14-17, 2007, Montreal Canada.
16. Althea De Souza, “How to understand computational fluid dynamics jargon”, 2003 NAFEMS LTD.
17. Dr. Jan Roskam and Dr. Chuan Tau Edward Lan, “Airplane aerodynamics and performance”, Published by Design, analysis and research corporation, 120 east ninth street, suite 2, Lawrence Kansas 66044, USA.
18. “Pilot’s handbook of Aeronautical Knowledge” Published by U.S. Department of transportation. Federal aviation administration Flight Standards Service
19. FLUENT 6.2 Documentation
20. Gambit 2.2.30 Documentation
21. Matlab Documentation

## A. Appendix 1

This is a program written in Matlab for 2D airfoil and it generates journal file, can be read in gambit.

```

function make_2Dgambit_journal(nr,a,b,delta_x1,delta_x2,y,alpha,beta,r)
%   MAKE_2DGAMBIT_JOURNAL   Make a journal file for Gambit
%   Input
%       nr: the simulation number
%       a : % number of nodes away from xminindex at above the
outer
%       profile
%       b : % number of nodes away from xminindex at below the
outer
%       profile
%       delta_x1 : certain distance from leading edge.
%       delta_x2 : certain distance from trialing edge.
%       y :perpendicular distance between to periodic boundaries.
%       alpha : % inflow angle
%       beta : % outflow angle
%       nob : % Number of blades
%       r : radial distance
% Copyright: © AA Narejo © David Lindström
% Author: AA Narejo & David Lindström
% University of Trollhättan Uddevalla
% 28/2-2007

% Default values
if nargin<1, nr=1;
a=1; % one above of minimum point
b=2; % 3rd point will be selected w.r.t minimum point.
delta_x1=2;
delta_x2=2;
alpha=0.1;
beta=0.2 ;
delta_y1= delta_x1 * tan (alpha);
delta_y2= delta_x2 * tan (beta);
r=15;
nob=22;
y= 2*pi*r / nob;
end

% Open the new journal file i.e. gambit_1 if nr=1
fid,message=fopen(['GMT2D',num2str(nr),'.jou'],'w');
if ~fid,disp([message]),end
fprintf(fid,'%s\n','reset');
fprintf(fid,'%s\n','/ Journal File for GAMBIT 2.2.30, Database
2.2.14, lnx86 BH04110220');
fprintf(fid,'%s\n',['identifier name "naca_',num2str(nr),'" new
saveprevious']);
fprintf(fid,'%s\n','solver select "FLUENT 5/6"');

% import data from files
fprintf(fid,'%s\n',['import vertexdata
"naca_',num2str(nr),'.inner"']);
fprintf(fid,'%s\n',['import vertexdata
"naca_',num2str(nr),'.outer"']);

% Fetch coordinates where xminindex lies outer profile.

```

```

coxy=edgeindexc(['naca_',num2str(nr),'.outer'],1);
fprintf(fid,'%s\n',[ 'vertex create "vertex.97" coordinates ' 
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.98" coordinates ' 
[num2str(coxy(1,1)-delta_x1) ' ' num2str(coxy(1,2)+y/2-delta_y1) ' 
0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.106" coordinates ' 
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)-y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.107" coordinates ' 
[num2str(coxy(1,1)-delta_x1) ' ' num2str(coxy(1,2)-y/2-delta_y1) ' 
0']]);
% Fetch coordinates where xmaxindex lies at outer profile.
coxy=edgeindexc(['naca_',num2str(nr),'.outer'],2);
fprintf(fid,'%s\n',[ 'vertex create "vertex.99" coordinates ' 
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.105" coordinates ' 
[num2str(coxy(1,1)+delta_x2) ' ' num2str(coxy(1,2)+y/2-delta_y2) ' 
0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.108" coordinates ' 
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)-y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.109" coordinates ' 
[num2str(coxy(1,1)+delta_x2) ' ' num2str(coxy(1,2)-y/2-delta_y2) ' 
0']]);
% Fetch coordinates where xmaxindex lies at inner profile.
conxyave=(nodeco(['naca_',num2str(nr),'.inner'],1)+
nodeco(['naca_',num2str(nr),'.inner'],47))/2 ;
fprintf(fid,'%s\n',[ 'vertex create "vertex.102" coordinates ' 
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.111" coordinates ' 
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)-y/2) ' 0']]);
% Fetch coordinates of n node at inner profile then takes as
average of co-ordinates.
conxyave=(nodeco(['naca_',num2str(nr),'.inner'],12)+
nodeco(['naca_',num2str(nr),'.inner'],36))/2 ;
fprintf(fid,'%s\n',[ 'vertex create "vertex.101" coordinates ' 
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.110" coordinates ' 
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)-y/2) ' 0']]);
fprintf(fid,'%s\n','vertex create "cut above" coordinates 0.45 0.4
0');
fprintf(fid,'%s\n','vertex create "cut below" coordinates 0.45 0
0');

% Creates edges from vertices
fprintf(fid,'%s\n','edge create "airfoil te" straight "vertex.47"
"vertex.1"');
fprintf(fid,'%s\n','edge create "outer te" nurbs "vertex.96"
"vertex.48" "vertex.49"');
fprintf(fid,'%s\n','edge create "up right vertex" straight
"vertex.1" "vertex.49"');
fprintf(fid,'%s\n','edge create "lo right vertex" straight
"vertex.47" "vertex.96"');
fprintf(fid,'%s\n','edge create "edge above right" straight
"vertex.105" "vertex.99"');
xminind=edgeindex(['naca_',num2str(nr),'.outer'])+47;
xminindi=edgeindex(['naca_',num2str(nr),'.inner']);

```

```

fprintf(fid,'%s\n',[ 'edge create "up left vertex" straight
"vertex.' num2str(xminindi-a) '' ' ' "vertex.' num2str(xminind-
a) ''']);
fprintf(fid,'%s\n',[ 'edge create "lo left vertex" straight
"vertex.' num2str(xminindi+b) '' ' ' "vertex.' 
num2str(xminind+b) ''']);
conxy=nodeco(['naca_',num2str(nr),'.outer'],2);
fprintf(fid,'%s\n',[ 'vertex create "on the te" coordinates '
[num2str(coxy(1,1)+delta_x2) ' ' num2str(conxy(1,2)-delta_y2) ' 
0']]);
conxy=nodeco(['naca_',num2str(nr),'.outer'],xminind-a-47);
coxy=edgeindex(['naca_',num2str(nr),'.outer'],1);
fprintf(fid,'%s\n',[ 'vertex create "on the le" coordinates '
[num2str(coxy(1,1)-delta_x1) ' ' num2str(conxy(1,2)-delta_y1) ' 
0']]);
fprintf(fid,'%s\n','edge create "edge on the te" straight
"vertex.49" "on the te"');
fprintf(fid,'%s\n',[ 'edge create "edge on the le" straight
"vertex.' num2str(xminind-a) '' "on the le"']);
fprintf(fid,'%s\n','edge create "edge above mid" nurbs "vertex.99"
"vertex.102" "vertex.101" \');
fprintf(fid,'%s\n',' "vertex.97"');
fprintf(fid,'%s\n','edge create "edge above left" straight
"vertex.97" "vertex.98"');
fprintf(fid,'%s\n','edge create "edge below right" straight
"vertex.109" "vertex.108"');
fprintf(fid,'%s\n','edge create "edge below mid" nurbs
"vertex.108" "vertex.111" "vertex.110" \');
fprintf(fid,'%s\n',' "vertex.106"');
fprintf(fid,'%s\n','edge create "edge below left" straight
"vertex.106" "vertex.107"');
fprintf(fid,'%s\n','edge create "vertex up right" straight
"vertex.99" "vertex.49"');
fprintf(fid,'%s\n','edge create "vertex lo right" straight
"vertex.108" "vertex.96"');
fprintf(fid,'%s\n','edge create "inlet 1" straight "vertex.98" "on
the le"');
fprintf(fid,'%s\n','edge create "inlet 2" straight "on the le"
"vertex.107"');
fprintf(fid,'%s\n','edge create "outlet 1" straight "vertex.105"
"on the te"');
fprintf(fid,'%s\n','edge create "outlet 2" straight "vertex.109"
"on the te"');
xminind=edgeindex(['naca_',num2str(nr),'.outer'])+47;
xminindi=edgeindex(['naca_',num2str(nr),'.inner']);
fprintf(fid,'%s\n',[ 'edge create "vertex up left" straight
"vertex.97" ' ' "vertex.' num2str(xminind-a) ''']);
fprintf(fid,'%s\n',[ 'edge create "vertex lo left" straight
"vertex.106" ' ' "vertex.' num2str(xminind+b) ''']);

% new geometry of outer lower profile
xminindv=xminind+b;
fprintf(fid,'%s\n','edge create "outer bottom" nurbs \');
for xminindv=xminindv:95
fprintf(fid,'%s\n',[ '"vertex.' num2str(xminindv) '"\']);
end
fprintf(fid,'%s\n','"vertex.96"');

% new geometry of outer upper profile

```

```

xminindv=xminind-a;
fprintf(fid,'%s\n','edge create "outer top" nurbs \');
for xminindv=xminindv:-1:50
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) '\']]);
end
fprintf(fid,'%s\n','"vertex.49"');

% new geometry of outer le profile
xminindv=xminind-a;
fprintf(fid,'%s\n','edge create "outer le" nurbs \');
for xminindv=xminindv:xminind+b-1
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) '\']]);
end
fprintf(fid,'%s\n',[ ' "vertex.' num2str(xminind+b) '""' ]);

% new geometry of inner lower profile
xminindv=xminindi+b;
fprintf(fid,'%s\n','edge create "airfoil bottom" nurbs \');
for xminindv=xminindi+b:46
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) '\']]);
end
fprintf(fid,'%s\n','"vertex.47"');

% new geometry of outer upper profile
xminindv=xminindi-a;
fprintf(fid,'%s\n','edge create "airfoil top" nurbs \');
for xminindv=1:xminindv-1
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) '\']]);
end
fprintf(fid,'%s\n',[ ' "vertex.' num2str(xminindi-a) '""']);

% new geometry of outer airfoil le profile
xminindv=xminindi-a;
fprintf(fid,'%s\n','edge create "airfoil le" nurbs \');
for xminindv=xminindv:xminindi+b-1
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) '\']]);
end
fprintf(fid,'%s\n',[ ' "vertex.' num2str(xminindi+b) '""' ]);

% Creation of faces from edges
fprintf(fid,'%s\n','face create "face airfoil above" wireframe
"outer top" "up left vertex" \');
fprintf(fid,'%s\n',' "airfoil top" "up right vertex" real');
fprintf(fid,'%s\n','face create "face airfoil below" wireframe
"airfoil bottom" "lo left vertex" \');
fprintf(fid,'%s\n',' "outer bottom" "lo right vertex" real');
fprintf(fid,'%s\n','face create "face airfoil le" wireframe "up
left vertex" "outer le" \');
fprintf(fid,'%s\n',' "lo left vertex" "airfoil le" real');
fprintf(fid,'%s\n','face create "face airfoil te" wireframe "up
right vertex" "airfoil te" \');
fprintf(fid,'%s\n',' "lo right vertex" "outer te" real');
fprintf(fid,'%s\n','face create "face outer above" wireframe "edge
above mid" "vertex up left" \');
fprintf(fid,'%s\n',' "outer top" "vertex up right" real');
fprintf(fid,'%s\n','face create "face outer below" wireframe
"outer bottom" "vertex lo left" \');
fprintf(fid,'%s\n',' "edge below mid" "vertex lo right" real');

