Master's thesis

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

Sindre Trefall

Numerical Simulation of a Pico Turbine

Master's thesis in Mechanical Engineering
Supervisor: Reidar Kristoffersen
November 2021

NTNU
Norwegian University of
Science and Technology

Sindre Trefall

Numerical Simulation of a Pico Turbine

Master's thesis in Mechanical Engineering
Supervisor: Reidar Kristoffersen
November 2021

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

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis records the development and to a certain degree production of a pico turbine at the Department of Energy and Process Engineering (EPT) at the Norwegian University of Science and Technology (NTNU). The thesis is submitted as part of a two year study program and comprises of 30 ETCS. It proposes a simplistic design that can be analysed numerically (CFD), experimentally and theoretically (control volume analysis). The thesis draft intended to design the runner using CFD analysis and conduct a laboratory experiment to verify the results, but due to the COVID-19 pandemic the experiment is canceled. Several attempts of building a homemade test bench was conducted, but failed due to lack of available parts and equipment.

Sindre Trefall
Tromsø, 12.11.2021

Acknowledgment

The master thesis has been an exciting and a frustrating challenge, exhibiting all levels between both extremes. I would like to extend my gratitude to my supervisor Reidar Kristoffersen for introducing me to CFD by letting me participate in an OpenFOAM crash course for PhD-students. I would also thank him for proposing his observation as my thesis and for Wednesday meetings (both in person and on Skype). These were very enjoyable and a much needed social interaction during the COVID-19 lockdown. I would also like to extend my thanks to the EPT's IT-department that provided onsite help when I lost connection with my remote computer. Without their help I could not have continued my work after the campus lockdown on March 12, 2020.

In early May 2020 I became sick which resulted in a 19 month sick leave. This has complicated the writing of this thesis. In this regard I would like to thank my family and friends for their support. There have been times I thought the thesis would never be finished, but I am grateful for all the encouragement and help I have received.

Abstract

This master thesis documents the process of designing a pico turbine that supplies a controlling unit of a urinal with power using approximately 15 % of the head pressure. The thesis envelopes the construction of a simplistic runner, setup of a numerical model (OpenFOAM 7) and a partially finished design of a laboratory experiment intended to validate the numerical model. Because of the lockdown of Gløshaugen campus March 12, 2020, the laboratory experiment is canceled. The thesis documents the work with this experiment until this point, even though it is not finished.

The results from the CFD model have not been validated due to the cancellation of the laboratory experiment. To validate the results a control volume analysis is conducted, but the results from this analysis contradicts the results from the numerical model.

The unvalidated results from the CFD analysis shows that the turbine is utilizing only a small portion of the energy extracted from the flow. This yields an efficiency of only 4 %. The analysis also shows that the turbine does not produce a high enough pressure drop to be able to sustain the controller unit of a urinal with power.

Sammendrag

I denne masteroppgaven dokumenteres prosessen ved et design av en pico-turbin som skal levere nok effekt til å drive kontrollenheten til en urinal ved å bruke ca. 15 % av trykket oppstrøms av turbinen. Oppgaven omfatter konstruksjonen av et løpehjul med et forenklet design, oppsett av numerisk modell (OpenFOAM 7), samt planlegging og et delvis design av et laboratorieforsøk for å teste turbinen og validere dataene fra den numeriske modellen. På grunn av nedstengingen av Gløshaugen campus 12. mars 2020, ble laboratorieforsøket avlyst. Arbeid med laboratorieeksperimentet fram til dette tidspunktet er tatt med i oppgaven, selv om arbeidet ikke er ferdigstilt.

Resultatene fra CFD-modellen er ikke validert på grunn av avlysningen av laboratorieeksperimentet. I stedet er det gjort en kontroll-volum-analyse over løpehjulet, som bestrider resultatene fra CFD-modellen.

De uvaliderte resultatene fra CFD-analysen viser at turbinen benytter energien fra strømningen på en svært dårlig måte. Dette gjøre at virkningsgrad til turbinen bare er 4 %. Analysen viser også at turbinen ikke klarer å skape nok trykkfall til å kunne hente ut nok effekt til å drive kontrollenheten til en urinal.

Table of Contents

List of Tables

List of Figures

Abbreviations

AMI	=	Arbitrary Mesh Interface
CAD	=	Computer Aided Design
CFD	=	Computational Fluid Dynamics
CPU	=	Central Processing Unit
CV	=	Control Volume
DNS	=	Direct Numerical Simulation
FDM	=	Fused Deposition Modeling
FVM	=	Finite Volume Method
GAMG	=	Geometric Algebraic Multigrid
GUI	=	Graphical User Interface
HPC	=	High Performance Computing
IC	=	Initial Condition
LES	=	Large Eddy Simulation
MC	=	Monotonized Central limiter
MUSCL	=	Monotonic Upstream-centered Scheme for Conservation Laws
NTNU	=	Norwegian University of Science and Technology
OpenFOAM	=	Open-source Field Operation And Manipulation
PaT	=	Pump as Turbine
PISO	=	Pressure-Implicit with Splitting of Operators
RAM	=	Random Access Memory
RANS	=	Reynolds-averaged Navier–Stokes
RAS	=	Reynolds-Averaged Simulation
SHM	=	SnappyHexMesh
SIMPLE	=	Semi-Implicit Method for Pressure Linked Equations
SST	=	Shear Stress Transport
STL	=	Sterolitography
TVD	=	Total Variation Diminishing
UPM	=	Upwind Method

Variables

A	=	Area	$[m^2]$
A_{hub}	=	Cross sectional area of hub	$[m^2]$
A_{pipe}	=	Cross sectional area of pipe	$[m^2]$
C_L	=	Lift coefficient	$[-]$
C_0	=	Static load	$[kN]$
Co	=	Courant number	$[-]$
D_{hub}	=	Hub diameter	$[m]$
D_i	=	Inner diameter	$[m]$
D_o	=	Outer diameter	$[m]$
D_{pipe}	=	Pipe diameter	$[m]$
dF_θ	=	Force acting on CV	$[N]$
F_z	=	Force acting on runner in axial-direction	$[N]$
f_p	=	Fraction of the computation that can be parallel processed	$[N]$
I	=	Turbulence intensity	$[-]$
k	=	Turbulent kinetic energy	$[m^2 \cdot s^{-2}]$
L_{entry}	=	Entry length	$[m]$
L_{exit}	=	Exit length	$[m]$
L_{runner}	=	Runner length	$[m]$
n_{blades}	=	Number of blades	$[-]$
n_c	=	Number of processors	$[-]$
P	=	Power	$[W]$
P_{CFD}	=	Power calculated using CFD model	$[W]$
P_{CV}	=	Power calculated using control volume analysis	$[W]$
p	=	Kinematic pressure	$[m^2 \cdot s^{-2}]$
p_s	=	Static pressure	$[Pa]$
Re_D	=	Reynolds number	$[-]$
R_{hub}	=	Hub radius	$[m]$
R_{pipe}	=	Pipe radius	$[m]$
r	=	Radius	$[m]$
$\bar{r}$	=	Leverage arm	$[m]$
S	=	Distance between blades (2D simulation)	$[m]$
S_T	=	Theoretical speed-up	$[-]$
T	=	Temperature	$[^\circ C]$
T_z	=	Torque about z-axis	$[Nm]$
t_{blade}	=	Blade thickness	$[m]$
t_{hub}	=	Hub thickness	$[m]$
u_{inlet}	=	Inlet velocity	$[m \cdot s^{-1}]$
$\dot{V}$	=	Volumetric flow rate	$[m^3 \cdot s^{-1}]$

w_1	=	Relative velocity at leading edge of runner blade	$[m \cdot s^{-1}]$
w_2	=	Relative velocity at trailing edge of runner blade	$[m \cdot s^{-1}]$
y^+	=	Dimensionless length	$[-]$
α	=	Blade angle (Prototype A)	$[^\circ]$
α_{entry}	=	Flow inlet angle	$[^\circ]$
α_{exit}	=	Flow outlet angle	$[^\circ]$
α_1	=	Blade inlet angle	$[^\circ]$
α_2	=	Blade outlet angle	$[^\circ]$
Δp	=	Kinematic pressure drop	$[m^2 \cdot s^{-2}]$
Δp_s	=	Static pressure drop	$[Pa]$
η_{runner}	=	Runner efficiency	$[-]$
μ	=	Dynamic viscosity	$[Pa \cdot s]$
μ_t	=	Turbulent dynamic viscosity	$[Pa \cdot s]$
ν	=	Kinematic viscosity	$[m^2 \cdot s^{-1}]$
ν_t	=	Turbulent kinematic viscosity	$[m^2 \cdot s^{-1}]$
ρ	=	Density	$[kg \cdot m^{-3}]$
ω	=	Turbulence frequency	$[s^{-1}]$
ω_{rot}	=	Rotational speed	$[rad \cdot s^{-1}]$

Chapter 1

Introduction

In households and industry the use of wireless microelectronic monitoring devices have grown with the technological advances. These low powered systems are commonly energized by battery electric power due to batteries' high power density. On a larger scale application, magnetically generated energy has been produced by hydro power for over a century and played an integral part in the advance of industry. During the latest decades there has been significant scientific achievements within microscale power systems (Arnold, 2007), which sparks interest in combining low powered monitoring devices with hydro power.

This thesis started with an observation by professor Reidar Kristoffersen. He observed that the supply-battery for the controlling unit for the urinal at the campus-building needed to be changed on a regular basis. This seems unnecessary if the water supply to the urinal can maintain the battery's energy level. This started the thought process of making a turbine that can be mounted to the urinal's water supply and siphon enough energy to make the urinal self-sustainable.

Removing the need of new batteries is one of the challenges the society is facing. Batteries are made of resources that are limited on a global scale. With the increase in demand of batteries, these should be used efficiently. This also opens a niche market where devices close to a water supply can be sustained using a small turbine. An industrial application of this concept is particularly interesting where the turbine is installed parallel to a valve and the pressure drop is utilized to sustain monitoring devices.

Most hydro power research conducted revolve around large hydro power plants that supplies the power grid with electricity. For smaller projects, research within creating power plants for rural villages has grown. Due to the humanitarian nature of these projects the cost and knowledge required to make them function is of peak interest (Greacen and Kerins, 2010). This has created the research field where commercial pumps are reversed and used as turbine, often labeled as pump as turbine (PaT). The benefit of using PaT is

that pumps are manufactured in a larger scale than turbines and thus much cheaper. The drawback is the lower efficiency, however this is where researchers are trying to optimize performance for a wide range of given scenarios. Smaller hydro turbines are less common as a product, but are often made as experiments that later will be scaled up for conventional use.

Using Computational Fluid Dynamics (CFD) model during the design phase of a product has grown in popularity with the technological advances. In this thesis a Reynolds-average Navier-Stokes (RANS) model with the capability of moving mesh-sections is used to simulate the flow. Other models such as Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS) are available, but require more knowledge and computational power to use. Due to the approximated nature of CFD models it is customary to compare the numerical results with known theory, results from a laboratory experiment or other simulations that have gone through a rigorous set of reviews and are thereby both verified and validated. The conclusion will present all findings during this thesis without censorship or external agendas.

This thesis will investigate if it is possible to sustain the controlling unit of a urinal by extracting 1-10 W that is available in the water supply without producing a large pressure drop using a one inch turbine. This will be conducted in five steps.

1. Design a runner and turbine casing.
2. Create a CFD model in OpenFOAM 7 by modifying the *Propeller* tutorial.
3. Build and conduct a laboratory experiment for the designed hydro turbine.
4. Compare the results from the CFD model and the laboratory experiment.
5. Conclude with the findings of this thesis.

Chapter 2

Theory

2.1 Hydro Power Turbines

The most common hydro turbines in conventional hydro power plants are Pelton, Francis and Kaplan turbines. The selection of turbine is primarily based on the available head and volume flow. Figure 2.1 shows the operation areas for Pelton, Francis and Kaplan turbines.

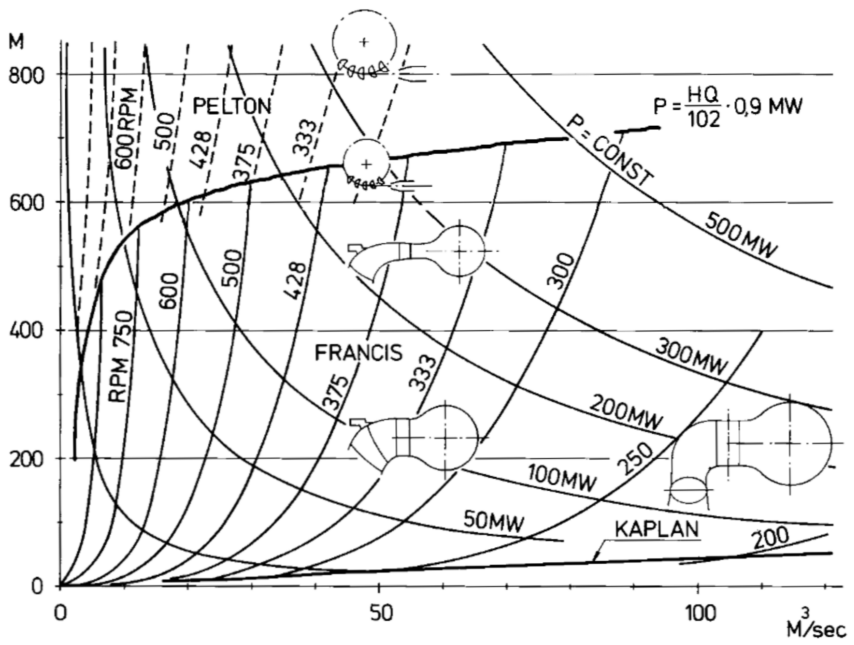


Figure 2.1: Operation areas for Pelton, Francis and Kaplan turbines (Brekke, 2003, page 3, figure 1.1).

The Pelton turbines are impulse turbines, where the pressure is converted into kinetic energy using nozzles. The water jets transfer the energy to the runner that is connected to a shaft driving the generator. Francis and Kaplan turbines are reactionary turbines, meaning approximately 50 % of the energy extracted from the stream is due to the pressure drop. The main components of the Francis and the Kaplan turbines are spiral casing, stay vanes, guide vanes, runner, draft tube and shaft. The guide vanes are adjustable to regulate and optimize the flow and induces a rotating flow that the runner can utilize leaving a non-rotating flow downstream of the runner. The Francis turbine has a radial inlet and an axial outlet, while the Kaplan has both an axial inlet and outlet.

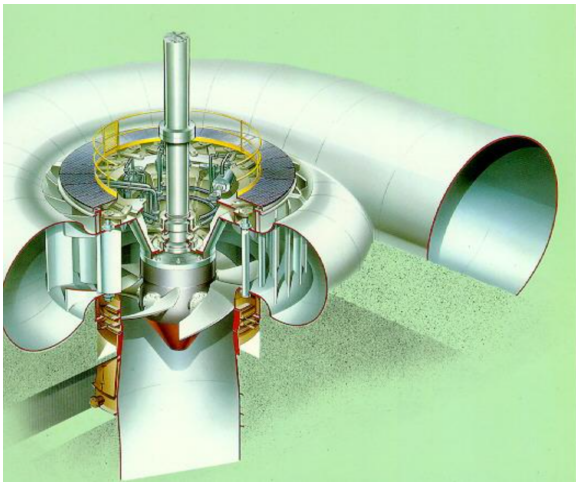


Figure 2.2: Kaplan turbine (Nielsen, 2014, page 11).

Another type of Kaplan turbine is the Kaplan bulb turbine. This is a variation where the turbine is used in a pipe without a spiral casing and stay vanes.

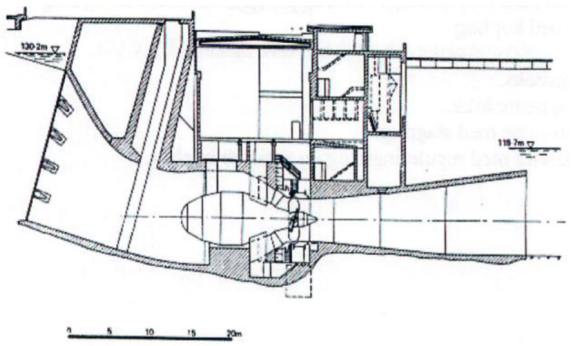


Figure 2.3: Kaplan bulb turbine (Brekke, 2008, page 132).

2.2 Control Volume Analysis

Control volume (CV) analysis is a useful tool to investigate gross flow effects happening within a defined area. This is useful as the analysis only evaluates the conditions on the border of the control volume, without computing the flow within the control volume.

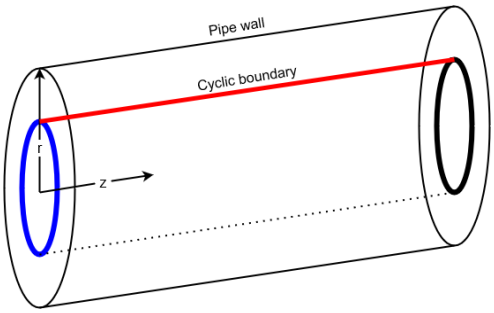
In this thesis the results from a control volume analysis will be used to compare the CFD results in order to estimate the validity of the CFD model. This comparison is done by evaluating the power estimated by both calculations. Control volume analysis uses the linear momentum equation which is the equivalent of Newton's second law (White, 2016, page 133).

$$\frac{d}{dt}(m\mathbf{V})_{sys} = \sum \mathbf{F} = \frac{d}{dt}\left(\int_{CV} \mathbf{V}\rho\, dV\right) + \int_{CS} \mathbf{V}\rho(\mathbf{V}_r \cdot \mathbf{n})\, dA \tag{2.1}$$

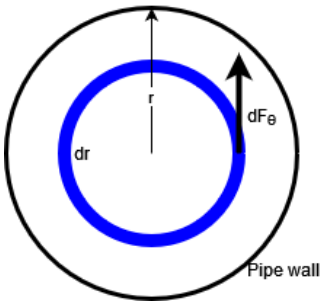
This is used to find the required external forces that is needed to maintain a non-accelerating control volume. The linear momentum equation states that a flow that changes its direction must have a resulting external force acting on the control volume.

