Universitetet
i Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/specialization: Construction and Material Science	Spring semester, 2015 Confidensial
Writer: Benjamin Pettersen	………………………………………… (Writer's signature)
Faculty supervisor: Bjørn Helge Hjertager External supervisor(s): Jon Dale Gjerstad	
Title of thesis: Nitrogas HeliNite System	
Credits (ECTS): 30	
Key words: - Leak Detection - HeliNite - Gas Mixing - Coefficient of Variance - Computational Fluid Dynamics - reactingFoam	Pages: 40 + Enclosure: 38 + CD Stavanger, 11.06.2015

Abstract

In today's leak detection market the substance HeliNite is widely used. It contains helium and nitrogen gas, and offers highly sensitive leak detection. The company Nitrogas AS has patented and constructed a nitrogen generator. During the construction process there was an idea of introducing a helium bottle to the system, which enables the generation of the substance HeliNite. Instead of acquiring prefilled bottles, Nitrogas AS intends to offer the possibility of on-site production of HeliNite – allowing versatility and customization of leak tests. Hopefully the system will meet an unsaturated demand.

The thesis has offered a draft of the Nitrogas HeliNite system. To ensure the accuracy and repeatability of the leak test the HeliNite mix needs to be properly blended. There has been completed a thorough investigation of the requirements for a uniform gas mix.

A presentation of required components, considerations and a walkthrough of the system have been provided. Whether or not the presentation of the Nitrogas HeliNite system is successful cannot be determined until the construction and testing of the first prototype, but the proposal should have provided a solid foundation.

The investigation of the gas mixing was done in the software OpenFOAM, using the solver `reactingFoam`. The experiments commenced by simulations on a clean pipe, followed by stepwise increases in inlet velocity ratios and lastly baffles in the flow direction. The quantification of the mixing was done by analyzing local concentration levels of helium by calculations of the coefficient of variance (CoV). The threshold for a uniform mix was set to CoV = 0.05. The results of the simulations suggested a velocity ratio of 48.8 to ensure a uniform mix, CoV = 0.04. The introduction of one baffle gave a CoV = 0.003, while two baffles ensured a CoV = 0.007. The conducted experiments claim that a uniform gas mix can be achieved either by a velocity ratio of 48.8 or minor obstructions.

Acknowledgements

The thesis is written for the company Nitrogas AS. It’s a part of a master’s degree in the field of construction and material science. I would like to give thanks to my supervisor Bjørn Helge Hjertager for guiding me through CFD related problems, providing me with relevant literature and for motivating me to get a move on. Additional thanks go to Jon Dale Gjerstad offering me the chance to write the thesis for the company Nitrogas AS, Knut Erik Teigen Giljarhus for quality viewpoints on CFD related issues, and my mother Anita Hetland Pettersen for hours of proofreading.

Table of Contents

List of Figures

List of Tables

Nomenclature

Greek Symbols

Symbol	Description	Units
δ_{ij}	Unit tensor	[ ]
Δt	Maximum time step	[s]
Δx	Local grid cell dimension	[m]
ε	Turbulent dissipation	[m^2/s^3]
θ	Angular coordinate	[rad.]
μ	Dynamic viscosity	[kg/(m·s)]
μ_t	Dynamic eddy viscosity	[kg/(m·s)]
ν	Kinematic viscosity	[m^2/s]
ν_t	Kinematic eddy viscosity	[m^2/s]
ρ	Density of mass	[kg/m^3]
σ_h	Mixture's Prandtl number	[ ]
τ_{ij}	Viscous stress tensor	[N/m^2]
$\dot{\omega}$	Source term	[…]

Roman Symbols

Symbol	Description	Units
A	Cross-sectional area	[m^2]
c_p	Specific heat	[J/(kg·K)]
C_μ	Turbulence model constant	[ ]
C_1	Turbulence model constant	[ ]
C_2	Turbulence model constant	[ ]
$\langle C \rangle_A$	Cross-sectional average of concentration	[mol/m^2]
C_i	Concentration at a given measurement point	[mol/m^2]
Co	Courant number	[ ]
D_1	Diameter of helium gas inlet	[m]
D_2	Inner diameter of pipe	[m]