```

```

fprintf(fid,'%s\n','face create "face outer le 1" wireframe "inlet
1" "edge on the le" \'');
fprintf(fid,'%s\n','"vertex up left" "edge above left" real');
fprintf(fid,'%s\n','face create "face outer le 2" wireframe "inlet
2" "edge on the le" "outer le" \'');
fprintf(fid,'%s\n','"vertex lo left" "edge below left" real');
fprintf(fid,'%s\n','face create "face outer te 1" wireframe
"vertex up right" "edge above right" \'');
fprintf(fid,'%s\n','"outlet 1" "edge on the te" real');
fprintf(fid,'%s\n','face create "face outer te 2" wireframe
"vertex lo right" "outer te" \'');
fprintf(fid,'%s\n','"edge on the te" "outlet 2" "edge below right"
real');

% edges are linked for periodic
fprintf(fid,'%s\n','edge link "edge above right" "edge below
right" directions 0 0');
fprintf(fid,'%s\n','edge link "edge above mid" "edge below mid"
directions 0 0');
fprintf(fid,'%s\n','edge link "edge above mid" "outer top"
directions 0 0');
fprintf(fid,'%s\n','edge link "edge above mid" "outer bottom"
directions 0 0');
fprintf(fid,'%s\n','edge link "edge above left" "edge below left"
directions 0 0');
fprintf(fid,'%s\n','edge link "up right vertex" "lo right vertex"
directions 0 0');
fprintf(fid,'%s\n','edge link "up right vertex" "up left vertex"
directions 0 0');
fprintf(fid,'%s\n','edge link "up right vertex" "lo left vertex"
directions 0 0');

% edges are meshed
fprintf(fid,'%s\n','edge mesh "airfoil top" successive ratio1 0.98
intervals 42');
fprintf(fid,'%s\n','edge mesh "airfoil le" successive ratio1 0.98
intervals 6');
fprintf(fid,'%s\n','edge mesh "airfoil bottom" successive ratio1
1.02 intervals 42');
fprintf(fid,'%s\n','edge mesh "airfoil te" successive ratio1 1
intervals 5');
fprintf(fid,'%s\n','edge mesh "outer top" successive ratio1 1
intervals 42');
fprintf(fid,'%s\n','edge mesh "outer le" successive ratio1 0.95
intervals 6');
fprintf(fid,'%s\n','edge mesh "outer te" successive ratio1 1
intervals 5');
fprintf(fid,'%s\n','edge mesh "up right vertex" successive ratio1
1.2 intervals 8');
fprintf(fid,'%s\n','edge mesh "edge above right" successive ratio1
0.96 intervals 12');
fprintf(fid,'%s\n','edge mesh "edge on the te" successive ratio1
0.96 intervals 12');
fprintf(fid,'%s\n','edge mesh "edge above left" successive ratio1
1.06 intervals 14');
fprintf(fid,'%s\n','edge mesh "edge on the le" successive ratio1
1.06 intervals 14');
fprintf(fid,'%s\n','edge mesh "vertex up right" successive ratio1
0.85 intervals 8');

```

```

fprintf(fid,'%s\n','edge mesh "outlet 1" successive ratio1 0.85
intervals 8');
fprintf(fid,'%s\n','edge mesh "vertex lo right" successive ratio1
0.85 intervals 7');
fprintf(fid,'%s\n','edge mesh "vertex up left" successive ratio1
0.85 intervals 8');
fprintf(fid,'%s\n','edge mesh "inlet 1" successive ratio1 0.85
intervals 8');
fprintf(fid,'%s\n','edge mesh "vertex lo left" successive ratio1
0.85 intervals 7');
fprintf(fid,'%s\n','edge mesh "inlet 2" successive ratio1 1
intervals 13');
fprintf(fid,'%s\n','edge mesh "outlet 2" successive ratio1 1
intervals 12');

% Creation of face meshes.
fprintf(fid,'%s\n','face mesh "face airfoil above" map size 1');
fprintf(fid,'%s\n','face mesh "face airfoil below" map size 1');
fprintf(fid,'%s\n','face mesh "face airfoil le" map size 1');
fprintf(fid,'%s\n','face mesh "face airfoil te" map size 1');
fprintf(fid,'%s\n','face mesh "face outer above" map size 1');
fprintf(fid,'%s\n','face mesh "face outer below" map size 1');
fprintf(fid,'%s\n','face mesh "face outer le 1" "face outer le 2"
map size 1');
fprintf(fid,'%s\n','face mesh "face outer te 1" "face outer te 2"
map ');

% Boundary settings
fprintf(fid,'%s\n','physics create "airfoil" btype "WALL" edge
"airfoil top" "airfoil le" \'');
fprintf(fid,'%s\n',' "airfoil bottom" "airfoil te"\'');
fprintf(fid,'%s\n','physics create "inlet" btype "PRESSURE_INLET"
edge "inlet 1" "inlet 2"\'');
fprintf(fid,'%s\n','physics create "outlet" btype
"PRESSURE_OUTLET" edge "outlet 1" "outlet 2"\'');
fprintf(fid,'%s\n','physics create "periodic left" btype
"PERIODIC" edge "edge above left" \'');
fprintf(fid,'%s\n',' "edge below left"\'');
fprintf(fid,'%s\n','physics create "periodic mid" btype "PERIODIC"
edge "edge above mid" \'');
fprintf(fid,'%s\n',' "edge below mid"\'');
fprintf(fid,'%s\n','physics create "periodic right" btype
"PERIODIC" edge "edge above right" \'');
fprintf(fid,'%s\n',' "edge below right"\'');
fprintf(fid,'%s\n','physics create "air" ctype "FLUID" face "face
airfoil above" \'');
fprintf(fid,'%s\n',' "face airfoil below" "face airfoil le" "face
airfoil te" "face outer above" \'');
fprintf(fid,'%s\n',' "face outer below" "face outer te 1" "face
outer te 2" "face outer le 2" "face outer le 1"\'');
fprintf(fid,'%s\n','vertex delete "cut below" "cut above"\'');
fprintf(fid,'%s\n',[ 'export fluent5 "fl2d',num2str(nr),'.msh"
nozval']);
fprintf(fid,'%s\n','save');
st=fclose(fid);

```

```
function xminind=edgeindex(filename)
% EDGEINDEX(filename) The function reads the data in file
"filename"
% and finds the index with smallest x-coordinate
% Input
% FILENAME: textstring with the filename
% Output
% XMININD: the appropriate index in the blade coordinate array
% Copyright: © David Lindström
% Authors: AA Narejo and David
% University of Trollhättan Uddevalla
% 20/1-2007

% Open the file with coordinates
[fid,message]=fopen(filename,'r');
if ~fid,disp(message),end

% Read the first row (number of coordinates)
nr=str2num(fgets(fid));

% Read the coordinates
s=fscanf(fid, '%g %g', [3 inf])';

% Find the smallest x coordinate and its index
xmin=min(s(:,1));
xminind=find(xmin==s(:,1));

% Check the number of coordinates
if nr~=size(s,1)
    disp('Number of coordinates is wrong')
    xminind=0;
end

fclose(fid);

function conxy=nodeco(filename,cno)
% NODECO(filename,x) The function reads the co-ordinates of
certain node from file "filename"
% Input
% FILENAME: textstring with the filename
% CNO: index number of node, to be entered as integer.
% Output
% CONXY: Get co-ordinate of any node point
% Copyright: © David Lindström
% Authors: AA Narejo and David
% University of Trollhättan Uddevalla
% 20/1-2007
% for minimum values

% Open the file with coordinates
[fid,message]=fopen(filename,'r');
if ~fid,disp(message),end

% Read the first row (number of coordinates)
nr=str2num(fgets(fid));

% Read the coordinates
s=fscanf(fid, '%g %g', [3 inf])';
```

```
% Find x & y co-ordinates and its index
xcor=s(cno,1);
ycor=s(cno,2);
conxy=[xcor,ycor];

% Check the total number of coordinates
if nr~=size(s,1)
    disp('Number of coordinates is wrong')
    xcnoind=0;
    end
fclose(fid);

function coxy=edgeindexc(filename,x)
% EDGEINDEX(filename) The function reads the data from file
"filename"
% and finds the index with smallest x-coordinate
% Input
% FILENAME: text string with the filename
% if x=1 calculate minimum values, x=2 calculate maximumum values
% and otherwise it will return nothing.
% Output
% COXY: fetch minimum co-ordinates if x=1 and
% maximum co-ordinates if x=2
% Copyright: © David Lindström
% Authors: AA Narejo and David
% University of Trollhättan Uddevalla
% 20/1-2007
% for minimum values

if x==1

    % Open the file with coordinates
    [fid,message]=fopen(filename,'r');
    if ~fid,disp(message),end

    % Read the first row (number of coordinates)
    nr=str2num(fgets(fid));

    % Read the coordinates
    s=fscanf(fid, '%g %g', [3 inf])';

    % Find the smallest x & y coordinate and its index
    xmin=min(s(:,1));
    xminind=find(xmin==s(:,1));
    xcor=s(xminind,1);
    ycor=s(xminind,2);
    coxy=[xcor,ycor];

    % Check the number of coordinates
    if nr~=size(s,1)
        disp('Number of coordinates is wrong')
        xminind=0;
        end
    fclose(fid);
    end

    % for maximum values
    if x==2
```

```
% Open the file with coordinates
fid,message=fopen(filename,'r');
if ~fid,disp(message),end

% Read the first row (number of coordinates)
nr=str2num(fgets(fid));

% Read the coordinates
s=fscanf(fid, '%g %g', [3 inf])';

% Find the highest x & y coordinate and its index
xmax=max(s(:,1));
xmaxind=find(xmax==s(:,1));
xcor=s(xmaxind,1);
ycor=s(xmaxind,2);
coxy=[xcor,ycor];

% Check the number of coordinates
if nr~=size(s,1)
    disp('Number of coordinates is wrong')
    xmaxind=0;
end
fclose(fid);
end
```

```
Program for outer profile generator
function [Xo,Yo,Zo] =in_out_prf_gen(filename,nr)
% IN_OUT_PRF_GEN Computes outer coordinates around the blade
% Coordinates a small distance outside of the blade are written
% to the file naca_NR.outer and reads airfoil data file writes
in
% naca_NR.inner with desired number of nodes.
[XO,YO,ZO]=IN_OUT_PRF_GEN(FILENAME,NR)
%
% Input
%     File name: FILENAME: text string with the filename
%                 that contains airfoil data e.g in_out_gen('airfoil.01',1)
%     NR:         profile number
%
% Output
%     writes data for inner profile naca_NR.inner
%     and outer profile naca_NR.outer
%
% Copyright: © David Lindström
% Author: David Lindström & AA Narejo
% University West
% 11/8-2007
dn=47; % Desired nodes
s=textread(filename);
x1=s(41:-1:3,1);
y1=s(41:-1:3,2);
s1=[x1,y1];
x2=s(43:81,1);
y2=s(43:81,2);
s2=[x2,y2];

for i=1:length(s1)
    t(i,1)=s1(i,1);
```

```

t(i,2)=s1(i,2);
end

for i=1:length(s2)
    t(i+length(s1),1)=s2(i,1);
    t(i+length(s1),2)=s2(i,2);
end
tl=length(s1)+length(s2);
for i=2:dn+1
    k=round(i*tl/(dn+1));
    u(i,1)=t(k,1);
    u(i,2)=t(k,2);
end
u(1,1)=dn;
u(1,2)=1;
u(2:(dn+1),3)=zeros;
plot(t(:,1),t(:,2))
hold on
(fid,message)=fopen(['naca_',num2str(nr),'.inner'],'w');
if fid==-1
    disp('Error obtained. Could not open the file')
    disp(['wrkdir','.naca_',num2str(nr),'.inner for writing.'])
    error(message)
end
fprintf(fid,'%4i %1i\n',[dn 1]);
for i=2:(dn+1)
    fprintf(fid,'%11.8f %11.8f %3.1f\n',[u(i,1);u(i,2);u(i,3)]);
end
fclose(fid);
z=0;
% Distance to outer curve
c=max(u(:,1))-min(u(:,1));
doc=c/125; %in the case of bad outer profile, change the
denominator
Xr=u(2:48,1)';
Yr=u(2:48,2)';
% Compute outer curve
dx=diff(Xr);
dy=diff(Yr);
dl=sqrt(dx.^2+dy.^2);
Xo=Xr(1:end-1)+0.5*dx+doc*dy./dl;
Yo=Yr(1:end-1)+0.5*dy-doc*dx./dl;

% Re-define the outer geometry to get rid of a possible loop in
the
% neighbourhood of a sharp bend at the inner geometry.
% To leave the number of points unchanged a spline technique is
used.
% However, it would be more stable to directly define the outer
geometry
% from the inner data, without defining any spline, and using the
distances
% between the inner points to define the distribution of the outer
ones.

% Compute distances between points of the outer curve
dx=diff(Xo);
dy=diff(Yo);
dl=sqrt(dx.^2+dy.^2);

