TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A

Report

On

Francis Runner Design For Low Head and Calculation of Degree of Reaction

Submitted by

Subodh Khanal (077bme040)

Surya Bahadur Waiba (077bme051)

Samir Karki (077bme032)

Nitesh Raj Dharel (077bme023)

Submitted to

DEPARTMENT OF MECHANICAL ENGINEERING

August 14

ABSTRACT

This research focuses on designing the Francis runner for low head and high specific speed. Inhouse Matlab codes are used to figure out the meridional plane of the runner blade and profiles.Ansys Bladegen is used to make the primary profile of the runner passage accounting for beta angle distribution at each layer ,TurboGrid is used for meshing and Ansys CFX is used in simulating the runner blade to obtain various parameters and efficiency .The research further takes a dive in the study and determination of the optimum degree of reaction.

Keywords:

Francis runner, low head, degree of reaction, Bovet method,Ansys

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Mr. Neeraj Adhikari, Associate Professor, Pulchowk Campus for supervising our work. With his proper mentorship and guidance, we were able to stay on track and complete this project.

We would also like to express our gratitude to Er. Bikki Chhantyal and Er. Srijan Satyal for their assistance in the project with the resources and research methodology.

Lastly, we would like to thank all our colleagues from Department of Mechanical engineering for their encouragement and support till the completion of this project. Their help was vital in understanding different terminology and processes to which were not familiar with.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MATLAB Mathematical Labaratory

ANSYS Analysis System

NOMENCLATURE

H	Head (m)
Q	Flow rate (m)
N	Rotational speed(rpm)
C1	Absolute inlet velocity(m/s)
C2	Absolute outlet velocity(m/s)
Cu1	Whirl velocity at inlet of runner (m/s)
Cm1	Meridional velocity at inlet (m/s)
Cm2	Meridional velocity at outlet (m/s)
g	Acceleration due to gravity (m/s^2)
n_o	Non-dimensional specific speed
Ns	Specific speed
P	Power output (watt)
Q	Flow rate(m2 /s)
u1	Runner peripheral velocity at leading edge
u2	Runner peripheral velocity at trailing edge
η_h	Hydraulic Efficiency
η_o	Overall efficiency
θ	Wrap angle (degree)
$\beta 1$	Inlet vane angle (degree)
$\beta 2$	Outlet vane angle (degree)
N_{st}	Number of streamlines
ω	Angular speed(rad/s)

CHAPTER ONE INTRODUCTION

1.1 Background

1.1.1 Francis turbine

Francis Turbine is a mixed-flow reaction turbine where the water or fluid enters runner radially and exits axially. It is based on the principle of conservation of angular momentum i.e., the change in pressure energy inside the runner is converted into kinetic energy thus resulting in the production of hydroelectricity. Also, the impulse effect is seen in the runner blades. Francis type turbines have a wide range of specific speeds. Due to its wide application, it can be used in low head as well. Francis type turbines are composed of five components. These are volute, stationary vanes, guide vanes, runner and draft tube. Out of these, runners are considered as the heart of the Francis turbine. The runner design parameters also affect the design of other parts of the turbine. So, the design and optimization of the runner is crucial.[1] High level of efficiency and cavitation free flow on the runner blades are the necessary requirements according to Daneshkah, K. and Zangeneh, M. [5].

1.1.2 Degree of Reaction

It is the change in pressure energy inside the runner per unit change in the total energy inside the runner. Besides specific speed, the reaction ratio is a very important parameter for the blade loading and cavitation performance of a reaction turbine [2]. Mathematically,

Degree of Reaction (R) = $1-(C1^2-C2^2)/(2*g*H_e)$

Where, C1= absolute inlet velocity

C2 = absolute outlet velocity

$H_e = 1/g*(Cm1*u1 \pm Cm2*u2)$

1.2. Objectives

The main objective of the project is to design a Francis runner for specified values of flow rate, head and rotational speed.

1.2.1. Specific Objectives

- To calculate the dimensions of the runner and design it.
- To simulate the design using CFD tools and determine the optimum degree of reaction.

1.4 Software used

- MATLAB
- ANSYS TurboGRID
- ANSYS BLADEGEN
- ANSYS CFX

CHAPTER 2: LITERATURE REVIEW

Designing the Francis runner for different heads is important for the better production of hydropower. Various articles are present discussing different methods for the design of the francis runner. Experimental model test method, Direct method, Inverse method, Bovet and Conformal mapping are some of the widely used methods [6]. Out of these, Bovet method was widely followed and used to determine different runner dimensions. "A numerical case study: Bovet approach to design a Francis turbine runner ", Kocak et.al, is an article based on Bovet method to design runner blade [6]. Milos, T. et al studied the use of CAD for the optimization of the shape of the runner blade [8]. Asbina Baral, Sirapa Shrestha and Suraj Subedi submitted the design tool to determine the dimensions of the low head Francis runner using MATLAB [7]. The relevant resources to determine the optimum degree of reaction for the low head had a few relevant articles available.

CHAPTER 3: METHODOLOGY

This runner design approach is totally based on the Bovet method. The input parameters were taken, and the required meridional profile was created in the MATLAB. The profile incorporated the necessary coordinates for the design of 2D model of the runner which was then transferred into ANSYS Bladegen to get the runner profile. Smoothing the runner curves to get the exact profile was necessary. An overview of the processes performed is represented diagrammatically.

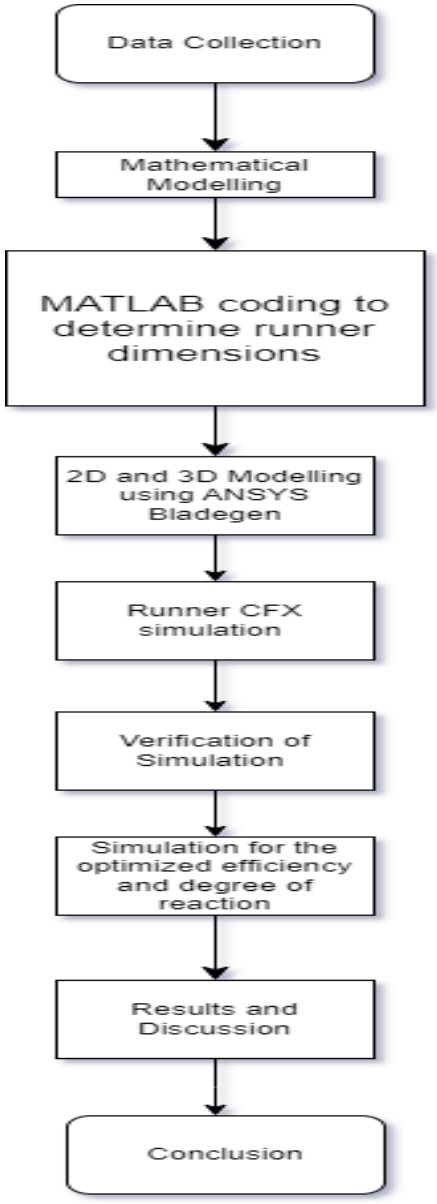

Figure 3.1: Flowchart on Design Process

3.1 Data Collection

The input parameters for the design of the runner are taken and listed below;