D_k	Specie diffusivity	[m^2/s]
F	Body force	[N]
h	Specific enthalpy	[J/kg]
I	Turbulence intensity	[ ]
k	Turbulent kinetic energy	[m^2/s^2]
K	Eddy diffusivity	[m^2/s]
l	Turbulence length scale	[m]
L	Length of pipe	[m]
n	Number of moles	[mole]
N	Number of measurement points	[ ]
p	Pressure	[N/m^2]
P_k	Production of turbulent kinetic energy	[kg/(m·s^2)]
Pr	Prandtl number	[ ]
$\dot{Q}$	Volumetric flow rate	[m^3/s]
r	Velocity ratio, nitrogen inlet velocity over helium inlet velocity	[ ]
R	Universal gas constant	[J/(mol·K)]
Re	Reynolds number	[ ]
Re_y	Reynolds number function of wall distance y	[ ]
S_{rad}	Source term for radiation	[J/(m^3·s)]
Sc	Schmidt number	[ ]
Sc_t	Turbulent Schmidt number	[ ]
t	Time	[s]
T	Temperature	[K]
u	Velocity in x-direction	[m/s]
u_{cell}	Local grid cell velocity	[m/s]
u_*	Friction velocity at nearest wall	[m/s]
u'	Fluctuating velocity in x-direction	[m/s]
U	Mean velocity	[m/s]

v	Velocity in y-direction	[m/s]
v'	Fluctuating velocity in y-direction	[m/s]
V	Volume	[m^3]
$\dot{V}$	Mean flow rate	[m^3/s]
w	Velocity in z-direction	[m/s]
w'	Fluctuating velocity in z-direction	[m/s]
x	Position	[m]
y	Wall distance	[m]
y^+	Non-dimensional wall distance	[ ]
Y_k	Variable containing specie properties	[…]
z	Axial distance in pipe	[ ]

Abbreviations

CFD	Computational Fluid Dynamics
CoV	Coefficient of Variance
DIC	Diagonal Incomplete-Cholesky
DILU	Diagonal Incomplete LU
OpenFOAM	Open-source Field Operation and Manipulation
PCG	Pre-conditioned Conjugate Gradient
PBiCG	Pre-conditioned BI-Conjugate Gradient
P&ID	Piping and Instrumentation Drawing
PIMPLE	PISO-SIMPLE (Piso-sIMPLE)
PISO	Pressure Implicit Splitting of Operators
RANS	Reynolds Averaged Navier-Stokes
SIMPLE	Semi-Implicit Method for Pressure Linked Equations

1. Introduction

1.1 Motivation

Leaks are unwanted phenomenon in engineering designs. They can cause increase in power consumption, lack of efficiency and illegal distribution of pollutants into the atmosphere. Thus, leak detection is a process of high importance. It's often an obligatory part of the certification of products. There is a range of leak detection methods in today's market. One of the more sensitive practices utilizes a small fraction of helium gas in an inert environmental gas, often nitrogen. Helium gas is used due to its beneficial properties. The substance can diffuse into small leaks with ease, and can be effortlessly detected due to its distinguishable nature. Leak detection utilizing the given substance, offers insurance of safe operation, simulation of operation conditions, reduction of humidity, an inert ambient environment and reduces operational downtime due to its simple application.

Figure 1 Nitrogas N2Work nitrogen generator [1]

A company that goes under the name of Nitrogas AS has patented and created a nitrogen generator, figure 1. [1] The author of this thesis has been working on the construction of these prototypes. During the building process an idea of introducing a helium bottle into the system, came to mind. Continuous flow of helium and nitrogen, generated from a small system, could meet an unsaturated demand in the market. The substance is usually available in the form of pre-filled bottles. The idea enables on site generation of a gas blend of helium and nitrogen. Such a system provides the possibility of versatile and customizable leak detection methods. There is a possibility of increasing pressure levels up to 350bar and volumetric flow rate to 500l/min [1]. In the industry the substance has been christened HeliNite.

