University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Construction and materials, specialization within mechanical constructions	Spring semester, 2015 Open
Writer: Ola Sirevaag	…… (Writer's signature)
Faculty supervisor: Bjørn Helge Hjertager External supervisor(s): Knut Erik Giljarhus	
Thesis title: CFD simulation of an offshore air intake and exhaust system	
Credits (ECTS): 30	
Key words: Pressure loss Fan OpenFOAM rhoSimpleFoam CFD	Pages: 61 + enclosure: 58 + CD Stavanger, 12.06.2015

ABSTRACT

The main purpose is to investigate whether the exhaust gases from an offshore turbine can be rerouted to heat the air entering the turbine system, thus keeping air humidity concentration above acceptable levels. To ensure this, temperature of the incoming airflow must be above 4,5 degrees Celsius. Currently the exhaust is vented out to the atmosphere and an electrical anti-icing system is used to heat the air intake. The objective of this thesis is therefore to make a CFD model in OpenFOAM to simulate the two proposed pipe designs that will connect the exhaust to the air intake. By evaluating the results from the simulation, a conclusion shall be drawn to whether the new pipe layout is a viable solution and can replace the current anti-icing system, thus saving money and electrical power. A critical part of the new system is the existing exhaust fan, which will be used to drive the exhaust gases towards the front of the air intake. With the provided fan performance curve, time has been spent to find the appropriate boundary condition that will simulate the fan behavior, and its effect on the flow conditions as accurately as possible. The same goes for the exhaust heat dissipation through the pipe wall, and to the surrounding environment. Operation conditions include cross winds up to 10m/s and atmospheric temperatures down to negative 10 degrees Celsius. To incorporate this into the CFD model, appropriate manual calculations had to be performed beforehand to find the local heat transfer coefficient. The system size and air flow velocities, results in a relatively large mesh model. Therefore, to ensure as low computational execution times as possible, multiple meshing settings are explored to ensure as few excess cells as possible. Temperature and flow results from the simulations shows that fan performance and both pipe designs are more than adequate to ensure turbine operation. Mesh quality is also verified using the yPlus value. However, in the attempt to confirm the simulated pressure loss with the Bernoulli equation, the loss coefficient found in reference literature does not produce an accurate result. In addition, it is found that the OpenFOAM documentation does not specify the unit of measurement used in the fan curve specification. Without being able to determine whether the fan curve should be defined using mass flow or volumetric flow, the final CFD models have a degree of inaccuracy regarding the fan behavior. Nevertheless, simulation results have a high enough tolerance, relative to system requirements, that the fan can be cleared suitable for operation.

TABLE OF CONTENTS

NOMENCLATURE

Latin symbols

a	Channel width	[m]
A	Area	$[m^2]$
b	Channel height	[m]
C	Equation constant	
C_μ	Equation constant	
C_1	Equation constant	
C_2	Equation constant	
d	Equation constant	
d_h	Hydraulic diameter	[m]
D	Diameter	[m]
e	Internal energy	$[m^2 \cdot kg/s^2]$
f	Friction factor	
F	Body forces	
g	Gravity	$[m/s^2]$
h	Enthalpy	[J]
h_{air}	Air heat transfer coefficient	$[W/m^2 \cdot K]$
h_{total}	Total heat transfer coefficient	$[W/m^2 \cdot K]$
I	Turbulence intensity	
k	Turbulent kinetic energy	$[m^2/s^2]$
k_T	Thermal conductivity	$[W/m \cdot K]$
k_{loss}	Loss coefficient	
l	Mixing length	[m]
L	Pipe length	[m]
m	Equation constant	
$\dot{m}$	Mass flow	[kg/s]
n	Number of moles	
Nu	Nusselt number	
p	Pressure	$[N/m^2]$
Pr	Prandtl number	
Pr_t	Turbulent Prandtl number	
q	Heat flux	$[W/m^2]$
r	Surface roughness	[mm]
R	Universal gas constant	[J/mol·K]
Re	Reynolds number	
Re_{d_h}	Reynolds number based on hydraulic diameter	
S	Wall thickness	[m]
$\dot{S}$	Source term	
t	Time	[s]
T	Temperature	[K]
U	Velocity	[m/s]
U_{Wind}	Wind velocity	[m/s]

$\widetilde{U}$	Favre-averaged velocity	[m/s]
V	Volume	[m^3]
$\dot{V}$	Volumetric flow	[m^3/s]
$\tilde{V}$	Favre-averaged velocity	[m/s]
w	Velocity in z-direction	[m/s]
$\widetilde{W}$	Favre-averaged velocity	[m/s]

Greek symbols

α_t	Turbulent thermal diffusivity	[m^2/s]
γ	Weight density	[kg/$m^2 \cdot s^2$]
δ	Unit tensor	
ε	Dissipation of turbulent kinetic energy	[m^2/s^3]
μ	Viscosity	[kg/m·s]
μ_t	Turbulent viscosity	[m^2/s]
ν	Kinematic viscosity	[m^2/s]
ρ	Density	[kg/m^3]
ρ_{air}	Density of air	[kg/m^3]
σ_ε	Prandtl number k-equation	
σ_k	Prandtl number ε-equation	
τ	Viscous stress	[N/m^2]

Abbreviations

RAM	Random access memory
RANS	Reynolds-averaged Navier-Stokes
STL	Stereolithography
Rpm	Revolutions per minute

1 INTRODUCTION

1.1 Background

Inspiration for this thesis was drawn from a project given by Conoco Phillips to Aibel in the autumn of 2014. The original scope of work was to replace a set of corroded ducts that was a part of the air intake for an offshore generator. The duct draws air from the atmosphere and directs it down to the generator. However, when temperatures drop down towards the freezing point, the air humidity levels get too high for generator operation. To solve the problem, an electrical heater has been used to keep the air temperature at an acceptable level. The heater itself draws electricity from the generators, leaving less power available for essential equipment on the platform. A proposal was made to redirect the generators own exhaust flow, to the front air intake. Using the exhaust flow to heat incoming air would make the electric anti-ice system obsolete. Unfortunately, with the downsizing and cutbacks made in the oil industry during the end of 2014 and start of 2015, work was never commenced on this project. This means that there will be no official report or documentation available to build the thesis on. Instead, this thesis will be written independently from Aibel and use the limited original scope of work as inspiration to create and solve a theoretical case.

1.2 Thesis objective

The main objective of this thesis is to develop a simulation in OpenFOAM that describes the pressure, mass flow and temperature conditions inside the duct. The model includes turbulent flow and temperature loss to the environment. Two different design cases under different operating conditions shall be tested to find the optimal duct layout. One key feature of the new system is the existing fan driving the exhaust flow. Previously its only task was to vent exhaust gases out into the atmosphere. Now it will have to drive the exhaust flow all the way towards the air intake. The fan features has to be tested to see if they are sufficient to complete this task, and thus save money by not replacing it with a new fan. The main criterion for the new system will be the airflow temperature entering the generator. As long as the temperatures are kept above recommended levels, then the new system will be an improvement to the existing electrical heaters.

2 THEORY

2.1 Governing equations

In the following chapter, the equations have been gathered from Hjertager (2009A & 2009B) and Versteeg et al. (2007). The governing equations in fluid dynamics are the mathematical representation of the conservation laws of physics. There are eight unknown variables, and thus there are eight equations to solve.

2.1.1 Continuity equation

The continuity equation describes mass conservation. Mass conservation implies that the rate of increase of mass in a fluid element must be equal to the net rate of mass flow into the said fluid element. Continuity equation on vector form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \qquad \textit{Equation 1}$$

First term is transient, and describes the rate of change of density over time. Second term, the convective term, describes the net flow of mass out of the fluid element across its boundaries. For a steady state case, the transient term is neglected:

$$\frac{\partial}{\partial x_i}(\rho u_i) = 0$$

2.1.2 Momentum balance

Momentum balance is derived from Newton's second law and defines that the rate of change in momentum of a fluid particle equals the sum of the forces on the particle. The momentum balance on vector form:

$$\rho \frac{D\vec{v}}{Dt} = -\vec{\nabla} \cdot p + \vec{\nabla} \cdot \vec{\vec{\tau}} + \vec{F} \qquad \textit{Equation 2}$$

First term is the rate of increase of momentum in three dimensions per unit volume of a fluid particle. First two terms on the right side of *equation 2*, account for the surface forces pressure and viscosity. While the last term include the body forces.

2.1.3 Energy balance

The energy equation is derived from the first law of thermodynamics. The equation express that the increase of energy in a fluid particle equals the rate of heat added to fluid particle in addition to the rate of work done on said fluid particle.

$$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho u_i e)}{\partial x_i} = -\frac{\partial q_j}{\partial x_j} - p\frac{\partial u_i}{\partial x_i} + \tau_{ij}\frac{\partial u_i}{\partial x_j} + \dot{S} \qquad \textit{Equation 3}$$

Equation 3 is the first law of thermodynamics on differential form, where *e*, is the internal energy as dependent variable. Often beneficial to use the relation:

$$e = h - \frac{p}{\rho}$$

To express the energy equation with enthalpy as dependent variable instead of energy. Now referred to as the enthalpy equation:

$$\frac{\partial(\rho h)}{\partial t} + \frac{\partial(\rho u_i h)}{\partial x_i} = -\frac{\partial q_j}{\partial x_j} + \frac{Dp}{Dt} + \tau_{ij}\frac{\partial u_i}{\partial x_j} + \dot{S}$$

Where $\dot{S}$ is the source term. It can both be a source term and sink term depending on whether there is, for example, radiation or a chemical reaction present.

2.1.4 Equation of state

The equation of state for an ideal gas:

$$pV = nRT \qquad \textit{Equation 4}$$

The equation of state provide an important relation between the energy balance and the momentum and continuity equations, for compressible fluids.

2.1.5 Empirical relations

First of the empirical relations is Newton's law of viscosity:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu \cdot \frac{\partial u_i}{\partial x_i} \cdot \delta_{ij} \qquad \textit{Equation 5}$$

Lastly, the second law of thermodynamics also known as Fourier's law:

$$q_i = -k_T \frac{\partial T}{\partial x_i} \qquad \textit{Equation 6}$$

2.2 Reynolds-averaged Navier-Stokes

The chosen method of simulating turbulence is the RANS turbulence models. The models utilize a set of partial differential equations that relies on approximated mean values to be solved. These are known as the RANS equations, Versteeg et al. (2007):

Equation 7.1

$$\frac{\partial(\bar{\rho}\tilde{U})}{\partial t} + div\left(\bar{\rho}\tilde{U}\tilde{U}\right) = -\frac{\partial \bar{p}}{\partial x} + div\left(\mu\ grad\ \tilde{U}\right) + \left[-\frac{\partial\left(\overline{\bar{\rho}u'^2}\right)}{\partial x} - \frac{\partial(\overline{\bar{\rho}u'v'})}{\partial y} - \frac{\partial(\overline{\bar{\rho}u'w'})}{\partial z}\right] + S_{Mx}$$

Equation 7.2

$$\frac{\partial(\bar{\rho}\tilde{V})}{\partial t} + div\left(\bar{\rho}\tilde{V}\tilde{U}\right) = -\frac{\partial \bar{p}}{\partial y} + div\left(\mu\ grad\ \tilde{V}\right) + \left[-\frac{\partial(\overline{\bar{\rho}u'v'})}{\partial x} - \frac{\partial\left(\overline{\bar{\rho}v'^2}\right)}{\partial y} - \frac{\partial(\overline{\bar{\rho}v'w'})}{\partial z}\right] + S_{My}$$

Equation 7.3

$$\frac{\partial(\bar{\rho}\tilde{W})}{\partial t} + div\left(\bar{\rho}\tilde{W}\tilde{U}\right) = -\frac{\partial \bar{p}}{\partial z} + div\left(\mu\ grad\ \tilde{W}\right) + \left[-\frac{\partial(\overline{\bar{\rho}u'w'})}{\partial x} - \frac{\partial(\overline{\bar{\rho}v'w'})}{\partial y} - \frac{\partial\left(\overline{\bar{\rho}w'^2}\right)}{\partial z}\right] + S_{Mz}$$

When time averaging the momentum equations, additional terms appear. The extra terms are known as the Reynolds stresses; $-\rho\overline{u'^2}, -\rho\overline{v'^2}, -\rho\overline{w'^2}, -\rho\overline{u'v'}, -\rho\overline{u'w'}, -\rho\overline{w'v'}$. These will be predicted with the k-ε turbulence model. The main advantage of the k-ε model is that it requires less processing power than the alternatives.

2.3 k-ε model

The k-ε model utilize two extra transport equations deduced from the Navier Stokes equations. These equations have been sampled from Hjertager (2009B). The first is turbulent kinetic energy k and is described by the k-equation:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial}{\partial x_i}(\rho U_i k) = \frac{\partial}{\partial x_i}\left[\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_i}\right] - \rho\overline{u_i' u_j'}\frac{\partial U_i}{\partial x_j} - \rho \cdot \varepsilon \qquad \textit{Equation 8}$$

Here, first term is the transient term. Second is the convective term. Third term is the diffusive transport and must be modelled. Fourth term is turbulent kinetic energy based on the mean flow velocity. Fifth term is the viscous dissipation of turbulent kinetic energy, which also must be modelled.

The second transport equation is the dissipation of turbulent kinetic energy and is named the ε-equation. Only the modelled form will be given:

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial}{\partial x_i}(\rho U_i \varepsilon) = \frac{\partial}{\partial x_i}\left[\frac{\mu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_i}\right] + C_1\frac{\varepsilon}{k}\left[-\rho\overline{u_i' u_j'}\frac{\partial U_i}{\partial x_j}\right] - C_2\rho\frac{\varepsilon}{k}\varepsilon \qquad \textit{Equation 9}$$

Again, the first term is the transient. Second is the convective term. The third term represents diffusive transport. Fourth and fifth term is the production and destruction of ε respectively.

2.4 Theoretical pressure loss

During this thesis, evaluating the overall pressure loss and system performance will be done with OpenFOAM simulations. However, to validate the results some manual calculations has to be done as well. The formulas used to calculate the system pressure loss has been gathered from ASHRAE (2009). Whenever there is a closed system like a pipe, the air movements will be controlled by the three fundamental laws of physics: conservation of mass, conservation of momentum and conservation of energy. Conservation of mass means that mass can be neither created nor destroyed. This means that the sum of mass flow entering at the fan inlet will be exiting the air intake outlet. Also for these calculations, it is assumed that the density remains constant throughout the pipe system. Therefore, the average air velocity remains constant as the cross sectional area is the same.

Mass conservation:

$$\dot{m} = \int \rho U dA = constant$$

Then for two points within the same system:

$$\dot{m}_1 = \dot{m}_2$$

$$\rho_1 U_1 A_1 = \rho_2 U_2 A_2$$

Assuming constant density and pipe cross section:

$$U_1 = U_2$$

The second law, conservation of energy, implies that energy cannot disappear. Only transformed from one form to another. This is the basic principle behind the Bernoulli's equation. The Bernoulli's equation relates the system pressure, fluid velocity and elevation:

$$\frac{p}{\gamma} + \frac{U^2}{2g} + \Delta z = constant$$

Where :

$$\gamma = \rho g$$

Insert and rearrange the Bernoulli's equation:

$$p + \frac{\rho U^2}{2} + \rho g \Delta z = constant$$

Now the three terms within the equation can be described accordingly:

$$p = static\ pressure$$

$$\frac{\rho V^2}{2} = dynamic\ pressure$$

$$\rho g \Delta z = hydrostatic\ pressure$$

Bernoulli's equation for two points along the pipeline:

$$\frac{p_1}{\gamma} + \frac{U_1^2}{2g} + z_1 = \frac{p_2}{\gamma} + \frac{U_2^2}{2g} + z_2 \qquad \textit{Equation 10}$$

Lastly, there is conservation of momentum, describes that the amount of momentum within a system remains constant unless acted upon by external forces. This will be used to calculate the friction losses through the pipe. Although the Bernoulli's equation was derived for ideal frictionless flow along a streamline, it can be modified to analyze airflow with friction. To accomplish this, the Darcy equation is introduced:

$$\Delta p = f\left(\frac{L}{D}\right)\left(\frac{U^2}{2g}\right) \qquad \textit{Equation 11}$$

Here, f is a dimensionless friction factor obtained by calculating Reynolds number:

$$Re = \frac{DU}{\nu} \qquad \textit{Equation 12}$$

The relative roughness of the inside wall:

$$Relative\ roughness = \frac{r}{D} \qquad \textit{Equation 13}$$

The Reynolds number and relative roughness are then used in a Moody chart to find the friction factor, Hjertager (2013).

The Darcy equation allows for calculation of pressure drop caused by friction in a fully developed flow. However, to account for pressure losses in fittings and bends there is need for a second equation:

$$\Delta p = k_{loss}\left(\frac{U^2}{2g}\right) \qquad \textit{Equation 14}$$

Where k_{loss} is a loss coefficient depending on the pipe bend angle, radius and pipe diameter.

Combining the two equations, the total pressure loss due to friction is presented as following:

$$\sum h_{loss} = \left(\frac{fL}{D} + \sum k_{loss}\right)\left(\frac{U^2}{2g}\right) \qquad \textit{Equation 15}$$

Inserting this into the Bernoulli's equation:

$$\frac{p_1}{\gamma} + \frac{U_1^2}{2g} + z_1 = \frac{p_2}{\gamma} + \frac{U_2^2}{2g} + z_2 + \sum h_{loss}$$

$$p_1 - p_2 = (z_2 - z_1)\rho g + \rho\left(\frac{fL}{D} + \sum k_{loss}\right)\left(\frac{U^2}{2}\right)$$

Finally, the total pressure drop within the pipe:

$$\Delta p = \Delta z \rho g + \left(\frac{fL}{D} + \sum k_{loss}\right)\left(\frac{\rho U^2}{2}\right) \qquad \textit{Equation 16}$$

2.5 Heat transfer

As the exhaust flow from the fan towards the air inlet, some heat will dissipate through the 316L stainless steel wall of the pipe. The pipe itself has no insulation or shielding from wind. In the OpenFOAM model, it is possible to account for these environmental influences, but some of the factors need to be calculated by hand before it is used as input to the model. More specifically, it needs the heat transfer coefficient between the hot exhaust gas, steel pipe wall and the outside wind. The formulas used in this chapter was incorporated from Incropera et al. (2009). First, the Zukauskas relation is used to obtain the Nusselt number:

$$Nu = CRe_D^m Pr^d \left(\frac{Pr}{Pr_s}\right)^{1/4} \quad \textit{Equation 17}$$

The heat transfer coefficient for the air are as following:

$$h_{air} = Nu\frac{k_T}{D} \quad \textit{Equation 18}$$

Then the total heat transfer coefficient including the pipe wall is:

$$\frac{1}{h_{total}} = \frac{1}{h_{outside}} + \sum \frac{S_i}{k_{T_i}} \quad \textit{Equation 19}$$

2.6 Fan

One of the main aspects to this thesis is to investigate whether the old exhaust fan can be used to bring the exhaust gases to the front of the air intake. The fan in question is an axial fan type with a nominal duty of 15 m^3/s, airflow at a pressure of 1600Pa and speed of 1776 rpm. Two criteria needs to be fulfilled to make the current fan a viable option. First of all the total pressure loss from the fan location to the air intake must not be greater than 1800 Pa. Second, the flowrate must be great enough to deliver a high volume of hot air to heat up the cold air from the air intake. A representation of the fan performance curve provided by Aibel is shown in *figure 1*.