S.N	Design Parameters	Value
1.	Flow rate(Q)	0.12 m^3/s
2.	Head(H)	15 m
3.	Rotational Speed(N)	1500rpm
4.	No of streamlines	3

Figure 3.2: Data Table

3.2 Meridional view

In order to get the meridional view, BOVET approach was performed in Matlab. Through this approach we determined the parameters based on a dimensionless parameter called specific speed number (n_o). Bovet provided the set of formulas to calculate the required parameters where the value of nominal radius r_{2e} is set as 1. The value of non-dimensional specific speed is given by:

$$n_o = \omega*(Q \pi)^{1/2}/ (2gH)^{3/4}$$

Based upon the value of 'n_o' other dimensions were determined using the following formula.

$$R_{2e} = (Q/(\pi\omega\psi_{2e})^{1/2}$$

$$bo = 0.8(2-n0)n0$$

$$r_{oi} = y_{mi} = 0.7 + 0.16\,(n_o + 0.08)$$

$$r_{oe} = r_{li} = 0.493\,/n_o^{(2/3)}\ (n0 < 0.275)$$

$$r_{oe} = 1.255 - 0.3n_o (n_o > 0.275)$$

$$l_i = 3.2 + 3.2(2 - n_o)n_o$$

$$l_e = 2.4 - 1.9(2 - n_o)n_o$$

$$y_{2e} = r_{oe} - 1$$

$$x_e = 0.5 \text{ for the Francis runner}$$

The meridional profile is fixed by the equation of hub, shroud, and parameters; yme, ymi, x2e and y2e, which are calculated as,

$y_{2e} = r_{oi} - r_{2e} = r_{oi} - 1$

$y_{me} = y_{2e}/(y_{2e}/y_{me})$

The actual dimensions of the runner is obtained by multiplying the obtained parameters with R_{2e}

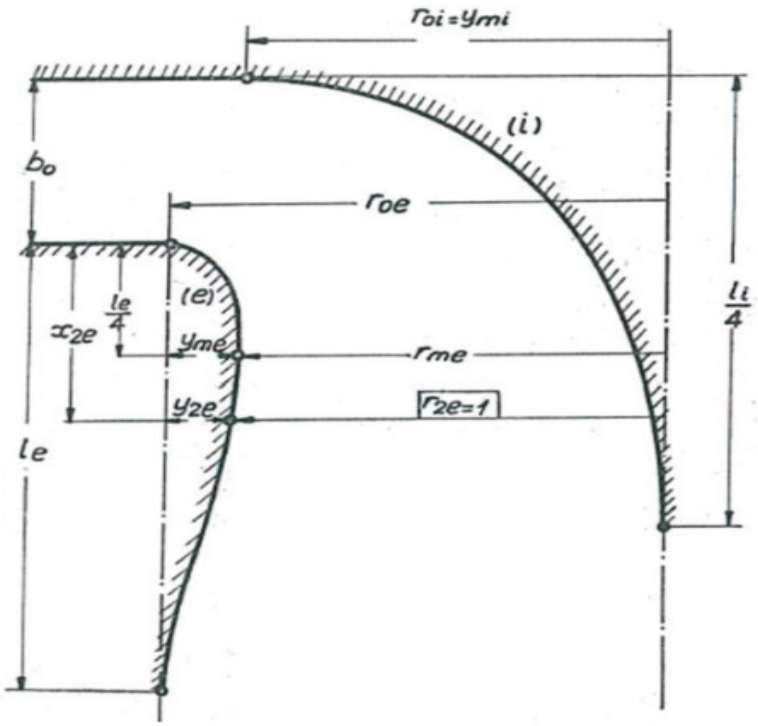

Fig 3.3: Parameters of runner channel in Meridional plane

3.3 CALCULATIONS

The necessary parameter defining the flow is calculated and their respective value is given.

Non-dimensional specific speed (n_o) = 0.432

Specific speed (N_s) = 202.76

Here, the range of N_s defines the runner type.

Slow runner N_s = 60 to 120

Slow runner N_s = 120 to 180

Slow runner N_s = 180 to 300

And all the other parameters were obtained from CFX analysis.

3.4 MATLAB Coding

BOVET method was mathematically modelled into the MATLAB code to determine the different parameters of the meridional plane. The input parameters for the program were flow rate(Q), head(H), rotational speed(N) and no. of streamlines (N_{st}). The output data's obtained from the BOVET method were interpolated which resulted in the curves of hub, shroud and different streamlines as well. The intersection points for the leading and trailing edges with the hub and shroud were directly displayed from the program. The coordinates from these intersection points were taken as reference for the blade design after denormalization of the BOVET parameters multiplying by $R_{2e.}$ The MATLAB codes used in this are also provided. Input for Ansys Bladegen was the intersection of hub and shroud curves with the leading and trailing edges.