1.2 Objective

The goal in this thesis is to define a complete HeliNite leak detection system. Required components need to be picked and presented, required considerations need to be investigated and a walkthrough of the system needs to be provided. An essential area in the system is the gas mixing process which will be emphasized in this thesis. To ensure accuracy and repeatability of HeliNite leak detection, the blend of the two constituents needs to be uniform. The meaning of a uniform gas mix is an adequate dispersion of helium in the nitrogen gas. A thorough investigation of the gas mixing mechanisms needs to be completed. The software OpenFOAM (Open source Field Operation And Manipulation) is known to handle the prediction of gas blending [2]. Thus, OpenFOAM seems to be a sufficient analysis tool. If a Nitrogas HeliNite system meets a not yet saturated demand in the marked there is possibility for mass production. Consequently, the system needs to be described with as few components as possible, to ensure a low cost construction.

Due to not knowing the final dimensions of the not yet constructed system, approximate dimensions are used in the simulations. There is a possibility that the final product's dimensions aren't equal to the ones in this thesis, but they should be close.

1.2 Leak Detection

There are a few essential points to leak detection: Confirmation, location and quantification of a leak. Most of today's leak tests do not offer all three properties. Their function is often only to validate that a leak is present.

1.2.1 Types of Leaks

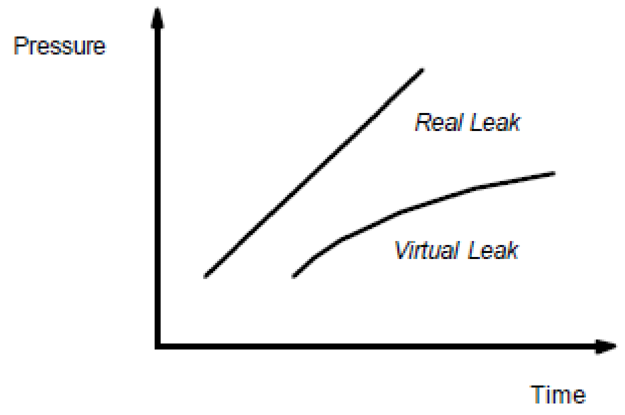

Figure 2 Type of leak confirmation by pressure increases versus time [3]

Residual and Virtual Leak [3]

Leaks come in numerous forms and shapes. They can be described as imperfections, holes, cracks and insufficient seals. These are all real leaks. However, when helium is the tracer gas there is a possibility of a second kind of leak. Such leaks can be defined as virtual leaks, figure 2. Virtual leaks can be detected when small volumes of trapped gas are forcibly moved by pressurizing the test domain. The virtual leaks may cause erroneous detections.

1.2.3 HeliNite

HeliNite is, as previously mentioned, a substance used for leak detection. Specifically the mixture consists of two gazes - namely helium and nitrogen. The mixture's volumetric relationship is often 1%-99% respectively. Helium acts as the tracer gas. There are advantages when using helium as a leak confirming substance:

- Size: Helium diffuses into small leaks in a rapid manner. It's the lightest and smallest molecule of the noble gases. Helium is only beaten by hydrogen in size and lightness, but hydrogen gas can be flammable and reactive.
- Non-reactivity: Helium is an inert gas, the second least reactive noble gas only beaten by neon. It's non-toxic and inflammable.
- Distinguishability: There is a small concentration of the given gas in the atmosphere – around 5ppm. The mass properties of helium are unique. The gas can be detected with ease.
- Price: Helium gas is cheap.

However, there are some drawbacks:

- High sensitivity: There is such a thing as too much sensitivity that a leak can be confused with diffusion through solids. Thus, confirming a very small leak where there actually isn't one. The phenomenon is only worth considering if one is detecting for minor leaks, and if the leak detection process is quite time consuming [4].
- Asphyxiation: Helium and nitrogen can displace breathable air and cause choking.
- Identification confusion: Helium is as previously mentioned, easy to distinguish. It may be confused with deuterium gas (4.028g/mol), due to similar mass properties as helium gas (4.003g/mol). However, potential deuterium concentrations are unlikely.