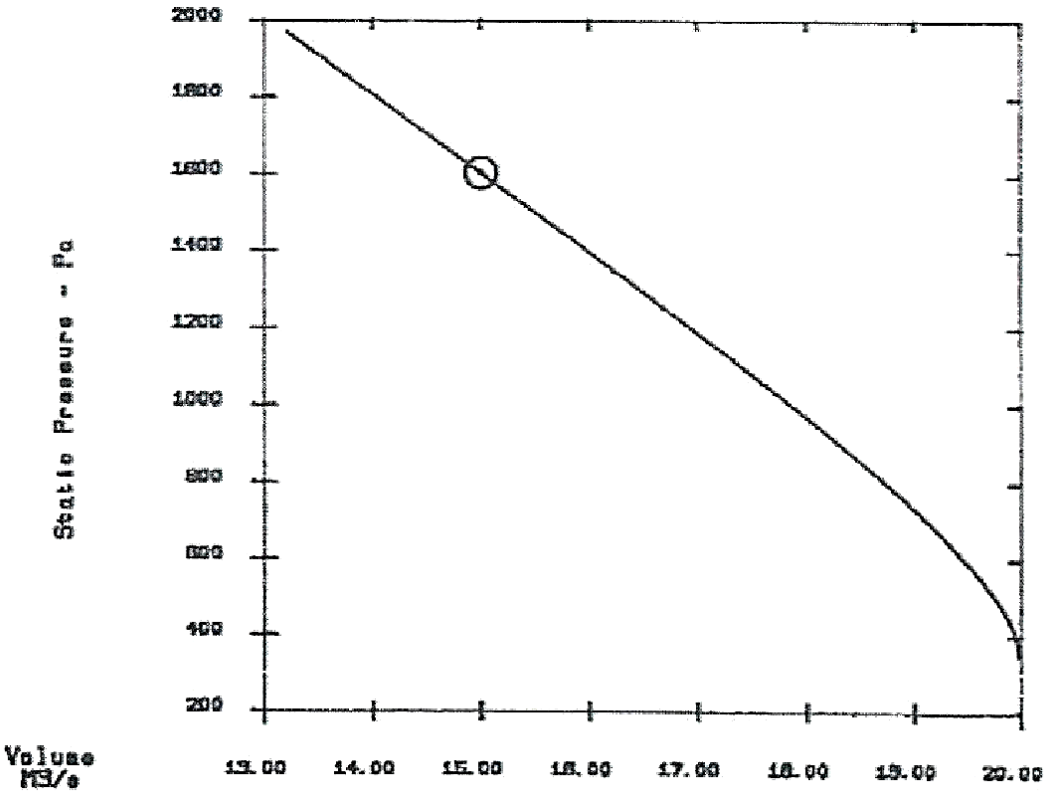


Figure 1: fan performance curve

2.7 Turbine

To ensure constant turbine operation, two criteria must be met. Firstly, the turbine requires a minimum air mass flow of 16 kg/s. Since the turbine was fully operational before the addition of exhaust flow, it is assumed that turbine suction is sufficient to maintain required mass flow at outlet without help of the fan. Unfortunately, there are no more available data describing the suction mechanism or pressure conditions. Secondly, the temperature of the air entering the turbine has to be above 4,5 degrees Celsius or 277,55 K. This is to ensure humidity levels below 70%. *Figure 2* describes the relative humidity based on air temperatures for the system and was provided by Aibel.

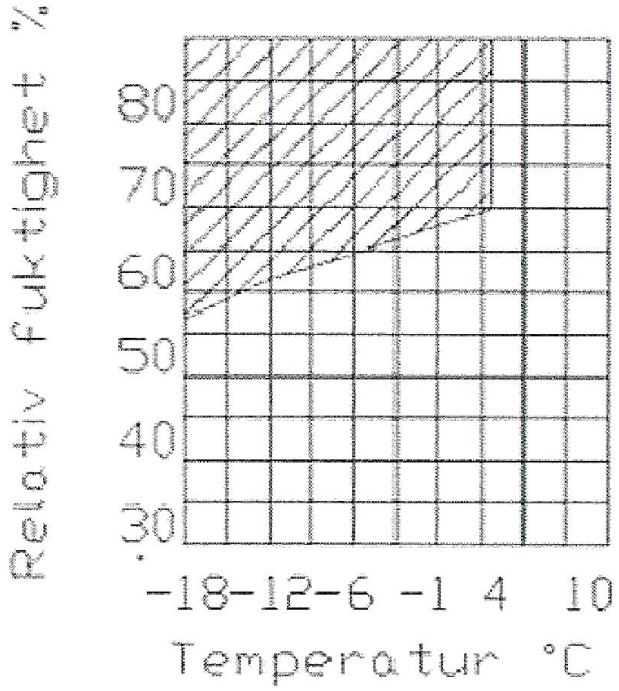


Figure 2: relative humidity

2.8 Technical data

2.8.1 Air properties at 1 atmospheric pressure

Table 1: air properties collected from table A-9 Hjertager (2013)

Temp T, Kelvin	Density ρ, kg/m^3	Dynamic Viscocity μ, kg/ms	Kinematic Viscocity v, m^2/s	Prandtl Pr	Thermal conductivity k_T, W/m·K
263,15	1,341	$1{,}68*10^{-5}$	$1{,}252*10^{-5}$	0,7387	0,02288
273,15	1,292	$1{,}729*10^{-5}$	$1{,}338*10^{-5}$	0,7362	0,02364
283,15	1,246	$1{,}944*10^{-5}$	$1{,}778*10^{-5}$	0,7336	0,02439
293,15	1,204	$1{,}825*10^{-5}$	$1{,}516*10^{-5}$	0,7309	0,02514
333,15	1,059	$2{,}008*10^{-5}$	$1{,}896*10^{-5}$	0,7202	0,02808

2.8.2 Constants for equation 17

Table 2: equation constants sampled from table 7.4 Incropera et al. (2009)

Re	C	m	d
1-40	0,75	0,4	0,37
40-1000	0,51	0,5	0,37
10^3-2×10^5	0,26	0,6	0,37
2×10^5-10^6	0,076	0,7	0,37

2.8.3 Properties of AISI 316 steel

Table 3: material properties gathered from table A.1 Incropera et al. (2009)

Stainless steel	Density ρ, kg/m^3	Thermal conductivity k_T, W/m·K
ASISI 316	8238	13,4

2.8.4 Loss coefficient

Table 4: fittings loss coefficient according to appendix A CRANE (1982)

Nominal size [mm]	100	125	150	200-250	300-400	450-600
Friction factor	0,017	0,016	0,015	0,014	0,013	0,012
k_{loss} 90° elbow	0,51	0,48	0,45	0,42	0,39	0,36
k_{loss} 45° elbow	0,272	0,256	0,24	0,224	0,208	0,192

Table 5: fittings loss coefficients from table A 8 Chappalaz et al. (1992)

Radius/Diameter	1	2	4	6	10
k_{loss} 15° elbow	0,03	0,03	0,03	0,03	0,03
k_{loss} 22,5° elbow	0,045	0,045	0,045	0,045	0,045
k_{loss} 45° elbow	0,14	0,09	0,08	0,075	0,07
k_{loss} 60° elbow	0,19	0,12	0,10	0,09	0,07
k_{loss} 90° elbow	0,21	0,14	0,11	0,09	0,08

3 OpenFOAM

3.1 Introduction

OpenFOAM is an open source CFD software written with the computer language of C++ and runs in Linux based operating systems. By being open source, OpenFOAM allows the user to alter and customize the functionalities to match the case requirements. Building on this, OpenFOAM has very few shortcomings and is limited mostly by the users programming skills. The open source code has led to a lively forum where users together solve problems and contribute with their own solver modifications and solutions, available for anyone to use. OpenFOAM operates using text files and unix style commands. The structure is built upon libraries containing pieces of code that is used by the various solvers and utilities. For each case, a new folder structure is built containing at least the three main folders: 0, constant and system. Inside the 0 folder is all the initial field conditions for variables such as pressure and velocity. These will vary depending on the chosen solver. Constant contains information about fluid and turbulence properties. In addition, there are two more folders describing the case geometry. PolyMesh describes the mesh, and triSurface holds STL files if the geometry is imported from a CAD program. The system folder determines solver and solution settings for each field. There is also settings for the output format, runtime and time step. As the simulation runs, multiple time dump folders are created inside the case folder containing field solutions for fixed iteration intervals.

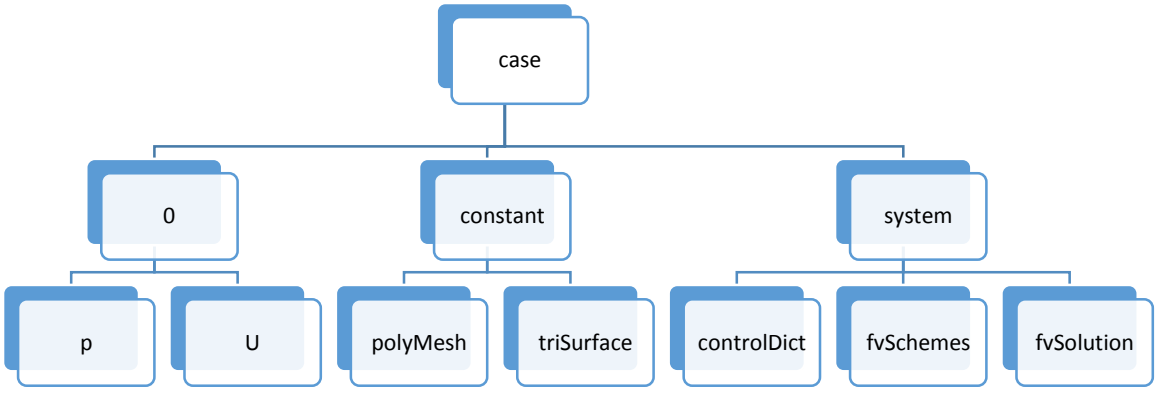


Figure 3: case structure OpenFOAM

3.2 Mesh

Mesh generation is an important part of OpenFOAM. Solution convergence and accuracy has a direct correlation to the mesh quality. OpenFOAM has its own built in mesh generation utility called blockMesh. BlockMesh is text based and works by defining hexahedral blocks in a local, right-handed coordinate system. By using these blocks, one can build more complex geometries. After running the blockMesh command, the mesh is built and geometry data is stored in the polyMesh folder. One challenge with the text-based construction of mesh is keeping track of the multiple coordinate points within a complex geometry. This can be negated by using a CAD software to produce the geometry described with STL surfaces. OpenFOAM can read these STL files and convert them in to mesh. This solution has been chosen as the strategy for this thesis.

3.2.1 Salome

The chosen CAD program is the open source software Salome. Salome is essentially a software for pre - and post-processing numerical solutions, but has a geometry feature suitable for exporting STL surfaces to OpenFOAM. The software has a graphical interface, and compared with the text-based blockMesh, makes it more comprehendible to work with. Salome is a simple CAD software with basic boolean operations such as cut and fuse. It allows the user to define vertices based on coordinates. These vertices are connected with lines, which then creates faces. To create pipes the extrusion feature is used. By defining a base surface, in this case a disc, the surface can be extruded along a path to create the pipe geometry. After all the surfaces are joined to finish the geometry, everything is converted to a completely solid figure. This is to be sure that there are no overlapping surfaces and non-real features. With a complete solid, the various surfaces can be extracted and placed in to groups. The surfaces are distinguished in groups so that different meshing options and field properties can be applied to different surfaces later on. These groups of surfaces are finally exported to OpenFOAM as STL files, and referred to in snappyHexMesh as patches.

3.2.2 SnappyHexMesh

With the geometry stored as STL files, the blockMesh utility is used to define initial cell size of the mesh. This is done by creating a cube that extends the outer edges of the STL geometry and divided in to the desirable mesh size. A finer mesh results in more cells and longer computational time, but a more accurate solution. With blockMesh, it is ensured that a suitable background mesh is generated. However, to create a mesh that traces the outer edges of the STL surfaces, the snappyHexMesh utility is needed. SnappyHexMesh identifies the outer surfaces of the geometry from the STL files, and then starts a process of splitting the cells located around the boundaries. This results in a rough representation of the surfaces and makes it possible to distinguish between the cells needed to construct the geometry, and the ones that will be removed. Once the splitting process produce a closed boundary, the remaining cells are deleted. The next step is to identify the cells intersecting the boundary, and relocate the cell vertexes onto the geometry surface. This is called the "snapping" feature and deforms the cell surface to match the shape of the geometry. Thus creating a smooth boundary and an accurate meshed representation of the STL figure.
As stated at the beginning of this chapter, the quality of the mesh is of great importance to the OpenFOAM solution. There are several quality parameters to be fulfilled before the mesh is considered ready for simulation. A few of them are values such as minimum cell volume, face skewness and concave. These can be evaluated by the checkMesh command straight after mesh generation. However, there are also restrains regarding the mesh refinement when it comes to the flow pattern one wishes to simulate. High-speed flow needs a finer mesh to create an accurate solution. The same goes for the non-dimensional distance yPlus when simulating with turbulence models. YPlus determines how coarse or fine a mesh is close to the domain walls, with respect to getting a correct representation of the flow field in this area. The two latter parameters can only be calculated once the simulation has been tested. To be able to control the mesh generation process, and fulfil these restraints, snappyHexMesh has many parameters that needs adjustment to obtain the desired result. The most important ones to this thesis will be discussed.

```
geometry // Load in STL files here
{
      inlet.stl {type triSurfaceMesh; name inlet;}
      outlet.stl {type triSurfaceMesh; name outlet;}
      walls.stl {type triSurfaceMesh; name walls;}
      refinementBox {type searchableBox; min (-0.1 -0.1 -0.05);
max ( 0.1 0.1 0.05);}
};
```

Figure 4: geometry section

First, all the relevant STL files needs to be inserted to the geometry section and named. There is also an option to create a new refinement reference geometry directly in snappyHexMesh as seen on the last line of *figure 4*.

```
// Surface based refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~

refinementSurfaces // Surface-wise min and max refinement
level
{
 inlet {level (0 0);}
  outlet {level (0 0);}
 walls {level (3 3);}
}
```

Figure 5: refinement surfaces

Then the surface refinement level is specified. Both the minimum and maximum refinement is selected, and each level represent an order of cell splitting. This means that level (2 2) refinement split all the cells twice for the given surface. Here the previously defined surface groups becomes important. As mentioned, it is not desirable to have more cells than necessary as it increases computational time. Therefore, it is useful to be able to refine only certain patches and not the entire geometry.

The second main section is internal cell refinement. Here one defines the level of cell splitting for given regions of the mesh.

```
refinementRegions        // In descending levels of fine-ness
      {walls {mode distance; levels ((0.02 3) (0.04 2));}}
      locationInMesh (0 0 0.2);
      allowFreeStandingZoneFaces true;
}
```

Figure 6: refinement regions

As before, the already defined group of cells are selected and given a level of refinement. It is also required to specify the mode that will be used to refine. In this example, the mode distance is selected. This implies that the given level of refinement will be applied to a given distance from the surface boundary. So for levels (0.02 3), all the cells within a distance 0,02m from the surface boundary are split with a level 3 of refinement. If the distance mode is selected it is also required to specify the locationInMesh entry. Here one selects whether the internal or external cells, with respect to the boundary surfaces, are to be kept.

In addition to the refinement levels, the different quality settings may affect the mesh outcome. For this thesis, the majority of these settings are kept at default values. However, there is always the risk that snapping process may be hindered by these and not completely snap to the surface geometry. Alternatively, snappyHexMesh simply do not recognize the geometry surfaces due to poorly constructed STL files. Both these cases can be solved to some degree by further increasing the surface refinement. However, if this is not desirable, there is an additional tool in snappyHexMesh to help improve the snapping feature.

To use the surface feature in snappyHexMesh, first include the surfaceFeatureExtractDict in the system folder. Then edit the text file to apply to the desired STL files as shown in *figure 7*, below. This allows the user to run the surfaceFeatureExtract command in the terminal window and produce a set of e.mesh files. These files describe the outer edges of the geometry and improve the snapping feature in snappyHexMesh.

```
walls.stl
{
    extractionMethod    extractFromSurface; // extractFromFile
or extractFromSurface
    extractFromSurfaceCoeffs
    {includedAngle   150;}
    writeObj                yes;    // Write options
}
```

Figure 7: surface feature extract

The e.mesh files are listed under features in snappyHexMesh. One thing to note with this feature is that geometrical shapes such as pipes are poorly converted in to e.mesh files. This is due to the lack of sharp edges along the pipe. The e.mesh file will only contain information about the inlet and outlet edges, and nothing about the potential bends along the pipe length.

```
// Explicit feature edge refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

features // taken from STL from each .eMesh file created
by "SurfaceFeatureExtract" command
(
    {file "walls.eMesh"; level 3;}

);
```

Figure 8: surface features

Last cell refinement step in snappyHexMesh is the add layers control. This feature is designed to expand an outer layer of cells that will follow the contour of the surface boundary.

```
// Settings for the layer addition.
addLayersControls
{
    relativeSizes false; // was true
    layers
    {
     walls
           {nSurfaceLayers 3;} // was 3
    }

    expansionRatio 1.3;
    finalLayerThickness 0.00016; //was 0.00016
    minThickness 0.00008; //was 0.00008
    nGrow 0; // was 1
```

Figure 9: add layers

Here it is required to define for which patch this feature should be applied, how many layers should be expanded and thickness of these layers. If the relative sizes option is turned on then the thickness is based on the size of the current outer cell size. Creating this outer layer can be challenging for complex geometry and lead to ill-defined cells. It is therefore recommended to leave this feature off, until a satisfactory mesh has been obtained by the previous refinement steps. Then run the snappyHexMesh again, leaving only the add layers feature on, and tweak the settings to expand a complete outer layer of cells.

3.3 rhoSimpleFoam

RhoSimpleFoam is a steady-state solver used for simulating turbulent RANS flow of compressible solvers. This solver allows for simulating the mass flow through the system, as well as the temperature dissipation to the outer environment. The chosen solver is as mentioned steady state, as the important data for this case is obtained once the flow has stabilized. The time-period right after fan startup is not relevant to the case solution. When running a steady state solver in OpenFOAM, the time steps does not represent the elapsed time, but rather as an iteration counter. Furthermore, the time dumps created during the simulation does not contain usable data. Only the last converged results can be used for solution analysis. When running the rhoSimpleFoam solver there are a number of solvers and numerical schemes located in the system folder. These contain settings for how the equations are to be solved.

3.3.1 fvSchemes

Within the fvSchemes text file there are options to assign which numerical schemes used for the terms to be solved. First, the time derivate scheme will be specified as steady state as it is not applicable for this case. Then the gradient schemes are all assigned Gauss linear as method of discretization of the divergence. By specifying linear after the Gauss theorem selection, the chosen interpolation scheme is set as central differencing. Next is the convection scheme, identified under divSchemes. Here the bounded Gauss is used, but the interpolation method is upwind for all values except for one. The exception is a part of the momentum equation, div((muEff*dev2(T(grad(U))))), which only works with Gauss linear. The upwind differencing is the most stable interpolation method available in OpenFOAM. From here on, all the schemes are specified using a default value, which means that all the terms will be assigned identical settings. The laplacian schemes are solved with Gauss linear corrected. Corrected is an explicit non-orthogonal correction. Lastly, the default interpolation schemes are linear, and the surface normal gradients set at corrected. The surface normal gradients are used to compute the gradients at cell faces.

3.3.2 fvSolution

In fvSolution, the settings specify how to solve the equations based on matrix inversions. Often the equations to be solved in OpenFOAM result in large matrices. These matrices are however mostly built by zero entries. Therefore, the traditional algebraic techniques become inefficient and iterative methods are adopted instead. There are three types of solvers to invert matrices in OpenFOAM. The first one is preconditioned (bi-) conjugate gradient, PCG/PBiCG, which distinguishes between symmetric and asymmetric matrices. The second is geometric-agglomerated algebraic multigrid, CAMG. CAMG requires a positive definite, diagonally dominant matrix to operate. Lastly, there is the smoothSolver, which operates for both symmetric and asymmetric matrices. The two last solvers has been chosen for this case, as it is recommended, Hjertager (2009A), to use the CAMG solver for pressure and smoothSolver for the remaining variables. As for smoothers, the Gauss Seidel and symmetric Gauss Seidel is recommended. Apart from the solvers and smoothers, there is also settings dedicated to the solver accuracy. Tolerance refers to how exact the solution is based on the initial residuals. The relative tolerance specifies how accurate the solution is solved for each iteration step.

3.3.3 Boundary conditions

In OpenFOAM, it is necessary to specify the initial field values and boundary conditions. All the values for these fields are stored in the 0 folder as text files. From earlier in the mesh generation process it is important to remember which boundary entries are defined as a patch type, and which are defined as a wall type. The equations used to describe the boundary conditions have been gathered from CFD-Wiki (2009, 2012A, 2012B & 2014) and OpenFOAM foundation (2011).

3.3.3.1 Alphat

Alphat describes the turbulent thermal diffusivity. The turbulent heat transfer is calculated using the equation:

$$\alpha_t = \frac{\mu_t}{Pr_t} \qquad \textit{Equation 20}$$

Here, α_t is turbulent thermal diffusivity. μ_t is mut, the turbulent viscosity. Lastly, Pr_t is turbulent Prandtl number with a default value of 0,85. The wall entries are defined with the boundary condition compressible::aplhatWallFunction, and patch entries are set to calculated.

3.3.3.2 Epsilon

The epsilon field allows for describing the turbulence dissipation rate at a boundary inlet and walls. Epsilon is calculated by the formula:

$$\varepsilon = \frac{C_\mu^{0,75} k^{1,5}}{l} \qquad \textit{Equation 21}$$

C_μ is a model constant with value of 0,09 and *l* is the mixing length. Mixing length is calculated using the formula:

$$l = 0{,}038 d_h \qquad \textit{Equation 22}$$

Where the value d_h, is the hydraulic diameter. For a circular inlet, such as a pipe, the hydraulic diameter equals the pipe diameter. However, with a rectangular duct the hydraulic diameter is calculated from:

$$d_h = 2\frac{ab}{a+b} \qquad \textit{Equation 23}$$

Here the value *a* is the duct width, and *b* is the duct height. The wall entries are defined with compressible::epsilonWallFunction and inlet patches have type compressible::turbulentMixingLengthDissipationRateInlet. Note that the outlet patch is given the inletOutlet boundary condition. This fixes the outlet field to a given inletValue to prohibit instability in case of inward flow during simulation. In addition, the Von Karman constant kappa and model coefficient *E* is defined with default value 0,41 and 9.8.