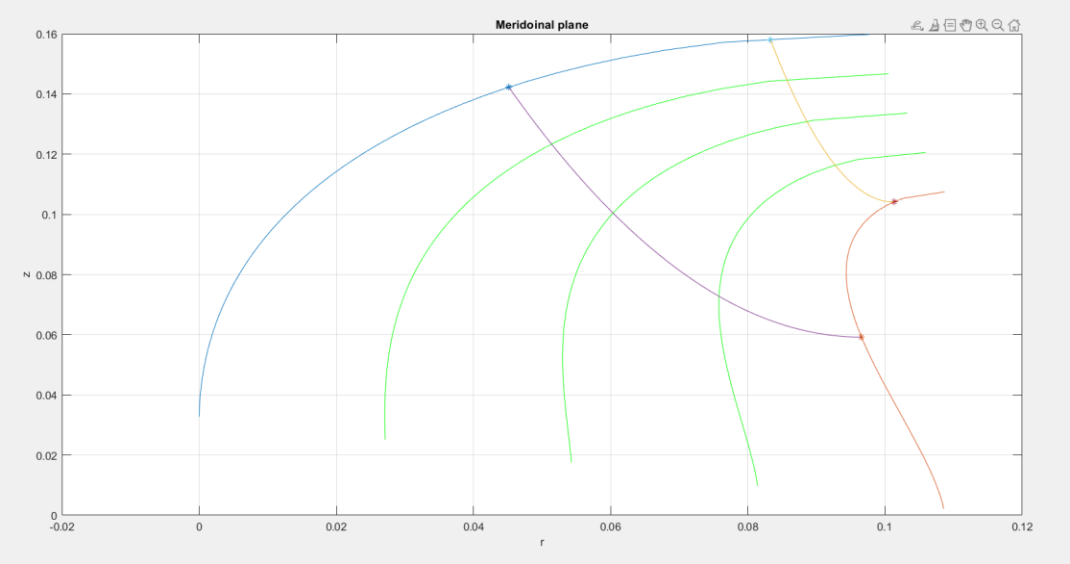

Fig 3.4: Meridonial profile from Matlab for given parameters

3.5 Design and Modelling

The parametric coordinates obtained from Matlab were dimensionalized and presented as an input for the Bladegen for the design of the Francis runner blade. The intersection coordinates for the leading and trailing edges with the hub and shroud, the beta angle (β) and thickness (t) were given input.15 runner blades with 3 streamlines in between hub and shroud curve were designed. The program generated a general 2D view of the Meridional plane where the curves of the edges were not clear and smooth. Continuous smoothing and fitting of data resulted in the best and optimum design of the blade. Curve fitting was done for each curve with the help of data obtained from Matlab codes. Beta angles and thickness for each layer were specified to make the runner blade smooth.

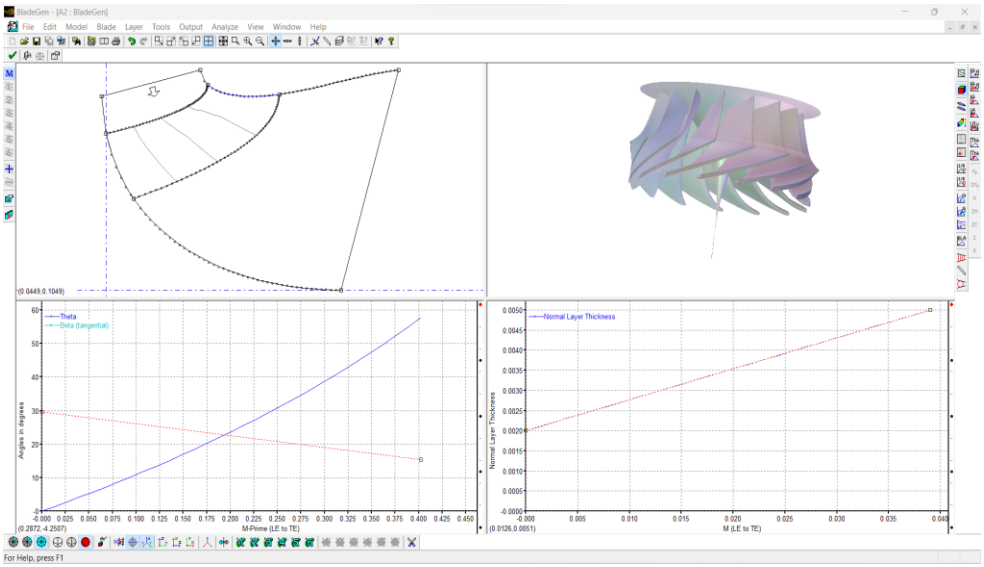

Fig 3.5: Bladegen model

A single runner blade was meshed using Turbogrid based on the y+ value of 1 for SST k-epsilon turbulence model. It resulted in 282126 nodes and 264670 elements.

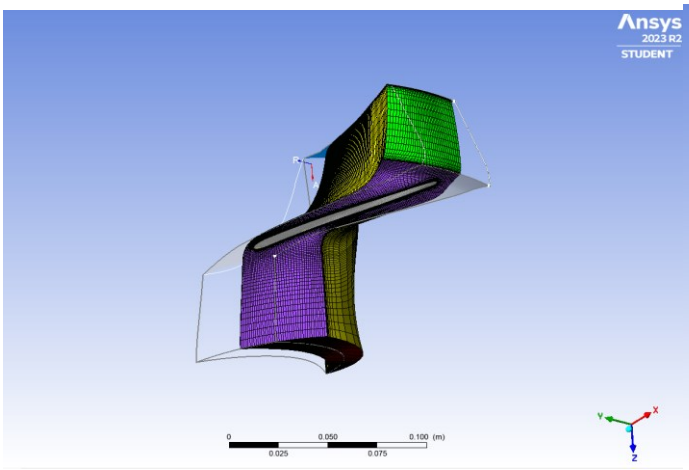

Fig 3.5.1:Meshed profile

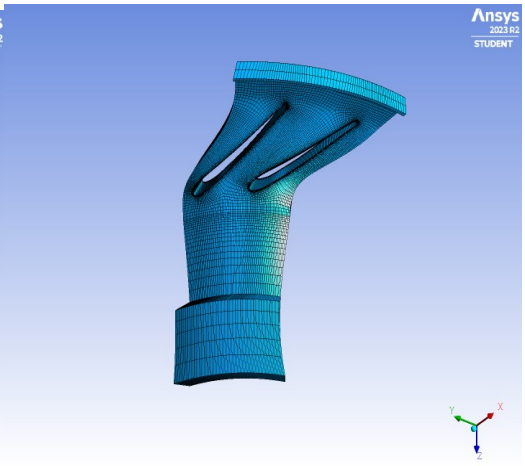

Fig 3.5.2: Mesh elements at span 50 view

3.5.1 Boundary Condition

In accordance with the available input,mass flow for inlet and static pressure for outlet was specified.The passage of runner was specified rotating in clockwise direction with 1500 rpm.