```

```

dlsum=zeros(1,length(dl));
for i = 1:length(dl)
    dlsum(i)=sum(dl(1:i));
end
% Number of points for outer spline
np=round(0.4*(1+length(Xr)));
% Compute indexes for np equidistant segments
lseg=linspace(dlsum(1),dlsum(end),np);
ind=zeros(1,np);
for i = 1:np
    ind(i)=min(find(dlsum>=lseg(i)));
end
ind(np)=ind(np)+1;
% Compute a spline through np of these points

p(1,:)=Xo(ind);
p(2,:)=Yo(ind);
t = 1:np;
pp = spline(t,p);
xy = ppval(pp, linspace(1,np,length(Xo)));
Xo=xy(1,:);
Yo=xy(2,:);

% Compute three extra points on the outer curve near the trailing
edge
xte=0.51*(Xr(1)+Xr(end));
yte=0.51*(Yr(1)+Yr(end));
xtd=xte-0.5*(Xr(2)+Xr(end-1));
ytd=yte-0.5*(Yr(2)+Yr(end-1));
dtd=sqrt(xtd^2+ytd^2);
xtd=doo*xtd/dtd;
ytd=doo*ytd/dtd;
xrr=xte+xtd;
yrr=yte+ytd;
xur=xte+(xtd-ytd)/sqrt(2);
yur=yte+(xtd+ytd)/sqrt(2);
xlr=xte+(xtd+ytd)/sqrt(2);
ylr=yte-(xtd-ytd)/sqrt(2);
Xo=[xrr,xur,Xo,xlr];
Yo=[yrr,yur,Yo,ylr];
Zo=0*Xo+z;

[fid,message]=fopen(['naca_',num2str(nr),'.outer'],'w');
if fid== -1
    disp('Error obtained. Could not open the file')
    disp(['wrkdir,'.naca_',num2str(nr),'.outer for writing.'])
    error(message)
end
fprintf(fid,'%4i %li\n',[length(Xo) 1]);
fprintf(fid,'%11.8f %11.8f %3.1f\n',[Xo;Yo;Zo]);
plot(Xo,Yo);
fclose(fid);

```

## B. Appendix 2

This is a program written in Matlab for 3D airfoil and it generates journal file, can be read in **gambit**.

```

function
make_gambit_journal(nr,a,b,delta_x1,delta_x2,y,alpha,beta,r)
%   MAKE_GAMBIT_JOURNAL   Make a journal file for Gambit
%
%   MAKE_GAMBIT_JOURNAL(NR,A,B,DELTA_X1,DELTA_X2,Y,ALPHA,BETA,R)
%   Input
%       nr: the simulation number or profile number
%       a : Number of nodes away from xminindex at above the outer
%           profile
%       b : Number of nodes away from xminindex at below the outer
%           profile
%       delta_x1 : certain distance from leading edge.
%       delta_x2 : certain distance from trialing edge.
%       y : Vertical distance between two periodic boundaries.
%       r : Radius for curvature.
%       alpha : Inflow angle
%       beta : Outflow angle
%       nob : Number of blades
%   Output
%       Writes calculated nodes and mesh in GBT3DNR.JOU file.
%       NR is number of layers or number data files for generating
the
%       profiles.
% Copyright: © AA Narejo and © David Lindström
% Author: AA Narejo
% University of Trollhättan Uddevalla
% 2/1-2006

% Default values
if nargin<1 ;
a=1; % one above of minimum point
b=2; % 3rd point will be selected w.r.t minimum point.
delta_x1=5;
delta_x2=5;
alpha=0;
beta=0 ;
nob=22;
delta_y1= delta_x1 * tan (alpha);
delta_y2= -delta_x2 * tan (beta);
r=[15 30 45];
ne=length(r);
os=r(ne)-27; % default value for cutter & sets location of cutting
profile
nr=ne; %
end

% Open the new journal file
[fid,message]=fopen(['GBT3D',num2str(nr),'.jou'],'w');
if ~fid,disp([message]),end
fprintf(fid,'%s\n','reset');
fprintf(fid,'%s\n','/ Journal File for GAMBIT 2.2.30, Database
2.2.14, lnx86 BH04110220');
fprintf(fid,'%s\n',['identifier name "naca_',num2str(nr),'" new
saveprevious']);

```

```

fprintf(fid,'%s\n','solver select "FLUENT 5/6"');
% Increments for setting the loops for profiles.
c1=115;
c2=116;
c3=117;
c4=118;
inc1=0;
inc2=171;
inc3=0;

% Reads data files to generate inner and outer profiles.
for i = 1:ne
nr=i;
y= 2*pi*r(i)/nob;
fprintf(fid,'%s\n',['import vertexdata
"naca_',num2str(nr),'.inner"']);
fprintf(fid,'%s\n',['import vertexdata
"naca_',num2str(nr),'.outer"']);
for j=1:96
if i~=1
fprintf(fid,'%s\n',['vertex modify "vertex.' num2str(j+inc1) '"'
label "vertex.' num2str(j) "'']);
end
end
% coxy calcualtes co-ordinates for lowest x-value in outer
profile.
coxy=edgeindexc(['naca_',num2str(i),'.outer'],1); % Fetch
coordinates where xminindex lies outer profile.
fprintf(fid,'%s\n',['vertex create "vertex.97" coordinates '
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.98" coordinates '
[num2str(coxy(1,1)-delta_x1) ' ' num2str(coxy(1,2)+y/2-delta_y1) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.106" coordinates '
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)-y/2) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.107" coordinates '
[num2str(coxy(1,1)-delta_x1) ' ' num2str(coxy(1,2)-y/2-delta_y1) ' 0']]);
% Fetch coordinates where xmaxindex lies at outer profile.
coxy=edgeindexc(['naca_',num2str(i),'.outer'],2);
fprintf(fid,'%s\n',['vertex create "vertex.99" coordinates '
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.105" coordinates '
[num2str(coxy(1,1)+delta_x2) ' ' num2str(coxy(1,2)+y/2-delta_y2) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.108" coordinates '
[num2str(coxy(1,1)) ' ' num2str(coxy(1,2)-y/2) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.109" coordinates '
[num2str(coxy(1,1)+delta_x2) ' ' num2str(coxy(1,2)-y/2-delta_y2) ' 0']]);
% Fetch coordinates where xmaxindex lies at inner profile.
conxyave=(nodeco(['naca_',num2str(i),'.inner'],1) +
nodeco(['naca_',num2str(i),'.inner'],47))/2 ;
fprintf(fid,'%s\n',['vertex create "vertex.102" coordinates '
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',['vertex create "vertex.111" coordinates '
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)-y/2) ' 0']]);
% Fetch coordinates of n node at inner profile then takes as
average of co-ordinates.

```

```

conxyave=(nodeco(['naca_ ',num2str(i),'.inner'],12)+
nodeco(['naca_ ',num2str(i),'.inner'],36))/2 ;
fprintf(fid,'%s\n',[ 'vertex create "vertex.101" coordinates '
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)+y/2) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.110" coordinates '
[num2str(conxyave(1,1)) ' ' num2str(conxyave(1,2)-y/2) ' 0']]);
fprintf(fid,'%s\n','vertex create "cut above" coordinates 0.45 0.4
0');
fprintf(fid,'%s\n','vertex create "cut below" coordinates 0.45 0
0');
fprintf(fid,'%s\n','edge create "airfoil te" straight "vertex.47"
"vertex.1"');
fprintf(fid,'%s\n','edge create "outer te" nurbs "vertex.96"
"vertex.48" "vertex.49"');
fprintf(fid,'%s\n','edge create "up right vertex" straight
"vertex.1" "vertex.49"');
fprintf(fid,'%s\n','edge create "lo right vertex" straight
"vertex.47" "vertex.96"');
fprintf(fid,'%s\n','edge create "edge above right" straight
"vertex.105" "vertex.99"');
xminind=edgeindex(['naca_ ',num2str(i),'.outer'])+47;
xminindi=edgeindex(['naca_ ',num2str(i),'.inner']);
fprintf(fid,'%s\n',[ 'edge create "up left vertex" straight
"vertex.' num2str(xminindi-a) '' ' ' ' "vertex.' num2str(xminind-
a) '''']);
fprintf(fid,'%s\n',[ 'edge create "lo left vertex" straight
"vertex.' num2str(xminindi+b) '' ' ' ' "vertex.'
num2str(xminind+b) '''']);
conxy=nodeco(['naca_ ',num2str(i),'.outer'],2);
fprintf(fid,'%s\n',[ 'vertex create "on the te" coordinates '
[num2str(coxy(1,1)+delta_x2) ' ' num2str(conxy(1,2)-delta_y2) ' 0']]);
conxy=nodeco(['naca_ ',num2str(i),'.outer'],xminind-a-47);
coxy=edgeindexc(['naca_ ',num2str(i),'.outer'],1);
fprintf(fid,'%s\n',[ 'vertex create "on the le" coordinates '
[num2str(coxy(1,1)-delta_x1) ' ' num2str(conxy(1,2)-delta_y1) ' 0']]);
fprintf(fid,'%s\n','edge create "edge on the te" straight
"vertex.49" "on the te"');
fprintf(fid,'%s\n','edge create "edge on the le" straight
"vertex.' num2str(xminind-a) '' "on the le"');
fprintf(fid,'%s\n','edge create "edge above mid" nurbs "vertex.99"
"vertex.102" "vertex.101" \');
fprintf(fid,'%s\n',' "vertex.97"');
fprintf(fid,'%s\n','edge create "edge above left" straight
"vertex.97" "vertex.98"');
fprintf(fid,'%s\n','edge create "edge below right" straight
"vertex.109" "vertex.108"');
fprintf(fid,'%s\n','edge create "edge below mid" nurbs
"vertex.108" "vertex.111" "vertex.110" \');
fprintf(fid,'%s\n',' "vertex.106"');
fprintf(fid,'%s\n','edge create "edge below left" straight
"vertex.106" "vertex.107"');
fprintf(fid,'%s\n','edge create "vertex up right" straight
"vertex.99" "vertex.49"');
fprintf(fid,'%s\n','edge create "vertex lo right" straight
"vertex.108" "vertex.96"');
fprintf(fid,'%s\n','edge create "inlet 1" straight "vertex.98" "on
the le"');