3.3.3.3 K

Inside the k file, there is the boundary conditions for the turbulence energy. The value of turbulent kinetic energy *k*, can be calculated from the equation:

$$k = \frac{3}{2}(UI)^2 \qquad \textit{Equation 24}$$

U is the mean flow velocity and *I* is the turbulence intensity calculated by:

$$I = 0{,}16 \cdot Re_{d_h}^{-0{,}125} \qquad \textit{Equation 25}$$

Here, Re_{d_h} is the Reynolds number based on the hydraulic diameter. The following equation is used to estimate Reynolds:

$$Re_{d_h} = \frac{U \cdot d_h}{\nu} \qquad \textit{Equation 26}$$

The hydraulic diameter is the same as used for the epsilon value above. Only new value is the kinematic viscosity ν. Inlet patches are given the boundary compressible::turbulentMixingLengthDissipationRateInlet and the outlet patch, inletOutlet. Walls are defined by compressible::epsilonWallFunction.

3.3.3.4 Mut

Mut is the turbulent kinematic viscosity and only needs to be defined at wall patches with mutkWallFunction. The remaining patches are calculated.

3.3.3.5 T

Temperature is fixed at the inlet patches and with inletOutlet on the outlet patch. The wall patch named pipe is applied with wallHeatTransfer. This boundary condition makes it possible to simulate heat loss to the surrounding environment. The alphaWall value is calculated in the heat transfer chapter 2.5, using *equations 17* to *19*.

3.3.3.6 Pressure

To simulate the static pressure given by a fan, the inlet is given a fanPressure boundary. This boundary condition allows the user to enter a fan performance curve, and let OpenFOAM calculate the stabilized relation between pressure and volumetric flow. The fan curve file is located inside the constant folder, and contains a simple table of corresponding values for the system pressure drop and volumetric flow at inlet. In addition to the fan curve, it is necessary to define initial pressure value, environmental total pressure and direction of flow with respect to the boundary. The outlet is defined with fixed value, while the remaining patches are set with zeroGradient.

4 DESIGN

The main parts of the cooling system consists of a fan, driving the exhaust flow, a series of pipes and the existing air intake ducting. There were two proposed designs to the new piping. One where the pipe enters the duct at the top of the duct, and one where it enters by the side. The original proposal did include some measurements to describe the overall shape of the designs. However, some had to be added as seemed fitted by a lack of information. Even though the geometry alters between the two cases, the name of the surface groups exported from Salome, remain the same. The boundary where the exhaust enters through the fan is named fan inlet. Similarly, where the outside air is drawn into the system is called air inlet. The two flows converge and exist the system through the outlet.

4.1 Top entry

The first design is shown in *figure 10* below. The piping is marked in red and has a diameter of one meter. There has been incorporated a series of bends to avoid a cooling tower situated between the fan inlet and the ducting. Bend number 3 is of 45 degrees while the rest are 90 degrees bends. Number 4 and 5 combine to create an s-bend. The existing ducting is displayed in blue. Marked in green is the air inlet and it is a direct opening to the surrounding atmosphere.

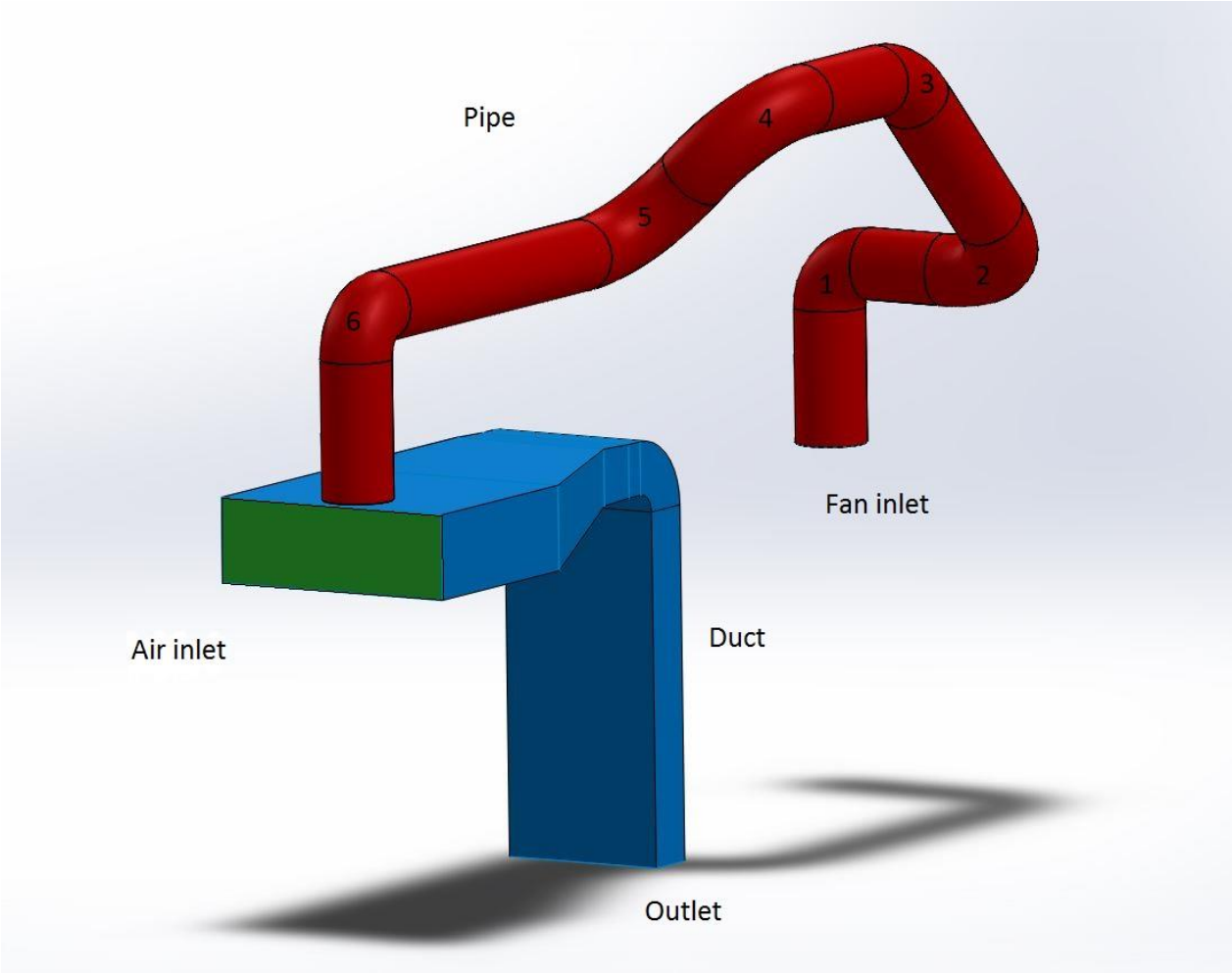


Figure 10: top entry design

4.1 Side entry

Second design is described in *figure 11* and it proposes a pipe side entry to the ducting. Bend number 1, 2 and 5 is of 90 degrees. While bend 3 and 4 is of 45 degrees. As before the air inlet is marked in green and the ducting marked in blue. The exact location of the cooling tower is not known in detail, and therefore only the two proposed project designs will be explored in this thesis. A full overview of both system designs with dimensions can be found in *appendix A*.

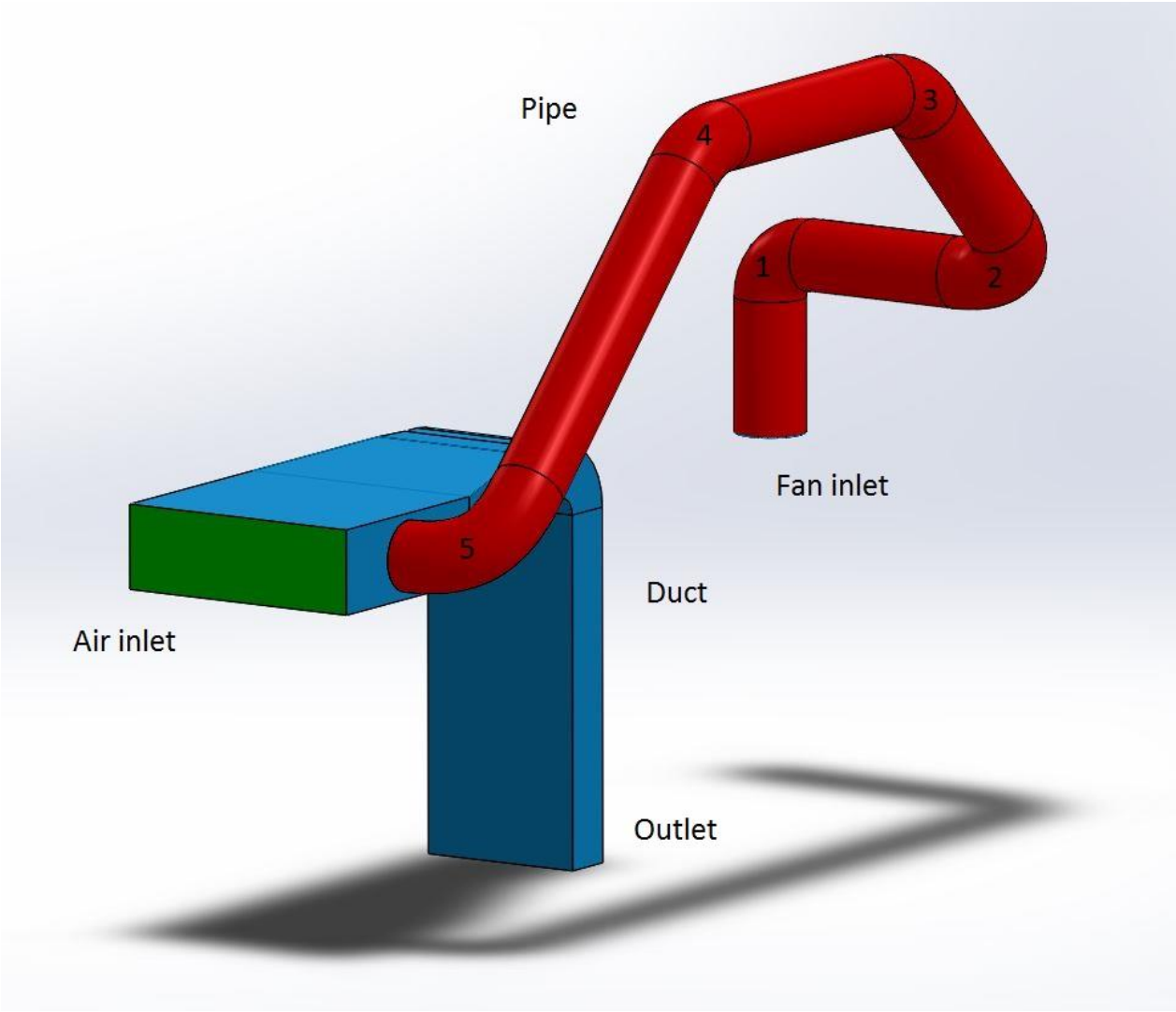


Figure 11: side entry design

5 PRE-PROCESSING

5.1 Mesh generation

The meshing process involves a lot of trial and error. The goal is to create a mesh fine enough to accurately represent the STL geometry, but not create a mesh with an unnecessary high number of cells. With the blockMesh utility, the initial cell size is set to a square with side length of 0,1m. From there the snappyHexMesh surface refinement is found to be adequate at only refinement level (1 1). To create a smooth transition between the boundary and internal mesh, a level 2 distance-based refinement is used. These settings seem to work well to represent the bends of the pipe, but the mesh would not properly snap to the sharp diagonal edges of the duct geometry. As seen in *figure 12*.

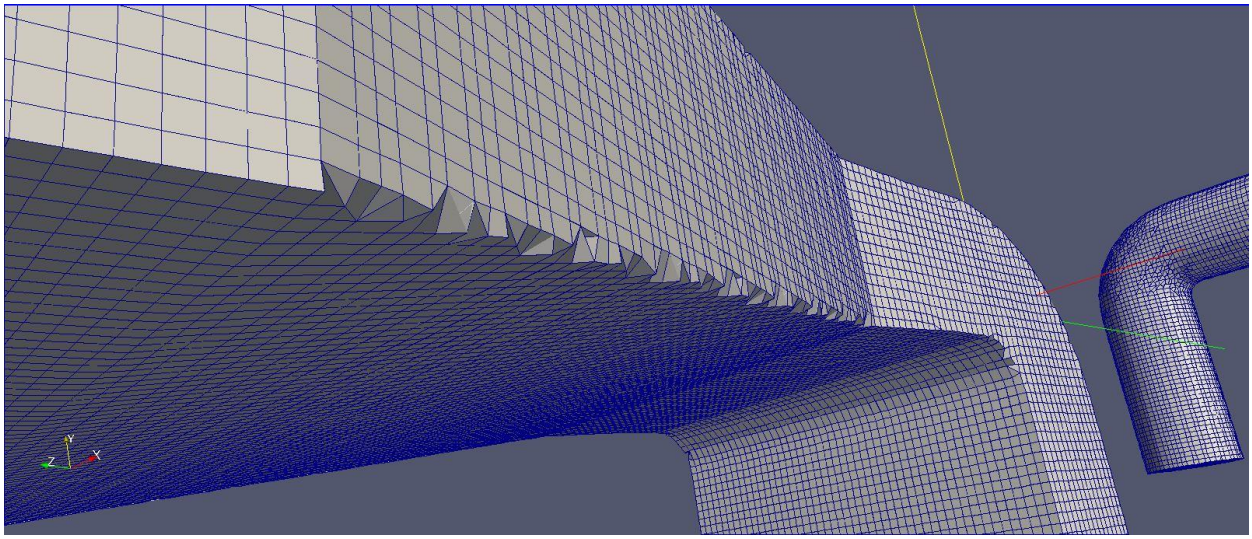


Figure 12: inaccurate representation of edges

The problem was however solved, by including the surfaceFeatureExtract feature. This was done by extracting an additional surface group including the duct surfaces, as well as the surfaces of air inlet and outlet. The e.mesh file was listed in snappyHexMeshDict with refinement level 2, and the resulting mesh was much more accurate.

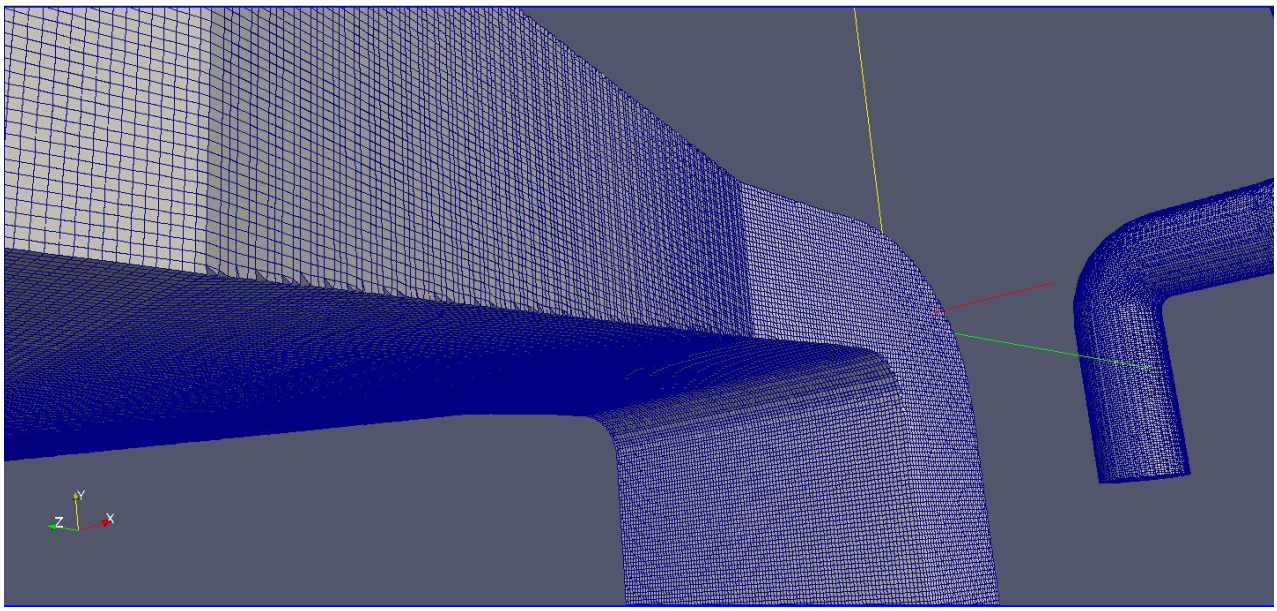


Figure 13: accurate representation of edges

After the base mesh was established, three surface layers were added. These surface layers are to extend the outer edges of the entire geometry. The layers add another level of refinement and ensure accurate solutions near the boundaries. However, creating the outer layers can be challenging, and often need very exact settings to form correctly. For both cases, the problem area is located in between the pipe and duct transition. As shown in *figure 14*, the layers collapse in on themselves around the tight 90 degrees angle.

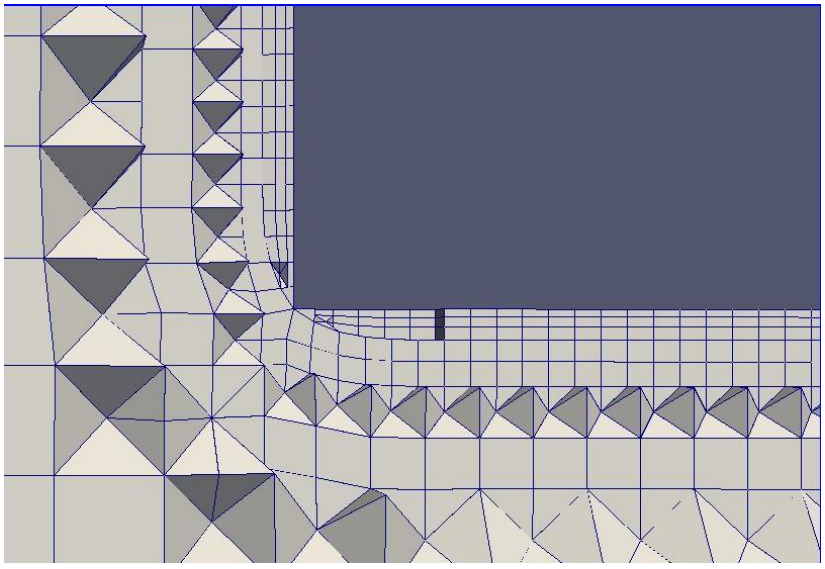


Figure 14: collapsed layer

By altering the settings for the addLayers feature, it was found that increasing the value for featureAngle the layers would form correctly. The featureAngle decides at what angle, between two existing cells, new layers should be extruded. In *figure 15*, the final internal mesh is depicted.

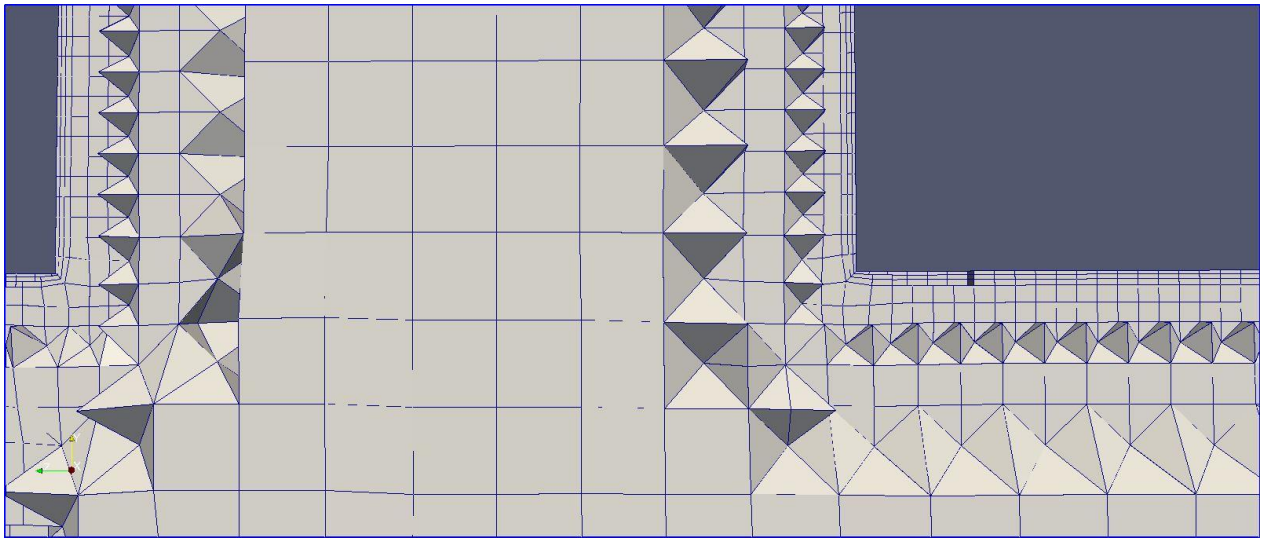


Figure 15: fully formed outer layer

The complete mesh result in a size of 1486140 cells for the top entry design, and a mesh size of 1765704 cells for the side entry case.

