

Jan-Karl Lasse Escher

Design of a reversible pump turbine for retrofitting at an existing hydropower plant

June 2020



Norwegian University of
Science and Technology

Design of a reversible pump turbine for retrofitting at an existing hydropower plant

Jan-Karl Lasse Escher

Energy and environment

Submission date: June 2020

Supervisor: Pål-Tore Selbo Storli

Co-supervisor: Helene Njølstad Dagsvik

Norwegian University of Science and Technology
Department of Energy and Process Engineering

First of all, I would like to thank my supervisor Pål Tore Storli and my co-supervisor Helene Dagsvik for their continued support throughout the last two semesters, both academical and moral. I would also like to thank all of my friends for keeping me going. Especially the guys from NTNUI JuJitsu and from EMIL idrettskom deserve credit for making the past five years memorable. Last but definitely not least I am grateful to my family, both in Germany and in Norway for believing in me way more than I do.

Summary

In this thesis, an attempt is made to design a reversible pump-turbine for retrofitting in an already existing power plant. Due to the design of the current power plant, the cross-sectional area increases from inlet to outlet for a pump, which leads to separation. A script to design the pump-turbine is written.

The pump-turbine is tested in CFD software, but no reliable results are obtained. Error margins for the unreliable results are given and the results are discussed.

Finally, choices for a better design and more accurate simulations are suggested for further work with the project.

A paper is also written and submitted for the conference “CRHT-X”. The paper is added in appendix B.

Samandrag

I denne oppgåva vert det gjort ein freistnad på å formgje ein reversibel pumpeturbin til innsetjing i eit allereie eksisterande kraftverk. På grunn av måten det det noværande kraftverket er forma utvidar det tverrsnittlege arealet seg frå innløp til utløp av pumpa, noko som førar til så kalla separasjon. Eit skript vert skrive for å designe pumpeturbinen.

Pumpeturbinen vert testa ved hjelp eit numerisk strømningsbereknings-program men ingen pålitelege resultat vert funne. Feilmargar for resultatata vert oppgjevne og fellestrekk i resultatata vert diskuterte.

Til slutt vert forbetningsforslag for ei forbetring av både pumpeturbinen og simulasjonane gjort for framtidig arbeid med prosjektet.

Eit paper vert òg skrive og levert inn til konferansen “CRHT-X”. Paperet er lagt ved i appendiks B.

CONTENTS

Summary	i
Samandrag	ii
Table of Contents	v
List of Tables	vii
List of Figures	x
Abbreviations	xi
Nomenclature	xi
Sub- and Superscripts	xiii
1 Introduction	1
1.1 Previous work	3
1.2 Scope of the thesis	3
2 Theory	5
2.1 Fundamental physics	5
2.1.1 Bernoulli's equation and different kinds of pressure	5
2.1.2 Energy equation	6
2.1.3 Euler's equation for turbomachines	8
2.1.4 Circular flow	8
2.2 Hydroelectric power plants	9
2.3 Pumps	11
2.4 Reversible pump turbines	12
2.5 The case: Roskrepp Powerplant	12
2.6 The runner	13
2.7 Velocity triangles	14
2.8 Slip	15

2.9	Power	16
2.10	Characteristics	17
2.11	Attached and separated flow	19
2.12	Recirculation	19
2.13	Cavitation	20
2.14	Differences between pumps and turbines	21
2.15	Design philosophies for pump turbines	22
2.16	Computational fluid dynamics	24
2.16.1	Convergence of steady state solutions	24
2.16.2	Transient solutions	25
2.16.3	Turbulence models	26
2.16.4	Wall models	26
2.16.5	Considerations for simulations of turbomachines	27
2.16.6	Verification and validation	27
3	Method and results	29
3.1	Geometry	29
3.1.1	Meridional view	29
3.1.2	Blade angles at the trailing edge	31
3.1.3	Blade angles at the Leading edge	32
3.1.4	Distribution of blade angles along the blade	32
3.1.5	Checking for cavitation	33
3.1.6	Blade thickness and 3D	33
3.2	Discussion of the design method	34
3.3	CFD setup	35
3.3.1	The computational domain and boundary conditions	35
3.3.2	The solver settings	36
3.3.3	The grid and the grid refinement process	37
3.4	CFD Results	40
3.4.1	Iterative convergence of the cases	40
3.4.2	Mesh refinement	41
3.4.3	Separation	42
3.4.4	Slip	44
3.4.5	Cavitation	46
3.4.6	Blade loading	46
3.4.7	Recirculation	47
3.4.8	Regions with large residual values	49
3.4.9	Transient simulations	49
4	Conclusion	51
4.1	Further work	51
	Bibliography	52
	Appendices	57

A	Description of the masters work	59
B	Paper for CRHT-X	63
C	Distribution of streamlines in the meridional plane	73
D	G and H curves	75
E	Handling of the loss models for the spiral casing and the stay- and guide vanes	79
F	Failed attempts to reach convergence	85
G	Matlab Scripts	87

LIST OF TABLES

2.1	Empirical data for factors related to the values, collected from [5, page 46]	21
3.1	Mesh reeinement factors and corresponding cell counts	39
3.2	Results with mesh refinement	42
3.3	Slip factor for different cases	45

LIST OF FIGURES

1.1	Amount of renewable energy sources in Europe, collected from [29] . . .	1
1.2	Amount of renewable energy sources in Norway, collected from [29] . . .	2
2.1	The energy balance for a system including a pump and a turbine. Collected from [6, page 208]	7
2.2	Types of circular flow.	9
2.3	Schematic view of a hydro power plant.	10
2.4	The three main turbine types	10
2.5	Top down schematic view of spiral casing, stay- and guide vanes. Note that the figure is not correctly proportioned and only used for illustration purposes	11
2.6	Radial, mixed flow and axial pumps, collected from [40]	12
2.7	The current arrangement with changeable area indicated.	14
2.8	Velocity triangle at the pump outlet	15
2.9	Illustration of the slip and its effects on the velocity triangle at the outlet of a pump	16
2.10	Sketch of characteristics for pumps with different blade outlet angles . . .	18
2.11	The streamlines and the velocity distributions in a separating flow.	19
2.12	Collapse of a cavitation bubble.	20
2.13	Disturbance of the flow with different leading edge shapes	22
2.14	The velocity triangles at 1 for a pump and a turbine at same Q and H_E . .	22
2.15	Variation of head for fixed change in volume flow rate in idealized pump characteristics with different slopes.	23
2.16	Dimensionless UC_u curve along dimensionless streamwise location . . .	24
3.1	Meridional view of the runner with control points, streamlines and blades indicated	30
3.2	Predicted Characteristic of the blade	31
3.3	The inlet of the pump at design conditions with irrotational inlet flow. . .	32
3.4	Distribution of the uc_u values along the blade	33
3.5	Figure of the blade	34
3.6	Meridional view of the entire domain.	37

3.7	Mesh refinement close to the wall.	38
3.8	The significant amount of skewness of the mesh in the outlet domain. . .	39
3.9	The monitored values of the head factor and the residuals for a simulation with $q^* = 1$ and mesh refinement factor of 1.7	40
3.10	The characteristic and overall efficiency for different mesh refinement factors	41
3.11	Span=0.1	43
3.12	Span=0.5	43
3.13	The blade outlet angles and the exiting flow angles for the case $q^* = 1$, Mesh refinement factor=1.2.	44
3.14	Cavitation on the blade at $q^* = 1$,	45
3.15	The static pressure distribution along the blade surface	46
3.16	The static pressure distribution along the blade surface	47
3.17	The location at the inlet shroud where recirculation occurred.	48
3.18	Separation at $q^* = 1$	48
3.19	Separation at $q^* = 0.7$	49
3.20	The largest residual values in the inlet and the passage.	50
C.1	Details of the iterative process to find points on the next streamline	74
D.1	G and H lines on one streamline rotated around the axis.	76
D.2	Relation between θ , R and H	77
E.1	Simplified geometry of guide vanes with rotation	80

NOMENCLATURE AND ABBREVIATIONS

List of Abbreviations

BEP	Best Efficiency Point
CFD	Computational Fluid Dynamics
NPSH	Net Positive Suction Head
RMS	Root Mean Square
RPT	Reversible Pump Turbine
SST	Shear Stress Transport

Nomenclature

α	Kinetic energy correction factor
β	Angle between runner velocity and relative flow velocity
β_B	Blade angle
Δt	Timestep
Δx	Element length
ϵ	Turbulent dissipation
η_h	Hydraulic efficiency
η_m	Mechanical efficiency
η_o	Overall efficiency

γ	Slip factor
ν_1, ν_2	Vibration orders
ω	Rotational velocity
ω	Specific dissipation
Ψ	Head coefficient
ρ	Density
τ	Blade blockage factor
A	Area
c	Velocity of the flow
c_u	Circumferential fluid velocity
Co	Courant number
d	Runner diameter
e	Blade Thickness
g	Gravitational acceleration
H	Head
H_E	Effective head
H_g	Gross head
h_L	Friction head loss
k	Kinetic turbulent energy
M	Moment, Torque
P	Power
P_d	Dynamic pressure
P_s	Static pressure

P_{head}	Static pressure head
P_{tot}	Total pressure
Q	Volumetric flow rate
R	Radius
u	Runner velocity
V	Velocity
w	Velocity of the flow relative to the runner
z	Vertical location
z_{la}	Number of runner blades
z_{le}	Number of guide vanes

List of Sub- and Superscripts

'	Component with slip
θ	Circumferential component
g	Gross
in	Property entering control volume
m	Meridional component
out	Property leaving control volume
p	Pump mode
r	Radial component
t	Turbine mode
x	Axial component
A	Available
R	Required

INTRODUCTION

In recent years, Europe as a whole has started to rely more and more heavily on renewable energy sources like wind and solar power, as shown in figure 1.1. The drawback

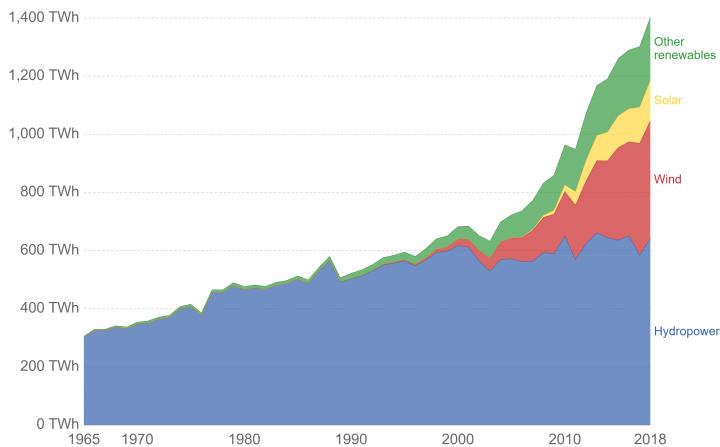


Figure 1.1: Amount of renewable energy sources in Europe, collected from [29]

with these sources is that they are directly influenced by the weather, and thus cannot be used on-demand. This necessitates good techniques for energy storage. One of the most economical technology for storing energy is the application of pumped storage technology[15], where excess energy is taken from the grid and used to pump water to a higher reservoir. When there is a demand for more energy in the grid the same water is run through a turbine to regenerate the energy. However, to be able to use pumped storage technology, one needs a reservoir pair and a height difference.

Norway has very good conditions for the application of pumped storage technologies, as it has many mountain lakes with short aerial distance and large vertical distance to other lakes or the sea. The country already has approximately 43% of Europes hydropower

reserves¹ and has utilized water as its primary energy source for a long time, as shown in figure 1.2. Therefore most of the available hydropower resources are either already

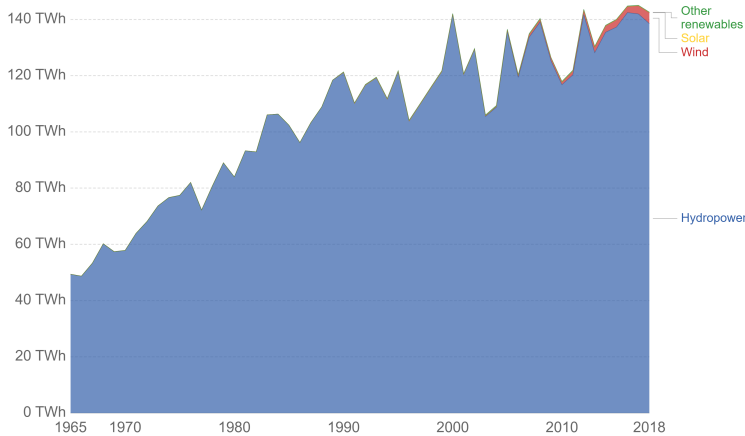


Figure 1.2: Amount of renewable energy sources in Norway, collected from [29]

utilized as regular hydropower plants or are protected due to environmental concerns². Furthermore, a significant portion of the remaining energy potential is not suitable for regulatory operation, and thus would be weather dependent as well [13, fig 3.3]. There are only 9 powerplants with pumping capability in Norway [25]. This means that there is room for more pumped storage powerplants in the Norwegian grid [16].

One idea to make new pumped storage hydropower plants at low costs is to reuse existing powerplants and retrofit them with reversible pump turbines which are to function both as pumps and as turbines [37].

There are some problems connected to this approach. One is that pumps need larger submergence than turbines to keep the fluid from evaporating due to low-pressure zones, so-called cavitation, as will be shown later. A possible solution for this may be to insert a booster pump downstream of the pump-turbine. Booster pump technology is currently under development in a project parallel to this one.

The other problem is that a Francis turbine and a pump with the same dimensions won't satisfy the same criteria related to pressure and flow rate.

In this thesis, an attempt will be made to design a reversible pump-turbine which fits the geometry and the pressure requirements of an already existing power plant.

The exact wording of the task description is added in appendix A.

¹Reservoir Capacity in Europe \approx 200 TWh [1, page 17].

Reservoir capacity in Norway \approx 86.9 TWh [26].

$86.9 \text{ TWh} / 200 \text{ TWh} \approx 0.43$

²Although estimates of the remaining potential vary [19]

1.1 Previous work

This project is part of an ongoing series of projects, all aiming at fitting a new pump turbine in the geometry of an already existing turbine and to facilitate this so-called retrofitting by adding an additional pump on the downstream side of the runner to counteract cavitation³. Earlier investigations have shown that a well-designed booster pump and a pump-turbine in series should be able to cooperate without problems if inserted into Roskrepp Power Plant [36]. Pål Dahle has made a program to design booster pumps and concludes with an initial estimate of 6.2 m of lifting height for the pump. Rune Larsen [18] simulated a reversible pump turbine and concluded with counterrotating inlet velocity being beneficial for the head, but not for the efficiency. Fernando Perán and Ademir Suárez have performed an economic study on the feasibility of retrofitting existing powerplants with reversible power plants [27]. They conclude that, in order for retrofitting to become feasible, research must be done on reducing the cavitation requirements for the reversible pump turbines and on bringing the pump discharge closer to the original turbine discharge.

To the author's knowledge, no attempt has been made to retrofit a reversible pump turbine without altering the external geometry of the runner.

1.2 Scope of the thesis

Ideally, the thesis would look at the entire hydraulic system, evaluate the structure of the runner and have a look at the economical benefits and disadvantages of the project, as well as comparing multiple different designs to find the best possible one. Unfortunately doing so would by far exceed the resources available to the author.

Therefore the scope of this thesis will be to design a reversible pump-turbine which should fit the geometry of a conventional Francis turbine and to test whether it can deliver sufficient pressure to meet the system requirements. The work will only evaluate the fluid dynamic part of the design, and disregard structural aspects. The testing will be performed by use of numerical simulations and the numerical results will be evaluated

³Cavitation is when the fluid in a turbine evaporates due to low pressure. More about this will be explained in the theory chapter.

■ This chapter will focus on the theory involved in the design process of a reversible pump-turbine. First, an overview of the underlying physics will be given, then the relevant components of a hydropower plant will be explained. Following this, the physical phenomena related specifically to pumps and turbines will be explored. Finally, a brief introduction to the ideas and processes of Computational Fluid Dynamics will be given.

2.1 Fundamental physics

To be able to understand the choices made in this thesis, it is important to have an understanding of some fundamental underlying physical principles related to fluid flow. This section will give a brief overview of these.

2.1.1 Bernoulli's equation and different kinds of pressure

Bernoulli's equation is a form of energy conservation equation. It concerns itself with the conservation of kinetic-, potential- and flow energy. It is derived in Çengel and Cimbala [6, page 187] and if written in terms of pressure, it states that the total pressure, P_{tot} is constant along a streamline¹:

$$P_{tot} = P_s + \frac{\rho V^2}{2} + \rho g z = \text{constant (along a streamline)} \quad [\text{Pa}] \quad (2.1)$$

Where P_s is the static pressure at any point along the streamline, ρ is the density of the fluid, V is the velocity of the fluid, g is the gravitational acceleration, z is the vertical location of the point which is evaluated.

¹A streamline is a line which is everywhere tangent to the velocity field[6, page 129]

The individual terms of equation (2.1) deserve a more thorough explanation, as they are important to the performance of turbomachines and will be used frequently in later chapters.

Pressure is defined as the force acting per area and thus is responsible for many of the phenomena occurring in a flow. The three terms in Bernoulli's equation are related to three different kinds of pressure.

The static pressure, P_s is the pressure acting on all sides of a fluid particle. It is this pressure which determines the chemical properties of a fluid, like its density and its phase.

The dynamic pressure, P_d is the kinetic energy of the fluid per fluid volume. It is given by:

$$P_d = \frac{1}{2}\rho V^2 \quad [\text{Pa}] \quad (2.2)$$

The static pressure head, P_{head} is the potential pressure relative to some datum elevation. It is given by

$$P_{head} = \rho g z \quad [\text{Pa}] \quad (2.3)$$

and must always be viewed relative to the height.

The sum of the above pressures is the total pressure, P_{tot} . The total pressure is what is constant along a streamline according to Bernoulli's equation. In flow with no significant change in elevation, the total pressure is also called the stagnation² pressure, as it is the static pressure of a fluid which is brought to rest. One should be cautious of the fact that Bernoulli's equation has to meet a number of requirements to be valid: Bernoulli's equation is only valid for flows which are steady³, do not include energy transfer, are incompressible⁴ and are along a streamline.

2.1.2 Energy equation

The first law of thermodynamics states that energy is conserved [23, page 51]. This means that all changes in energy for a system must be accounted for. For flows in which there is a change from mechanical to thermal energy, or an in or outflux of energy to the system, the Bernoulli equation does no longer hold and a different principle is required. In such cases the energy equation is used as it includes terms to account for energy exchange between the flow and the surroundings. For a fluid flowing through a control volume⁵, this is conventionally expressed in the form of changes in the head, H . Head is a convenient way to express the mechanical energy of a system. It describes energy as the height of a fluid column with an equivalent potential energy, or in terms of total pressure:

$$H = \frac{P}{\rho g} \quad [\text{m}] \quad (2.4)$$

As will be explained in section 2.2, hydro power plants operate due to differences in water levels. The head gives an intuitive way of relating the energy to the water levels.

²A fluid with the same velocity as the reference system is called stagnated. Points with stagnated fluid in an otherwise moving fluid are called stagnation points

³Steady flows do not vary with time

⁴Incompressible fluids have constant density

⁵A control volume is a "black box" for a flow. What happens inside is ignored and all calculations are performed for forces acting on- and flow through the surface.

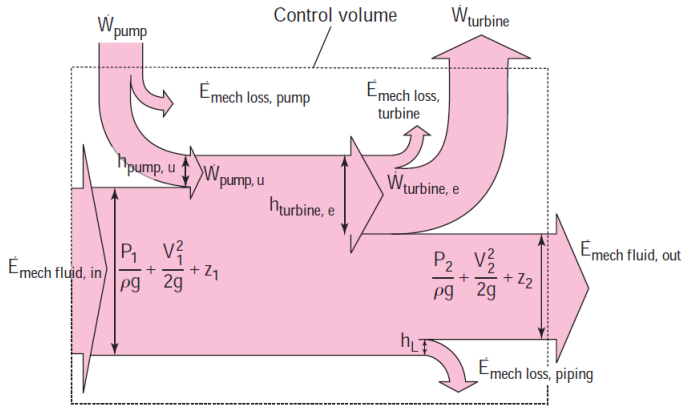


Figure 2.1: The energy balance for a system including a pump and a turbine. Collected from [6, page 208]

Most energy losses in fluid flows occur due to friction between the fluid and the surrounding surface, or by disturbances of the fluid. These losses transform mechanical energy to heat. Other than this, the increase or decrease of the mechanical energy will occur due to so called shaft work, which is work exerted by the fluid on a turbine, or by a pump on the fluid. An explanation of turbines and pumps will be given in sections 2.2 and 2.3 respectively. The equation for the energy balance for such a system expressed in terms of head is given by

$$\begin{aligned} \frac{P_{in}}{\rho_{in}g} + \alpha_{in} \frac{V_{in}^2}{2g} + z_{in} + h_p \\ = \frac{P_{out}}{\rho_{out}g} + \alpha_{out} \frac{V_{out}^2}{2g} + z_{out} + h_t + h_L \end{aligned} \quad [\text{m}] \quad (2.5)$$

Where *in* and *out* denote properties respectively entering and leaving the control volume, *P* is the static pressure and h_L is the irreversible head loss due to friction. The factor α is the kinetic energy correction factor which is necessary to correct for the velocity profile of the flow⁶. For flow through a pipe it is 2 if the flow is laminar and between 1.04 and 1.11 for turbulent⁷ flows. For most cases the influence of α is however insignificant, either because the values of kinetic energy are negligible or because the flow is turbulent and α thus comes close to unity. h_p and h_t are changes in the head due to interaction between the fluid and a pump or turbine, respectively. Note that the term h_p actually adds energy to the system as the pump exerts a work on the fluid. A diagram of the energy balance for a flow with a turbine and a pump is shown in figure 2.1. The schematic includes mechanical losses for pumps and turbines which will be explained in section 2.9.

⁶A velocity profile describes the velocity of the flow in direction perpendicular to the flow direction[7]

⁷Flows with lots of mixing are called turbulent. The opposite of a turbulent flow is a laminar flow where there is very little mixing. The velocity profile of a laminar flow is much more pointy than that of a turbulent flow [7].

2.1.3 Euler's equation for turbomachines

The power, P , transferred between a shaft and a flow is given by

$$P = M\omega \quad [\text{W}] \quad (2.6)$$

where M is the torque acting on the shaft and ω is the rotational velocity of the shaft [7, page 216]. A convenient way to calculate this power expressed by the flow around the shaft is by application of the equation for conservation of angular momentum [7, page 266]. For a turbomachine, it can be simplified by setting a cylindrical control volume around the turbine. Then the following assumptions are made, as shown in [6, pages 254-255]. Steady-state flow is assumed for the control volume, and it is observed that flow flowing parallel or perpendicular to the axis does not contribute to the torque on the shaft. When the result is combined with equation (2.6), the following equation emerges:

$$P = \rho Q(u_{out}c_{u,out} - u_{in}c_{u,in}) \quad [\text{W}] \quad (2.7)$$

Where u is the velocity of the runner, c_u is the circumferential fluid velocity and Q is the volumetric flow rate⁸. This equation is commonly known as Euler's turbine equation and holds for all turbomachines⁹.

2.1.4 Circular flow

Flows which have a rotational component around a centre of rotation are called circular flows. They are also commonly referred to as vortices. There are two main types of vortices, namely free and forced vortices [12, page 30]. The flow through a radial turbomachine usually involves both free and forced vortices at some point. Free vortices are vortices which occur purely due to initial rotation in the flow. Examples of free vortices are the swirl which occurs when a sink is drained, or the flow observed in a tornado. They are described by

$$c_u = \frac{K}{R} \quad [\text{m s}^{-1}] \quad (2.8)$$

Where R is the radial distance to the centre of the vortex and K is a constant defining the strength of the vortex. It should be noted that a real free vortex deviates from equation 2.8 close to the centre of rotation. Otherwise, the rotation would approach infinity there.

Forced vortices occur when the flow is driven by a rotating object, like a spoon in a cup of tea or the rotation of a turbine. Their velocity distributions take the same form as the velocity distribution of solid body rotation:

$$c_u = \omega R \quad [\text{m s}^{-1}] \quad (2.9)$$

The different vortices can be seen in figure 2.2.

⁸The volumetric flow rate describes the volume flowing in and out of a control volume. It originates from the continuity equation and is constant for incompressible fluids like water [7, page 192]

⁹Turbomachines are defined by Dixon & Hall [30, page 1] as "...all those devices in which energy is transferred either to, or from, a continuously flowing fluid by the dynamic action of one or more moving blade rows".

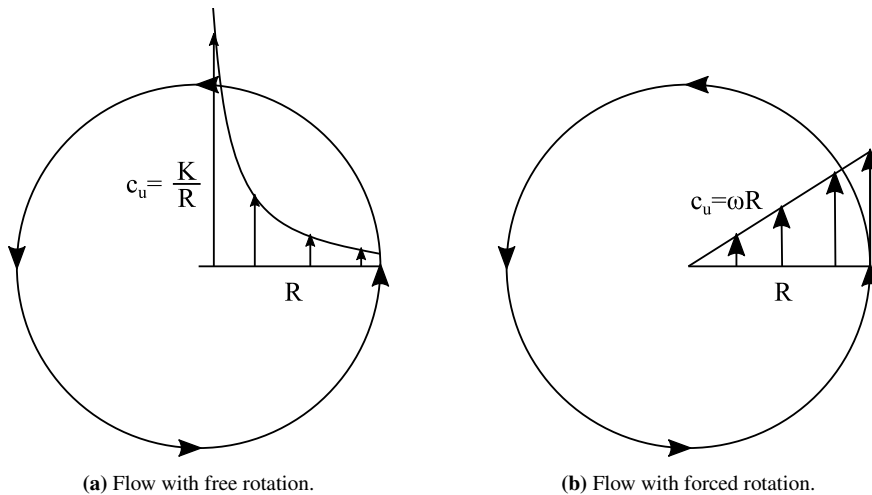


Figure 2.2: Types of circular flow.

2.2 Hydroelectric power plants

With this, the reader should have a broad enough basis of fluid flows to be able to understand the physics of turbomachines such as pumps and turbines, as well as the basics of hydropower plants. Hydropower plants deliver electricity to the electric grid by utilizing the energy contained in water moving between two reservoirs at different heights. The difference in elevation between the two reservoirs is called the gross head H_g . The water from the upper reservoir is led to the lower reservoir via a waterway called the penstock. At the lower end of the penstock, there is a turbine through which the energy from the water is transferred to a generator and then is delivered to the grid. This removed energy is the term h_t in the energy equation (2.5). A schematic view of a general hydro power plant is shown in figure 2.3. The numbers indicated on the figure correlate with the subscripts used in this thesis.

In the context of this thesis, the most interesting part of a hydropower plant is the turbine.

The three most common types of hydraulic turbines are Pelton, Francis and Kaplan turbines. The kind of turbine which should be used is decided by the relation between the head of the turbine and the volume flow rate at which it operates.

Pelton turbines are applied for cases with high head and low volume flow rate while Kaplan turbines are applied for cases with low head and high flow rates. Francis turbines are usually used for cases in between the two others. The three turbine types are shown in figure 2.4

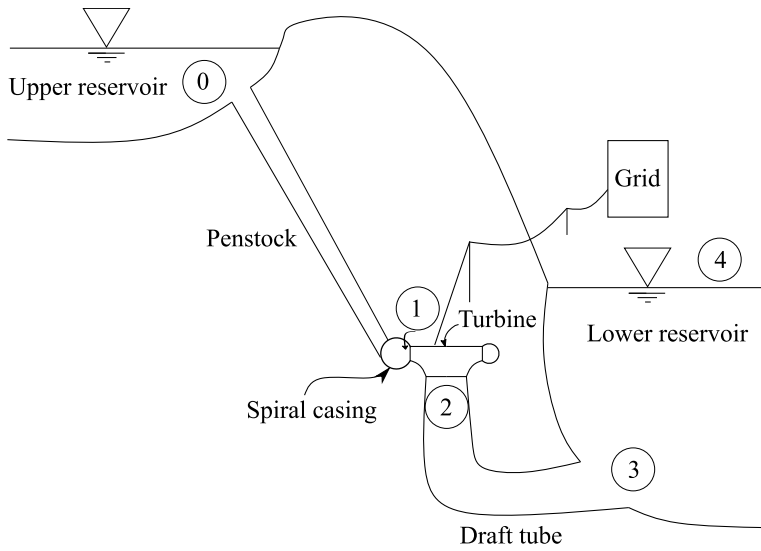


Figure 2.3: Schematic view of a hydro power plant.

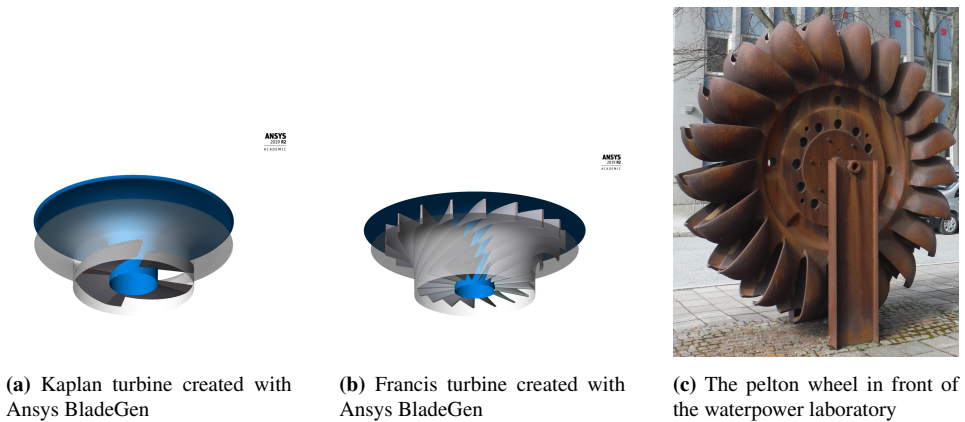


Figure 2.4: The three main turbine types

Both Francis and Kaplan turbines are reaction turbines, which means that only part of the drop in static pressure has occurred before the turbine, while the rest occurs in the runner itself. They are driven due to lift¹⁰ and impulse on the blades. The turbine type which exists in the hydropower plant used as a case for this thesis is a Francis turbine. Therefore the components commonly found in a Francis turbine will be explained in greater detail in the following.

When the water exits the penstock, it is first led into a spiral casing. The spiral casing

¹⁰Lift is the force working on a body perpendicular to the flow direction, due to pressure differences on the sides of the body[12]. It is the same principle which makes planes fly.

is a component which ensures that the water is guided into the turbine at a uniform volume flow rate. The outlet of the spiral casing is equipped with so-called stay vanes, which are streamlined bodies designed to keep the spiral casing from being deformed by the water pressure. After passing through the spiral casing and the stay vanes, the flow is directed into the turbine at an angle by use of guide vanes. Guide vanes are also streamlined bodies, but they are mounted on stems which can be used to rotate them to change the inlet angle of the flow, depending on the volume flow rate and the head. A schematic top-down view of the spiral casing with guide vanes and stay vanes is shown in figure 2.5

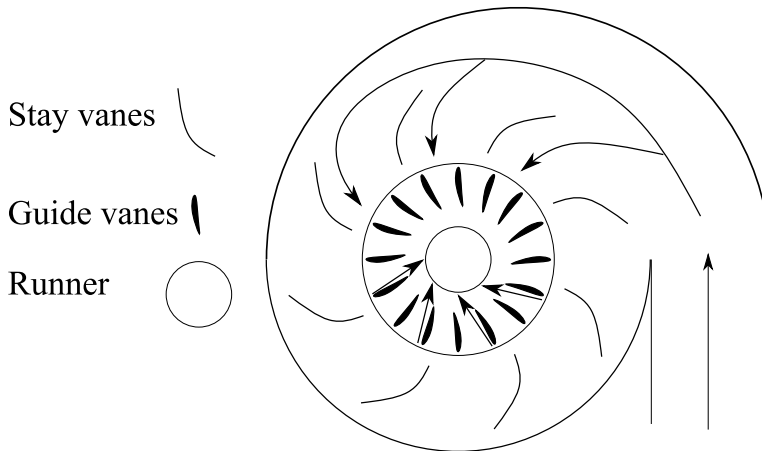


Figure 2.5: Top down schematic view of spiral casing, stay- and guide vanes. Note that the figure is not correctly proportioned and only used for illustration purposes

After passing through the runner, the flow is led into a draft tube. A draft tube is a pipe with cross-section expanding towards the lower reservoir. This expansion leads to a decrease in velocity and a corresponding increase in static pressure. This facilitates a recovery of the energy remaining in the flow which otherwise would be lost [5, page 75].

2.3 Pumps

The pump is another kind of turbomachine. It adds energy to a system, and thus serves to increase the pressure and lift a fluid upwards. The rotating part of a pump is called the impeller. As with turbines, there are different kinds of pumps. The two extrema being radial pumps and axial pumps. Axial pumps look like propellers and the flow both enters and leaves the pump in the axial direction. In such pumps, head is generated only by the lift of the pump blades. In radial pumps, on the other hand, the flow is purely radial so it enters the pump at one radius and leaves at a greater one. The head is created by centrifugal forces in the radial direction only.

Most pumps are however somewhere between the two extrema and produce head both due to lift created by the blades and due to centrifugal forces. These pumps are called mixed flow pumps, as they have flow in both the radial and the axial direction. Mixed flow

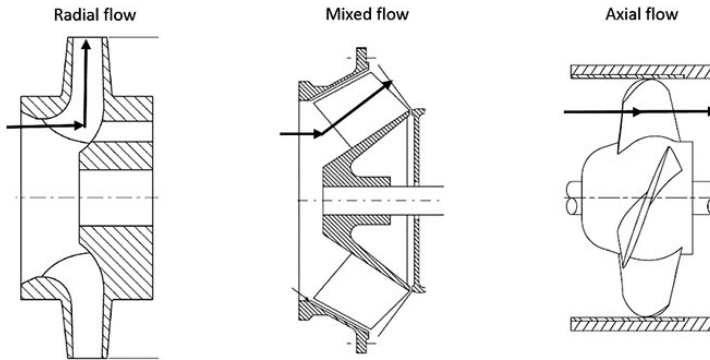


Figure 2.6: Radial, mixed flow and axial pumps, collected from [40]

pumps have a general shape which is quite similar to the shape of Francis turbines with one opening in the radial and one in the axial direction. The main difference is the rotational direction of the pump, which reverses the flow and adds energy instead of extracting it. These three kinds of pumps are illustrated in figure 2.6

2.4 Reversible pump turbines

Because the fundamental workings that allow radial, axial and mixed flow pumps to add energy to a flow are the same as the ones that allow Francis and Kaplan turbines to extract energy from a flow, a pump can be reversed to work as a turbine and vice versa. Unfortunately, a reversed pump does perform worse as a turbine than a pump, as will be shown in section 2.14. A turbomachine which works both as a pump and a turbine is called a reversible pump-turbine or RPT for short. As mentioned in the introduction, RPTs can be used to produce energy when there is a lack of energy in the grid and used as storage devices when there is an energy surplus. To do this, the upper and lower reservoir has to be sufficiently large so the turbomachine has a source to draw water from in both operational modes.