```

```

fprintf(fid,'%s\n','edge create "inlet 2" straight "on the le"
"vertex.107"');
fprintf(fid,'%s\n','edge create "outlet 1" straight "vertex.105"
"on the te"');
fprintf(fid,'%s\n','edge create "outlet 2" straight "vertex.109"
"on the te"');

xminind=edgeindex(['naca_ ',num2str(i),'.outer'])+47;
xminindi=edgeindex(['naca_ ',num2str(i),'.inner']);
fprintf(fid,'%s\n',['edge create "vertex up left" straight
"vertex.97" ' ' "vertex.' num2str(xminind-a) '''']);
fprintf(fid,'%s\n',['edge create "vertex lo left" straight
"vertex.106" ' ' "vertex.' num2str(xminind+b) '''']);
% new geometry of outer lower profile
xminindv=xminind+b;
fprintf(fid,'%s\n','edge create "outer bottom" nurbs \'');
for xminindv=xminindv:95
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) ''\'']]);
end
fprintf(fid,'%s\n',"vertex.96");
% new geometry of outer upper profile
xminindv=xminind-a;
fprintf(fid,'%s\n','edge create "outer top" nurbs \'');
for xminindv=xminindv:-1:50
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) ''\'']]);
end
fprintf(fid,'%s\n',"vertex.49");
% new geometry of outer le profile
xminindv=xminind-a;
fprintf(fid,'%s\n','edge create "outer le" nurbs \'');
for xminindv=xminindv:xminind+b-1
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) ''\'']]);
end
fprintf(fid,'%s\n',[ ' "vertex.' num2str(xminind+b) '''' ''']);
% new geometry of inner lower profile
xminindv=xminindi+b;
fprintf(fid,'%s\n','edge create "airfoil bottom" nurbs \'');
for xminindv=xminindi+b:46
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) ''\'']]);
end
fprintf(fid,'%s\n',"vertex.47");
% new geometry of outer upper profile
xminindv=xminindi-a;
fprintf(fid,'%s\n','edge create "airfoil top" nurbs \'');
%for xminindv=xminindv:-1:2
for xminindv=1:xminindv-1
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) ''\'']]);
end
fprintf(fid,'%s\n',[["vertex.' num2str(xminindi-a) ''']]);

% new geometry of outer airfoil le profile
xminindv=xminindi-a;
fprintf(fid,'%s\n','edge create "airfoil le" nurbs \'');
for xminindv=xminindv:xminindi+b-1
fprintf(fid,'%s\n',[['vertex.' num2str(xminindv) ''\'']]);
end
fprintf(fid,'%s\n',[ "vertex.' num2str(xminindi+b) '''' ''']);
fprintf(fid,'%s\n','face create "face airfoil above" wireframe
"outer top" "up left vertex" \'');

```

```

fprintf(fid,'%s\n',' "airfoil top" "up right vertex" real');
fprintf(fid,'%s\n','face create "face airfoil below" wireframe
"airfoil bottom" "lo left vertex" \');
fprintf(fid,'%s\n',' "outer bottom" "lo right vertex" real');
fprintf(fid,'%s\n','face create "face airfoil le" wireframe "up
left vertex" "outer le" \');
fprintf(fid,'%s\n',' "lo left vertex" "airfoil le" real');
fprintf(fid,'%s\n','face create "face airfoil te" wireframe "up
right vertex" "airfoil te" \');
fprintf(fid,'%s\n',' "lo right vertex" "outer te" real');
fprintf(fid,'%s\n','face create "face outer above" wireframe "edge
above mid" "vertex up left" \');
fprintf(fid,'%s\n',' "outer top" "vertex up right" real');
fprintf(fid,'%s\n','face create "face outer below" wireframe
"outer bottom" "vertex lo left" \');
fprintf(fid,'%s\n',' "edge below mid" "vertex lo right" real');
fprintf(fid,'%s\n','face create "face outer le 1" wireframe "inlet
1" "edge on the le" \');
fprintf(fid,'%s\n',' "vertex up left" "edge above left" real');
fprintf(fid,'%s\n','face create "face outer le 2" wireframe "inlet
2" "edge on the le" "outer le" \');
fprintf(fid,'%s\n','"vertex lo left" "edge below left" real');
fprintf(fid,'%s\n','face create "face outer te 1" wireframe
"vertex up right" "edge above right" \');
fprintf(fid,'%s\n',"outlet 1" "edge on the te" real');
fprintf(fid,'%s\n','face create "face outer te 2" wireframe
"vertex lo right" "outer te" \');
fprintf(fid,'%s\n','"edge on the te" "outlet 2" "edge below right"
real');

% new geometry 3D model.
% Fetch coordinates where xminindex lies outer profile.
coxy=edgeindexc(['naca_',num2str(i),'.outer'],1);
minx = coxy(1,1)-delta_x1;
miny = coxy(1,2)-y/2-delta_y1;
coxy=edgeindexc(['naca_',num2str(i),'.outer'],2);
maxx =coxy(1,1)+delta_x2;
maxy=coxy(1,2)+y/2-delta_y2;
yaxesmid=(miny+maxy)/2;
fprintf(fid,'%s\n',[ 'vertex create "vertex.121" coordinates '
[num2str(minx) ' ' num2str(yaxesmid) ' 0']]);
fprintf(fid,'%s\n',[ 'vertex create "vertex.168" coordinates '
[num2str(maxx) ' ' num2str(yaxesmid) ' 0']]);

% Radius for cylinder is defined in next line.
% loop scheme
fprintf(fid,'%s\n',[ 'edge create "radius b' num2str(i) '" radius '
num2str(r(i)) 'startangle 0 endangle 180 center \']);
fprintf(fid,'%s\n',"vertex.168" yzplane arc');
fprintf(fid,'%s\n',[ 'edge create "radius a' num2str(i) '" radius '
num2str(r(i)) 'startangle 0 endangle 180 center \']);
fprintf(fid,'%s\n',"vertex.121" yzplane arc');

% Two Radii for cylinder is connected.
fprintf(fid,'%s\n',[ 'edge create "radii cd' num2str(i) '" straight
"vertex.' num2str(c1+incl) '" "vertex.' num2str(c3+incl) "'");
fprintf(fid,'%s\n',[ 'edge create "radii cu' num2str(i) '" straight
"vertex.' num2str(c2+incl) '" "vertex.' num2str(c4+incl) "'");

```

```
% creation of cylindrical face
fprintf(fid,'%s\n',['face create "half cylinder' num2str(i) ''
wireframe "radii cu' num2str(i) '" "radii cd' num2str(i) ''
"radius b' num2str(i) '" "radius a' num2str(i) '" \']);
fprintf(fid,'%s\n',' real');

%project on cylinder
fprintf(fid,'%s\n',['edge create "inlet c a' num2str(i) '" project
"inlet 1" face "half cylinder' num2str(i) '" vector 0 0 1']);
fprintf(fid,'%s\n',['edge create "outlet c a' num2str(i) '"'
project "outlet 1" face "half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "inlet c b' num2str(i) '" project
"inlet 2" face "half cylinder' num2str(i) '" vector 0 0 1']);
fprintf(fid,'%s\n',['edge create "outlet c b' num2str(i) '"'
project "outlet 2" face "half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "edge on the te c' num2str(i) '"'
project "edge on the te" face "half cylinder' num2str(i) '" vector
0 0 1']);
fprintf(fid,'%s\n',['edge create "edge on the le c' num2str(i) '"'
project "edge on the le" face "half cylinder' num2str(i) '" vector
0 0 1']);
fprintf(fid,'%s\n',['edge create "edge below left c' num2str(i) '"'
project "edge below left" face \']);
fprintf(fid,'%s\n',['"half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "edge below mid c' num2str(i) '"'
project "edge below mid" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n',' vector 0 0 1');
fprintf(fid,'%s\n',[' edge create "edge below right c' num2str(i)
'" project "edge below right" face \']);
fprintf(fid,'%s\n',['"half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "edge above right c' num2str(i)
'" project "edge above right" face \']);
fprintf(fid,'%s\n',['"half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "edge above mid c' num2str(i) '"'
project "edge above mid" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n','vector 0 0 1');
fprintf(fid,'%s\n',['edge create "edge above left c' num2str(i) '"'
project "edge above left" face \']);
fprintf(fid,'%s\n',['"half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',[' edge create "vertex to right c' num2str(i)
'" project "vertex to right" face \']);
fprintf(fid,'%s\n',['"half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "vertex up right c' num2str(i) '"'
project "vertex up right" face \']);
fprintf(fid,'%s\n',['"half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "outer te c' num2str(i) '"'
project "outer te" face "half cylinder' num2str(i) '" vector 0 0
1']);
fprintf(fid,'%s\n',['edge create "outer top c' num2str(i) '"'
project "outer top" face "half cylinder' num2str(i) '" vector 0 0
1']);
```

```

fprintf(fid,'%s\n','1');
fprintf(fid,'%s\n',['edge create "vertex up left c' num2str(i) "' project "vertex up left" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n','vector 0 0 1');
fprintf(fid,'%s\n',['edge create "vertex lo left c' num2str(i) "' project "vertex lo left" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n',' vector 0 0 1');
fprintf(fid,'%s\n',['edge create "outer le c' num2str(i) "' project "outer le" face "half cylinder' num2str(i) '" vector 0 0 1']);
fprintf(fid,'%s\n',['edge create "outer bottom c' num2str(i) "' project "outer bottom" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n',' vector 0 0 1');
fprintf(fid,'%s\n',['edge create "up left vertex c' num2str(i) "' project "up left vertex" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n',' vector 0 0 1');
fprintf(fid,'%s\n',['edge create "lo left vertex c' num2str(i) "' project "lo left vertex" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n','vector 0 0 1');
fprintf(fid,'%s\n',['edge create "airfoil bottom c' num2str(i) "' project "airfoil bottom" face "half cylinder' num2str(i) '" \']);
fprintf(fid,'%s\n',' vector 0 0 1');
fprintf(fid,'%s\n',['edge create "airfoil le c' num2str(i) "' project "airfoil le" face "half cylinder' num2str(i) '" vector 0 \']);
fprintf(fid,'%s\n',' 0 1');
fprintf(fid,'%s\n',['edge create "airfoil top c' num2str(i) "' project "airfoil top" face "half cylinder' num2str(i) '" vector \']);
fprintf(fid,'%s\n',' 0 0 1');
fprintf(fid,'%s\n',['edge create "airfoil te c' num2str(i) "' project "airfoil te" face "half cylinder' num2str(i) '" vector 0 \']);
fprintf(fid,'%s\n',' 0 1');
fprintf(fid,'%s\n',[' edge create "lo right vertex c' num2str(i) "' project "lo right vertex" face \']);
fprintf(fid,'%s\n',['    "half cylinder' num2str(i) '" vector 0 0 1']);
fprintf(fid,'%s\n',[' edge create "up right vertex c' num2str(i) "' project "up right vertex" face \']);
fprintf(fid,'%s\n',['    "half cylinder' num2str(i) '" vector 0 0 1']);
fprintf(fid,'%s\n',['face create "face airfoil above c' num2str(i) "' wireframe "up left vertex c' num2str(i) '" "outer top c' num2str(i) '" \']);
fprintf(fid,'%s\n',['    "airfoil top c' num2str(i) '" "up right vertex c' num2str(i) '" real']);
fprintf(fid,'%s\n',['face create "face airfoil below c' num2str(i) "' wireframe "lo left vertex c' num2str(i) '" \']);
fprintf(fid,'%s\n',['    "airfoil bottom c' num2str(i) '" "outer bottom c' num2str(i) '" "lo right vertex c' num2str(i) '" real']);
fprintf(fid,'%s\n',['face create "face airfoil le c a' num2str(i) "' wireframe "vertex up right c' num2str(i) '" \']);
fprintf(fid,'%s\n',['    "edge above right c' num2str(i) '" "edge on the te c' num2str(i) '" "outlet c a' num2str(i) '" real']);
fprintf(fid,'%s\n',['face create "face outer te c b' num2str(i) "' wireframe "edge on the te c' num2str(i) '" "outlet c b' num2str(i) '" \']);