5.2 Initial field values

The initial field values are not affected by the two different design cases. Dimensions at inlets and outlet remain the same, and pipe diameter is unchanged. However, the change in outside temperature will have an impact on some of the initial conditions.

5.2.1 Velocity

The fan inlet velocity is not defined in boundary conditions, as the flow is pressure driven and governed by the fan curve. From initial simulations, the flow velocity at fan inlet is measured at approximately 20m/s. This velocity value will be the basis for further calculations at the fan inlet. At the air inlet, the mass flow is fixed at 2kg/s. Given the air density and area of the inlet:

$$U_{air} = \frac{\dot{m}}{\rho \cdot A} = \frac{2\frac{kg}{s}}{1{,}292\frac{kg}{m^3} \cdot 4{,}32m^3} = 0{,}36\ m/s$$

5.2.2 Pressure

The initial atmospheric pressure is set to 101325Pa. Pressure is fixed at the outlet, and adjusted by the fan curve at fan inlet.

Table 6: fan curve

Volumetric flow $[m^3/s]$	Static pressure [Pa]
14	1800
15	1600
16	1400
17	1200
18	950
19	700
20	400

5.2.3 Temperature

Cases are to be simulated for outside temperatures at both 263,15K and 273,15K. The exhaust entering the system through the fan inlet is fixed at 333,15K. In addition to the temperatures at the inlets and outlet, the heat loss through the pipe wall needs to be accounted for. To calculate this, one first needs to find the Nusselt number, and Reynolds number for the outside wind shear. In this thesis, the crosswind speed is set to 10m/s.

$$Re_{273} = \frac{U_{Wind} \cdot D}{\upsilon_{273K}} = \frac{10\,\frac{m}{s} \cdot 1m}{1{,}338 \cdot 10^{-5} m^2/s} = 7{,}474 \cdot 10^5$$

Using the formula constants defined in section 2.8.2, the Nusselt number is calculated with *equation 17*:

$$Nu_{273} = 0{,}076 \cdot (7{,}474 \cdot 10^5)^{0{,}7} \cdot 0{,}7362^{0{,}37} \cdot \left(\frac{0{,}7362}{0{,}7202}\right)^{\frac{1}{4}} = 882{,}013$$

Now the convection heat coefficient is calculated from *equation 18*:

$$h_{air,273} = 882{,}013 \cdot \frac{0{,}02364\,\frac{W}{m} \cdot K}{1\,m} = 20{,}85\ W/m^2 \cdot K$$

Equation 19 combines the pipe wall thickness and conductivity, thus the total heat transfer coefficient is found:

$$\frac{1}{h_{total,273}} = \frac{1}{20{,}85\ W/m^2 \cdot K} + \frac{0{,}003m}{13{,}4\ W/m \cdot K}$$

$$h_{total,273} = 20{,}75\ W/m^2 \cdot K$$

Heat loss for the second case, with outside temperature at 263,15K, is solved in similar fashion:

$$Re_{263} = \frac{U_{Wind} \cdot D}{\upsilon_{263K}} = \frac{10\,\frac{m}{s} \cdot 1m}{1{,}252 \cdot 10^{-5} m^2/s} = 7{,}987 \cdot 10^5$$

$$Nu_{263} = 0{,}076 \cdot (7{,}987 \cdot 10^5)^{0{,}7} \cdot 0{,}7387^{0{,}37} \cdot \left(\frac{0{,}7387}{0{,}7202}\right)^{\frac{1}{4}} = 925{,}911$$

$$h_{air,263} = 925{,}911 \cdot \frac{0{,}02288\,\frac{W}{m} \cdot K}{1\,m} = 21{,}19\ W/m^2 \cdot K$$

$$\frac{1}{h_{total,263}} = \frac{1}{21{,}19\ W/m^2 \cdot K} + \frac{0{,}003m}{13{,}4\ W/m \cdot K}$$

$$h_{total,273} = 21{,}09\ W/m^2 \cdot K$$

The total heat transfer coefficient will be used as the alphaWall value for the wallHeatTransfer boundary condition.

5.2.4 Turbulence energy

To calculate turbulence energy and intensity, one first need to define the hydraulic Reynolds number. The Reynolds number will vary for the two inlets, as the shape of the inlets differs. When estimating the Reynolds number for a cylindrical pipe the hydraulic diameter equals the pipe diameter, hence the Reynolds number for fan inlet is calculated with *equation 26*:

$$Re_{d_{h,fan}} = \frac{20m/s \cdot 1m}{1{,}896 \cdot 10^{-5} m^2/s} = 1{,}055 \cdot 10^6$$

The air inlet has a rectangular shape, and therefore *equation 23* is used to find the hydraulic diameter:

$$d_{h,air} = 2\frac{3{,}6m \cdot 1.2m}{3{,}6m + 1{,}2m} = 1{,}8m$$

This gives the Reynolds number:

$$Re_{d_{h,air}} = \frac{0{,}36\ m/s \cdot 1{,}8m}{1{,}338 \cdot 10^{-5}\ m^2/s} = 4{,}843 \cdot 10^4$$

Given the Reynolds number, the turbulence intensity is now calculated from *equation 25*:

$$I_{fan} = 0{,}16 \cdot 1{,}055 \cdot 10^{6^{-0{,}125}} = 0{,}028$$

$$I_{air} = 0{,}16 \cdot 4{,}843 \cdot 10^{4^{-0{,}125}} = 0{,}042$$

Finally, *equation 24* yields the two turbulence energy values:

$$k_{fan} = \frac{3}{2}(20m/s \cdot 0{,}028)^2 = 0{,}470\ m^2/s^2$$

$$k_{air} = \frac{3}{2}(0{,}36m/s \cdot 0{,}042)^2 = 0{,}0003\ m^2/s^2$$

5.2.5 Turbulence dissipation

The turbulence dissipation rate is estimated from the turbulence mixing length. Again, the mixing length is calculated from the hydraulic diameter, which has been defined in the above turbulence energy section 5.2.4. This means that there will be two different values for the two inlets.
Entering the hydraulic diameter into *equation 22*, yields the two mixing lengths:

$$l_{fan} = 0{,}038 d_h = 0{,}038 \cdot 1m = 0{,}038m$$

$$l_{air} = 0{,}038 d_h = 0{,}038 \cdot 1{,}8m = 0{,}068m$$

Now the two dissipation rates are calculated with *equation 21*:

$$\varepsilon_{fan} = \frac{C_\mu^{0,75} k^{1,5}}{l} = \frac{0{,}09^{0,75} \cdot (0{,}470 m^2/s^2)^{1,5}}{0{,}038m} = 1{,}39 m^2/s^3$$

$$\varepsilon_{air} = \frac{C_\mu^{0,75} k^{1,5}}{l} = \frac{0{,}09^{0,75} \cdot (0{,}0003 m^2/s^2)^{1,5}}{0{,}038m} = 0{,}00002 m^2/s^3$$

6 SOLVING

Inside the controlDict text file there are entries regarding the simulation settings. Settings that control for what time step to start and stop the simulation, as well as which time step to write the temporary results, also known as time dumps. Since this is a steady state solver, the time step simply refers to number of iterations instead of seconds of simulation time. Only once the solution has converged, can the data be used for analysis. The previous intermediate data is not accurate. As the simulation starts, OpenFOAM produces information regarding the solving of the different equations. By monitoring this information, one can deduce whether or not the solution has converged. The first piece of information is the initial residual. The initial residual should approach zero when the solution convergence. Second, the final residual should always remain small and always be less than the initial residual. This value describes the residual of the current equation being solved. Lastly, the number of iterations show how many times the matrix was solved for the current equation. This value should also start to decrease once the solution approach convergence. Monitoring the residuals as they move over the screen may in some cases prove difficult and time consuming. To help with this task the third party program PyFoam is utilized.

6.1 PyFoam

PyFoam allows for real time plotting of residuals, Gschaider (2015A). Once installed the plotting utility starts when the OpenFOAM solver is initialized. The PyFoam plotter makes it a lot easier to keep track of the information displayed by the OpenFOAM solver and help notice trends as they appear. The graphs in PyFoam should ideally flatten out and become straight lines once the solution is fully converged.

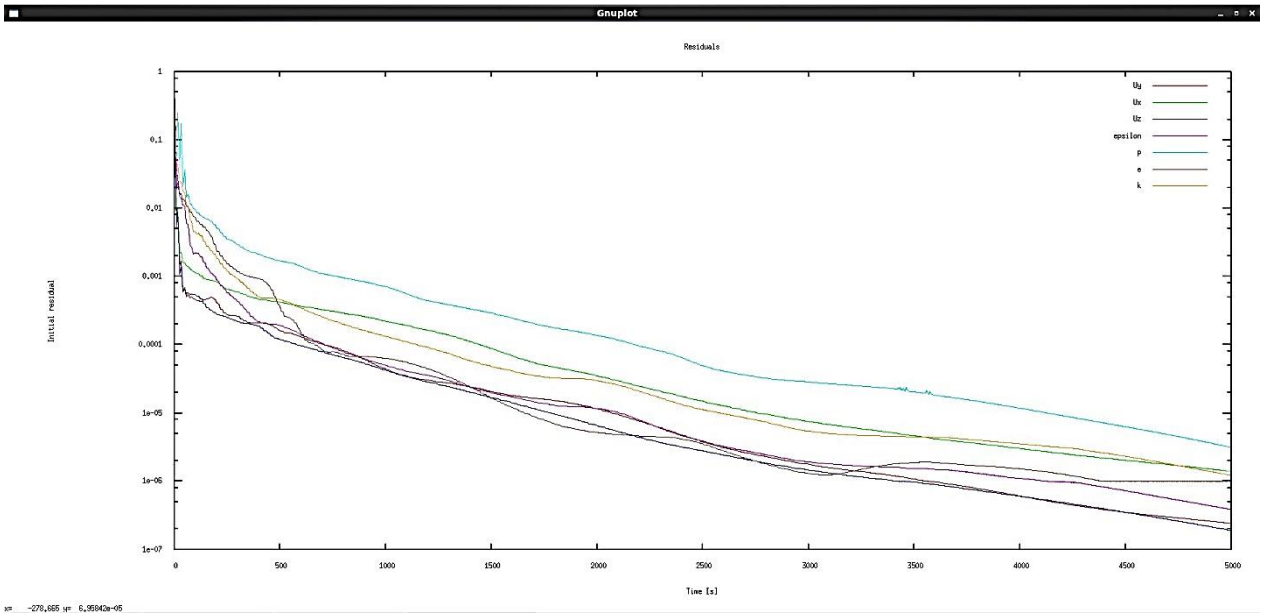


Figure 16: example of the PyFoam plotter

6.2 Parallel processing

Even though best efforts where used to try and keep the mesh size as small as possible, the final mesh requires a significant amount of computational power and time to be solved. To help speed up the solving process, one can split up the workload and manually assign it to different processing cores. This helps OpenFOAM to utilize the available processing power as efficient as possible. To accomplish this, the decomposeParDict needs to be added to the system folder. Inside the number of available processor cores are specified. After running the decomposePar command, additional folders are constructed inside the case folder, representing the number of cores. These need to be reconstructed in to one whole solution again, once the solver is completed.

6.3 Swak4Foam

Swak4Foam is an OpenFOAM library with many additional utilities. The library is installed separately as it is not included in standard OpenFOAM software. Swak4Foam combines the functionality of groovyBC and funkySetFields. This allows the user to create expressions and boundary conditions based on field values. With the ability to create such expressions, the

swak4Foam can be tailored to fit most situations. Hence why swak4Foam stands for SWiss Army Knife for Foam, Gschaider (2015B). The option of swak4Foam was explored in this thesis, with regards to create an expression to control the total mass flow entering the system over both inlets. This was never implemented for reasons that will be discussed in detail later on. However, a secondary function of swak4Foam was included in the final simulation. Function objects can carry out a variety of tasks, such as data sampling and monitoring, while the OpenFOAM simulation is running. They are defined inside the controlDict file and can be customized with swak4Foam expressions. The most useful for this thesis are the utility to define the face-flow field phi, and sum over the patches. This then gives the total mass flow for each inlet and outlet, OpenFOAM Wiki (2012). In addition, expressions are created to directly calculate the pressure drop between inlets and outlet.

7 POST-PROCESSING

Once the OpenFOAM solver is complete, the field values are stored separately in text files. Inside each file, the value for each cell of the given field is listed. Analyzing these values as they are presented, without any positional data, does not offer much insight. However, OpenFOAM comes with its own post-processing software called ParaView. ParaView is an open-source software capable of performing data analysis and visualization.

7.1 ParaView

ParaView combines the field values and positional data with the mesh model to create a visual representation of the OpenFOAM simulation. All the fields included in the simulation can be inspected and colored according to magnitude. In addition, there are several tools to closer inspect the results, such as graphing tools, hiding parts of the mesh and so on.

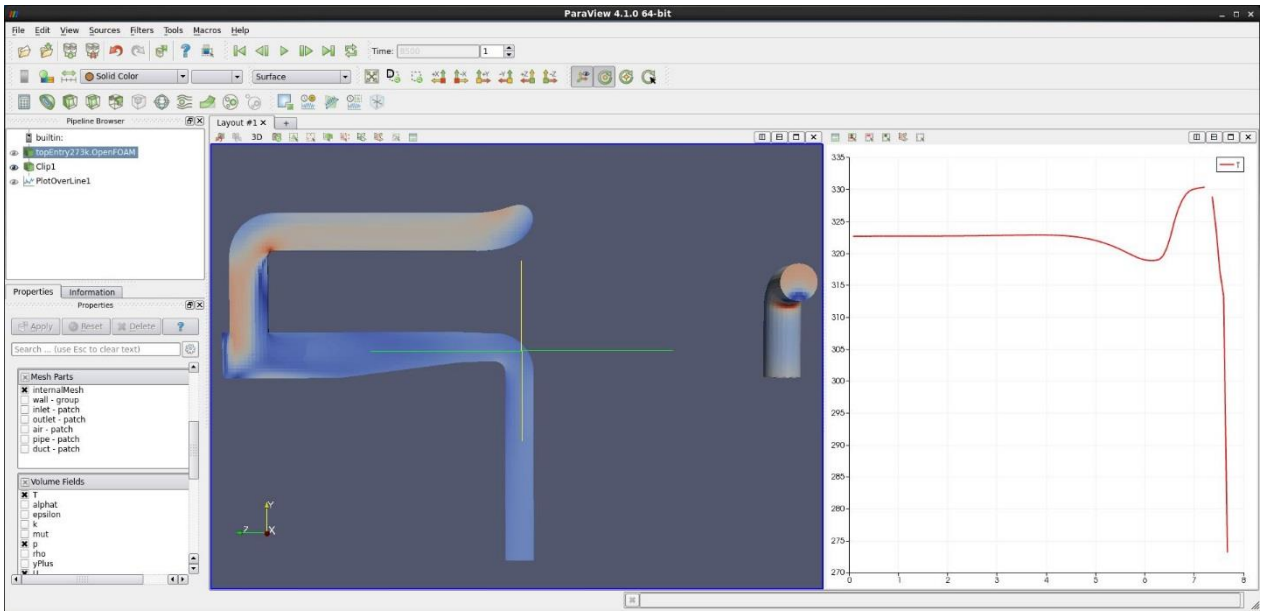


Figure 17: example of the paraView interface

7.2 SampleDict

The sample feature, in some ways serve the same purpose as the function objects. It is manly used to collect data samples from the simulation. One key difference from the function objects is that the sample feature can collect the data after the simulation process has ended. Usually the data collected are used to analyze the obtained results. One alternative use of sampleDict however, is to collect all of the relevant field values for an outlet patch, and then use these values as the starting conditions for the inlet patch of a different case. This is useful in situations where the mesh becomes too big for the computer to handle. More specifically, OpenFOAM runs out of available RAM. Then the mesh needs to be split in two halves and simulated separately. To accomplish this, a new folder structure named boundaryData/inlet/0 must be created inside the constant directory of the new case. The sampled faceCentres file is renamed to points, and placed inside the constructed boundaryData/inlet directory. Then the sampled fields are transferred to the boundaryData/inlet/0 directory. The text files containing the field values do not have the correct file headers and these need to be constructed according to standard headers for vectors and scalars. Now the new case can be simulated based on the previous results from the old case. The boundary condition for the inlet patch in the new case is defined with timeVaryingMappedFixedValue. This method was researched but not implemented in this thesis, as the mesh size was kept at a manageable size.

7.3 topoSetDict

The feature topoSet operates using boolean operations to create new cell or face sets within the existing mesh. These cells and faces can then be defined into specific zones, faces and patches. The topoSet will be used in this thesis to supplement the sample feature. To extract temperature values from the cases, the sample feature defines a plane from coordinates. Cells that then intersects with the plane will be extracted. The need for the topoSet feature arises when the mesh geometry intersects the plane at multiple locations. This will lead to incorrect data where unwanted cell values are included in the sample. By defining zones within the mesh, the topoSet can restrict the plane sample within the desired boundaries.

7.4 YPlus

As mentioned earlier, yPlus is a non-dimensional wall distance value. To ensure that the mesh cell size is fine enough near the wall boundary, it is recommended to keep the yPlus value below within the range of 0 to 300. The yPlus value can be evaluated by running the yPlus command in the terminal window and opening the yPlus field values in ParaView.

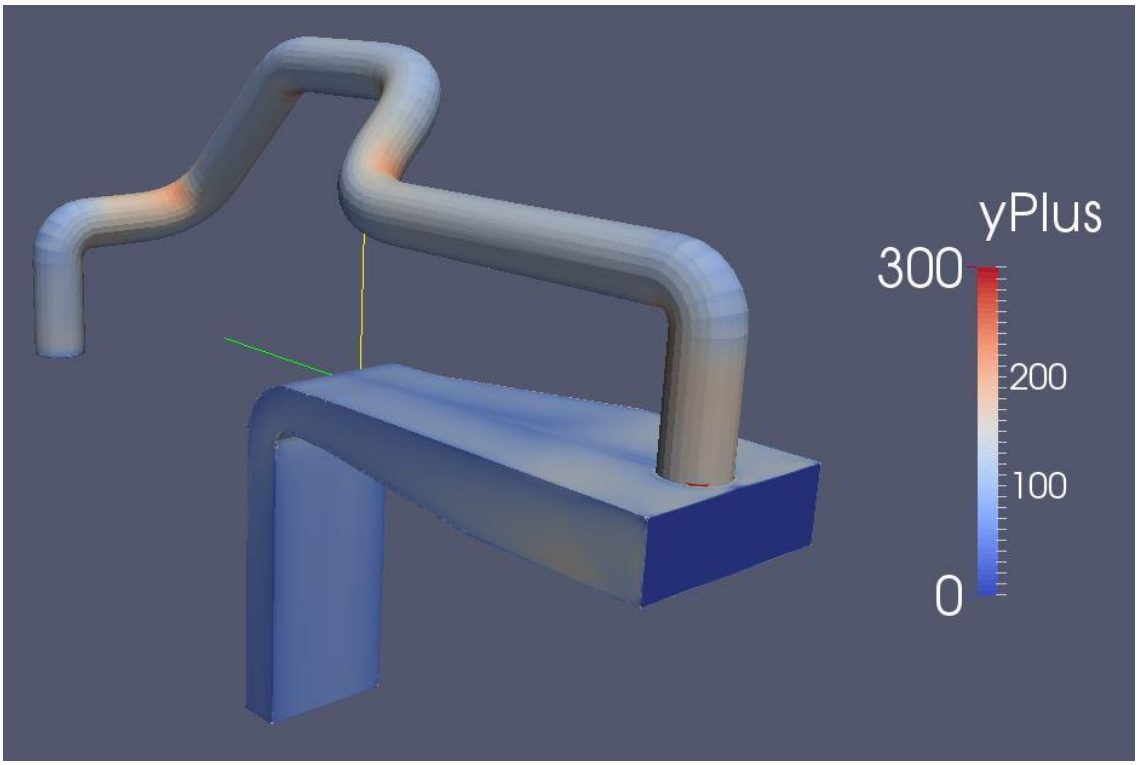


Figure 18: yPlus in ParaView

8 RESULTS AND DISCUSSION

The goal of the OpenFOAM model is to create a simulation that mimics real life conditions. This then becomes a test bench for the fan in question, to see whether it can fulfill the required criteria. In the following chapter, the results from the simulation will be presented along with estimations and calculations.