2.5 The case: Roskrepp Powerplant

The powerplant which will be used as a basis for the calculations in this thesis is “Roskrepp Powerplant”, owned by Sira-Kvina Kraftselskap[17]. Due to confidentiality concerns, all dimensions and values connected to the powerplant will be given in dimensionless form

as follows:

$$q^* = \frac{Q}{Q_{BEP}} \quad [-] \quad (2.10)$$

$$R^* = \frac{R}{R_{GV}} \quad [-] \quad (2.11)$$

$$c^* = \frac{c}{u_1} \quad [-] \quad (2.12)$$

$$\Psi = 2g \frac{H}{u_1^2} \quad [-] \quad (2.13)$$

where Q_{BEP} is the volume flow rate at the best efficiency¹¹ point (BEP), R_{GV} is the radius of the guide vane stems, and $u_{1,m}$ is the mean runner velocity at the high pressure side of the runner. The factor Ψ is the head coefficient, which is a common way to describe the dimensionless head of a turbomachine. The turbine currently installed in Roskrepp is a Francis turbine. It operates at a gross head coefficient of $\Psi_g = 0.71$ between the higher and the lower reservoir and at a maximum volumetric flow rate of $q^* = 1.3725$.

As mentioned in the introduction, the objective of this work will be focused on designing an RPT which should fit the same external geometry as the current runner installed in Roskrepp and at the same time produce enough head to satisfy the requirements of the site. This means that the mean dimensions of the runner will remain the same. A sketch of the runner is shown in figure 2.7. The area in which modifications can be made is indicated by the shaded area.

2.6 The runner

As an RPT is both a pump and a turbine, the rotating component will be referred to as the runner in the remainder of this document to avoid confusion.

The runner consists of a number of runner blades which serve to transfer the energy from the fluid to the blade. The channel between two blades, through which the water passes is referred to as the passage. The wall of the passage which is closest to the axis is called the hub and the other wall is the shroud. Figure 2.7 shows the components of a meridional¹² projection of a runner.

The number of blades in the runner depends on whether the runner is designed for a pump or a turbine. In either case one wants to avoid harmonic interference between the runner blades and the guide vanes. The runner in this project will be designed as a pump, as will be explained in chapter 2.15. For a pump it is recommended that the number of runner vanes is less than 8.

To avoid interference, Gülich [14, page 342] recommends to ensure that m in equation (2.14) fits the condition $m \neq \{0, 1\}$ for integer values of $\nu_{1,2}$ up to 3.

$$m = |\nu_2 \cdot z_{la} - \nu_1 \cdot z_{le}| \quad [-] \quad (2.14)$$

¹¹Efficiency is the rate of power input to useable power, described more extensively in section 2.9.

¹²A meridional projection of a turbine or a pump shows the cross section of the component, with all components projected onto the same angular plane.

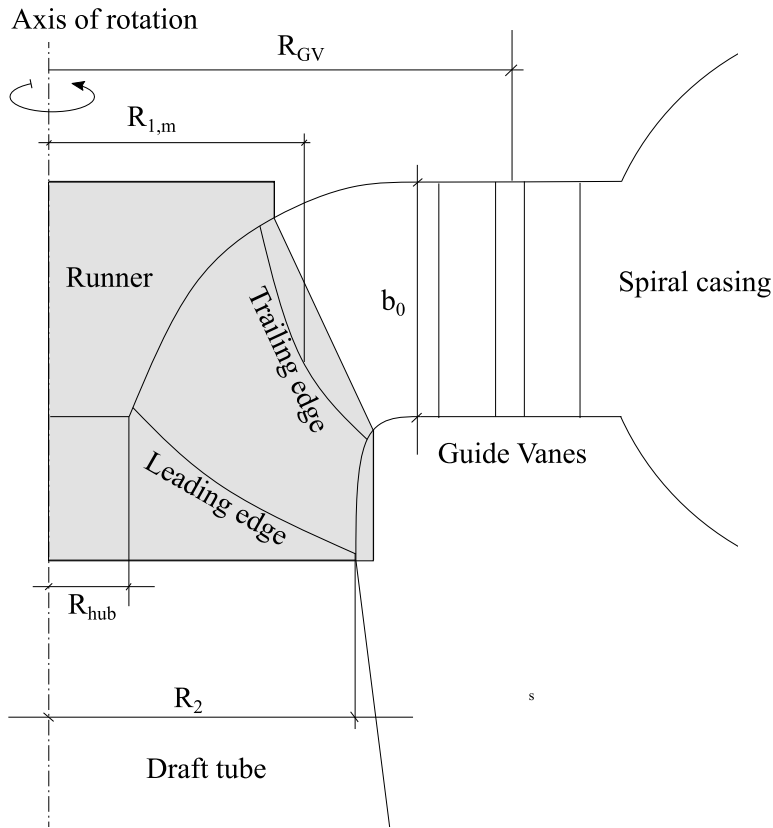


Figure 2.7: The current arrangement with changeable area indicated.

ν_1 and ν_2 are vibration orders z_{1a} is the number of runner blades and z_{1e} is the number of guide vanes.

2.7 Velocity triangles

A powerful tool when working with turbomachines is the velocity triangle. A velocity triangle shows the relationship between the velocity of the fluid and the velocity of the runner itself. The three main elements of a velocity triangle are the velocity of the flow, c , the velocity of the runner, u and the relative velocity of the flow to the runner, w . As such, w can be expressed as the difference between c and u . See figure 2.8 for a better understanding.

The c and w components of a velocity triangle can be divided into their radial, axial and circumferential components, denoted by subscripts r , x and θ , respectively. As the u component is purely circumferential, there is no need to further decompose it. Another commonly used component of the fluid velocity is the meridional velocity, subscript m .

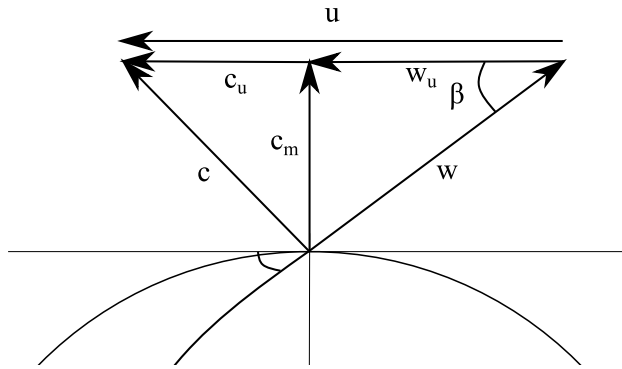


Figure 2.8: Velocity triangle at the pump outlet

The meridional velocity is the velocity in the axial and radial plane¹³, so

$$c_m = \sqrt{c_x^2 + c_r^2} \quad [\text{m s}^{-1}] \quad (2.15)$$

When one again uses a cylindrical control volume around the runner, it can be seen that the circumferential velocity components are everywhere tangential to the surface. Therefore the meridional velocity is the only velocity relevant for the volume flow rate:

$$Q = c_m \cdot A_m \quad [\text{m}^3 \text{s}^{-1}] \quad (2.16)$$

where A_m is the meridional area, so the area which c_m is normal to.

The angle between u and w is labeled β .

Ideally, the angles of the runner blades at inlet and outlet would be the same as the β angles of the flow to avoid incidence losses, which are losses originating because of disturbances in the flow caused by the blades. This is only possible for one operation point of a turbine and impossible for pumps, as will be explained in section 2.8.

2.8 Slip

Slip is a term which can have multiple meanings when referred to in a pump-context. One meaning is the behaviour of flow along a wall, where no-slip signifies that the flow close to a wall sticks to the wall due to intermolecular forces [41, page 8]. This is the most common meaning of slip in fluid mechanics.

The other meaning of slip is more pump specific. In a pump, slip is a phenomenon which occurs due to the pressure difference between the pressure side and the suction side at the outlet¹⁴. It leads to the actual outlet angle of the flow being different from the blade outlet angle as illustrated in figure 2.9.

¹³Also called the meridional plane

¹⁴All turbomachines create high pressure on one side of their blades and low pressure on the other side. These are referred to the pressure and suction side of the blade, respectively

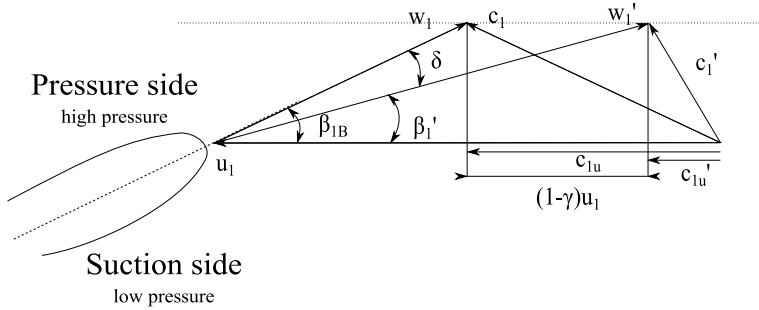


Figure 2.9: Illustration of the slip and its effects on the velocity triangle at the outlet of a pump

While slip has a significant effect in pumps, it is negligible for turbines because the pressure differences should be completely transferred to the runner at the outlet, in the form of energy.

The slip is calculated by use of the slip factor γ , which relates the theoretical and actual value of c_{u1} to u , as shown in figure 2.9. In the following, values with slip are denoted by a $'$. The slip factor can be found by the equation

$$\gamma = 1 - \frac{c_m}{u_1} \left(\frac{1}{\tan \beta_1'} - \frac{1}{\tan \beta_{1B}} \right) \quad (2.17)$$

where τ_1 is the blade blockage factor at the outlet of the pump and β_{1B} is the blade angle at the outlet of the pump. τ is given by:

$$\tau_1 = \left(1 - \frac{z_{La}e_1}{\pi d_1 \sin \beta_{1B}} \right)^{-1} \quad [-] \quad (2.18)$$

where e is the blade thickness and d_1 is the diameter of the runner at 1.

The blade blockage factor depends on the amount of the blade passage which is blocked due to the thickness of the blades and serves to increase the meridional velocity accordingly.

While the slip factor depends on several conditions, it is usually in the range of 0.7 to 0.8 [14, page 147].

2.9 Power

A hydropower plant should utilize as much of the power accessible to it as possible. When relating the energy equation, eq. (2.5), to the mass flow rate of a system, it can be seen that the largest theoretical obtainable power, $P_{t,theoretical}$ for the turbine is given by

$$P_{t,theoretical} = \rho g h_t Q \quad [W] \quad (2.19)$$

Assuming an idealized system where the only losses in pressure are due to the turbine, and evaluating the system between the higher and lower reservoir where the velocity may

be assumed to be zero, yields the result that $h_t = z_0 - z_4 = H_g$. H_g is the so called gross head. When calculating the power of a turbomachine, one has to include losses in the system. The losses in the waterway and in the turbine are usually treated separately. The head losses in the waterway are added to the gross head to give the head which the turbomachine is exposed to. This head is called the effective head, H_E . Whether the effective head is larger or smaller than the gross head depends on whether the turbomachine is a pump or a turbine:

$$H_E = H_G + h_L(Q)$$

$$\begin{aligned} \text{turbine} &\rightarrow h_L < 0 \\ \text{pump} &\rightarrow h_L > 0 \end{aligned} \quad [\text{m}] \quad (2.20)$$

The curve describing H_E as a function of Q is called the system characteristic.

The part of the losses occurring due to hydraulic phenomena in the turbomachine, that is, in the spiral casing, guide- and stay vanes, draft tube and runner, are collected in the form of the hydraulic efficiency, η_h . The hydraulic efficiency is the ratio of the turbine head to the effective head for a turbine or the ratio of the effective head to the pump head for a pump. This can be expressed by application of the Euler equation, 2.7, as

$$\eta_{h,t} = \frac{u_{out}c_{u,out} - u_{in}c_{u,in}}{gH_E} \quad [-] \quad (2.21)$$

$$\eta_{h,p} = \frac{gH_E}{u_{out}c_{u,out} - u_{in}c_{u,in}} \quad [-] \quad (2.22)$$

There are also losses connected to the energy transfer from the turbine to the generator and to the grid. these are collected under the term mechanical efficiency, η_m . The efficiencies are often combined as an overall efficiency, η_o thus the power delivered to the grid becomes:

$$P = \rho g H_E Q \eta_o \quad [\text{W}] \quad (2.23)$$

and the power delivered to the flow by a pump becomes

$$P = \frac{\rho g H_E Q}{\eta_o} \quad [\text{W}] \quad (2.24)$$

2.10 Characteristics

A change in the head of a turbine always leads to a variation of the volume flow rate as well, given that the rotational speed is fixed. The efficiency and power also vary with varying H and Q values. A pump characteristic is a graphic representation of this variation of the head and flow rate. The operation point will be at the flow rate where the pump or turbine characteristic crosses the system characteristic (equation (2.20)). In a properly designed turbine, the BEP, will coincide with the design point of the turbine. The relation between the head and the volume flow of the turbomachine is obtained by combining Euler's turbine equation (2.7) and the continuity equation for a turbomachine (2.16) and relating these to

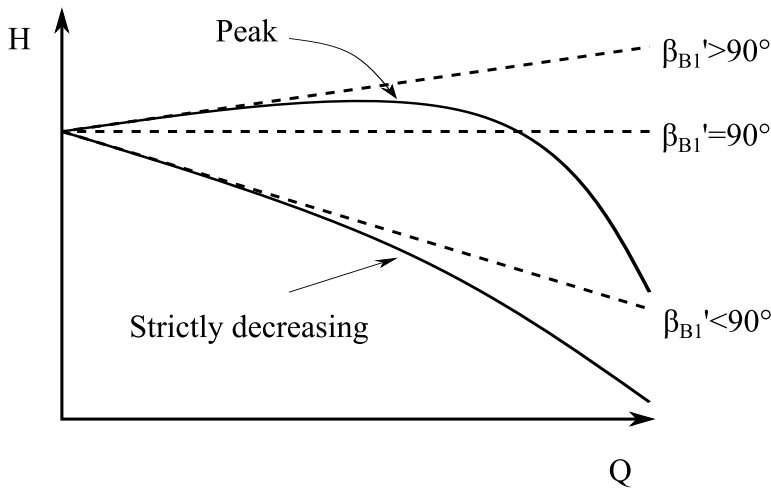


Figure 2.10: Sketch of characteristics for pumps with different blade outlet angles

the geometry of the RPT. This yields the following equation for the pump head H_p

$$H_p = \frac{\eta_h u_1^2}{g} \left(\gamma - \frac{\tau_1 Q}{A_1 u_1 \tan \beta_{1B}} \right) \quad [\text{m}] \quad (2.25)$$

This equation is a simplified form of the one found in Gülich [14, page 348]. The equation in the source material contained a term which would be zero in the current design process and thus has been removed from this version of equation (2.25). By differentiation of equation (2.25) it can be shown that the slope of the curve is dependant on β_{1B} . More specifically the curve is decreasing if $\beta_{1B} < 90^\circ$, constant if $\beta_{1B} = 90^\circ$ and increasing if $\beta_{1B} > 90^\circ$. This is however only true for the entire curve if the case has no losses. In reality, the hydraulic efficiency will change with changing Q , and ultimately the losses will become large enough to reduce the head to zero, regardless of β_{1B} . A schematic view of characteristics with different angles of β_{1B} is shown in figure 2.10

If a pump has $\beta_{1B} > 90^\circ$, the characteristic will have a global maximum for some value of $Q > 0$, and thus also have multiple different volume flow rates which result in the same head. In such a case the flow rate could begin to oscillate between the two flow rates, which is unwanted. To avoid this, pumps have to be designed with $\beta_{1B} \leq 90^\circ$.

The other way to increase the head would be to increase the velocity, u_1 of the RPT. To do this either the diameter or the rotational speed of the turbomachine would have to be increased. However, as the objective of this thesis is to keep everything outside the runner constant, the diameter cannot be increased any further, and a change of rotational speed would encompass modifications of the generator. Thus the main parameter of interest for this work is β_{1B} and the shape of the blade.

2.11 Attached and separated flow

A flow which follows a geometry closely is called attached flow while flow which at some point starts to deviate from the geometry is called separated flow. A flow is prone to separate when it is decelerating, as a decelerating flow is less restrictive than an accelerating one [14, page 8].

Separation in a hydrofoil¹⁵ will lead to stall, which means that the foil does not produce lift anymore[12, page 41]. In a radial runner, separation will lead to areas of recirculation blocking parts of the passage, which again lead to large hydraulic losses.[14, page 8]. An RPT is a radial runner which applies both lift and centrifugal forces to produce head. Therefore separation is a negative phenomenon which should be avoided.

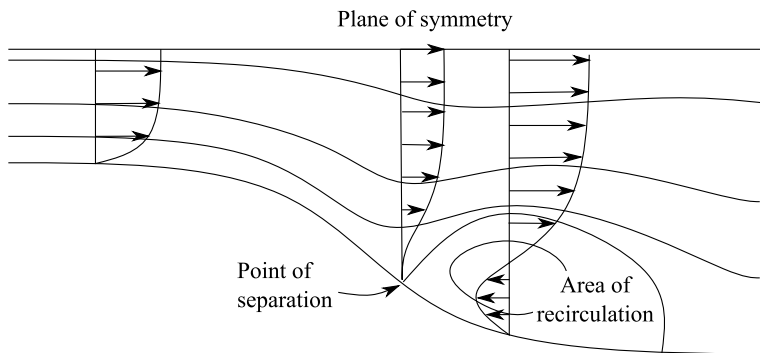


Figure 2.11: The streamlines and the velocity distributions in a separating flow.

Because the Current arrangement at Roskrepp is designed for a turbine, the cross-sectional area at 1 is larger than at 2. By relation to the continuity equation¹⁶, this means that the velocity is larger at 2 than at 1. In turbine mode, this is no problem as the flow will accelerate through the turbine. For pump mode however, the flow will decelerate and separation will most likely occur. Therefore it will be easier to achieve good efficiency for turbine mode than for pump mode. Separation in a flow due to an increase of the cross-sectional area is shown in figure 2.11.

2.12 Recirculation

When a pump is operating against a closed valve, the volume flow rate will be zero and the head will be the so-called shut off head. When this happens there will be recirculation flow both at the runner inlet and outlet. This happens because the pressure at the shroud of the rotor is larger than the pressure at the hub of the rotor. This creates an imbalance between the pressure in the rotating and the stationary parts of the fluid. Thus the fluid flows out of the runner at the shroud and enters at the hub. This happens for all pumps

¹⁵A hydrofoil is a body formed to produce lift in an incoming flow.

¹⁶For an incompressible fluid, the volume flow rate into a control volume is equal the volume flow out of a control volume[7, page page 192]

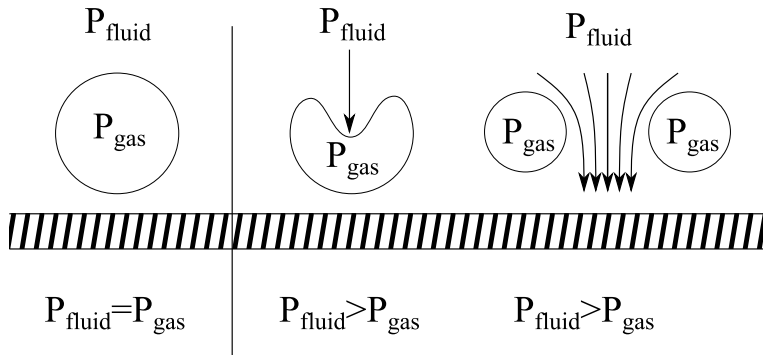


Figure 2.12: Collapse of a cavitation bubble.

with significant radial differences between the inner and outer streamline at either inlet or outlet.

As such a flow pattern cannot simply appear out of nowhere, similar conditions must develop already at higher volume flow rates. While there are no general indicators of when recirculation will occur, it appears in all pumps operating at flow rates considerably lower than their BEP flow rates. There are also two prerequisites to be fulfilled for recirculation to occur, namely locally separated flow and strong pressure gradients perpendicular to the mean flow direction [14, page 201].

2.13 Cavitation

A turbomachine may encounter problems if the static pressure in any place becomes too low. As explained in section 2.1.1, the static pressure decreases if the flow velocity or the elevation increases. If the static pressure decreases below the vaporisation pressure of a fluid, the fluid will experience a change in phase from liquid to gas form. Such a phase shift is called cavitation.

Cavitation is a serious problem in turbomachines. The vapour bubbles block part of the runner opening and thus increase the fluid velocity of the surrounding flow, which again leads to larger pressure losses. Furthermore, the bubbles collapse when they are transported into an area with higher static pressure. When this happens tiny jets with high velocities are formed which are capable of damaging the runner, this is illustrated in figure 2.12. Therefore it is important to avoid cavitation in turbomachines.

To ensure that the pressure is high enough, one usually looks at the Net Positive Suction Head (NPSH). It describes the difference between the available head and the vapour pressure, also expressed as a head, at the low-pressure side of the turbomachine.

Brekke [5, page 44] suggests the application of two different kinds of NPSH, namely the required and the available NPSH, labeled NPSH_R and NPSH_A respectively. While the available head can be readily found by application of the energy equation on the system, the required NPSH is a machine property. Based on empirical data, Brekke suggests using

the formula

$$\text{NPSH}_R = a \frac{c_m^2}{2} + b \frac{u^2}{2} \quad [\text{m}] \quad (2.26)$$

where a and b are empirical variables defined for pumps and turbines as shown in table 2.1. c_m is the average value at the low pressure side and u is evaluated at the outer diameter at the low pressure side. The requirement to avoid cavitation is that $\text{NPSH}_A > \text{NPSH}_R$.

Pump	Turbine
$1.6 < a < 2.0$	$1.05 < a < 1.15$
$0.2 < b < 0.25$	$0.05 < b < 0.25$

Table 2.1: Empirical data for factors related to the values, collected from [5, page 46]

In the design of a new hydropower plant, sufficient NPSH_A is ensured by moving the runner far enough below the lower reservoir. However, as seen by the values in table 2.1, the requirements are stricter for a pump than for a turbine because inflow at low pressure is more likely to result in cavitation than outflow at low pressure [5, page 46]. Therefore the problem of cavitation in the process of retrofitting will be solved by installing a booster pump between the runner and the lower reservoir, which should produce enough head to avoid cavitation.

The shape of the blades also has an impact on the cavitation behaviour. Abrupt changes in the geometry lead to sharp increases in the velocity, which in turn reduce the static pressure. One location where this may occur is the leading edge of the blade. If the incoming flow does not have the same angle as the blade angle at the inlet, so-called incidence¹⁷, high velocities will be generated around the leading edge. The low static pressure values corresponding to the high velocities increase the risk of cavitation. The profile of the blade leading edges has a lot to say for the entering flow, as seen in figure 2.13. The use of circular profile shapes is discouraged as it is very unfavourable in this regard. While wedge-like shapes are optimal for zero incidence, they perform even worse than circular ones for flow with incidence. The compromise is to use an elliptical shape, which performs reasonable well both with incidence and with zero incidence.

2.14 Differences between pumps and turbines

Now that all relevant flow phenomena linked to turbomachines are explained, it is time to take a step back and to look at runners and runner blades as a whole.

As mixed-flow pumps and Francis turbines look similar, one may believe that an RPT should be able to operate at BEP under the same conditions for both pumping and turbine operation. As shown in Brekke [5, pages 147-149] this is not the case. First of all, for constant gross head the effective head will be different for pumps and turbines, as the sign of the friction losses in equation (2.20) will change. Thus the operating conditions experienced by the pump will differ from pump to turbine mode. In the case with a booster pump installed it could be possible to increase the pressure on the suction side of the pump to compensate for the difference in effective head. Even with the same H_E for both

¹⁷The incidence angle is the difference between the blade inlet angle and the flow inlet angle.

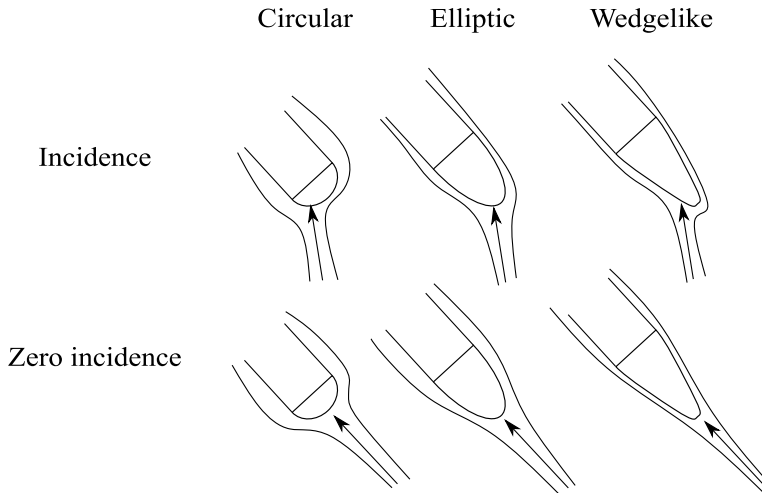


Figure 2.13: Disturbance of the flow with different leading edge shapes

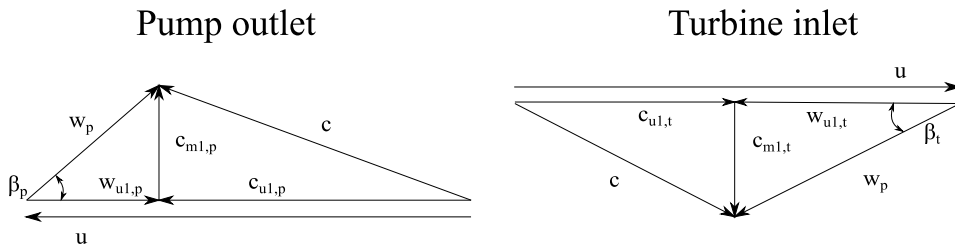


Figure 2.14: The velocity triangles at 1 for a pump and a turbine at same Q and H_E

cases, the operation conditions still wouldn't be the same. Combination of the hydraulic efficiencies for turbines and pumps, eq. (2.21) and (2.22), with identical effective head and zero circulation at the low pressure side yields:

$$\eta_{h,t}\eta_{h,p}|c_{u1,p}| = |c_{u1,t}| \quad [\text{m s}^{-1}] \quad (2.27)$$

Here the subscripts p and t denote pump and turbine mode, respectively. The absolute values of both sides have been taken to avoid confusion with sign conventions. In reality, the hydraulic efficiencies will always be smaller than unity so there will be a discrepancy between $|c_{u1,p}|$ and $|c_{u1,t}|$. As illustrated in figure 2.14 the angle of β_1 must be larger for a pump than for a turbine. This effect is further amplified by slip as well, as it already forces the pump blade angle to be larger than the flow angle.

2.15 Design philosophies for pump turbines

Due to the difference in velocity triangles as explained in section 2.14 a turbine operating at its BEP will become a pump operating away from its BEP if it is reversed. Likewise, a

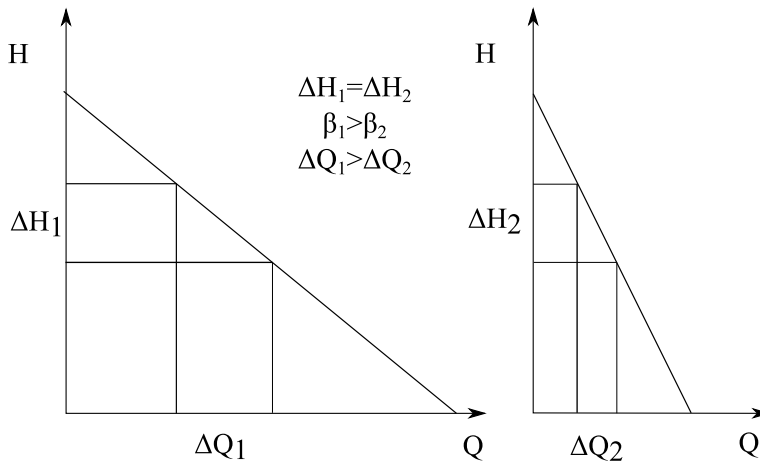


Figure 2.15: Variation of head for fixed change in volume flow rate in idealized pump characteristics with different slopes.

reversed pump at BEP will become an off-BEP turbine.

A reversed pump will always be able to produce some power at lower heads as well, but a reversed turbine will not necessarily be able to produce sufficient head to pump water from the lower to the higher reservoir. Therefore it is more important to ensure good efficiency in pump mode than in turbine mode.

For this reason, the RPT will primarily be designed with sufficient lifting head in mind while good efficiency in turbine mode will be of secondary concern.

As explained in section 2.10, a change in β_1 leads to a change in the slope of the characteristic, and thus to a change in head.

An increased slope is unfavourable for the operation of the RPT because a moderate slope will lead to large variations of Q with small variations of H_p , as illustrated in figure 2.15. This may be problematic because the volume flow rate is decided by the effective head which varies considerably as the reservoir level varies greatly throughout the year. To keep the turbine operating close to the BEP flowrate, the variations in Q should be insensitive to the variations in head. Therefore one wants to find a value for β_1 which is large enough to deliver sufficient head, but not larger, to avoid major variations in Q .

Two different design philosophies have been considered in this project. One is the application of the energy distribution by application of uc_u curves, as is customary at the waterpower laboratory of NTNU [39, page 19]. As seen in Euler's turbine equation (2.7), the power transferred to or from the flow is proportional to the change of $u \cdot c_u$, this also goes for the energy. This means that by controlling this coefficient, one can control where along the blade the energy transfer will take place. To reduce slip in pump operation, most of the energy should already be transferred from the runner to the fluid ahead of reaching the outlet, so the pressure difference between the pressure and suction side has the possibility to equalize. The uc_u value should also be zero at the outlet of the turbine, so all of the available power is transferred to the runner. Thus we want the distribution of uc_u to look approximately like shown in figure 2.16. The other method is to simply

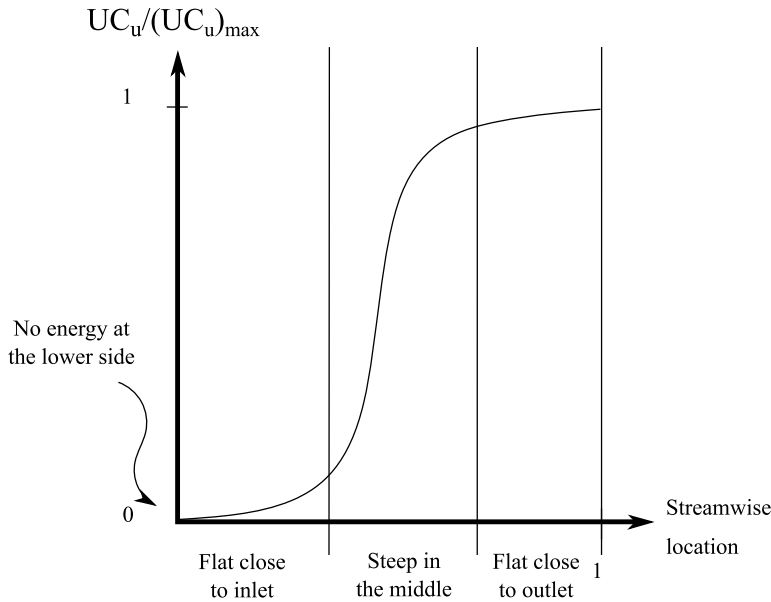


Figure 2.16: Dimensionless UC_u curve along dimensionless streamwise location

distribute the values of β continuously from β_1 to β_2 , as is recommended by Stepanoff [33, page 99]. This second method grants no control over the power throughout the turbine, but more direct control over the actual shape of the blade, and ensures the blade to be smooth everywhere. It may be advisable to start by using the uc_u distribution for an initial design and to improve the hydrodynamic behaviour of the blade by altering the β values directly.

2.16 Computational fluid dynamics

To test the RPT without having to make a physical version of every geometry to be tested, computational fluid dynamics, hereafter called CFD, are taken in use. As with all numerical methods, CFD divides a fluid domain into smaller subdomains and approximates partial differential equations by simpler algebraic equations. The boundary conditions are stated and the equations are solved iteratively for all elements in the domain. The CFD solver which will be used in this thesis is ANSYS CFX, so only theory applicable to this program will be treated in the following.

2.16.1 Convergence of steady state solutions

The kind of CFD-simulation which usually requires the least amount of computing power is a steady-state solution. The premise of steady-state solutions is that the flow, if given enough time, will arrive at the same, unchanging state, independent of the initial condition of the flow. Therefore a steady-state solver does not concern itself with accuracy in time,

which usually leads to quicker convergence, as the time step can be chosen arbitrarily large[42].

In order to get rid of flow conditions originating due to the specified initial conditions of the domain, it is common practice to run each simulation at least long enough for the flow to pass through the entire domain once.

It is important to determine whether a simulation has converged sufficiently before accepting it. Most solvers, therefore, output the residual values of the cells, to give an indication of the convergence. Residual values are the imbalances between the right-hand side and the left-hand side of the equations being solved in the numerical approach[8, chapter 11.2.2]. Usually the Root Mean Square¹⁸ (RMS) value of all of the residuals is given, as well as the largest residual of the iteration. Generally speaking, the smaller the residual values are, the more converged the equation will be. However, the residuals always have to be viewed relative to the scale of the entire problem. Therefore residuals are often written normalized to the scale of the entire domain.

Besides looking at the residual values of a simulation it is useful to monitor properties relevant for the case. Such properties could be the thrust on runner blades or the head difference between the outlet and the inlet. If the residual values have converged but the other relevant properties are oscillating, it means that the simulation is either transient or unstable due to poorly defined boundary conditions or poor grid quality. If the flow never converges to one state, it could be inherently transient, which means that the actual flow always will be changing. Then a transient solution should be run, to resolve the temporal variation of the flow.

2.16.2 Transient solutions

If the simulation is transient, a transient solver must be used instead of a steady-state solver. For transient solvers, the Courant number, Co is an important factor for convergence and accuracy.

$$Co = c \frac{\Delta t}{\Delta x} \quad [-] \quad (2.28)$$

Δt is the value of the timestep chosen for the simulation, Δx is the length of a grid cell and c is the local velocity of the flow at the grid cell[3, chapter 1.1.3]. The allowable value for the Courant number varies from solver to solver and from case to case. The solver used by ANSYS CFX is an implicit solver, which means that it can reach convergence with any value of Co but smaller values tend to give more accurate results.

Transient solvers will work for steady-state solutions as well but they generally take longer to converge and require larger amounts of memory to save states for different time steps of the simulation. Therefore a steady-state solution is usually attempted first.

A transient solver uses an initial value for the flow field and then marches it forward in time. Again the residuals are monitored to see if convergence has been reached and each time step is iterated until the residuals have reached a sufficiently low level before the time is incremented to the next time step.

¹⁸For n discrete elements with value x , the RMS value is found by $x_{RMS} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_n^2)}$

Even if the residuals of each time-wise iteration are very small and the solution thus converges, the accuracy of the simulation still has to be evaluated. This is done by use of a so-called grid sensitivity study. This means that parameters like the convergence tolerance, time step size and mesh size need to be changed to see whether it leads to significant changes in the solution. If it does change, the case must be modified, usually by either making the mesh finer or decreasing the timestep until further refinement leads to insignificant changes in the flow.

2.16.3 Turbulence models

Turbulent flow includes flow phenomena of very varying length and time scales [34, pages 21-22]. If one would intend to solve equations for all scales this would require an extremely fine mesh and a small timestep. This would make even the simplest turbulent simulations very costly in terms of computation capacity. Therefore most solvers use the average values of the equations to be solved and apply models to adjust for the fluctuating components, so-called turbulence models. A large number of turbulence models have been developed, all of which have their own advantages and disadvantages regarding complexity, computation time and flow conditions. As separation is expected in the turbine, stability with regard to pressure gradients and separation is important for the turbulence model to be used.