2.2.1 Control Volume Analysis of Runner

For a runner blade, a control volume can be established by using the surface area enclosed by the circumference at radius r and the length of the runner L_{runner}. This creates a cylindrical slice that can be flattened into a rectangle. By investigating only the flow between two blades this creates a control volume with inlet and outlet patches and two surfaces (upper and lower part of the blade). To simplify, the flow is considered incompressible and inviscid, and the blades are assumed rigid and infinitesimally thin.



(a) Isometric view of the cylindrical slice of one runner blade.



(b) Front view with a cylindrical slice made into a control volume.

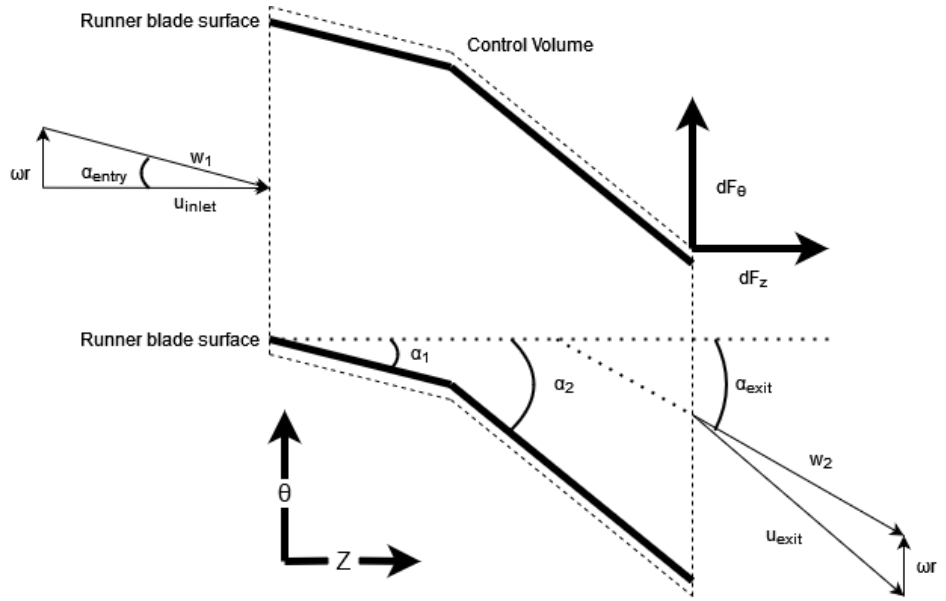


Figure 2.5: Velocity triangle with stationary control volume diagram.

In the turbine a section dr can be investigated where the flow enters with the absolute velocity u_{inlet} and exits the turbine with the absolute velocity u_{exit}. Due to the rotation of the runner it is easier to use relative velocity where the runner's rotational velocity is accounted for yielding inlet velocity w_1 and outlet velocity w_2. Assuming incompressible flow, the continuity dictates that $w_1 = w_2 = w$ because the inlet and outlet patch are of equal area. The change of direction in the flow yields a force that given the leverage arm r produces a torque that drives the runner. Given flow with the inlet angle α_{entry} and assuming inviscid flow, where the flow exit angle α_{exit} is equal to the blade outlet angle α_2, the force acting on the hollow cylindrical section is

$$dF_\theta = \dot{m}\left(\frac{dA}{A}\right)|w|\left[sin(\alpha_2) - sin(\alpha_{entry})\right]) \tag{2.2}$$

where $\dot{m}\frac{dA}{A}$ is the mass flow rate that passes the cross sectional area dA. This calculates the force dF_θ acting on that section of the runner blade.

$$dA = \pi \cdot ((r + \frac{1}{2}dr)^2 - (r - \frac{1}{2}dr^2)) \tag{2.3}$$

$$A = \pi \cdot r^2 \tag{2.4}$$

Knowing the distance r from the center of the pipe to the center of the runner blade section gives the torque that drives the runner.

$$T_z = dF \times r = |dF| \cdot |r| sin(\theta) \tag{2.5}$$

To estimate the power produced by all the turbine sections. The torque contribution from each section is multiplied by the rotational velocity and summed up.

$$P = \sum_{r=r_{hub}}^{r_{pipe}} T_z(r) \cdot \omega_{rot} dr \tag{2.6}$$

Leverage Arm

The distributed load represented for each section by equation (2.2) can be assumed to be a continuous function dependent on r^2. This distributed load can be transformed into a point load at $\bar{r}$ in the radial direction. $\bar{r}$ is found using

$$\bar{r} = \frac{\int_{R_{hub}}^{R_{pipe}} \zeta \cdot r^3 dr}{\int_{R_{hub}}^{R_{pipe}} \zeta \cdot r^2 dr} = \frac{3(R_{pipe}^4 - R_{hub}^4)}{4(R_{pipe}^3 - R_{hub}^3)} \tag{2.7}$$

$$\frac{\bar{r}}{R_{pipe}} = 0.78 \tag{2.8}$$

where ζ is all independent factors of radius r in equation (2.2). This is used during the verification in section 6.2

2.3 Entry Length for Turbulent Pipe Flow

The entry length of an inlet pipe is important to ensure a fully developed flow in an empirical experiment. This is determined by its correlation to the Reynolds number (White, 2016, page 313).

$$\frac{L_{entry}}{D_{pipe}} \approx 1.6 Re_D^{\frac{1}{4}} = 1.6 \left(\frac{u_{inlet} D_{pipe}}{\nu} \right)^{\frac{1}{4}} \qquad \text{for} \qquad Re_D \leq 10^7 \tag{2.9}$$

where L_{entry} is the entry length, u_{inlet} is the bulk velocity, D_{pipe} is the pipe diameter and ν is the kinematic viscosity.

2.4 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) investigates the changes within the flow on a small scale (differential form). This can reproduce flows by solving the Navier-Stokes equation numerically. This yields a detailed picture of the flow within the computed domain, but is based on approximations. This dictates that CFD results must be verified and validated by empirical results.

2.4.1 Governing Equation

Navier-Stokes equation for mass continuity

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.10}$$

Navier-Stokes equation for conservation of momentum on differential form

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{\nabla p}{\rho} + \mathbf{g} + \nu \nabla^2 \mathbf{u} \tag{2.11}$$

2.4.2 Finite Volume Method

Finite Volume Method (FVM) is a CFD technique using control volume integration. Section 2.4.2 - 2.4.4 represents a summary of the discretization given in "An Introduction to Computational Fluid Dynamics" (Versteeg and Malalasekera, 2007).

$$\frac{\partial}{\partial t}\int_{CV} \rho\phi \, dV + \int_A (\rho \mathbf{u}\phi)\cdot \mathbf{n} \, dA - \int_A \mu(\nabla \phi)\cdot \mathbf{n} \, dA - \int_V \rho \mathbf{S}_\phi dV = -\int_A p \cdot \mathbf{n} dA \tag{2.12}$$

The discretization of the right hand side (transient convection diffusion) is summarized in this section, while the left hand side is coupled to the equation in section 2.4.4. By integrating the 1D transient convection diffusion equation over space and time yields

$$\frac{\rho \Delta V}{\Delta t}(\phi_P - \phi_P^0) + (\rho u A \phi)_e - (\rho u A \phi)_w = \left(\Gamma A \frac{d\phi}{dx}\right)_e - \left(\Gamma A \frac{d\phi}{dx}\right)_w + \bar{S}\Delta V \tag{2.13}$$

where $\bar{S}$ is the average of the source strength in the control volume. The capital suffixes denotes at cell center while the lowercase suffixes denotes at cell face. To evaluate the equation the gradients in the diffusion term must be approximated.

$$\frac{d\phi_e}{dx} \approx \frac{\phi_P - \phi_E}{\delta x} \qquad \frac{d\phi_w}{dx} \approx \frac{\phi_W - \phi_P}{\delta x} \tag{2.14}$$

By implementing the approximation and assuming that all faces have a equal face area and rearranging, the following equation is obtained

$$\frac{\rho \Delta x}{\Delta t}(\phi_P - \phi_P^0) + (\rho u \phi)_e - (\rho u \phi)_w = \left(\frac{\Gamma}{\delta x}\right)_e (\phi_P - \phi_E) - \left(\frac{\Gamma}{\delta x}\right)_w (\phi_W - \phi_P) + \bar{S}\Delta x \tag{2.15}$$

To simplify the equation a variable for convective mass flux F, and diffusion conductance D is introduced to the equation.

$$F_e = (\rho u)_e \qquad F_w = (\rho u)_w \tag{2.16}$$

$$D_e = \left(\frac{\Gamma}{\delta x}\right)_e \qquad D_e = \left(\frac{\Gamma}{\delta x}\right)_w \tag{2.17}$$

The source term $\bar{S}\Delta V$ can be substituted into a linear function if the source is a function of the dependent variable. This yields a uniform source term which is equal for all cells and a non-uniform term which is cell specific.

$$\bar{S}\Delta V = S_u + S_p\phi_p \tag{2.18}$$

This leads to the equation

$$\frac{\rho\Delta x}{\Delta t}(\phi_P - \phi_P^0) + F_e\phi_e - F_w\phi_w = D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W) + S_u + S_p\phi_p \tag{2.19}$$

The unknowns in this equation are the cell face values ϕ_e and ϕ_w. These two must be computed using a scheme for the convection terms. In OpenFOAM several schemes are available. The initial choice to calculate the convective fluxes is the first order upwind method (UPM). This is known as a robust scheme, but lacks in accuracy due to artificial diffusion (Versteeg and Malalasekera, 2007, page 150). To gain higher accuracy, second order schemes can be applied. These schemes have less robustness, but higher accuracy. This demands a stricter requirement to the initial conditions (IC) and in some cases it is beneficial to run a first order simulation and use the results as initial condition for the second order simulation. When the cell face values are found the solution can be written on general form with coefficients depending on chosen schemes.

$$a_P\phi_P = \sum_{nb} a_{nb}\phi_{nb} + S_u + a_p^0\phi_P^0 \tag{2.20}$$

where

$$a_p = \sum_{nb} a_{nb} - S_P + a_p^0 \tag{2.21}$$

and

$$a_p^0 = \frac{\rho\Delta x}{\Delta t} \tag{2.22}$$

First Order Upwind Scheme

To determine the value at the cell faces it must be established how this value is calculated. The first order upwind scheme uses the value at the upstream cell center as its cell face value. This is a valid option for convective flows where the direction of the flow is known because this is mandatory to determine which cell is upstream. In a stream going from west to east ($u > 0$) the cell face values are as following.

$$\phi_w = \phi_W \qquad\qquad \phi_e = \phi_E \tag{2.23}$$

Substituting the new cell face values into (2.19)

$$\frac{\rho\Delta x}{\Delta t}(\phi_P - \phi_P^0) + F_e\phi_P - F_w\phi_W = D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W) + S_u + S_P\phi_P \tag{2.24}$$

And rearranging into general form

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + S_u + a_p^0\phi_P^0 \tag{2.25}$$

Giving the central coefficient

$$a_P = a_W + a_E + (F_e - F_w) - S_P + a_p^0 \tag{2.26}$$

By adding $(F_e - F_w = 0)$ to the central coefficient and the maximum function to the neighbouring coefficient the implementation is valid for both $u > 0$ and $u < 0$. With the neighbouring coefficients

$$a_W = max(F_w, D_W + \frac{F_w}{2}, 0) \qquad a_e = D_e + max(-F_e, D_e - \frac{F_e}{2}, 0) \tag{2.27}$$

MUSCL

MUSCL is the acronym for Monotonic Upstream-Centered Scheme for Conservation Laws. This method uses the neigbouring points in order to reconstruct the values at the cell faces. This yield a higher order of accuracy, but can have problems with oscillations near discontinuities. To combat this numerical phenomenon Total Variation Diminishing (TVD) schemes are implemented where limiters prevents the solver from creating unphysical extremas which can cause instability (Versteeg and Malalasekera, 2007, 165). The method implements a deffered correction source term (S^{DC}) that enforces a first order scheme near discontinuities and a second order scheme elsewhere.

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + S_u + S_u^{DC} + a_P^0\phi_P^0 \tag{2.28}$$

With central coefficient

$$a_P = a_W + a_E + (F_e - F_w) - S_P + a_P^0 \tag{2.29}$$

Neighbouring coefficients

$$a_W = max(F_w, D_W + \frac{F_w}{2}, 0) \qquad a_e = D_e + max(-F_e, D_e - \frac{F_e}{2}, 0) \tag{2.30}$$

The deffered correction source term

$$\begin{aligned} S_u^{DC} = &\frac{1}{2}F_e[(1-\alpha_e)\Psi(r_e^-) - \alpha_e \cdot \Psi(r_e^+)](\phi_E - \phi_P) \\ &+ \frac{1}{2}F_w[\alpha_w\Psi(r_w^-) - (1-\alpha_w) \cdot \Psi(r_w^+)](\phi_P - \phi_W) \end{aligned} \tag{2.31}$$

To maintain bi-directional validity α is introduced

$$\alpha_w = 1 \quad \text{for} \quad F_w > 0 \qquad \alpha_e = 1 \quad \text{for} \quad F_e > 0 \tag{2.32}$$

$$\alpha_w = 0 \quad \text{for} \quad F_w < 0 \qquad \alpha_e = 0 \quad \text{for} \quad F_e < 0 \tag{2.33}$$

where r_e^+, r_e^+, r_e^- and r_W^-

$$r_e^+ = \frac{\phi_P - \phi_W}{\phi_E - \phi_P} \qquad r_w^+ = \frac{\phi_W - \phi_{WW}}{\phi_P - \phi_W} \tag{2.34}$$

$$r_e^- = \frac{\phi_{EE} - \phi_E}{\phi_E - \phi_P} \qquad r_w^- = \frac{\phi_E - \phi_P}{\phi_P - \phi_W} \tag{2.35}$$

The limiter function $\Psi(r)$ has several options, the implementation of the *MUSCL* scheme in OpenFOAM uses the Monotonized Central limiter (openfoam.com, 2019).

$$\Psi^{MC}(r) = max(min(min(2 * r, 0.5 * r + 0.5), 2), 0) \qquad \lim_{r\to\infty} = 2 \tag{2.36}$$

The discretization stated above will solve the transient convection diffusion equation, but in order to solve the Navier-Stokes equation the pressure gradient must be included. In this thesis, this is done by coupling the velocity and the pressure using the *PIMPLE* algorithm.

2.4.3 3D Descretization

The discretizaton performed in the previous chapters can be extended from 1D space (east and west) into 3D space (east, west, north, south, front and back). This yields solution to the transient convection-diffusion equation on general form:

$$a_p\phi_P = a_W\phi_W + a_E\phi_E + a_S\phi_S + a_N\phi_N + a_B\phi_B^0 + a_F\phi_F^0 + a_P^0\phi_P^0 + S_u \tag{2.37}$$

where the central coefficient

$$a_p = \sum_{nb} a_{nb} + a_P^0 + \Delta F - S_P \tag{2.38}$$

where

$$a_P^0 = \frac{\rho\Delta V}{\Delta t} \tag{2.39}$$

and

$$\Delta F = (F_e - F_w) + (F_n - F_s) + (F_f - F_b) \tag{2.40}$$

The neighbouring coefficients (a_E, a_W, a_S, a_N, a_F and a_B) are determined by the chosen scheme for the convective fluxes as shown in chapter 2.4.2.

2.4.4 Pressure-Velocity Coupling

To couple the pressure and the velocity fields a "guess and correct" method is used. The *PIMPLE* algorithm combines the transient *PISO* algorithm with the steady state *SIMPLE* algorithm. The *PISO* algorithm is restricted by the Courant criterion $|Co| = \frac{|u|\Delta t}{\Delta x} \leq 1$ which can prove computational costly due to the small time step. For the *PIMPLE* algorithm to handle Courant numbers larger than 1, and thus larger time steps, an outer loop adopted from the *SIMPLE* algorithm is implemented to find a steady state solution at the given time level before advancing in time. To converge, the *SIMPLE* algorithm requires diagonal dominance (Versteeg and Malalasekera, 2007, page 143).

$$\frac{|\sum a_{nb}|}{|a_p'|} \begin{cases} \leq 1 \text{ for all nodes} \\ < 1 \text{ for at least one node} \end{cases} \tag{2.41}$$

where $a_p' = a_p - b_i - S_p$ is the net central coefficient and b_i is additional source terms. The algorithms uses a staggered grid where the pressure is located at the centroid and the velocities are located at the cell faces (Harlow and Welch, 1965).