```

```

fprintf(fid,'%s\n',[ ' "edge below right c' num2str(i) '" "vertex
to right c' num2str(i) '" "outer te c' num2str(i) '" real
tolerance -1']);
fprintf(fid,'%s\n',[ 'face create "face outer le c a' num2str(i) '"'
wireframe "edge above left c' num2str(i) '" \']);
fprintf(fid,'%s\n',[ ' "vertex up left c' num2str(i) '" "edge on
the le c' num2str(i) '" "inlet c a' num2str(i) '" real']);
fprintf(fid,'%s\n',[ 'face create "face outer le c b' num2str(i) '"'
wireframe "inlet c b' num2str(i) '" "edge on the le c' num2str(i)
'" "outer le c' num2str(i) '" \']);
fprintf(fid,'%s\n',[ ' "vertex lo left c' num2str(i) '" "edge
below left c' num2str(i) '" real']);
fprintf(fid,'%s\n',[ 'face create "face outer above c' num2str(i)
'" wireframe "outer top c' num2str(i) '" "vertex up left c'
num2str(i) '" \']);
fprintf(fid,'%s\n',[ ' "edge above mid c' num2str(i) '" "vertex up
right c' num2str(i) '" real']);
fprintf(fid,'%s\n',[ 'face create "face outer below c' num2str(i)
'" wireframe "edge below mid c' num2str(i) '" \']);
fprintf(fid,'%s\n',[ ' "vertex lo left c' num2str(i) '" "outer
bottom c' num2str(i) '" "vertex to right c' num2str(i) '" real']);
fprintf(fid,'%s\n',[ 'face create "face airfoil le c' num2str(i) '"'
wireframe "airfoil le c' num2str(i) '" "up left vertex c'
num2str(i) '" \']);
fprintf(fid,'%s\n',[ ' "outer le c' num2str(i) '" "lo left vertex
c' num2str(i) '" real']);
fprintf(fid,'%s\n',[ 'face create "face airfoil te c' num2str(i) '"'
wireframe "up right vertex c' num2str(i) '" "outer te c'
num2str(i) '" \']);
fprintf(fid,'%s\n',[ ' "airfoil te c' num2str(i) '" "lo right
vertex c' num2str(i) '" real']);
if i>1
inc1=inc1+169;
inc2=inc2+114+55;
else
inc1=inc1+171;
inc2=inc2+114;
end
fprintf(fid,'%s\n',[ 'face delete "half cylinder' num2str(i) '"'
lowertopology']);
if i<ne
fprintf(fid,'%s\n',[ 'face delete "face outer te 1" "face outer te
2" "face airfoil above" \']);
fprintf(fid,'%s\n',[ ' "face airfoil below" "face outer below"
"face outer above" \']);
fprintf(fid,'%s\n',[ ' "face airfoil le" "face outer le 1" "face
outer le 2" "face airfoil te" onlymesh']);
fprintf(fid,'%s\n',[ 'face delete "face outer te 1" "face outer te
2" "face airfoil above" \']);
fprintf(fid,'%s\n',[ ' "face airfoil below" "face airfoil te"
"face outer below" \']);
fprintf(fid,'%s\n',[ ' "face outer above" "face airfoil le" "face
outer le 1" "face outer le 2" \']);
fprintf(fid,'%s\n',[ 'lowertopology']);
fprintf(fid,'%s\n',[ 'vertex delete "cut above" "cut below" ']);
for j=1:96
fprintf(fid,'%s\n',[ 'vertex delete "vertex.' num2str(j) ''']);
end

```

```

fprintf(fid,'%s\n',[ 'vertex delete "vertex.168" "vertex.111"
"vertex.102" "vertex.110" \'']);
fprintf(fid,'%s\n',[ '"vertex.101" "vertex.121"' ]);
end
fprintf(fid,'%s\n','window modify vertex invisible');
end

% Creates lateral Faces
fprintf(fid,'%s\n','face create "edge above right face" skin "edge
above right c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"edge above right c' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "outlet facel" skin "outlet c a1"
\'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"outlet c a' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "vertex up right face" skin
"vertex up right c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"vertex up right c' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "outer te face" skin "outer te c1"
\'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"outer te c' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "vertex to right face" skin
"vertex to right c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"vertex to right c' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge below right face" skin "edge
below right c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"edge below right c' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "outlet face 2" skin "outlet c b1"
\'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"outlet c b' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge above mid face" skin "edge
above mid c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[ '"edge above mid c' num2str(i) '\'']);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "outer top face" skin "outer top
c1" \'');
for i = 2:ne

```

```

fprintf(fid,'%s\n',[["outer top c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "airfoil top face" skin "airfoil
top cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["airfoil top c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "airfoil le face" skin "airfoil le
c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["airfoil le c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "airfoil bottom face" skin
"airfoil bottom cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["airfoil bottom c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "airfoil te face" skin "airfoil te
c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["airfoil te c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "outer bottom face" skin "outer
bottom cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["outer bottom c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "lo left vertex face" skin "lo
left vertex cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["lo left vertex c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "vertex to left face" skin "vertex
lo left cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["vertex lo left c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge below mid face" skin "edge
below mid cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["edge below mid c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge above left face" skin "edge
above left cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["edge above left c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "inlet facel" skin "inlet c al"
\'');

```

```

for i = 2:ne
fprintf(fid,'%s\n',[["inlet c a' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "inlet face2" skin "inlet c b1"
\'');
for i = 2:ne
fprintf(fid,'%s\n',[["inlet c b' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge below left face" skin "edge
below left c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["edge below left c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge on the le face" skin "edge
on the le c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["edge on the le c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "edge on the te face" skin "edge
on the te c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["edge on the te c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "vertex up left face" skin "vertex
up left c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["vertex up left c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "outer le face" skin "outer le c1"
\'');
for i = 2:ne
fprintf(fid,'%s\n',[["outer le c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "up left vertex face" skin "up
left vertex c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["up left vertex c' num2str(i) '"\']]);
end
fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "airfoil right face" wireframe
"airfoil le c1" "airfoil top c1" \'');
fprintf(fid,'%s\n','"airfoil bottom c1" "airfoil te c1" real');
fprintf(fid,'%s\n',[["face create "airfoil left face" wireframe
"airfoil le c' num2str(nr) '" "airfoil top c' num2str(nr) "
\'"]);
fprintf(fid,'%s\n',[["airfoil bottom c' num2str(nr) '" "airfoil
te c' num2str(nr) '" real']);
fprintf(fid,'%s\n','face create "lo right vertex face" skin "lo
right vertex c1" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["lo right vertex c' num2str(i) '"\']]);
end

```

```

fprintf(fid,'%s\n','directions 0 0 0');
fprintf(fid,'%s\n','face create "up right vertex face" skin "up
right vertex cl" \'');
for i = 2:ne
fprintf(fid,'%s\n',[["up right vertex c' num2str(i) '"\'']]);
end
fprintf(fid,'%s\n','directions 0 0 0');

% scheme for deletion of layers between profiles
for i = 2:ne-1
fprintf(fid,'%s\n',[["face delete "face airfoil le c' num2str(i) ''
"face outer te c b' num2str(i) '"\'']]);
fprintf(fid,'%s\n',[[""face airfoil below c' num2str(i) '' "face
airfoil te c' num2str(i) '' "face outer below c' num2str(i) ''
\']]);
fprintf(fid,'%s\n',[[""face outer above c' num2str(i) '' "face
airfoil above c' num2str(i) '' "face outer le c b' num2str(i) ''
\']]);
fprintf(fid,'%s\n',[[""face airfoil le c a' num2str(i) '' "face
outer le c a' num2str(i) '' lowertopology']]);
end

% Creation of volumes
fprintf(fid,'%s\n','volume create "volume1" stitch "face airfoil
le c a1" "edge on the te face" \'');
fprintf(fid,'%s\n','"vertex up right face" "edge above right face"
"outlet facel" \'');
fprintf(fid,'%s\n',[[""face airfoil le c a' num2str(ne) '' real']]);
fprintf(fid,'%s\n','volume create "volume2" stitch "vertex up left
face" "outer top face" \'');
fprintf(fid,'%s\n',[[""vertex up right face" "face outer above c'
num2str(ne) '' "edge above mid face" \']]);
fprintf(fid,'%s\n','"face outer above c1" real');
fprintf(fid,'%s\n',[["volume create "volume3" stitch "face outer le
c a' num2str(ne) '' "edge above left face" \']]);
fprintf(fid,'%s\n','"inlet facel" "edge on the le face" "vertex
up left face" \'');
fprintf(fid,'%s\n','"face outer le c a1" real');
fprintf(fid,'%s\n',[["volume create "volume4" stitch "edge on the
le face" "face outer le c b' num2str(ne) '' \']]);
fprintf(fid,'%s\n','"inlet face2" "edge below left face" "vertex
to left face" "outer le face" \'');
fprintf(fid,'%s\n','"face outer le c b1" real');
fprintf(fid,'%s\n','volume create "volume5" stitch "up left vertex
face" "outer le face" \'');
fprintf(fid,'%s\n',[[""lo left vertex face" "face airfoil le c'
num2str(ne) '' "airfoil le face" \']]);
fprintf(fid,'%s\n','"face airfoil le c1" real');
fprintf(fid,'%s\n',[["volume create "volume6" stitch "face airfoil
below c' num2str(ne) '' "outer bottom face" \']]);
fprintf(fid,'%s\n','"airfoil bottom face" "lo left vertex face"
"lo right vertex face" \'');
fprintf(fid,'%s\n','"face airfoil below c1" real');
fprintf(fid,'%s\n',[["volume create "volume7" stitch "vertex to
left face" "face outer below c' num2str(ne) '' \']]);
fprintf(fid,'%s\n','"edge below mid face" "vertex to right face"
"outer bottom face" \'');
fprintf(fid,'%s\n','"face outer below c1" real');

```

```

fprintf(fid,'%s\n','volume create "volume8" stitch "lo right
vertex face" "outer te face" \'');
fprintf(fid,'%s\n','"airfoil te face" "up right vertex face" "face
airfoil te cl" \'');
fprintf(fid,'%s\n',[["face airfoil te c' num2str(ne) '" real']]);
fprintf(fid,'%s\n','volume create "volume9" stitch "face airfoil
above cl" "up left vertex face" \'');
fprintf(fid,'%s\n','"outer top face" "airfoil top face" "up right
vertex face" \'');
fprintf(fid,'%s\n',[["face airfoil above c' num2str(ne) '" real']]);
fprintf(fid,'%s\n','volume create "volume10" stitch "edge on the
te face" "face outer te c bl" \'');
fprintf(fid,'%s\n','"outlet face 2" "edge below right face"
"vertex to right face" \'');
fprintf(fid,'%s\n',[["outer te face" "face outer te c b'
num2str(ne) '" real']]);

% design of above cutting plane by importing data
fprintf(fid,'%s\n','import vertexdata "dwall.dim"');
fprintf(fid,'%s\n','edge create "lnerb" nurbs \'');
inc1=inc1+1
for j=inc1:inc1+78
    fprintf(fid,'%s\n',[["vertex.' num2str(j) '\']] );
end
fprintf(fid,'%s\n',[["vertex.' num2str(j+1) '" interpolate']]);
fprintf(fid,'%s\n','vertex delete \'');
for j=inc1+1:inc1+77
    fprintf(fid,'%s\n',[["vertex.' num2str(j) '\']] );
end
fprintf(fid,'%s\n',[["vertex.' num2str(j+1) ''']] );

% Creates cutting face profile
fprintf(fid,'%s\n',[["face create "cutter1" revolve "lnerb" \'']]);
fprintf(fid,'%s\n','dangle -180 vector -0.50699 0 0 origin 0.3967
0.040078 0');
fprintf(fid,'%s\n',[["face move "cutter1" offset 0 0 '
num2str(os)]]);
fprintf(fid,'%s\n','face delete "airfoil right face" "airfoil left
face" lowertopology');

% Splits volumes by certain profile
fprintf(fid,'%s\n','volume split "volumel" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume2" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume3" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume4" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume5" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume6" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume7" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume8" faces "cutter1"
connected keeptool');