Two commonly used turbulence models are the k - ϵ and the k - ω models, which use equations for the kinetic turbulent energy k and either for the turbulent dissipation ϵ or for the specific dissipation ω . The k - ϵ performs good in free shear layers, but struggles in dealing with areas of separation, while the k - ω model performs better close to walls but struggles with free shear layers and adverse pressure gradients.

To circumvent the weaknesses of these models, a hybrid approach, called the Shear Stress Transport (or SST) k - ω model, has been developed. The SST k - ω is among the most accurate turbulence models with regard to adverse pressure gradients [43] and is able to handle separation quite well[21].

2.16.4 Wall models

No matter how fast a fluid flows past a surface, and how turbulent the mean flow is, the flow will always become laminar if it is close enough to the wall. This is due to the no-slip condition which ensures that the velocity at the wall is zero, so the turbulence at the wall will be zero as well. Because of the sharp velocity gradient this creates, the flow at the wall has to be modeled [22]. Most turbulence models require the flow to be outside of the viscous sublayer due to difficulties with modelling the turbulence in this part of the flow. Another advantage of the k - ω SST model is that an analytical equation for ω in the laminar boundary layer is known. Therefore it can make use of automatic near-wall treatment, which means that it does not matter if the first grid point is inside the laminar or the turbulent part of the flow. In order to resolve the boundary layer sufficiently, it is still advisable to have ten grid points inside the boundary layer¹⁹ [8, chapter 2.8.1].

¹⁹The boundary layer is defined as the part of the flow close to the wall which is slower than 0.99 times the free stream velocity[41, page 149]

2.16.5 Considerations for simulations of turbomachines

If a turbomachine can be assumed to be periodic in the circumferential direction, i.e. that the flow is identical for all blades, this can be used to decrease the computational requirements of the turbine drastically. In such a case only one blade is simulated, and the flow exiting the simulated domain on one of the circumferential boundaries enters the same domain on the other side.

The simulation of flow through a turbomachine includes frame changes due to the rotating parts. To be able to simulate these, a couple of different methods have been developed. The frozen rotor approach assumes that the rotating and stationary domains of the geometry are stationary relative to one another, but changes the frame of the flow in the rotor-stator interface. The obvious disadvantage of using the frozen rotor approach is that the flow is only calculated for one rotor-stator configuration, and thus is inaccurate if the flow in the stator is not uniform in the circumferential direction.

Another way of transforming the flow components between rotor and stator is to use circumferential averaging. When doing this the flow is averaged in circumferential slices around the axis, and thus the average flow experienced by the runner is captured. The second approach, called the mixing plane approach, is usually more correct than the frozen rotor approach[20], but it performs poorly at off-design conditions for pumps and is a bit slower than the frozen rotor approach.

The last option is to perform a fully transient rotor-stator simulation. The only assumption made here is that the flow is periodic in the circumferential direction, so the flow in all blade rows is the same. The rotor is then actually rotated around the axis and the domain interfaces between rotor and stator are connected by periodicity. This option is the most correct one, but it also requires the most computation time.

2.16.6 Verification and validation

One should keep in mind that CFD only is an approximation of a real flow. Therefore it is important to remain sceptical to the results obtained by simulations. This is done in the verification and validation parts of the simulation process.

Verification concerns itself with the mathematical aspect of a CFD code and serves to find errors in the code and implementation of the equations to be solved. Verification is sometimes described as making sure one is “solving the equations right”. The verification process is divided into two aspects[32]: One aspect is the verification of a code where the program itself is tested to find errors and bugs. The other is verification of a calculation, which is focused on evaluating the error of a single calculation. While verification of the code should be done by the software distributor prior to the release of the CFD software, verification of the calculation has to be done by the user. It includes performing a grid convergence study to determine how converged the solution is and how large the errors are. While experimental solutions should not be used to compare the numerical solution in the verification process, exact analytical solutions can be used.

The validation process serves to ensure that the simulation matches the physical conditions which are to be studied[31]. In other words, validation is making sure the code is “solving the right equations”. This is done by comparing the results obtained with the CFD solver to experimental data.

To reduce the errors found in the validation process it may be worthwhile to change some of the model parameters of the simulation. Particularly the turbulence models are known to have varying accuracy for different flow conditions[43].

METHOD AND RESULTS

■ In this chapter, the method and results of the thesis are explained and presented. First, the method used to design the blade is outlined, then the resulting shape of the blade is presented, as well as the angles along the blade. Following this, the process for running the CFD simulations is explained and the results are presented and discussed.

3.1 Geometry

As explained in section 2.15, the RPT will be designed mainly as a pump, but with minor modifications to make it work at a better efficiency as a turbine as well.

This section will describe the design process of the geometry, beginning with the meridional view of the pump and ending with a three-dimensional digital model which can be interpreted by a CFD software. This process is the same as the one utilized in the scripts supplied in appendix G.

3.1.1 Meridional view

The first step in defining the geometry is to find the outline of the meridional view. The meridional view is a cross-section through the axis of the runner, on which the blade is projected by means of radial projection, as shown in figure 3.1.

In this case, the main dimensions are already given by the current shape of the runner. To digitalize the shape of the current runner, Bézier curves [28, page 11] are used together with three control points through which the curves have to pass. First, the hub is defined. The three points which it has to pass through are the point at the hub at which the guide vane's stem is located, the point where the runner meets stationary part and the point at the centre of the runner, where the runner meets the draft tube. As the hub does not extend

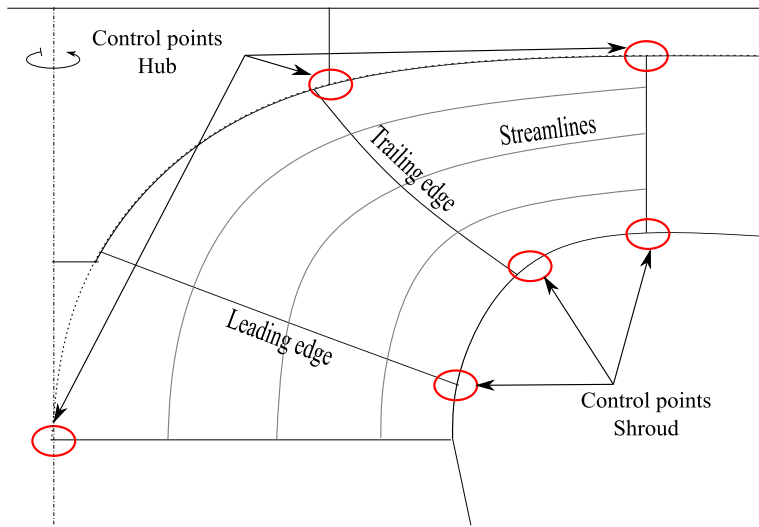


Figure 3.1: Meridional view of the runner with control points, streamlines and blades indicated

entirely down to the draft tube, the last point will not be a part of the actual geometry but is used to define the initial runner geometry.

When the hub is defined, a number of equidistant points are placed from the hub down to the shroud at the guide vane. These points will be used to divide the entire passage into lines which will be assumed to possess streamline¹ properties. Thus it is assumed that the flow follows the lines in the meridional plane. The area at the inlet and outlet of the runner is known, but how and where it changes is up to the designer. Thus the cross-sectional area at any streamwise location is found by letting the variation of the area from inlet to outlet follow a smooth Bèzier curve. The process is explained in greater detail in appendix C. The last streamline to be found is that of the shroud itself. The curve describing the change in area is varied until the shroud streamline coincides with its control points. In addition to giving good control over the cross-sectional area of the runner in the streamwise direction, the streamlines are later used to shape the blade and to calculate velocity components along it.

Following this the leading edge and trailing edge² of the blades are defined in the meridional view. For the trailing edge the runner hub and shroud are simply connected close to the outlet. For the leading edge, close to the low-pressure side, the hub is ended at the same diameter as the current geometry, and the blade ends at the same location.

In this way, the outline of the meridional view is found, as well as the cross-sectional area of the passage.

¹Streamlines are tangential to the velocity everywhere in a flow field, flow cannot pass through a streamline and the volume flow rate between two streamlines is constant.[7, page 141]

²The leading edge is the edge of the blade which the water encounters first in pumping mode. The Trailing edge is the edge which the water leaves.

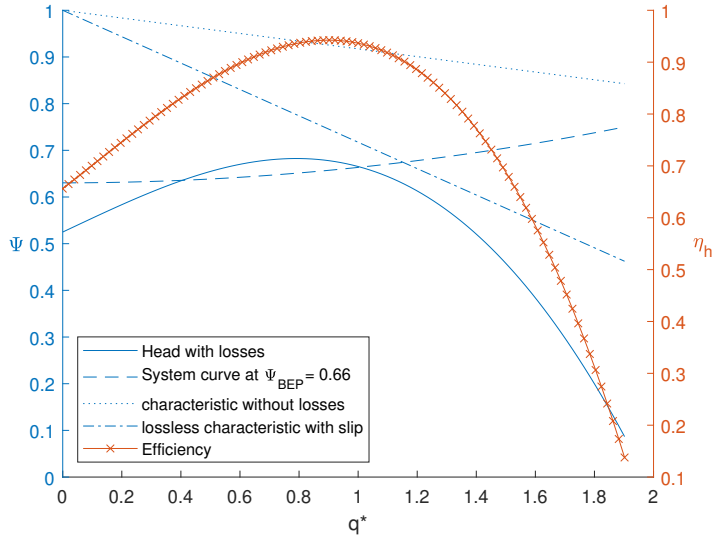


Figure 3.2: Predicted Characteristic of the blade

3.1.2 Blade angles at the trailing edge

To find the angles of the blades, the mean diameter of the trailing edge is found. This diameter is used as a reference for all other values at the outlet. Then the value for β_{1B} is found by application of the pump characteristic, equation (2.25), and the system characteristic, equation (2.20). After adjusting for the losses between the runner and the lower reservoir, and adding the assumed head of the booster pump, the required effective head coefficient is found to be $\Psi_E = 0.66$. The values of the system characteristic are found by empirical data for the gross head and the head losses in the current case and the efficiency for the pump characteristic is found by use of an equation³ suggested in “Centrifugal pumps” [14, page 166]. The slip factor is initially guessed to be $\gamma = 0.7$.

The value of β_{1B} is to be kept as small as possible to obtain sufficiently steep curves as discussed in section 2.15. The predicted characteristic is shown in figure 3.2.

While the head curve of the predicted characteristic does have a peak, the values used in the source material to find the equation for the efficiency include lots of scatter, especially away from the BEP, so a predicted instability does not necessarily mean that the actual pump is unstable and vice versa. Because the radius of the trailing edge varies from hub to shroud, it has to be changed accordingly. This is done by assuming the flow between the runner and the guide vanes to follow a free vortex. In other words, β_{1B} is chosen by using the mean radius and the corresponding circumferential velocity as reference for all other values R and c_u along the blade trailing edge:

$$Rc_u = R_m c_{u,m} \quad (3.1)$$

³ $\eta_h/\eta_{h,max} = 1 - 0.6(q^* - 0.9)^2 - 0.25(q^* - 0.9)^3$ where q^* is the design flow rate.

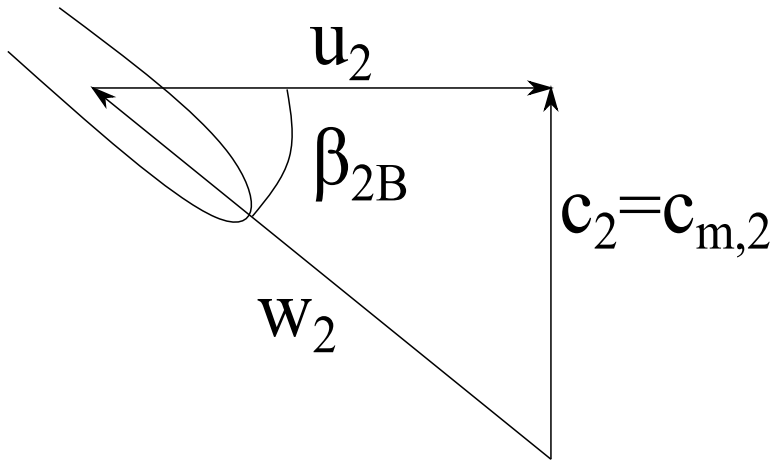


Figure 3.3: The inlet of the pump at design conditions with irrotational inlet flow.

In this way, the flow leaving the pump will be uniform and have the same velocity components at equal radii, which is important to ensure shock-less entry to guide vanes.

3.1.3 Blade angles at the Leading edge

In order to find the head, equation (2.25) assumes irrotational flow at the inlet. The same assumption is used when finding the blade angles at the leading edge, β_{2B} . In order to have shockless entry in pump mode, the relative velocity has to have the same angle as the runner blades, as shown in figure 3.3, so:

$$\beta_{2B} = \arctan\left(\frac{c_{m,2}}{u_2}\right) \quad (3.2)$$

as the value u_2 decreases with decreasing radius and the meridional velocity is assumed to be constant, the angles increase close to the hub.

3.1.4 Distribution of blade angles along the blade

The blades are shaped by distributing the blade angles smoothly from leading edge to trailing edge. This is done by once again using a quadratic Bézier curve while monitoring the predicted shape of uc_u values along the streamwise direction. Allowing the uc_u curve to get too steep at low spans⁴ leads to sharp changes in the blade which would further reinforce the conditions supporting separation. The final uc_u curve is shown in figure 3.4. The x-axis of the figure describes the dimensionless distance from the pump inlet to the pump outlet. The highest curve is the curve at the shroud and the lowest is at the hub.

⁴The span is the distance from the runner hub to a point of the blade. When nondimensionalized, the span is 0 at the hub and 1 at the shroud.

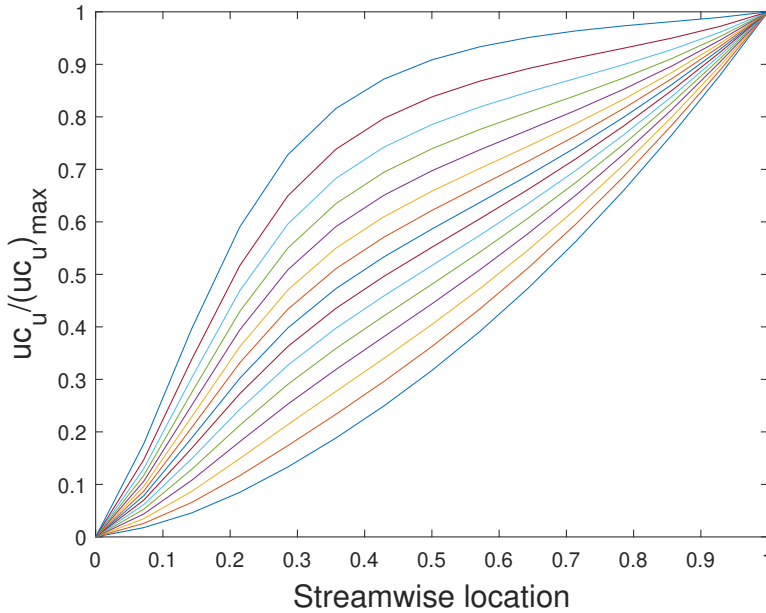


Figure 3.4: Distribution of the uc_u values along the blade

3.1.5 Checking for cavitation

To do an initial check of whether the blade shape would lead to cavitation, the $NPSH_R$ of the entire blade is compared to the sum of the empirical values for $NPSH_A$ and the assumed head value of the booster pump. The velocities are found by use of velocity triangles along the blade. The factors used to find $NPSH_R$ as in equation (2.26) are $a = 2$ and $b = 0.25$. These are the values which give the strictest $NPSH_R$, in order to give a conservative estimate of the cavitation for the runner. This gives

$$\Psi_{NPSHR} = 0.2085 < 0.2590 = \Psi_{NPSHA} \quad (3.3)$$

Which means that cavitation requirements should be satisfied. However, the real flow is three dimensional, and the calculated velocity components do not account for complex flow patterns, so cavitation could still occur.

3.1.6 Blade thickness and 3D

With all of the values for beta known, the three-dimensional shape of the blades can be found. First, the meridional projection of the blade is rotated around the axis corresponding to the values of β for each streamline, as explained in appendix D.

Then thickness is added to the blade, according to the recommendations given by Gülich[14, page 347]:

$$e \in [0.016 \cdot D_1, 0.022 \cdot D_1] \quad (3.4)$$

The value chosen for the blade thickness is $0.019 \cdot D_1$, so in the middle of the two extremes.

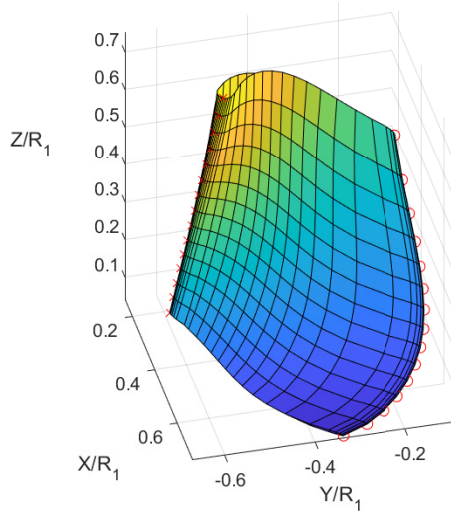


Figure 3.5: Figure of the blade

The leading edge and trailing edge are then shaped like half ellipses with half lengths equal to the blade thickness⁵

To easily export the design to the CFD solver, the geometry is written as `.crv` files, one containing the data for the hub, one for the shroud and one for the blade. The finished blade is shown in figure 3.5

3.2 Discussion of the design method

While distributing the starting points of the streamlines in the design process along the stem of the guide vane is simple in terms implementation, it does lead to some differences between the actual shape of the waterway and the shape used in the design. The actual shape has a flat portion at the hub, so the guide vane can rotate unhindered. This is not true for the digital model.

Another aspect is the blade loading. While the current design was modelled while keeping the uc_u distribution in mind, and viewing the shape of the blades to ensure no sudden changes, there is no evidence for this to be the right procedure. The current method resulted in rather short blades when comparing the geometry to other pumps. For the pressure to equalize through the runner, longer blades would be beneficial. The blade length could be increased by using other blade loading methods.

There is also the matter of choosing how the blade angles at the trailing edge behave. As explained in section 3.1.2, the current design principle uses free vortex theory in the area after the trailing edge. While this should yield flow with uniform circumferential

⁵So the ellipse is twice as long as it is thick.

velocity downstream of the runner, it leads to part of the blades trailing edge to be larger than 90° , as the flow needs a larger circumferential component at small radii and the blade trailing edge has smaller radius at smaller span. While this is not assumed to result in an unstable characteristic, it can lead to disturbances in the flow, as will be shown in section 3.4.3

3.3 CFD setup

This section will describe the setup for the CFD-simulations, give reasons for why choices were made and describe the uncertainties correlated to the simulations. It should be mentioned that no simulation has fully converged, although many different attempts to reach convergence have been made.

3.3.1 The computational domain and boundary conditions

The geometry is simplified to only include the geometry of the runner itself and the draft tube. The spiral casing is excluded as it is not axisymmetric and thus the entire circumference would have to be meshed and simulated. Instead it is modelled by an empirical model provided by Gülich[14, page 141]. The guide vanes have also been omitted and are modelled by another empirical model. This is because the flow angle of the exiting flow has to match the angle of the guide vanes to minimize the losses. While an approximation of the flow angles can be found by use of velocity triangles and slip factors, the real flow will likely have deviations from those. Thus every case would have to be simulated multiple times to find optimal angles for the rotor-stator interaction. With the empirical model, on the other hand, the guide vane angles can just be input after the outlet velocity angles are known. Additionally, replacing the guide vanes reduces the numbers of cells needed for the simulation. The stay vanes are treated similarly as the guide vanes. As no reliable data for the shape of the stay vanes was available, it was assumed that they were the same shape as the guide vanes. The models and model assumptions are described in appendix E

The last simplification is that the centre of the draft tube is not included in the fluid volume. This is again done to reduce the number of grid points in the domain, as the centre part would either have to have very thin and long “pizza slice shaped” cells in the middle or its entire cylindrical shape would have to be modelled. Either way, the cell count would increase significantly in exchange for what is assumed to be a minor change in the head.

To obtain a pump characteristic for the RPT, there are two possible ways in which the boundary conditions can be defined. Either the head could be defined by imposing different pressures at the inlet and outlet, to calculate the resulting volume flow rate, or the volume flow rate could be defined on one side, and a pressure at the other boundary, and use this to measure the pressure rise over the runner.

There is no guarantee that the characteristic will be stable, so one head value could result in multiple volume flow rates, which would make it difficult to obtain the right Q -values. Therefore the second option is chosen. The mass flow rate corresponding to the required volume flow rate is specified at the outlet. At the inlet, the total pressure is known from average pressure data from Roskrepp Powerplant but the CFD solver “CFX” requires the inlet velocity direction to be specified in order to use the total pressure option.

With static pressure, this is not a requirement. Therefore the inlet boundary is set to static pressure with constant gradient chosen for both the velocity and the turbulence. The static pressure is found by use of the energy equation (2.5) with an energy correction coefficient for turbulent flow of $\alpha = 1.075$.

Simulations are performed for volume flow rates of $q^* = 0.7, q^* = 1$ and $q^* = 1.3$ in order to get an idea of the shape of the characteristic.

The walls at the hub, shroud and blade are specified to be either rotating or stationary no-slip walls. The exception is the false wall at the centre of the inlet domain. This wall is chosen to have free slip, i.e. it does not interact with the fluid by creating a boundary layer or requiring any special form for wall treatment. It only prevents the flow from flowing through.

CFX encounters problems when faced with flow entering an outlet boundary or exiting an inlet boundary. The response of the software is to place an artificial wall where the fluid would flow in the wrong direction. As this does not correspond to the actual physics of the system, two possibilities exist to avoid it. The first one is to define the boundary conditions as openings, which would allow for flow in both directions. However, the opening condition does not allow for the flow passing through it to have zero gradient conditions. In other words, one would have to specify the angle of the flow passing through, which could influence the interaction between the pump and the incoming or leaving flow. The second option is to move the inlet and outlet conditions far enough away from the RPT that any recirculation caused by the runner has evened out prior to reaching the boundary. The second option requires a larger computation domain, but will most likely yield a more accurate result, and is therefore chosen.

The inlet is extended downward, following the expansion of the draft tube. While Gülich states that recirculation can have a visible effect on the flow up to more than $10D_1$ away from the runner [14, page 200], some trial and error showed that $5D_1$ was sufficient for the flow to be uniformly entering the runner.

The outlet is also extended, but as it is extended in the radial direction, the cross-sectional area increases with increasing distance between the runner and the outlet. As explained in section 2.11, an increase in area may lead to separation which could amplify the problems with inflow at the outlet. To avoid separation, the outlet area is kept constant by decreasing the height corresponding to the increase in radius. The meridional outline of the geometry is shown in figure 3.6.

3.3.2 The solver settings

Due to the simplifications made to the geometry as explained in the previous section, the only blade in the geometry is the rotor blade. The frozen rotor model only simulates the rotor and stator at one relative location, but as the stator now is axisymmetric, the relative location of the rotor and the stator is the same, independent of the rotational location of the rotor. This means that the frozen rotor approach should be equally as correct as the mixing plane approach, but the frozen rotor approach demands less computation than the mixing plane. Therefore the frozen rotor approach is selected as the steady-state rotor-stator model.

The buoyancy of the model was turned on in order to include hydrostatic pressure in the calculation. The gravity was chosen to be -9.82 m s^{-2} in positive z-direction.

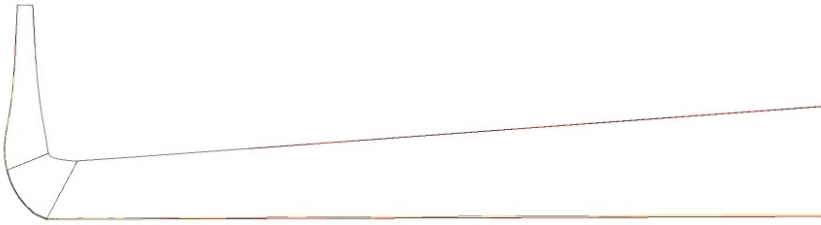


Figure 3.6: Meridional view of the entire domain.

The reference pressure was chosen to be 0 Pa, to avoid confusion with pressures in the results, as all pressures would then be output relative to 0.

As the average air temperature at Roskrepp is approximately 5 °C [35] and there are no temperature measurements available, the water temperature in the model is chosen to be the same as the air temperature.

The discussion of turbulence models in the theory chapter leads to the conclusion that the most sensible choice for the current simulation is the $k-\omega$ SST model with automatic wall functions, to deal with the separation. For the automatic wall functions to work properly, ten cells should be located inside the boundary layer. This is ensured by first running a simulation with coarse mesh close to the blade, examining the boundary layer thickness of the initial run and then performing a mesh refinement with enough mesh cells inside the boundary layer.

The advection models for both the flow and the turbulence are set to high resolution. The other option would be the first order upwind method which is known to introduce large numerical diffusion to the case, which wouldn't be there in a real flow. In other words, the first-order upwind method smears out the values in the simulation, which is not wanted[38, page 119].

3.3.3 The grid and the grid refinement process

The grid is generated in an iterative process, as the accuracy of the simulations depend on the grid, and the grid should be finer where the simulations require more accuracy.

The grid is generated in the semi-automated meshing program ANSYS Turbogrid. Turbogrid allows for a hexahedral mesh generation around a curved surface like a blade. Hexahedral structured meshes have the advantage that they require lower memory than unstructured meshes, as their structure makes it easier to index adjacent cells.

In Turbogrid, the user is given some control over the mesh close to the blade, like the size of the smallest grid point close to the blade, and the number of cells in the spanwise direction, as well as general fineness of the grid. The software also enables the generation of the inlet and outlet region, but the only control over the grid given to the user in these domains is the streamwise expansion from the grid cells close to the runner and to the

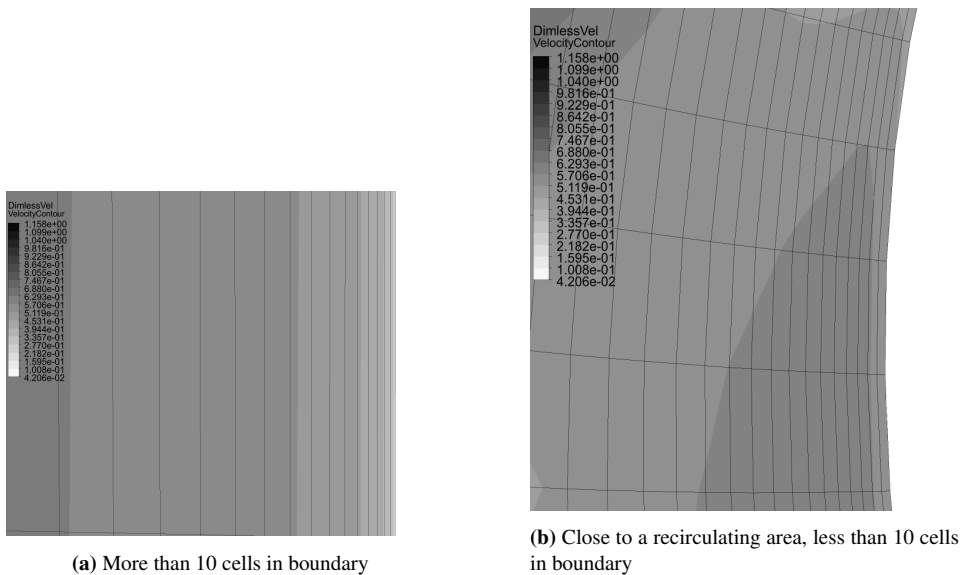


Figure 3.7: Mesh refinement close to the wall.

boundary condition. Turbogrid does not allow for components with a radius close to or equal zero, which also played a role in not creating a fluid volume for the centre of the draft tube, mentioned earlier. The expansion factors for the inlet and the outlet are sat to 1.09 and 1.1, respectively.

As mentioned earlier, the grid close to the boundaries is found iteratively by inspecting the boundary layer thickness and fitting ten cells inside of it. As the boundary layer became very thin close to an area of separation, a compromise had to be made between fitting sufficient grid cells inside the layer and keeping a low cell size and a reasonable mesh quality.

A section of the mesh with a velocity contour is shown in figure 3.7. The velocity is at its largest where it is the darkest. Thus ten cells have to fit inside the grid between the wall and the darkest flow area. This is achieved in figure 3.7a, as the flow is approximately parallel with the wall, but it is not achieved in figure 3.7b, as this is close to an area of recirculation. In order to refine the entire mesh, the software offers a mesh refinement factor. Increasing this increases the fineness of the mesh. The refinement factors of meshes used for simulations are 1.2, 1.5 and 1.7 figure shows the mesh around the blade for a mesh refinement factor of 1.2. Table 3.1 shows the mesh refinement factors together with the number of cells in the inlet, outlet and passage of the blade.

A disadvantage of structured mesh is that it can become very skewed for complex geometries[4, page 300]. This happens for the outlet mesh of the runner, as shown in figure 3.8, and no amount of mesh refinement has been able to solve the problem. Therefore the assumption is made that the affected area is far enough downstream of the runner for not to have significance for the head. Far upstream, the cells close to the wall got very thin compared to the streamwise length of the cells. While this is usually problematic, it is

Mesh refinement factor	Cells in:		
	passage	inlet	outlet
1.2	1 037 100	409 500	245 700
1.5	4 467 852	873 792	801 837
1.7	6 759 540	1 153 803	1 060 038

Table 3.1: Mesh refinement factors and corresponding cell counts

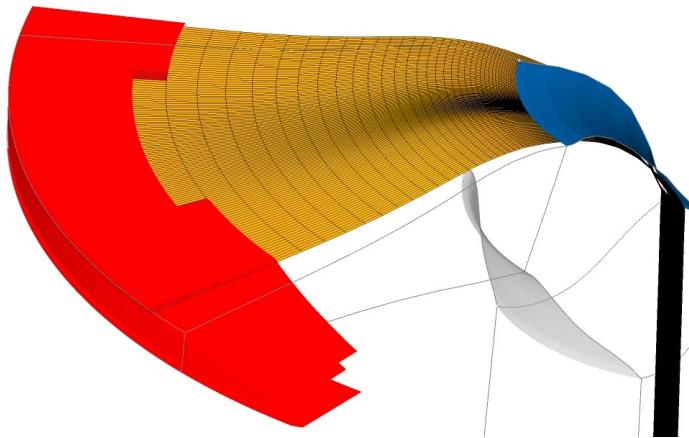
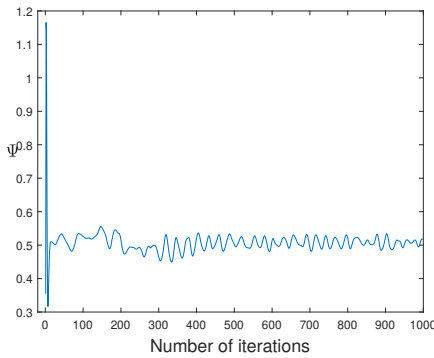
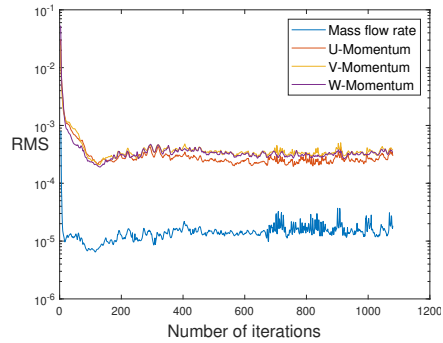


Figure 3.8: The significant amount of skewness of the mesh in the outlet domain.



(a) Iterative convergence of the head coefficient for the case $q^* = 1$, Mesh refinement factor=1.7



(b) The RMS values of the residuals

Figure 3.9: The monitored values of the head factor and the residuals for a simulation with $q^* = 1$ and mesh refinement factor of 1.7

less of a problem close to walls, as the flow changes much more rapidly perpendicular to the wall than tangential to it[2, chapter 6.3.3]. That means that the flow in the streamwise direction does not have to be as fine as the mesh perpendicular to the flow.

3.4 CFD Results

As mentioned in section 3.3.1, the designed blade was simulated for three different volume flow rates, $q^* = 0.7$, $q^* = 1$ and $q^* = 1.3$. Complete convergence was never reached for any of the cases, although multiple attempts have been made to improve the convergence of the simulations. The attempts which led to the most converged results are discussed in the remainder of this chapter. The less successful attempts and their results are described in appendix F.

3.4.1 Iterative convergence of the cases

For a CFD case to be considered converged, all relevant properties should remain approximately constant from one iteration to the other. Low residual values are also typically a good sign for convergence.

The residuals of the simulated cases tended to decrease for a while, before oscillating around one level. Similarly the monitored variables tended to converge to a level around which they then oscillated. These oscillations appear to be due to numerical instability rather than transient effects, as the frequency of the oscillations changes when the time scale of the steady-state simulation is changed [9, chapter 1.2.1]. The monitored head values and RMS residual values of a typical simulation are shown in figures 3.9a and 3.9b. While the values of the residuals varied slightly from the shown case, none stabilized at a level lower than 1×10^{-4} for the momentum and 1×10^{-5} for the RMS values of the mass flow.

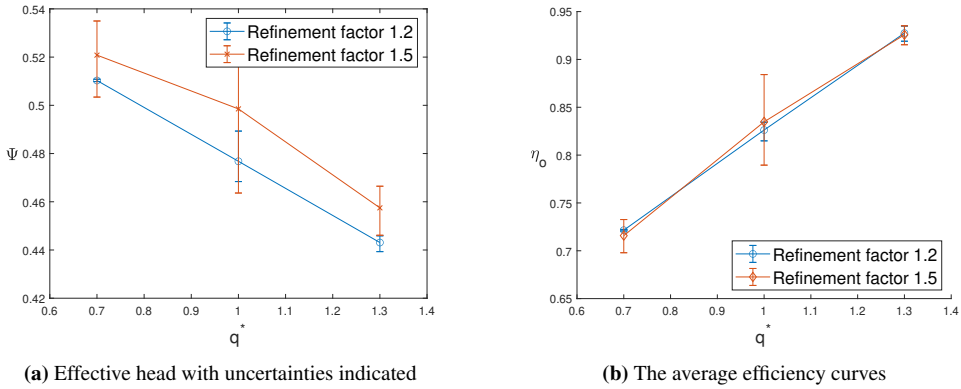


Figure 3.10: The characteristic and overall efficiency for different mesh refinement factors

In order to produce approximate values regardless of the non-convergence of the monitors, average values as well as maximal deviations are used. The values are found by visually inspecting each monitor curve, finding the iteration number at which it converges and using the head values from that point and to the end of the simulation in order to find average, maximum and minimum values.

The characteristic resulting from the average values of the head monitors is shown in figure 3.10, together with the overall efficiency.