There are several benefits with the use of nitrogen as cover gas:

- Stability: When in gas form nitrogen is stable. The triple bonds ensure the non-reactive nature. Nitrogen gas can be classified as inert.
- Compressibility: Gas is more stable than hydraulic fluid due to its property of compensating for leaks by volume expansions or contractions [5].
- Temperature independency: The compressibility ensures a stable environment with volume expansions or contractions due to temperature changes.
- Safety: Nitrogen gas produces no pollutants. Any exhaust may either be released into the atmosphere or extracted for recycling.

1.2.4 Today's HeliNite Functionality [6]

When performing a leak test appropriate test areas are often transitional regions such as flanges, manifolds and seals. The substance can be applied to entire pipe walls to ensure the tightness of the whole product. Whether the substance is applied on the inside or outside of the test domain depends on the test method.

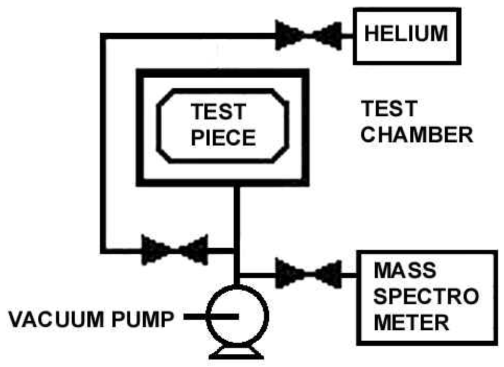

Figure 3 Standard helium leak detection system [7]

A common HeliNite leak test setup is displayed in figure 3.

Vacuum Test

A test object is evacuated with the use of a vacuum pump – ensuring a suitable vacuum pressure level inside the test object. To remove contaminations the test chamber and the test part is flushed with clean nitrogen, and yet again evacuated. Then the test chamber is filled with pressurized HeliNite resulting in a positive pressure difference between the test object and the chamber. A measurement of pressure levels should be completed to exclude the possibility of bigger leaks. If bigger leaks were present the pressure difference would be close to nothing. The favorable flow direction is outside-in. The helium mass spectrometer is connected to the domain delimited by the test piece. If any helium is detected a leak is present. As previously mentioned the concentration of helium is around 5ppm in the atmosphere. An increment of 5ppm in concentration of helium is usually the reference for a valid detection of a leak.

Pressure test

The pressure differs from the vacuum test in the area of having a negative pressure difference between the test object and the chamber. The test object is filled with pressurized HeliNite. The HeliNite substance flows in the inside-out direction. The tracer gas acts in the same way and diffuses into small leaks. The helium mass spectrometer is connected to the test chamber, and potential leaks can be detected. This test can be done without a test chamber. The detection process is in this case handled by a helium sniffer. This method enables the user to locate in addition to verify the leak.