Boundaries	
Boundary - R1 Inlet	
Type	INLET
Location	Passage INFLOW
Settings	
Flow Direction	Cylindrical Components
Unit Vector Axial Component	0.0000e+0
Unit Vector Theta Component	9.3791e-1
Unit Vector r Component	3.4687e-1
Flow Regime	Subsonic
Mass And Momentum	Mass Flow Rate
Mass Flow Rate	1.1977e+2 [kg s^-1]
Mass Flow Rate Area	Total for All Sectors
Turbulence	Medium Intensity and Eddy Viscosity Ratio

Boundary - R1 Outlet	
Type	OUTLET
Location	OUTBlock OUTFLOW
Settings	
Flow Regime	Subsonic
Mass And Momentum	Average Static Pressure
Pressure Profile Blend	5.0000e-2
Relative Pressure	1.0000e+0 [atm]
Pressure Averaging	Average Over Whole Outlet

Fig 3.5.1.1 Boundary conditions

3.5.2 Post Processing

The 3D view of the runner along with hub and shroud was obtained.

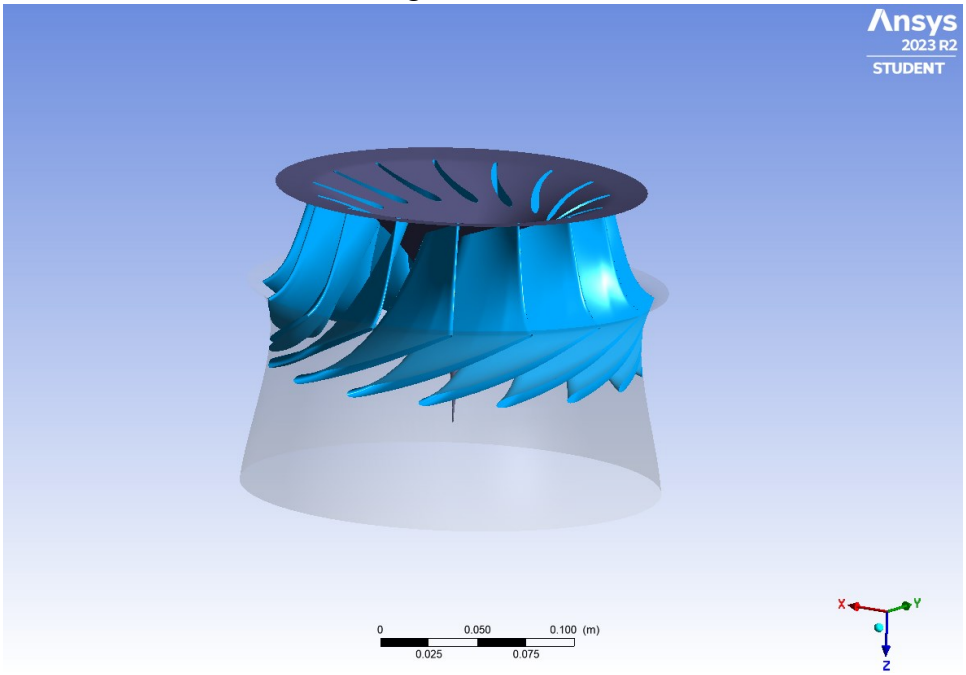

Fig 3.5.2.1: 3D isometric view of blade geometry

Degree of reaction is inherently depended upon the guide vane angles, pressure drop and velocity differences at the inlet and outlet. So, we observe the plots and contours of these parameters which affect degree of reaction directly.

The water at the trailing edge of the turbine blades moves faster due to the combination of centrifugal force and blade angle, which directs water outward and accelerates it. The water at the leading edge moves slower because of shallower blade angles and lower pressure, causing it to be directed towards the hub and resulting in slower velocity. The following picture depicts the velocity difference.