The equation used to find the efficiency is a combination of equation (2.6) and (2.24):

$$\eta_o = \frac{\rho g H_E Q}{M \omega} \quad (3.5)$$

From the characteristic it seems like the curve does not have any instabilities in the domain $q^* = 0.7$ to $q^* = 1.3$. It also seems like the overall efficiency is increasing with increasing volume flow rate. This means that the BEP is at a larger location than the design point. While the head factor does not reach the required head at any of the simulated volume flow rates, the efficiency is still quite large. This is because the efficiency used in the calculation of the turbine geometry assumes that the velocities in the euler equation are actually reached, and thus has an other input power than the simulation does. In other words, while the pump produces a lower head than expected, it also requires less power than expected, and therefore gets a reasonable good efficiency. Another reason for the good overall efficiency of the pump may be due to errors in the loss models for the guide vanes and spiral casing. After all, the loss models are based on pumps which are initially designed as pumps instead of turbines.

3.4.2 Mesh refinement

The mesh is refined once for the off-design conditions, and twice for the design conditions. The number of grid cells, average head coefficients, max and min values as well as the change from the baseline case are shown in table 3.2.

Frame change	refinement factor	Mean Ψ	Amplitude of Ψ	Ψ/Ψ_{ref}
Frozen	1.2	0.4767	0.0209	1
Rotor	1.5	0.4985	0.0706	1.0457
	1.7	0.5036	0.0660	1.1562
Mixing plane	1.2	0.4447	0.0014	0.9329
	1.7	0.55115	0.0094	1.1562

Table 3.2: Results with mesh refinement

It is clear from the values that the simulation is mesh dependent. Firstly the average values of each simulation change significantly with an increasingly fine mesh. Secondly, the difference between the max and min values generally increases with increasing mesh refinement. The cases also tend to be more converged when mixing plane is used as stage interface, in stead of frozen rotor because the large residual values originate from the leading and trailing edge of the blade which is close to the rotor-stator interface. This will be shown in section 3.4.8. When the flow is averaged at the interfaces, the conditions leading to oscillations are smeared out as well, which leads to smaller variations of the monitors. While the mixing plane is a much more promising approach than the frozen rotor approach, it diverged when applied to the two other volume flow rates.

As the simulations do not converge, are mesh dependent and there is no actual geometry to verify the results against, all results obtained are unreliable and most likely wrong. With that in mind, the results will still be examined and discussed in order to make suggestions for both the simulations and the blade design for further work with the project. The next chapters will focus on inspecting the flow conditions through the runner and on discussing the observations.

3.4.3 Separation

The figures of 3.11 and 3.12 show the streamlines along the blade at constant spans. Unless otherwise specified, all of the figures in the rest of the chapter are for a volume flow rate of $q^* = 1$ and a mesh refinement factor of 1.7 (The finest mesh which was simulated). As expected the flow separates on the suction side of the blade. This can be seen clearly in figure 3.12. The separation leads to large slip angles and cavitating flow, as will be discussed in the next two chapters. There is one other interesting property to be seen in the streamlines. The flow at low span, i.e. close to the runner hub, has a stagnation point close to the trailing edge of the blade's pressure side. This appears because the blade angle at the outlet is quite large. This sharp curvature means that the flow does not get deflected smoothly enough, and meets the rear end of the blade perpendicularly. The effect is enhanced by the separation taking place on the suction side, which deflects the flow towards the pressure side. This stagnation point makes the blade behave more like a bluff body⁶. Bluff bodies can cause vortex shedding⁷, which could explain the oscillations in the simulations.

⁶Bluff bodies are objects which are not streamlined and thus do not create lift, but rather obstruct the flow

⁷The body creates circulations in the flow, which are transported downstream.

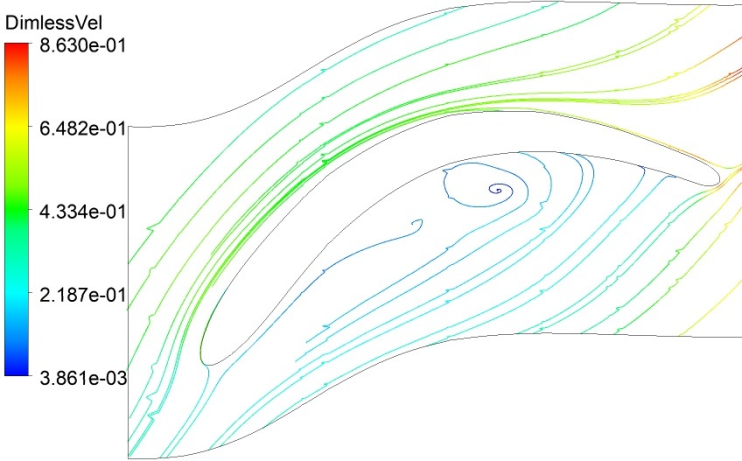


Figure 3.11: Span=0.1

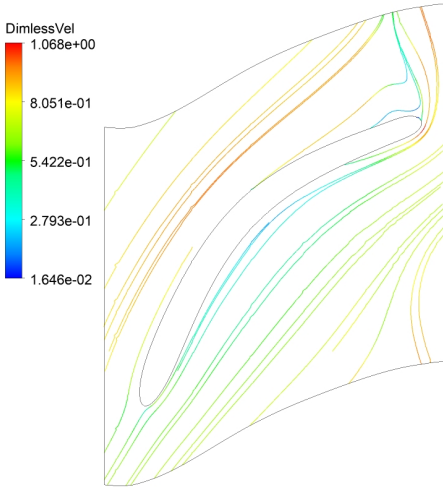


Figure 3.12: Span=0.5

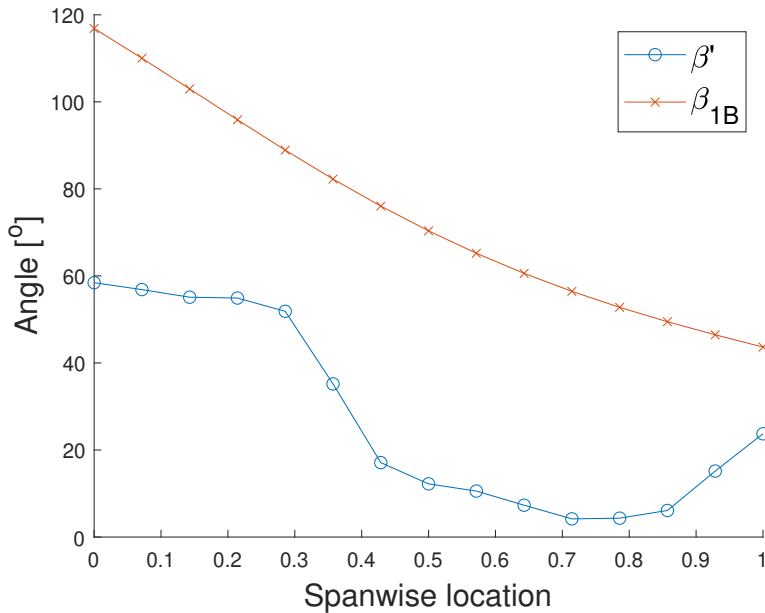


Figure 3.13: The blade outlet angles and the exiting flow angles for the case $q^* = 1$, Mesh refinement factor=1.2.

3.4.4 Slip

The amount of slip at the runner outlet is larger than expected, and the fluid is flowing back in to the runner at parts of the outlet. The difference between the flow angle β' and the blade angle β_{1B} at the trailing edge is illustrated in figure 3.13. These angles are calculated by taking circumferentially averaged values of the angles at the trailing edge. Therefore all of the angles are positive, i.e. leaving the runner although it is obvious from the figures of 3.12 that the flow is reentering the runner at the outlet at some places, which should result in local negative angles. While figures 3.11, 3.12 and 3.13 are for the case with a mesh refinement factor of 1.7 and $q^* = 1$, average slip factors along the runner trailing edge of all frozen-rotor cases are shown in table 3.3.

Notice that the table again shows a significant lack of convergence, as the slip factors vary much both with changing q^* and changing refinement.

As explained in section 2.8, the slip factor shows the difference between the theoretical and actual circumferential velocity. The fact that most slip factors are smaller than 0 means that this difference is larger than the circumferential velocity of the runner.

The reason why the slip values get so large is that the pressure does not get any opportunity to equalize across the passage, as shown in section 3.4.6, as well as the separation which deflects the flow. Thus the outlet velocity angles are reduced and, consequently, so is the head.

q^*	Refinement factor	Average γ
0.7	1.2	-0.1303
0.7	1.5	0.2779
1	1.2	-0.418
1	1.5	-0.0221
1	1.7	-0.449
1.3	1.2	0.3487
1.3	1.5	0.347

Table 3.3: Slip factor for different cases

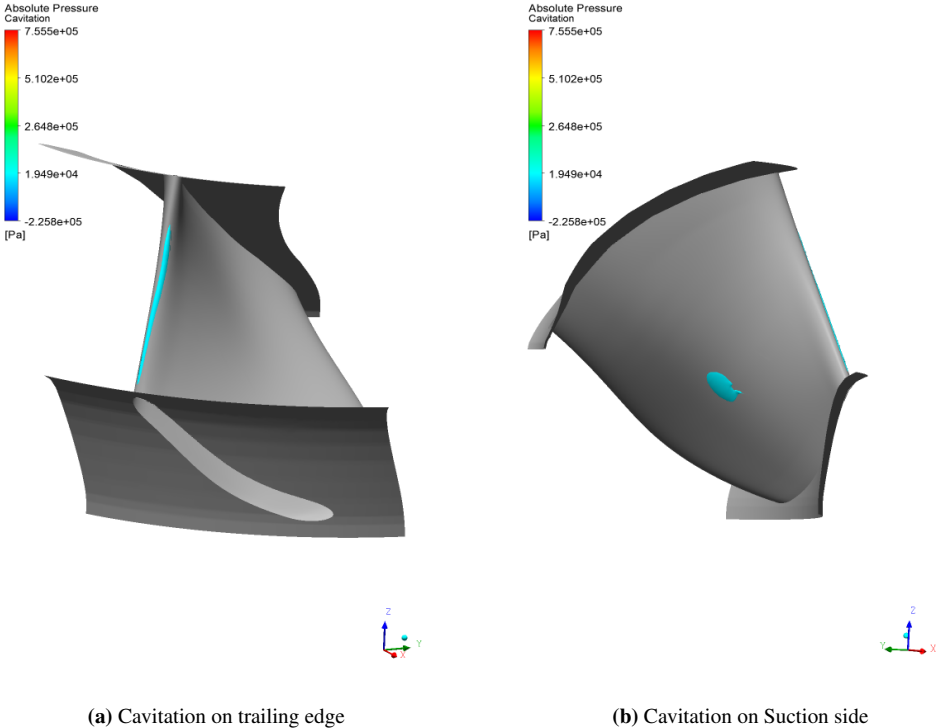


Figure 3.14: Cavitation on the blade at $q^* = 1$,

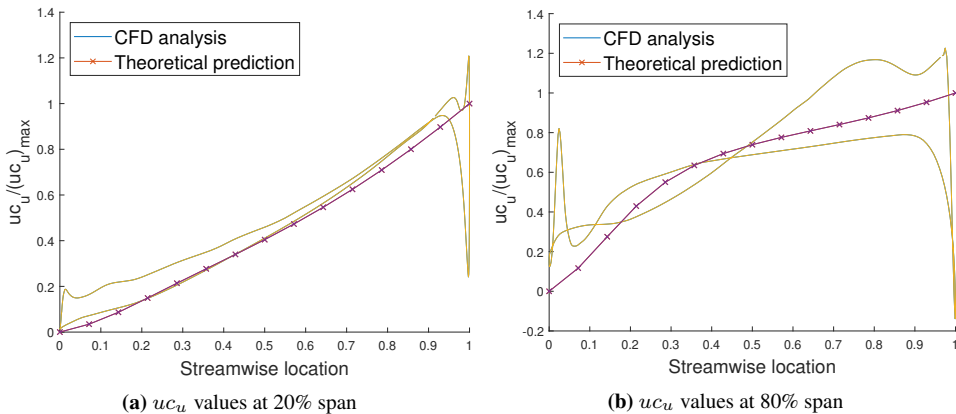


Figure 3.15: The static pressure distribution along the blade surface

3.4.5 Cavitation

The large slip factor has another disadvantage, namely an area at the trailing edge of the blade where the pressure gradient from the suction side to the pressure side creates very high velocities, accompanied by a drop in static pressure. The static pressure drops below the vaporization pressure of water and causes cavitation. The size of the cavitation bubble for $q^* = 1$ is shown in figures 3.14a and 3.14b. The amount of cavitation increases with increasing volume flow rate. This is as expected as the static pressure decreases with increasing flow velocity. Cavitation bubbles also appear on the suction side of the blade. As expected, the amount of cavitation increases with increasing volume flow rate. The solver shows that the absolute pressure at this location drops below 0 Pa, which is physically impossible. Nonphysical pressure is a major moment of uncertainty which is purely numeric. It occurs because the solver used expects a single fluid, without phase change. This nonphysical behaviour could be a contributing factor for why no convergence is achieved. In order to get rid of the nonphysical pressure values, different solver settings would have to be used. This has not been attempted, as other aspects of the flow are prioritized, and the computational resources available are limited.

3.4.6 Blade loading

Figures 3.15a and 3.15b show the simulated values of the uc_u curves at the pressure and suction side of the blades at constant span, together with the theoretical values from the geometry design. Both series are nondimensionalized against the largest predicted uc_u -value. While the average of the CFD results follows the general shape of the theoretical values, the actual maximum and minimum values deviate significantly the higher the span gets, and the flatter the curve gets at the outlet. In other words, the more the theoretical results resemble the s-shaped curve discussed in chapter 2.15, the more the CFD results deviate from these values. The deviation also grows significantly close to the trailing edge, due to the large c_u values appearing there because of the slip.

The blade loading curves, shown in figure 3.16, show the static pressure on the pressure

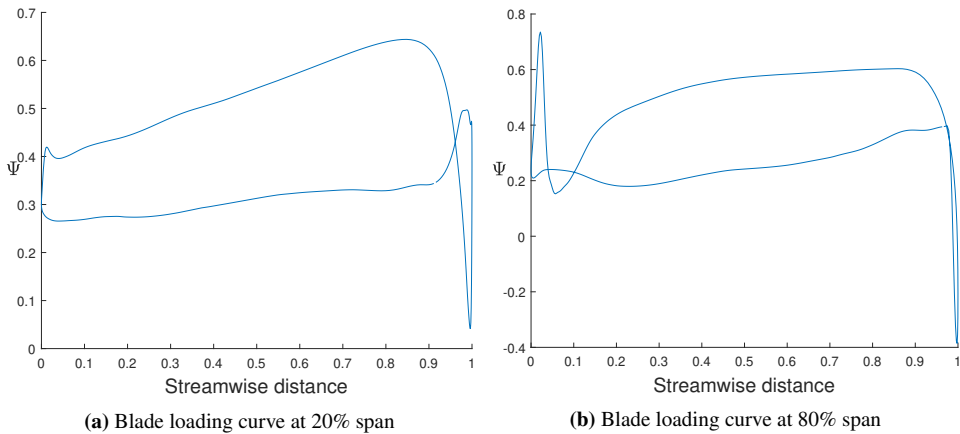


Figure 3.16: The static pressure distribution along the blade surface

side and the suction side of the blade. It can be seen that the pressure difference close to the trailing edge is large for all cases before it encounters a sudden drop for both leading edge and trailing edge. The drop occurs due to slip, as explained in section 2.8. The static pressure difference which has built up over the streamwise direction of the blade equalizes, and the static pressure is transformed to velocity in direction of the suction side. While a large pressure difference between the two sides is positive, as it creates more force and consequently torque on the blade, it should be reduced towards the trailing edge to avoid the slip. Another thing which can be seen on the blade loading curves is that the pressure on the pressure side decreases shortly after the leading edge, simultaneously as the pressure on the suction side increases. This is due to a slight incidence of the on the blade which creates a stagnation point with increased pressure on the suction side and increases the velocity on the pressure side. This is also supported by the streamlines in figure 3.12. The stagnation point being on the suction side means that the flow incoming flow velocity is larger than expected, which likely is due to the center of the inlet not being modelled. This leads to a smaller cross sectional area, which increases the flow velocity.

3.4.7 Recirculation

Figure 3.18 shows that there is flow exiting the runner at the leading edge, close to the shroud. This is recirculation as described in chapter 2.12. The location of the recirculation close to the shroud is indicated by the black area in figure 3.17. Although recirculation is perfectly normal for runners operating at lower volume flow rates than the design flow rate, recirculation at $q^* = 1$ is unexpected. This may be a sign that the pump is ill designed, and that the actual best efficiency point is at a higher q than it was designed for. This is supported by the efficiency values of figure 3.10b. Otherwise the amount of recirculation behaves as expected as it increases with decreasing volume flow rate and decreases with increasing flow rate, as illustrated in figure 3.19

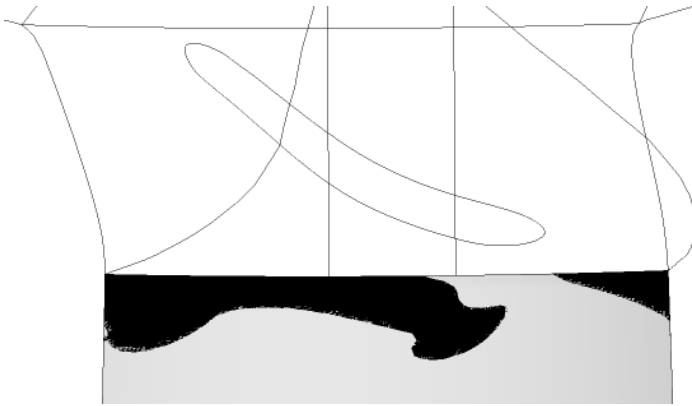


Figure 3.17: The location at the inlet shroud where recirculation occurred.

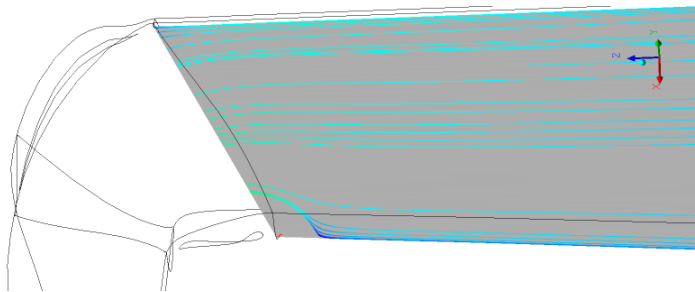


Figure 3.18: Separation at $q^* = 1$

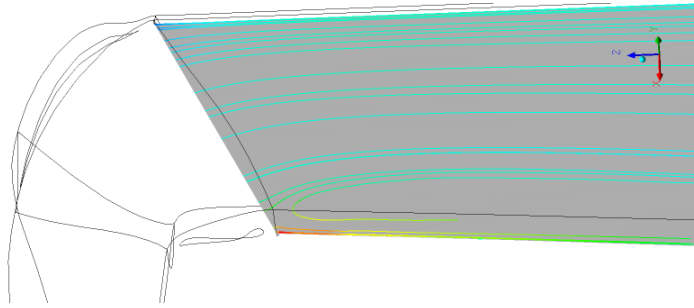


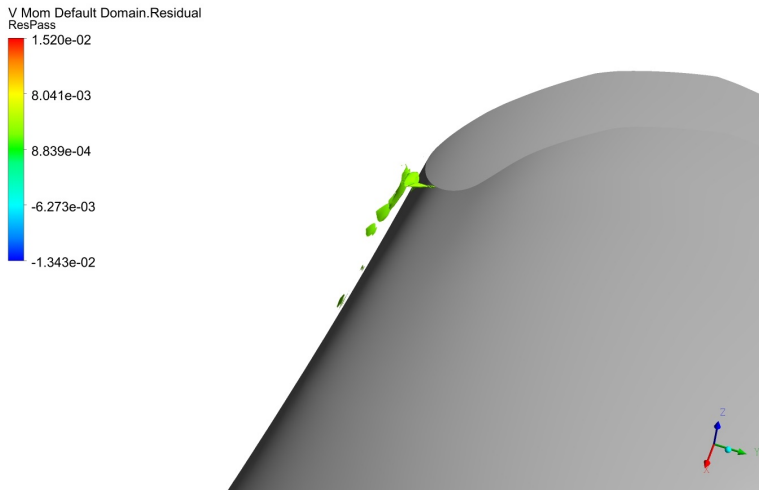
Figure 3.19: Separation at $q^* = 0.7$

3.4.8 Regions with large residual values

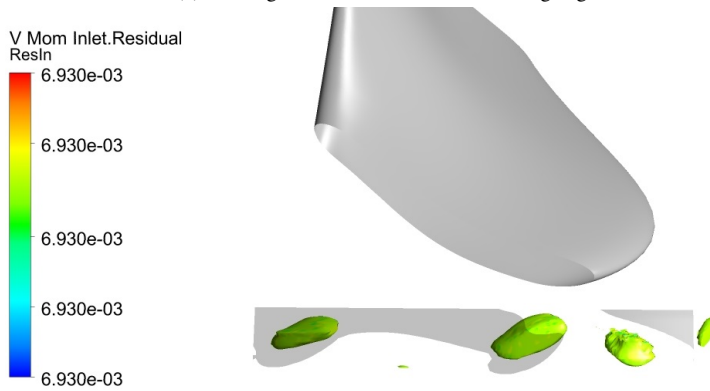
To find out which areas cause the residual values to stay large, the residual values of the solution is examined. The outlet has remarkable small maximum residual values, compared to the rest of the domain. The highest residuals for both the outlet and the passage are found close to the hub, immediately downstream of the trailing edge as seen in figure 3.20a. This supports the hypothesis that the stagnation point close to the trailing edge of the blade creates unsteadiness in the flow. The highest residuals in the inlet domain are present at the location where the incoming flow meets the recirculating flow as shown in figure 3.20b.

3.4.9 Transient simulations

In order to test whether the simulation is steady or time dependent, a transient simulation is run. The main settings of this simulation are the same as the settings of the steady state simulations. The main difference is that the frame change model has is set to transient rotor-stator interface. The results of the frozen rotor case are used as initial conditions for the transient simulation. The solution is found to be stable for a maximum Courant number of approximately 5. As the case is only meant to be indicative and due to limited resources, convergence of the simulation is sufficient and accuracy is of less concern for this simulation. The results show a large amount of fluctuations in the flow at the runner outlet, but whether these are due to actual transient behaviour or due to initial transient effects is yet uncertain. The stagnation point at low span also fluctuates back and forth along the streamwise location of the blade, but again, this could also be due to initial disturbances of the flow.



(a) The largest residual values at the trailing edge



(b) The areas with the largest residual values at the Leading edge.

Figure 3.20: The largest residual values in the inlet and the passage.

CONCLUSION

Although complete convergence was never reached for the simulations, there is much evidence that the blade design is far from optimal. The recirculation at what was supposed to be the BEP, combined with a strictly increasing efficiency curve hints at the actual BEP being at a larger volume flow rate than $q^* = 1$. The slip is perhaps the most significant deficit of the design, as it both reduces the delivered head and induces cavitation on the pressure side of the RPT.

While the simulation ideally should converge before drawing conclusion based on it, the unfavourable design of the turbine is likely to be the reason for the lack of convergence. Upon examining the residual values of the simulation, they are largest at the shroud of the inlet, where the area of recirculation appears and at the hub, downstream of the trailing edge, where the stagnation point may induce a fluctuating flow.

Therefore a runner design which does not result in recirculation at the inlet, and with a smoother blade shape which does not obstruct the flow close to the outlet will most likely result in a better blade. A runner with longer blades will likely produce better convergence, as well as give better blade performance.

The current design seems to exhibit transient behaviour, as well as cavitation. In order to get a reasonable simulation for it, a transient multiphase simulation is required.

4.1 Further work

Future projects should aim to create a more favourable geometry than the current one. In order to do this, the total length of the blade should be increased, so the pressure has the opportunity to equalize prior to reaching the outlet. The sharpness of the blades close to the hub should also be reduced further, to avoid transient behaviour. Looking into different blade loading methods than the uc_u distribution might also be useful, as the results of section 3.4.6 indicate that specification of the uc_u distribution does not yield the expected results. If convergence is not reached for a blade design, it is recommended to examine the semi-converged solution and make improvements to the blade based on the results in order to make a blade which facilitates good convergence. The philosophy should be that

a blade for which convergence can be reached will have more favourable flow conditions for a runner.

Future geometries should also be tested together with the guide vanes and spiral casing, to evaluate their effects on the performance. The entire geometry of the draft tube should also be included in the simulations. To do this, it is advisable to create the mesh for the non-blade parts in other meshing software than Turbogrid, as Turbogrid does not allow for geometries with zero radius and struggles with skewed grid cells far away from the blades.

If a geometry with sufficient head and sufficiently converged simulations is found, it should be tested as a turbine as well, to ensure that the runner has a turbine-efficiency good enough for it to be economically feasible to retrofit into a power plant.

The design should also be simulated together with the booster pump to ensure that there are no pressure pulsations or similar effects, which could harm the power plant.

Finally, a physical model of the final geometry of the RPT and the booster pump should be made and tested in a test rig, in order to validate the CFD simulations.

BIBLIOGRAPHY

- [1] Jochen Aberle et al. *Strategic Research Agenda of the EERA Joint Programme Hydropower*. URL: <https://www.eera-set.eu/component/attachments/attachments.html?task=attachment&id=264> (visited on 15/05/2020).
- [2] ANSYS. *CFX Reference guide*. (Visited on 18/06/2020).
- [3] *Ansys CFX-Solver Theory Guide*. 14.0. Southpointe, 275 Technology Drive, Canonsburg, PA 15317: Ansys, Inc., Nov. 2011.
- [4] Marshall Bern and Paul Plassmann. “Structured two-dimensional meshes”. In: *Handbook of Computational Geometry*. ISBN: 978-0-444-82537-7.
- [5] Hermod Brekke. *Pumper og Turbiner*. NTNU, 2003.
- [6] Yunus A. Çengel and John M. Cimbala. *Fluid mechanics: fundamentals and applications*. McGraw-Hill series in mechanical engineering. Boston: McGraw-Hill Higher Education, 2006. 956 pp. ISBN: 978-0-07-247236-3.
- [7] Yunus A. Çengel and John M. Cimbala. *Fluid Mechanics: fundamentals and applications, Third edition in SI units*. McGrawhill, 2014.
- [8] *CFX-Solver manual*. (Visited on 22/05/2020).
- [9] *CFX-solver modeling guide*. Ansys, Inc.
- [10] John D’Errico. *distance2curve*. Library Catalog: [se.mathworks.com](https://se.mathworks.com/matlabcentral/fileexchange/34869-distance2curve). URL: <https://se.mathworks.com/matlabcentral/fileexchange/34869-distance2curve> (visited on 22/06/2020).
- [11] John D’Errico. *interparc.m*. 13th Jan. 2020. URL: <https://www.mathworks.com/matlabcentral/fileexchange/34874-interparc> (visited on 13/01/2020).
- [12] James R. Dawson, Jason R. Hearst and Jonas Moeck. *Aerodynamics TEP4160, course notes*. 23rd Mar. 2019.
- [13] Finansdepartementet. *NOU 2019: 16*. Regjeringen.no. Library Catalog: www.regjeringen.no Publisher: [regjeringen.no](http://www.regjeringen.no). 30th Sept. 2019. URL: <https://www.regjeringen.no/no/dokumenter/nou-2019-16/id2670343/> (visited on 28/04/2020).
- [14] Johan Friedrich Gülich. *Centrifugal Pumps*. Second edition. Springer, 2010. ISBN: 978-3-642-12823-3.

-
- [15] Ioannis Hadjipaschalis, Andreas Poullikkas and Venizelos Efthimiou. “Overview of current and future energy storage technologies for electric power applications”. In: *Renewable and Sustainable Energy Reviews* 13.6 (1st Aug. 2009), pp. 1513–1522. ISSN: 1364-0321. DOI: 10.1016/j.rser.2008.09.028. URL: <http://www.sciencedirect.com/science/article/pii/S1364032108001664> (visited on 24/02/2020).
- [16] Ånund Killingtveit. “Norges ressurser/muligheter, magasiner, effekt, pumpekraft”. Norge som Europas grønne batteri – visjoner og realiteter. Holberg Terrasse kurs- og konferansesenter Stensberggaten 27, Oslo, 16th Nov. 2016. URL: <https://www.cedren.no/Portals/Cedren/3-Norges%20ressursermuligheter%20magasiner%20pumpekraft%20etc-Seminar%20gront%20batteri-16112016-Anund%20Killingtveit-NTNU.pdf> (visited on 15/05/2020).
- [17] Sira-Kvina Kraftselskap. *Roskrepp Kraftverk*. URL: <https://www.sirakvina.no/roskrepp-kraftverk/roskrepp-kraftverk-article257-921.html>.
- [18] Rune Larsen. “Pre-rotation of inlet flow for a reversible pump turbine in pump mode”. Master’s Thesis. NTNU, 2019.
- [19] Leif Lia and Ånund Killingtveit. *Småkraftforeninga*. Library Catalog: www.smakraftforeninga.no. URL: <https://www.smakraftforeninga.no/2020/02/05/stort-potensial-for-ny-vannkraft/> (visited on 23/06/2020).
- [20] Z Liu and D L Hill. “Issues Surrounding Multiple Frames of Reference Models for Turbo Compressor Applications”. In: (2000), p. 11.
- [21] F. Menter. “Zonal Two Equation k-w Turbulence Models For Aerodynamic Flows”. In: *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*. eprint: <https://arc.aiaa.org/doi/pdf/10.2514/6.1993-2906>. American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.1993-2906. URL: <https://arc.aiaa.org/doi/abs/10.2514/6.1993-2906> (visited on 01/06/2020).
- [22] Bijan Mohammadi and Olivier Pironneau. “On wall laws in CFD”. In: *Fourteenth International Conference on Numerical Methods in Fluid Dynamics*. Ed. by Suresh M. Deshpande, Shivaraj S. Desai and Roddam Narasimha. Lecture Notes in Physics. Berlin, Heidelberg: Springer, 1995, pp. 31–38. ISBN: 978-3-540-49228-3. DOI: 10.1007/3-540-59280-6_97.
- [23] Michael J Moran et al. *Principles of engineering thermodynamics: SI version*. OCLC: 1039264916. Hoboken, N.J.: John Wiley & sons, 2015. ISBN: 978-1-118-96088-2.
- [24] Lakshmi Narasimhan. *Bezier.m*. 13th Jan. 2020. URL: <https://www.mathworks.com/matlabcentral/fileexchange/33828-generalised-bezier-curve-matlab-code> (visited on 13/01/2020).
- [25] *NVE Vannkraft utbygd og ikke utbygd*. URL: <https://gis3.nve.no/link/?link=vannkraft> (visited on 26/02/2020).
- [26] *Om magasinstatistikken - NVE*. Library Catalog: www.nve.no. URL: <https://www.nve.no/energiforsyning/energiforsyningsdata/om-magasinstatistik> (visited on 15/05/2020).
-

-
- [27] Fernando Perán and Ademir Suárez. *Transformation of conventional hydro into pumped-storage: an new future business line for the electricity sector*. Published: Presented at Hydro 2019. 2019.
- [28] Hartmut Prautzsch, Wolfgang Boehm and Marco Paluszny. *Bézier and B-spline techniques*. OCLC: 1120902461. 2002. ISBN: 978-3-662-04919-8.
- [29] *Renewable energy generation*. Our World in Data. URL: <https://ourworldindata.org/grapher/modern-renewable-energy-consumption> (visited on 26/02/2020).
- [30] S.L.Dixon and C.A.Hall. *Fluid Mechanics and Thermodynamics of Turbomachinery*. Butterworth-Heinemann, 2014.
- [31] John W. Slater. *Validation Assessment*. URL: <https://www.grc.nasa.gov/WWW/wind/valid/tutorial/valassess.html> (visited on 05/05/2020).
- [32] John W. Slater. *Verification Assessment*. URL: <https://www.grc.nasa.gov/WWW/wind/valid/tutorial/verassess.html> (visited on 05/05/2020).
- [33] Alexey J. Stepanoff. *Centrifugal and axial flow pumps*. John Wiley & Sons, inc., 1948.
- [34] Hendrik Tennekes, John Leask Lumley et al. *A first course in turbulence*. MIT press, 1972.
- [35] *Været for Roskreppfjord*. yr.no. Library Catalog: www.yr.no. URL: <https://www.yr.no/nb/historikk/graf/1-2605761/Norge/Agder/Sirdal/Roskreppfjord?q=siste-13-m%C3%A5neder> (visited on 16/06/2020).
- [36] John Valstad. “Simulation of a booster pump and a reversible pump turbine in series”. Master’s Thesis. NTNU, 2018.
- [37] Andrés Vargas-Serrano et al. “Economic benefit analysis of retrofitting a fixed-speed pumped storage hydropower plant with an adjustable-speed machine”. In: *2017 IEEE Manchester PowerTech*. 2017 IEEE Manchester PowerTech. ISSN: null. June 2017, pp. 1–6. DOI: 10.1109/PTC.2017.7981008.
- [38] H. K. Versteeg and W. Malalasekera. *An introduction to Computational Fluid Dynamics, The Finite Volume Method*. 1st. Essex CM0 2JE, England: Longman Scientific & Technical, 1995. ISBN: 0-582-21884-5.
- [39] Zhao Wei et al. “EP 8407 High Pressure Hydraulic Machinery Autumn 2009”.
- [40] *What is the Difference Between Centrifugal & Rotodynamic Pumps*. Pumps and Systems Magazine. Library Catalog: www.pumpsandsystems.com. 27th Feb. 2019. URL: <https://www.pumpsandsystems.com/what-difference-between-centrifugal-rotodynamic-pumps> (visited on 14/06/2020).
- [41] Frank M White and Isla Corfield. *Viscous fluid flow*. Vol. 3. McGraw-Hill New York, 2006.
-

-
- [42] F.D. Witherden, A. Jameson and D.W. Zingg. “The Design of Steady State Schemes for Computational Aerodynamics”. In: *Handbook of Numerical Analysis*. Vol. 18. Elsevier, 2017, pp. 303–349. ISBN: 978-0-444-63910-3. DOI: 10.1016/bs.hna.2016.11.006. URL: <https://linkinghub.elsevier.com/retrieve/pii/S1570865916300618> (visited on 21/06/2020).
- [43] C.P. Yorke and Gary.N. Coleman. “Assessment of common turbulence models for an idealised adverse pressure gradient flow”. In: *European Journal of Mechanics - B/Fluids* 23.2 (Apr. 2004). Library Catalog: reader.elsevier.com, pp. 319–337. DOI: 10.1016/j.euromechflu.2003.07.002. URL: <https://www.sciencedirect.com/science/article/pii/S0997754603001195#!> (visited on 05/05/2020).

Appendices

DESCRIPTION OF THE MASTERS WORK

MASTER WORK

for
student Jan-Karl Lasse Escher

Spring 2020

*“Design of a reversible pump turbine”
“Design av reversible pumpeturbin”*

Background

Norway has 50 % of the European hydro reservoir energy storage, and many of these sites are highly suitable for retrofitting of pumped-storage capabilities. To be able to reuse existing power plants by retrofitting with Reversible Pump Turbines (RPTs), the problem of cavitation in pumping mode must be solved. Assuming this to be solved by an axial booster pump (on-going parallel work), the challenge of designing an RPT suited for the operation within limitations on dimensions imposed by the existing turbine unit existing must be resolved

Objective

The candidate shall develop a new RPT design to fit in the existing turbine arrangement in Roskrepp hydropower plant for the relevant operational conditions at Roskrepp.