SIMPLE Algorithm

The *SIMPLE* algorithm is an iterative procedure that uses a guessed fields denoted * to calculate a corrector denoted $'$. The algorithm couples pressure and velocity for each time step to create a steady state solution that satisfies continuity.

$$u = u^* + u' \tag{2.42}$$

$$v = v^* + v' \tag{2.43}$$

$$p = p^* + p' \tag{2.44}$$

The correction for the velocity field is found using the current and the guessed pressure field

$$a_{i,J}u'_{i,J} = \overbrace{\sum_{nbf} a_{nbf}u'_{nbf}}^{\approx 0} + \left(\overbrace{(p_{I-1,J} - p^*_{I-1,J})}^{p'_{I-1,J}} - \overbrace{(p_{I,J} - p^*_{I,J})}^{p'_{I,J}} \right) A_{i,J} \tag{2.45}$$

$$a_{i,J}v'_{i,J} = \overbrace{\sum_{nbf} a_{nbf}v'_{nbf}}^{\approx 0} + \left(\overbrace{(p_{I,J-1} - p^*_{I,J-1})}^{p'_{I,J-1}} - \overbrace{(p_{I,J} - p^*_{I,J})}^{p'_{I,J}} \right) A_{I,j} \tag{2.46}$$

The *SIMPLE* algorithm's main assumption is to neglect the contribution to the corrector from the neighbouring cells. This is justified because this contribution goes towards zero when the guessed velocity field converges towards the correct velocity field for each iteration.

$$u'_{i,J} = \frac{A_{i,J}}{a_{i,J}} \left((p_{I-1,J} - p^*_{I-1,J}) - (p_{I,J} - p^*_{I,J} \right) \tag{2.47}$$

$$v'_{I,j} = \frac{A_{I,j}}{a_{I,j}} \left((p_{I,J-1} - p^*_{I,J-1}) - (p_{I,J} - p^*_{I,J} \right) \tag{2.48}$$

calculating corrected velocity field

$$u_{i,J} = u^*_{i,J} + u'_{i,J} = u^*_{i,J} + \frac{A_{i,J}}{a_{i,J}} \left((p_{I-1,J} - p^*_{I-1,J}) - (p_{I,J} - p^*_{I,J} \right) \tag{2.49}$$

$$v_{I,j} = v^*_{I,j} + v'_{I,j} = v^*_{I,j} + \frac{A_{I,j}}{a_{I,j}} \left((p_{I,J-1} - p^*_{I,J-1}) - (p_{I,J} - p^*_{I,J} \right) \tag{2.50}$$

The inconsistency caused by disregarding for the terms $\sum_{nb} a_{nb}u'_{nb}$ and $\sum_{nb} a_{nb}v'_{nb}$ in equation (2.45) and (2.46) can cause the algorithm to become unstable. This is solved by inflating a_p by dividing by the under-relaxation factor α_u and α_v in equation (2.51) and (2.52), where $0 < \alpha < 1$.

$$\frac{a_{i,J}}{\alpha_u}u_{i,J} = \sum_{nb} a_{nb}u_{nb} + A_{i,J}(p_{I-1,J} - p_{I,J}) + b_{i,J} + \left[(1 - \alpha_u)\frac{a_{i,J}}{\alpha_u} \right] u^{n-1}_{i,J} \tag{2.51}$$

$$\frac{a_{I,j}}{\alpha_v}v_{I,j} = \sum_{nb} a_{nb}v_{nb} + A_{I,j}(p_{I,J-1} - p_{I,J}) + b_{I,j} + \left[(1 - \alpha_v)\frac{a_{I,j}}{\alpha_u} \right] v^{n-1}_{I,j} \tag{2.52}$$

where u and v are the corrected velocities. Due to the correction to the velocity field the pressure must also be corrected to maintain continuity. The pressure correction term is calculated using the continuity equation using $u = u^* + u'$ and $v = v^* + v'$.

$$\begin{aligned} &\Big[\rho(u^* + u')A\Big]_{i+1,J} - \Big[\rho(u^* + u')A\Big]_{i,J} \\ &+\Big[\rho(v^* + v')A\Big]_{i,J+1} - \Big[\rho(v^* + v')A\Big]_{i,J} = 0 \end{aligned} \tag{2.53}$$

which leads to

$$p'_{I,J} = \frac{\sum_{nbc} a_{nbc} p'_{nbc} + \Big[(\rho u^* A)_{i,J} - (\rho u^* A)_{i+,j} + (\rho v^* A)_{I,j} - (\rho v^* A)_{I,j+1}\Big]}{\sum_{nbc} a_{nbc}} \tag{2.54}$$

where nbc denotes the neighbouring cell centers.

2.4.5 Handling of Non-Orthogonal Mesh

A non-orthogonal mesh is caused when two or more cells have a cell face f which leads to a misalignment between the cell face unit normal vector $\mathbf{n}$ and the centroid vector $\mathbf{d}$ (from point P to N).

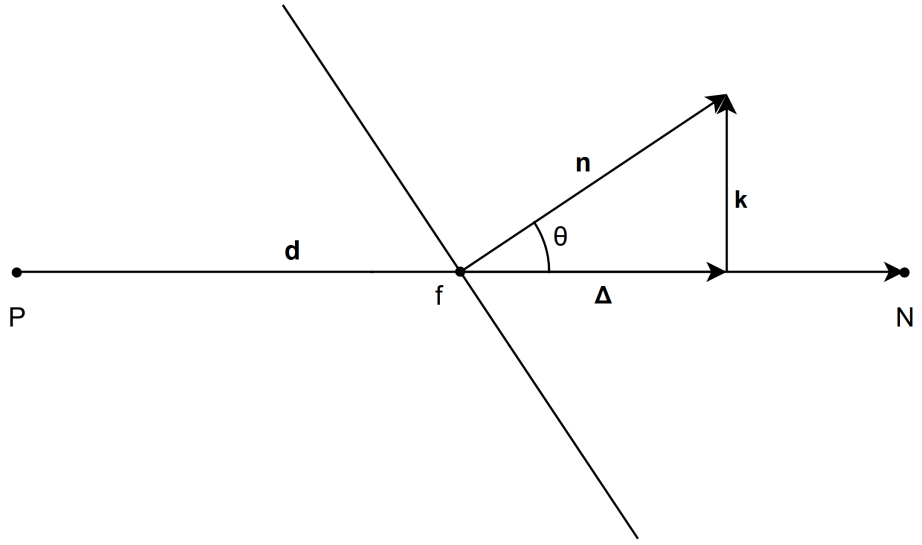


Figure 2.6: Sketch of non-orthogonal cells, in accordance with (Jasak, 1996, figure 3.4, page 84)

The angle θ between the centroid vector and the connecting face unit normal vector is given by

$$\theta_{ort} = cos^{-1}\left(\frac{\mathbf{d}\cdot\mathbf{n}}{|\mathbf{d}|\cdot|\mathbf{n}|}\right) \tag{2.55}$$

This causes a problem during the discretization of the diffusion term in the momentum equation.

$$\int_V [\nabla \cdot (\mu \nabla \mathbf{u}] dV = \sum_f [\mu_f (\nabla \mathbf{u}) \cdot \mathbf{n}] A_f \tag{2.56}$$

If $\theta_{ort} > 0^\circ$ there is a non-orthogonal contribution. OpenFOAM solves the problem by splitting the unit normal vector to the surface into an orthogonal and a non-orthogonal vector (Jasak, 1996, page 83-85).

$$\mathbf{n} = \mathbf{\Delta} + \mathbf{k} \tag{2.57}$$

$$\int_V [\nabla \cdot (\mu \nabla \mathbf{u}] dV = \overbrace{\sum_f [\mu_f (\nabla \mathbf{u}) \cdot \mathbf{\Delta}_f] A_f}^{orthogonal} + \overbrace{\sum_f [\mu_f (\nabla \mathbf{u}) \cdot \mathbf{k}_f] A_f}^{non-orthogonal} \tag{2.58}$$

The orthogonal contribution is determined implicitly in the *PIMPLE*-loop by

$$(\nabla \mathbf{u}) \cdot \mathbf{\Delta} = \frac{\mathbf{u}_P - \mathbf{u}_{nb}}{|d|} \mathbf{\Delta}_f \tag{2.59}$$

and the non-orthogonal contribution is determined explicitly using the values of the last time step

$$(\nabla \mathbf{u}) \cdot \mathbf{k} = \frac{\mathbf{u}_P^{n-1} - \mathbf{u}_{nb}^{n-1}}{|d|} \mathbf{k}_f \tag{2.60}$$

OpenFOAM limits the contribution of the non-orthogonal term using the factor $0 \leq \gamma \leq 1$.

$$(\nabla \mathbf{u})_f \cdot k_f \leq \gamma \left(\frac{\mathbf{u}_P - \mathbf{u}_N}{|d|} |\Delta|_f \right) \tag{2.61}$$

The non-orthogonal term is therefore added as a source term. This causes a smaller $a'_p = a_p - b_p - S_p$ where the non-orthogonal contribution is lumped into the source term b_p. The smaller a'_p can lead to instability if the matrix is lacking diagonal dominance (eq. 2.41). To prevent a too large contribution, the non-orthogonal contribution is limited to be less than orthogonal contribution. This is practical for meshes with a few cells with high non-orthogonality that causes instability. At the sacrifice of accuracy in the non-orthogonal term of these cells, the stability is improved which can yield a converged solution. In PimpleFOAM the *nNonOrthogonalCorrectors* found in *fvSolution* sets the number of recalculations of the pressure field.

2.4.6 Turbulence Modeling

To reproduce turbulence in a CFD model, the turbulence scales must be resolved. This is know as Direct Numerical Simulation and is computationally demanding type of simulation. By developing the RANS equations using the Reynolds decomposition, where the instantaneous variable ψ is separated into a time mean quantity $\bar{\psi}$ and a fluctuating quantity ψ'.

$$\psi = \bar{\psi} + \psi' \tag{2.62}$$

This allows the Reynolds stress tensor, represents the turbulent effects, to be separated from the rest of the equation and modelled at a cost of accuracy.

$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{\partial x_j} = \bar{f}_i + \frac{\partial}{\partial x_j}\left[-\frac{\bar{p}}{\rho}\delta_{ij} + \nu\left(\frac{\partial \bar{u}_j}{\partial x_i} + \frac{\partial \bar{u}_i}{\partial x_j}\right) - \overline{u_i' u_j'}\right] \tag{2.63}$$

To model the unresolved turbulence the Boussinesq's approximation simplifies the Reynolds stress tensor using an artificial contribution to the viscosity called turbulent viscosity ν_t.

$$-\overline{u_i' u_j'} \approx \nu_t\left(\frac{\partial \bar{u}_j}{\partial x_i} + \frac{\partial \bar{u}_i}{\partial x_j}\right) - \frac{2}{3}k\delta_{ij} \tag{2.64}$$

The Boussinesq's approximation opens a range of possibilities in regard to how the turbulent viscosity is calculated. The two-equation model k-ω-$Shear\,Stress\,Transport$ (k-ω-SST) proposed by Menter (Menter and Esch, 2001) is selected in this thesis . It uses the turbulent kinetic energy k and the turbulence frequency ω. The model equations are summarized below with some changes based on the OpenFOAM k-ω-SST source code.

$$\nu_t = \frac{\mu_t}{\rho} = \frac{f(k,\omega)}{\rho} \tag{2.65}$$

The model requires two additional transport equations (eq. 2.66 and eq. 2.69) to be solved.

The transport equation for turbulent kinetic energy:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho \bar{u}_j k}{\partial x_j} = \tilde{P}_k - \beta^* \rho\omega k + \frac{\partial}{\partial x_j}\left[(\mu + \alpha_k \mu_t)\frac{\partial k}{\partial x_j}\right] \tag{2.66}$$

where the production term has been limited to prevent turbulence in stagnation regions (Menter et al., 2003, page 2).

$$\tilde{P}_k = min(P_k,\, 10\beta^*\rho k\omega) \tag{2.67}$$

$$P_k = \mu_t \frac{\partial \bar{u}_i}{\partial x_j}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{2.68}$$

The transport equation for turbulence frequency:

$$\begin{aligned}\frac{\partial \rho\omega}{\partial t} + \frac{\partial \rho \bar{u}_j \omega}{\partial x_j} = \frac{\gamma}{\nu_t}P_k - \beta\rho\omega^2 + \frac{\partial}{\partial x_j}\left[\left(\mu + \alpha_\omega \mu_t\right)\frac{\partial \omega}{\partial x_j}\right] \\ + 2(1-F_1) + \frac{\rho}{\omega}\alpha_{\omega_2}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}\end{aligned} \tag{2.69}$$

where γ and β are model constants.

The k-ω-SST model is developed based on the strengths of the k-ϵ model and the k-ω model, where the k-ϵ model is used outside of the boundary layer and a blending function F_1 is used to give a gradual change to the k-ω model inside the boundary layer.

$$F_1 = tanh(arg_1^4) \tag{2.70}$$

$$arg_1 = min\left(max\left[\left(\frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\nu}{d^2\omega}\right), \frac{4\rho\alpha_{\omega_2} k}{CD_{k\omega} d^2}\right]\right) \tag{2.71}$$

$$CD_{k\omega} = max\left(2\rho\alpha_{\omega_2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-20}\right) \tag{2.72}$$

where d is the distance normal to closest wall, and $\beta^* = 0.09$ and $\alpha_{\omega,2}$ are model constants. The model constants are weighted when $0 < F_1 < 1$ according to

$$\phi = F_1\phi_1 + (1 - F_1)\phi_2 \tag{2.73}$$

In the k-ω-SST model source code there is a remark that in Menter's paper (Menter and Esch, 2001, eq. 2) there is an error in the last term (Foundation, 2010, line 65). The source code has corrected this error in accordance with Menter's paper (Menter et al., 2003, eq. 1). The model constants are also updated with the latter paper, due to the α-notation. The source code dispute the blending of σ_k and σ_ω, arguing that it is not consistent with the blending of k-ϵ and k-ω, and proposes a blend of α_k and α_ω. Table 2.1 summarizes the use of constants for each turbulence model.

Table 2.1: Two-equation model constants.

Model	F_1	ϕ_i	$\alpha_{k,i} = \frac{1}{\sigma_{\omega,i}}$	$\alpha_{\omega,i} = \frac{1}{\sigma_{k,i}}$	β_i	γ_i
k-ω	0	ϕ_1	0.85	0.5	$3/40$	$5/9$
k-ϵ	1	ϕ_2	1	0.856	0.0828	0.44

The turbulent viscosity is calculated using

$$\mu_t = \frac{\rho a_1 k}{max(a_1\omega, SF_2)} \tag{2.74}$$

$$F_2 = tanh(arg_2^2) \tag{2.75}$$

$$arg_2 = max\left(2\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right) \tag{2.76}$$

where $a_1 = 0.31$ and F_2 is a blending function.

2.4.7 Dynamic Mesh

Dynamic mesh is a useful tool when investigating flows with moving boundaries. This operation can change both the topology of the geometries and/or the mesh without inflicting discretization errors (Jasak, 2009, page 2). The moving mesh FVM uses the integral form of the governing equations similar to equation (2.12) where the volume is moving with the velocity $\mathbf{u}_s$.

$$\frac{\partial}{\partial t}\int_V \rho\phi\partial V + \oint_A \rho\cdot(\mathbf{u} - \mathbf{u}_s)\phi\partial A - \oint_A \mu\mathbf{n}\cdot\nabla\phi\partial A = \int s_\phi \partial V \tag{2.77}$$

This allows a mesh region to move with respect to the rest of the domain. This is shown in figure 2.7 where a mixer is rotated.

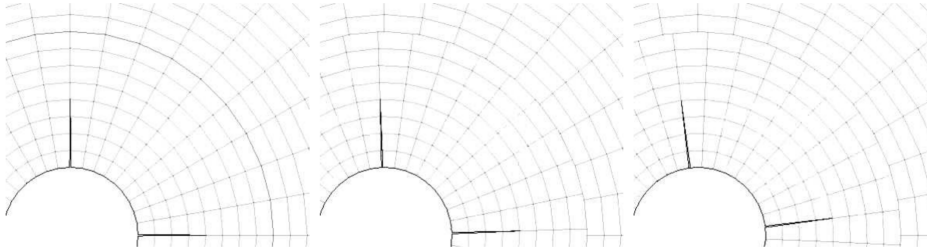


Figure 2.7: Rotating mesh with sliding interface (Jasak, 2009, page 5, figure 5)

The interface between the rotating and non-rotating mesh is called an Arbitrary Mesh interface (AMI) and is used to couple the meshes into one domain. The PimpleFOAM solver has the dynamic mesh feature which is controlled by adding the *dynamicMeshDict*-file to the *constant* folder. This file states which mesh region is rotated, origin of rotation, axis of rotation and rotational speed ω_{rot}.

2.5 Kinematic Pressure

OpenFOAM's PimpleFOAM is developed for incompressible flow. The constant density is therefore divided in all terms in the Navier-Stokes equations which produces the kinematic pressure term p.

$$p = \frac{p_s}{\rho} \tag{2.78}$$

This is of key importance during post processing because the density must be factored in to yield the static pressure p_s.

2.6 3D Printing

The runner can be manufactured using additive manufacturing techniques (3D printing) such as Fused Deposition Modeling (FDM) or stereolithography. The main difference between the two additive manufacturing methods are the cost and knowledge requirement versus detail. FDM is a method where plastic/metal in a semi molten state is extruded onto a surface, layer by layer. This is the most common 3D printing method, however the downside to FDM is the restriction on the precision of the layers created by the nozzle diameter and the vibration induced when moving the nozzle. FDM requires a self supporting structure that imposes a limit of less than 45-60° overhang while printing. If the limit is exceeded, the overhanging surface might be printed with an incorrect surface, bad surface quality and/or failed print.

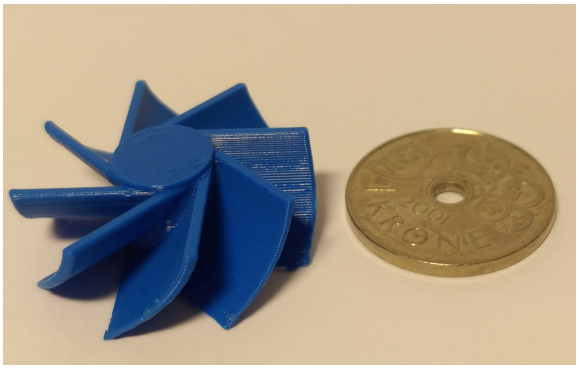


Figure 2.8: 3D printed runner. Photo and print by Lars Røed Ramstad.

The surface imperfections shown in figure 2.8 can be improved by using a thin layer of plastic fillers and sanding the runner to the intended shape.

Stereolithography is a more advanced manufacturing technique using a low powered laser beam in order to harden liquid photopolymers. The precision and accuracy is vastly superior for small 3D prints compared to FDM, which is evident by comparing figure 2.8 and 2.9. However, due to toxic fumes during printing it is less available than FDM.

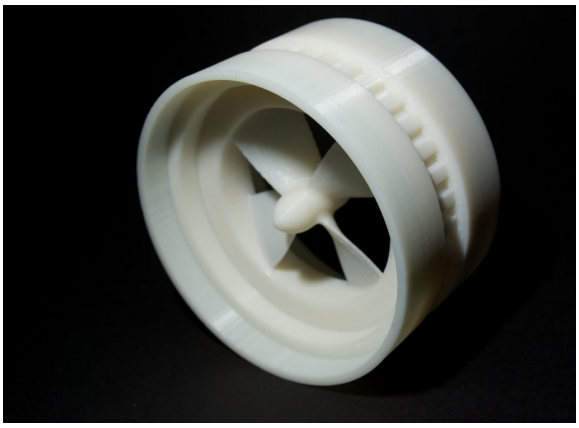


Figure 2.9: 3D printed propeller (Listek, 2019).

Chapter 3

Design

The aim is to design a runner that produces enough power to sustain a controlling unit (1-10 W) using some of the pressure in the water supply. To simplify the design the guide vanes and bulb are not included, and in difference to conventional runner designs the runner blades only curve in two dimensions (section 2.1). This is mainly to simplify the construction of the runner in the CAD program Autodesk Fusion 360, but also simplifies the comparison with the control volume analysis. For the initial calculations the following variables have to be set:

- Pipe diameter
- Head pressure
- Volumetric flow rate
- Rotational velocity of the runner

The turbine is intended to be connected to a 25.4 mm (1 inch) pipe with a head pressure of 1 bar. Due to the COVID-19 lockdown employed March 12, 2020, some measurements had to be done at home. The volumetric flow rate 20 $^{L}/_{min}$ is measured in a household sink yielding an inlet velocity u_{inlet} of 0.658 $^{m}/_{s}$. With these constraints, 15 $\%$ pressure loss equates to 5 W.