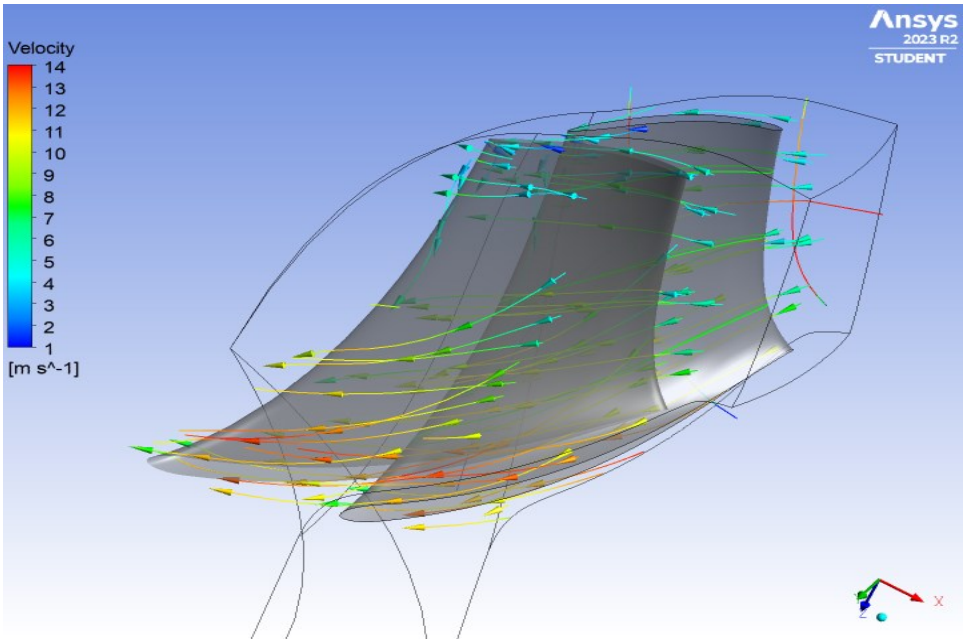

Fig 3.5.2.2 : Velocity Streamlines blade TE view

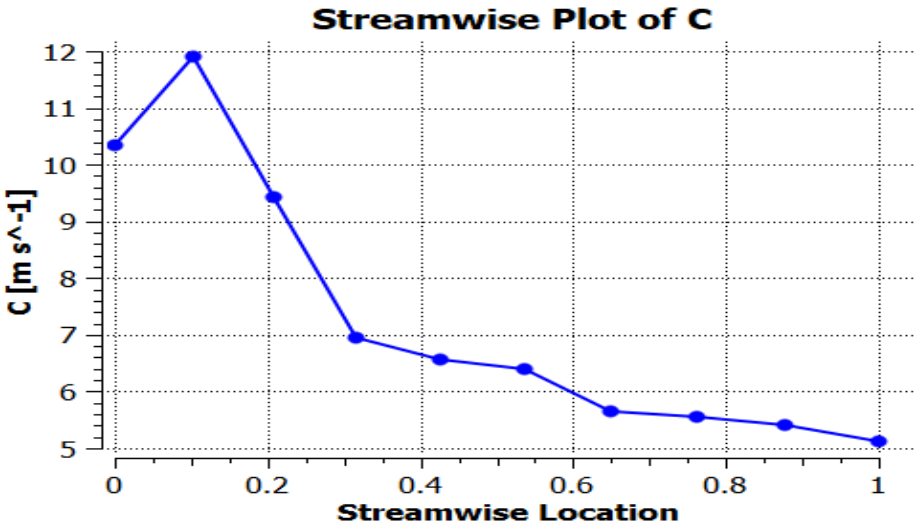

Fig 3.5.2.3: Streamwise plot of peripheral velocity

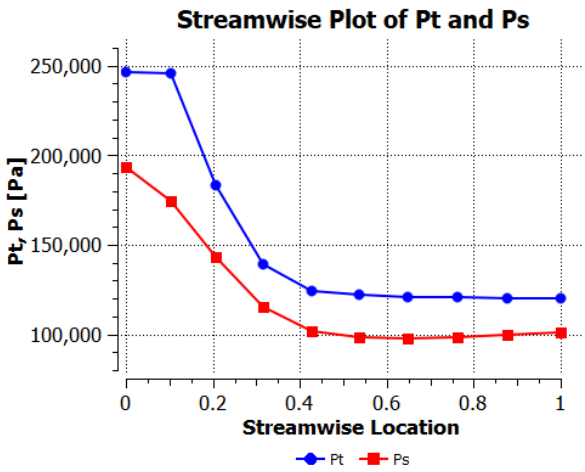

Fig 3.5.2.4: Streamwise plot of total and static pressure

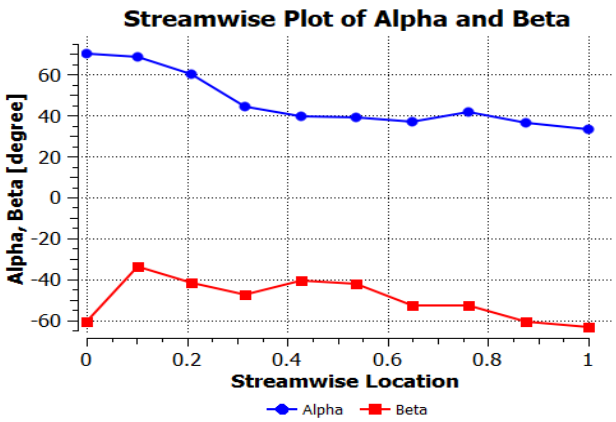

Fig 3.5.2.5: Streamwise plot of alpha and beta

The pressure loss from leading edge to trailing edge accounts for the degree of reaction the francis turbine.The following total pressure contour depicts the pressure loss between the two.

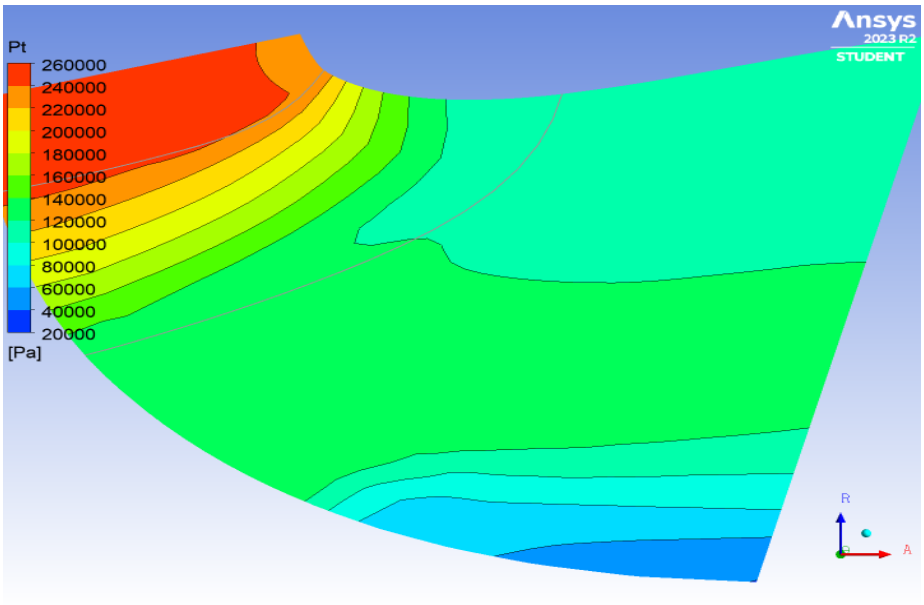

Fig 3.5.2.6: Contour of total pressure on meridional surface view

CHAPTER FOUR RESULT AND ANALYSIS

From the CFX analysis we found out different flow parameters. The results thus obtained is tabulated below.

Quantity	Inlet	LE Cut	TE Cut	Outlet	TE/LE	TE-LE	Units
Density	997.0000	997.0000	997.0000	997.0000	1.0000	0.0000	[kg m^-3]
Pstatic	294677.0000	271266.0000	205384.0000	202550.0000	0.7571	-65881.3000	[Pa]
Ptotal	347896.0000	347122.0000	224373.0000	221450.0000	0.6464	-122749.0000	[Pa]
Ptotal (rot)	192378.0000	192529.0000	188786.0000	186316.0000	0.9806	-3742.5500	[Pa]
U	16.0844	13.6388	10.7768	11.3805	0.7902	-2.8621	[m s^-1]
Cm	3.5646	3.9887	4.5580	3.6188	1.1427	0.5693	[m s^-1]
Cu	-9.6980	-11.3539	-4.2094	-3.4720	0.3707	7.1445	[m s^-1]
C	10.3324	12.1289	6.3891	5.1199	0.5268	-5.7398	[m s^-1]
Distortion Parameter	1.0000	1.0727	1.1923	1.4158	1.1116	0.1197	
Flow Angle: Alpha	70.1942	66.9426	40.5171	33.1535	0.6053	-26.4255	[degree]
Wu	6.3865	2.2850	6.5676	7.9084	2.8742	4.2826	[m s^-1]
W	7.3199	5.0238	8.3790	9.8059	1.6679	3.3552	[m s^-1]
Flow Angle: Beta	-61.1540	-31.4789	-53.6926	-63.1810	1.7057	-22.2137	[degree]

Fig 4.1 Flow parameters

From this data we calculated the degree of reaction as 0.654 which lies in the range of the fast runner category.

$$\text{Degree of reaction}(R)=1-(C1^2-C2^2)/(2*g*H_e)= 0.654$$

The efficiency with reduced head from inlet to outlet, shaft power is also presented below in the tabular form.