2. Nitrogas HeliNite System Walkthrough

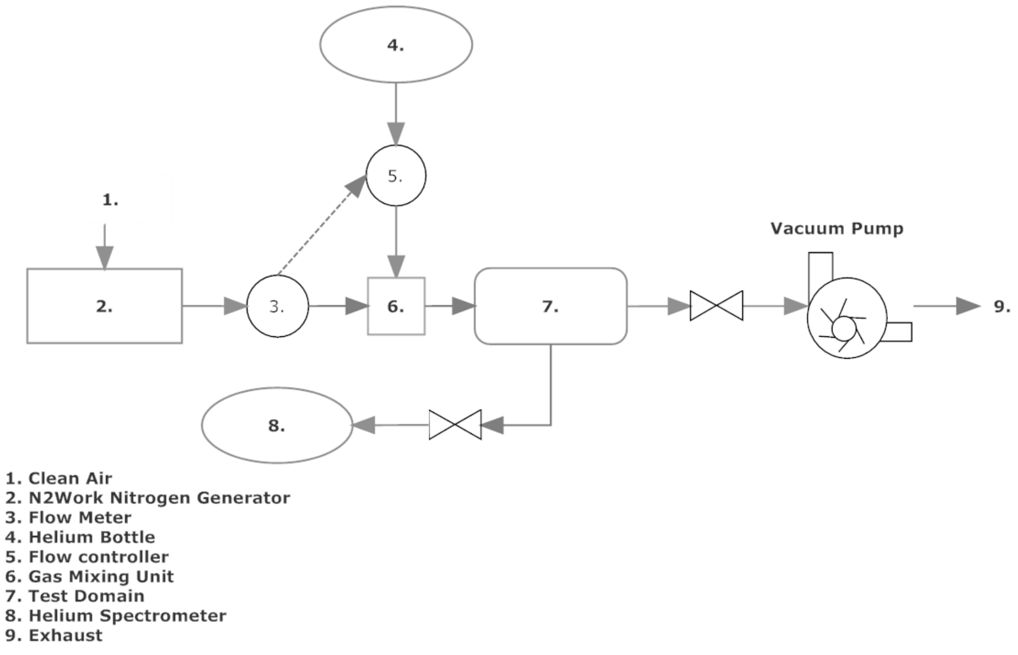

Figure 4 Piping and instrumentations diagram (P&ID) of Nitrogas HeliNite system

2.1 Components and Requirements

Numbering and explanation of components and requirements has figure 4 as reference.

1. Clean air: Proper production of nitrogen gas requires clean ambient air.
2. N2Work Nitrogen Generator: The generator ensures continuous production of high quality nitrogen gas (95%-99%, depending on the flow rate [1]). The only requirements are clean ambient air and power supply.
3. Flow meter: The instrument provides the measurement of the volumetric flow rate of nitrogen gas, and sends the information to the flow controller.
4. Helium bottle: A helium bottle offers a simple access to the gas helium.
5. Flow controller: The flow controller is needed due to fluctuations in the flow rate of nitrogen gas. The flow of nitrogen is measured and the flow controller makes continuous calculations based on information from the flow meter. The flow controller compensates the helium flow to ensure the volumetric flow relationship of 1% helim and 99% nitrogen.
6. Gas mixing unit: The unit creates insurance of a uniform gas blend between the two constituents.
7. Test domain: The box creates a safe environment for the test object. It provides the possibility of performing the vacuum and pressure leak test. The test chamber or the inside domain of the test object, can be filled with pressurized HeliNite. Consequently, allowing the vacuum or the pressure test.
8. Helium Spectrometer: The instrument allows the detection of leaks by proving increments of at least 5ppm in helium concentration. Helium spectrometers are highly sensitive to leaks. Standard equipment tests discontinuities with leak rates around $1*10^{-9}$ ml/s [8], or one thousandth of a liter per 30 years.
9. Exhaust: The exhaust can either be released into the atmosphere or collected for recycling. The substance produces no pollutants.

10. Mechanical pump: The component is used for evacuation, ensuring vacuum levels of pressure. It’s needed for leak tests at pressure levels similar to atmospheric pressure.
11. Connectors: Connectors ensures necessary transitions between components and piping.
12. Pipes/tubes: The pipes/tubes enable flow between components.
13. Valves: Valves are needed for appropriate stages – i.e. evacuation, testing, venting etc.

Note: The connector, pipes and valves need leak tightness higher than the threshold for the test.

2.2 Nitrogas HeliNite Leak Testing Guide

2.2.1 Key points pre-leak detection [7]

- Analysis of possible leak size (Applicability of leak tests)
- Necessary pre-requisitions (Clean air and power supply)
- Investigation of test object (Accessibility, trapped gas, cleanliness, surface finishing etc.)
- Determination of leak detection method (Vacuum or pressure test)
- Environmental influences (Temperature, cleanliness, humidity etc.)

2.2.2 Guide to Nitrogas HeliNite Leak Testing

1. Set up the Nitrogas HeliNite system
Construct the system according to figure 4.

2. Use reference leak to test detection equipment
After the system is up and running a reference test is required. A test valve is mounted to the chamber containing the HeliNite substance. The test valve has a known flow rate and is used to double check if the helium mass spectrometer predicts the value correctly.