The rotational speed of the runner ω_{rot} is dependent of the volumetric flow rate, geometry inside the turbine, the friction in the turbine and the friction produced by the generator. The available bearings for this project proved to have a substantial rolling friction and a low rotational speed of 6.125 $^{rad}/_{s}$. The total friction of the system is dictated by the friction of the ball bearing, the load of the generator circuit and viscous friction. This reveals a large amount of uncertainties, but also the fact that the rotational speed can be regulated using a dummy load if the rotational speed is too high. The low rotational speed is chosen to increase the torque produced and ensure that the torque is large enough to overcome the friction. This is evident by the equation

$$P = T_z \cdot \omega_{rot} \tag{3.1}$$

where P is power and T_z is torque about the z-axis (along the pipe). By lowering the rotational speed the torque will increase if the power is constant. This is primarily because of the high friction in the suboptimal bearings that are available for this project (section 5.5).

The turbine casing for a final product is not designed, but a turbine casing using Plexiglas is designed (section 5.2-5.4). A similar design using a material such as stainless steal or aluminium is possible.

3.1 Runner Design

In this design an axial turbine is the most beneficial because it requires less space than a radial turbine. Another option is to use a Pelton turbine at the outlet to the urinal and extract all the energy while flushing. The drawback of this option is that the turbine is limited to extracting energy while flushing, the benefit is that the turbine utilizes all the pressure energy during this time interval. The axial turbine has a continues operation where it siphon some of the energy from the pressure while the flow still remains, this is beneficial because the turbine can be located further upstream to include more water usage in the area.

During the design phase it is emphasized that the runner should be easily manufactured using 3D printing. The most common 3D printers use Fused Deposition Modeling (FDM) or Sterolithography (section 2.6), where FDM is a good option for prototyping, while sterolithography is better during the production of the finished product due to a higher surface quality. A limitation with FDM is the requirement of a self supporting structure. This imposes a restriction of around 45° overhang while printing that has to be included as a design criterion. If the angle is less than 45°, this can result in a incorrect surface, bad surface quality and/or failed print.

3.1.1 Prototype A

During the outline of the thesis, the design of the runner was intended to be a hollow centered cylinder with five exterior angled blades. The angle of the blades α is determined by the diameter D_{hub}, thickness of the hollow hub t_{hub} and the length of the runner L_{runner}.

$$\alpha = tan^{-1}\left(\frac{D_{hub} - t_{hub}}{L_{runner}}\right) \tag{3.2}$$

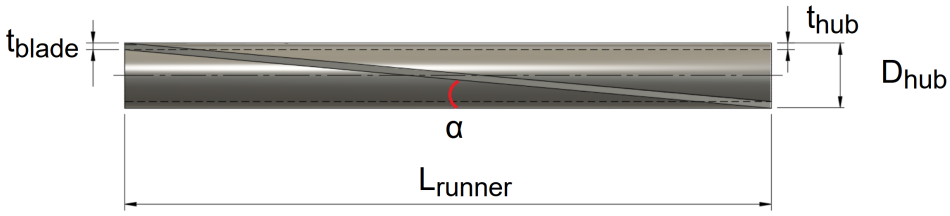


Figure 3.1: Side view of one blade and hub of prototype A.

Table 3.1 shows the parameters of prototype A.

Table 3.1: Prototype A parameters.

L_{runner} [mm]	D_{hub} [mm]	t_{hub} [mm]	t_{blade} [mm]	α [°]
100	10	1	1	5.143

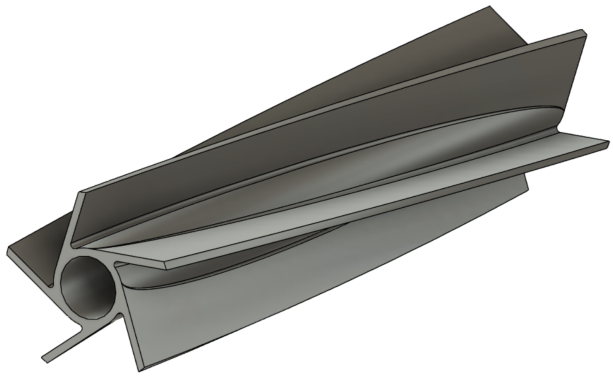


Figure 3.2: Isometric view of prototype A.

The CFD model estimates that prototype A produced too little torque to be feasible. The first observation is that the flow passes through the hollow hub instead of rotating the runner, this is logical because its the path of least resistance. The second observation is that the angle of the blades is to small to produce the required torque. This requires the length of the runner L_{runner} to be shorter and/or the diameter of the hub to be larger.

3.1.2 Prototype B

To find a better shape and parameters for the runner blades, a 2D steady state case is made in OpenFOAM by modifying the *Cavity* tutorial. The model simulates the flow at a constant radius between two blades. In order to test several profiles a Bash-script is constructed to make a test battery for different blade outlet angles α_2 (figure 2.5).

2D CFD Model

The 2D CFD model simulates the flow between two runner blades at ${}^{r}/_{R_{pipe}} = {}^{3}/_{4}$. The script modifies the *blockMeshDict*-file to create blade profiles with different blade outlet angles. This allows the program to test multiple 3-point B-spline curves that represents both the top and bottom of two blades with a distance S apart. The parameter S is dependent of the number of blades n_{blades}, the radius R_{pipe} and the blade thickness t_{blade}.

$$S = \frac{2\pi r}{n_{blades}} - t_{blade} = \frac{\pi \frac{3}{2} R_{pipe}}{n_{blades}} - t_{blade} \tag{3.3}$$

Using "trial and error", $n_{blades} = 8$ is chosen because it is the least amount of blades needed to cover the cross sectional area of the pipe. Table 3.2 summarizes the parameters to calculate the domain height S.

Table 3.2: Parameters to calculate the domain height S.

${}^{3}/_{4}R_{pipe}$ [mm]	n_{blades} [-]	t_{blade} [mm]	S [mm]
9.525	8	1	6.48

The thickness of the blade t_{blades} is set to 1 mm. This is equal of two passes for each layer with a FDM 3D printer, discussed in section 2.6. This implies that the sharp edges of the 3D print must be sanded before the runner can be tested in a laboratory experiment.

The simplification of calculating a uniform blade, leads to a geometry where point p_1, p_2 and p_{3x} are fixed while the y-component of p_3 determines the blade outlet angle α_2.

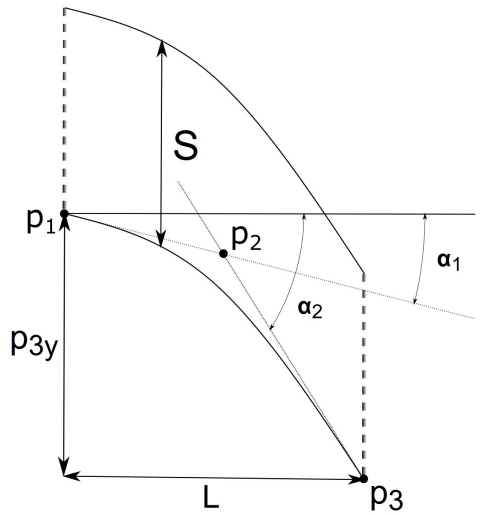


Figure 3.3: 2D domain, point coordinates are given in table 3.3.

This leads to a hub diameter

$$D_{hub} = |p_{3y}| + t_{blade} \tag{3.4}$$

Due to the blockage of the hub (runner blades neglected) the relative velocity w_1 is calculated.

$$w_1 = \sqrt{u_{inlet}^2 + (\omega_{rot} r)^2} \frac{A_{pipe}}{A_{hub}} = \sqrt{u_{inlet}^2 + (\omega_{rot} r)^2} \frac{1}{1 - \frac{D_{hub}^2}{D_{pipe}^2}} \tag{3.5}$$

The blade's inlet angle α_1 is constant because it is set equal to the flow entry angle α_{entry} at $r = {}^3\!/_4 R_{pipe}$ for all r.

$$\alpha_1 = \alpha_{entry} = tan^{-1}\left(\frac{{}^3\!/_4 R_{pipe} \omega_{rot}}{u_{inlet}}\right) \tag{3.6}$$

Table 3.3: Equations for point coordinates.

Point	x-coordinate	y-coordinate
p_1	0	0
p_2	0.5 L_{runner} = 5 mm	$-{}^1\!/_2 \cdot L_{runner} \cdot tan(\alpha_1)$
p_3	L_{runner} = 10 mm	$-{}^1\!/_2 \cdot L_{runner} \cdot [tan(\alpha_1) + tan(\alpha_2)]$

To create this domain using *blockMesh* the outlet is shifted downwards and the top and bottom wall is curved with a b-spline curve function. This is conducted using a Bash-script that modifies *blockmeshDict*. The simulation uses the steady state solver SimpleFOAM with no turbulence model ("Laminar") implemented. The mesh has a $y^+ = 1.25$ and wall functions are therefore not used (Guerrero, 2014, page 903). The function object "forceCoeffs" is included in the *controlDict* to calculate the lift coefficient C_L to the blade profile.

$$C_L = \frac{dF_\theta}{\frac{1}{2}\rho A u_{inlet}^2} \tag{3.7}$$

where dF_θ is the vertical force acting on the control volume and A is set equal to the length of the blade. After finding a steady state solution, the script calculates the net vertical force on the blade and the angle of the velocity at the exit of the runner blade. The script labels each simulation with a CASEID that contains the coordinates needed to represent the curve and sorts the results according to the equation

$$CASEID = max\left(\frac{dF_\theta}{\alpha_2 - \alpha_{exit}}\right) \tag{3.8}$$

where α_2 is the blade outlet angle and α_{exit} is the flow exit angle at the trailing edge of the runner. The equation promotes the profile that yields the highest force and a late separation of the flow. Figure 3.4 shows the highest performing blade curve obtained after sorting the 2D simulations.

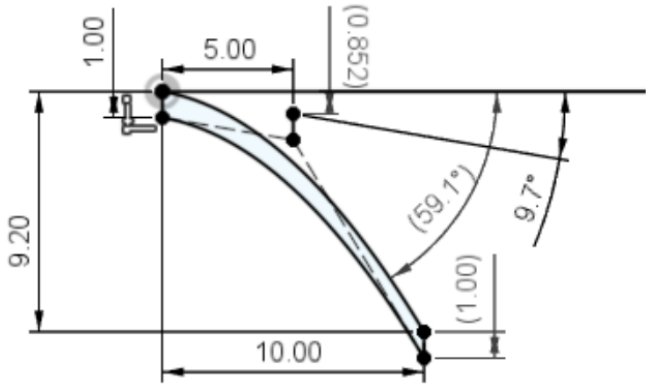


Figure 3.4: 2D shape transferred to Autodesk Fusion 360 (unit: mm).

3D Model of the Runner

The blade profile shown in figure 3.4 is exported using the extract surface filter in Paraview as a STL-file and imported into Autodesk Fusion 360. The sharp edges on the runner are rounded using the fillet function (figure 3.5) and the sharp chamfers at the runner hub are rounded (0.1 mm) to prevent small and skewed cells while meshing. The length of the runner hub is also extended by 0.5 mm upstream and downstream of the runner blades. This is a result of Autodesk Fusion 360 having problems with colliding fillet/chamfer from the hub edge and the junction between the blades. The leading and the trailing edge of the 2D blade profile is rounded to give the blades a more aerodynamic form which promote less disturbance downstream of the runner.

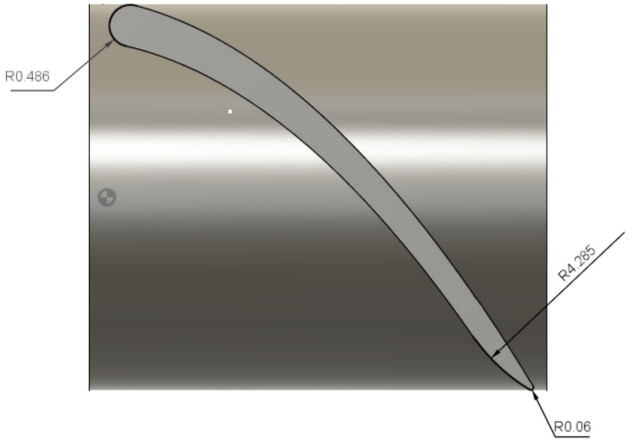


Figure 3.5: Side view of one blade and hub with fillet dimensions of blade edges (unit: mm).

Using the revolve function in Autodesk Fusion 360 the number of blades of the runner is set, yielding prototype B.

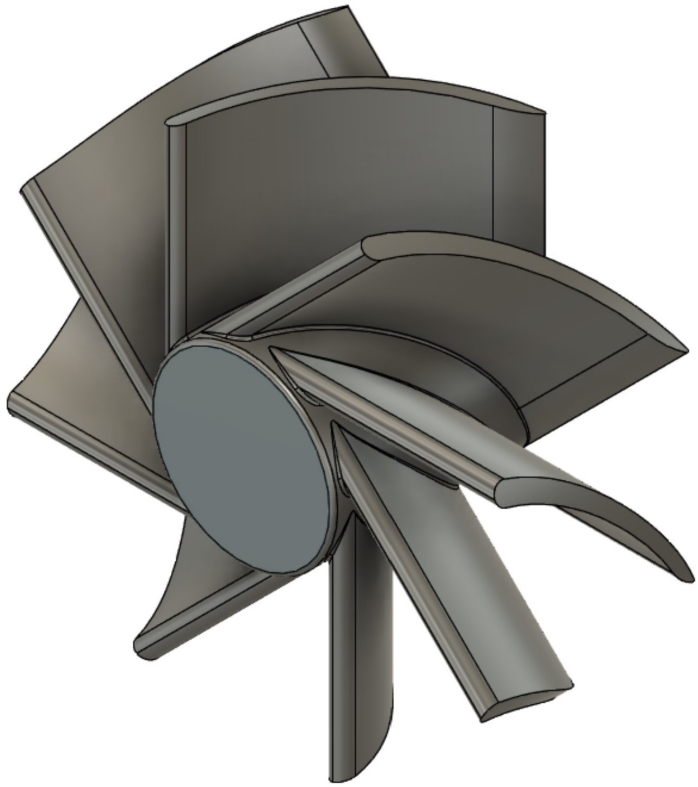


Figure 3.6: Prototype B constructed in Autodesk Fusion 360.

Chapter 4

CFD Model

Computational Fluid Dynamics (CFD) is used in a large variety of industrial and scientific applications like optimizing the shape of aerodynamic features, chemical reactions in a mixture and multiphase flow problems.

OpenFOAM is a free open source software. The source code is written in C++, but has a higher level "interface" for users. It is native to Linux and available for Windows and Mac, however Linux is the recommended option. OpenFOAM does not have a user-friendly GUI (graphical user interface), but relies on text-files in a specific case structure to organize the simulation. A recommended option is to use a text editing program such as Atom to structure and edit the case files.

One of the largest benefits with OpenFOAM is the ability to edit or use custom made code. This is in contrast to most commercial code where the user gives input and settings and receives the output without the ability to check the source code. Another benefit of OpenFOAM is its library of tutorials. This is a selection of base cases within several fields that can be copied and altered into the flow problem that is investigated.

4.1 OpenFOAM, *Propeller* Tutorial

The OpenFOAM *Propeller* tutorial is the closest tutorial to this thesis' flow problem. The major difference is the enclosed flow by the pipe, creating an internal flow, compared to the *Propeller* tutorial which is an external flow. The *Propeller* tutorial is a transient simulation using RANS turbulence modeling (k-ϵ-model). The propeller is rotated using a cell-zone with Arbitrary Mesh Interface (AMI) patches that is rotated at a constant rotational speed ω_{rot} (section 2.4.7). This is established by creating a cylindrical mesh for the rotating region and an equal void in the external mesh where the rotating region can be integrated. By merging the two meshes this creates a patch between the rotating region and the external mesh where the rotating region can slide. The rotational axis, speed and direction is set using the *dynamicMeshDict*-file.

4.2 OpenFOAM, Pico Turbine

Using the setup in the *Propeller* tutorial the first change is to create a mesh for the interior flow. The main question is where the AMI patches should be located. To mimic the design the entire domain should be surrounded by the rotating region except for the inlets and the outlets. However this creates difficulties with coinciding patches. The rotating region is therefore chosen to contain the runner and a section upstream and downstream of the runner. The pipe wall is also included because the pipe and the runner patch coincide. If the pipe wall is stationary and the runner is rotated, an undefined hole in the pipe wall patch will appear and a double defined edge where the runner coincides with the pipe wall patch will exist. This causes the simulation to crash, due to the undefined and double defined patches. The possibility of only rotating the runner demands a gap between the runner and the pipe wall. This creates a problem when this gap has to be filled with extremely small cells. This dictates an unnecessary small time step with regard to the Courant number for this simulation.

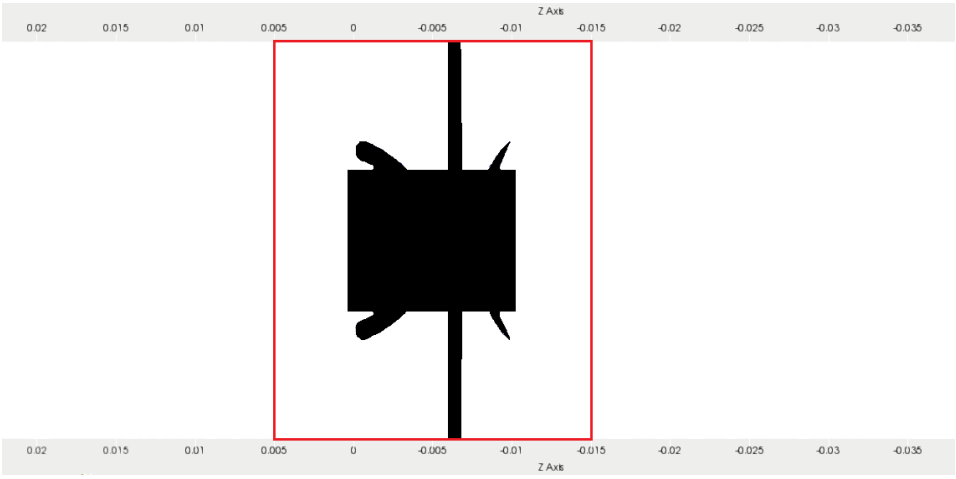


Figure 4.1: 2D representation of domain with rotating region marked with red lines.