Rotation Speed	-157.0800	[radian s^-1]
Reference Diameter	0.1372	[m]
Volume Flow Rate	0.1201	[m^3 s^-1]
Head (LE-TE)	12.5545	[m]
Head (IN-OUT)	12.9327	[m]
Flow Coefficient	0.2960	
Head Coefficient (IN-OUT)	0.2730	
Shaft Power	14111.1000	[W]
Power Coefficient	0.0751	
Total Efficiency (IN-OUT) %	92.8958	

Fig 4.2 Output data

From these data set we were able to determine the optimum degree of reaction for the given input conditions with the optimum overall efficiency.

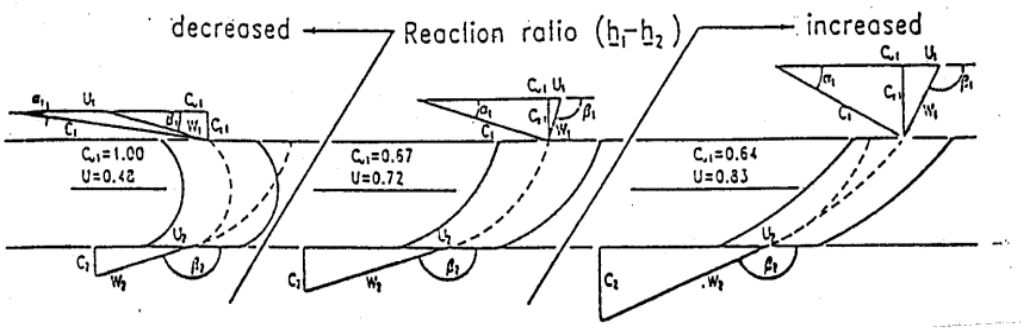

Fig 4.3 Blade cascades and velocity vector diagrams for different reaction ratios for cu2=0, and ηh = u1cu1/(gH) = const.

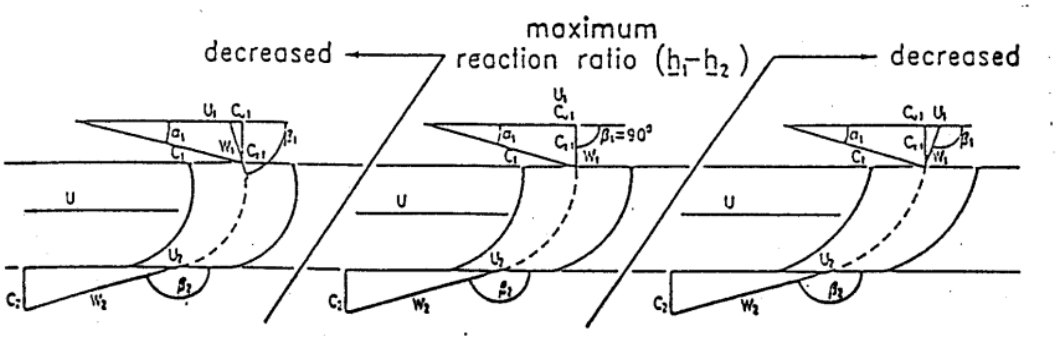

Fig 4.4: Influence from blade inlet angle if u1 = const. and cu2 = 0.

(HYDRAULIC TURBINE, Hermod Brekke)

It is very clear from fig 3.7 that degree of rection is optimum for beta angle =90 degree at inlet when u1 is constant and discharge is radial and from fig 3.6 that degree of reaction is optimum when the ratio u1/cu1 is maximum. Optimum degree of reaction for fast runners has to be between 0.55 to 0.65 and from the simulations , it was found to be 0.654.

Recommendations

- Degree of reaction could be further optimised by running simulation several times for different beta angle assumptions at outlet.
- Meshing in Turbogrid can be made better by refining the mesh at boundaries and edges.
- Whole turbine can be simulated instead of runner passage only.
- Effects of guide vanes and stay vanes can be considered to better approximate degree of reaction.

REFERENCES

Bovet T. Contribution to the study of Francis –Turbine Runner Design. Lausanne; 1963

K. Daneshkah and M. Zangeneh, "Parametric design of a Francis turbine runner by means of a three-dimensional inverse design method," in 25th IAHR Symposium on Hydraulic Machinery and Systems, Timisoara, Romania, 2010, pp. 1-5.

Kocak, E., Karaaslan, S., Yucel, N., Arundas¸, F.(2017). "*A Numerical Case Study: Bovet Approach toDesign a Francis Turbine Runner."* Energy Procedia,111, 885-894.

Biswakarma I B and Shrestha R 2017 "*Mathematical Modeling for the Design of Francis Runner"* Proceedings of IOE Graduate Conference vol 5.

Archana Lamsal, Sacchita Tiwari, Sunita Pokharel, and Susmita Jha. "*Design and modelling francis turbine using bovet method."* Master's thesis, Pulchowk Campus, Institute of Engineering, Tribhuvan University, 2016.

Prasanna Koiral, Samundra Karki, Srijan Satyal, Krishna Prasad Rijal. "*Design and Optimization of Bovet Based Francis Runner Using Meta-Model and Genetic Algorithm."*

"Comparative CFD analysis of Kali-Gandaki "A" Francis runner with runner generated from Bovet method." S. Karki , S. Satyal , KP Rijal, P. Koirala and N. Adhikari.

"A methodology for designing Francis runner blade to find minimum sediment erosion using CFD." Krishna Khanal , Hari P. Neopane, Shikhar Rai, Manoj Thapa, Subendu Bhatt, Rajendra Shrestha.

"Design and simulation of Francis runner", S. Karki, S. Satyal, K. Risal.

"Design and Optimization of Bovet Based Francis Runner Using Meta-Model and Genetic Algorithm" P. Koirala, S. Karki, S. Satyal, K. Risal.

"Design and Simulation of Francis Turbine Runner for Betan Karnali Hydroelectric Project for Original and Reduced Head Condition" Bikki Chhantyal, Sulav Parajuli Smarika Tamrakar Hari Bahadur Dura.

"*HYDRAULIC TURBINES Design, Erection and Operation."* Hermod Brekke, 2001.