The following tasks are to be considered:

1. Literature review of reversible pump-turbines and design philosophy
2. Decide on a philosophy for designing the RPT and develop a numerical model of this
3. Perform numerical investigations using CFD on the numerical model obtained with the objective of estimating the performance characteristics.
4. If the student will go to Nepal for an excursion, earlier and further work will be presented as a publication and presented at the conference; 10th *International symposium on Current Research in Hydropower Technologies (CRHT-IX)* at Kathmandu University

-- “ --

The master work comprises 30 ECTS credits.

The work shall be edited as a scientific report, including a table of contents, a summary in Norwegian, conclusion, an index of literature etc. When writing the report, the candidate must emphasise a clearly arranged and well-written text. To facilitate the reading of the report, it is important that references for corresponding text, tables and figures are clearly stated both places.

By the evaluation of the work the following will be greatly emphasised: The results should be thoroughly treated, presented in clearly arranged tables and/or graphics and discussed in detail.

The candidate is responsible for keeping contact with the subject teacher and teaching supervisors.

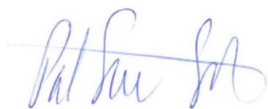
Risk assessment of the candidate's work shall be carried out according to the department's procedures. The risk assessment must be documented and included as part of the final report. Events related to the candidate's work adversely affecting the health, safety or security, must be documented and included as part of the final report. If the documentation on risk assessment represents a large number of pages, the full version is to be submitted electronically to the supervisor and an excerpt is included in the report.

According to "Utfyllende regler til studieforskriften for teknologistudiet/sivilingeniørstudiet ved NTNU" § 20, the Department of Energy and Process Engineering reserves all rights to use the results and data for lectures, research and future publications.

Submission deadline: To be found in Inpera.

- Work to be done in lab (Water power lab, Fluids engineering lab, Thermal engineering lab)
 Field work

Department for Energy and Process Engineering 4/1 2020



Pål-Tore Storli
Supervisor

Co-Supervisor(s): Helene Njølstad Dagsvik

PAPER FOR CRHT-X

The following is the work handed in for the 10th annual symposium on Current Research in Hydropower Technologies, CRHT-X.

Design of a reversible pump turbine

J K L Escher, H N Dagsvik, P T S Storli

Waterpower laboratory, Alfred Getz' vei 4, 7491 Trondheim, Norway

E-mail: jkescher@stud.ntnu.no, helene.n.dagsvik@ntnu.no, pal-tore.storli@ntnu.no

Abstract. The design of reversible pump turbines for the purpose of retrofitting existing hydropowerplants has been investigated. A preliminary design for further analysis with computational fluid dynamics has been made. As this is current work, the project has not yet been finished, and the author does not yet know if the resulting pump turbine has good efficiency or not.

1. Introduction

In recent years, Europe as a whole has started to rely more and more heavily on renewable energy sources like wind and solar power, as shown in figure 1. The drawback with these sources

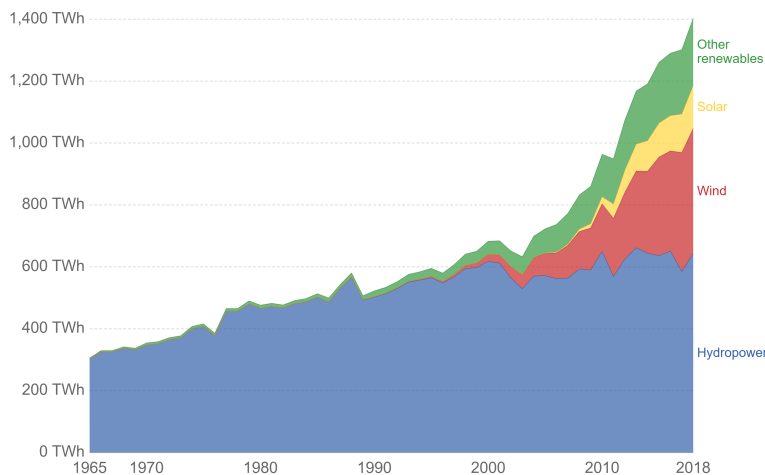


Figure 1. Amount of renewable energy sources in Europe, collected from [1]

is that they are directly influenced by the weather, and thus cannot be used on demand. This necessitates good techniques for energy storage. The most economical technology for storing energy is the application of pumped storage technology[2], where excess energy is taken from the grid and used to pump water to a higher reservoir. When there is a demand for more energy

in the grid the same water is run through a turbine to regenerate the energy. However, to be able to use pumped storage technology, one needs a reservoir pair and a height difference.

Norway has very good conditions for the application of pumped storage technologies. The country already has approximately 50% of Europes hydropower reserves and has utilized water as its primary energy source for a long time, as shown in figure 2. Therefore most of the

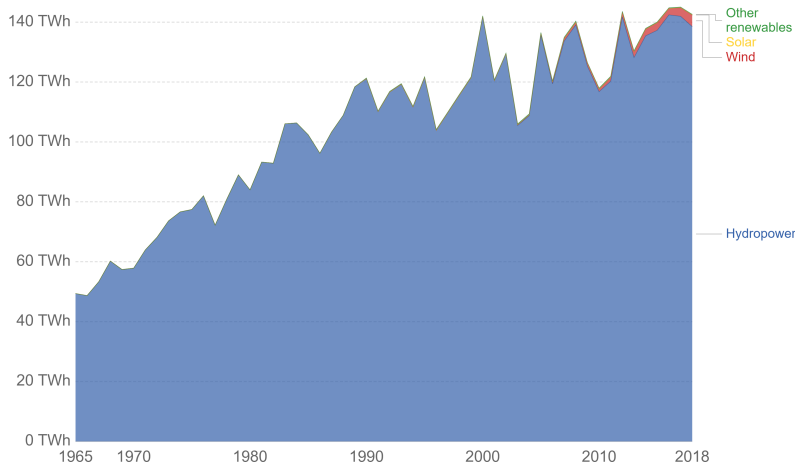


Figure 2. Amount of renewable energy sources in Norway, collected from [1]

available hydropower resources are either already utilized as regular hydropower plants or are protected due to environmental concerns. There are only 9 powerplants with pumping capability in Norway[3]. Therefore there is room for more pumped storage powerplants in the Norwegian grid.

One idea to make new pumped storage hydropower plants at low costs is to reuse existing powerplants and retrofit them with reversible pump turbines (RPTs) which are to function both as pumps and as turbines.

There are some problems connected to this approach. One is that pumps need larger submergence than turbines to avoid the fluid from evaporating due to low-pressure zones, so-called cavitation. A possible solution for this may be to insert a booster pump downstream of the RPT. Booster pump technology is currently under development in a project parallel to this one.

The other problem is that a Francis turbine requires a lower radius to operate at best point efficiency (BEP) for a given upstream water height than a radial pump requires to pump water up to the same height, with the same volumetric flow rate. Additionally, there is a pressure height difference created by the friction in the penstock which reduces the pressure height experienced by a turbine and increases the pressure height a pump has to deliver.

In this paper, Roskrepp powerplant, owned by the Sira-Kvina Power company will be used as a baseline for the design of an RPT for retrofitting. The turbine will be designed for $H = 59$ m. This is not the maximum difference between the reservoirs but assumed to be a reasonable height for the pump to operate in. While the maximum volumetric flow rate of the turbine is $Q = 70 \text{ m}^3 \text{ s}^{-1}$, the actual design flow rate is $Q = 50 \text{ m}^3 \text{ s}^{-1}$. The current rotational velocity is 250 rpm.

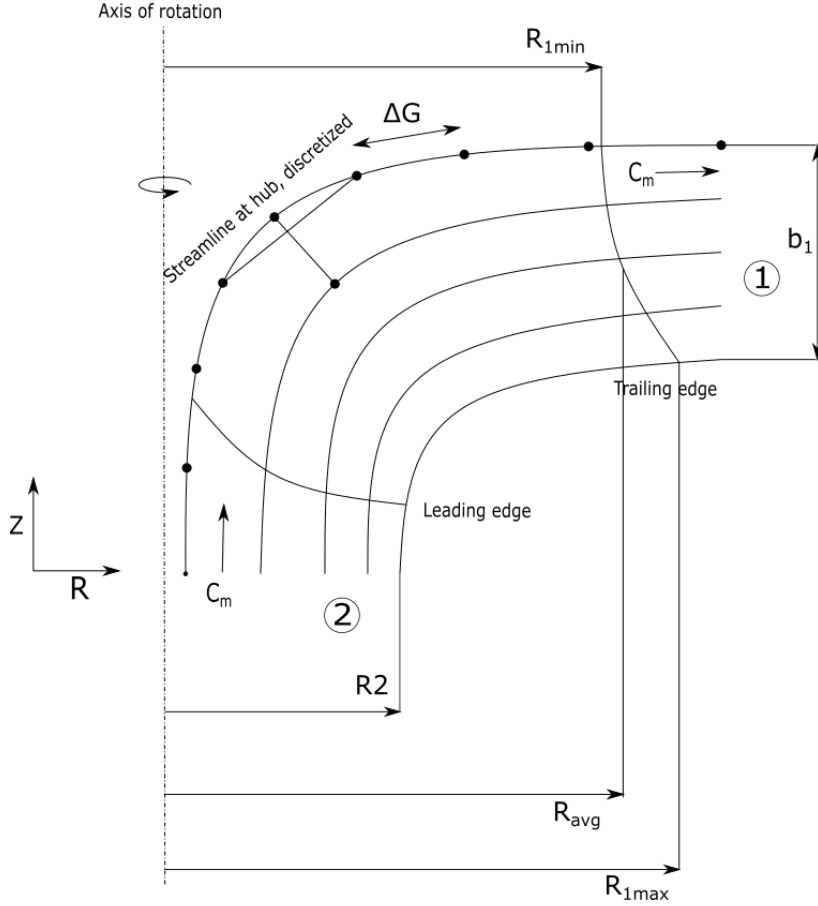


Figure 3. Projection of the R-Z plane in a radial turbomachine

2. Theory and methodology

In the remainder of this text, the subscripts related to location in the RPT will be as shown in figure 3. Inlet and outlet will be the inlet and outlet in pump mode, so 2 and 1, respectively.

2.1. Fundamental idea

In the context of turbomachines, pressure is usually called head and is given in meters of water column [mWc]. The lifting head of a pump is given by the Euler equation for turbomachines:

$$H_{th} = \frac{u_1 c_{u1} - u_2 c_{u2}}{g} \quad [\text{mWc}] \quad (1)$$

where g is the gravitational acceleration, H_{th} is the theoretical possible pumping head, u is the peripheral velocity of the runner and c_u is the peripheral velocity of the fluid[4]. A similar equation can be used to find the power generated by a Francis turbine.

The main idea when designing an RPT is as follows: A radial pump can function as a Francis turbine when put in reverse. While the power production of a pump put in reverse will be

sub-optimal, it will still produce some power. A Francis turbine will also work as a pump if it is reversed, but there is nothing to guarantee that it will produce the required amount of head. For this reason, an RPT will always be designed as a pump, to ensure that it behaves satisfactory both in pump and turbine mode.

Because the main dimensions of the RPT are predefined by the current arrangement, these cannot be freely designed and optimised as one would do if one were to design an entirely new powerplant. If the booster pump works very well, it may make up for the lack of head on behalf of the RPT, but as it is still under development, the booster pump performance is unknown.

The other possibility to increase the head of the pump is the internal design of the runner, more precisely the design of the runner vanes.

By application of trigonometry to the velocity-diagram at the outlet and the assumption of inlet flow without swirl, equation (1) may be rewritten to

$$H_{th} = \frac{u_1^2}{g} - \frac{u_1 c_{m1}}{g \tan \beta_1} \quad [\text{mWc}] \quad (2)$$

where c_m is the meridional velocity component of the flow and β_1 is the angle of the blade relative to the radial direction.

A graph relating the volumetric flow rate and the head of a pump is commonly called a pump characteristic. as seen in equation (2) is particularly important for the shape of the characteristic, as it is related to the slope and the lifting height. The inlet angles also play an important role, to make the swirl-free inlet flow shockless.

The peripheral outlet velocity is also influenced by the phenomenon of slip, which is deflection of the outgoing flow due to pressure differences between the suction side and the pressure side of the blade, as illustrated in figure 4. Slip is only a relevant phenomenon in pump mode. During turbine operation, all pressure differences between the pressure side and the suction side should be transferred to the runner in the form of energy, and so the outgoing flow should have uniform pressure distribution. Another aspect influencing the pumping head is the hydraulic efficiency,

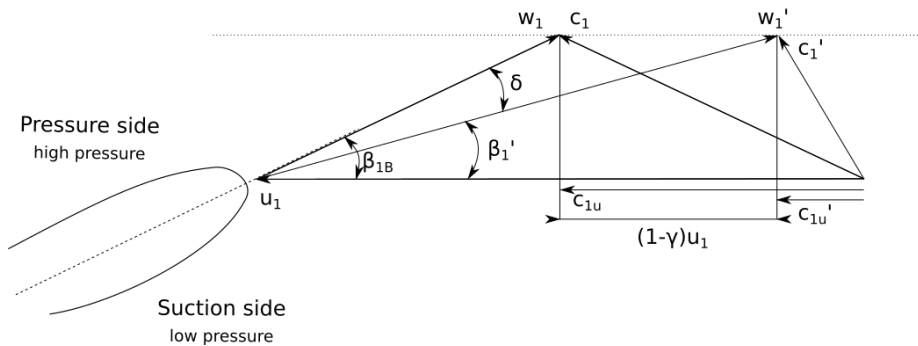


Figure 4. illustration of slip at the outlet of a pump, illustrated by use of velocity diagrams

η_h . Hydraulic efficiency is the ratio of the energy which may be utilized to the energy input to the pump[5]. It can be written in terms of head as:

$$\eta_h = \frac{H}{H_{th}} \quad [-] \quad (3)$$

where H is the actual lifting head and H_{th} is the theoretical possible lifting head, assuming no hydraulic losses between the suction and discharge nozzle of the pump. Hydraulic losses are losses related to skin friction and turbulent dissipation[6],[7]. Some sources include the slip in the models for slip factor, as incidence losses[8], while others use it as a separate value[5]. The approach utilized in this paper applies a model for a slip factor, γ , to estimate the slip. γ is shown in figure 4.

There are also hydraulic losses in the penstock of the powerplant. Those are not incorporated in the hydraulic efficiency of the pump but are added to the system curve of the powerplant. The system curve is the pump head as observed by the turbine at any volume flow rate, Q . Thus the lifting height the pump has to overcome at the design conditions, $H_{required}$, is actually given by:

$$H_{required} = \Delta H_{reservoir} + H_{loss}(Q) \quad [\text{mWc}] \quad (4)$$

where $\Delta H_{reservoir}$ is the difference in height between the upper and the lower reservoir at the design point, and $H_{loss}(Q)$ is the head loss in the penstock as a function of Q . $H(Q)$ has previously been measured for the current powerplant.

2.2. Design

In the following, a process for the design of the blade is outlined[9].

One starts by designing the radial projection of the turbine, which is one blade in the Z - R plane, as shown in figure 3. The outlet height is defined as b_1 , and is given by the existing geometry, the same goes for the outlet radius, R_1 . This is used to define the projection of the streamline along the hub. Initially, the hub is chosen to be shaped like part of an ellipse. then the shape of the shroud is found by defining a number of streamlines, starting at the outlet, and extending them to the inlet by using the hub as a reference point. When one does this, one utilizes that the mass flow rate between two streamlines is always constant[10]. For water, which is incompressible, this translates to a constant volume flow rate between the streamlines.

In regular pumps and turbines, one usually designs for the flow to accelerate through the runner. This is to avoid separation in the flow due to adverse pressure gradients, so due to flow which flows from a low-pressure area to a high-pressure area. Separation leads to stall in a radial turbine, in the same way as it would lead to stall in regular airfoils.

In an RPT, however, the fluid has to flow in both directions. Therefore there can be no significant acceleration of the flow through the runner in either direction, as this would lead to retardation in the opposite direction. For a flow with constant volume flow rate, and constant meridional velocity, the meridional area between the streamlines has to be constant along each streamline. So:

$$c_{m1} = c_{m2} = c_m \quad \text{m s}^{-1} \quad (5)$$

$$\Rightarrow A_{m1} = A_{m2} = A_m \quad \text{m}^2 \quad (6)$$

where A_m is the meridional area. The new streamlines can, therefore, be found by discretizing the initial streamline along the hub and defining the start of all the other streamlines along the outlet. Then one requires the meridional area between the streamlines to be constant and thus the distance between two streamlines at any given point can be found.

When the streamlines are found, the blades trailing edge (TE) and leading edge (LE) can be defined. These are also projected in the R - Z plane. The shape of these are very much up to the designer, and are initially chosen to be in the form of two Bezier curves, see figure 3

Now that the shape of the blade in the R - Z plane is fully designed, the distribution of the angles throughout the blade may be defined.

First of all, the inlet and outlet angles have to be found. This is done by using a model for the hydraulic efficiency and applying equations (2) and (3) to find a suitable value for the outlet blade angle.

Requirements for the blade angle are that the desired pump head is achieved at the desired volume flow rate and that the pump characteristic is as steep as possible. Steepness is desired to be able to operate at many different head values while changing the volume flow as little as possible. This is illustrated in figure 5. Because the steepness of the pump characteristic

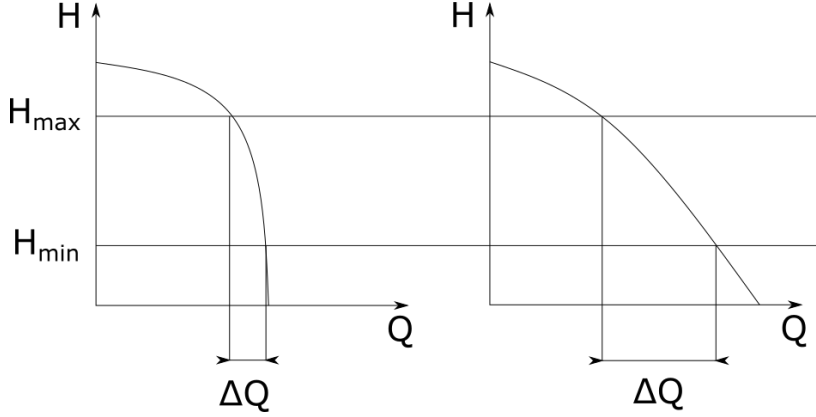


Figure 5. Relation between steepness of pump characteristics, and change in Q to obtain the desired value for H .

decreases with increasing blade outlet angle, β_{1B} , the desired outlet blade angle will always be the lowest angle which provides sufficient head. When finding β_{1B} one uses values at an average outlet radius, $R_{1,avg}$. However, when the radius varies along the trailing edge, so will the blade angle. To find the corresponding blade angles for a varying blade, one uses $R_{1,avg}$ and the corresponding blade angle as reference values and applies free vortex theory to find the remainder of the angles. A free vortex is defined by

$$c_u = \frac{K}{R} \quad [\text{m s}^{-1}] \quad (7)$$

where K is a constant.

The values for the blade angles can be found by

$$\beta_B = \arctan\left(\frac{c_m}{u - c_u}\right) \quad [^\circ] \quad (8)$$

Due to the requirement of no acceleration of the flow, c_m will remain constant. u and c_u will both be dependent of the radius.

The angles at the inlet are defined by no pre-swirl in the incoming flow, i.e. $c_{u2} = 0$. The angles are found by use of equation 8 there as well.

To find the angles from LE to TE for the initial guess, the only requirement for the distribution of the angles along the blade is that they are changing smoothly from inlet to outlet.

To be able to transform the distribution of β along the blade into points in the radial plane, an additional plane is defined to simplify the transformation, namely the G - H -plane. G is the length of a streamline in the R - Z plane and H is the length of the streamline in the R - θ plane. the length of G between two discrete points along a streamline is shown as ΔG in figure 3 and

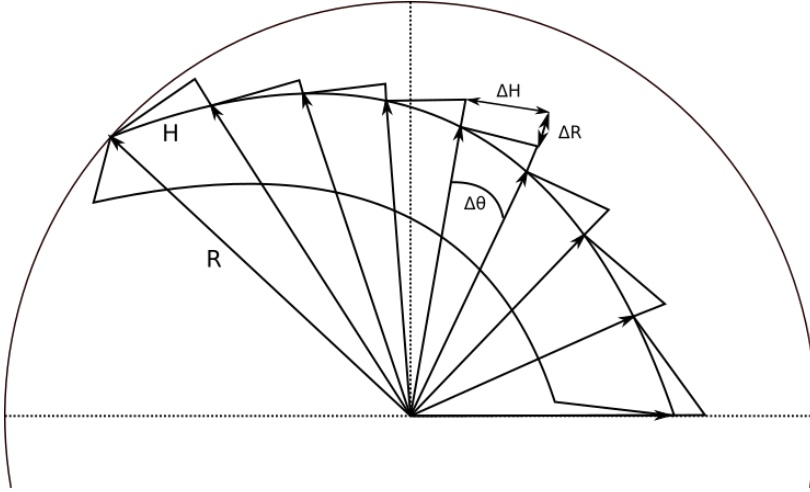


Figure 6. Schematic of the R - θ plane.

the length of H between two discrete points is shown in figure 6. The values of G can be found from the streamlines in the R - Z plane, and the values of H can be found as by using the values for β along the blade[11].

When the streamlines are defined on the G - H plane, they can be defined on the R - θ plane by using the knowledge about the values of R and θ for each point, as in figure 6. By further applying the known values of Z from the Z - R projection, the blade is modelled in three dimensions.

Now all that remains is to add thickness to the blade. This is done by adding a layer on both sides of the blade, offset normally from the surface created by the streamlines. These two layers are further connected at both the LE and the TE with an elliptical profile. While a regular pump or a regular turbine would have an elliptical-shaped LE and a sharper TE, this would be unfavourable for an RPT, as it would lead to less streamlined design in one operation mode.

3. Results

When assuming the design conditions to be $Q = 50 \text{ m}^3 \text{ s}^{-1}$, $H = 59 \text{ mWc}$ and $n = 250 \text{ rpm}$, and by following the steps outlined in the paper, the geometry for one blade was as shown in figure 7. The blade has the modelled characteristic as shown in figure 8. In the applied model the hydraulic efficiency and the slip are found separately. The distribution of β along the blade is shown in figure 9.

4. Discussion

There are many different philosophies for the design of a blade. The one outlined in this paper is a very basic one, which takes hold in theoretic, idealized design methods, and utilizes models for the slip and the hydraulic efficiency. The reason for this is that retrofitting RPTs has not been done before, so empiric data for other kinds of pump turbines may not apply to the case. Instead of empirical data, the further design of the RPT will be based on CFD analysis for optimization.

Models for slip are usually quite uncertain. Therefore it might be more reasonable to make the initial design of the turbine without accounting for slip and adjusting the outlet angle according to the results from a CFD-analysis. The same may be valid for the efficiency.

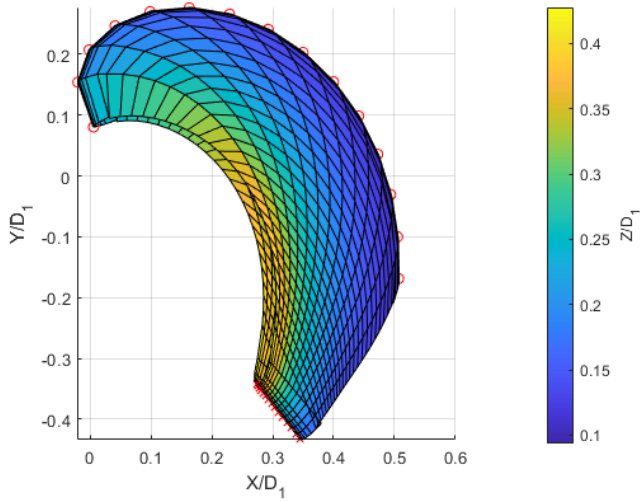


Figure 7. The geometry of one blade. The red crosses denote the trailing edge, and the red circles denote the leading edge. The values have been made dimensionless by dividing all by D_1

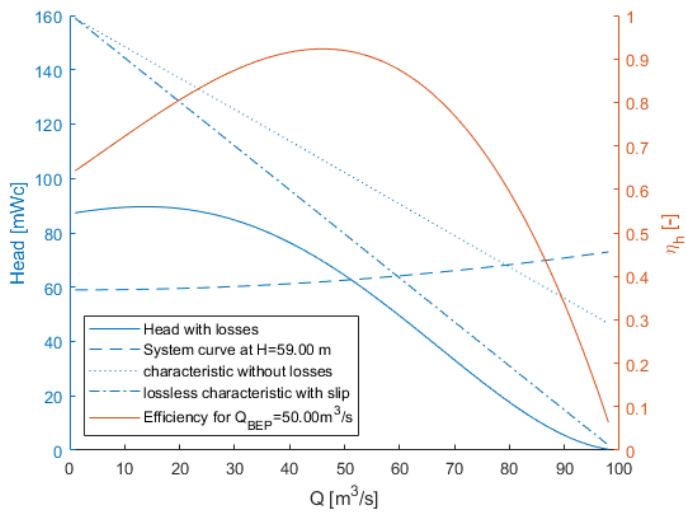


Figure 8. Theoretical pump characteristic for the current design and modelled efficiency curve

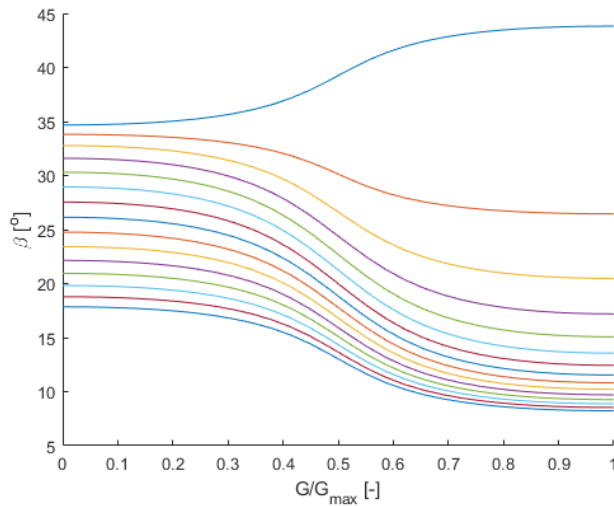


Figure 9. Distribution of the values for β along the blade, plotted against dimensionless values of G . Here G starts at the outlet and reaches G_{max} at the inlet

5. Further work

With the initial geometry of the blade ready, it remains to test it with Computational Fluid Dynamics (CFD).

Once the CFD results for the initial design are ready, the blade should be adjusted according to the results. If the blade does not produce enough head, the angle of β_1 must be increased. If the blade has pressure zones below the critical pressure, these have to be removed by smoothing out the β -distribution.

The behaviour of the RPT should also be examined at off-design conditions.

When a satisfactory design for the blade has been reached, it has to be simulated together with the booster pump, to ensure good cooperation between the two machines, both in pump mode and in turbine mode.

References

- [1] Renewable energy generation URL <https://ourworldindata.org/grapher/modern-renewable-energy-consumption>
- [2] Hadjipaschalis I, Poullikkas A and Efthimiou V 2009 *Renewable and Sustainable Energy Reviews* **13** 1513–1522 ISSN 1364-0321 URL <http://www.sciencedirect.com/science/article/pii/S1364032108001664>
- [3] NVE Vannkraft utbygd og ikke utbygd URL <https://gis3.nve.no/link/?link=vannkraft>
- [4] Brekke H, Rhrich A D and Finseraas K R 2000 *Introduction to hydraulic machinery* (NTNU)
- [5] Gülich J F 2010 *Centrifugal Pumps* second edition ed (Springer) ISBN 978-3-642-12823-3
- [6] White F M and Corfield I 2006 *Viscous fluid flow* vol 3 (McGraw-Hill New York)
- [7] Tennekes H, Lumley J L and others 1972 *A first course in turbulence* (MIT press)
- [8] Iliev I, Trivedi C and Dahlhaug O G 2018 *Journal of Physics: Conference Series* **1042** 012003 ISSN 1742-6588, 1742-6596
- [9] Wei Z, Finstad P H, Olimstad G, Walseth E and Eltvik M EP 8407high Pressure Hydraulic Machinery Autumn 2009
- [10] Çengel Y A and Cimbala J M 2014 *Fluid Mechanics: fundamentals and applications, Third edition in SI units* (McGrawhill)
- [11] Stepanoff A J 1948 *Centrifugal and axial flow pumps* (John Wiley & Sons, inc.)

DISTRIBUTION OF STREAMLINES IN THE MERIDIONAL PLANE

The streamlines are distributed by first seeding equally spaced points at the pressure side of the RPT, assuming constant volumetric flow rate between these and then extending the lines from the pressure side to the suction side. This is done in an incremental approach. First the hub is divided into a number of points. Then the following equations are found for various properties of the streamlines, as shown in figure C.1:

$$\alpha_{i,j} = \arctan \left(\frac{Z_{i-1,j} - Z_{i+1,j}}{R_{i-1,j} - R_{i+1,j}} \right) \quad (C.1)$$

$$r_{i,j} = \frac{R_{i,j} + R_{i,j+1}}{2} \quad (C.2)$$

$$b_{i,j} = \frac{R_{i,j+1} - R_{i,j}}{\sin \alpha_{i,j}} \quad (C.3)$$

$$A_{i,j} = 2\pi r_{i,j} b_{i,j} \quad (C.4)$$

$$(C.5)$$

Where R is the radial position of a point, Z is the axial position, b is the distance between two points and α is the gradient between the last and the next point of the streamline. The indices i and j denote the location along a streamline and the streamline across the passage respectively. i is unity at the pressure side and j is unity at the hub. then the equations are combined into

$$R_{i,j+1} = \sqrt{R_{i,j}^2 + \frac{A_{i,j} \sin \alpha_{i,j}}{\pi}} \quad (C.6)$$

and

$$Z_{i,j+1} = Z_{i,j} - b_{i,j} \cos \alpha_{i,j} \quad (C.7)$$

The value of $A_{i,j}$ is found by utilizing continuity and specifying the velocity along each streamline.

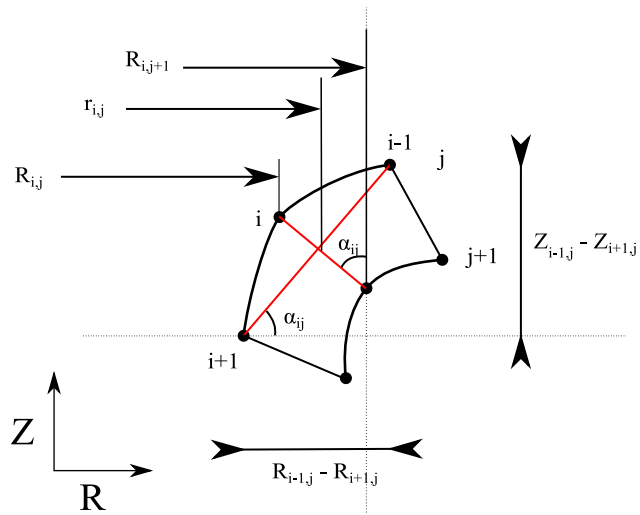


Figure C.1: Details of the iterative process to find points on the next streamline

When each point along the next streamline has been found, the points are redistributed so they are spaced equidistant along the streamline. Then the same is done for the next streamline until all points are found from the hub and down to the shroud.

G AND H CURVES

In the design process, two additional curves are defined in order to distribute the blades around the runner axis. These are the G and H curves. The G curve defines the length of one of the streamlines, and is thus given as

$$\Delta G_{i,j} = \sqrt{(\Delta R_{i,j})^2 + (\Delta Z_{i,j})^2} \quad [\text{m}] \quad (\text{D.1})$$

Where i is the point on the streamline starting from the pressure side of the runner and j is the streamline number, starting with $j = 1$ at the hub. The length of streamline G_j is thus:

$$G_j = \sum_{i=1}^{i=i_{End}} (\Delta G_{i,j}) \quad [\text{m}] \quad (\text{D.2})$$

H on the other hand defines the distance in the radial plane, so the displacement of the meridional planes in the circumferential direction. To better visualize this, inspect figure D.1 where the G and H lines of one streamline are shown on a body of rotation based on a streamline. β_B is thus related to G and H by

$$\beta_B = \frac{\Delta G}{\Delta H} \quad [\text{rad}] \quad (\text{D.3})$$

When designing the turbine, the values of β_B are decided upon first, and then the values of H are found.

H can again be related to the angular position of a point by using the known points of the radius R as shown in figure D.2, which results in values of θ :

$$\Delta\theta = \arcsin\left(\frac{\Delta H}{R}\right) \quad [\text{rad}] \quad (\text{D.4})$$

And with this the geometry is known in cylindrical coordinates. The transform from cylindrical to cartesian coordinates is straight forward:

$$\begin{aligned} X &= r \cos(\theta) \\ Y &= r \sin(\theta) \end{aligned} \quad [\text{m}] \quad (\text{D.5})$$

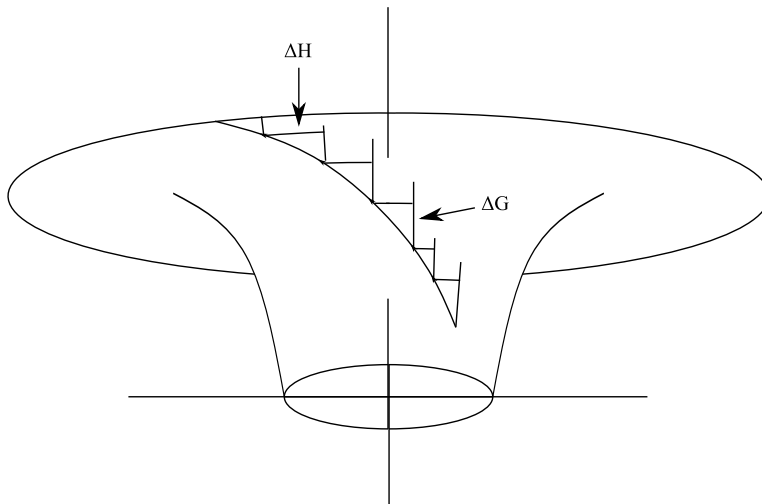


Figure D.1: G and H lines on one streamline rotated around the axis.