Figure 4.1 also shows the short domain compared to the estimation made in section 5.1 (including the shortening of the entry length). The short length of the domain is a consequence of the high amount of cells it takes to create a semi smooth runner surface, this will be further addressed in section 4.3.3 and 4.4.

4.3 Meshing Process

In spirit of only using open source software the OpenFOAM functions *blockMesh* and *snappyHexMesh* and the CAD program SALOME is intended to create the mesh. This is a choice made because the software is available for free after graduating. During the testing of the prototype A runner, OpenFOAM's mesh functions create a mesh within OpenFOAM's mesh quality criterion and the simulation yields a converged solution. When

changing to the prototype B runner the mesh quality remained within the quality criterion, but crashs due to a ”divide by zero”-error. This proves to be because *snappyHexMesh* is not able to capture the pipe geometry and can not create a perfect circle (section 4.3.3) for the AMI to slide along. At this point the meshing process is moved to SALOME that creates unstructured meshes. The meshing follows the meshing guide by Ali Ikhsanul (Ikhsanul, 2017). The problem with this mesh is high non-orthogonality that causes divergence after the runner started to rotate. An effort is made to tweak and tune the mesh with the information from the tutorial, but the meshes have a maximum non-orthogonality of 78°. Without the possibility of improving the SALOME mesh the focus is directed back to the *blockMesh* and creating a cylindrical domain instead of using *snappyHexMesh* to cut the *blockMesh* into a cylinder. This is possible using the curve feature (similar to the 2D model) in *blockMesh*, however the cells close to the center in a cylindrical mesh have a high aspect ratio that is problematic. To circumvent this problem an O-grid mesh is preferred (figure 4.9). With an O-grid mesh provided by wolfDynamics (Guerrero, 2019a, page 60-66) the mesh has acceptable mesh quality that produces a converging solution.

4.3.1 SnappyHexMesh

The mesh is created using the *snappyHexMesh* (SHM) function. It manipulates a premade hexahedral mesh (*blockMesh*) in order to import geometries and create cell refinement. SHM has four phases where it imports the geometry and creates refinement regions, castellation (removal of cells with 50 % of its cellbody inside the geometry), snapping (moving vertices onto surfaces) and layering. The parameters used by SHM are stored in the system folder under *snappyHexMeshDict*. The geometry file for the runner (*.stl/*.obj file) is stored in the *triSurface* folder, which is located in the *constant* folder. The *snappyHexMesh* flowchart found in the OpenFOAM v1912 User Guide (OpenFoam.com, 2019) is shown in figure 4.2.

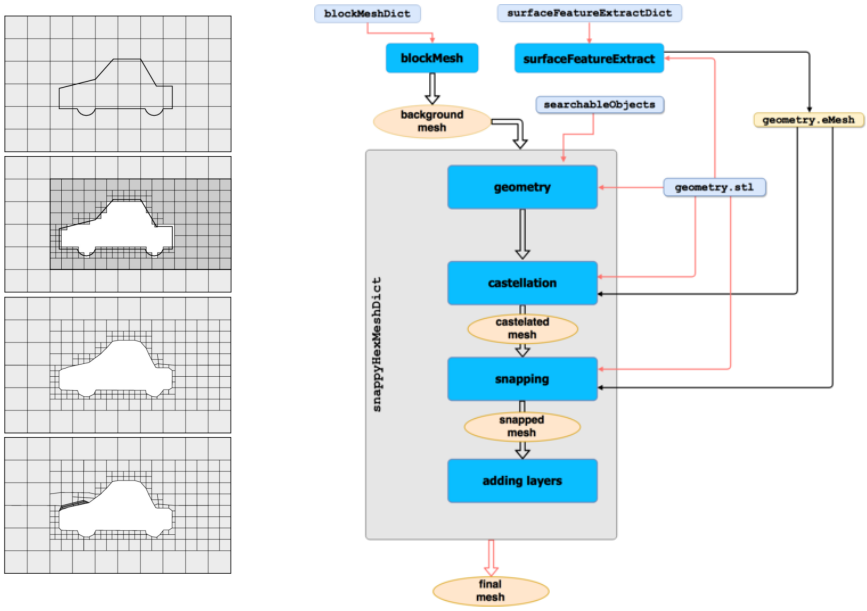


Figure 4.2: *SnappyHexMesh* flowchart (OpenFoam.com, 2019).

4.3.2 Import Geometry, Defining Cell Regions and Castellation

After importing the geometry files, SHM creates searchable regions. This feature is used to create the refinement regions used by SHM and to define the rotating zone used by the *dynamicMesh* function. A cylindrical cell zone called *innerCylinder* is created around the runner which is initialized by two points that creates the height of the cylinder and a radius. The radius is set larger than the pipe radius to ensure that all interior cells and the pipe wall patch is included in the cell selection.

```
innerCylinder
{
type            searchableCylinder;
point1          (0 0 0.005);
point2          (0 0 -0.015);
radius          0.0254;
}
```

Figure 4.3: Implementation of a searchable region (cylinder) used to create the rotating and refinement region in *snappyHexMeshDict*.

This cell zone is used to create the AMI which allows the mesh to be rotated. The zone is also used as a refinement region in order to have smaller cells close to the runner surface. During testing a numerical error was observed at the AMI patches, as shown in figure 4.4 and 4.5, using an intermediate prototype runner based on prototype A (section 3.1.1) with length 10 mm, blade angle 45° and a hollow hub. The figures show the velocity magnitude and the pressure along the center of the pipe.

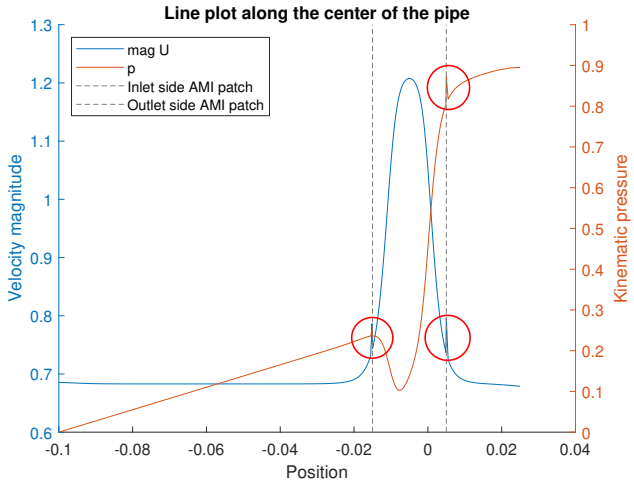
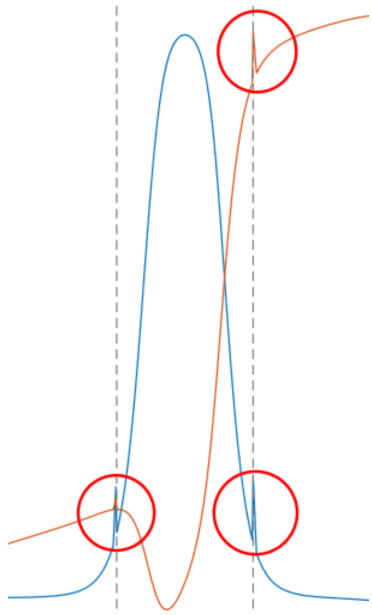


Figure 4.4: Unequal cell refinement level.



.

Figure 4.5: Enlargement of oscillation in figure 4.4 to show the oscillations marked with red circles.

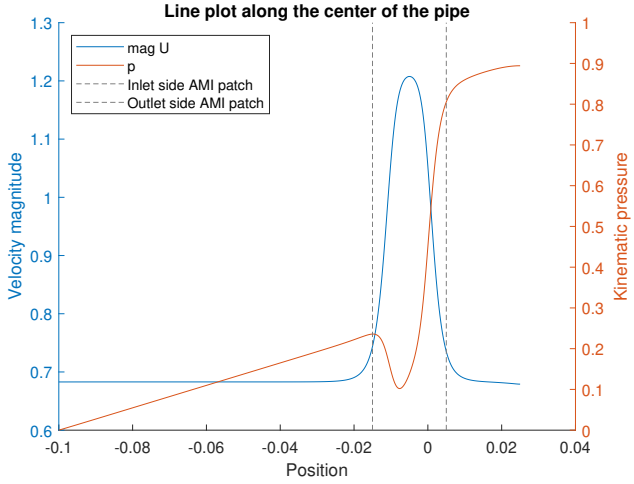


Figure 4.6: Equal cell refinement level.

This numerical error is introduced by the abrupt change in cell size at the AMI patches and is remedied by creating a refinement region outside the *innerCylinder* zone. This zone is called *outerCylinder*. The refinement regions are seen in the plane along the pipe center shown in figure 4.7. Table 4.1 states the refinement levels used for each area where the darker/denser areas in figure 4.7 have a higher refinement level. The region *innerCylinder* is located between the AMI patches (denoted by red lines in figure 4.7) and the refinement region *outerCylinder* is the mesh area located on the outside of the dynamic mesh region. The coarser center section is a result of how the O-grid mesh is constructed and efforts are made to balance the inner and outer section to create a quality mesh.

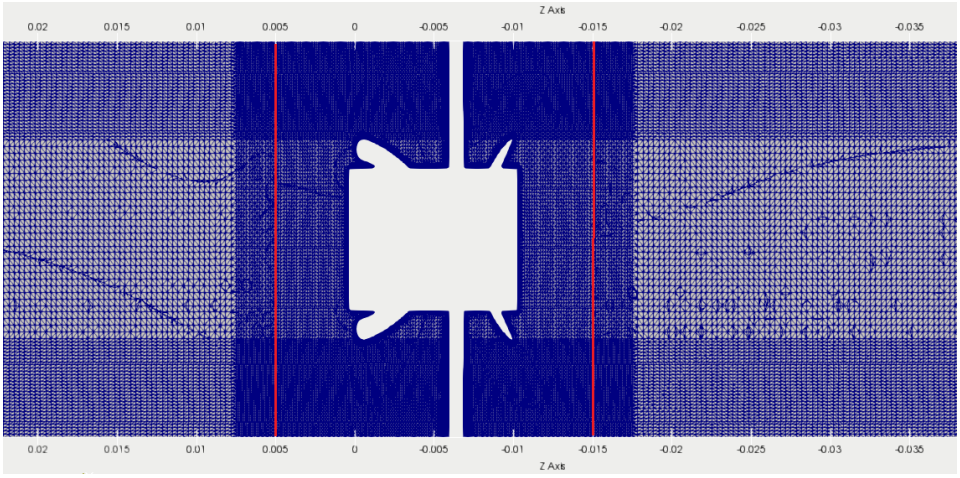


Figure 4.7: Refinement of regions and surfaces in the region close to the runner.

Table 4.1 summarizes the refinement levels for each section where a higher refinement level dictates smaller cells in the area.

Table 4.1: Cell refinement levels of edges, surfaces and regions.

Edge, surface, region	Refinement level
Runner edge	4
Runner surface	3
Pipe surface	1
Inner cylinder/rotating cell zone	1
Outer refinement zone	1

4.3.3 Snapping

The snapping process is where the hexhedral cells are deformed by moving the cell vertex points onto the surfaces. In order to snap the geometry successfully, the cells close to the surface must have an aspect ratio close to unity (Greenshields, 2019, section 5.4.2). Initially a rectangular formed hexhedral mesh is constructed. This creates a good quality mesh in regard to non-orthogonality (40° <), maximum aspect ratio (1.5 <) and maximum cell skewness (1.5 <), but requires to snap the pipe geometry using an additional stl-file. While running the simulation this causes a zero division error due to misalignment after rotating a non-circular AMI patch as shown in figure 4.8.

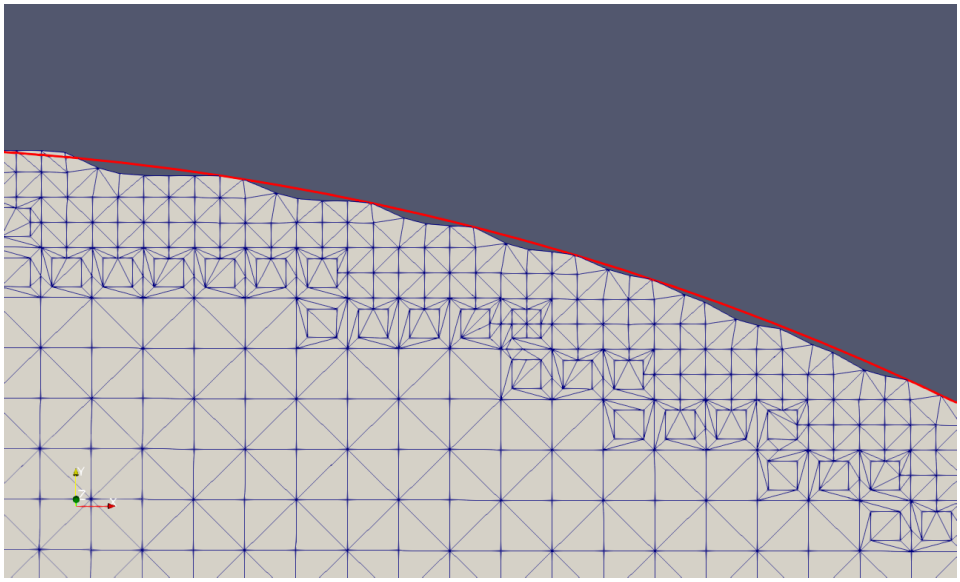


Figure 4.8: Imperfect pipe topology.

The red line denotes the surface created by the imported pipe stl-file, which shows the deviation caused by a too coarse initial mesh. The cause of the error is proven by running three simulations with different rotational speeds (ω_{rot}). Table 4.2 shows that the crash occurs after rotating the mesh 1.178°, which implies that it is caused by the rotation of the mesh.

Table 4.2: Crash time during rotating mesh test.

ω_{rot} [$^{rad}/_{s}$]	Simulated time before instability [s]
0	∞
6.125	0.0032
12.50	0.0016

To circumvent this problem a *blockMeshDict* created by Wolf Dynamics (Guerrero, 2019a, section Supplement: blockMesh) is used to create the O-grid mesh. Figure 4.9 shows the O-grid structure and the structural blocks created in *blockMeshDict*.

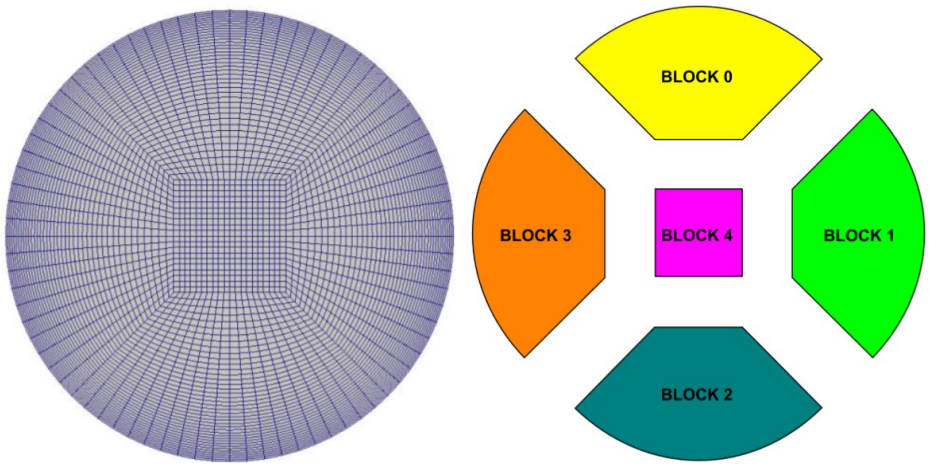


Figure 4.9: Wolf Dynamics' O-grid mesh (Guerrero, 2019a).

The advantage of using the O-grid mesh is the structured cylindrical mesh that has a perfectly circular outer edge. The disadvantage is the different cell shapes that are in conflict with the aspect ratio snapping requirement that is evident by inspecting the surface topology of the runner. The runner surface after snapping is shown in figure 4.10.

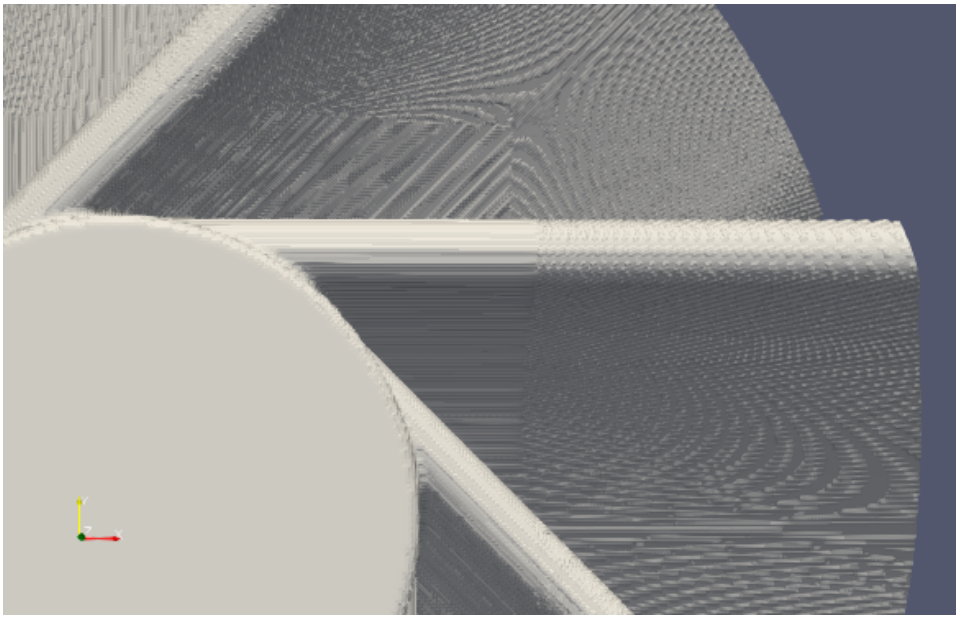


Figure 4.10: Runner surface after snapping, O-grid mesh can be seen along the surface.