Appendix A: Code for Francis runner design

```
%%%%%%% main.m %%%%%%%%%%

 clc

clear

Vo=0.12; % in m^3/s

H=15; % in m

nrpm=1500; %in rpm

nr=2*pi*nrpm/60; % in rad per second

N=3;

[no,R2e,Bo,Roi,Ymi,Rli,Roe,Li,Le,X2e,Y2e,Yme,Rme]=meridoinaldim (Vo,H,nr,N);

[x1,y1,x2,y2,x,y,LE1,LE2,TE1,TE2,xl,yl,xt,yt]=meri_lines(Vo,H,nr,N);

[lead_zr trail_zr hub_1_zr hub_2_zr hub_3_zr shroud_1_zr shroud_2_zr shroud_3_zr] = bladegen(x1,y1,x2,y2,xl,yl,xt,yt);

lead_zr=[lead_zr(:,2) lead_zr(:,1)];

 trail_zr=[trail_zr(:,2) trail_zr(:,1)];

%%%%%%% meridoinaldim.m%%%%%%%

function [no,R2e,Bo,Roi,Ymi,Rli,Roe,Li,Le,X2e,Y2e,Yme,Rme] =meridoinaldim (Vo,H,nr,N)

%constant

g=9.81

v2eo=0.27

x2e=0.5

%dimensionless dimension of meridoinal plane

no=(nr*(Vo/pi)^(0.5))/((2*g*H)^(3/4))

bo=0.8*(2-no)*no

roi=0.7+0.16/(no+0.08)

ymi=roi

rli=0.493/(no^(2/3))

if no<0.275
```

```
roe=0.493/(no^(2/3))
else
roe=1.255-0.3*no
end
li=3.2+3.2*(2-no)*no
le=2.4-1.9*(2-no)*no
x2e=0.5 %suppose constant
y2e=roe-1
y2ebyyme=curve(x2e,le)
yme=y2e/y2ebyyme
rme=roe-yme
% Normalizing dimensions
R2e=((Vo/pi)/(v2eo*nr))^(1/3);
Bo=bo*R2e
Roi=roi*R2e
Ymi=ymi*R2e
Rli=rli*R2e
Roe=roe*R2e
Li=li*R2e
Le=le*R2e
X2e=x2e*R2e
Y2e=y2e*R2e
Yme=yme*R2e
Rme=rme*R2e
%%%%%InterX%%%%%%%%
function P = InterX(L1,varargin)
%INTERX Intersection of curves
% P = INTERX(L1,L2) returns the intersection points of two curves L1
```

```
% and L2. The curves L1,L2 can be either closed or open and are described
% by two-row-matrices, where each row contains its x- and y- coordinates.
% The intersection of groups of curves (e.g. contour lines, multiply
% connected regions etc) can also be computed by separating them with a
% column of NaNs as for example
%
% L = [x11 x12 x13 ... NaN x21 x22 x23 ...;
% y11 y12 y13 ... NaN y21 y22 y23 ...]
%
% P has the same structure as L1 and L2, and its rows correspond to the
% x- and y- coordinates of the intersection points of L1 and L2. If no
% intersections are found, the returned P is empty.
%
% P = INTERX(L1) returns the self-intersection points of L1. To keep
% the code simple, the points at which the curve is tangent to itself are
% not included. P = INTERX(L1,L1) returns all the points of the curve
% together with any self-intersection points.
%
% Example:
% t = linspace(0,2*pi);
% r1 = sin(4*t)+2; x1 = r1.*cos(t); y1 = r1.*sin(t);
% r2 = sin(8*t)+2; x2 = r2.*cos(t); y2 = r2.*sin(t);
% P = InterX([x1;y1],[x2;y2]);
% plot(x1,y1,x2,y2,P(1,:),P(2,:),'ro')
% Author : NS
% Version: 3.0, 21 Sept. 2010
% Two words about the algorithm: Most of the code is self-explanatory.
% The only trick lies in the calculation of C1 and C2. To be brief, this
```

```
% is essentially the two-dimensional analog of the condition that needs
% to be satisfied by a function F(x) that has a zero in the interval
% [a,b], namely
% F(a)*F(b) <= 0
% C1 and C2 exactly do this for each segment of curves 1 and 2
% respectively. If this condition is satisfied simultaneously for two
% segments then we know that they will cross at some point.
% Each factor of the 'C' arrays is essentially a matrix containing
% the numerators of the signed distances between points of one curve
% and line segments of the other.
%...Argument checks and assignment of L2
error(nargchk(1,2,nargin));
if nargin == 1
L2 = L1; hF = @lt; %...Avoid the inclusion of common points
else
L2 = varargin{1}; hF = @le;
end
%...Preliminary stuff
x1 = L1(1,:)'; x2 = L2(1,:);
y1 = L1(2,:)'; y2 = L2(2,:);
dx1 = diff(x1); dy1 = diff(y1);
dx2 = diff(x2); dy2 = diff(y2);
%...Determine 'signed distances'
S1 = dx1.*y1(1:end-1) - dy1.*x1(1:end-1);
S2 = dx2.*y2(1:end-1) - dy2.*x2(1:end-1);
C1 = feval(hF,D(bsxfun(@times,dx1,y2)-bsxfun(@times,dy1,x2),S1),0);
C2 = feval(hF,D((bsxfun(@times,y1,dx2)-bsxfun(@times,x1,dy2))',S2'),0)';
%...Obtain the segments where an intersection is expected
```

```
[i,j] = find(C1 & C2);
if isempty(i),P = zeros(2,0);return; end;
%...Transpose and prepare for output
i=i'; dx2=dx2'; dy2=dy2'; S2 = S2';
L = dy2(j).*dx1(i) - dy1(i).*dx2(j);
i = i(L~=0); j=j(L~=0); L=L(L~=0); %...Avoid divisions by 0
%...Solve system of eqs to get the common points
P = unique([dx2(j).*S1(i) - dx1(i).*S2(j), ...