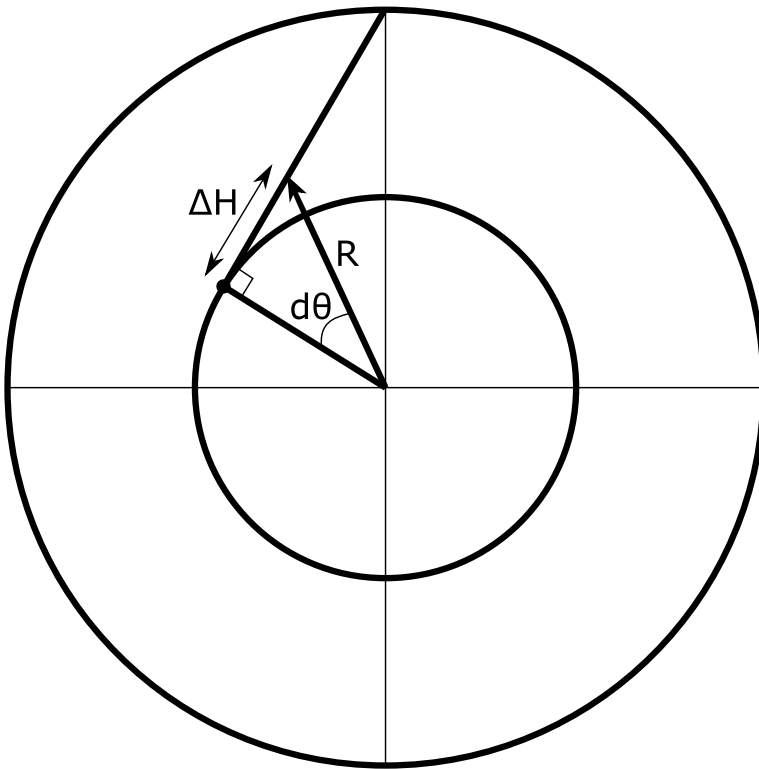


Figure D.2: Relation between θ , R and H

HANDLING OF THE LOSS MODELS FOR THE SPIRAL CASING AND THE STAY- AND GUIDE VANES

The stayvanes, guide vanes and spiral casing of the power plant are modeled by the models described in [14, page 141]. The equations will not be reproduced here, suffice to say that the dimensions required for the calculations of the losses in the guide vanes are the inlet and outlet dimensions of the passage between two guide vanes, as well as the velocity components for the flow.

In order to simplify the model, the guide vanes are assumed to be straight and flat. The resulting simplification is shown in figure E.1. $L_{1/2}$ and $L_{2/2}$ are the lengths of the guide vanes before and after the stem. r_3 and r_4 are the radiuses to the leading and trailing edges of the guide vane, and a_3 and a_4 are the distance between the leading and trailing edge. The height of the guide vanes is known and so is the distance R_{GV} to the stems. The angle α is the angle between the radial direction and the guide vane. using trigonometry, it can be shown that:

$$r_3 = \sqrt{(R - L_{1/2} \cos \alpha)^2 + (L_{1/2} \sin \alpha)^2} \quad (\text{E.1})$$

$$r_4 = \sqrt{(R + L_{2/2} \cos \alpha)^2 + (L_{2/2} \sin \alpha)^2} \quad (\text{E.2})$$

$$a_3 = 2 \cdot r_3 \sin \frac{360^\circ}{z_{Le} \cdot 2} \quad (\text{E.3})$$

$$a_4 = 2 \cdot r_4 \sin \frac{360^\circ}{z_{Le} \cdot 2} \quad (\text{E.4})$$

Which is used to find the guide vane losses. The losses in the stay vanes are found similarly, but with a larger Radius to the stems. As the geometry for the guide vanes is not found in the supplied data, it is assumed that they have equal dimensions as the guide vanes, but are fewer in number.

The losses in the spiral casing require the volumetric flow rate through it, as well as data for the geometry. The data for the geometry is found by dividing the spiral casing into parts, and assuming that the velocity in each part is constant.

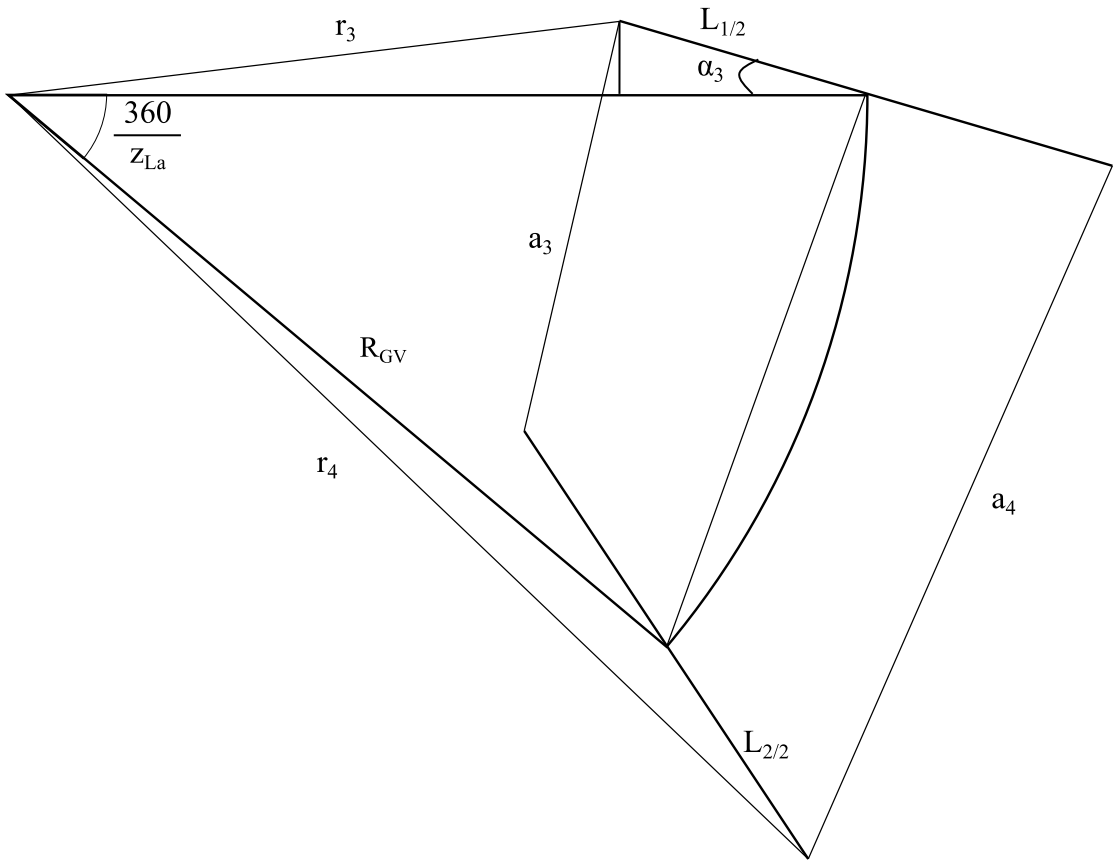


Figure E.1: Simplified geometry of guide vanes with rotation

There is also a diffuser connected to the end of the spiral casing, which leads to the penstock, which is also modelled by using the expansion from the diffuser inlet to outlet.

The scripts used to find the losses are supplied here: DiffuserData.m

```

%DiffuserData
%chemical properties
mu=1.002e-3;
rho=999.9;
roughness=0;
g=9.82;%gravitational acceleration
%The diameter at volute outlet is 2930mm
%Assuming constant velocity through the volute:
%Case specific properties
Q=51;
c2u=sum(cu(1,:))/jEllipse;%mean tangential velocity at outlet
c2m=sum(cm(1,:))/jEllipse;%meridional velocity at outlet
u2=mean(ufun(R(1,:)));%runner velocity at outlet
alpha=deg2rad(0);

%Dimensions
LGuideVane=%Length of first part of guide vane
LGuideVane2=%Length of second part of guide vane
LGuideVane1=LGuideVane-LGuideVane2;%Length of guide vane
zle=%number of guide vanes
DOutlet=%diameter of the volute at the inlet to the diffuser
D2Outlet=%diameter of the diffuser at the outlet to the penstock
LDiffuser=%length of the diffuser
RGuideVanes=%location of the guide vane stem
SpiralCasingRadii=%Radius of the outer volute wall
SpiralCasingDiameter=SpiralCasingRadii-RGuideVanes;%Diameter of
the volute
OuterSpiralCasingLengths=%length of each outer volute element

b3=%height guide vane inlet
b4=%height guide vane outlet

%Calculated dimenions
% assume the guide vanes to be flat
r3=sqrt((RGuideVanes-cos(alpha)*LGuideVane1)^2+
(sin(alpha)*LGuideVane1)^2);
r4=sqrt((RGuideVanes+cos(alpha)*LGuideVane2)^2+
(sin(alpha)*LGuideVane2)^2);

a3=2*r3*sin(2*pi/zle/2);%width guide vane inlet
a4=2*r4*sin(2*pi/zle/2);%width guide vane outlet

c_volute=4*Q/(pi*DOutlet^2);%assuming constant velocity in the

```

```

entire volute

B=(SpiralCasingDiameter(1:end-1)./2+RGuideVanesh);%Length from
    turbine centre to volute centre
C=(SpiralCasingDiameter(2:end)./2+RGuideVanesh);
a=OuterSpiralCasingLengthsh;%Renaming to make expression easier
b=SpiralCasingRadii(1:end-1);%same
c=SpiralCasingRadii(2:end);%same
A=sqrt(B.^2+C.^2+B.*C./b./c.*(a.^2-b.^2-c.^2));%finding length at
    the centre of volute, using law of cosines on two triangles
    with one shared angle

meanSpiralCasingLengthsh=A;%Renaming to make sense
meanSpiralCasingDiameterh=(SpiralCasingDiameter(1:end-1)+
    SpiralCasingDiameter(2:end))/2;%finding mean diameter of each
    element

%Losses
HDiffuser=DiffuserLossesPump(c2u,c2m,Q,a3,b3,a4,b4,LGuideVane,
    mean(u2),zle,g);%calculating losses in the guide vanesh
% HStayvanesh=%Should calculate losses in stay vanesh as well.
%%
AR=D2Outlet^2/DOutlet^2;%AR of volute outlet
fprintf('\nGuelich, figure 1.19, L/R1= %.2f,
    AR-1=%.2f\n',2*LDiffuser/DOutlet,AR-1)
cp=0.55;%input('\ninput cp:\n');%cp of volute outlet
cx=c_volute;
HVolute=voluteLossesPump(roughness,
    meanSpiralCasingLengthsh,c_volute,mu,rho
    ,meanSpiralCasingDiameterh,Q,u2,cx,cp,AR); %calculating
    losses in the volute

```

DiffuserLossesPump.m

```

%Diffusor losses pump, eqns found on p 141 guelich
function H =
    DiffuserLossesPump(c1u,c1m,Q,a3,b3,a4,b4,LGuideVane,u1,zle,g)
c2=sqrt(c1u.^2+c1m.^2);
c3q=Q./zle./a3./b3;
AR=a4*b4/a3/b3;
fprintf('Read the value of c_p for L/h=%f and A_R-1=%f in
    Guelich figure 1.18, page
    28',LGuideVane.*sqrt(pi./a3./b3),AR-1);
cp=input('\nc_p: ');
zetaLe=(c3q./u1).^2.*(0.3.*(c2./c3q-1).^2+1-cp-(1./(AR.^2)));
H=zetaLe.*u1.^2./g;
end

```

voluteLossesPump.m

```
function H =  
    voluteLossesPump(roughness,L,c,mu,rho,D,Q,u2,cx,cp,AR)  
    Area=pi.*D.*L;  
    zeta_spR=1/(Q.*u2.^2).* (sum((getFrictionFactor(roughness,L,c  
        ,mu ,rho)+0.0015).*c.^3.*Area));  
    zeta_spD=cx^2/u2^2.*(1-cp-1/AR^2);  
    H=zeta_spD+zeta_spR;  
end
```

FAILED ATTEMPTS TO REACH CONVERGENCE

Multiple attempts to reach better convergence of the simulations in this thesis, some with better success than others. The less successful attempts are listed in this appendix. Keep in mind that these results are more of a researchers log than actual results of the thesis.

Attempt: Transient solution

In case the solution should turn out to be transient, multiple transient solutions have been run with different time steps. The hope was that one solution would become periodic after a while, and the transient results of this solution would then have been used as results. One solution did become periodic after approximately 0.2 seconds simulation time, but it turned out that the frozen rotor stage model was used for this, which is a transient model, which rendered the simulation useless. Attempts with the same settings but with transient blade rows have not yielded any periodic results. Although it is possible that the simulation would turn periodic if given enough time, the time left to this thesis was not long enough to attempt this.

Increased time step factor for steady state solutions

Increasing the time step factor of steady state solutions is essentially the same as to increase the amount by which the simulation is changed in each iteration. This could potentially get rid of false oscillations, but could also remove actual transient conditions. One attempt run with a time step factor of 2 did converge, but the author has not been able to reproduce this convergence with other simulations with changed geometries, changed grid sizes or changed mass flow rate.

decreased time step factor

A decrease in the time step factor led to a decrease of the period of the oscillating monitor values. The amplitude also decreased slowly with decreasing time step factor, but a time step factor at 0.001 or lower led to divergence, most likely due to round off errors. The

lowest time step factor which was run was 0.005, but there were still significant oscillations apparent in the flow.

Refinement of the inlet and outlet areas

The significance of the inlet and outlet domains for the convergence of the simulation was evaluated by refining only these domains and leaving the passage unrefined. The result was a slightly lower oscillation amplitude in exchange for a much finer mesh. It may be worth looking more into this more, but the idea was dismissed in order to find a less computational costly solution.

MATLAB SCRIPTS

This appendix contains all of the scripts used to obtain the geometry which has been modelled in the thesis. All user defined values are given in the `getValues.m` script. In order to ensure confidentiality, the real turbine data are removed from the script. If the reader is so inclined, they may input values of a geometry of their choice, and simulate a corresponding RPT. The scripts are also added as a zip file to the handed in work. In order to shorten the length of this appendix, the scripts which are selectable by changing the options in the `Choices.m` file have been removed.

TurbinDesign2.m

```

%%TurbinDesign2
clear
close all
clc
figno=1;
%%
dir='AnsysFiles';
if ~exist(dir,'dir')
    mkdir(dir)
end
global g rho mu nu patm pvapour Values
%Choices%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Choices;                                %the values for switch
    statements
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Physical quantities
g=9.82;
patm=101325;                             %Atmospheric pressure in Pa
rho=999.9;                                %density of water
mu=1.519e-3;
nu=mu/rho;
pvapour=872.1;                            %vapourization pressure of
    water at T = 5 [deg]

```

Appendix G

```
%Geometry values
getValues;
%Arbitrarily chosen temperature: 5 degrees.
HeadDiffSubmergence=Values.HeadDiffSubmergence; %The height
    difference between the average lower reservoir height and the
    RPT
R1GuideVanes=Values.R1GuideVanes; %Radius of the
    guideVanes %g in norway
Head=Values.Head; %The design head
Q=Values.Q; %The design Volume flow rate
D1=Values.D1;%2.877;%3.36]; %Current diameter at the
    guide vanes
%D1=2*R1GuideVanes./1.05; %Rainpower, I think? D1
    \approx eq 1.05D_guidevanes
b=Values.b;
Dhub=Values.Dhub;%*.6; %hub diameter
A=pi.*D1.*b; %Area at 1, no blade blockage
D2=Values.D2; %Diameter imposed by
    existing geometry
A2=pi*(D2.^2-Dhub.^2)./4;
acceleration=A/A2;
D2m=D2; %THIS SHOULD BE CALCULATED BY
    sqrt(0.5*(D2min^2+D2max^2), Geometric diameter
e=0.019*D1; %blade thickness
    [0.016,0.022]*D1 Should be in the higher range for high heads.
n= Values.n; %current rotational
    velocity increased by one synchronous vel.
iEllipse=Values.iEllipse;
jEllipse=Values.jEllipse;
ufun=@(R) (2.*pi.*R.*n)./(60);
R2=getDeltaX2(D2,Dhub,jEllipse);
runnerheight=Values.runnerheight; %From draft tube to top of
    guide vane.
eta=getEta_h(Q,Q,n,Head); %By use
    of equations from Guelich
Routlet(1)=Values.Routlet(1);
    %Values from the supplied figure.
Zoutlet(1)=Values.Zoutlet(1);
Routlet(2)=Values.Routlet(2);
Zoutlet(2)=Values.Zoutlet(2);
RLe1=Values.RLe1;
    %The target values for The leading edge
ZLe1=Values.ZLe1;
RLe2=Values.RLe2;
ZLe2=Values.ZLe2;
% starting point for streamlines
aEllipse=R1GuideVanes; %The
    half length of the ellipsis in the x-direction
bEllipse=runnerheight; %The
    half length of the ellipsis in the y-direction
```

```

switch (Geometry)
case 'Ellipse'
    xEllipse=linspace(0,-aEllipse); %to
        -D1 etc. because are in the 2nd quadrant of ellipsis.
    yEllipse= bEllipse.*sqrt(1-(xEllipse./aEllipse).^2);
    xyEllipse=interparc(iEllipse,xEllipse,yEllipse,'spline');
    xEllipse=xyEllipse(:,1);
    xEllipse=R1GuideVanes+xEllipse;
    yEllipse=xyEllipse(:,2);
case 'Bezier'
    P1=[D1/2,runnerheight];
    P2=Values.P2;%ginput();%[0,runnerheight];
    P3=[0,0];
    bezierpoints= [P1
                   P2
                   P3];
    BezierCurve=extractBezierByPoints(0,bezierpoints);
    BezierCurve=ppval(BezierCurve,linspace(0,R1GuideVanes));
%    BezierCurve=Bezier(0,-aEllipse,bEllipse,'Plotnt');
%Creating an arbitrary chosen bezier-curve to connect the
outlet and inlet
    xyBezier=interparc(iEllipse,linspace(0,1.6),BezierCurve,
        'spline'); %Dividing it into iEllipse equally spaced
        parts
    xBezier=xyBezier(:,1);
        %Extracting data for the points as two vectors
    yBezier=xyBezier(:,2);
    xEllipse=xBezier(end:-1:1); %and
        putting them in the same form as would have been used
        for the Ellipsis
    yEllipse=yBezier(end:-1:1);
case 'BezierDefByPoints'
    P1=[D1/2,runnerheight];
        %Defining start and stop of hub curve
    P3=[0,0];
    distLim=0.001;
        %Criterium for when the curve is close enough to the
        required point at the inlet of the current geometry
    dR=0; %initial
        value for iterative value for R
    dZ=0.1*runnerheight;
        %iterative change in Z
    rPoint=Routlet(1)./2;
        %Radius for the control point for the curve
    zPoint=runnerheight; %z-value
        for the bezier control point
    if Recording
        OuterStreamlineObj=VideoWriter('outer.avi');
        OuterStreamlineObj.FrameRate=10;
        open(OuterStreamlineObj);

```

```

end
while true
    %do-while loop in matlab
    rPoint=rPoint+dR; %change
        the radius
    P2=[rPoint,zPoint]; %Define
        the control point
    bezierpoints= [P1 %vector
        with all the control points
        P2
        P3];
    BezierCurve=extractBezierByPoints(0,bezierpoints);
    BezierCurve=ppval(BezierCurve,linspace(0,R1GuideVanes));
    xyBezier=interpac(iEllipse,linspace(0,1.6),
        BezierCurve,'spline'); %Dividing it into iEllipse
        equally spaced parts
    xBezier=xyBezier(:,1);
        %Extracting data for the points as two vectors
    yBezier=xyBezier(:,2);
    xEllipse=xBezier(end:-1:1);
        %and putting them in the same form as would have been
        used for the Ellipsis
    yEllipse=yBezier(end:-1:1);
    [pt,dist,t]=distance2curve([xEllipse,yEllipse],
        [Routlet(1),Zoutlet(1)],'pchip'); %find the distance
        to the hub line and the point on the line closest to
        the required point
    if dist<distLim %If the
        distance is small enough, the solution has converged
    if yEllipse(2)>yEllipse(1) %don't
        want the stream to flow upwards again.
        zPoint=zPoint-dZ; %If it
        does, reduce Z of the point
    else
        break %else
        the solution has converged. Exit the loop
    end
end
end
if pt(1)>Routlet(1)
    dR=-1*dist;
else
    dR=1*dist;
end
figure(figno)
hold off
plot(xEllipse./Values.DimensionlessRadius,yEllipse
    ./Values.DimensionlessRadius)
hold on
xlabel('R/R_1')
ylabel('Z/R_1')

```

```

title('Streamline along the hub')
figname.hubStreamline=figno;

plot([P1(1),P2(1),P3(1)]./Values.DimensionlessRadius,
      [P1(2),P2(2),P3(2)] ./Values.DimensionlessRadius,'bo')
plot(pt(1)./Values.DimensionlessRadius,pt(2)
      ./Values.DimensionlessRadius,'rd')
plot(Routlet(1)./Values.DimensionlessRadius,Zoutlet(1)
      ./Values.DimensionlessRadius,'ks')
line([P1(1),P2(1),P3(1)]./Value.DimensionlessRadius,
      [P1(2),P2(2),P3(2)]./Values.DimensionlessRadius)
drawnow
if Recording
currFrame=getframe(gcf);
writeVideo(OuterStreamlineObj,currFrame);
end
end
if Recording
close(OuterStreamlineObj)
end
figno=figno+1;
otherwise
error('ERROR: The variable "Geometry" has no matching
value.')
```

```

end
%% Streamlines
% x2R=@(x) R1GuideVanes+x;
%Translation from x-coordinate to R-coordinate.
deltaYinlet=b/(jEllipse-1); %inlet
value spacing for all the other streamlines
deltaXoutlet=getDeltaX2(D2,Dhub,jEllipse); %outlet
value placing for all the other streamlines calculated with
same annular area at outlet
R=zeros(iEllipse,jEllipse);
%initializing matrix of R-values
Z=R;
%initializing matrix of Z-values
R(:,1)=xEllipse; %setting
values along the ring
Z(:,1)=yEllipse; %setting
values along the ring
Z(iEllipse,:)=0;
%Z-values at the outlet
R(iEllipse,:)=deltaXoutlet(1:1:end);
Z(1,:)=Z(1,1)-[0:deltaYinlet:b]; %I get
red lines, but it seems to work anyhow.
R(1,:)=D1./2;
%R-values at the inlet
```

```

alphaij=@(iv,jv,Rv,Zv) abs(atan((Zv(iv-1,jv)-
    Zv(iv+1,jv))./(Rv(iv-1,jv)-Rv(iv+1,jv)))); %Function for
    the angle alphaij
dA1=pi*D1*b/(jEllipse-1);
dA2=pi./4*(D2.^2-Dhub.^2)./(jEllipse-1);
    %The meridional area is
    constant for a RPT If acceleration is introduced, this would
    have to change be a function of the distance along the blade.
bij=@(iv,jv,Rv,Zv)
    abs((Rv(iv,jv)-Rv(iv,jv+1))/(sin(alphaij(iv,jv,Rv,Zv))));
figure(figno)
figname.meridionalProjection=figno;
figno=figno+1;
hold on
axis equal
GGeom=zeros(size(R));
%plot(R(1,:)./Values.DimensionlessRadius,
    Z(1,:)./Values.DimensionlessRadius,'o')
%plot(R(:,1)./Values.DimensionlessRadius,
    Z(:,1)./Values.DimensionlessRadius,'o')
switch Geometry
case 'BezierDefByPoints'
    dist=0; %initial
        distance between the blades
    ax=Values.ax;
        %initial x value to define the acceleration along the
        blade
    ay=Values.ay;
        %same for y
    dx=Values.dx;
        %Value to change ax and ay with for each iteration
    dy=Values.dy;
    if Recording
        shroudVideoObject=VideoWriter('shroud.avi');
        shroudVideoObject.FrameRate=10;
        open(shroudVideoObject)
    end
    while true %do-while
        P.Aij= [ %Define
            Bezier control point which defines the change in
            area along the blade
            ax ay
            ax ay
            ]; %At the
                moment it is defined by two equal points,
                because the algorithm didn't converge with
                only one point.
        % plot(R(:,1)./Values.DimensionlessRadius,Z(:,
            1)./Values.DimensionlessRadius,'-bo')
    for j=1:1:(jEllipse-1)

```



```

%jEllipse-1 that's where we find the streamline nr
jEllipse.
for i=2:1:iEllipse
    GGeom(i,j)=GGeom(i-1,
        j)+sqrt((R(i-1,j)-R(i,j)).^2+(Z(i-1,
        j)-Z(i,j)).^2);
end
for i=2:1:(iEllipse-1)
    % plot(R(i,j)./Values.DimensionlessRadius,
        Z(i,j)./Values.DimensionlessRadius,'-ro')
    R(i,j+1)=sqrt(R(i,j)^2+((getdAij(GGeom(i,j),dA1,dA2,P.Aij,
        max(GGeom(:,j)))*sin(alphaij(i,j,R,Z))/pi)));
    Z(i,j+1)=Z(i,j)-bij(i,j,R,Z)*cos(alphaij(i,
        j,R,Z));

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RZInterp=interparc(iEllipse,R(:,j+1),Z(:,j+1),
    'Spline');
R(:,j+1)=RZInterp(:,1); % To NOT distribute the
    points for every iteration, remove these three
    lines.
Z(:,j+1)=RZInterp(:,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %plot(R(:,j+1)./Values.DimensionlessRadius,Z(:,
        j+1)./Values.DimensionlessRadius,'-b') %debugging
end
ptLast=pt; %To
    check wether the line has changed the side of the
    point
[pt,dist,t]=distance2curve([R(:,jEllipse),Z(:,jEllipse)]
    ,[Routlet(2),Zoutlet(2)],'pchip'); %nearest point on
    the line, distance between the points and parametric
    variable
if sign(ptLast(2)-Zoutlet(2))~=sign(pt(2)-Zoutlet(2))
    %If the line switches which side of the point it is
    on
    dy=.5*dy; %Reduce
        the iterative increase by a factor of 0.5, to
        ensure convergence
end
if dist<distLim %If the
    line has converged sufficiently
    break %Exit
        the while loop
elseif Z(1,jEllipse)<Z(2,jEllipse) %if the
    streamline goes upward at the outlet (1)
    ax=ax+dx;
        %increase the acceleration at the outlet (1)

```

```

elseif pt(2)>Zoutlet(2)                                %If the
    point is greater than the curve
    ay=ay-dy;                                          %Need
    less steep curve for acceleration in the start
else                                                    %If the
    point is below the curve
    ay=ay+dy;                                          %Need
    more acceleration in the start, to contract curve
end
if max(((getdAij(GGeom(:,jEllipse-1), dA1,
    dA2,P.Aij,max(GGeom(:, jEllipse-1))))-dA1)./(dA2-
    dA1))>1 %if the area has a local maximum which is
    not at the inlet
    ay=ay-dy;                                          %reduce
    the area
    ax=ax-2*dx;                                       %change
    conditions and try again
end
figure(figno)
subplot(2,1,1)
hold off
plot(R./Values.DimensionlessRadius,
    Z./Values.DimensionlessRadius)
hold on
xlabel('R/R_1')
ylabel('Z/R_1')
plot(pt(1)./Values.DimensionlessRadius,
    pt(2)./Values.DimensionlessRadius,'rd')
plot(Routlet(2)./Values.DimensionlessRadius,
    Zoutlet(2)./Values.DimensionlessRadius,'ks')
subplot(2,1,2)
hold off
plot(GGeom(:,jEllipse-1)./max(GGeom(:,
    jEllipse-1)),((getdAij(GGeom(:, jEllipse-1), dA1
    ,dA2,P.Aij,max(GGeom(:,jEllipse-1))))-dA1)./(dA2
    -dA1))
hold on
xlabel('G/G_{max}')
ylabel('dA')

plot(ax,ay,'dk')
drawnow
if Recording
    currFrame=getframe(gcf);
    writeVideo(shroudVideoObject,currFrame)
end
end
if Recording
    close(shroudVideoObject)
end
end

```

```

    filename.shroudStreamline=figno;
    figno=figno+1;
otherwise
P.Aij=Values.P.Aij;
for j=1:1:(jEllipse-1)%jEllipse-1 that's where we find the
    streamline nr jEllipse.
    for i=2:1:iEllipse
        GGeom(i,j)=GGeom(i-1,
            j)+sqrt((R(i-1,j)-R(i,j)).^2+(Z(i-1,j)-Z(i,j)).^2);
    end
    for i=2:1:(iEllipse-1)
        R(i,j+1)=sqrt(R(i,j)^2+(getdAij(GGeom(i,j),dA1,dA2,
            P.Aij,max(GGeom(:,j)))*sin(alphaij(i,j,R,Z))/pi));
        Z(i,j+1)=Z(i,j)-bij(i,j,R,Z)*cos(alphaij(i,j,R,Z));
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    RZInterp=interparc(iEllipse,R(:,j+1),Z(:,j+1),'Spline');
    R(:,j+1)=RZInterp(:,1); % To NOT distribute the points for
        every iteration, remove these three lines.
    Z(:,j+1)=RZInterp(:,2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
end
dAij=getdAij(GGeom(i,j),dA1,dA2,P.Aij,max(GGeom(:,j)));
dAijMat=getdAij(GGeom,dA1,dA2,P.Aij);

GGeom(:,end)=GGeom(:,end-1)+sqrt((R(:,end)-R(:,end-1)).^2+
    (Z(:,end)-Z(:,end-1)).^2);
for i=2:1:iEllipse
    GGeom(i,jEllipse)=GGeom(i-1,jEllipse)+sqrt((R(i-1,
        jEllipse)-R(i,jEllipse)).^2+(Z(i-1,
        jEllipse)-Z(i,jEllipse)).^2);
end
figure(filename.meridionalProjection)
hold on
plot(R./Values.DimensionlessRadius,
    Z./Values.DimensionlessRadius)
axis equal
xlabel('R/R_1')
ylabel('Z/R_1')
title('Streamlines of the turbine in the R-Z plane')
axis equal
%% QH calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Angles below
%The entire design collapses if the trailing edge is too far down
the
meridional view, so don't do that.
%Cutting off the blades
%columns like this: [xvalue,yvalue;...]. decreasing x-values,
increasing

```

```

%y-values
switch Geometry
case 'BezierDefByPoints'
P.LEp=Values.P.LEp;%The points which generate the Leading edge by
    use of Bezier curves
P.TEpMid=Values.P.TEpMid;
if isempty(P.TEpMid)
    P.TEpMid=[Routlet(2)+(Routlet(1)-Routlet(2))/2
        Zoutlet(2)+(Zoutlet(1)-Zoutlet(2))/2];
end

P.thetaTE1=atan((abs(Routlet(2)-
    P.TEpMid(1)))/(abs(Zoutlet(2)-P.TEpMid(2))));
P.TEp1=[P.TEpMid(1)+ 1.05*(sqrt((Routlet(2)-
    P.TEpMid(1))^2+(Zoutlet(2)-P.TEpMid(2))^2))*sin(P.thetaTE1)
    ...
    P.TEpMid(2)-1.05*(sqrt((Routlet(2)
    -P.TEpMid(1))^2+(Zoutlet(2)-
    P.TEpMid(2))^2))*cos(P.thetaTE1)];

P.thetaTE2=atan((abs(Routlet(1)-
    P.TEpMid(1)))/(abs(Zoutlet(1)-P.TEpMid(2))));
P.TEp2=[P.TEpMid(1)- 1.05*(sqrt((Routlet(1)-
    P.TEpMid(1))^2+(Zoutlet(1)-P.TEpMid(2))^2))*sin(P.thetaTE2)
    ...
    P.TEpMid(2)+1.05*(sqrt((Routlet(1) -P.TEpMid(1))^2+
    (Zoutlet(1)-P.TEpMid(2))^2))*cos(P.thetaTE2)];

P.TEp=[
    P.TEp1
    P.TEpMid
    P.TEp2
];%The points which generate the Trailing edge by use of
    Bezier curves
otherwise
    P.LEp=Values.P.LEp;%The points which generate the Leading
        edge by use of Bezier curves
    P.TEp=Values.P.TEp;
end
RGeom=R;
ZGeom=Z;
[R,Z]=cutOffBlade(RGeom,ZGeom,P.TEp,P.LEp,iEllipse,jEllipse);
figure(figureName.meridionalProjection)
plot(R./Values.DimensionlessRadius
    ,Z./Values.DimensionlessRadius,'o')
P.TERunner=Values.P.TERunner;
P.LERunner=Values.P.LERunner;
[RRunner,ZRunner]=cutOffBlade(RGeom,
    ZGeom,P.TERunner,P.LERunner,iEllipse,jEllipse);

```

```

%% G-streamline
G=zeros(size(Z)); %Initializing the G matrix
for i=2:iEllipse
    G(i,:)=G(i-1,:)+sqrt((R(i-1,
        :)-R(i,:)).^2+(Z(i-1,:)-Z(i,:)).^2);
end
DeltaG=G(2:end,:)-G(1:end-1,:);

%Now we want to find beta1 and beta2 for all values along LE and
TE
%To find beta1 along TE, we assume free vortex theory:
    cu*r=const, where
%the reference value is found at D1.
GToAdd=getGGeomAtG0(R(1,:),Z(1,:),RGeom,ZGeom); %Finds
    the value of GGeom where G of the blade itself is 0.
cm=zeros(iEllipse,jEllipse);
for j=1:1:jEllipse
    cm(:,j)=(Q./(jEllipse-1)./getdAij((G(:
        ,j)+GToAdd(:,j)),dA1,dA2,P.Aij,GGeom(end,j)));
end
QHCalculations
switch FindBetaBlade
    case 'freeVortex'
        beta1=getBeta1B(gamma,cm1,tau_1,D1,n,beta1B,R);
            %finds values for beta1 corresponding to free vortex
            theory
        beta2=zeros(size(beta1));
            %Initializing vector
        for j=1:1:jEllipse
            beta2(j)=atan((Q./(jEllipse-1)./getdAij((G(end,j)+GToAdd(j))
                ,dA1,dA2,P.Aij,GGeom(end,j))./ufun(R(end,j))));
                %finding beta2 according to the assumed cm at the
                point in the runner.
        end
        % beta2=atan((4.*Q)./(ufun(R(iEllipse,
            :)).*pi.*(D2.^2-Dhub.^2))); %values for beta2 if the
            blades would end at the end of the geometry.
        beta1=beta1+(beta1<0).*pi;
            %if the angle gets larger than 90 degrees, trigonometry
            leads to it being 180deg smaller than intended.
            therefore we add 180deg (or pi radians) to fix this.
    case 'streamlines'
        beta2=zeros(size(beta1));
            %Initializing vector
        for j=1:1:jEllipse
            beta2(j)=atan((Q./(jEllipse-1)./getdAij((G(end,j)+
                GToAdd(j)),
                dA1,dA2,P.Aij,GGeom(end,j))./ufun(R(end,j))));%finding
                beta2 according to the assumed cm at the point in
                the runner.