4.4 Domain

The domain is created with three times the pipe diameter upstream and five times the pipe diameter downstream of the runner's leading edge. The short domain is due to the large requirement of cells in order to capture the runner surface. The main interest is to compute the torque on the runner which is geometry dependent and thus the domain length has to be sacrificed to gain better surface quality because of RAM-restriction. This causes the wake to be in contact with the outlet boundary which is not optimal.

4.5 Mesh Quality

The mesh quality is checked using the *checkMesh* function, and the criteria are located in the *snappyHexMeshDict*-file under quality control. Table 4.3 shows the output from the *checkMesh* function of important properties for the mesh quality.

Table 4.3: Mesh quality.

	Maximum	Average
Number of cells	6547341	-
Non-orthogonality	68.8196°	12.3743°
Skewness	2.5633	-
Cell aspect ratio	5.31822	-

Because of the difficulties to create an acceptable mesh, a grid refinement study is not conducted. This is mainly due to the high amount of cells needed to capture the geometry of the runner during snapping. The high amount of cells required results in a problem during post processing where the RAM is the limiting factor. There are possibilities to circumvent this problem, however due to the time limitation the project is continued without a grid refinement study.

4.6 Constants

In this thesis the density ρ and dynamic viscosity μ are assumed constant. Both quantities are dependent on the temperature T, however these changes are negligible in this flow scenario. The temperature equation is not solved in this simulation, however the water is presumed to have a constant temperature $5\,^oC$. This yields a kinematic viscosity of $1.547e^{-6}\,{}^{m^2}/_{s}$ and a density of $1000\,{}^{kg}/_{m^3}$. The temperature and density are irrelevant, except their implication on the viscosity and calculation of static pressure (equation 2.78). Table 4.4 summarizes the transport properties.

Table 4.4: Transport properties (White, 2016, page 738).

$T\,[^oC]$	$\rho\,[{}^{kg}/_{m^3}]$	$\nu\,[{}^{m^2}/_{s}]$
5	1000	1.547e-6

4.7 Initial Conditions

4.7.1 Velocity and Pressure

The internal velocity field is set equal to the inlet velocity $[0,\ 0,\ -0.658]\,{}^{m}/_{s}$, where the z-axis is orientated in the counter flow direction. The internal field of the kinematic pressure is set to $0\,{}^{m^2}/_{s^2}$.

4.7.2 Turbulence Modeling

The Reynolds number calculated in table 4.5 estimates a turbulent flow (White, 2016, page 310).

Table 4.5: Reynolds number.

D_{pipe} [m]	$\lvert u_{inlet}\rvert\,[{}^{m}/_{s}]$	$\nu\,[{}^{m^2}/_{s}]$	Re_D [-]
0.0254	0.658	1.547e-6	10804

This dictates the use of a turbulence model (section 2.4.6) as the turbulence scales are not resolved (temporal and spatial) in the simulation. Due to familiarity with 2-equation models the k-ϵ, k-ω and k-ω-SST models where of initial interest. When using the k-ϵ model

it is beneficial to use a wall function because the model does not reproduce the boundary layer accurately. Due to the small cells needed to capture the geometry this led to a $y^+ < 6$ which is unsuited for wall functions in combination with the k-ϵ model (Guerrero, 2021, page 162). The k-ϵ model proved numerically unstable and due to the previous arguments the model was not further used.

The k-ω causes instability for all available initial condition. The k-ω-SST is the only turbulence model that produces a converged solution. In an effort to use the k-ω the fields from a k-ω-SST is transferred to the simulation with k-ω, but the simulation remains unstable. Efforts are also made to investigate the use of different timesteps, but this proved unfruitful. To elaborate on this instability, the simulation is conducted without a turbulence model. By omitting turbulence modeling OpenFOAM is conducting a direct numerical simulation on a too coarse grid with a large time step. This means that information is lost. The simulation without the additional turbulent viscosity proves also stable which disproves the hypothesis that the increased viscosity causes the stability. The reason why k-ϵ and k-ω creates instability and why k-ω-SST (section 2.4.6) and "laminar" model creates a converged solution, is a question left as further work.

Initial Condition Turbulence Properties

The ANSYS userguide gives a relation between the initial conditions for the turbulence kinetic energy, turbulence intensity, Reynolds number and the turbulence dissipation frequency (ANSYS, 2020, section 7.3.2).

$$k = \frac{3}{2}(|u_{inlet}| \cdot I)^2 \tag{4.1}$$

The turbulence intensity I is defined by $I = \frac{u'}{\bar{u}}$ where u' is the fluctuation in the velocity and $\bar{u}$ is the mean velocity.

$$I = 0.16 \cdot (Re_D)^{-\frac{1}{8}} \tag{4.2}$$

The turbulence dissipation frequency is calculated using

$$\omega = \frac{\rho k}{\mu}\left(\frac{\mu_t}{\mu}\right)^{-1} \tag{4.3}$$

At low turbulence the turbulence viscosity ratio is approximately unity. This yields the following initial conditions for the turbulence quantities shown in table 4.6.

Table 4.6: Initial conditions for turbulent properties.

$\lvert u_{inlet}\rvert$ [$^m/_s$]	Re_D [-]	I [-]	k [$^{m^2}/_{s^2}$]	$^{\nu_t}/_{\nu}$ [-]	ω [s^{-1}]
0.658	10804	0.0501	1.6308e-3	1	1054

4.8 Boundary Conditions

The inlet velocity is set to a constant $[0,\ 0,\ -0.658]\ ^m/_s$. Ideally this should have been a velocity profile, but due to the time restriction this is not implemented. The outlet velocity condition is the *inletOutlet* velocity condition. This handles backflow by giving it a constant velocity while operating as a zero gradient boundary condition elsewhere. This boundary condition is selected due to the short exit length in case of eddies producing counter flow at the outlet. The runner has the *movingWall* boundary condition which sets the velocity to $[0,\ 0,\ 0]\ ^m/_s$ relative to the runner while the mesh section is moving. The pressure at outlet is set to zero $^{m^2}/_{s^2}$ and at inlet is set to zero gradient. To enhance numerical robustness all turbulent properties (k, ω, ν_t) are set to $1e^{-10}$ instead of zero (Versteeg and Malalasekera, 2007, page 91). This is to prevent divide by zero in any calculation. Due to the rotating mesh the wall functions are not used, even though the boundary layer is not resolved. This also made it possible to obtain an $y^+ \leq 6$ which arguably allows a constant boundary condition on the walls (Guerrero, 2014, page 861). This is advantageous because wall functions are derived with the following assumptions (Versteeg and Malalasekera, 2007, page 279); the velocity is parallel to the wall and varies only in the direction normal to the wall, no pressure gradients in the flow direction and high Reynolds number. Neither of these assumptions are valid in this CFD model, thus a constant boundary may be more accurate even though the viscous boundary layer is not fully resolved ($y^+ \leq 1$).

Table 4.7: Initial conditions for the simulation.

Field	inlet	outlet	pipe	runner
U	[0 0 -0.658]	inletOutlet [0 0 0]	noSlip	movingWall [0 0 0]
p	zeroGradient	fixedValue uniform 0	zeroGradient	zeroGradient
k	zeroGradient	inletOutlet uniform 0	fixedValue uniform 1e-10	fixedValue uniform 1e-10
ω	zeroGradient	inletOutlet uniform 0	fixedValue uniform 1e-10	fixedValue uniform 1e-10
ν_t	zeroGradient	inletOutlet uniform 0	fixedValue uniform 1e-10	fixedValue uniform 1e-10

The CFD model does not require an entry length because it can set a velocity profile. It is therefore custom to make a separate simulation in order to find a steady state inlet velocity profile or use a known function for velocity profile. This shortens the computational domain which leads to a faster computational time.

The exit length L_{exit} is of equal importance. Due to the design choice of not implementing guide vanes to create a rotational free outlet, a swirl will be generated downstream of the runner. Ideally the CFD model should contain this swirl. However this will demand a large domain as the energy in the swirl is only dissipated by viscous friction. The most

logical approach would therefore be to use the laboratory experiment to find the length of the swirl and use this length with a margin for the computational domain. Without this option the exit length is assumed to be longer than the entry length because the viscous friction has to eliminate the swirl in the same matter as creating the boundary layer at the inlet.

4.9 OpenFOAM Settings

The setup used in this thesis is based on "Tips and tricks"-pdf by Guerrero (2019b). The lecture notes are a summary of experiences during his research and suggest general OpenFOAM guidelines.

Due to stability, mesh quality dictates a numerical setup where different schemes, limiters, blending factors and under-relaxation are used to enhance stability and accuracy. In this regard, the maximum non-orthogonality is important during the calculation of the diffusion term. The non-orthogonal vector is split into an orthogonal and a non-orthogonal component (section 2.4.5). The maximum non-orthogonal cell in the mesh is 68°. The selected setup for *fvScheme* and *fvSolution* found in the "Tips and tricks"-pdf is based on a maximum orthogonality of 70-80° (Guerrero, 2019b, page 36). The choice of using more correctors and under-relaxation is based on the possibility of instability during the transition from static to dynamic mesh at the startup. The *PIMPLE* settings are set accordingly

Table 4.8: *PIMPLE* settings.

Setting	Option
momentumPredictor	no
correctPhi	yes
nOuterCorrectors	2
nCorrectors	3
nNonOrthogonalCorrectors	3

To ensure stability the following under-relaxation factors are used.

Table 4.9: Under-relaxation settings.

Field variable	Under-relaxation factor
p	0.3
pFinal	1
U	0.7
k	0.7
ω	0.7

The initial divergence scheme for all quantities is the linear upwind scheme. Due to instability while calculating the turbulent quantities, the linear upwind scheme is changed

to the first order accurate upwind scheme for k and ω. The explicit Euler method is used to march forward in time. This introduces some numerical diffusion, but this effect can arguably be less detrimental because all the turbulence is modelled as turbulent viscosity. The pressure is solved using the geometric algebraic multigrid (GAMG) method and for the velocity the symmetric Gauss Seidel method is used. The time step is initially set to 1 ms, but after a few iterations the automated time step function is used with a limit of $Co_{max} \leq 0.5$. After four full revolutions of the runner the divergence scheme for the turbulent quantities is changed to the *MUSCL* scheme, which provides a converged solution with a second order accuracy in space. The simulation is afterwards ran until the torque calculated by PimpleFOAM is converged.

4.10 High Performance Computing

OpenFOAM is capable of parallel computing. This is advantageous because the domain can be separated into regions and calculated simultaneously. This shortens the wall clock time spent to run the simulation. The parallel processing is controlled by the function *decomposePar*. This function divides the domain into subdomains that each processor core calculates and then handles the cross communication between the processor boundaries. The *decomposePar* function is controlled by the *decomposeParDict*-file where number of processor cores and method is set. There are several options for the decomposition and how the regions are divided. The simplest option is to divide the domain using a geometric decomposition, however because most of the cells in this simulation are located at the runner this will create a computational bottleneck. Another option is the Scotch method, where the decomposition is based on minimizing the amount of CPU boundaries (openfoam.com, 2020) which is used in this thesis.

To determine a suitable number of processor cores there are some considerations that needs to be taken. The most important factor is the amount of available cores. If a simulation requires a large amount of cores the waiting time in queue to run the simulation will increase. Another limiting factor is weak scaling, where using more processors will yield diminishing returns. The weak scaling is described by Amdahl's law (Amdahl, 1967) which estimates a theoretical speed-up of the processing that can be computed using parallel processed.

$$S_T = \frac{1}{(1 - f_p) + \frac{f_p}{n_c}} \tag{4.4}$$

where S_T is the theoretical speed-up, f_p is the fraction of the computation that can be parallel processed and n_c number of processor cores. The equation states that the serial part of the computation represented by $\frac{1}{(1-f_p)}$, is the limiting factor when the number of processors become many enough. It is possible to perform a weak scaling study to find the optimal amount of cores, however the IDUN high performance computing (HPC) cluster used in this thesis does not have a uniform selection of processors on each node. Meaning that each node have different computational efficiency.

In regard to primarily queue and wall clock time, 40 processor cores are used to run this

simulation. Compared to running locally on 10 processors this gives a significantly lower wall clock time to run the simulation.

Chapter 5

Design of Laboratory Experiment

The design of the laboratory experiment is based on the turbine being connected to a 1 inch water pipe and having access to a drain. The runner is intended to be printed at NTNU Make where they have both filament (FDM) and resin (stereolithography) 3D printers. The turbine housing is intended to be built by chemically welding Plexiglas to have a transparent pipe wall. The laboratory experiment is designed with the following requirements:

- robustness
- simple to build and change parts
- mounted to a water supply pipe using a valve
- the rotor must be able to turn freely (least amount of friction)
- the energy must be transformed to electrical power
- the water leakages must be contained

The sensory equipment needed to compare with the CFD results are an inlet and an outlet pressure gauge, a tachometer to measure the rotational speed of the runner and a flow meter to measure the flow velocity (this can be measured at the outlet).

5.1 Entry Length

To ensure a fully developed flow upstream of the runner in the laboratory experiment, the entry length L_{entry} is estimated using equation (2.9). For this flow problem this yields an entry length of 16.31 pipe diameters or 0.414 m. Table 5.1 summarizes the calculation of the entry length.

Table 5.1: Entry length.

D_{pipe} [m]	$\lvert u_{inlet} \rvert$ [m/s]	ν [m^2/s]	Re_D [-]	L_{entry} [m]
0.0254	0.658	1.547e-6	10804	0.414

5.1.1 Generator

In all designs a flooded generator is planned, this means that the gap in the generator is filled with water and that the coils and the magnets are cast in epoxy to prevent damage. This is based on the same principle as the flooded generators in tidal turbines (Wani et al., 2020, chapter 4).

5.2 Laboratory Concept 1: 90° Bend or T-Pipe

Laboratory concept 1 uses a shaft and a 90° bend or T-pipe to transfer the mechanical energy from the runner to the generator. In this setup the runner is mounted to a shaft that passes through the pipe wall using a bearing. The simplistic design has fewer parts than the other concepts (section 5.3 and 5.4) and a smaller bearing is more likely to be designed for this application with regard to friction/lubrication. This allows the turbine to transfer the mechanical energy directly to the generator. Because the runner is not in contact with the pipe wall, there will be some leakages between the runner and the pipe wall. To prevent flow induced vibration, high precision to align the bearing, shaft and runner along the center of the pipe is needed. This ultimately makes the concept less viable.

This concept has some practical issues with leakage through the bearing. To prevent further leakage an enclosing box has to seal the leakage as shown in figure 5.1 (dotted lines).

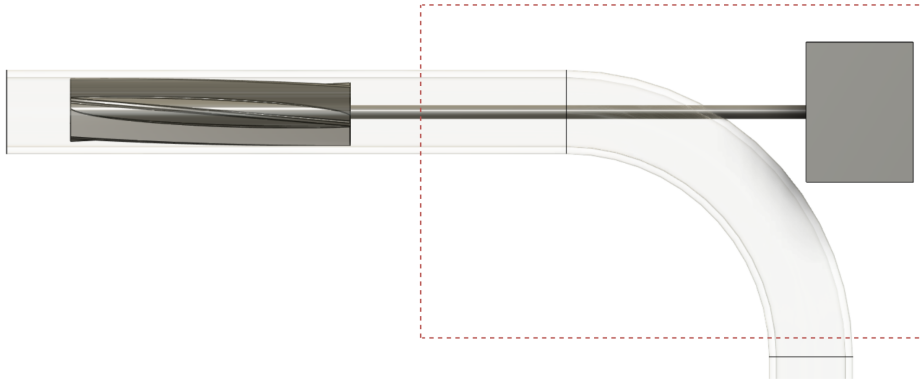


Figure 5.1: Top view of concept 1 with suggested box (red dotted line).

5.3 Laboratory Concept 2: Rim Driven Transmission

Laboratory concept 2 uses a rim driven transmission where a ring gear and a pinion are used to transfer energy from the runner to the generator. In this setup the runner is glued to the inside of a pipe, that in turn is mounted to two bearings on the outside. The ring gear is glued to the outside of the pipe to allow the runner to drive the pinion and generator. The gear system gives the option to have different rotational speed of the runner and the generator. This option is also less prone to vibrations, however using a gear and pinion will create friction which is problematic for this low energy device. In this concept an emphasize on a precise and efficient gear system is important. Compared to concept 1 this option uses an additional bearing and bearings that are much larger, this dictates additional friction/loses. The bearings in this option is also acting as seals which dictates that the turbine have to be enclosed by a box to prevent leakage in a similar matter as in concept 1.

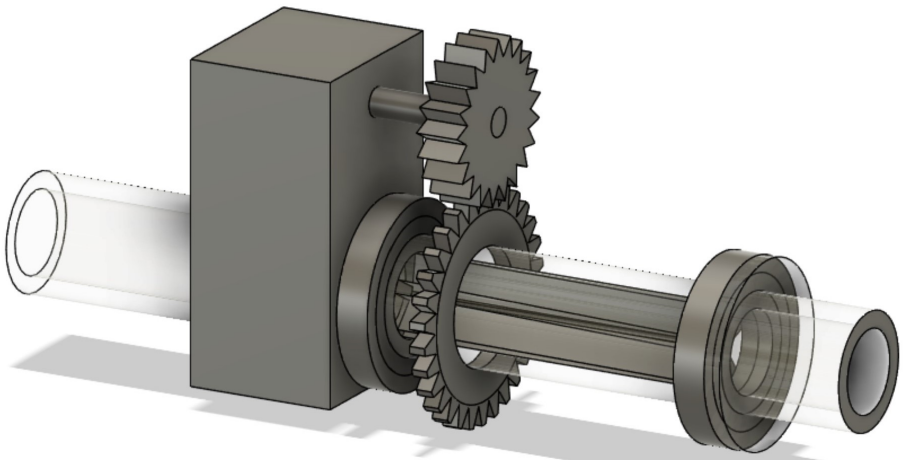


Figure 5.2: Isometric view of concept 2.

5.4 Laboratory Concept 3: Internal Generator

Laboratory concept 3 uses an internal generator design which is a further development of concept 2. In this option the pipe that the runner is glued to is used as a shaft that rotates the rotor of the generator directly. The stator of the generator is glued to the external pipe that prevents further leakage through the bearings.