```

```

fprintf(fid,'%s\n','volume split "volume9" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','volume split "volume10" faces "cutter1"
connected keeptool');
fprintf(fid,'%s\n','face delete "cutter1" lowertopology');

% Merging unit volumes into one volume
fprintf(fid,'%s\n','volume delete "volume1" "volume2" "volume3"
"volume4" "volume5" "volume6" \');
fprintf(fid,'%s\n','"volume7" "volume8" "volume9" "volume10"
lowertopology');
fprintf(fid,'%s\n','volume merge "volume.11" "volume.12"
"volume.13" "volume.14" "volume.15" \');
fprintf(fid,'%s\n','"volume.16" "volume.17" "volume.18"
"volume.19" "volume.20" real');

% Deletes 2D face
fprintf(fid,'%s\n','face delete "face airfoil above" "face airfoil
below" "face airfoil le" \');
fprintf(fid,'%s\n','"face airfoil te" "face outer above" "face
outer below" "face outer le 1" \');
fprintf(fid,'%s\n','"face outer le 2" "face outer te 1" "face
outer te 2" lowertopology');

% links the periodic faces
if ne==2
fprintf(fid,'%s\n','face link "face.84" "edge below left face"
edges "edge.173" \');
fprintf(fid,'%s\n','"edge below left cl" vertices "vertex.143"
"vertex.125" reverse periodic');
fprintf(fid,'%s\n','face link "face.80" "edge below mid face"
edges "edge.174" \');
fprintf(fid,'%s\n','"edge below mid cl" vertices "vertex.142"
"vertex.135" reverse periodic');
fprintf(fid,'%s\n','face link "edge above right face" "face.110"
edges "edge.128" \');
fprintf(fid,'%s\n','"edge below right cl" vertices "vertex.140"
"vertex.134" reverse periodic')
else
fprintf(fid,'%s\n',[ 'face link "face.' num2str(106+23*(ne-3)) ' "
"face.' num2str(110+23*(ne-3)) ' " edges "edge.'
num2str(233+61*(ne-3)) ' " "edge below left cl" \']);
fprintf(fid,'%s\n',[ 'vertices "vertex.143" "vertex.125" reverse
periodic']);
fprintf(fid,'%s\n',[ 'face link "edge above mid face" "edge below
mid face" edges "edge.' num2str(234+59*(ne-3)) ' " \']");
fprintf(fid,'%s\n',[ '"edge below mid cl" vertices "vertex.142"
"vertex.135" reverse periodic']);
fprintf(fid,'%s\n',[ 'face link "edge above right face" "face.'
num2str(132+31*(ne-3)) ' " edges "edge.' num2str(188+60*(ne-3)) ' "
\']);
fprintf(fid,'%s\n',[ '"edge below right cl" vertices "vertex.140"
"vertex.134" reverse periodic'])
end

% Meshing the volume
fprintf(fid,'%s\n','volume mesh "volume.11" tetrahedral size
0.4');

```

```
% Creates B.Cs for the domain
if ne==2
fprintf(fid,'%s\n','physics create "pressure inlet" btype
"PRESSURE_INLET" face "inlet face1" "inlet face2"');
fprintf(fid,'%s\n','physics create "pressure outlet" btype
"PRESSURE_OUTLET" face "outlet face 2" "face.75"');
fprintf(fid,'%s\n','physics create "be wall" btype "WALL" face
"face outer le c a1" \'');
fprintf(fid,'%s\n','"face outer le c b1" "face outer below c1"
"face outer te c b1" \'');
fprintf(fid,'%s\n','"face airfoil le c a1" "face outer above c1"
"face airfoil above c1" \'');
fprintf(fid,'%s\n','"face airfoil le c1" "face airfoil below c1"
"face airfoil te c1" \'');
fprintf(fid,'%s\n','physics create "up wall" btype "WALL" face
"face.83" "face.87" \'');
fprintf(fid,'%s\n','"face.100" "face.109" "face.74" "face.79"
"face.107" "face.92" "face.96" "face.103"');
fprintf(fid,'%s\n','physics create "airfoil" btype "WALL" face
"face.94" "airfoil bottom face" "face.108" "face.104"');
fprintf(fid,'%s\n','physics create "air" ctype "FLUID" volume
"volume.11"');
fprintf(fid,'%s\n','physics create "periodic" btype "PERIODIC"
face "face.80" "face.84" \'');
fprintf(fid,'%s\n','"edge above right face" "edge below mid face"
"edge below left face" "face.110"');
fprintf(fid,'%s\n',[ 'export fluent5 "f13d',num2str(nr),'.msh' ]);
else
fprintf(fid,'%s\n',[ 'physics create "pressure inlet" btype
"PRESSURE_INLET" face "inlet face1" "face.' num2str(111+23*(ne-
3)) ''']);
fprintf(fid,'%s\n',[ 'physics create "pressure outlet" btype
"PRESSURE_OUTLET" face "outlet face 2" "face.' num2str(97+21*(ne-
3)) ''']);
fprintf(fid,'%s\n',[ 'physics create "be wall" btype "WALL" face
"face outer le c a1" \'']);
fprintf(fid,'%s\n',[ '"face outer le c b1" "face outer below c1"
"face outer te c b1" \'']);
fprintf(fid,'%s\n',[ '"face airfoil le c a1" "face outer above c1"
"face airfoil above c1" \'']);
fprintf(fid,'%s\n',[ '"face airfoil le c1" "face airfoil below c1"
"face airfoil te c1"']);
fprintf(fid,'%s\n',[ 'physics create "up wall" btype "WALL" face
"face.105" "face.109" "face.122" \'']);
fprintf(fid,'%s\n',[ '"face.131" "face.96" "face.101" "face.114"
"face.118" "face.125" "face.129"']);
fprintf(fid,'%s\n',[ 'physics create "airfoil" btype "WALL" face
"face.' num2str(116+23*(ne-3)) '' "airfoil bottom face" "face.'
num2str(130+31*(ne-3)) '' "airfoil te face"']);
fprintf(fid,'%s\n',[ 'physics create "air" ctype "FLUID" volume
"volume.11"']);
fprintf(fid,'%s\n',[ 'physics create "periodic" btype "PERIODIC"
face "edge above mid face" "face.' num2str(106+23*(ne-3)) '' \'']);
fprintf(fid,'%s\n',[ '"edge above right face" "edge below mid face"
"face.' num2str(110+23*(ne-3)) '' "face.' num2str(132+31*(ne-3))
''']);
fprintf(fid,'%s\n',[ 'export fluent5 "f13d',num2str(nr),'.msh' ]);
end
st=fclose(fid);
```

```
function xminind=edgeindex(filename)
% EDGEINDEX(filename) The function reads the data in file
"filename"
% and finds the index with smallest x-coordinate
% Input
% FILENAME: textstring with the filename
% Output
% XMININD: the appropriate index in the blade coordinate array
%
% Copyright: © David Lindström
% Authors: Narejo and David
% University of Trollhättan Uddevalla
% 20/11-2006

% Open the file with coordinates
(fid,message)=fopen(filename,'r');
if ~fid,disp(message),end

% Read the first row (number of coordinates)
nr=str2num(fgets(fid));

% Read the coordinates
s=fscanf(fid, '%g %g', [3 inf])';

% Find the smallest x coordinate and its index
xmin=min(s(:,1));
xminind=find(xmin==s(:,1));

% Check the number of coordinates
if nr~=size(s,1)
    disp('Number of coordinates is wrong')
    xminind=0;
end
fclose(fid);

function conxy=nodeco(filename,cno)
% NODECO(filename,x) The function reads the co-ordinates of
certain node from file "filename"
% Input
% FILENAME: textstring with the filename
% CNO: index number of node, to be entered as integer.
% Output
% CONXY: Get co-ordinate of any node point
% Copyright: © David Lindström
% Authors: Narejo and David
% University of Trollhättan Uddevalla
% 20/11-2006
% for minimum values

% Open the file with coordinates
(fid,message)=fopen(filename,'r');
if ~fid,disp(message),end

% Read the first row (number of coordinates)
nr=str2num(fgets(fid));

% Read the coordinates
s=fscanf(fid, '%g %g', [3 inf])';
```

```
% Find x & y co-ordinates and its index
xcor=s(cno,1);
ycor=s(cno,2);
coxy=[xcor,ycor];

% Check the total number of coordinates
if nr~=size(s,1)
    disp('Number of coordinates is wrong')
    xcnoind=0;
end
fclose(fid);

function coxy=edgeindexc(filename,x)
% EDGEINDEX(filename) The function reads the data from file
"filename"
% and finds the index with smallest x-coordinate
% Input
% FILENAME: text string with the filename
% if x=1 calculate minimum values, x=2 calculate maximumum values
% and otherwise it will return nothing.
% Output
% COXY: fetch minimum co-ordinates if x=1 and
% maximum co-ordinates if x=2
% Copyright: © David Lindström
% Authors: Narejo and David
% University of Trollhättan Uddevalla
% 20/11-2006
% for minmum values
if x==1
    % Open the file with coordinates
    [fid,message]=fopen(filename,'r');
    if ~fid,disp(message),end

    % Read the first row (number of coordinates)
    nr=str2num(fgets(fid));

    % Read the coordinates
    s=fscanf(fid, '%g %g', [3 inf]');

    % Find the smallest x & y coordinate and its index
    xmin=min(s(:,1));
    xminind=find(xmin==s(:,1));
    xcor=s(xminind,1);
    ycor=s(xminind,2);
    coxy=[xcor,ycor];

    % Check the number of coordinates
    if nr~=size(s,1)
        disp('Number of coordinates is wrong')
        xminind=0;
    end
    fclose(fid);
end

% for maximum values
if x==2

    % Open the file with coordinates
```

```
[fid,message]=fopen(filename,'r');
if ~fid,disp(message),end

% Read the first row (number of coordinates)
nr=str2num(fgets(fid));

% Read the coordinates
s=fscanf(fid, '%g %g', [3 inf]');

% Find the highest x & y coordinate and its index
xmax=max(s(:,1));
xmaxind=find(xmax==s(:,1));
xcor=s(xmaxind,1);
ycor=s(xmaxind,2);
coxy=[xcor,ycor];

% Check the number of coordinates
if nr~=size(s,1)
    disp('Number of coordinates is wrong')
    xmaxind=0;
end
fclose(fid);
end
```

## C. Appendix 3

This is a program written in Matlab for Fluent journal file and it generates journal file, can be read in Fluent.

```

function make_fluent_journal(nr)
% make_fluent_journal(nr)
% Generates fluent journal file
% Input
% NR: Simulation number
% Copyright: © David Lindström
% Authors: AA Narejo
% University of Trollhättan Uddevalla
% 20/09-2007
if nargin<1 ;
nr=2;
end
(fid,message)=fopen(['FLT3D',num2str(nr),'.jou'],'w');

fprintf(fid,'%s\n','file');
%Reading "d:\fnlprj\f13d2.msh"...

fprintf(fid,'%s\n',[ 'read-case
d:\fnlprj\\','FL3D',num2str(nr),'.msh']);
%read-profile d:\fnlprj\inlet1.prf

fprintf(fid,'%s\n','read-profile d:\fnlprj\inlet1.prf');
fprintf(fid,'%s\n','quit');

fprintf(fid,'%s\n','define');
fprintf(fid,'%s\n','models');
fprintf(fid,'%s\n','energy');
%Enable energy model? [no] yes
fprintf(fid,'%s\n','yes');
%Compute viscous energy dissipation? [no] no
fprintf(fid,'%s\n','no');
%include pressure work in energy equation? [no] no
fprintf(fid,'%s\n','no');
%include kinetic energy in energy equation? [no] no
fprintf(fid,'%s\n','no');
%Include diffusion at inlets? [yes] no
fprintf(fid,'%s\n','no');

% define/models> viscous
fprintf(fid,'%s\n','viscous');
%/define/models/viscous> kw-standard
fprintf(fid,'%s\n','kw-standard');
%Enable the standard k-omega turbulence model? [no] y
fprintf(fid,'%s\n','y');
%/define/models/viscous> kw-transitional
fprintf(fid,'%s\n','kw-transitional');
%Enable the k-omega transitional flow modification? [no] y
fprintf(fid,'%s\n','y');
fprintf(fid,'%s\n','quit');
fprintf(fid,'%s\n','quit');

%/define> materials
fprintf(fid,'%s\n','materials');