8.1 Temperature

Temperature is the main deciding factor for the simulations. If all other variables are found to be reliable, then the temperature readings will show whether the design cases are suitable for turbine operation. First to be evaluated is the top entry design case. The temperature is sampled at the inlets and outlet, as well as at cross sections along the geometry. Sample locations are shown in *figure 19*. *Table 7* shows the results for outside temperature at both 273,15K and 263,15K. Even though the air entering the system experiences a temperature drop of 10 degrees, the air leaving the system only drop a maximum of 2 degrees. This then means that for the top entry case, the lowest air temperature entering the turbine is 321,2K

Table 7: temperature samples, top entry case

	Fan inlet	T1	T2	T3	T4	Air inlet	T5	T6	Outlet
Temperature [K]	333,15	330,7	329,2	328,1	327,4	273,2	323,0	323,0	323,0
	333,15	330,2	328,5	327,1	326,4	263,2	321,2	321,2	321,2

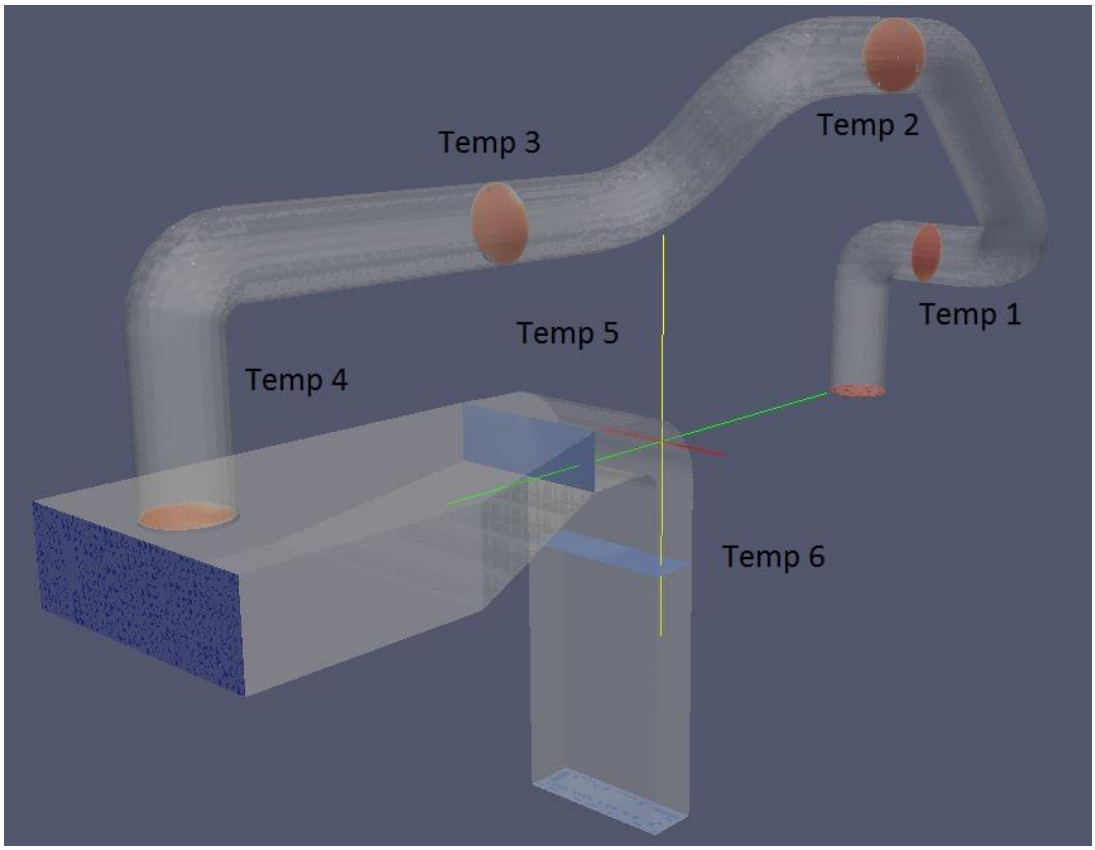


Figure 19: temperature sample, top entry case

As before, the side entry case have the temperature values sampled from the inlets, outlet and along the geometry path. Temperature readings show that the 10 degrees drop in atmospheric temperature, only results in a 1,5 degree drop in air temperature at the outlet.

Table 8: temperature data, side entry case

	Fan inlet	T1	T2	T3	Air inlet	T4	T5	Outlet
Temperature [K]	333,15	330,9	329,2	328,3	273,15	324,7	324,9	324,8
	333,15	330,5	328,5	327,5	263,15	323,2	323,4	323,3

For both cases, the air temperatures leaving the system is well above the 277,65K temperature level that ensures reasonable humidity values, according to *figure 2*.

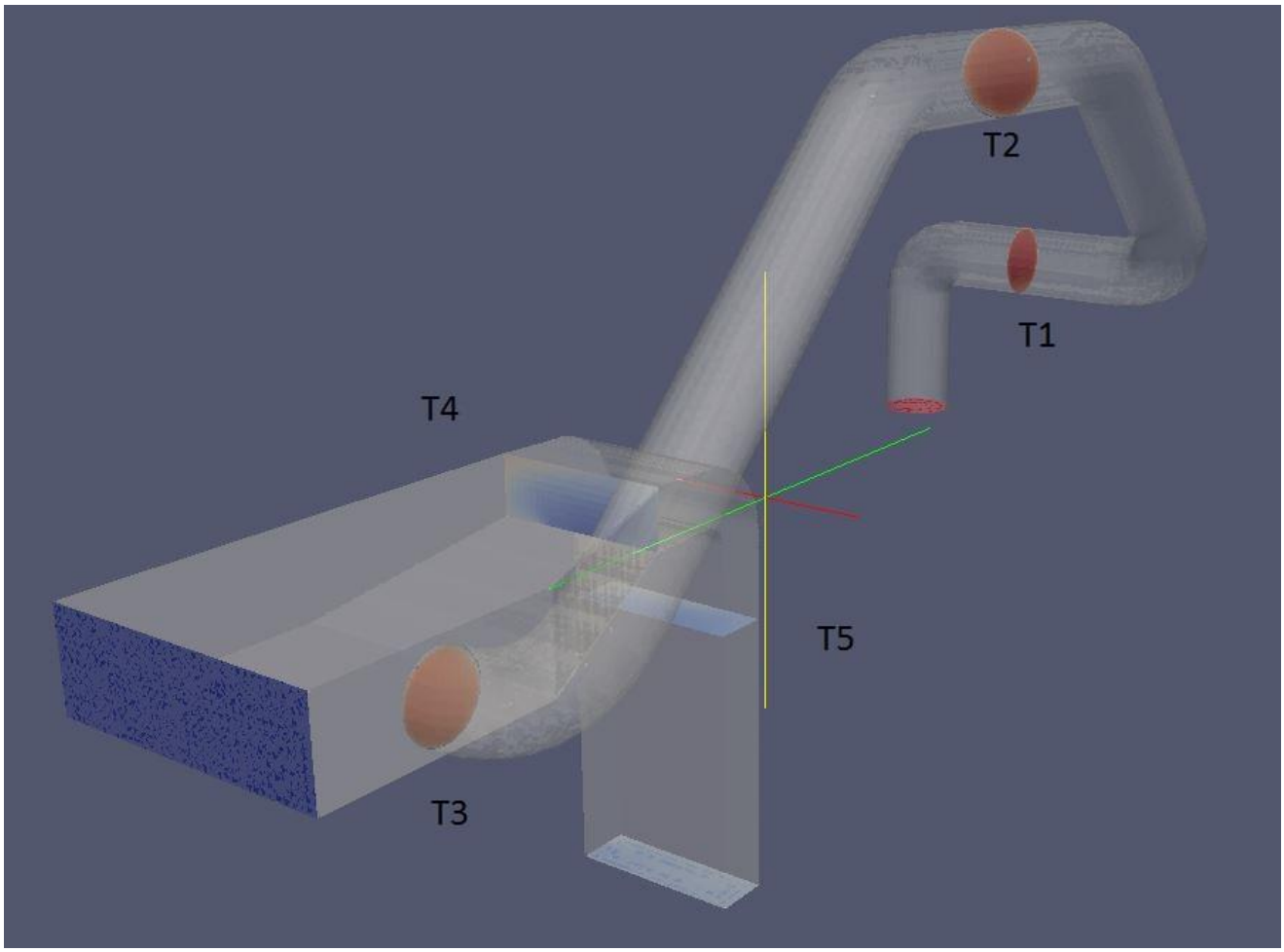


Figure 20: temperature sample, side entry case

8.2 Pressure loss

To evaluate the results given by the OpenFOAM simulation, the theoretical pressure loss between the fan inlet and air inlet is calculated. The calculated pressure loss for the two cases are then compared to the pressure loss indicated by the simulations. Theoretical pressure loss is estimated with *equation 16*. By examining the equation, one can see that the pressure loss is particularly susceptible to change in velocity. Therefore, the fan inlet velocity used in these calculations are drawn directly from the simulation to ensure an as accurate result as possible. As can be observed in *table 12* and *13*, the fan inlet velocity for the top entry case is measured at 23,1m/s and 24,4m/s for the side entry case. Using the measured velocities, and loss coefficients from *table 4*, the theoretical pressure is calculated:

$$\Delta p_{topEntry} = 1{,}255m \cdot 1{,}059kg/m^3 \cdot 9{,}81m/s^2 + \left(\frac{0{,}0235 \cdot 24{,}3m}{1m} + 1{,}94\right)\left(\frac{1{,}059kg/m^3 \cdot (23{,}1m/s)^2}{2}\right) = 722{,}5\ Pa$$

$$\Delta p_{sideEntry} = 0{,}655m \cdot 1{,}059kg/m^3 \cdot 9{,}81m/s^2 + \left(\frac{0{,}0235 \cdot 22{,}9m}{1m} + 1{,}43\right)\left(\frac{1{,}059kg/m^3 \cdot (24{,}4m/s)^2}{2}\right) = 627{,}3\ Pa$$

Comparing the two results in *table 9*, one can see that the pressure results for the top entry case deviate with about 188 Pa. Even more for the side entry case, which have a deviation of approximately 328 Pa. These results suggest that one of the two methods are faulty.

Table 9: theoretical and simulated pressure loss between fan inlet and air inlet

Top entry			Side entry		
Theoretical	Simulated		Theoretical	Simulated	
	273,15 K	263,15 K		273,15 K	263,15 K
722,5 Pa	534,3 Pa	533,8 Pa	627,3 Pa	299,8 Pa	299,3

Reevaluating *equation 16*, one can see that one other constant has a relatively large impact on the equation outcome. The loss coefficient, also known as minor losses, needs to be found in reference literature. By evaluating different literature sources, the loss coefficient seem to vary in value. Another unforeseen challenge was to find a loss coefficient suitable for the pipe diameter

used in this thesis. Most available lists, does not extend up to a pipe diameter of 1 meter. This is also true for the loss coefficient used in the theoretical pressure calculations above. As can be seen in *table 4*, the loss coefficient is only given for the maximum diameter of 0,6m. To circumvent the problem, a new reference that bases the loss coefficient on the relative value between the bend radius and pipe diameter is utilized. The theoretical pressure is calculated with revised loss coefficients from *table 5*:

$$\Delta p_{topEntry} = 1{,}255m \cdot 1{,}059kg/m^3 \cdot 9{,}81m/s^2 + \left(\frac{0{,}0235 \cdot 24{,}3m}{1m} + 1{,}19\right)\left(\frac{1{,}059kg/m^3 \cdot (23{,}1m/s)^2}{2}\right) = 510{,}6\ Pa$$

$$\Delta p_{sideEntry} = 0{,}655m \cdot 1{,}059kg/m^3 \cdot 9{,}81m/s^2 + \left(\frac{0{,}0235 \cdot 22{,}9m}{1m} + 0{,}91\right)\left(\frac{1{,}059kg/m^3 \cdot (24{,}4m/s)^2}{2}\right) = 463{,}3\ Pa$$

The new theoretical pressure calculations show similarities to the simulated results. As seen in *table 10*, the variation for the top entry case is only at 24 Pa. However, the side entry case still have a relatively large deviation of approximately 164 Pa. This leads to believe that the loss coefficients are not accurate enough for these type of calculations.

Table 10: revised theoretical and simulated pressure loss

Top entry			Side entry		
Theoretical	Simulated		Theoretical	Simulated	
	273,15 K	263,15 K		273,15 K	263,15 K
510,6 Pa	534,3 Pa	533,8 Pa	463,3 Pa	299,8 Pa	299,3 Pa

Table 11: simulated total system pressure loss

Top entry		Side entry	
273,15 K	263,15 K	273,15 K	263,15 K
651,2 Pa	649,8 Pa	381,2 Pa	380,4 Pa

8.3 Flow conditions

With the fanPressure boundary condition, the mass flow of the fan inlet is pressure driven. Static pressure of the fan stabilizes to match the total pressure drop within the system, and then deliver the corresponding volumetric flow from the fan curve. *Tables 12* and *13* show the flow readings from the two cases. Mass flow differences between the two temperature setups within each case are marginal, and the difference in mass flow between the two cases are relatively small. Both cases are able to deliver well above the required 16 kg/s airflow at the outlet.

Table 12: flow conditions top entry

Atmospheric temperature	273,15K			263,15K		
Patch	Fan inlet	Air inlet	Outlet	Fan inlet	Air inlet	Outlet
Mass flow [kg/s]	19,20	2,00	21,20	19,21	2,00	21,21
Density [kg/m^3]	1,06	1,29	1,09	1,06	1,34	1,10
Volumetric flow [m^3/s]	18,11	1,55	19,45	18,12	1,49	19,28
Velocity [m/s]	23,09	0,36	9,39	23,09	0,35	9,34

Table 13: flow conditions side entry

Atmospheric temperature	273,15			263,15		
Patch	Fan inlet	Air inlet	Outlet	Fan inlet	Air inlet	Outlet
Mass flow [kg/s]	20,17	2,00	22,17	20,17	2,00	22,17
Density [kg/m^3]	1,06	1,29	1,08	1,06	1,34	1,09
Volumetric flow [m^3/s]	19,03	1,55	20,53	19,03	1,49	20,34
Velocity [m/s]	24,37	0,36	11,30	24,37	0,35	11,25

8.4 Fan performance

By inspecting the pressure and volumetric flow results from chapter 8.2 and 8.3, the fan performance can be evaluated. The fanPressure boundary condition, coupled with the fan curve, should ensure similar fan conditions as the fan performance curve in figure 1. *Table 14* show again the fan curve table used for the OpenFOAM simulation.

Table 14: fan curve

Volumetric flow $[m^3/s]$	Static pressure [Pa]
14	1800
15	1600
16	1400
17	1200
18	950
19	700
20	400

The results of the total system pressure drop are shown in *table 11*. First result for the top entry case is a pressure drop of 650Pa. To compare the corresponding volumetric flowrate, the fan curve in *table 14* is interpolated:

$$\frac{p - p_1}{p_2 - p_1} = \frac{\dot{V} - \dot{V}_1}{\dot{V}_2 - \dot{V}_1}$$

Rearrange to solve for the volumetric flow:

$$\dot{V} = \frac{p - p_1}{p_2 - p_1}\left(\dot{V}_2 - \dot{V}_1\right) + \dot{V}_1$$

$$\dot{V} = \frac{650 - 700}{400 - 700}(20 - 19) + 19 = 19{,}17$$

Comparing the volumetric flowrate found by interpolating the fan curve, and the measured fan inlet flow rate from *table 12*, they do not coincide. The variance between the two values are relatively small, $1{,}06 m^3/s$, and may be contributed to inaccuracy in either the OpenFOAM model or manual calculations. However, the interpolated value of expected volumetric flow does match the value of measured mass flow at fan inlet, also shown in *table 12*. This therefore points

towards a theory that the fan curve in OpenFOAM does not specify the relations between pressure loss and volumetric flow. But rather the connection between pressure loss and mass flow. Unfortunately, attempts to find the specified unit of measurement within the OpenFOAM documentation has failed. Regardless of this error in assumption, if the fan curve should have been specified in mass flow, the resulting changes in fan performance are not that significant. This is due to the air density value being close to 1, at the fan inlet temperature. The fan curve with mass flow values instead of volumetric flow are shown in *table 15* below.

Table 15: fan curve mass flow

Mass flow [kg/s]	Static pressure [Pa]
14,8	1800
15,9	1600
16,9	1400
18,0	1200
19,1	950
20,1	700
21,2	400

8.5 Mesh quality

To ensure accurate results, the mesh is verified using the yPlus value. Both meshes with temperature variations are tested and the results are shown in *table 16*. The recommended yPlus value is within the range of 0 to 300. Which is true for the largest portion of the mesh, however the maximum values are relatively high.

Table 16: yPlus

Case	Top entry		Side entry	
Atmospheric temperature	273,15 K	263,15 K	273,15 K	263,15 K
yPlus	Max: 1080 Min: 6 Average: 107	Max: 1090 Min: 6 Average: 108	Max: 805 Min: 3 Average: 107	Max: 815 Min: 3 Average: 108
Mesh size	1486140 cells	1486140 cells	1765704 cells	1765704 cells

The cells outside the recommended yPlus range are mostly located in the transition area between the pipe and duct. As can be seen in *figure 21*, the pipe form a sharp edge where it meets the duct. The elevated yPlus values are a result of high flow velocity around the edge, and subsequently the mesh not being fine enough in this particular part of the geometry.

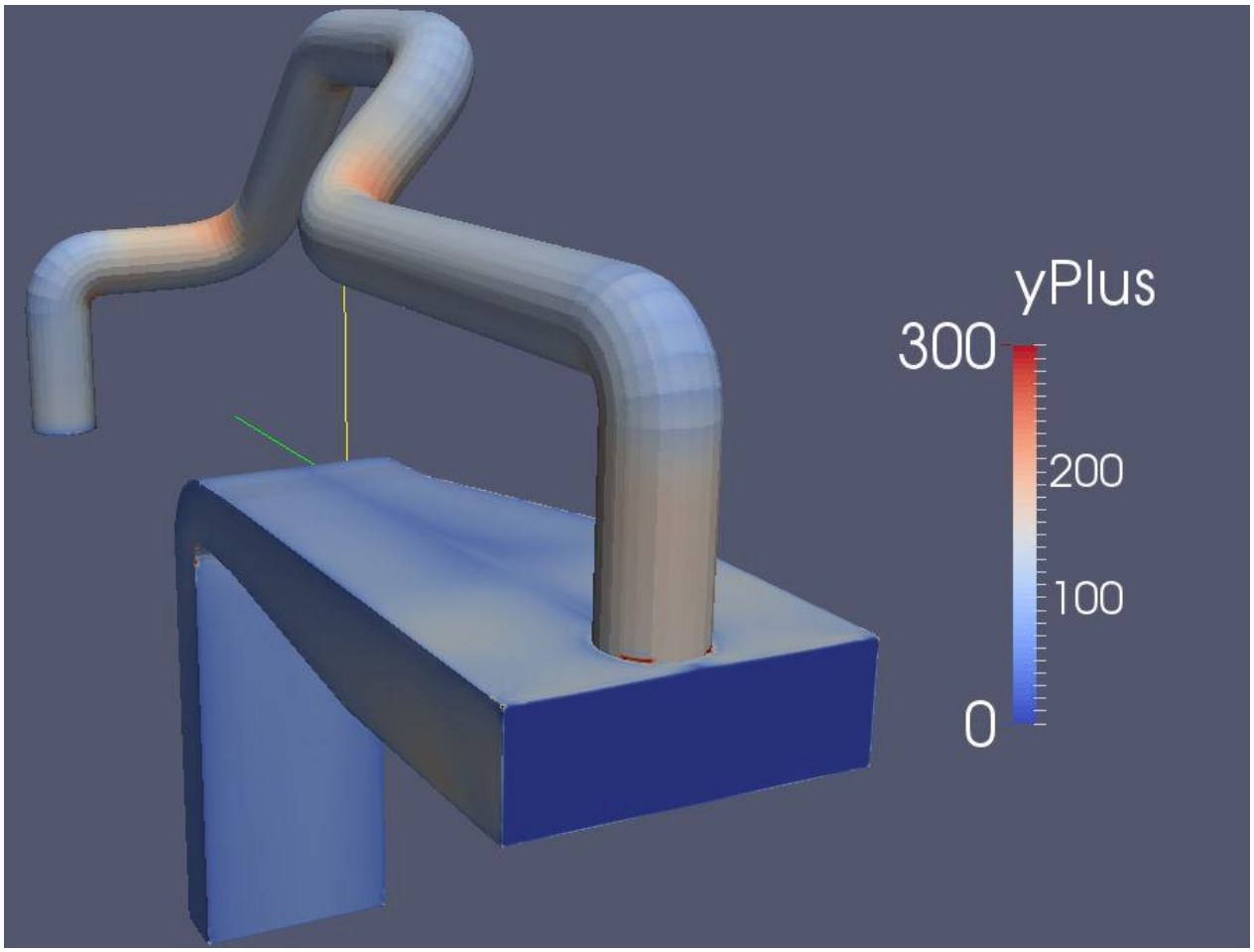


Figure 21: yPlus values top entry case

High yPlus values might indicate that the simulated cases are inaccurate. Therefore, to validate the obtained results, a refined mesh of the top entry case is constructed. The purpose of this improved mesh is to run the case with the same boundary conditions and compare it to the previous results. As seen in *figure 22*, mesh refinement is focused around where the pipe and duct merges. The mesh is now so big in size that it is important to limit the generation of unnecessary cells.