Figure 5.3: Side view of concept 3 with generator on the right side.

5.5 Final design

Because of the campus lockdown caused by the COVID-19 pandemic, the turbine casing and generator is not built. The most promising concept is the internal generator concept. This is because it allows the generator to be rotated with a lower risk of vibration (no shaft or centering) in contrast to concept 1 and with less mechanical losses (no gear) than concept 2. The drawback of this concept is the requirement of a tailor made generator, while concept 1 and 2 can use a stock item generator that matches the turbine specifications.

The bearings intended for this laboratory experiment is the SKF 6006-2RS (Table 5.2). The main concern with the ball bearings are the rolling friction. The bearings are designed for much higher speed (maximum 8000 RPM) and load, this require a more viscous lubricant than needed for this thesis. The lubricant is therefore intended to be removed if the bearings have too high friction. The external pipe that prevents leakage is designed to have an inner diameter of 55 mm to fit the bearings, but this can be adjusted with 3D printed parts if the generator requires more space. To lock the two ball bearings between

Table 5.2: Deep grove ball bearing (SKF, 2020).

Bearing number	D_i [mm]	D_o [mm]	axial length [mm]	static load: C_0 [kN]
6006-2RS	30	55	13	8.3

the internal and the external pipe, 3D printed rings are glued onto the pipes. A ring is glued to the interior wall of the external pipe, acting as a rest for the bearing to be pushed into and to separate the main compartment from the generator. Two rings are glued onto the

exterior wall of the internal pipe and are used to push the ball bearing against rest when tightening the flanges.

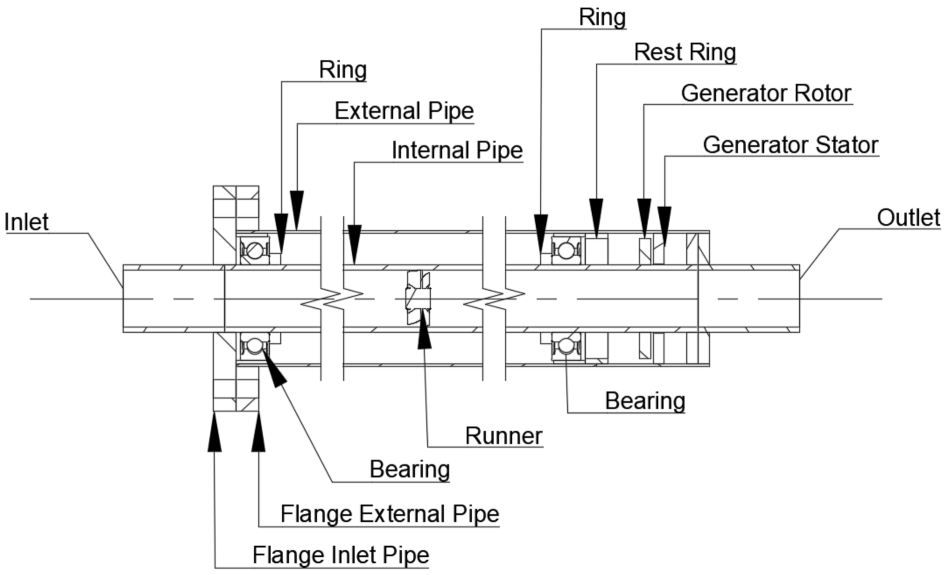


Figure 5.4: Final design sketch.

Chapter 6

Results and Discussion

6.1 3D CFD Model

The objective of the CFD model is to calculate power output of the runner by calculating the forces and the torque acting on the runner and the total pressure loss. In the initial design the turbine and generator are intended to extract approximately 15 % of the inlet pressure and convert it to electrical power. The CFD model estimates a pressure drop of approximately 1.5 % if the inlet pressure is 1 bar, which is one order of magnitude smaller than the design aimed for. To improve the turbine guide vanes should be installed.

The axial forces acting on the bearings are shown in table 6.1. The axial forces $\sum F_z$ are very small compared to the axial forces the bearings are capable to handle. This confirms the assumption that the bearings are oversized for this purpose.

Table 6.1: Forces and pressure drop in the axial direction.

Model	$F_{z,pressure}$ [N]	$F_{z,viscous}$ [N]	$\sum F_z$ [N]	Δp_s [Pa]	Δp_s [%]
Laminar	-0.4838	-0.02376	-0.5068	1475.18	1.475
k-ω-SST	-0.4807	-0.03040	-0.5111	1492.82	1.492
Difference:	$3.1 \cdot 10^{-3}$	$-6.64 \cdot 10^{-3}$	$4.3 \cdot 10^{-3}$	-17.64	-0.017

The torque acting on the runner creates a power potential given in table 6.2. To determine the real power output of the generator, several unknowns have to be determined empirically. To rotate the runner the flow must produce enough torque to overcome the generator's inertia and the friction in the turbine. Because non of the parts of the generator are made, it is impossible to calculate the rotational speed accurately, thus the rotational speed set in this thesis has to be changed with the result of an empirical value determined after the generator is built. Depending on the torque created by the runner, the friction (static/-dynamic) in the bearings and the electrical load of the generator, the rotational speed will

vary until it settles on its equilibrium (no acceleration). It is also notable that this rotational speed must satisfy the requirements to generate electricity.

Table 6.2: Torque about z-axis and power.

Model	$T_{z,pressure}$ [Nm]	$T_{z,viscous}$ [Nm]	$\sum T_z$ [Nm]	P_{CFD} [W]
Laminar	$3.4065 \cdot 10^{-3}$	$-1.71308 \cdot 10^{-4}$	$3.2352 \cdot 10^{-3}$	0.0196
k-ω-SST	$3.3924 \cdot 10^{-3}$	$-2.22364 \cdot 10^{-4}$	$3.1700 \cdot 10^{-3}$	0.0194

With the set rotational speed (meaning the bearings friction has been overcome), OpenFOAM estimates the turbine to produce approximately 20 mW. However, the generator will cause losses and the turbine output has to be multiplied by the generator efficiency factor to find the total power output.

6.2 Verification

Verification of the calculated torque can be conducted using control volume analysis. The analysis evaluates if the sum of forces acting on the control volume is equal to 0, in accordance with Newton's 1st law. Equation 6.1 is given by the systems force balance where the denominator is the systems driving force, while the numerator is the sum of all forces acting on the runner. An additional force contributed by the pipe wall is neglected because it is much smaller than forces acting on the runner, this implies that the expected result should be less than one.

$$error = \frac{\sum F}{\Delta p \cdot A_{pipe}} = \frac{\sum F_z + \frac{\sum T_z}{\bar{r}}}{\Delta p \cdot A_{pipe}} \tag{6.1}$$

In equation 6.1, $\bar{r}$ is the length of the leverage arm and Δp is the pressure drop from inlet to outlet. The undetermined leverage arm can be estimated by assuming that the torque acting on the runner blade is produced by a distributed load that can be transformed into a point force. The function of the distributed load on the runner blade is unknown, but does increase radially by r^2 due to the $\frac{dA}{A}$-factor in equation (2.2). Using this assumption the leverage arm must be located at $\bar{r} = 0.78R_{pipe}$ in the radial direction (derived in section 2.2.1).

Table 6.3: Verification of calculated forces.

Model	*error*
Laminar	107.89 %
k-ω-SST	106.39 %

Table 6.3 summarizes these calculations. The results show that the forces acting on the runner are larger than the driving force. This is problematic and causes a major concern

about where the calculation is wrong. Most likely this is a result of the sacrifices in the mesh generation.

6.3 Validation

Validation of the CFD model is a hard challenge without any external results to compare. A control volume analysis can be conducted over a volume containing the runner. The derivation uses the linear momentum equation to calculate the force acting on a control volume given by the circumference at radius r, length of the runner blades L_{runner} and the cylindrical slice thickness dr. This calculates point forces along the runner blades in the radial direction and is used to calculate the power contribution from each section which is integrated to find the total power of the runner. It is based on an inviscid flow where the outlet flow angle is equal to the outlet angle of the runner blade. This gives a maximum potential power output available for the turbine. The derivation is given in section 2.2.

Table 6.4: Comparing the theoretical maximum power output (P_{CV}) and the estimated power output by the CFD model (P_{CFD}).

	P_{CFD} [W]	P_{CV} [W]	P_{CFD}/P_{CV}[-]
Power	0.0194	0.01335	1.45

The calculated power output is summarized in table 6.4 and estimates that the power output calculated by the control volume analysis is greatly exceeded by the CFD analysis. This renders the CFD model unfit for further use and has to be modified (change mesh). Due to the overestimated forces acting on the runner (section 6.2) and the overestimated power output, it is without doubt that the torque calculated by the CFD model is inaccurate because it is the common denominator.

6.4 Runner Efficiency

When the runner efficiency is evaluated, it is important to address the use of the power that is taken from the pressure loss. To find the runner efficiency η_{runner} the power created by the runner is divided by the power from the pressure drop.

$$\eta_{runner} = \frac{P_{CFD}}{\dot{V} \cdot \Delta p} = \frac{T_z \cdot \omega_{rot}}{\dot{V} \cdot \Delta p} \tag{6.2}$$

The variable T_z is the momentum about the z-axis, ω_{rot} is the rotational velocity, $\dot{V}$ is the volumetric flow rate and Δp is the pressure loss from inlet to outlet.

The efficiency of the runner is quite low compared to other turbine arrangements such as Kaplan turbines (Brekke, 2003, section 9.5.4). From the start of this thesis it has been known that the efficiency of the turbine would be lower than commercial turbines due to its simplistic geometry and by omitting the guide vanes in the design. This causes the

runner to lose more than 96 $\%$ of the energy extracted from the flow to create a swirl behind the runner, while less than 4 $\%$ is used to drive the runner. This indicates that the simplistic runner can be improved significantly by installing guide vanes and using a more sophisticated design criteria to create a vortex free outlet.

Table 6.5: Runner efficiency.

Model	P_{CFD}[W]	$\dot{V} \cdot \Delta p_s$ [W]	η_{runner} [-]
Laminar	0.0196	0.4921	0.0398
k-ω-SST	0.0194	0.4976	0.0390

From a practical CFD point of view, including the guide vanes will require more cells due to the new imported geometry. Due to the large cell count in the current simulation it is not viable to have another geometry present. This begs the conclusion that with the writers current user knowledge of *snappyHexMesh*, it is not suitable for this project. The effort to use SALOME as a mesh generator was not successful due to low mesh quality, and introducing a new geometry will not improve the problems (section 4.3), however a commercial mesh generator such as ANSYS could yield better results.

6.5 Turbulence

The turbulence is causing a decrease of power output and efficiency. The increased viscous friction results in a decrease in net torque implying a reduction in the power output. The viscous friction also causes a larger non-contributing pressure loss yielding a lower runner efficiency. This is in line with viscous pipe flow theory (White, 2016, chapter 6.6). The effects of turbulence are shown in table 6.6

Table 6.6: Effects of turbulence using the k-ω-SST model (subscript tm denotes turbulence model and l denotes laminar).

P_{tm} -P_l [mW]	$\frac{P_{tm}-P_l}{P_{tm}}$[-]	$\frac{\eta_{tm}-\eta_l}{\eta_{tm}}$[-]	$\frac{T_{z,tm}}{T_{z,l}}$ [-]	$\frac{\Delta p_{s,tm}-\Delta p_{s,l}}{\Delta p_{s,tm}}$
-0.02	-0.01	-0.0205	1.3	0.01

Chapter 7

Conclusion

The goal of this master thesis is to determine if a turbine in a 1 inch water pipe can produce enough power to run low powered sensory equipment such as a controlling unit for a urinal. The CFD results (not validated) argue that the runner can produce 19.4 mW if it can reach a stationary rotational speed of 6.125 rad/s. This is only a fraction of the 1-10 W that is needed to run the controlling unit. The reasons for this discrepancy are the low efficiency (4 %) of the runner and that the runner is not capable of producing a large enough pressure drop to be able to generate the required power.

7.1 Runner Design

Prototype A and B are proven to be lackluster and a more complex design with guide vanes is advantageous because this will increase the runner efficiency. Prototype A proved to have a design flaw where the flow passed through the hollow center of the runner. Because the rotational velocity is set, the CFD model estimated a negative torque and at no rotation the runner produces an insignificant amount of torque. The design of prototype B is interesting because, in theory, the 2D model is acceptable. However, the simplification of constructing the runner blade with uniform blade angles proved erroneous even though the runner has a small diameter. During the writing of this thesis an error in the Bash-script that edits the case files of the 2D CFD model has been found. This error in the 2D model causes the inlet velocity to be constant while the hub diameter is changing, this defies continuity. The benefit creating this design is the experience with automating text-file manipulation using a Bash-script and how to run and extract results from a series of simulation in OpenFOAM.

Prototype B has been manufactured using a FDM 3D printer with a sufficient surface quality (can be improved with filler and sandpaper). However, a SLA printer will provide a much better runner in regard to both structural integrity and surface quality.

7.2 CFD Model

Because the CFD model is not validated there are several key-aspects that is worth exploring. The turbulence is modeled using the k-ω-SST model and the laminar model (no turbulence). Two other models have been tried; k-ϵ and k-ω, but they are disregarded due to instability. This raises the question to why k-ϵ and k-ω produces instability and k-ω-SST model does not. Especially when k-ω-SST model is a blend of the former. During the meshing it is found that all the problematic cells with high non-orthogonality are located close to the surfaces. This raises a question to whether these cells are playing a part in causing the instability or if there is another source to this problem.

The largest concern with the CFD model is the large number of cells required to make a mesh with barely acceptable non-orthogonality ($< 70^{\circ}$). This is mainly because SHM is trying to mesh an internal geometry where the mesh is restricted by the features of the runner and the pipe wall. Another related issue is to create a smooth surface on the runner geometry. This is caused by a misalignment during the snapping stage where the algorithm is trying to fit the runner geometry onto the O-grid mesh. This causes a deviation on the surfaces of the runner represented with jagged surfaces and edges. This imperfection is not all bad, the jagged surface feature resembles the same kind of imperfection as a FDM 3D print which is intended for the laboratory experiment. Arguably these two imperfections resemble each other.

7.3 Laboratory Experiment

The laboratory experiment was canceled due to the COVID-19 pandemic. At the current state the laboratory experiment is designed, but the implementation of the sensory equipment is lacking. This raises some concerns because the design might change according to which sensors that needs to be implemented to measure the rotational speed, flow rate, pressure drop and torque.

Chapter 8

Further Work

This chapter summarizes ideas, discoveries and tasks that have not been explored or completed.

- The simplistic runner design is inadequate to generate enough power and the efficiency proves that a more in-depth design including guide vanes is necessary to reach the required power generation.
- The mesh function of both OpenFOAM and SALOME is used with variable results. Due to inexperience with meshing, the poor mesh can be a result of user error or that meshing a complex geometry is more complicated than most tutorials are made for. A new mesh is mandatory in any continuation of this work.
- The modeling of turbulence has some unexpected results and would be interesting to explore further. There are also other ways of modeling turbulence available in OpenFOAM such as Reynolds stress transport models.
- Implementing sensory equipment into the laboratory experiment design and purchase/manufacture needed parts.
- Validation of CFD results by comparing to the data from the laboratory experiment.
- Depending on which concept chosen in section 5, a generator has to be purchased or built. Concept 2 also requires a gear and pinion to be designed and manufactured.

Bibliography

Amdahl, G.M., 1967. Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, spring joint computer conference, pp. 483–485.

ANSYS, 2020. Ansys fluent 12.0/12.1 documentation. URL: https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node238.htm.

Arnold, D.P., 2007. Review of microscale magnetic power generation.

Brekke, H., 2003. Pumper & Turbiner.

Brekke, H., 2008. Konstruksjon av pumper og turbiner.

Foundation, O., 2010. komegasstbase.h. URL: https://github.com/OpenFOAM/OpenFOAM-7/blob/master/src/TurbulenceModels/turbulenceModels/Base/kOmegaSST/kOmegaSSTBase.H.

Greacen, C., Kerins, M., 2010. Pump as turbine (pat) manual. URL: https://energypedia.info/wiki/File:Pump_as_Turbine_(PaT)_Manual.doc.

Greenshields, C., 2019. Openfoam v7 user guide. URL: https://cfd.direct/openfoam/user-guide.

Guerrero, J., 2014. Openfoam introductory training. URL: https://doi.org/10.6084/m9.figshare.16783657.

Guerrero, J., 2019a. Supplement meshing with blockmesh. URL: http://www.wolfdynamics.com/wiki/meshing_OF_blockmesh.pdf.

Guerrero, J., 2019b. Tips and tricks. URL: http://www.wolfdynamics.com/wiki/tipsandtricks.pdf.

Guerrero, J., 2021. Turbulence modeling in openfoam: Theory and applications. URL: http://www.wolfdynamics.com/training/turbulence/OF2021/turbulence_2021_OF8.pdf.

Harlow, F.H., Welch, J.E., 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. The Physics of Fluids 8, 2182–2189. URL: https://aip.scitation.org/doi/abs/10.1063/1.1761178, doi:10.1063/1.1761178, arXiv:https://aip.scitation.org/doi/pdf/10.1063/1.1761178.

Ikhsanul, A., 2017. Salome & openfoam tutorial: Propeller - preparing the mesh. URL: https://www.youtube.com/watch?v=z8J6euEVCvg&t.

Jasak, H., 1996. Error analysis and estimation for the finite volume method with applications to fluid flows .

Jasak, H., 2009. Dynamic mesh handling in openfoam. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2009-341, doi:10.2514/6.2009-341, arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2009-341.

Listek, V., 2019. Shipboard 3d printing expanding worldwide. URL: https://3dprint.com/240346/shipboard-3d-printing-expanding-worlwide/.

Menter, F., Esch, T., 2001. Elements of industrial heat transfer predictions, in: 16th Brazilian Congress of Mechanical Engineering (COBEM), p. 650.

Menter, F.R., Kuntz, M., Langtry, R., 2003. Ten years of industrial experience with the sst turbulence model. Turbulence, heat and mass transfer 4, 625–632.