```

```

    end
    otherwise
        error('ERROR, no matching value for variable FindBetaBlade')
    end
end

%% Energy distribution and finding beta along the blade
switch axialFlowAtOutlet %If the flow at the outlet has an axial
    component, the radial and the meridional flow angle are
    different.
    case 'yes'%Then the value of beta1B has to be transformed
        from the radial to the meridional plane.
        epsilonOutlet=atan((Z(1,:)-Z(2,:))./(R(1,:)-R(2,:)));
        epsilonInlet=atan((Z(iEllipse-1,
            :)-Z(iEllipse,:))./(R(iEllipse-1,:)-R(iEllipse,:)));
        beta1B=atan(tan(beta1B)./cos(mean(epsilonOutlet)));
        beta1= atan(tan(beta1)./cos(epsilonOutlet))
            +pi.*(atan(tan(beta1)./cos(mean(epsilonOutlet))) <0);
        beta2=atan(tan(beta2)./sin(epsilonInlet))
            +pi.*(atan(tan(beta2)./cos(mean(epsilonInlet))) <0);
    case 'no'
    otherwise
        error('ERROR: no matching values found for variable
            axialFlowAtOutlet')
    end
end
switch bladeLoadingMethod
    case 'UcU'
        velAngular=n.*2.*pi./60;%is a scalar, angular velocity of
            the turbine.
        uAlongBlade=velAngular.*R;% get one value for each node,
            peripheral velocity of the runner
        FractionOfBladelength=G./G(end,:);%fraction of the
            streamline along the blade
        cmAlongBlade=zeros(iEllipse,jEllipse);
        for j=1:1:length(cmAlongBlade)
            cmAlongBlade(:,j)=Q./(jEllipse
                -1)./getdAij((G(:,j)+GToAdd(:,j)),dA1,
                    dA2,P.Aij,GGeom(end,j));
        end
        UcU1=uAlongBlade(1, :).*(uAlongBlade(1,:)-(cmAlongBlade(1,
            :))./(tan(beta1)));
        UcU2=uAlongBlade(end, :).*(uAlongBlade(end,:)-
            (cmAlongBlade(end, :))./(tan(beta2)));%Should be 0 (or
            very close to 0.
        P.UcU=Values.P.UcU;
        UcUAlongBlade=getUcU(FractionOfBladelength,
            UcU1,UcU2,iEllipse, jEllipse,P.UcU);%Should return a
            distribution for UcU along the blade.
        cuAlongBlade=UcUAlongBlade./uAlongBlade;
        betaAlongBlade=atan(cmAlongBlade./(uAlongBlade-
            cuAlongBlade)); %finding beta by use of velocity
    end
end

```

```

    triangles along the blade.
    betaAlongBlade=betaAlongBlade+(betaAlongBlade<=0).*pi;%so
    my theory is that the un-smoothness of the blade
    originates from atan switching to the wrong quadrant.
    adding pi to all negative values should solve this.
    betaAlongBladeW=betaAlongBlade; %To make it compatible with
    later updates, I think...
case 'BetaDistribution'
    %This part is supposed to sketch a smooth curve for beta,
    based on
    %a guess.

    P.BetaDist=Values.P.BetaDist;
    P.Spanwise=Values.P.Spanwise; %To control how the blade
    changes in the spanwise direction
    switch betaEval
        case 'lineSegment' %To evaluate values of beta at line
            segment between two nodes, and thus get more
            accurate values of theta.
                %The exception is at start and end,
                because
                %beta1 and 2 MUST be as calculated
                FractionOfBladelength=G./G(end,:);
                betaAlongBladeW=getBetaDist(beta1,
                    beta2,iEllipse,jEllipse,
                    FractionOfBladelength,
                    'Node',P.BetaDist,P.Spanwise);%So the
                    distribution seems to be equally spaced,
                    and not

                FractionOfBladelength=(G(1:end-1,
                    :)+DeltaG./2)./G(end,:);
                betaAlongBlade=getBetaDist(beta1,beta2,
                    iEllipse, jEllipse, FractionOfBladelength,
                    betaEval,P.BetaDist, P.Spanwise); %to
                    monitor relative velocity
        case 'Node'
            FractionOfBladelength=G./G(end,:);

            betaAlongBlade=getBetaDist(beta1,beta2 ,
                iEllipse,jEllipse ,
                FractionOfBladelength,betaEval,P.BetaDist,P.Spanwise);
            betaAlongBladeW=betaAlongBlade;%to monitor
            relative velocity, w, later on.
        otherwise
            error('ERROR: no fitting value for betaEval')
    end

otherwise
    error('ERROR: The variable "bladeLoadingMethod" has no

```

```

        matching value.')
end
w=sqrt(cm.^2+(cm./tan(betaAlongBladeW)).^2);
c=sqrt(cm.^2+(ufun(R)-cm./tan(betaAlongBladeW)).^2);
cu=ufun(R)-cm./tan(betaAlongBladeW);
%plotting c, w, and UCu
% ucu
figure(figno)
figname.UcU=figno;
figno=figno+1;
plot(G./max(G),(ufun(R).*cu)./max(ufun(R).*cu))
xlabel('G/G_{max}')
ylabel('u \cdot c_u/max(u \cdot c_u)')
title('u \cdot c_u distribution')
%w
figure(figno)
figname.w=figno;
figno=figno+1;
plot(G./max(G),w./Values.DimensionlessVelocity)
xlabel('G/G_{max}')
ylabel('w/u_1')
title('w distribution')
%c
figure(figno)
figname.c=figno;
figno=figno+1;
plot(G./max(G),c./Values.DimensionlessVelocity)
xlabel('G/G_{max}')
ylabel('c/u_1')
title('c distribution')
%cm
figure(figno)
figname.cm=figno;
figno=figno+1;
plot(G./max(G),cm./Values.DimensionlessVelocity)
xlabel('G/G_{max}')
ylabel('c_m/u_1')
title('c_m distribution')
%u
figure(figno)
figname.u=figno;
figno=figno+1;
plot(G./max(G),ufun(R)./Values.DimensionlessVelocity)
xlabel('G/G_{max}')
ylabel('u/u_1')
title('u distribution')
%% H-streamline
H=G;
switch betaEval
    case 'Node'

```

```

DeltaH=DeltaG./tan(betaAlongBlade(1:end-1,:));%Not including the
    last beta, because I get trouble with the sizes of the
    matrices. maybe i should take the mean values of beta?
    case 'lineSegment'
        DeltaH=DeltaG./tan(betaAlongBlade);
    end
for i=2:iEllipse
    H(i,:)=H(i-1,:)+DeltaH(i-1,:);
end
%% G-H plane
figure(figno)
figname.GHplane=figno;
figno=figno+1;
plot(H./max(max(H)),G./max(max(G)),'o-')
hold on
xlabel('H/H_{max}')
ylabel('G/G_{max}')
title('Plot of the G-H plane')
%% Polar coordinates
deltaTheta=2.*asin((DeltaH./2./R(2:end,:)));%More accurate in my
    opinion
theta=zeros(size(R));%Initializing theta
for i=2:iEllipse
    theta(i,:)=theta(i-1,:)+deltaTheta(i-1,:);%finding theta for
        the domain by setting the initial value of theta equal to
        zero
end
figure(figno)
figname.RThetaPlane=figno;
figno=figno+1;
polarplot(theta,R./Values.DimensionlessRadius)%Plotting theta in
    polar coordinates
title('Theta-R/R_{max} in polar coordinates')
%% Back to cartesian coordinates
BladesPlotted='all';
figure(figno)
figname.streamPlane=figno;
figno=figno+1;
hold on
grid on
switch BladesPlotted
    case 'one'
        X=R.*cos(theta);
        Y=R.*sin(theta);
        mesh(X./Values.DimensionlessRadius
            ,Y./Values.DimensionlessRadius ,
            Z./Values.DimensionlessRadius) %
            (rad2deg(betaAlongBlade)>=90).*2)%or surf. Whatever you
            prefer
    case 'all'

```

```

for i=0:1:(zla-1)
X=R.*cos(theta+i*2*pi/zla);
Y=R.*sin(theta+i*2*pi/zla);
surf(X./Values.DimensionlessRadius
      ,Y./Values.DimensionlessRadius ,
      Z./Values.DimensionlessRadius)
      %(rad2deg(betaAlongBlade)>=90).*2) %or surf. Whatever
      you prefer
end
[XH,YH,ZH]=cylinder(RGeom(:,1));
[XS,YS,ZS]=cylinder(RGeom(:,jEllipse));
surf(XH./Values.DimensionlessRadius
      ,YH./Values.DimensionlessRadius, ZGeom(:,
      1).*ones(1,21)./Values.DimensionlessRadius)
surf(XS./Values.DimensionlessRadius
      ,YS./Values.DimensionlessRadius, ZGeom(:,
      jEllipse).*ones(1,21)./Values.DimensionlessRadius)
otherwise
      error('ERROR: unrecognised value of BladesPlotted')
end
xlabel('X/R_1')
ylabel('Y/R_1')
zlabel('Z/R_1')
title('3D-plot of the streamlines')
% Plot of the UcU dist. and the beta dist.

figure(figno)
figname.betaDist=figno;
figno=figno+1;
hold on
% yyaxis left
switch betaEval
      case 'Node'
for j=1:jEllipse
plot(G(:,j)./G(iEllipse,j),rad2deg(betaAlongBlade(:,j)));
end
      case 'lineSegment'
      for j=1:jEllipse
      plot((G(1:end-1,j)+ DeltaG(:,j)./2)./G(iEllipse,j),
            rad2deg(betaAlongBlade(:,j)));
      end
      otherwise
      error('ERROR: no matching betaEval')
end
xlabel('G/G_{max}')
ylabel('\beta')
title('Beta distribution')

fclose('all');
% The shapes of the trailing and leading edges.

```

```

LEshape
TEshape
figname.LETE=figno;
figno=figno+1;
curveStart=3;
curveEnd=2;
BladeCurveX=Xup (curveStart:1:end-curveEnd, :);
for i=[1,29:-1:25,3,30:1:34,2]
    BladeCurveX=[BladeCurveX;TE(:,1,i)'];
end
BladeCurveX=[BladeCurveX;Xdown (end-curveEnd:-1:curveStart, :)];
for i=[2,34:-1:30,3,25:1:29,1]#[2,34:-1:30,3,25:1:29,1]
    BladeCurveX=[BladeCurveX;LE(:,1,i)'];
end
BladeCurveX=[BladeCurveX;Xup (curveStart, :)];

LeIndex=iEllipse-(curveStart-1)-curveEnd+1+5+1;%number of
    elements in a str.line-elements removed at start and
    end+elements up to, and including the LE
TeIndex=LeIndex+5+1+iEllipse-curveEnd- (curveStart-1)+1+
    5+1;%Elements to the LE, +elements until streamline starts
    again, -elements not included in streamline+elements at TE,
    including TE.

BladeCurveY=Yup (curveStart:1:end-curveEnd, :);
for i=[1,29:-1:25,3,30:1:34,2]
    BladeCurveY=[BladeCurveY;TE(:,2,i)'];
end
BladeCurveY=[BladeCurveY;Ydown (end-curveEnd:-1:curveStart, :)];
for i=[2,34:-1:30,3,25:1:29,1]
    BladeCurveY=[BladeCurveY;LE(:,2,i)'];
end
BladeCurveY=[BladeCurveY;Yup (curveStart, :)];

BladeCurveZ=Zup (curveStart:1:end-curveEnd, :);
for i=[1,29:-1:25,3,30:1:34,2]
    BladeCurveZ=[BladeCurveZ;TE(:,3,i)'];
end
BladeCurveZ=[BladeCurveZ;Zdown (end-curveEnd:-1:curveStart, :)];
for i=[2,34:-1:30,3,25:1:29,1]
    BladeCurveZ=[BladeCurveZ;LE(:,3,i)'];
end
BladeCurveZ=[BladeCurveZ;Zup (curveStart, :)];

figure (figno)
figname.curveSurf=figno;
figno=figno+1;
surf (BladeCurveX./Values.DimensionlessRadius,
    BladeCurveY./Values.DimensionlessRadius,
    BladeCurveZ./Values.DimensionlessRadius)%, 'b')

```

```

hold on
plot3(BladeCurveX(LeIndex,
    :)./Values.DimensionlessRadius,BladeCurveY(LeIndex,
    :)./Values.DimensionlessRadius,BladeCurveZ(LeIndex,
    :)./Values.DimensionlessRadius,'ro')
plot3(BladeCurveX(TeIndex,
    :)./Values.DimensionlessRadius,BladeCurveY(TeIndex,
    :)./Values.DimensionlessRadius,BladeCurveZ(TeIndex,
    :)./Values.DimensionlessRadius,'rx')
cb=colorbar;
cb.Label.String='Z/D_1';
xlabel('X/R_1')
ylabel('Y/R_1')
title('The blade')
axis equal
view(2)
% If the TE is not willing to cooperate and you rather remove
% one point at the edge than continue altering the curveStart
% value:
switch axialFlowAtOutlet
    case 'yes'
        BladeCurveX=[BladeCurveX(2:35,:)
            ;BladeCurveX(37:end-1,);BladeCurveX(2,)];
        BladeCurveY=[BladeCurveY(2:35,:)
            ;BladeCurveY(37:end-1,);BladeCurveY(2,)];
        BladeCurveZ=[BladeCurveZ(2:35,:)
            ;BladeCurveZ(37:end-1,);BladeCurveZ(2,)];
    case 'no'
    otherwise
        error('ERROR: No matching value for axialFlowAtOutlet')
end
%
% This section should be commented out if you don't want to
% overwrite the files written for turbogrid
makeCurveFile
getVolume
% BC for cfx and other output:
%clc
fprintf('Pressure at 1: \t \t\t\t\t\t\t\t%.2f
\t\t[kPa]\n',rho*g*(Head+hf(Q)-hBooster(Q))/1000)
fprintf('Pressure at 2: \t\t\t\t\t\t\t\t%.2f \t\t[kPa] \n',
    (hBooster(Q) +HeadDiffSubmergence-
    hfDraftTube(Q)).*rho.*g./1000)
fprintf('Volume flow rate: \t \t\t\t\t\t\t%.2f \t\t[m^3/s]\n', Q)
fprintf('Meridional velocity at 1 boundary: \t\t\t%.2f
\t\t[m/s]\n',Q/(pi*2*R1GuideVanes*b))
fprintf('Meridional velocity at 2 boundary:
\t\t\t%.2f\t\t[m/s]\n', Q/(pi*(D2.^2)./4))
fprintf('Mass flow rate: \t\t\t\t\t\t\t%.2f \t\t[kg/s]\n',Q*rho)
fprintf('Greatest value for c: \t\t\t\t\t\t\t%.2f \t\t[m/s]\n',

```

```

    max(max(c)))
fprintf('Greatest value for w: \t\t\t\t\t%.2f \t\t[m/s]\n',
    max(max(w)))
fprintf('Required head:\t\t\t\t\t\t\t%.2f \t\t[m]\n',Hreq)
fprintf('Time for one passing is:\t\t\t\t\t%.4f
\t\t[s]\n',60/n/zla)
if exist('V','var')
    fprintf('\nApproximate volume of the geometry:
\t\t%.2f\t\t[m^3]\n',V)
    fprintf('Time for flow to go from inlet to
outlet:\t%.2f\t\t[s]\n',V/Q)
end

deHaller=(Q/(pi*2*R1GuideVanes*b)*sin(deg2rad(betaDim)))/(Q/(pi*(D2.^2)./4));
%(DIXON page 85&86)
if deHaller <0.72
    fprintf('\nWARNING: The DeHaller criterion is not fulfilled:
(c1/c2) = %.2f <0.72\n',deHaller)
else
    fprintf('\nThe DeHaller criterion is fulfilled: (c1/c2) = %.2f
>=0.72\n',deHaller)
end
fprintf('Smallest near wall grid size dS wrt. y+:\t%.2e
\t[m]\n',min(min(getGridSize(200,rho,nu,max(max(G),w))))
switch addDraftTube
    case 'yes'
        AIn=pi.*(max(ZDraft).*tan(draftTubeAngle)
+RGeom(iEllipse,jEllipse)).^2;
        VIn=Q./AIn;
        PtotIn=patm+
            rho*g*(HeadDiffSubmergence+hBooster(Q)-hfDraftTube(Q)
+LDraftTube);
        REIn=rho*VIn*sqrt(AIn/pi)*2/mu; %Reynolds number at inlet
to draft tube
        if REIn>2300
            PstatIn=PtotIn-
                Values.TurbulentEnergyCorrectionFactor.*rho.*VIn.^2./2;
        else
            PstatIn=PtotIn-Values.LaminarEnergyCorrectionFactor.*rho.*VIn.^2./2;
        end
        fprintf('Static pressure at inlet:\t
\t\t\t\t\t%.2f\t\t[kPa]\n',PstatIn./1000)
    end
save('P.mat','P')%saves the struct with the points for later
reference. Should be moved to individual folder if you decide
to run a simulation with this design.
NPSHA=getNPSHA(hBooster(Q), patm,pvapour,HeadDiffSubmergence,
'pump');
NPSHR=getNPSHR(cm,ufun(R),'pump');
if NPSHA<=max(max(NPSHR))

```

Appendix G

```
fprintf('Warning, cavitation will occur at some point\nNPSH_A=
%.2f < %.2f =max(NPSH_R)\n', NPSHA,max(max(NPSHR)))
else
    fprintf('NPSHA= %.2f > %.2f =max(NPSHR)\n',
        NPSHA,max(max(NPSHR)))
end
fprintf('Smallest boundary layer thickness:\t\t\t%f\t[m]\n',
    min(min(getBLThickness(w,(max(G)-G))))
n_q=n*sqrt(Q/f)/((Head-hBooster(Q)+hf(Q)).^0.75);
fprintf('speed number n_q=\t\t\t\t\t\t\t%.2f\t\t[-]\n',n_q)
```

Choices.m

```
Geometry='BezierDefByPoints';
Recording=false;
bladeLoadingMethod='BetaDistribution';
betaEval='lineSegment';
FindBetaBlade='freeVortex';
addDraftTube='yes';
addGuideVanes='yes';
addGuideVaneGeometry='no';
empiricalGamma='no';
nondimensional='yes';
axialFlowAtOutlet='no';
```

getValue.m

```
%Values
%All case specific values here have been redacted to avoid
disclosing sensitive data to
%the public.
global Values g rho mu nu patm pvapour
Values.ax=0.25; %initial
    x value to define the acceleration along the blade
Values.ay=0.75; %same
    for y
Values.b=%Height at the guide vane stem
%Values for cul from CFD:
Values.culCFD=[];
Values.D1=%Current diameter at the guide vanes
Values.D2=%Diameter imposed by existing geometry
Values.Dhub=%hub diameter
Values.dx=0.01; %Value to
    change ax and ay with for each iteration
Values.dy=0.01;
Values.f=0.98; %Different for semi
    axial turbine. not quite sure wether the pump is radial or
    semi-axial (It's not PURELY radial at least).
Values.gamma=0.8;
```

```

Values.hBooster=@(Q)%Assuming ideal slipless characteristic for
    boosterpump, working in the same range of Q as the RPT.
Values.Head=%The design head
Values.HeadDiffSubmergence=%The height difference between the
    average lower reservoir height and the RPT
Values.hf=@(Q)%friction losses for the penstock
Values.hfDraftTube=@(Q)%FrictionLosses from draft tube to the
    lower reservoir.
Values.iEllipse=15;
Values.jEllipse=15;
Values.LaminarEnergyCorrectionFactor=2;%Energy correction factor
    for laminar flow
Values.P2=
Values.P.Aij=[
    0.0495  0.3134
    0.3122  0.7536
    0.8813  0.9694
    ];%Describes the initial variation of the cross sectional area.
Values.P.BetaDist=[0 1;0.7 0.8;0.7 0.1;1 0];%Points describing
    the blade angle variation
switch Geometry
case 'BezierDefByPoints'
Values.P.LEp=[
    %P for points
    ];%The points which generate the Leading edge by use of Bezier
    curves
Values.P.TEpMid=[
    ];
otherwise
Values.P.LEp=[
    ];%The points which generate the Leading edge by use of Bezier
    curves
Values.P.TEp=[
    ];
end
Values.P.LERunner=[
    ];
Values.P.TERunner=[
    ];
Values.P.Spanwise=[0 1;0.9 1;1 2];%To control how the blade
    changes in the spanwise direction
Values.P.UcU=[0 1; 0.3 1 ;0.5 0; 1 0];
Values.RlGuideVanes=%Radius of the guideVanes
Values.RGuidevanes=%Distance from TE to points of extended runner
    geometry
Values.RLe1=%The target values for The leading edge
Values.RLe2=
Values.Routlet(1)=%Values from the supplied
    figure.Values.runnerheight=1.32;%0.527; %From draft tube to
    top of guide vane.

```

Appendix G

```
Values.Routlet(2)=
Values.TurbulentEnergyCorrectionFactor=1.075;%Energy correction
    factor for turbulent flow (approximate)
Values.n=250; %current rotational
    velocity
Values.Q=%Design volume flow rate
Values.zla=8; %initial guess for runner blades. Has to be smaller
    or equal to 8.
Values.zle=%number of guide vanes
Values.ZLe1=
Values.ZLe2=
Values.Zoutlet(1)=
Values.Zoutlet(2)=
switch nondimensional
    case 'yes'
        Values.DimensionlessRadius=Values.RlGuideVaness;%Value for
            nondimensionalization of lengths
        Values.DimensionlessVelocity=
            Values.DimensionlessRadius*Values.n/60*2*pi; %Value for
            nondimensionalization of velocities
        Values.DimensionlessQ=Values.Q;%DesignFlowRate
        Values.DimensionlessH=(Values.RlGuideVaness*Values.n/60*pi*2)^2/g/2;
            %u_1^2/2g (head coefficient)
    case 'no'
        Values.DimensionlessRadius=1; %Value for
            nondimensionalization of lengths
        Values.DimensionlessVelocity=1; %Value for
            nondimensionalization of velocities
        Values.DimensionlessQ=1; %DesignFlowRate
        Values.DimensionlessH= 1;
    otherwise
        error('ERROR: no matching value for variable
            "nondimensional". Must be either yes or no.')
end
```

getDeltaX2.m

```
function deltaXoutlet=getDeltaX2(D2,Dhub,jEllipse)
%This function is supposed to output a vector for x-values (or
    R-values for
%dividing a circular outlet into jEllipse-1 parts of equal area.

deltaAoutlet=pi*(D2./2).^2-(Dhub./2).^2./(jEllipse-1); %outlet
    area between two streamlines
deltaXoutlet=zeros(jEllipse,1);
deltaXoutlet(1)=Dhub/2;
deltaXoutlet(end)=D2/2;
for i=2:1:jEllipse-1
    deltaXoutlet(i)=sqrt(deltaXoutlet(i-1).^2+deltaAoutlet./pi);
```

```
end
end
```

```
getEta_h.m
```

```
function eta_h=getEta_h(Q,QBEP,n,HBEP)
nq=n.*sqrt(QBEP)/(HBEP.^0.75);
m=0.08*0.5.*(1/QBEP).^0.15.*(45./nq).^0.06;
eta_hMax=1-0.055*(1/QBEP).^m-0.09.*(log(nq/45)).^2.5; %Guelich
    page 142
qRed=Q/QBEP;
eta_h=eta_hMax.*(1-0.6.*(qRed-0.9).^2-0.25.*(qRed-0.9).^3); %as
    in Guelich page 166
end
```

The script `extractBezierByPoints.m` is an altered version of a script collected from [24].

```
%Hello! this will plot Bezier curve for n control points
%This is a replacement of the program 'Parametic Cubic Bezier
    Curve'
%submitted before ...Bezier for any number of points ...enjoy
function P=extractBezierByPoints(figNo,points)
%Outputs a bezier curve as pchip.
n=size(points,1);%input('Enter no. of points ');
w=2;%input('Press 1 for entry through mouse or 2 for keyboard
    repectively-->');
n1=n-1;
if w==1
    figure(figNo)
    [p]=ginput(n);
end
if w==2
    [p]=points;%input('Enter co-ordinates of points within brackets
        ->[x1 y1;x2 y2;x3 y3;...;xn yn] ');
end

for i=0:1:n1
sigma(i+1)=factorial(n1)/(factorial(i)*factorial(n1-i)); % for
    calculating (x!/(y!(x-y)!)) values
end
l=[];
UB=[];
for u=0:0.002:1
for d=1:n
UB(d)=sigma(d)*((1-u)^(n-d))*(u^(d-1));
end
l=cat(1,l,UB); %catenation
end
```

Appendix G

```
P=1*p;
if figNo~=0
    line(P(:,1),P(:,2))
    %line(p(:,1),p(:,2))
end
P=pchip(P(:,1),P(:,2));
end

% books for reference on the subject of cad/cam author Roger
    Adams ,
% Ibrahim Zeid
%Prepared by Mechanical engg. student NIT Allahabad , India
% for any questions feel free to mail me at
    slnarasimhan89@gmail.com
```

The script `interparc.m` was collected from [11]
the script `distance2curve.m` was collected from [10] `getdAij.m`

```
function dAij=getdAij(G,dA1,dA2,Pmid,Gmax)
if nargin==4%If G is a matrix/vector, we can use that to find the
    maximal value of G
    Gfrac=G./max(G);
else
    Gfrac=G/Gmax;%Else we need a reference input to find G
end
P=[
    0      0
    Pmid
    1      1
    ];%Some points by which the acceleration is defined in this
    script. May be changed if necessary.
dAijChip=extractBezierByPoints(0,P);%Input these points to get a
    bezier curve as pchip.
dAij=dA1+(ppval(dAijChip,Gfrac)).*(dA2-dA1);%evaluate the curve
    at the points along the Length of the streamline.
end
```

`cutOffBlade.m`

```
function [Rout,Zout]=cutOffBlade(R,Z,TE,LE,iEllipse,jEllipse)
TEpchip=extractBezierByPoints(0,TE);
Rout=zeros(size(R));
Zout=zeros(size(Z));
for j=1:1:jEllipse
    %Cut off at TE, first
    Zpchip=pchip(R(:,j),Z(:,j)); %Piecewise
    polynomial of streamline wrt r
    rr=linspace(max(R(end,j),min(TE(:,1))),min(R(1,j),max(TE(:,1))
```

```

        ,1))))); %Interval where both streamline and TE are
        defined
zDiff=ppval(Zpchip,rr)-ppval(TEpchip,rr);
zDiffpchip=pchip(rr,zDiff);
Rcut=unique(fnzeros(zDiffpchip));
rz=interparc(iEllipse , linspace(R(end,j),
    Rcut),ppval(Zpchip,linspace(R(end,j),Rcut)), 'pchip');
Rout(:,j)=rz(:,1);
Zout(:,j)=rz(:,2);
end
R=Rout(end:-1:1,:);
Z=Zout(end:-1:1,:);
%Switch axis and cut off LE. Had problems with this one due to
    vertical
%lines at the outlet.
LEpchip=extractBezierByPoints(0,LE(:,end:-1:1));
hold on
    for j=1:1:jEllipse
        Rpchip=pchip(Z(:,j),R(:,j)); %Piecewise
            polynomial of streamline wrt r
        zz=linspace(max(Z(end,j),min(LE(end,2),LE(1,2)))
            min(Z(1,j),max(LE(1,2),LE(end,2))))); %Interval where
            both streamline and TE are defined
        rDiff=ppval(Rpchip,zz)-ppval(LEpchip,zz);
        rDiffpchip=pchip(zz,rDiff);
        Zcut=unique(fnzeros(rDiffpchip));
        zr=interparc(iEllipse,linspace(Zcut,Z(1,j)),ppval(Rpchip
            ,linspace(Zcut,Z(1,j))) , 'pchip');
        Rout(:,j)=zr(:,2);
        Zout(:,j)=zr(:,1);
    end
Rout=Rout(end:-1:1,:);
Zout=Zout(end:-1:1,:);
end

```

GetGGeomAtG0.m

```

function GToAdd =
    getGGeomAtG0(Rintersection,Zintersection,RGeom,ZGeom)
%gets the value of GGeom where G starts.
GToAdd=zeros(1,size(RGeom,2));
%initializing GToAdd
for j=1:size(GToAdd,2)
    %Iterating
    through the lines
    i=2;
    %Starting a t i=2 to include i=1 at i-1
    while RGeom(i,j)>Rintersection

```

```

                                %While we are away
    from the starting point
    GToAdd(1, j)=GToAdd(1, j)+sqrt((RGeom(i-1, j)-RGeom(i, j)).^2
        +(ZGeom(i-1, j)-ZGeom(i, j)).^2); %adding the length to
        the value of G
    i=i+1;

    %incrementing index
end
GToAdd(1, j)=GToAdd(1, j)+sqrt((RGeom(i-1, j)-Rintersecion(1, j)).^2+
    (ZGeom(i-1, j)- Zintersecion(1, j)).^2); %Adding the rest
    of the line
end
end

```

QHCcalculations.m

```

%% 5. Impeller blade number
%zla number of blades
%zle number of vanes
resonance=true;
zle=Values.zle;
zla=Values.zla;
nu1=[1,2,3].*[1;1;1];
nu2=nu1';
m=zeros(3,3);
while resonance==true
    m=abs(nu2.*zla-nu1.*zle);
    if sum(sum((m==0)+(m==1)))==0
        resonance=false;
    else
        zla=zla-1;
    end
end
%% Initializing variables
switch FindBetaBlade
case 'freeVortex'
    D2m=sqrt(0.5*((2*R(iEllipse, jEllipse))^2
        +(2*R(iEllipse, 1))^2));%Mean diameter at 2 (Suction
        side)
    %D1=(R(1, jEllipse) +R(1, 1));%Assuming D1 is the arithmetic
        average value.
    D1=sqrt(0.5*((2*R(1, jEllipse))^2+(2*R(1, 1))^2)); %assuming
        that D1 ia supposed to be the geometric average as well.
    switch nondimensional
    case 'yes'
        Values.DimensionlessH=(D1*Values.n/60*pi)^2/g/2
            ;%u_1^2/2g (head coefficient)
        Values.DimensionlessVelocity=(D1*Values.n/60*pi);%u_1

```

```

case 'no'
    Values.DimensionlessH=1;
    Values.DimensionlessVelocity=1;
otherwise
    error('ERROR: no matching value for variable
          "nondimensional"')
end
ATE=0;%initializing area at the trailing edge
for j=1:1:jEllipse-1
ATE=ATE+
    getdAij(GToAdd(:,j),dA1,dA2,P.Aij,max(GGeom(:,j)));
    %Find the area at the runner outlet
end
u=(pi.*D1.*n)./(60); %tangential velocity at
1
switch nondimensional
case 'yes'
    Values.DimensionlessH=u^2/g;%u_1^2/2g (head
    coefficient)
end
A2=pi.*(R(iEllipse,jEllipse).^2-R(iEllipse,1).^2); %Area
at 2
Hmax=(u).^2./g; %Highest theoretical
shutoff height
lambdaLa=pi./2; %Angle between blades
and plates
alpha2=pi./2; %Angle between
circumferential direction and absolute velocity
Hfun=@( Q,beta1B,u,A1,A2,eta_h,gamma,tau_1,D2m,D1,alpha2,
f) (eta_h*u.^2)/(g).* (gamma
-(Q)./(f*A1*u*tan(beta1B)).*(tau_1+
(A1*(D2m./D1)*tan(beta1B))/(A2*tan(alpha2)))); %p.133
Guelich
Hideal=@(u,Q,A,beta) u./g.*(u- Q./(A.*tan(beta)));
hf= Values.hf; %Friction losses of the
existing system
hfDraftTube=Values.hfDraftTube; %FrictionLosses from
draft tube to the lower reservoir.
hBooster=Values.hBooster;%Assuming ideal slipless
characteristic for boosterpump, working in the same
range of Q as the RPT.
%Beta with slip:
beta1B=deg2rad(10); %Setting initial value
for blade angle
deltaBeta=1;
bladeBlockage= @(beta1B,lambdaLa) 1./(1-(zla.*e
)./(pi.*D1.*sin(beta1B).*sin(lambdaLa)));%Guelich page
XXX
while true %Lifhack for "do
while" loop in matlab

```

```

beta1B=beta1B+deg2rad(deltaBeta);
f=Values.f; %Different for
            semi axial turbine. not quite sure wether the pump
            is radial or semi-axial (It's not PURELY radial at
            least).
%       cm1=mean(cm(1,:));
       cm1=Q/ATE;
       c_luB=u-(cm1)./(tan(beta1B));
switch empiricalGamma
case 'yes'
       culCFD=Values.culCFD(1);%Not sure how I'll extract
       the values from CFX yet...
       gamma=getGammaEmpirical(u,culCFD,c_luB);
case 'no'
       gamma=Values.gamma;%frequently between 0.7 and 0.8
otherwise
       error('ERROR: no valid value for variable
       EmpiricalGamma')
end
beta1Slip=atan(cm1/(u-(c_luB-(1-gamma).*u)));
tau_1=bladeBlockage(beta1B,lambdaLa);
eta_h=getEta_h(Q,Q,n,Head);%finding the hydraulic
            efficiency at Q=QBEP. therefore same Q twice.
Hreq=Hfun(Q,beta1B,u,ATE,A2,eta_h,gamma,tau_1,D2m,
            D1,alpha2,f);
if (Head+hf(Q)-hBooster(Q)) <Hreq
       break;
end
end
Qfun=@(Hfun tan(beta1Slip).*ATE.*(u.^2-Hfun.*g); %function
            to find Q
steps=97;%number of Q-values to plot the head for.
figure(figno)%plotting the pump characteristic
figname.PumpChar=figno;
figno=figno+1;
hold on
yyaxis left
plot([0:1:steps]./Values.DimensionlessQ, Hfun(0:1:steps,
            beta1B, u, ATE, A2, getEta_h(0:1:steps,Q,n,Head),
            gamma, tau_1, D2m,D1, alpha2,
            f)./Values.DimensionlessH);
%getGamma(beta1B,zla,D2m,D1,f),
tau_1,D2m,D1,alpha2,f));
%beta1B,u,A,getEta_h(0:1:105,Q,n,Head));
plot([0:1:steps]./Values.DimensionlessQ,
            (Head+hf(0:1:steps)-
            hBooster(0:1:steps))./Values.DimensionlessH)
plot([0:1:steps]./Values.DimensionlessQ,
            (Hideal(u,0:1:steps,ATE
            ,beta1B))./Values.DimensionlessH)

```

```

plot([0:1:steps]./Values.DimensionlessQ,
      (Hideal(u,0:1:steps ,ATE,
              betaSlip))./Values.DimensionlessH)
switch nondimensional
  case 'yes'
    ylabel('\Psi')
    ylh = get(gca,'ylabel');
    ylp = get(ylh, 'Position');
    set(ylh, 'Rotation',0, 'Position',ylp,
        'VerticalAlignment','middle',
        'HorizontalAlignment','right')
  case 'no'
    ylabel('Head')
end
yyaxis right
plot([0:1:steps]./Values.DimensionlessQ ,getEta_h(0:1:steps
,Q,n,Head),'-x')
switch nondimensional
  case 'yes'
    legend('Head with losses', ['System curve at
      \Psi_{BEP}=
      ', sprintf('%.2f', (Head-hBooster(Values.Q)+hf(Values.Q))./Values.
      without losses', 'lossless characteristic with
      slip' , 'Efficiency', 'Location', 'southwest')
  case 'no'
    legend('Head with losses', sprintf('System curve at
      H=%.2f m', Head), 'characteristic without
      losses', 'lossless characteristic with
      slip', sprintf('Efficiency for
      Q_{BEP}=%.2fm^3/s', Q), 'Location', 'southwest')
end
ylabel('\eta_h')
ylh = get(gca,'ylabel');
ylp = get(ylh, 'Position');
set(ylh, 'Rotation',0, 'Position',ylp,
    'VerticalAlignment','middle',
    'HorizontalAlignment','left')
switch nondimensional
  case 'yes'
    xlabel('q*')
  case 'no'
    xlabel('Q')
end
betaDim=rad2deg(beta1B);
case 'streamlines'
  beta1=zeros(1,jEllipse);
  Qold=Q;
  for j=1:1:jEllipse
    D2m=2*R(iEllipse,j);% decided to use the radius of the
      streamlines as reference, and to use average data