```

```
%/define/materials> change-create
fprintf(fid,'%s\n','change-create');
%material-name> air
fprintf(fid,'%s\n','air');
fprintf(fid,'%s\n','air');
%change Density? [no] y
fprintf(fid,'%s\n','y');
%Density methods: (constant ideal-gas incompressible-ideal-gas
boussinesq piecewise-linear piecewise-polynomial polynomial user-
defined)
%new method [constant] ideal-gas
fprintf(fid,'%s\n','ideal-gas');
%change Cp (Specific Heat)? [no] n
fprintf(fid,'%s\n','n');
%change Thermal Conductivity? [no] n
fprintf(fid,'%s\n','n');
%change Viscosity? [no] n
fprintf(fid,'%s\n','n');
%change Molecular Weight? [no] n
fprintf(fid,'%s\n','n');
%change L-J Characteristic Length? [no] n
fprintf(fid,'%s\n','n');
%change L-J Energy Parameter? [no] n
fprintf(fid,'%s\n','n');
%change Thermal Expansion Coefficient? [no] n
fprintf(fid,'%s\n','n');
%change Degrees of Freedom? [no] n
fprintf(fid,'%s\n','n');
%change Speed of Sound? [no] n
fprintf(fid,'%s\n','n');
fprintf(fid,'%s\n','quit');

%/define> boundary-conditions
fprintf(fid,'%s\n','boundary-conditions');
%/define/boundary-conditions> fluid
fprintf(fid,'%s\n','fluid');
%zone id/name [air] air
fprintf(fid,'%s\n','air');

%material-name [air]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Specify source terms? [no] no
fprintf(fid,'%s\n','no');
%Specify fixed values? [no] no
fprintf(fid,'%s\n','no');
%Motion Type: Stationary [yes] yes
fprintf(fid,'%s\n','yes');
%X-Origin of Rotation-Axis (m) [0] 0
fprintf(fid,'%s\n','0');
%Y-Origin of Rotation-Axis (m) [0] 0
fprintf(fid,'%s\n','0');
%Z-Origin of Rotation-Axis (m) [0] 0
fprintf(fid,'%s\n','0');
%X-Component of Rotation-Axis [0] 1
fprintf(fid,'%s\n','1');
%Y-Component of Rotation-Axis [0] 0
fprintf(fid,'%s\n','0');
%Z-Component of Rotation-Axis [1] 0
fprintf(fid,'%s\n','0');
```

```

%Deactivated Thread [no] no
fprintf(fid,'%s\n','no');
%Laminar zone? [no] no
fprintf(fid,'%s\n','no');
%Porous zone? [no] no
fprintf(fid,'%s\n','no');

%/define/boundary-conditions> wall
fprintf(fid,'%s\n','wall');
%zone id/name [airfoil] airfoil
fprintf(fid,'%s\n','airfoil');
%Wall Thickness (m) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Heat Generation Rate? [no] no
fprintf(fid,'%s\n','no');
%Heat Generation Rate (w/m3) [0] 0
fprintf(fid,'%s\n','0');
%material-name [aluminum]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Thermal BC Type [heat-flux]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Heat Flux? [no] no
fprintf(fid,'%s\n','no');
%Heat Flux (w/m2) [0] 0
fprintf(fid,'%s\n','0');
%Enable shell conduction? [no] no
fprintf(fid,'%s\n','no');
%Wall Motion [motion-bc-stationary]: Change current value? [no]
yes
fprintf(fid,'%s\n','yes');
%Wall Motion [motion-bc-stationary]> motion-bc-moving
fprintf(fid,'%s\n','motion-bc-moving');
%Shear Boundary Condition [shear-bc-noslip]: Change current value?
[no] no
fprintf(fid,'%s\n','no');
%Define wall motion relative to adjacent cell zone? [yes] yes
fprintf(fid,'%s\n','yes');
%Apply a rotational velocity to this wall? [no] yes
fprintf(fid,'%s\n','yes');
%Define wall velocity components? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Wall Roughness Height? [no] no
fprintf(fid,'%s\n','no');
%Wall Roughness Height (m) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Wall Roughness Constant? [no] no
fprintf(fid,'%s\n','no');
%Wall Roughness Constant [0.5] 0.5
fprintf(fid,'%s\n','0.5');
%Rotation Speed (rad/s) [0] 0
fprintf(fid,'%s\n','0');
%X-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Y-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Z-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%X-Component of Rotation-Axis Direction [0] 1
fprintf(fid,'%s\n','1');

```

```
%Y-Component of Rotation-Axis Direction [0] 0
fprintf(fid,'%s\n','0');
%Z-Component of Rotation-Axis Direction [1] 0
fprintf(fid,'%s\n','0');

%/define/boundary-conditions> wall
fprintf(fid,'%s\n','wall');
%zone id/name [airfoil] be_wall
fprintf(fid,'%s\n','be_wall');
%Wall Thickness (m) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Heat Generation Rate? [no] no
fprintf(fid,'%s\n','no');
%Heat Generation Rate (w/m3) [0] 0
fprintf(fid,'%s\n','0');
%material-name [aluminum]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Thermal BC Type [heat-flux]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Heat Flux? [no] no
fprintf(fid,'%s\n','no');
%Heat Flux (w/m2) [0] 0
fprintf(fid,'%s\n','0');
%Enable shell conduction? [no] no
fprintf(fid,'%s\n','no');
%Wall Motion [motion-bc-stationary]: Change current value? [no]
yes
fprintf(fid,'%s\n','yes');
%Wall Motion [motion-bc-stationary]> motion-bc-moving
fprintf(fid,'%s\n','motion-bc-moving');
%Shear Boundary Condition [shear-bc-noslip]: Change current value?
[no] no
fprintf(fid,'%s\n','no');
%Define wall motion relative to adjacent cell zone? [yes] yes
fprintf(fid,'%s\n','yes');
%Apply a rotational velocity to this wall? [no] yes
fprintf(fid,'%s\n','yes');
%Define wall velocity components? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Wall Roughness Height? [no] no
fprintf(fid,'%s\n','no');
%Wall Roughness Height (m) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Wall Roughness Constant? [no] no
fprintf(fid,'%s\n','no');
%Wall Roughness Constant [0.5] 0.5
fprintf(fid,'%s\n','0.5');
%Rotation Speed (rad/s) [0] 0
fprintf(fid,'%s\n','0');
%X-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Y-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Z-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%X-Component of Rotation-Axis Direction [0] 1
fprintf(fid,'%s\n','1');
%Y-Component of Rotation-Axis Direction [0] 0
fprintf(fid,'%s\n','0');
```

```

%Z-Component of Rotation-Axis Direction [1] 0
fprintf(fid,'%s\n','0');

%/define/boundary-conditions> wall
fprintf(fid,'%s\n','wall');
%zone id/name [airfoil] up_wall
fprintf(fid,'%s\n','up_wall');
%Wall Thickness (m) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Heat Generation Rate? [no] no
fprintf(fid,'%s\n','no');
%Heat Generation Rate (w/m3) [0] 0
fprintf(fid,'%s\n','0');
%material-name [aluminum]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Thermal BC Type [heat-flux]: Change current value? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Heat Flux? [no] no
fprintf(fid,'%s\n','no');
%Heat Flux (w/m2) [0] 0
fprintf(fid,'%s\n','0');
%Enable shell conduction? [no] no
fprintf(fid,'%s\n','no');
%Wall Motion [motion-bc-stationary]: Change current value? [no]
yes
fprintf(fid,'%s\n','yes');
%Wall Motion [motion-bc-stationary]> motion-bc-moving
fprintf(fid,'%s\n','motion-bc-moving');
%Shear Boundary Condition [shear-bc-noslip]: Change current value?
[no] no
fprintf(fid,'%s\n','no');
%Define wall motion relative to adjacent cell zone? [yes] yes
fprintf(fid,'%s\n','yes');
%Apply a rotational velocity to this wall? [no] yes
fprintf(fid,'%s\n','yes');
%Define wall velocity components? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Wall Roughness Height? [no] no
fprintf(fid,'%s\n','no');
%Wall Roughness Height (m) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Wall Roughness Constant? [no] no
fprintf(fid,'%s\n','no');
%Wall Roughness Constant [0.5] 0.5
fprintf(fid,'%s\n','0.5');
%Rotation Speed (rad/s) [0] 0
fprintf(fid,'%s\n','0');
%X-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Y-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Z-Position of Rotation-Axis Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%X-Component of Rotation-Axis Direction [0] 1
fprintf(fid,'%s\n','1');
%Y-Component of Rotation-Axis Direction [0] 0
fprintf(fid,'%s\n','0');
%Z-Component of Rotation-Axis Direction [1] 0
fprintf(fid,'%s\n','0');

```

```
%/define/boundary-conditions> periodic
fprintf(fid,'%s\n','periodic');
%zone id/name [periodic] periodic
fprintf(fid,'%s\n','periodic');
%Rotationally Periodic? [no] y
fprintf(fid,'%s\n','y');

%/define/boundary-conditions> pressure-inlet
fprintf(fid,'%s\n','pressure-inlet');
%zone id/name [pressure_inlet]
fprintf(fid,'%s\n','pressure_inlet');
%Use Profile for Gauge Total Pressure? [no] y
fprintf(fid,'%s\n','y');
%Use UDF Profile for Gauge Total Pressure? [no] n
fprintf(fid,'%s\n','no');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp press presso)
%data name ["span"] "press"
fprintf(fid,'%s\n',"press");
%Use Profile for Supersonic/Initial Gauge Pressure? [no] n
fprintf(fid,'%s\n','n');
%Supersonic/Initial Gauge Pressure (pascal) [0] 0
fprintf(fid,'%s\n','0');
%Use Profile for Total Temperature? [no] y
fprintf(fid,'%s\n','y');
%Use UDF Profile for Total Temperature? [no] n
fprintf(fid,'%s\n','n');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp press presso)
%data name ["span"] "temp"
fprintf(fid,'%s\n',"temp");
%Direction Specification Method: Direction Vector [no] yes
fprintf(fid,'%s\n','yes');
%Coordinate System: Cartesian (X, Y, Z) [yes] no
fprintf(fid,'%s\n','no');
%Coordinate System: Cylindrical (Radial, Tangential, Axial) [no]
yes
fprintf(fid,'%s\n','yes');
%Use Profile for Radial-Component of Flow Direction? [no] yes
fprintf(fid,'%s\n','yes');
%Use UDF Profile for Radial-Component of Flow Direction? [no] no
fprintf(fid,'%s\n','no');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp press presso)
%data name ["span"] "y"
fprintf(fid,'%s\n',"y");
%Use Profile for Tangential-Component of Flow Direction? [no] yes
fprintf(fid,'%s\n','yes');
```

```

%Use UDF Profile for Tangential-Component of Flow Direction? [no]
no
fprintf(fid,'%s\n','no');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp presso)
%data name ["span"] "z"
fprintf(fid,'%s\n','"z"');
%Use Profile for Axial-Component of Flow Direction? [no] yes
fprintf(fid,'%s\n','yes');
%Use UDF Profile for Axial-Component of Flow Direction? [no] no
fprintf(fid,'%s\n','no');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp presso)
%data name ["span"] "x"
fprintf(fid,'%s\n','"x"');
%Turbulence Specification Method: K and Omega [yes] yes
fprintf(fid,'%s\n','yes');
%Use Profile for Turb. Kinetic Energy? [no] no
fprintf(fid,'%s\n','no');
%Turb. Kinetic Energy (m2/s2) [1] 1
fprintf(fid,'%s\n','1');
%Use Profile for Spec. Dissipation Rate? [no] no
fprintf(fid,'%s\n','no');
%Spec. Dissipation Rate (1/s) [1] 1
fprintf(fid,'%s\n','1');

%/define/boundary-conditions> pressure-outlet
fprintf(fid,'%s\n','pressure-outlet');
%zone id/name [pressure_outlet] pressure_outlet
fprintf(fid,'%s\n','pressure_outlet');
%Use Profile for Gauge Pressure? [no] y
fprintf(fid,'%s\n','y');
%Use UDF Profile for Gauge Pressure? [no] n
fprintf(fid,'%s\n','n');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp presso)
%data name ["span"] "presso"
fprintf(fid,'%s\n','"presso"');
%Radial Equilibrium Pressure Distribution [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Backflow Total Temperature? [no] yes
fprintf(fid,'%s\n','yes');
%Use UDF Profile for Backflow Total Temperature? [no] no
fprintf(fid,'%s\n','no');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp presso)
%data name ["span"] "temp"