Nielsen, T.K., 2014. Turbiner - virkningsgrader og slukeevne. URL: https://docplayer.me/17071327-Turbiner-virkningsgrader-og-slukeevne.html.

openfoam.com, 2019. Openfoam muscl. URL: https://develop.openfoam.com/Development/openfoam/blob/OpenFOAM-v2006/src/finiteVolume/interpolation/surfaceInterpolation/limitedSchemes/MUSCL/MUSCL.H.

OpenFoam.com, 2019. Openfoam: User guide v1912. URL: https://www.openfoam.com/documentation/guides/latest/doc/guide-meshing-snappyhexmesh.html.

openfoam.com, 2020. 3.2 running applications in parallel. URL: https://www.openfoam.com/documentation/user-guide/3-running-applications/3.2-running-applications-in-parallel.

Santos, B., 2015. Paraview slice type - cylinder. URL: https://www.cfd-online.com/Forums/paraview/117815-paraview-slice-type-cylinder-2.html.

SKF, 2020. Skf 6006-2rs1. URL: https://www.skf.com/group/products/rolling-bearings/ball-bearings/deep-groove-ball-bearings/productid-6006-2RS1.

Versteeg, H., Malalasekera, W., 2007. An Introduction to Computational Fluid Dynamics: The Finite Volume Method. Pearson Education Limited. URL: https://books.google.no/books?id=RvBZ-UMpGzIC.

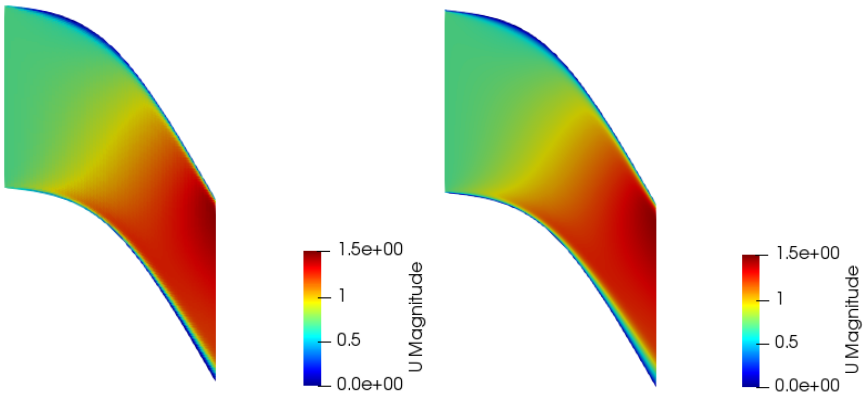
Wani, F., Dong, J., Polinder, H., 2020. Tidal turbine generators, in: Advances in Modelling and Control of Wind and Hydrogenerators. IntechOpen.

Weiß, C., 2015. Transform matrix. URL: https://www.cfd-online.com/Forums/paraview/117815-paraview-slice-type-cylinder-2.html.

White, F., 2016. Fluid Mechanics. McGraw-Hill series in mechanical engineering, McGraw-Hill Education. URL: https://books.google.no/books?id=7AEzjwEACAAJ.

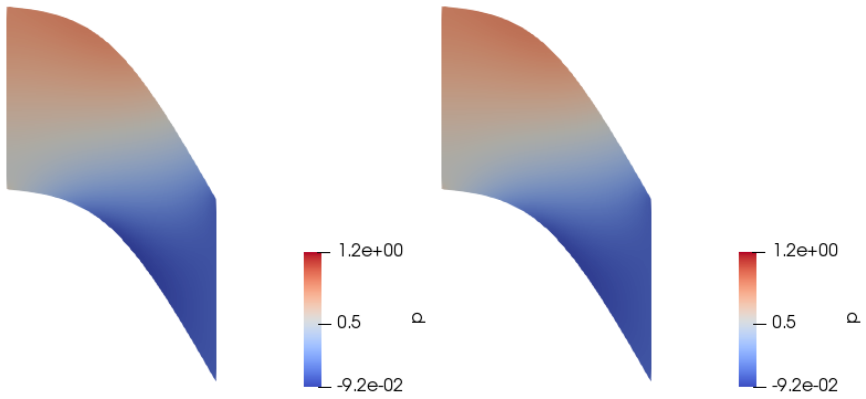
Appendices

A 3D Model Comparison with the Prototype B 2D Model



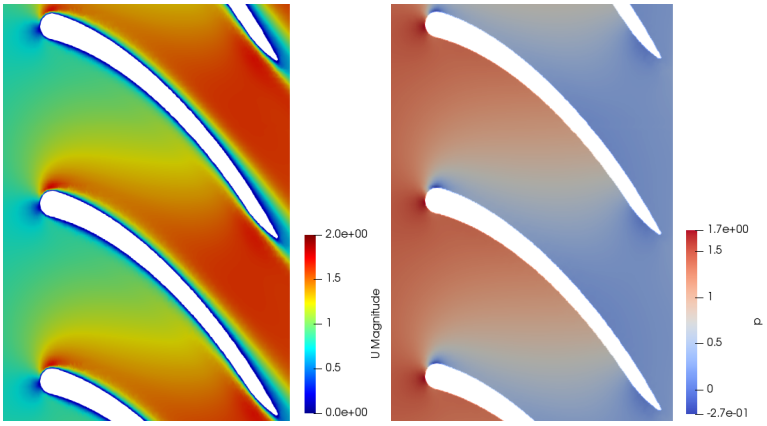
(a) Laminar. **(b)** k-ω-SST.

Figure 1: Velocity plot, 2D simulation at $^{r}/_{R} = {}^{3}/_{4}$.



(a) k-ω-SST. **(b)** Pressure.

Figure 2: Kinematic pressure plot, 2D simulation at $^{r}/_{R} = {}^{3}/_{4}$.



(a) Velocity **(b)** Pressure

Figure 3: k-ω-SST, Cylindrical plot, 3D simulation at $r/R = 3/4$.

By comparing the velocity fields of the 2D and the 3D simulation, it is evident that the 3D model has a higher inlet and maximum velocity. Upon investigation and revision of the script controlling the 2D model, it is evident that the script has failed to either calculate or set the adjusted inlet velocity caused by the blockage of the hub diameter. This results in a lower inlet velocity (axial direction) than experienced in the 3D model.

B Control Volume Calculation

```
clear all
close all
clc

%Result from 3D CFD-model (k-omega-SST)
P_CFD = 0.0194; %[W]

%Density
rho = 1000;  %[kg/m3]

%Rotational speed
omega_rot = 1*6.125; %[rad/s]

%Inlet velocity
u_inlet = 0.697; %[m/s]

%Total volume flowrate (Q = dot(V))
Q_tot = 20/60*10^-3;%[m3/s]

%Radius
r_pipe = 0.0254/2; %[m]
r_hub = 0.01; %[m]

%Outlet angle (from CAD model)
alpha_2 = 59*pi/180; %[rad]

%Number of sections
N_sections = 10^3; %[-]

%Section width
dr = (r_pipe-r_hub)/N_sections; %[m]

%Total unoccupied area of turbine
A_tot = pi*(r_pipe^2-r_hub^2); %[m2]

%Total area of pipe
A_pipe = pi*(r_pipe^2); %[m2]

%Variables
A_sum = 0;
Q_sum = 0;
Q = 0;
P = 0;
P_z = 0;
```

```
45
46 R = 0.01+0.5*dr:dr:0.0127-0.5*dr;
47
48 for r = 0.01+0.5*dr:dr:0.0127-0.5*dr
49     %Calculate inlet angle to turbine, function of radius
50     theta_1 = atan(omega_rot*r/(u_inlet*(A_pipe/A_tot)));
51
52     %Calculate cross-section area of the turbine blade
53     dA = pi*((r+0.5*dr)^2-(r-0.5*dr)^2);
54
55     %Checking that the area pieces adds up to total cross
           sections area
56     A_sum = A_sum+dA;
57
58     %Calculating force on CV, using dA/A_tot to calculate
           mass flow through dA
59     F_theta = rho*Q_tot*(dA/A_tot)*sqrt(u_inlet^2+(
           omega_rot*r)^2)*(sin(alpha_2)-sin(alpha_1));  %[N]
60
61     %Checking that the volume flow rate adds up to total
           volume flow rate
62     Q_sum  = Q_sum + Q*(dA/A_tot);
63
64     %Calculate Torque of section
65     T = F_theta*(r); %[Nm]
66
67     %Add together total power of the turbine
68     P = P + T*omega_rot; %[W]
69 end
70
71 %check that sum of area and volume flow rate correct
72 if ((A_sum/A_tot-1) < N_sections) && ((Q_sum/Q_tot-1) <
       N_sections)
73     disp('Sum area and Sum Q is OK')
74 else
75     disp ('Sum area and Sum Q is not OK')
76 end
77
78 disp(['P/P_CFD: ', num2str(P/P_CFD)])
```

C Cylinder to 2D Plane Filter

In order to compare the fields created by the 2D model and the 3D model a cylinder to plane filter was created. The filter created in Paraview uses the function ”Programmable filter” where the following code is entered in script.

“input = self.GetInputDataObject(0,0)
inp_copy = input.NewInstance()
inp_copy.ShallowCopy(input)
inp_copy.UnRegister(None)
cutter = vtk.vtkCutter()
transf = vtk.vtkTransform()
transf.RotateX(90)
cyl = vtk.vtkCylinder()
cyl.SetCenter(0,0,0)v cyl.SetRadius(0.009375)
cyl.SetTransform(transf)
cutter.SetCutFunction(cyl)
cutter.SetInputData(inp_copy)
cutter.Update()
self.GetOutputDataObject(0).ShallowCopy(cutter.GetOutputDataObject(0))”

Posted online at cfd-online.com by Wyldckat/Santos (2015). The code creates a hollow cylinder, without front or back wall and at a predefined radius as seen in figure 4.

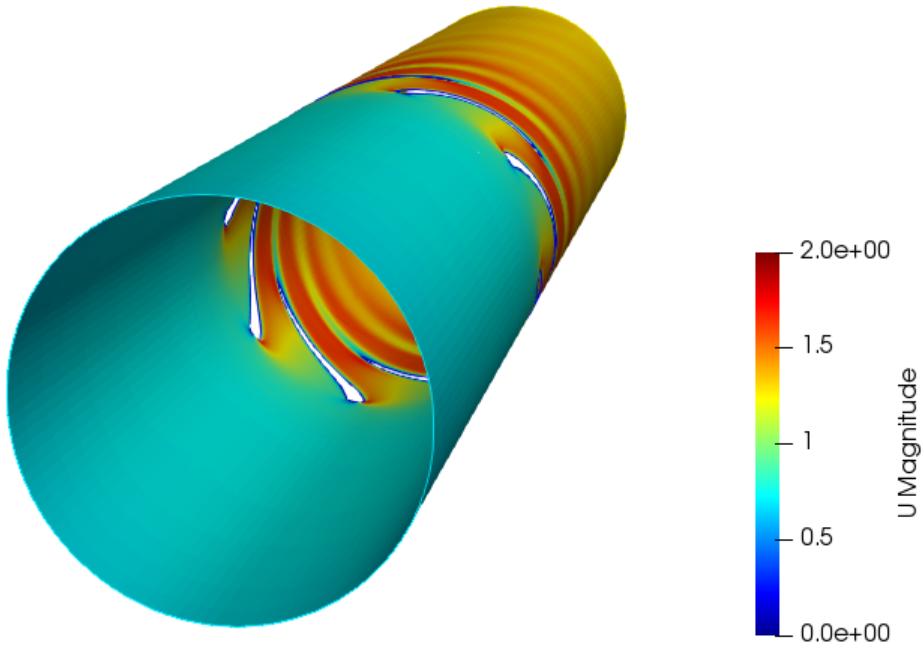


Figure 4: Cylindrical cut created by programmable filter.

Before applying the calculator function, the cylinder must be cut in half using the clip-function. The calculator uses a transformation vector to project the half cylinder wall onto a flat 2D plot shown in figure 5. The calculator equation is the following:

"$0 * iHat + (asin(coordsY/(coordsX^2 + coordsY^2)^(1/2)) * (coordsX^2 + coordsY^2)^(1/2)) * jHat + coordsZ * kHat$"

posted by Nephi/Weiß (2015).

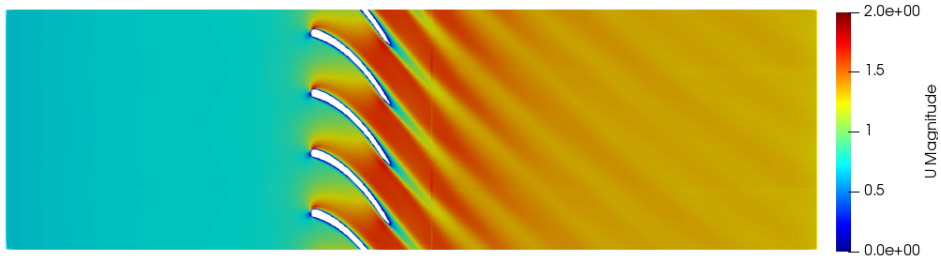
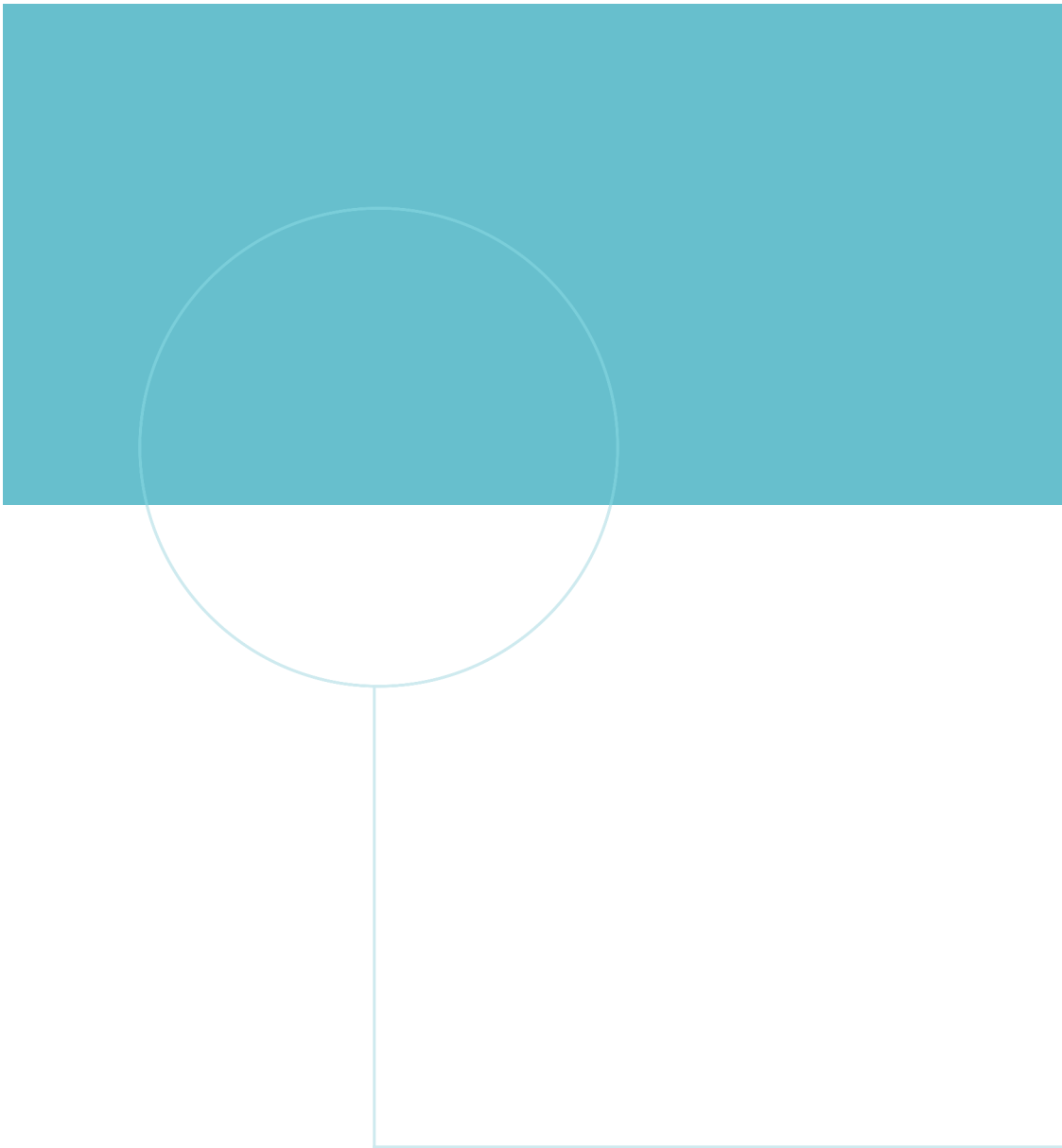


Figure 5: Cylindrical cut transformed to a 2D surface by the calculator function.

D Slurm-Script

```
#!/bin/sh
#############################################################
#  Running openFOAM cluster job                             #
#############################################################
#
#SBATCH --partition=CPUQ                # partition the batch job will be put in
#SBATCH --account=share-iv-ept
#SABTCH --account=sindrtr                # Account for consumed resources
#SBATCH --time=120:00:00                 # the walltime length of the job (eg. 30 minutes)
#SBATCH --nodes=2
#SBATCH --mem=40GB                      # number of nodes requested
#SBATCH --ntasks-per-node=20             # number of processes per node
#SBATCH --job-name="turbine"  # name of the job
#SBATCH --output=turbine.out  # name of output file
#SBATCH -J turbine            # Name for the job

WORKDIR=${SLURM_SUBMIT_DIR}

echo "we are running from this directory: $SLURM_SUBMIT_DIR"
echo " the name of the job is: $SLURM_JOB_NAME"
echo "Th job ID is $SLURM_JOB_ID"
echo "The job was run on these nodes: $SLURM_JOB_NODELIST"
echo "Number of nodes: $SLURM_JOB_NUM_NODES"
echo "We are using $SLURM_CPUS_ON_NODE cores"
echo "We are using $SLURM_CPUS_ON_NODE cores per node"
echo "Total of $SLURM_NTASKS cores"

module purge
module load GCC/8.3.0
module load OpenMPI/3.1.4
module load OpenFOAM/7

source $FOAM_BASH

#run the application:
decomposePar >log.decomposePar
mpirun -np 40 pimpleFoam -parallel 2>&1 | tee log.pimpleFoam
```

```
38 reconstructPar >log.reconstructPar
39
40 uname -a
```



NTNU

Norwegian University of
Science and Technology