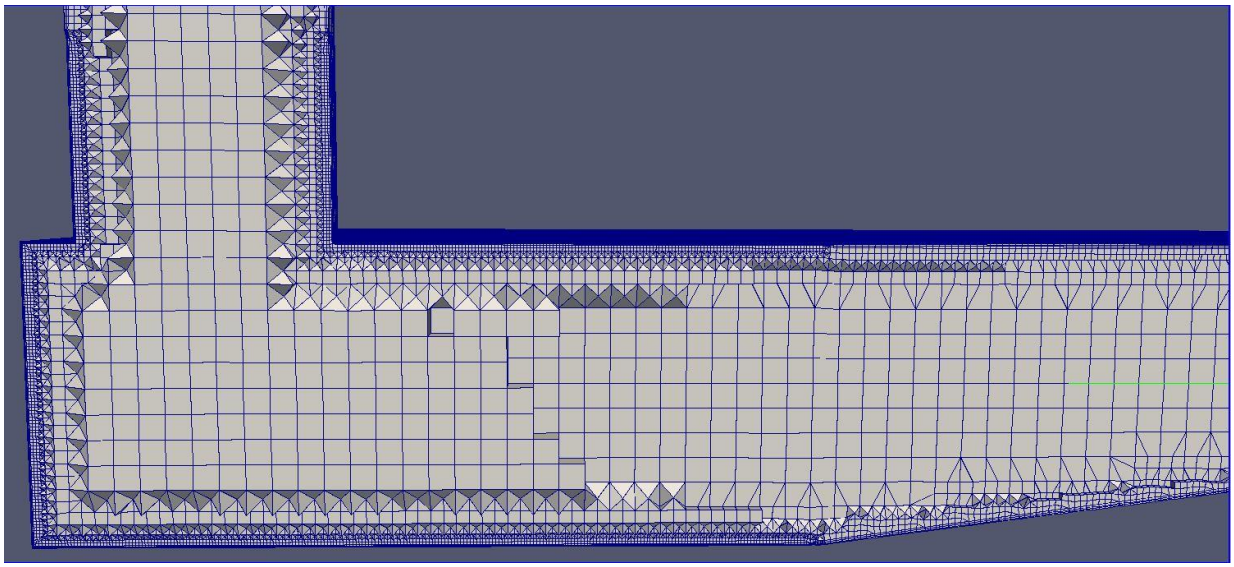


Figure 22: Refined mesh top entry

Table 17: Results of top entry case with finer mesh compared to original mesh

Mesh	Refined			Original		
Patch	Fan inlet	Air inlet	Outlet	Fan inlet	Air inlet	Outlet
Mass flow [kg/s]	19,22	2,00	21,22	19,20	2,00	21,20
Density [kg/m^3]	1,06	1,29	1,09	1,06	1,29	1,09
Temperature [K]	333,15	273,15	323,26	333,15	273,15	323,0

The results with the new mesh is compared to the results from the original mesh. As can be seen in *table 17*, there is an elevation in mass flow and temperature between the two meshes. The increase is small, and assumed not relevant when it comes to mesh quality. Reasons for this deviation in the results will be discussed further in chapter 8.6.

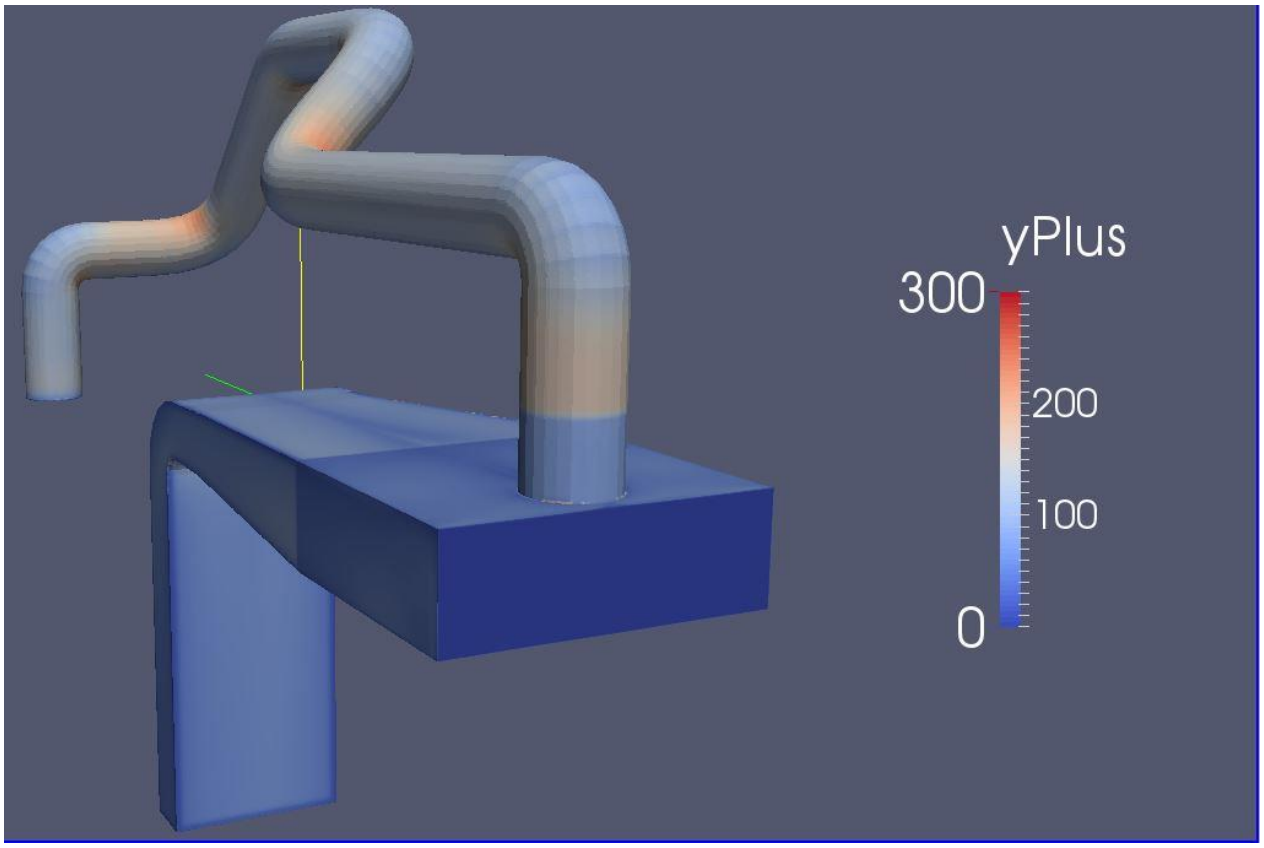


Figure 23: yPlus value for top entry case with refined mesh

The new mesh in *figure 23* show lower yPlus values and deliver a satisfactory result. Even though there are still some high value zones, they are not enough to question the simulation outcome. The low yPlus value and similar field results from *table 17* are both proof of an accurate OpenFOAM model.

Table 18: yPlus from refined mesh

Case	Top entry with refined mesh
Atmospheric temperature	273,15 K
yPlus	Max: 576 Min: 7 Average: 89
Mesh size	2454899 cells

8.6 Residuals

In the previous chapters, some of the evaluated values show some minor deviations from the expected result. As mentioned, they may stem from OpenFOAM inaccuracy or user error. However, by inspecting the behavior of the equation residuals as the simulation is being solved, the variable results may be traced back to the automatically adjusting fanPressure boundary. As the residuals reach acceptable levels and convergence, some of the values start oscillating. This may be due to the fanPressure constantly adjusting the mass flow to match the total system pressure drop. The adjustment of static pressure and flow velocity may cause OpenFOAM to persistently alter the results and not find one steady state solution. Attempts to run the case for longer simulation times does not yield any difference in behavior. In *figure 24*, a test case was run for 30000 iterations and the residuals did not show any trends to stop oscillating. These oscillations are then what may cause the minor deviation in field values discussed in chapter 8.5.

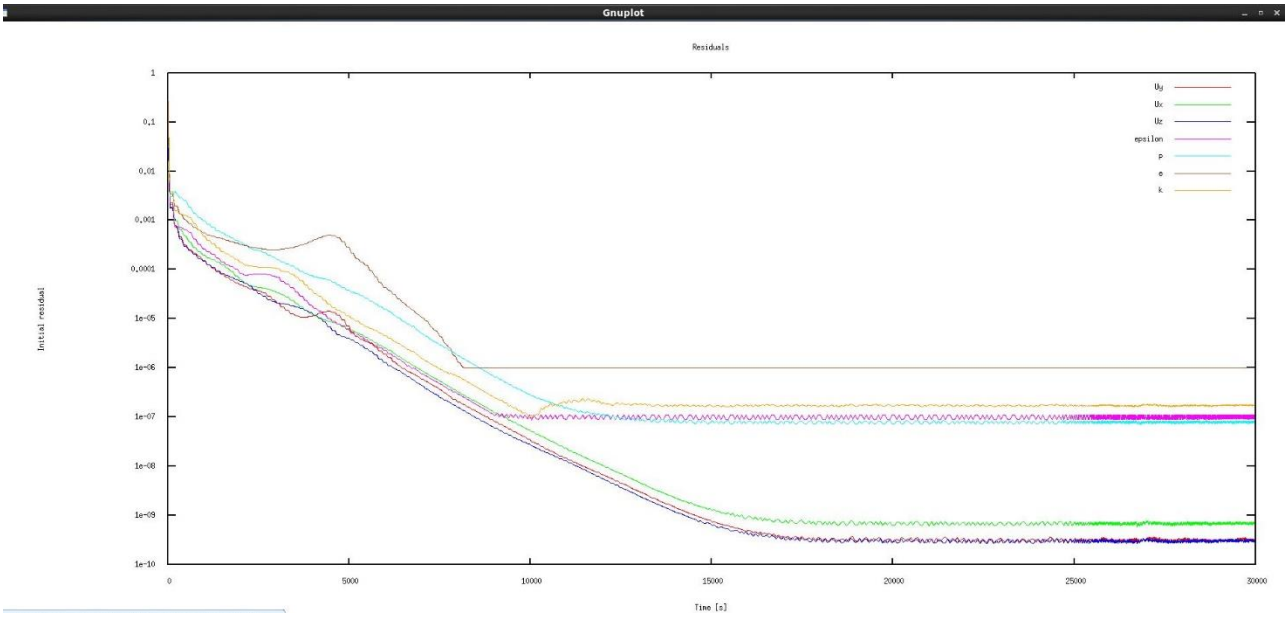


Figure 24: Example of oscillating residuals

9 CONLUSION

9.1 Project conclusion

Evaluating the results obtained from the OpenFOAM model, the fan in question passes all the necessary criteria set by the system requirements. For both designs, the performance is well above the temperature and mass flow limits for even the lowest atmospheric temperature. Still, the side entry design did produce a slightly more favorable result compared to the top entry design. Initially there were some problems finding the correct boundary conditions to simulate the system behavior. Since one of the requirements where a minimum mass flow leaving the system, it would be ideal to fix the flow rate at the outlet and let the air inlet adjust accordingly. This would however, result in an unstable model. The substitute solution of utilizing mass conservation and fixing the mass flow to air inlet did nevertheless produce a satisfactory result. As pointed out in chapter 2.7 information regarding the mechanism drawing air into the turbine is unavailable. Therefore, as long as the two inlet flows add up to more than the required minimum mass flow at outlet, the simulation functions as intended. In addition, the swak4Foam boundary condition could have been used if no other solution had presented itself. The second challenge was to find a boundary condition that could accurately recreate the fan behavior. The fanPressure boundary fulfilled all of the criteria, but the lacking documentation did present some additional problems. A lack of knowledge in C++ language limited the usefulness of the OpenFOAM code and exploring this as a source of information, did not yield any significant results. Unfortunately, this lack of information did manifest itself in the final results. The fan curve text-file was assumed to use volumetric flow as input and was not questioned until all the final OpenFOAM models had been completed. This leaves some uncertainty in the resulting values. Another issue that did arise was the inaccuracy for the chosen method of verifying pressure loss in the simulations. The Bernoulli's equation coupled with Darcy's equation was found to be heavy reliant on the loss coefficient. Finding a reliable value for the loss coefficient did turn out to be a challenge as the different sources did not use similar values. Thus leading to such a wide range of pressure loss values that it could not accurately validate simulation results. Regardless, the OpenFOAM models where validated with the yPlus value and found to have a high quality mesh.

The same is true for solution convergence, reliable results was achieved even though there was some issues with the residuals oscillating.

By evaluating all of the results, it is concluded that the fan in question is fit for system operation. The accuracy of the simulation solution is questioned, but not to a degree that outweighs the tolerance shown in the results compared to system requirements. Throughout this thesis, there has not been found any evidence that the OpenFOAM model is faulty. Therefore, with only a few inaccuracy issues it is concluded that the OpenFOAM models perform as expected and delivers answers to the questions one set forth to find at the beginning of this thesis.

9.2 Future work

In this thesis, a complete model of the air intake system has been created. The OpenFOAM models cover all of the established system requirements. However, before starting work on such an OpenFOAM analysis, a more complete information gathering should be carried out. The assumptions made regarding current turbine operation may be inaccurate. As for validation of pressure calculations, access to accurate data regarding the loss coefficient would improve the results and verify the integrity of the OpenFOAM model. Lastly, with the lack of official OpenFOAM documentation, more knowledge regarding the C++ language would lead to a greater understanding of the boundary conditions. Being able to read the OpenFOAM code would open up a new source of reliable information, otherwise unavailable or found as less reliable second hand information on the OpenFOAM forums.

10 REFERENCES

ASHRAE. 2009. *ASHRAE handbook, fundamentals*, ASHRAE Inc, p. 3.1-3.7.

CFD-WIKI. 2008. *Hydraulic diameter* [Online]
Available: http://www.cfd-online.com/Wiki/Hydraulic_diameter
[Accessed 05.02.2015]

CFD-WIKI. 2012A. *Turbulence length scale* [Online]
Available: http://www.cfd-online.com/Wiki/Turbulence_length_scale
[Accessed 05.02.2015]

CFD-WIKI. 2012B. *Turbulence intensity* [Online]
Available: http://www.cfd-online.com/Wiki/Turbulence_intensity
[Accessed 05.02.2015]

CFD-WIKI. 2014. *Turbulence free-stream boundary conditions* [Online]
Available: http://www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions
[Accessed 05.02.2015]

CHAPALLAZ, J. M., EICHENBERGER, P. & FISCHER, G. 1992. *Manual on pumps used as turbines, MHPG series, harnessing water power on a small scale*, appendix A, section 5.3 [online]
Available: http://www.nzdl.org/gsdlmod?e=d-00000-00---off-0hdl--00-0----0-10-0---0---0direct-10---4-------0-0l--11-en-50---20-help---00-0-1-00-0-0-11-1-0utfZz-8-00-0-0-11-10-0utfZz-8-00&cl=CL3.46&d=HASH011f05bf8734d88d1a080257&gt=1
[Accessed 15.05.2015]

CRANE, 1982. *Flow of fluids through valves, fittings and pipe*, Crane Co, New York, p. A26-A29.

GSCHAIDER, B. 2015A. *PyFoam* [Online]
Available: https://openfoamwiki.net/index.php/Contrib/PyFoam
[Accessed 23.03.2015]

GSCHAIDER, B. 2015B. *swak4Foam* [Online]
Available: https://openfoamwiki.net/index.php/Contrib/swak4Foam
[Accessed 01.05.2015]

HJERTAGER, B. H. 2009A. *Lecture notes in OpenFOAM*, University of Stavanger, Stavanger, p. 11-24, 37-55, 73-85, 111-121.

HJERTAGER, B. H. 2009B. *Computational analysis of fluid flow processes*, University of Stavanger, Stavanger, p. 7-29.

HJERTAGER, B. H. 2009C. *Turbulence theory and modelling*, University of Stavanger, Stavanger, p. 13-26.

HJERTAGER, B. H. 2013. *Fluid dynamics*, McGraw-Hill, New York, p. 420-424.

INCROPERA, F. P., DEWITT, D. P., BERGMAN, T. L. & LAVINE, A. S. 2009. *Introduction to heat transfer*, John Wiley & Sons, Hoboken, p. 399-409, 840-841.

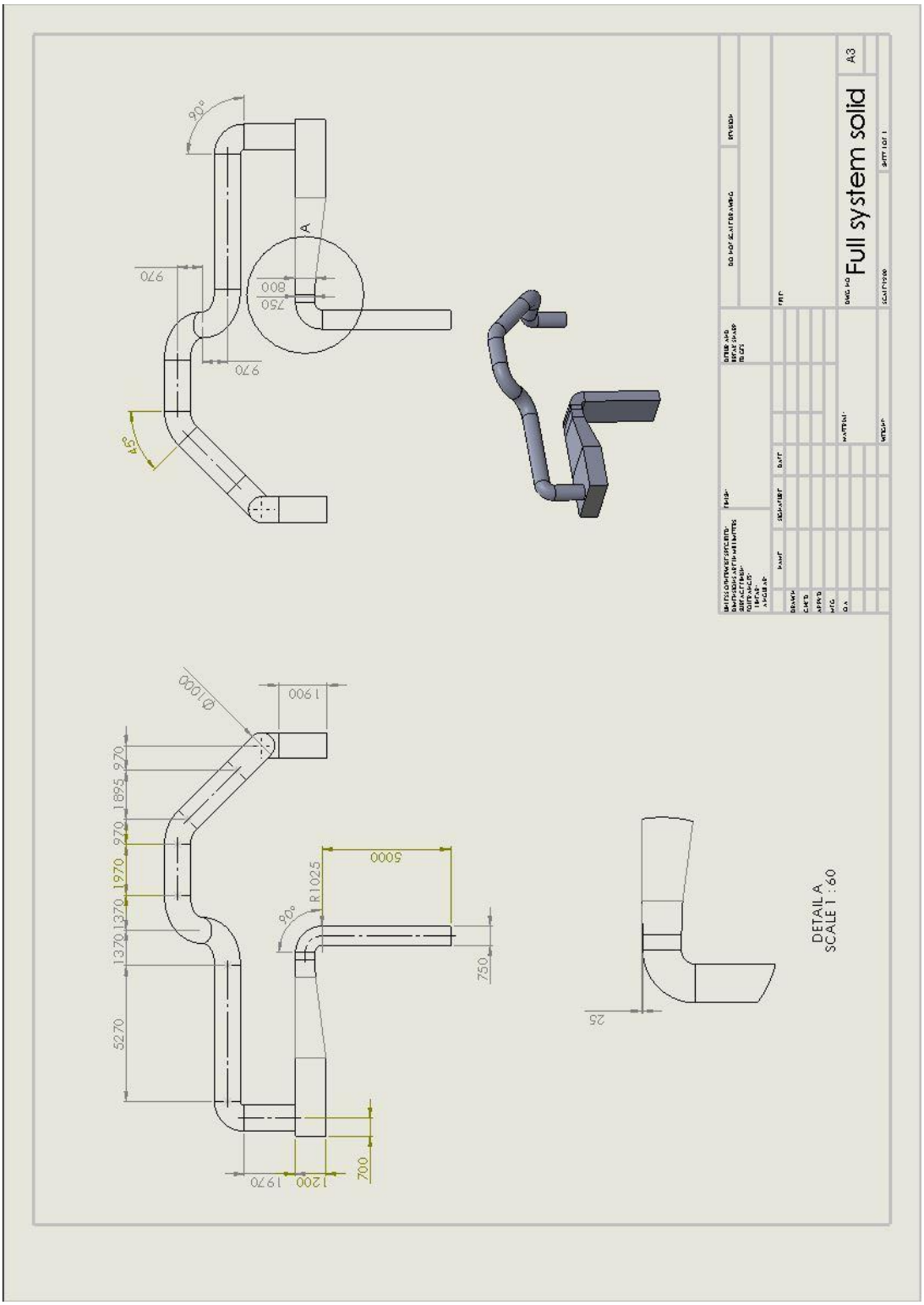
OPENFOAM FOUNDATION. 2011. *Compressible turbulence wall functions* [Online] Available: http://openfoam.github.io/Documentation-dev/html/a00043.html#details [Accessed 02.04.2015]