dy2(j).*S1(i) - dy1(i).*S2(j)]./[L L],'rows')';
function u = D(x,y)
u = bsxfun(@minus,x(:,1:end-1),y).*bsxfun(@minus,x(:,2:end),y);
end
end
%%%%%meri_lines.m%%%%%
function [x1,y1,x2,y2,x,y,LE1,LE2,TE1,TE2,xl,yl,xt,yt]=meri_lines(Vo,H,nr,N)
[no,R2e,Bo,Roi,Ymi,Rli,Roe,Li,Le,X2e,Y2e,Yme,Rme]= meridoinaldim (Vo,H,nr,N);
%for hub curve
n=50
x1=nan(1,n);
x2=nan(1,n);
y1=nan(1,n);
y2=nan(1,n);
p=Li/4;
q=p/n;
b=0;
for i=1:n
x1(1,i)=b;
y1=Ymi.*curve(x1,Li);
```

```
b=b+q;
end
%transforming hub curve
x1=-x1;
y1=-y1;
x_trans=Le+Bo;
y_trans=Roi;
for i=1:n
x1(1,i)=x1(1,i)+x_trans;
y1(1,i)=y1(1,i)+y_trans;
end
figure(1)
plot(y1,x1)
hold on
grid on
%%% shroud curve
p=Le;
q=p/n;
b=0;
for i=1:n
x2(1,i)=b;
y2=Yme.*curve(x2,Le);
b=b+q;
end
%transforming shroud curve
x2=-x2;
y2=-y2;
x_tran=Le
```

```
y_tran=Roe
for i=1:n
x2(1,i)=x2(1,i)+x_tran;
y2(1,i)=y2(1,i)+y_tran;
end
plot(y2,x2)
% meridional view with streamline
x = nan(N,n);
y = nan(N,n);
for i=1:N
x(i,:) = x1 + (x2-x1).*i/(N+1);
y(i,:) = y1 + (y2-y1).*i/(N+1);
end
for i=1:N
plot(y(i,:),x(i,:),'g')
end
title('Meridoinal plane')
%%%% Leading and trailing edge point
%%intersection of leading and trailing edge in hub and shroud curve respectively
x_le1=0:0.001:(Le+Bo);
y_le1=x_le1*0+Rli;
LE1=InterX([y_le1;x_le1],[y1;x1])
plot(LE1(1,1),LE1(2,1),'*')
x_le2=0:0.001:Le+Bo;
y_le2=0*x_le2+R2e*r1e(no);
Le2=InterX([y_le2;x_le2],[y2;x2]);
[pnt,idx]=max(Le2(2,:));
LE2=Le2(:,idx)
```

```
plot(LE2(1,1),LE2(2,1),'*')
x_te1=0:0.001:Le+Bo;
y_te1=0*x_te1+R2e*r2i(no);
TE1=InterX([y_te1;x_te1],[y1;x1])
plot(TE1(1,1),TE1(2,1),'*')
x_te2=0:0.001:Le+Bo;
y_te2=0*x_te2+R2e;
Te2=InterX([y_te2;x_te2],[y2;x2]);
[pnt,idx]=min(Te2(2,:));
TE2=Te2(:,idx)
plot(TE2(1,1),TE2(2,1),'*')
%%%Leading and trailing edge curve
xl=nan(1,n);
yl=nan(1,n);
xt=nan(1,n);
yt=nan(1,n);
%Leading edge
a=(LE1(2,1)-LE2(2,1))/((LE1(1,1)-LE2(1,1))^2);
p=LE2(1,1)-LE1(1,1);
q=(p)/(n-1);
r=LE1(1,1);
for i=1:n
xl(1,i)=r;
yl(1,i)=a*(xl(1,i)-LE2(1,1))^2+LE2(2,1);
r=r+q;
end
plot(xl,yl)
%trailing edge
```

```
a=(TE1(2,1)-TE2(2,1))/((TE1(1,1)-TE2(1,1))^2);

p=abs(TE2(1,1)-TE1(1,1))

q=p/(n-1)

r=TE1(1,1);

for i=1:n

xt(1,i)=r;

yt(1,i)=a*(xt(1,i)-TE2(1,1))^2+TE2(2,1);

r=r+q;

end

plot(xt,yt)

xlabel('r');

ylabel('z');

hold off

%%%%% curve.m%%%%%

function y= curve(x , l) x=x./l;

 y=3.08.*(1-x).^(3/2).*(x).^(0.5);

 End

 %%%%%%r1e.m%%%%%

function r1e= r1e(no) r1e=-125.6*no^5 + 299.4*no^4 - 278.0*no^3 + 126.3*no^2 - 28.52*no + 3.674;

 End

 %%%%%% r2i%%%%%%

function r2i = r2i(no) r2i= -25.824*no^5 + 61.275*no^4 - 56.035 *no^3 + 25.096*no^2 - 5.8833*no + 1.0977;

 End

%%%%%bladegen.m%%%%%

function [lead_zr trail_zr hub_1_zr hub_2_zr hub_3_zr shroud_1_zr shroud_2_zr shroud_3_zr] = bladegen(x1,y1,x2,y2,xl,yl,xt,yt)

tmp_m=zeros(numel(x1),1);
```

```
tmp_m=tmp_m+yl(1,1);
hub_zr=[y1' (tmp_m-x1')];
shroud_zr=[y2' (tmp_m-x2')];
lead_zr=[xl' (tmp_m-yl')];
trail_zr=[xt' (tmp_m-yt')];
%hub curve in three pieces
[a b]=min((hub_zr(:,1)-lead_zr(1,1)).^2); %finding nearest point in hub and le
[p q ]=min((hub_zr(:,1)-trail_zr(1,1)).^2);%finding nearest point in hub and te
hub_zr(b,:)=lead_zr(1,:); %replacing hub point with le point at b
hub_zr(q,:)=trail_zr(1,:); %replacing hub point with te point at y
hub_1_zr= hub_zr(1:b,:); %definig hub into 3 curves
hub_2_zr=hub_zr(b:q,:);
hub_3_zr=hub_zr(q:end,:);
%shroud curve in three pieces
[a b] =min((shroud_zr(:,1)-lead_zr(end,1)).^2+(shroud_zr(:,2)-lead_zr(end,2)).^2)
%finding nearest point in shroud and le
[p q] =min((shroud_zr(:,1)-trail_zr(end,1)).^2+(shroud_zr(:,2)- trail_zr(end,2)).^2)
%finding nearest point in shroud and te
shroud_zr(b,:)=lead_zr(end,:); %replacing shroud point with le point at b
shroud_zr(q,:)=trail_zr(end,:); %replacing shroud point with te point at y
shroud_1_zr= shroud_zr(1:b,:); %definig shroud into 3 curves
shroud_2_zr=shroud_zr(b:q,:);
shroud_3_zr=shroud_zr(q:end,:);
hub_1_zr=[hub_1_zr(:,2) hub_1_zr(:,1)]; %definig hub into 3 curves
hub_2_zr=[hub_2_zr(:,2) hub_2_zr(:,1)];
hub_3_zr=[hub_3_zr(:,2) hub_3_zr(:,1)];
shroud_1_zr=[shroud_1_zr(:,2) shroud_1_zr(:,1)];
%definig shroud into 3 curves
```

```
shroud_2_zr=[shroud_2_zr(:,2) shroud_2_zr(:,1)];
shroud_3_zr=[shroud_3_zr(:,2) shroud_3_zr(:,1)];
```

***