```

```

fprintf(fid,'%s\n','"temp"');
%Backflow Direction Specification Method: Direction Vector [no] no
fprintf(fid,'%s\n','no');
%Backflow Direction Specification Method: Normal to Boundary [yes]
yes
fprintf(fid,'%s\n','yes');
%Turbulence Specification Method: K and Omega [yes] yes
fprintf(fid,'%s\n','yes');
%Use Profile for Backflow Turb. Kinetic Energy? [no] no
fprintf(fid,'%s\n','no');
%Backflow Turb. Kinetic Energy (m2/s2) [1] 1
fprintf(fid,'%s\n','1');
%Use Profile for Backflow Spec. Dissipation Rate? [no] no
fprintf(fid,'%s\n','no');
%Backflow Spec. Dissipation Rate (1/s) [1] 1
fprintf(fid,'%s\n','1');
%Specify targeted mass-flow rate [no] no
fprintf(fid,'%s\n','no');

%/define/boundary-conditions> zone-type
fprintf(fid,'%s\n','zone-type');
%zone id/name [] default-interior
fprintf(fid,'%s\n','default-interior');
%internal-type> fan
fprintf(fid,'%s\n','fan');
%/define/boundary-conditions> fan
fprintf(fid,'%s\n','fan');
%zone id/name [default-interior] default-interior
fprintf(fid,'%s\n','default-interior');
%Flow Direction (-1,0,1) [1] 1
fprintf(fid,'%s\n','1');
%Calculate Pressure-Jump from Average Conditions? [no] y
fprintf(fid,'%s\n','y');
%methods: (constant piecewise-linear piecewise-polynomial
polynomial)
%new method [polynomial] polynomial
fprintf(fid,'%s\n','polynomial');
%number of coefficients [1] 1
fprintf(fid,'%s\n','1');
%coeff 1 (pascal) [0] 0
fprintf(fid,'%s\n','0');
%Limit Polynomial Velocity Range? [no] no
fprintf(fid,'%s\n','no');
%Polynomial Range: Minimum Velocity Magnitude (m/s) [0] 0
fprintf(fid,'%s\n','0');
%Polynomial Range: Maximum Velocity Magnitude (m/s) [1e+10]
fprintf(fid,'%s\n','1e+10');
%Profile Specification of Pressure-Jump? [no] no
fprintf(fid,'%s\n','no');
%Use Profile for Pressure Jump Profile? [no] no
fprintf(fid,'%s\n','no');
%Pressure Jump Profile (pascal) [0] 0
fprintf(fid,'%s\n','0');
%Swirl-Velocity Specification? [no] yes
fprintf(fid,'%s\n','yes');
%Radial-Velocity Polynomial Coefficient(1) [()]
fprintf(fid,'%s\n','');
%Tangential-Velocity Polynomial Coefficient(1) [()]
fprintf(fid,'%s\n','');

```

```

%Fan Hub Radius (m) [1e-06] 1.5
fprintf(fid,'%s\n','1.5');
%X-Coordinate of Fan Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Y-Coordinate of Fan Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%Z-Coordinate of Fan Origin (m) [0] 0
fprintf(fid,'%s\n','0');
%X-Component of Fan Axis [1] 1
fprintf(fid,'%s\n','1');
%Y-Component of Fan Axis [0] 0
fprintf(fid,'%s\n','0');
%Z-Component of Fan Axis [0] 0
fprintf(fid,'%s\n','0');
%Profile Specification of Tangential Velocity? [no] y
fprintf(fid,'%s\n','y');
%Use Profile for Tangential Velocity Profile? [no] y
fprintf(fid,'%s\n','y');
%Use UDF Profile for Tangential Velocity Profile? [no] n
fprintf(fid,'%s\n','n');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp press presso)
%data name ["span"] "z"
fprintf(fid,'%s\n','"z"');
%Profile Specification of Radial Velocity? [no] y
fprintf(fid,'%s\n','y');
%Use Profile for Radial Velocity Profile? [no] y
fprintf(fid,'%s\n','y');
%Use UDF Profile for Radial Velocity Profile? [no] no
fprintf(fid,'%s\n','no');
%profile names list: (turb-prof turb-prof turb-prof turb-prof
turb-prof turb-prof turb-prof)
%profile name ["turb-prof"]
fprintf(fid,'%s\n','');
%data names list: (span x y z temp press presso)
%data name ["span"] "y"
fprintf(fid,'%s\n','"y"');
fprintf(fid,'%s\n','quit');
fprintf(fid,'%s\n','quit');
%solve
fprintf(fid,'%s\n','solve');
%Dicretization shceme
fprintf(fid,'%s\n','controls');
fprintf(fid,'%s\n','set');
fprintf(fid,'%s\n','discretization-scheme');
fprintf(fid,'%s\n','pressure');
fprintf(fid,'%s\n','14');
fprintf(fid,'%s\n','density');
fprintf(fid,'%s\n','4');
fprintf(fid,'%s\n','k');
fprintf(fid,'%s\n','4');
fprintf(fid,'%s\n','mom');
fprintf(fid,'%s\n','4');
fprintf(fid,'%s\n','omega');
fprintf(fid,'%s\n','4');
fprintf(fid,'%s\n','temperature');

```

```
fprintf(fid,'%s\n','4');
fprintf(fid,'%s\n','quit');
fprintf(fid,'%s\n','quit');
%solve> initialize
fprintf(fid,'%s\n','initialize');
fprintf(fid,'%s\n','set-defaults');
fprintf(fid,'%s\n','set-defaults');
fprintf(fid,'%s\n','pressure');
fprintf(fid,'%s\n','150000');
fprintf(fid,'%s\n','x-velocity');
fprintf(fid,'%s\n','35');
fprintf(fid,'%s\n','quit');
fprintf(fid,'%s\n','quit');
%solve> iterate
fprintf(fid,'%s\n','iterate');
%Number of iterations [1] 1
fprintf(fid,'%s\n','1');
fprintf(fid,'%s\n','quit');
fprintf(fid,'%s\n','file');
fprintf(fid,'%s\n','stop-journal');
fprintf(fid,'%s\n','quit');
fclose(fid);
end
```

## D. Appendix 4

Inlet Boundary conditions Data

Spanwise position	Flow angle in Deg.	Meridional flow angle in Deg.	Temperature *273	Total pressure in atm
0.000000E+00	-59.43130	2.619064	1.105988	0.9868594
2.557495E-04	-59.43130	2.619064	1.105988	0.9868594
8.435168E-04	-58.85743	2.652396	1.174264	1.274678
1.606816E-03	-57.76258	2.731959	1.176256	1.361142
2.598043E-03	-56.62917	2.818759	1.177630	1.443318
3.884952E-03	-55.42479	2.915369	1.178326	1.513982
5.556100E-03	-54.17538	3.014502	1.178416	1.568451
7.726197E-03	-52.93872	3.106767	1.178054	1.606046
1.054400E-02	-51.77590	3.184270	1.177419	1.630327
1.420287E-02	-50.72655	3.243554	1.176655	1.646610
1.895372E-02	-49.79868	3.285385	1.175859	1.659069
2.512234E-02	-48.97790	3.310500	1.175116	1.669679
3.313158E-02	-48.24751	3.316039	1.174522	1.678792
4.353007E-02	-47.60294	3.296685	1.174172	1.686111
5.703020E-02	-47.03962	3.243357	1.174119	1.691511
7.455606E-02	-46.52813	3.141864	1.174345	1.695608
9.743783E-02	-46.00704	2.987197	1.174743	1.699904
0.1273863	-45.40612	2.742301	1.175129	1.704832
0.1612984	-44.69377	2.423107	1.175304	1.708092
0.1951082	-43.88348	2.092746	1.175130	1.707394
0.2288211	-43.01268	1.765015	1.174532	1.702219
0.2624398	-42.11658	1.434348	1.173459	1.692859
0.2959610	-41.21555	1.102778	1.171885	1.679758
0.3293820	-40.31385	0.7776054	1.169805	1.663665
0.3627037	-39.40242	0.4613946	1.167261	1.645613
0.3959275	-38.46272	0.1574713	1.164347	1.626891
0.4290509	-37.49093	-0.1273896	1.161219	1.608812
0.4620632	-36.50381	-0.4151635	1.158066	1.591674
0.4949866	-35.55301	-0.6899090	1.155097	1.574974
0.5277794	-34.70527	-0.9332208	1.152455	1.559023
0.5604572	-33.97100	-1.174916	1.149990	1.543176
0.5930238	-33.31676	-1.409164	1.147359	1.525417
0.6254687	-32.69381	-1.629327	1.144265	1.505117
0.6578017	-32.06256	-1.845203	1.140604	1.483117
0.6900359	-31.41363	-2.063039	1.136563	1.460893
0.7221613	-30.80511	-2.260708	1.132685	1.439996
0.7541652	-30.32141	-2.403303	1.129591	1.421315
0.7860591	-30.00074	-2.469016	1.127611	1.405084
0.8178464	-29.81214	-2.443120	1.126704	1.391583
0.8495308	-29.72282	-2.355684	1.126649	1.380522
0.8810970	-29.79248	-2.233298	1.127376	1.370620
0.9089404	-30.14686	-2.090034	1.128914	1.360430
0.9303187	-30.90148	-1.936189	1.131184	1.348573
0.9467465	-32.09114	-1.768655	1.133917	1.334475
0.9593778	-33.65088	-1.611352	1.136744	1.318937

---

0.9690936	-35.45617	-1.487164	1.139379	1.303306
0.9765694	-37.37168	-1.402617	1.141695	1.288574
0.9823228	-39.28510	-1.350302	1.143675	1.275101
0.9867513	-41.11705	-1.318790	1.145344	1.262795
0.9901636	-42.81532	-1.298874	1.146734	1.251362
0.9927936	-44.34659	-1.284930	1.147866	1.240445
0.9948183	-45.69057	-1.274322	1.148751	1.229681
0.9963774	-46.84079	-1.265624	1.149391	1.218666
0.9975778	-47.81339	-1.258278	1.149784	1.206843
0.9985020	-48.66262	-1.251730	1.149928	1.193468
0.9992136	-49.49541	-1.242444	1.149840	1.178180
0.9997616	-49.95850	-1.234605	1.138697	1.127620
1.000000	-49.95850	-1.234605	1.138697	1.127620

Outlet boundary conditions, the static pressure can be set 0.9 to 1.05 times of the inlet of total pressure.