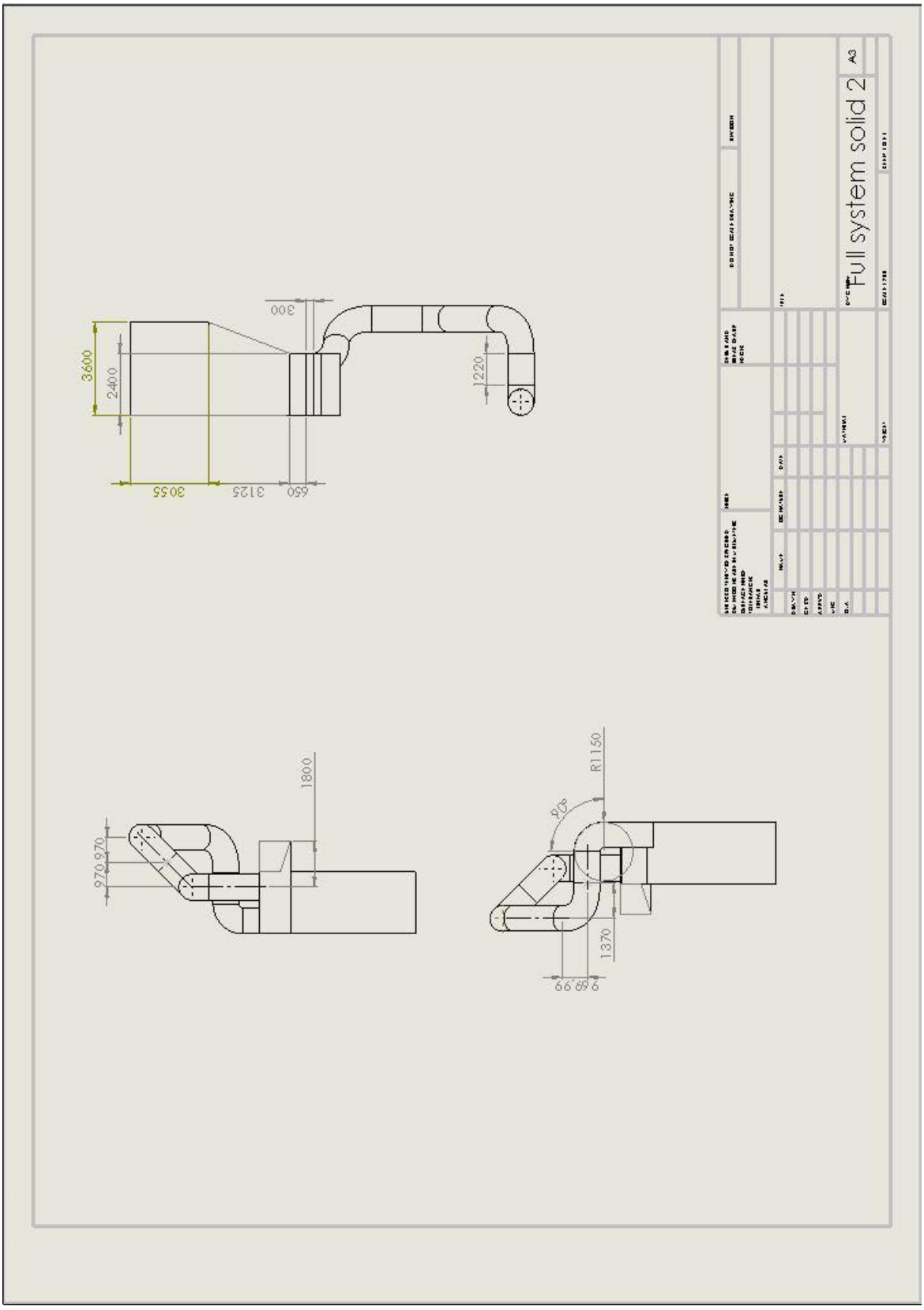
OPENFOAM WIKI. 2012. *Example calcMassFlow* [Online] Available: https://openfoamwiki.net/index.php/Contrib/swak4Foam/Example_calcMassFlow [Accessed 03.05.2015]

VERSTEEG, H. K. & MALALASEKERA, W. 2007. *An introduction to computational fluid dynamics, the finite volume method*, Pearsons Education Limited, Harlow, p. 9-26, 62-76.

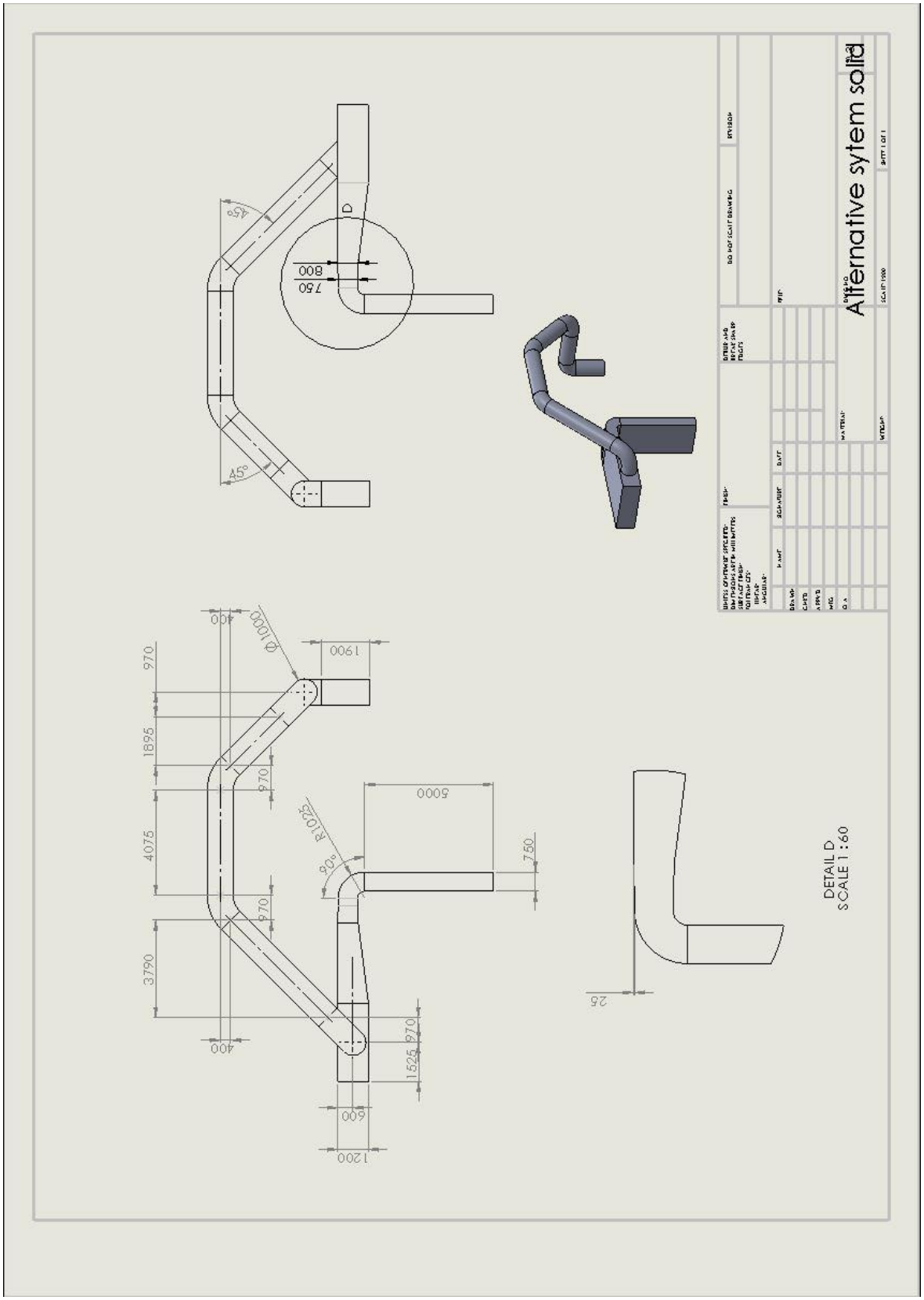
APPENDIX A

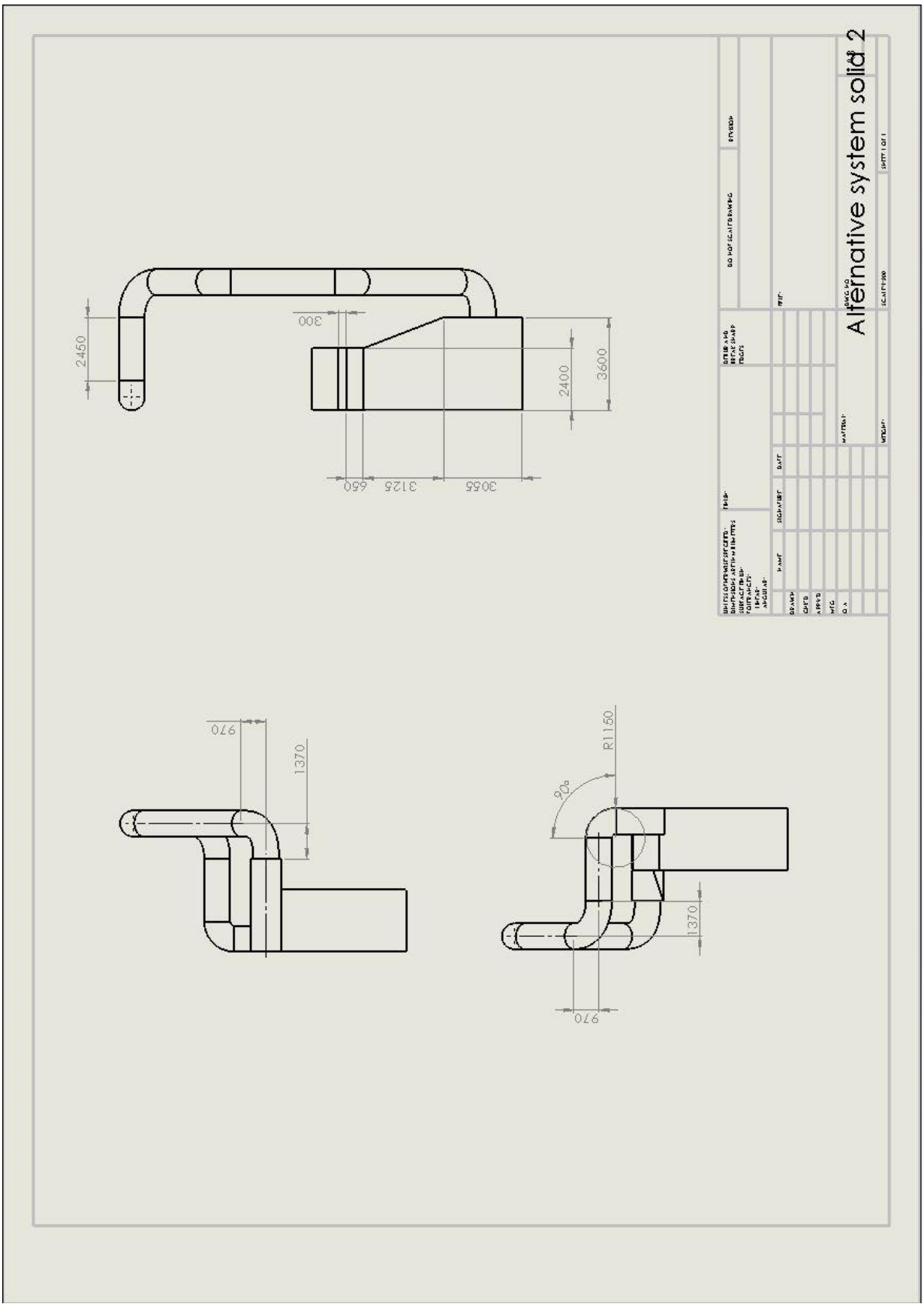
A.1 System dimensions, top entry design




Full system solid 2
A3
3600
2400
300
1220
3055
3125
650
1800
970
970
R1150
90°
1370
66
69
6

A.2 System dimensions, side entry design




Alternative system solid 2
2450
300
2400
3600
660
3125
3055
970
1370
R1160
90°
1370
970

APPENDIX B

B.1 Boundary conditions

B.1.1 Top entry, 273K

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    duct
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value                 uniform 0;
    }
    inlet
    {
        type            calculated;
        value                 uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 1.39;
    }
    duct
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0.00002;
    }
    inlet
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.038;
        value           uniform 1.39;
    }
    air
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.068;
        value           uniform 0.00002;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 1.39;
        value           uniform 1.39;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    duct
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    inlet
    {
        type            fixedValue;
       intensity      0.028;
        value           uniform 0.47;
    }
    air
    {
        type            fixedValue;
       intensity      0.042;
        value           uniform 0.0003;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      0.028;
        value           uniform 0.47;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    duct
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }

}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{

    pipe
    {
        type            zeroGradient;
    }

    duct
    {
        type            zeroGradient;
    }

    inlet
    {
        type            fanPressure;
       patchType      totalPressure;
       fileName       "./constant/fanCurve";
       outOfBounds    clamp;
       direction      in;
       U              U;
       phi            phi;
       psi            none;
       rho            rho;
       p0             uniform 101325; //environmental total pressure
       value          uniform 101325; //initial pressure
       gamma          1;
    }

    air
    {
        type            zeroGradient;
    }


    outlet
    {
        type            fixedValue;
       value          uniform 101325;
    }

}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 293.15;

boundaryField
{
    pipe
    {
        type            wallHeatTransfer;
       alphaWall      uniform 20.75;
       Tinf           uniform 273.15;
       value          uniform 273.15;
    }
    duct
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 333.15;
    }
    air
    {
        type            fixedValue;
        value           uniform 273.15;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 293.15;
        value           uniform 293.15;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    pipe
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    duct
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
    air
    {
        type            flowRateInletVelocity;
        massFlowRate   2;
       value          uniform (0 0 0);
     }
     outlet
    {
        type                  inletOutlet;
       inletValue             uniform (0 0 0);                    //m³/s
        value                 uniform (0 0 0);
    }
}

// ************************************************************************* //
```

B.1.2 Top entry, 263K

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    duct
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value                 uniform 0;
    }
    inlet
    {
        type            calculated;
        value                 uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 1.39;
    }
    duct
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0.00002;
    }
    inlet
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.038;
        value           uniform 1.39;
    }
    air
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.068;
        value           uniform 0.00002;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 1.39;
        value           uniform 1.39;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    duct
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    inlet
    {
        type            fixedValue;
       intensity      0.028;
        value           uniform 0.47;
    }
    air
    {
        type            fixedValue;
       intensity      0.042;
        value           uniform 0.0003;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      0.028;
        value           uniform 0.47;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    duct
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }

}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{

    pipe
    {
        type            zeroGradient;
    }

    duct
    {
        type            zeroGradient;
    }

    inlet
    {
        type            fanPressure;
       patchType      totalPressure;
       fileName       "./constant/fanCurve";
       outOfBounds    clamp;
       direction      in;
       U              U;
       phi            phi;
       psi            none;
       rho            rho;
       p0             uniform 101325; //environmental total pressure
       value          uniform 101325; //initial pressure
       gamma          1;
    }

    air
    {
        type            zeroGradient;
    }


    outlet
    {
        type            fixedValue;
       value          uniform 101325;
    }

}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 263.15;

boundaryField
{
    pipe
    {
        type            wallHeatTransfer;
       alphaWall      uniform 21.09;
       Tinf           uniform 263.15;
       value          uniform 263.15;
    }
    duct
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 333.15;
    }
    air
    {
        type            fixedValue;
        value           uniform 263.15;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 333.15;
        value           uniform 263.15;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    pipe
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    duct
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
    air
    {
        type            flowRateInletVelocity;
        massFlowRate   2;
       value          uniform (0 0 0);
     }
     outlet
    {
        type                  inletOutlet;
       inletValue             uniform (0 0 0);                    //m³/s
        value                 uniform (0 0 0);
    }
}

// ************************************************************************* //
```

B.1.3 Side entry, 273K

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    duct
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value                 uniform 0;
    }
    inlet
    {
        type            calculated;
        value                 uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 1.39;
    }
    duct
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0.00002;
    }
    inlet
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.038;
        value           uniform 1.39;
    }
    air
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.068;
        value           uniform 0.00002;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 1.39;
        value           uniform 1.39;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    duct
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    inlet
    {
        type            fixedValue;
       intensity      0.028;
        value           uniform 0.47;
    }
    air
    {
        type            fixedValue;
       intensity      0.042;
        value           uniform 0.0003;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      0.028;
        value           uniform 0.47;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    duct
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }

}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{

    pipe
    {
        type            zeroGradient;
    }

    duct
    {
        type            zeroGradient;
    }

    inlet
    {
        type            fanPressure;
       patchType      totalPressure;
       fileName       "./constant/fanCurve";
       outOfBounds    clamp;
       direction      in;
       U              U;
       phi            phi;
       psi            none;
       rho            rho;
       p0             uniform 101325; //environmental total pressure
       value          uniform 101325; //initial pressure
       gamma          1;
    }

    air
    {
        type            zeroGradient;
    }


    outlet
    {
        type            fixedValue;
       value          uniform 101325;
    }

}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 273.15;

boundaryField
{
    pipe
    {
        type            wallHeatTransfer;
       alphaWall      uniform 20.75;
       Tinf           uniform 273.15;
       value          uniform 273.15;
    }
    duct
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 333.15;
    }
    air
    {
        type            fixedValue;
        value           uniform 273.15;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 333.15;
        value           uniform 273.15;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    pipe
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    duct
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
    air
    {
        type            flowRateInletVelocity;
        massFlowRate   2;
       value           uniform (0 0 0);
     }
     outlet
    {
        type                  inletOutlet;
       inletValue             uniform (0 0 0);                    //m³/s
        value                 uniform (0 0 0);
    }
}

// ************************************************************************* //
```

B.1.4 Side entry, 263K

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alphat;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    duct
    {
        type            compressible::alphatWallFunction;
        Prt             0.7309;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value                 uniform 0;
    }
    inlet
    {
        type            calculated;
        value                 uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 1.39;
    }
    duct
    {
        type            compressible::epsilonWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0.00002;
    }
    inlet
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.038;
        value           uniform 1.39;
    }
    air
    {
        type            compressible::turbulentMixingLengthDissipationRateInlet;
        mixingLength    0.068;
        value           uniform 0.00002;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 1.39;
        value           uniform 1.39;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    pipe
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    duct
    {
        type            compressible::kqRWallFunction;
        value           uniform 1;
    }
    inlet
    {
        type            fixedValue;
       intensity      0.028;
        value           uniform 0.47;
    }
    air
    {
        type            fixedValue;
       intensity      0.042;
        value           uniform 0.0003;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      0.028;
        value           uniform 0.47;
    }
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{

    pipe
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    duct
    {
        type            mutkWallFunction;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
        value           uniform 0;
    }
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    air
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }

}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{

    pipe
    {
        type            zeroGradient;
    }

    duct
    {
        type            zeroGradient;
    }

    inlet
    {
        type            fanPressure;
       patchType      totalPressure;
       fileName       "./constant/fanCurve";
       outOfBounds    clamp;
       direction      in;
       U              U;
       phi            phi;
       psi            none;
       rho            rho;
       p0             uniform 101325; //environmental total pressure
       value          uniform 101325; //initial pressure
       gamma          1;
    }

    air
    {
        type            zeroGradient;
    }


    outlet
    {
        type            fixedValue;
       value          uniform 101325;
    }

}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 263.15;

boundaryField
{
    pipe
    {
        type            wallHeatTransfer;
       alphaWall      uniform 21.09;
       Tinf           uniform 263.15;
       value          uniform 263.15;
    }
    duct
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 333.15;
    }
    air
    {
        type            fixedValue;
        value           uniform 263.15;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 333.15;
        value           uniform 263.15;
    }
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    pipe
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    duct
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
    air
    {
        type            flowRateInletVelocity;
        massFlowRate   2;
       value           uniform (0 0 0);
     }
     outlet
    {
        type                  inletOutlet;
       inletValue             uniform (0 0 0);                    //m³/s
        value                 uniform (0 0 0);
    }
}

// ************************************************************************* //
```

APPENDIX C

C.1 Fan curve

```
7
(
(14 1800)
(15 1600)
(16 1400)
(17 1200)
(18 950)
(19 700)
(20 400)
)
```

C.2 RASProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;


// ************************************************************************* //
```

C.3 thermophysicalProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       sutherland;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles      1;
        molWeight   28.9;
    }
    thermodynamics
    {
        Cp          1005;
        Hf          0;
    }
    transport
    {
        As          1.4792e-06;
        Ts          116;
    }
}


// ************************************************************************* //
```