```

```

        from half the interval above nde half below for
        calculations of volume flow rate etc.
D1=2*R(1,j);%Assuming D1 is the arithmetic average
        value.=sqrt(0.5*((2*R(1,jEllipse))^2+(2*R(1,1))^2));
        %assuming that D1 ia supposed to be the geometric
        average as well.
u=(pi.*D1.*n)./(60);                %tangential velocity
        at 1
if j==1
    A2=0.5.*(getdAij((G(iEllipse,j)+GToAdd(:,j)),
        dA1,dA2,P.Aij,GGeom(end,j)));% using only half of
        the area for first and last line
    A1=0.5.*(getdAij((G(1,j)+GToAdd(:,j)),dA1,dA2,
        P.Aij,GGeom(end,j)));
    Q=Qold./(jEllipse-1)./2;
elseif j==jEllipse
    A2=0.5.*(getdAij((G(iEllipse,j-1)+GToAdd(:,j-1)),dA1,
        dA2,P.Aij,GGeom(end,j-1)));
    A1=0.5.*(getdAij((G(1,j-1)+GToAdd(:,j-1)),dA1,dA2,
        P.Aij,GGeom(end,j-1)));
    Q=Qold./(jEllipse-1)./2;
else
    A2=0.5.*(getdAij((G(iEllipse,j)+GToAdd(:,j)),dA1,
        dA2, P.Aij,GGeom(end,j)) +getdAij((G(iEllipse,
        j-1)+ GToAdd(:,j-1)), dA1 ,dA2,P.Aij,
        GGeom(end,j-1)));%As the code uses dAij to find
        the area between streamline j and j+1, the area
        for each streamline is assumed to be the average
        of the area for j and j+1
    A1=0.5.*(getdAij((G(1,j)+GToAdd(:,j)),dA1,dA2 ,
        P.Aij,GGeom(end,j))+getdAij((G(1,j-1)+GToAdd(:,j-1)),dA1,dA2,P.A
        Q=Qold./(jEllipse-1);
end
Hmax=(u).^2./g;                %Highest theoretical
        shutoff height
lambdaLa=pi./2;                %Angle between
        blades and plates
alpha2=pi./2;                %Angle between
        circumferential direction and absolute velocity
Hfun=@(Q,beta1B,u,A1,A2,eta_h,gamma,tau_1,D2m,D1,
        alpha2,
        f)(eta_h*u.^2)/(g).*(gamma-(Q)./(f*A1*u*tan(beta1B)).*(tau_1
        +(A1*(D2m./D1)*tan(beta1B))/(A2*tan(alpha2))));%p.133
        Guelich
Hideal=@(u,Q,A1 ,beta) u./g.*(u-Q./(A1.*tan(beta)));
hf=Values.hf;                %Friction losses of the
        existing system
hfDraftTube=Values.hfDraftTube; %FrictionLosses from
        draft tube to the lower reservoir.
hBooster=Values.hBooster; %Assuming ideal slipless

```

```

        characteristic for booster pump, working in the same
        range of Q as the RPT.
beta1B=deg2rad(10); %Setting initial
        value for blade angle
deltaBeta=1;
bladeBlockage= @(beta1B, lambdaLa)
    1./(1-(zla.*e)./(pi.*D1.*sin(beta1B).*sin(lambdaLa)));
    %Guelich page XXX
while true %Lifhack for "do
    while" loop in matlab
    beta1B=beta1B+deg2rad(deltaBeta);
    f=Values.f; %Different for
        semi axial turbine. not quite sure whether the
        pump is radial or semi-axial (It's not PURELY
        radial at least).
    cml=Q/ATE;
    c_luB=u-(cml)./(tan(beta1B));
    switch empiricalGamma
    case 'yes'
        gamma=getGammaEmpirical(u,culCFD(j),c_luB);
    case 'no'
        gamma=Values.gamma;
    otherwise
        error('ERROR: no matching values for variable
            empiricalGamma')
    end
    beta1Slip=atan(cml/(u-(c_luB-(1-gamma).*u)));
    tau_1=bladeBlockage(beta1B,lambdaLa);
    eta_h=getEta_h(Q,Q,n,Head); %finding the hydraulic
        efficiency at Q=QBEP. therefore same Q twice.
    Hreq=Hfun(Q,beta1B,
        u,A1,A2,eta_h,gamma,tau_1,D2m,D1,alpha2, f);
    if (Head+hf(Q)-hBooster(Q)) <Hreq
        break;
    end
end
Qmax=Q*1.5;
Qval=linspace(0,Qmax);
steps=97;%number of Q-values to plot the head for.
figure(figno)%plotting the pump characteristic
figname.PumpChar=figno;
figno=figno+1;
hold on
yyaxis left
plot(Qval./Values.DimensionlessQ,Hfun(Qval,beta1B,u,A1
    ,A2,eta_h,gamma,tau_1,D2m,D1,alpha2, f)./Values.DimensionlessH);
    %getGamma(beta1B,zla,D2m,D1,f),tau_1,D2m,D1,alpha2,f));
    %beta1B,u,A1,getEta_h(0:1:105,Q,n,Head));
plot(Qval./Values.DimensionlessQ,
    (Head+hf(Qval)-hBooster(Qval))./Values.DimensionlessH)

```

```

    plot(Qval./Values.DimensionlessQ,Hideal(u,Qval,A1,
        beta1B)./Values.DimensionlessH)
    plot(Qval./Values.DimensionlessQ,Hideal(u,Qval,A1,
        beta1Slip)./Values.DimensionlessH)
    legend('with slip','system curve','\eta=1 without
        slip','\eta=1 with slip')
    ylabel('Head')
    yyaxis right
    ylabel('Efficiency')
    xlabel('Q')
    betaDim=rad2deg(beta1B);
    beta1(j)=beta1B;
end
Q=Qold;
otherwise
    error('ERROR, no matching value for variable FindBetaBlade')
end
switch axialFlowAtOutlet %If the flow at the outlet has an axial
    component, the radial and the meridional flow angle are
    different.
case 'yes'
    epsilonOutlet=atan((Z(1,:)-Z(2,:))./(R(1,:)-R(2,:)));
    beta1B=atan(tan(beta1B).*cos(mean(epsilonOutlet)));
        %Transforming meridional blade angle to radial blade
        angle
case 'no'
otherwise
    error('ERROR: no valid value for variable
        axialFlowAtOutlet')
end
end

```

getBeta1B.m

```

function beta1B=getBeta1B(gamma,c1m,tau1,D1m,n,beta1Bref,R)
%Code based on Johan Guelichs centrifugal pumps page 77
ulref=pi*D1m*n/60;%Finding reference u at D1m
cluRef=(gamma-(c1m.*tau1)/(ulref.*tan(beta1Bref))).*ulref;
    %finding flow velocity at D1m
I=cluRef.*(D1m./2);%assuming free vortex (cu/r=const)
clu=I./R(1,:);%finding cul at all of the inlet nodes
u1=2.*pi.*R(1,:).*n/60;%finding u1 at all inlet nodes
beta1B=atan((c1m.*tau1)./(u1.*gamma-clu));%finding beta1B for all
    of the inlet nodes
end

```

getBetaDist.m

```

function betaAlongBlade=getBetaDist(beta1B,beta2B,iEllipse,
    jEllipse,Fraction,option,P,Pspanwise)

```

```

if nargin==7
    Pspanwise=[0 1;1 1];
end
switch option
    case 'Node'
        betaAlongBlade=zeros(iEllipse,jEllipse);
    case 'lineSegment'
        betaAlongBlade=zeros(iEllipse-1,jEllipse);
    otherwise
        error('ERROR: "Option" not valid')
end
TransversalChangeFactor=ppval(extractBezierByPoints(0,Pspanwise),
    [0:1:jEllipse-1]./(jEllipse-1));%To change the blade by a
    factor along its span
Pold=P;
for j=1:1:jEllipse
    P(2:end-1,2)=Pold(2:end-1 ,2).*TransversalChangeFactor(j);
    betaAlongBlade(:, j)=ppval(extractBezierByPoints(0,P),
        Fraction(:,j)).*(beta1B(j)-beta2B(j)) +beta2B(j);
end
betaAlongBlade(1,:)=beta1B;
betaAlongBlade(end,:)=beta2B;
end

```

LEshape.m

```

%LEshape.m
% The creation of the thickness will be as in the powerpoint
    supplied by
% Rainpower.
% Finding the normal to the surface created by the blades:

[Nx,Ny,Nz]=surfnorm(X,Y,Z);
Xup=X-Nx.*e./2;
Xdown=X+Nx.*e./2;
Yup=Y-Ny.*e./2;
Ydown=Y+Ny.*e./2;
Zup=Z-Nz.*e./2;
Zdown=Z+Nz.*e./2;
% Defining the ratios of the ellipses to be constructed later
bEllipse=1*e;
aEllipse=e/2;

LE=zeros(jEllipse,3,34);%will obtain coordinates for jEllipse
    streamlines, 3 dimensions and 37 points pr streamline.
aAbs=sqrt((Xup(1,:)- Xup(2,:)).^2+(Yup(1,:)-
    Yup(2,:)).^2+(Zup(1,:)- Zup(2,:)).^2);
bAbs=bEllipse;

```

Appendix G

```
LE(:,1,1)=-bAbs./aAbs.*(Xup(1,:)-Xup(2,:))+Xup(1,:);
LE(:,2,1)=-bAbs./aAbs.*(Yup(1,:)-Yup(2,:))+Yup(1,:);
LE(:,3,1)=-bAbs./aAbs.*(Zup(1,:)-Zup(2,:))+Zup(1,:);

aAbs=sqrt((Xdown(1,:)-Xdown(2,:)).^2+(Ydown(1,:)-Ydown(2,:)).^2+(Zdown(1
, :)-Zdown(2,:)).^2);
bAbs=bEllipse;
LE(:,1,2)=-bAbs./aAbs.*(Xdown(1,:)-Xdown(2,:))+Xdown(1,:);
LE(:,2,2)=-bAbs./aAbs.*(Ydown(1,:)-Ydown(2,:))+Ydown(1,:);
LE(:,3,2)=-bAbs./aAbs.*(Zdown(1,:)-Zdown(2,:))+Zdown(1,:);

aAbs=sqrt((Xup(1,:)-Xdown(1,:)).^2+(Yup(1,:)-Ydown(1,:)).^2+
(Zup(1,:)-Zdown(1,:)).^2);
bAbs=e/2;
LE(:,1,3)=-bAbs./aAbs.*(Xup(1,:)-Xdown(1,:))+Xup(1,:);
LE(:,2,3)=-bAbs./aAbs.*(Yup(1,:)-Ydown(1,:))+Yup(1,:);
LE(:,3,3)=-bAbs./aAbs.*(Zup(1,:)-Zdown(1,:))+Zup(1,:);

aAbs=sqrt((LE(:,1,1)-LE(:,1,2)).^2+(LE(:,2,1)-LE(:,2,2)).^2+
(LE(:,3,1)-LE(:,3,2)).^2);
bAbs=e/2;
LE(:, :, 4)=-bAbs./aAbs.*(LE(:, :, 1)-LE(:, :, 2))+LE(:, :, 1);

% Should be able to construct both sides from knowledge of LE 1-4.
%upside:
bAbs=e/12;
LE(:, :, 9)=LE(:, :, 4);

%Points away from the tip
for i=5:1:9
    aAbs=sqrt((LE(:,1,1)-LE(:,1,i-1)).^2+(LE(:,2,1)-
LE(:,2,i-1)).^2+(LE(:,3,1)-LE(:,3,i-1)).^2);
    LE(:, :, i)=bAbs./aAbs.*(LE(:, :, 1)-LE(:, :, i-1))+LE(:, :, i-1);
    LE(:, :, i+5)=bAbs./aAbs.*(LE(:, :, 2)-LE(:, :, (i +5)-1))+LE(
, :, (i+5)-1);
end

%Points on the tip, but flat.
%upside
bAbs=e/12;
aAbs=e/2;
LE(:,1,15)=bAbs./aAbs.*(Xup(1,:)'-LE(:,1,3))+LE(:,1,3);
LE(:,2,15)=bAbs./aAbs.*(Yup(1,:)'-LE(:,2,3))+LE(:,2,3);
LE(:,3,15)=bAbs./aAbs.*(Zup(1,:)'-LE(:,3,3))+LE(:,3,3);

for i=16:1:19
    aAbs=sqrt((Xup(1,:)'-
LE(:,1,i-1)).^2+(Yup(1,:)'-LE(:,2,i-1)).^2+(Zup(1,:)'-
LE(:,3,i-1)).^2);
    LE(:,1,i)=bAbs./aAbs.*(Xup(1,:)'-LE(:,1,i-1))+LE(:,1,i-1);
```

```

    LE(:,2,i)=bAbs./aAbs.*(Yup(1,:)-LE(:,2,i-1))+LE(:,2,i-1);
    LE(:,3,i)=bAbs./aAbs.*(Zup(1,:)-LE(:,3,i-1))+LE(:,3,i-1);
end
%downside
bAbs=e/12;
aAbs=e/2;
LE(:,1,20)=bAbs./aAbs.*(Xdown(1,:)-LE(:,1,3))+LE(:,1,3);
LE(:,2,20)=bAbs./aAbs.*(Ydown(1,:)-LE(:,2,3))+LE(:,2,3);
LE(:,3,20)=bAbs./aAbs.*(Zdown(1,:)-LE(:,3,3))+LE(:,3,3);

for i=21:1:24
    aAbs=sqrt((Xdown(1,:)-LE(:,1,i-1)).^2+(Ydown(1,:)-LE(:,2,i-1)).^2+(Zdown(1,:)-LE(:,3,i-1)).^2);
    LE(:,1,i)=bAbs./aAbs.*(Xdown(1,:)-LE(:,1,i-1))+LE(:,1,i-1);
    LE(:,2,i)=bAbs./aAbs.*(Ydown(1,:)-LE(:,2,i-1))+LE(:,2,i-1);
    LE(:,3,i)=bAbs./aAbs.*(Zdown(1,:)-LE(:,3,i-1))+LE(:,3,i-1);
end

%Continue from here with the ellipsis-form
j=0;
aAbs=bEllipse;
for i=25:1:29
    j=j+1;
    bAbs=sqrt(1-((j./6.*e./2)./(aEllipse)).^2).*bEllipse;
    LE(:, :, i)=bAbs./aAbs.*(LE(:, :, i-10)-LE(:, :, i-20))+LE(:, :, i-20);
end
j=0;
for i=30:1:34
    j=j+1;
    bAbs=sqrt(1-((j./6.*e./2)./(aEllipse)).^2).*bEllipse;
    LE(:, :, i)=bAbs./aAbs.*(LE(:, :, i-10)-LE(:, :, i-20))+LE(:, :, i-20);
end

figure(figno)
hold on
mesh(Xup,Yup,Zup)
mesh(Xdown,Ydown,Zdown)
for i=25:1:34
    plot3(LE(:,1,i),LE(:,2,i),LE(:,3,i),'o')
end

```

TEshape.m

```

%TEshape.m
%% Because the Blade Editor refused to be automatized, we'll have
    to define the thickness here, and just load stuff into
    turboGrid ourselves.

[Nx,Ny,Nz]=surfnorm(X,Y,Z);

```

Appendix G

```
Xup=X-Nx.*e./2;
Xdown=X+Nx.*e./2;
Yup=Y-Ny.*e./2;
Ydown=Y+Ny.*e./2;
Zup=Z-Nz.*e./2;
Zdown=Z+Nz.*e./2;
% Defining the ratios of the ellipses to be constructed later
bEllipse=2*e;
aEllipse=e/2;
%Changing the order of the
Xup(1:end,:)=Xup(end:-1:1,:);
Yup(1:end,:)=Yup(end:-1:1,:);
Zup(1:end,:)=Zup(end:-1:1,:);

Xdown(1:end,:)=Xdown(end:-1:1,:);
Ydown(1:end,:)=Ydown(end:-1:1,:);
Zdown(1:end,:)=Zdown(end:-1:1,:);

TE=zeros(jEllipse,3,34);%will obtain coordinates for jEllipse
streamlines, 3 dimensions and 37 points pr streamline.
aAbs=sqrt((Xup(1,:)-Xup(2,:)).^2+
(Yup(1,:)-Yup(2,:)).^2+(Zup(1,:)-Zup(2,:)).^2);
bAbs=bEllipse;
TE(:,1,1)=-bAbs./aAbs.*(Xup(1,:)-Xup(2,:))+Xup(1,:);
TE(:,2,1)=-bAbs./aAbs.*(Yup(1,:)-Yup(2,:))+Yup(1,:);
TE(:,3,1)=-bAbs./aAbs.*(Zup(1,:)-Zup(2,:))+Zup(1,:);

aAbs=sqrt((Xdown(1,:)-Xdown(2,:)).^2+(Ydown(1,:)-Ydown(2,
:)).^2+(Zdown(1,:)-Zdown(2,:)).^2);
bAbs=bEllipse;
TE(:,1,2)=-bAbs./aAbs.*(Xdown(1,:)-Xdown(2,:))+Xdown(1,:);
TE(:,2,2)=-bAbs./aAbs.*(Ydown(1,:)-Ydown(2,:))+Ydown(1,:);
TE(:,3,2)=-bAbs./aAbs.*(Zdown(1,:)-Zdown(2,:))+Zdown(1,:);

aAbs=sqrt((Xup(1,:)-Xdown(1,:)).^2+(Yup(1,:)-Ydown(1,:)).^2+(Zup(1,:)-Zdown(1,:)).^2);
bAbs=e/2;
TE(:,1,3)=-bAbs./aAbs.*(Xup(1,:)-Xdown(1,:))+Xup(1,:);
TE(:,2,3)=-bAbs./aAbs.*(Yup(1,:)-Ydown(1,:))+Yup(1,:);
TE(:,3,3)=-bAbs./aAbs.*(Zup(1,:)-Zdown(1,:))+Zup(1,:);

aAbs=sqrt((TE(:,1,1)-TE(:,1,2)).^2+(TE(:,2,1)-TE(:,2,2)).^2+
(TE(:,3,1)-TE(:,3,2)).^2);
bAbs=e/2;
TE(:, :, 4)=-bAbs./aAbs.*(TE(:, :, 1)-TE(:, :, 2))+TE(:, :, 1);

% Should be able to construct both sides from knowledge of LE 1-4.
%upside:
bAbs=e/12;
TE(:, :, 9)=TE(:, :, 4);
```

```

%Points away from the tip
for i=5:1:9
    aAbs=sqrt((TE(:,1,1)-TE(:,1,i-1)).^2+(TE(:,2,1)-TE(:,2,i-1)
        )).^2+(TE(:,3,1)-TE(:,3,i-1)).^2);
    TE(:, :, i)=bAbs./aAbs.*(TE(:, :, 1)-TE(:, :, i-1))+TE(:, :, i-1);
    TE(:, :, i+ 5)=bAbs./aAbs.*(TE(:,
        :, 2)-TE(:, :, (i+5)-1))+TE(:, :, (i+5)-1);
end

%Points on the tip, but flat.
%upside
bAbs=e/12;
aAbs=e/2;
TE(:,1,15)=bAbs./aAbs.*(Xup(1,:)'-TE(:,1,3))+TE(:,1,3);
TE(:,2,15)=bAbs./aAbs.*(Yup(1,:)'-TE(:,2,3))+TE(:,2,3);
TE(:,3,15)=bAbs./aAbs.*(Zup(1,:)'-TE(:,3,3))+TE(:,3,3);

for i=16:1:19
    aAbs=sqrt((Xup(1,:)'-TE(:,1,i-1)).^2+(Yup(1,:)'-TE(:,2,i-1)).^2+(Zup(1,:)'
        -TE(:,3,i-1)).^2);
    TE(:,1,i)=bAbs./aAbs.*(Xup(1,:)'-TE(:,1,i-1))+TE(:,1,i-1);
    TE(:,2,i)=bAbs./aAbs.*(Yup(1,:)'-TE(:,2,i-1))+TE(:,2,i-1);
    TE(:,3,i)=bAbs./aAbs.*(Zup(1,:)'-TE(:,3,i-1))+TE(:,3,i-1);
end
%downside
bAbs=e/12;
aAbs=e/2;
TE(:,1,20)=bAbs./aAbs.*(Xdown(1,:)'-TE(:,1,3))+TE(:,1,3);
TE(:,2,20)=bAbs./aAbs.*(Ydown(1,:)'-TE(:,2,3))+TE(:,2,3);
TE(:,3,20)=bAbs./aAbs.*(Zdown(1,:)'-TE(:,3,3))+TE(:,3,3);

for i=21:1:24
    aAbs=sqrt((Xdown(1,:)' -TE(:,1,i-1)).^2
        +(Ydown(1,:)'-TE(:,2,i-1)).^2
        +(Zdown(1,:)'-TE(:,3,i-1)).^2);
    TE(:,1,i)=bAbs./aAbs.*(Xdown(1,:)'-TE(:,1,i-1))+TE(:,1,i-1);
    TE(:,2,i)=bAbs./aAbs.*(Ydown(1,:)'-TE(:,2,i-1))+TE(:,2,i-1);
    TE(:,3,i)=bAbs./aAbs.*(Zdown(1,:)'-TE(:,3,i-1))+TE(:,3,i-1);
end

%Continue from here with the ellipsis-form
j=0;
aAbs=bEllipse;
for i=25:1:29
    j=j+1;
    bAbs=sqrt(1-((j./6.*e./2)./(aEllipse)).^2).*bEllipse;
    TE(:, :, i)=bAbs./aAbs.*(TE(:, :, i-10)-TE(:, :, i-20))+TE(:, :, i-20);
end
j=0;

```

```

for i=30:1:34
    j=j+1;
    bAbs=sqrt(1-((j./6.*e./2)./(aEllipse)).^2).*bEllipse;
    TE(:, :, i)=bAbs./aAbs.*(TE(:, :, i-10)-TE(:, :, i-20))+TE(:, :, i-20);
end

figure(figno)
hold on
mesh(Xup, Yup, Zup)
mesh(Xdown, Ydown, Zdown)
for i=25:1:34
    plot3(TE(:, 1, i), TE(:, 2, i), TE(:, 3, i), 'o')
end

Xup(1:end, :)=Xup(end:-1:1, :);
Yup(1:end, :)=Yup(end:-1:1, :);
Zup(1:end, :)=Zup(end:-1:1, :);

Xdown(1:end, :)=Xdown(end:-1:1, :);
Ydown(1:end, :)=Ydown(end:-1:1, :);
Zdown(1:end, :)=Zdown(end:-1:1, :);

```

makeCurveFile.m

```

%makeCurveFile
%(this is the one used in BladeDesign2 written by Karl E. Not to
  be confused
%with makeCurveFiles written by Helene N.D.
%% blade.crv
% Format: X, Y and Z. leading edge and trailing edge have a
  fourth column
% indexed by "le1" and "te1", respectively. Blade curves are
  separated by a
% single line with a #.
%The curves have to start at the hub and go towards the shroud
switch addGuideVaneGeometry
    case 'yes'
        makeGuideVaneCurves
    otherwise
end
fidcsv=fopen([dir, '/blade.csv'], 'w');
fidBlade=fopen([dir, '/blade.crv'], 'w'); %Opening
    a txt file for the blade
for j=(1:1:size(BladeCurveX, 2)) %For
    number of streamline
    fprintf(fidBlade, '#\n'); %# to
        seperate blades
    for i=1:1:length(BladeCurveX) %For

```

```

    number of elements in each curve
if i==LeIndex                                     %If this
    is the curve element corresponding to le
    string="le1";                                  %write
        this in the file
elseif i==TeIndex                                 %same
    for te
        string="te1";
    else
        string="";
        %otherwise, just don't write anything
    end
fprintf(fidBlade, '%f\t%f\t%f\t%s\n', BladeCurveX(i, j), BladeCurveY(i, j)
    , BladeCurveZ(i, j), string); %print out the corresponding
    X Y and Z values
fprintf(fidcsv, '%f,%f,%f\n'
    , BladeCurveX(i, j), BladeCurveY(i, j), BladeCurveZ(i, j));
    %Trying to create a polysurface for use in cfd Post
end
end
fclose(fidBlade);                                 %Close
    the file
fclose(fidcsv);
%% hub.crv
% format: The axial projection of the shroud and hub,
    respectively(in
% R,theta,Z coordinates, but theta may remain 0)
figure(figno)                                     %just to
    see if it does what it should
figname.HubShroud=figno;
figno=figno+1;
hold on
fid=fopen([dir, '/hub.crv'], 'w');                %open a
    hub file
j=1;
    %initialize j
i=iEllipse;                                       %i
    starts at the end, so at the circular outlet
switch addDraftTube
    case 'yes'
        LDraftTube=RGeom(iEllipse, jEllipse)*10;
        ZDraft=linspace(LDraftTube, LDraftTube/20, 20);
        draftTubeAngle=deg2rad(4);
        for iDraft=1:length(ZDraft)
            fprintf(fid, '%f\t%f\t%f\n', R(iEllipse, j), 0, ZGeom(i, j)
                -ZDraft(iDraft));
            plot(R(iEllipse,
                j)./Values.DimensionlessRadius, (ZGeom(i, j)
                -ZDraft(iDraft))./Values.DimensionlessRadius, 'o')
        end
end

```

```

    otherwise
end
while(ZGeom(i, j)<Z(iEllipse, j)) %while
    the hub has not reached the blade yet
    fprintf(fid, '%f\t%f\t%f\n', R(iEllipse, j), 0, ZGeom(i, j)); %write
        the coordinates of the hub
    plot(R(iEllipse, j)./Values.DimensionlessRadius, ZGeom(i,
        j)./Values.DimensionlessRadius, 'o')
    i=i-1;
end
for i=iEllipse:-1:1 %Then
    write all the elements along the blade(from inlet to outlet)
    fprintf(fid, '%f\t%f\t%f\n', R(i, j), 0, Z(i, j));
    plot(R(i, j)./Values.DimensionlessRadius,
        Z(i, j)./Values.DimensionlessRadius, 'o')
end
iloop=iEllipse; %index to
    find when the hub is not on the blade anymore
while(RGeom(iloop, j)<R(1, j)) %and
    corresponding loop
    iloop=iloop-1;
end
for i=iloop:-1:1 %Print
    the remainder of the hub
    switch addGuideVanes
        case 'yes'
            switch addGuideVaneGeometry
                case 'yes'
                    if RGeom(i, j)<P.TipMin
                        fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
                        plot(RGeom(i, j)./Values.DimensionlessRadius,
                            ZGeom(i, j)./Values.DimensionlessRadius, 'o')
                    end
                otherwise
                    fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
                    plot(RGeom(i, j)./Values.DimensionlessRadius,
                        ZGeom(i, j)./Values.DimensionlessRadius, 'o')
                end
            end
        otherwise
            fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
            plot(RGeom(i, j)./Values.DimensionlessRadius,
                ZGeom(i, j)./Values.DimensionlessRadius, 'o')
        end
    end
end
switch addGuideVanes
    case 'yes'
        switch addGuideVaneGeometry
            case 'yes'
                fidGV=fopen([dir, '/hubGV.crv'], 'w');
                RGuidevanes=[P.TipMin, P.TipMin+0.54, P.TipMin+1];

```

```

fprintf(fid, '%f\t%f\t%f\n', RGuidevanes(1), 0,
        ZGeom(i, j));
plot(RGuidevanes(iGuidevanes)./Values.DimensionlessRadius,
      ZGeom(i, j)./Values.DimensionlessRadius, 'o');
for iGuidevanes= 1:length(RGuidevanes)
    fprintf(fidGV,
            '%f\t%f\t%f\n', RGuidevanes(iGuidevanes),
            0, ZGeom(i, j));
    plot(RGuidevanes(iGuidevanes)./Values.DimensionlessRadius,
          ZGeom(i, j)./Values.DimensionlessRadius, 'o');
end
fclose(fidGV);
otherwise
RGuidevanes=Values.RGuidevanes;
AGuidevanes=2*pi*RGeom(1, j)*(Z(1,1)-Z(1, jEllipse));%want to
keep the area after guide vanes constant to avoid
separation
DeltaZGuidevanes=AGuidevanes./(2.*pi.*(RGeom(1, j))
-RGuidevanes./(2.*pi.*(RGeom(1, j) + RGuidevanes)).*(1
-RGuidevanes./max(RGuidevanes).*0.1); %Making the area
smaller with increasing radius to avoid recirculation
for iGuidevanes=1:length(RGuidevanes)
fprintf(fid, '%f\t%f\t%f\n'
        , RGeom(i, j)+RGuidevanes(iGuidevanes), 0,
        ZGeom(i, j)-DeltaZGuidevanes(iGuidevanes)./2);
plot((RGeom(i, j)
+RGuidevanes(iGuidevanes))./Values.DimensionlessRadius,
      (ZGeom(i, j)-
DeltaZGuidevanes(iGuidevanes)./2)./Values.DimensionlessRadius, 'o');
end
end
otherwise
end
fclose(fid);
%% shroud.crv %Mostly the same as for hub.
fid=fopen([dir, '/shroud.crv'], 'w');
j=jEllipse;
i=iEllipse;
switch addDraftTube
    case 'yes'
        for iDraft=1:length(ZDraft)
            RDraft=ZDraft(iDraft).*tan(draftTubeAngle);
            fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j) +RDraft, 0,
                    ZGeom(i, j)- ZDraft(iDraft));
            plot((RGeom(i, j)+RDraft)./Values.DimensionlessRadius,
                  (ZGeom(i, j)-ZDraft(iDraft))./Values.DimensionlessRadius, 'o')
        end
    otherwise
end
while (ZGeom(i, j)<Z(iEllipse, j))

```

```

    fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
    plot (RGeom(i,
        j) ./Values.DimensionlessRadius, ZGeom(i, j) ./Values.DimensionlessRadius, 'o')
    i=i-1;
end
for i=iEllipse:-1:1
    fprintf(fid, '%f\t%f\t%f\n', R(i, j), 0, Z(i, j));
    plot (R(i, j) ./Values.DimensionlessRadius,
        Z(i, j) ./Values.DimensionlessRadius, 'o')
end
iLoop=iEllipse;
while (RGeom(iLoop, j) < R(1, j))
    iLoop=iLoop-1;
end
switch addGuideVaneGeometry
    case 'yes'
        fidGV=fopen([dir, '/shroudGV.crv'], 'w');
    otherwise
end
for i=iLoop:-1:1
    switch addGuideVaness
        case 'yes'
            switch addGuideVaneGeometry
                case 'yes'
                    if i==iLoop
                        fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0,
                            ZGeom(i, j));
                        plot (RGeom(i, j) ./Values.DimensionlessRadius,
                            ZGeom(i, j) ./Values.DimensionlessRadius, 'o')
                    end
                    fprintf(fidGV, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
                    plot (RGeom(i, j) ./Values.DimensionlessRadius,
                        ZGeom(i, j) ./Values.DimensionlessRadius, 'o')
                otherwise
                    fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
                    plot (RGeom(i, j) ./Values.DimensionlessRadius, ZGeom(i,
                        j) ./Values.DimensionlessRadius, 'o')
                end
            otherwise
                fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j), 0, ZGeom(i, j));
                plot (RGeom(i,
                    j) ./Values.DimensionlessRadius, ZGeom(i, j) ./Values.DimensionlessRadius,
            end
        end
    end
switch addGuideVaness
    case 'yes'
        switch addGuideVaneGeometry
            case 'yes'
                for iGuidevaness=2:length(RGuidevaness)
                    fprintf(fidGV,

```

```

        '%f\t%f\t%f\n', RGuidevanes(iGuidevanes), 0, ZGeom(i, j));
    plot(RGuidevanes(iGuidevanes) ./ Values.DimensionlessRadius,
        ZGeom(i, j) ./ Values.DimensionlessRadius, 'o');
    end
    fclose(fidGV);
otherwise
    for iGuidevanes=1:length(RGuidevanes)
        fprintf(fid, '%f\t%f\t%f\n', RGeom(i, j)
            +RGuidevanes(iGuidevanes), 0, ZGeom(i, j)
            +DeltaZGuidevanes(iGuidevanes) ./ 2);
        plot((RGeom(i, j)
            +RGuidevanes(iGuidevanes)) ./ Values.DimensionlessRadius, (ZGeom
            j)+DeltaZGuidevanes(iGuidevanes) ./ 2) ./ Values.DimensionlessRad
            'o');
    end
end
otherwise
end
axis equal
fclose(fid);

```

getVolume.m

```

%getVolume.m
%Finds the volume of the geometry, to use in transient simulation.
switch addDraftTube %Approximation of volume from leading edge of
    the blade to the point added at the "inlet" in CFX
case 'yes'
    VDraft=sum((Z(end, 1:end-1) - (ZGeom(end, 1:end-1) - max(ZDraft))) .* pi .* (R(end,
        2:end).^2 - R(end, 1:end-1).^2));
case 'no'
    VDraft=0;
otherwise
    error('ERROR: No valid value for addDraftTube')
end
switch addGuideVaness %Approximation of volume of the part after
    the blade and up to the "outlet" in CFX
case 'yes'
    VGuide=sum(((RGeom(1, 1:end-1) + RGuidevanes(end)).^2 - R(1, 1:end-1).^2) .* (Z(1
        , 1:end-1) - Z(1, 2:end)
        + ZGeom(1, 1:end-1) - ZGeom(1, 2:end)) ./ 2 .* pi);
case 'no'
    VGuide=0;
otherwise
    error('ERROR: No valid value for addGuideVaness')
end
dAijV=zeros(iEllipse, jEllipse-1); %The cross sectional area at
    almost all streamlines
for j=1:1:jEllipse-1

```

Appendix G

```
dAijV(:,j)=getdAij((G(:,j)+GToAdd(:,j)),dA1,dA2,P.Aij
,GGeom(end,j));
end
Vmain=sum(sum((G(2:end,1:end-1)-G(1:end-1,1:end-1)).*getdAij((G(2:end,1:end-1)+
dA1,dA2,P.Aij,GGeom(end,1:end-1))));%multiplying the length
of an element with its area to get the volume
V=Vmain+VDraft+VGuide;%Adding all approximate volumes.
```

getGridSize.m

```
function deltaS=getGridSize(yPlusReq,rho,nu,L,U_inf)
Re=U_inf.*L./nu;
Cf=0.370.*(log10(Re)).^(-2.584);%Schultz-Grunov,
https://www.cfd-online.com/Wiki/Skin\_friction\_coefficient
wallStress=0.5.*Cf.*rho.*U_inf.^2;
u_wall=sqrt(wallStress/rho);
deltaS=yPlusReq.*nu./u_wall;
end
```

getNPSHA.m

```
function NPSHA=getNPSHA(hBooster,p_atm,p_vapour,
headDiffSubmergence,turbomachine)
global rho g
h_atm=p_atm./rho./g;
h_vapour=p_vapour./rho./g;
switch upper(turbomachine)
case 'TURBINE'
NPSHA=h_atm+hBooster+headDiffSubmergence-h_vapour;
case 'PUMP'
NPSHA=h_atm+hBooster+headDiffSubmergence-h_vapour;
otherwise
error('ERROR: no matchig value for variable
"turbomachine"')
end
```

getNPSHR.m

```
function NPSHR=getNPSHR(cm2,u2,Turbomachine)
global g
Turbomachine=upper(Turbomachine);
switch Turbomachine
case 'PUMP'
a=2; b=0.25;%Strictest
case 'TURBINE'
a=1.15;b=0.15;%Strictest
end
NPSHR=a.*cm2.^2/(2.*g)+b.*u2.^2/(2.*g);
end
```

getBLThickness.m

```
function delta=getBLThickness(U,x)
    global mu rho
    delta=5.5./sqrt(rho.*U.*x./mu).*x;
end
```