APPENDIX D

D.1 ControlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     rhoSimpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         10000;

deltaT          1;

writeControl    timeStep;

writeInterval   500;

purgeWrite      0;

writeFormat     ascii;

writePrecision  8;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

libs (
    "libgroovyStandardBCs.so"
    "libOpenFOAM.so"
    "libgroovyBC.so"
    "libsimpleSwakFunctionObjects.so"
    "libswakFunctionObjects.so"
    "libswakTopoSources.so"
    "libcompressibleRASModels.so"
    "libswakPythonIntegration.so"
) ;

functions {
patchMassFlow
    {
        type patchExpression;
        accumulations (
```

```
            sum
        );
        patches (
            inlet
            outlet
            air
        );
        expression "phi";
        verbose true;
    }
patchDensity
    {
        type patchExpression;
        accumulations (
            average
        );
        patches (
            inlet
            outlet
            air
        );
        expression "rho";
        verbose true;
    }
patchVelocity
    {
        type patchExpression;
        accumulations (
            average
        );
        patches (
            inlet
            outlet
            air
        );
        expression "U";
        verbose true;
    }
pressureDropInletOutlet
    {
        type patchExpression;
        variables ( "pOut{patch'outlet}=sum(p*area())/sum(area());");
        accumulations (
            average
        );
        patches (
            inlet
        );
        expression "p-pOut";
        verbose true;
    }
pressureDropInletAir
    {
        type patchExpression;
        variables ( "pAir{patch'air}=sum(p*area())/sum(area());");
        accumulations (
            average
        );
        patches (
            inlet
        );
        expression "p-pAir";
        verbose true;
    }
patchPressure
    {
        type patchExpression;
        accumulations (
            average
        );
        patches (
```

```
            inlet
            outlet
            air
        );
        expression "p";
        verbose true;
    }
}


// ************************************************************************* //
```

D.2 decomposeParDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 2; // # of CPU cores

method          simple;

simpleCoeffs
{
    n               ( 2 1 1 ); // needs to multiply to = # cores
    delta           0.001;
}

hierarchicalCoeffs
{
    n               ( 1 1 1 );
    delta           0.001;
    order           xyz;
}

manualCoeffs
{
    dataFile        "cellDecomposition";
}

// ************************************************************************* //
```

D.3 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;

    div(phi,U)      bounded Gauss upwind;
    div((muEff*dev2(T(grad(U))))) Gauss linear;
    div(phi,e)      bounded Gauss upwind;
    div(phi,epsilon) bounded Gauss upwind;
    div(phi,k)      bounded Gauss upwind;
    div(phi,Ekp)    bounded Gauss upwind;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

D.4 fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-08;
        relTol          0.05;
        smoother        GaussSeidel;
        cacheAgglomeration on;
        nCellsInCoarsestLevel 20;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-06;
        relTol          0.1;
    }

    e
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }

    "(k|epsilon)"
    {
        $U;
        tolerance       1e-07;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    rhoMin          rhoMin [ 1 -3 0 0 0 ] 0.5;
    rhoMax          rhoMax [ 1 -3 0 0 0 ] 1.5;

    residualControl
    {
        p               1e-9;
        U               1e-10;
        e               1e-10;
```

```
        // possibly check turbulence fields
        "(k|epsilon|omega)" 1e-6;
    }
}

relaxationFactors
{
    fields
    {
        p               0.5;
        rho             0.05; //0.01
    }
    equations
    {
        U               0.5;
        "(k|epsilon)"   0.7; //0.01
        e               0.5; //0.01
    }
}


// ************************************************************************* //
```

D.5 Top entry

D.5.1 sampleDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      sampleDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Set output format : choice of
//      xmgr
//      jplot
//      gnuplot
//      raw
//      vtk
//      ensight
//      csv
setFormat raw;

// Surface output format. Choice of
//      null        : suppress output
//      ensight     : Ensight Gold format, one field per case file
//      foamFile    : separate points, faces and values file
//      dx          : DX scalar or vector format
//      vtk         : VTK ascii format
//      raw         : x y z value format for use with e.g. gnuplot 'splot'.
//
// Note:
// other formats such as obj, stl, etc can also be written (by proxy)
// but without any values!
surfaceFormat foamFile;

// optionally define extra controls for the output formats
formatOptions
{
    ensight
    {
        format  ascii;
    }
}

// interpolationScheme. choice of
//      cell          : use cell-centre value only; constant over cells
//                      (default)
//      cellPoint     : use cell-centre and vertex values
//      cellPointFace : use cell-centre, vertex and face values.
//      pointMVC      : use point values only (Mean Value Coordinates)
//      cellPatchConstrained : like 'cell' but uses cell-centre except on
//                             boundary faces where it uses the boundary value.
//                             For use with e.g. patchCloudSet.
// 1] vertex values determined from neighbouring cell-centre values
// 2] face values determined using the current face interpolation scheme
//    for the field (linear, gamma, etc.)
interpolationScheme cellPoint;
```

```
// Fields to sample.
fields
(
    T
);

// Set sampling definition: choice of
//      uniform             evenly distributed points on line
//      face                one point per face intersection
//      midPoint            one point per cell, inbetween two face intersections
//      midPointAndFace     combination of face and midPoint
//
//      polyLine            specified points, not nessecary on line, uses
//                          tracking
//      cloud               specified points, uses findCell
//      triSurfaceMeshPointSet  points of triSurface
//
// axis: how to write point coordinate. Choice of
// - x/y/z: x/y/z coordinate only
// - xyz: three columns
//  (probably does not make sense for anything but raw)
// - distance: distance from start of sampling line (if uses line) or
//             distance from first specified sampling point
//
// type specific:
//      uniform, face, midPoint, midPointAndFace : start and end coordinate
//      uniform: extra number of sampling points
//      polyLine, cloud: list of coordinates
//      patchCloud: list of coordinates and set of patches to look for nearest
//      patchSeed: random sampling on set of patches. Points slightly off
//                 face centre.
// Surface sampling definition
//
// 1] patches are not triangulated by default
// 2] planes are always triangulated
// 3] iso-surfaces are always triangulated
surfaces
(
    plane1
    {
        type            plane;    // always triangulated
        basePoint       (1.26 2.55 0);
        normalVector    (1 0 0);

        //- Optional: restrict to a particular zone
        zone           zone1;
    }
    plane2
    {
        type            plane;    // always triangulated
        basePoint       (3.24 5.815 4.5);
        normalVector    (0 0 1);

        //- Optional: restrict to a particular zone
        zone           zone2;
    }
    plane3
    {
        type            plane;    // always triangulated
        basePoint       (1.3 3.875 10);
        normalVector    (0 0 1);

        //- Optional: restrict to a particular zone
        zone           zone2;
    }
    plane4
    {
        type            plane;    // always triangulated
        basePoint       (1.3 1.3 14.465);
        normalVector    (0 1 0);
```

```
        //- Optional: restrict to a particular zone
        zone           zone2;
    }
    plane5
    {
        type            plane;    // always triangulated
        basePoint       (0.7 0.87 8.985);
        normalVector    (0 0 1);

        //- Optional: restrict to a particular zone
        zone           zone3;
    }
    plane6
    {
        type            plane;    // always triangulated
        basePoint       (0.7 -1 7.385);
        normalVector    (0 1 0);

        //- Optional: restrict to a particular zone
        zone           zone3;
    }

    inlet
    {
        type            patch;
        patches         ( ".*inlet.*" );
        interpolate     false;
    }
    air
    {
        type            patch;
        patches         ( ".*air.*" );
        interpolate     false;
    }
    outlet
    {
        type            patch;
        patches         ( ".*outlet.*" );
        interpolate     false;
    }


);


// ************************************************************************* //
```

D.5.2 snappyHexMesh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh false;    // make basic mesh ?
snap            false;    // decide to snap back to surface ?
addLayers       true;   // decide to add viscous layers ?


geometry // Load in STL files here
{
        inlet.stl {type triSurfaceMesh; name inlet;}
        outlet.stl {type triSurfaceMesh; name outlet;}
        air.stl {type triSurfaceMesh; name air;}
        pipe.stl {type triSurfaceMesh; name pipe;}
        duct.stl {type triSurfaceMesh; name duct;}
        refinementArea.stl {type triSurfaceMesh; name refinementArea;}
        volume.stl {type triSurfaceMesh; name volume;}
        refinementBox {type searchableBox; min (-0.5 0.055 13.3); max (3.1 1.6 15.165);}
};

castellatedMeshControls
{
    maxLocalCells 100000;  //max cells per CPU core
    maxGlobalCells 2000000; //max cells to use before mesh deletion step
    minRefinementCells 10;  //was 0 - zero means no bad cells are allowed during refinement
stages
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;  // expansion factor between each high & low refinement zone

    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    features // taken from STL from each .eMesh file created by "SurfaceFeatureExtract" command
    (
        {file "ductClosed.eMesh"; level 2;}
    );

    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    refinementSurfaces // Surface-wise min and max refinement level
    {
        inlet {level (0 0);}
         outlet {level (0 0);}
        air {level (1 1);}
        pipe {level (1 1);}
        duct {level (1 1);}
    }


    resolveFeatureAngle 30;  // Resolve sharp angles // Default 30
    refinementRegions        // In descending levels of fine-ness
    {pipe {mode distance; levels ((0.05 2));} // was ((0.001 4) (0.003 3) (0.01 2))
    duct {mode distance; levels ((0.05 2));}
```

```
    air {mode distance; levels ((0.05 2));}}
    locationInMesh (0 1 0);  //to decide which side of mesh to keep **
    allowFreeStandingZoneFaces true;
}


// Settings for the snapping.
snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 30;
    nRelaxIter 5;
    nFeatureSnapIter 15; // default is 10

// New settings from openfoam 2.2 onwards for SHMesh

implicitFeatureSnap true; // default is false - detects without doing surfaceFeatureExtract
explicitFeatureSnap true; // default is true
multiRegionFeatureSnap false; // deafault is false - detects features between multiple surfaces

}



// Settings for the layer addition.
addLayersControls //add the PATCH names from inside the STL file so STLpatchName_insideSTLName
{
    relativeSizes true; // was true
    layers
    {
       pipe
              {nSurfaceLayers 3;} // was 3
       duct
              {nSurfaceLayers 3;} // was 3
       air
              {nSurfaceLayers 3;} // was 3
    }

    expansionRatio 1.3;
    finalLayerThickness 0.3; //was 0.00016
    minThickness 0.1; //was 0.00008
    nGrow 0; // was 1


    // Advanced settings

    featureAngle 180; // was 70 //- When not to extrude surface. 0 is flat, 90 is right angle.
    nRelaxIter 3;  //- Max# of snapping relaxation iter. Should stop before upon reaching a
correct mesh.
    nSmoothSurfaceNormals 1;  // Number of smoothing iterations of surface normals
    nSmoothNormals 3; // Number of smoothing iterations of interior mesh movement direction
    nSmoothThickness 10;  // Smooth layer thickness over surface patches
    maxFaceThicknessRatio 0.5; // Stop layer growth on highly warped cells
    maxThicknessToMedialRatio 0.3; // Reduce layer growth where ratio thickness to medial
distance is large
    minMedianAxisAngle 130;  // Angle used to pick up medial axis points
    nBufferCellsNoExtrude 0;   // Create buffer region for new layer terminations
    nLayerIter 50; // Overall max number of layer addition iterations
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
```

```
    minFlatness 0.5;
    minVol 1e-13;
    minTetQuality 1e-20;
    minArea -1;
    minTwist 0.02;
    minDeterminant 0.001;
    minFaceWeight 0.02;
    minVolRatio 0.01;
    minTriangleTwist -1;

    // Advanced

    nSmoothScale 4;
    errorReduction 0.75;
}

// Advanced

debug 0;


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1E-6;


// ************************************************************************* //
```

D.5.3 topoSetDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      topoSetDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    zone1;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            box               (-1 -0.5 -1) (4 7 3.835);
        }
    }
    {
        name    zone1;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set               zone1;
        }
    }

    {
        name    zone2;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            box               (-1 1.255 3.835) (4 7 16);
        }
    }
    {
        name    zone2;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set               zone2;
        }
    }

    {
        name    zone3;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
```

```
            box               (-1 -5 2) (4 1.255 16);
        }
    }
    {
        name    zone3;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set               zone3;
        }
    }


);

// ************************************************************************* //
```

D.6 Side entry

D.6.1 sampleDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      sampleDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Set output format : choice of
//      xmgr
//      jplot
//      gnuplot
//      raw
//      vtk
//      ensight
//      csv
setFormat raw;

// Surface output format. Choice of
//      null        : suppress output
//      ensight     : Ensight Gold format, one field per case file
//      foamFile    : separate points, faces and values file
//      dx          : DX scalar or vector format
//      vtk         : VTK ascii format
//      raw         : x y z value format for use with e.g. gnuplot 'splot'.
//
// Note:
// other formats such as obj, stl, etc can also be written (by proxy)
// but without any values!
surfaceFormat foamFile;

// optionally define extra controls for the output formats
formatOptions
{
    ensight
    {
        format  ascii;
    }
}

// interpolationScheme. choice of
//      cell          : use cell-centre value only; constant over cells
//                      (default)
//      cellPoint     : use cell-centre and vertex values
//      cellPointFace : use cell-centre, vertex and face values.
//      pointMVC      : use point values only (Mean Value Coordinates)
//      cellPatchConstrained : like 'cell' but uses cell-centre except on
//                             boundary faces where it uses the boundary value.
//                             For use with e.g. patchCloudSet.
// 1] vertex values determined from neighbouring cell-centre values
// 2] face values determined using the current face interpolation scheme
//    for the field (linear, gamma, etc.)
interpolationScheme cellPoint;

// Fields to sample.
```

```
fields
(
    T
);

// Set sampling definition: choice of
//      uniform             evenly distributed points on line
//      face                one point per face intersection
//      midPoint            one point per cell, inbetween two face intersections
//      midPointAndFace     combination of face and midPoint
//
//      polyLine            specified points, not nessecary on line, uses
//                          tracking
//      cloud               specified points, uses findCell
//      triSurfaceMeshPointSet  points of triSurface
//
// axis: how to write point coordinate. Choice of
// - x/y/z: x/y/z coordinate only
// - xyz: three columns
//  (probably does not make sense for anything but raw)
// - distance: distance from start of sampling line (if uses line) or
//             distance from first specified sampling point
//
// type specific:
//      uniform, face, midPoint, midPointAndFace : start and end coordinate
//      uniform: extra number of sampling points
//      polyLine, cloud: list of coordinates
//      patchCloud: list of coordinates and set of patches to look for nearest
//      patchSeed: random sampling on set of patches. Points slightly off
//                 face centre.
// Surface sampling definition
//
// 1] patches are not triangulated by default
// 2] planes are always triangulated
// 3] iso-surfaces are always triangulated
surfaces
(
    plane1
    {
        type            plane;    // always triangulated
        basePoint       (1.8 2.55 0);
        normalVector    (1 0 0);

        //- Optional: restrict to a particular zone
        zone           zone1;
    }
    plane2
    {
        type            plane;    // always triangulated
        basePoint       (4.47 5.815 5.9);
        normalVector    (0 0 1);

        //- Optional: restrict to a particular zone
        zone           zone2;
    }
    plane3
    {
        type            plane;    // always triangulated
        basePoint       (3.11 0.655 13.64);
        normalVector    (1 0 0);

        //- Optional: restrict to a particular zone
        zone           zone2;
    }
    plane4
    {
        type            plane;    // always triangulated
        basePoint       (0.7 0.87 8.985);
        normalVector    (0 0 1);

        //- Optional: restrict to a particular zone
```

```
        zone           zone3;
    }
    plane5
    {
        type            plane;    // always triangulated
        basePoint       (0.7 -1 7.385);
        normalVector    (0 1 0);

        //- Optional: restrict to a particular zone
        zone           zone3;
    }

    inlet
    {
        type            patch;
        patches         ( ".*inlet.*" );
        interpolate     false;
    }
    air
    {
        type            patch;
        patches         ( ".*air.*" );
        interpolate     false;
    }
    outlet
    {
        type            patch;
        patches         ( ".*outlet.*" );
        interpolate     false;
    }


);

// ************************************************************************* //
```

D.6.2 snappyHexMesh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh false;    // make basic mesh ?
snap            false;    // decide to snap back to surface ?
addLayers       true;   // decide to add viscous layers ?


geometry // Load in STL files here
{
        inlet.stl {type triSurfaceMesh; name inlet;}
        outlet.stl {type triSurfaceMesh; name outlet;}
        air.stl {type triSurfaceMesh; name air;}
        pipe.stl {type triSurfaceMesh; name pipe;}
        duct.stl {type triSurfaceMesh; name duct;}
        volume.stl {type triSurfaceMesh; name volume;}
        refinementBox {type searchableBox; min (-0.5 0.055 13.3); max (3.1 1.6 15.165);}
};

castellatedMeshControls
{
    maxLocalCells 100000;  //max cells per CPU core
    maxGlobalCells 2000000; //max cells to use before mesh deletion step
    minRefinementCells 10;  //was 0 - zero means no bad cells are allowed during refinement
stages
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;  // expansion factor between each high & low refinement zone

    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    features // taken from STL from each .eMesh file created by "SurfaceFeatureExtract" command
    (
        {file "ductClosed.eMesh"; level 2;}
    );

    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    refinementSurfaces // Surface-wise min and max refinement level
    {
        inlet {level (0 0);}
        outlet {level (0 0);}
        air {level (1 1);}
        pipe {level (1 1);}
        duct {level (1 1);}
    }


    resolveFeatureAngle 5;  // Resolve sharp angles // Default 30
    refinementRegions        // In descending levels of fine-ness
    {pipe {mode distance; levels ((0.1 2));} // was ((0.001 4) (0.003 3) (0.01 2))
    duct {mode distance; levels ((0.1 2));}
```

```
    air {mode distance; levels ((0.05 2));}}
    locationInMesh (0 1 0);  //to decide which side of mesh to keep **
    allowFreeStandingZoneFaces true;
}


// Settings for the snapping.
snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 30;
    nRelaxIter 5;
    nFeatureSnapIter 15; // default is 10

// New settings from openfoam 2.2 onwards for SHMesh

implicitFeatureSnap true; // default is false - detects without doing surfaceFeatureExtract
explicitFeatureSnap true; // default is true
multiRegionFeatureSnap false; // deafault is false - detects features between multiple surfaces

}



// Settings for the layer addition.
addLayersControls //add the PATCH names from inside the STL file so STLpatchName_insideSTLName
{
    relativeSizes true; // was true
    layers
    {
       pipe
              {nSurfaceLayers 3;} // was 3
       duct
              {nSurfaceLayers 3;} // was 3
       air
              {nSurfaceLayers 3;} // was 3
    }

    expansionRatio 1.3;
    finalLayerThickness 0.3; //was 0.00016
    minThickness 0.1; //was 0.00008
    nGrow 0; // was 1


    // Advanced settings

    featureAngle 220; // was 70 //- When not to extrude surface. 0 is flat, 90 is right angle.
    nRelaxIter 3;  //- Max# of snapping relaxation iter. Should stop before upon reaching a
correct mesh.
    nSmoothSurfaceNormals 1;  // Number of smoothing iterations of surface normals
    nSmoothNormals 3; // Number of smoothing iterations of interior mesh movement direction
    nSmoothThickness 10;  // Smooth layer thickness over surface patches
    maxFaceThicknessRatio 0.5; // Stop layer growth on highly warped cells
    maxThicknessToMedialRatio 0.3; // Reduce layer growth where ratio thickness to medial
distance is large
    minMedianAxisAngle 130;  // Angle used to pick up medial axis points
    nBufferCellsNoExtrude 0;   // Create buffer region for new layer terminations
    nLayerIter 50; // Overall max number of layer addition iterations
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
```

```
    minFlatness 0.5;
    minVol 1e-13;
    minTetQuality 1e-20;
    minArea -1;
    minTwist 0.02;
    minDeterminant 0.001;
    minFaceWeight 0.02;
    minVolRatio 0.01;
    minTriangleTwist -1;

    // Advanced

    nSmoothScale 4;
    errorReduction 0.75;
}

// Advanced

debug 0;


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1E-6;


// ************************************************************************* //
```

D.6.3 topoSetDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      topoSetDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    zone1;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            box               (-1 -0.5 -1) (5.5 7 3.835);
        }
    }
    {
        name    zone1;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set               zone1;
        }
    }

    {
        name    zone2;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            box               (3.1 0 3.835) (5.5 7 16);
        }
    }
    {
        name    zone2;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set               zone2;
        }
    }

    {
        name    zone3;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
```

```
            box               (-1 -5 2) (3.1 1.255 16);
        }
    }
    {
        name    zone3;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set               zone3;
        }
    }


);

// ************************************************************************* //
```

APPENDIX E

E.1 Content of enclosed CD

The CD contains the case folders used in OpenFOAM in a ZIP format. There are five cases which describe the two design cases run at both 273,15K and 263,15K atmospheric temperature. In addition, the case which utilized a refined mesh to rerun the top entry case at 273,15K, to ensure the mesh quality. All cases include complete 0, constant, system and postProcessing folders. Also included is a copy of the thesis in a pdf and docx format.