3. Determine which type of leak test is the most appropriate
Depending on the accessibility of the test object the user should determine which leak test is the most suitable. Outside-in leak tests are the most appropriate for underwater housings and other parts with a vacuum nature. The inside-out leak test is often used for larger test objects.

4. Flush system with clean nitrogen gas
The inside domain of the part and the test chamber needs to be free of contaminants. Thus, a good test environment is created. The nitrogen needs to be evacuated before filling the test domain with HeliNite.

5. Fill test domain with pressurized HeliNite
Fill the system with the HeliNite substance at operating pressure to simulate working conditions.

6. Test for leaks
Operate the helium spectrometer to confirm potential leaks.

3. Theory

3.1 Governing Equations

3.1.1 Mass conservation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0 \tag{1}$$

The equation has the general unsteady and compressible form. Tensor notation is used $\mathrm{x_i} = \mathrm{x, y, z}$ and $u_i = u, v, w$ for $\mathrm{i} = 1,2,3$. ρ is density of mass, t is time, x is position and u is velocity.

3.1.2 Momentum Equations

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_i}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_i} + F_i \tag{2}$$

where p is pressure, F_i is body forces including gravity and τ_{ij} is the viscous stress tensor given by:

$$\tau_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k}) \tag{3}$$

Index k represents property ownership for species number k, μ is dynamic viscosity and δ_{ij} is the unit tensor.

3.1.3 Transport Equation

$$\frac{\partial}{\partial t}(\rho Y_k) + \frac{\partial}{\partial x_i}(\rho u_i Y_k) = \frac{\partial}{\partial x_i}\left(\rho D_k \frac{\partial Y_k}{\partial x_i}\right) + \dot{\omega}_k \tag{4}$$

where Y_k is a variable that represents a specie property, D_k is the diffusion coefficient for each specie and $\dot{\omega}_k$ is the source term. The first part is the transient term, followed by the convection term, preceded by the diffusion term and finally the source term.

3.1.4 Energy Equation

$$\frac{\partial}{\partial t}(\rho h) + \frac{\partial}{\partial x_i}(\rho u_i h) = \frac{\partial}{\partial x_i}\left[\frac{\mu}{\sigma_h}\frac{\partial h}{\partial x_i} + \mu\left(\frac{1}{Sc_k} - \frac{1}{\sigma_h}\right)\sum_{k=1}^{N} h_k \frac{\partial Y_k}{\partial x_i}\right] + \frac{\partial \rho}{\partial t} + S_{rad} \tag{5}$$

where h is enthalpy, σ_h is the mixture's Prandtl number, S_{c_k} is the species' Schmidt number and S_{rad} is the source term for radiation.

Note: The following assumption is often used in simulations $D_k = D$. The assumption is valid if the diffusion of the species is dominated by turbulent diffusion.

3.1.5 Ideal Gas Law

$$pV = \mathrm{n}RT \tag{6}$$

where V is volume, n is number of moles, R is the universal gas constant and T is temperature.

3.2 Turbulence modeling

3.2.1 Reynolds-Averaged Navier-Stokes (RANS) Equations

Time-averaged momentum equation in the x-direction:

$$\frac{\partial U}{\partial t} + div(Uu_i) = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\, div\big(grad(U)\big) + \frac{1}{\rho}\left[\frac{\partial(-\rho\overline{u'^2})}{\partial x} + \frac{\partial(-\rho\overline{u'v'})}{\partial y} + \frac{\partial(-\rho\overline{u'w'})}{\partial z}\right] \tag{7}$$

similarly in y- and z-direction. Velocity is defined as the sum of a mean and a fluctuating velocity component: $u = U + u'$.

3.2.2 Standard k-epsilon [9]

A RAS turbulence model for compressible fluids with parameters k (turbulent kinetic energy) and ε (turbulent dissipation), has the following transport equations for the two parameters respectively:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \rho\varepsilon \tag{8}$$

$$\frac{\partial}{\partial t}(\rho\varepsilon) + \frac{\partial}{\partial x_i}(\rho\varepsilon u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial\varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}P_k - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} \tag{9}$$

where $\mu_t = \rho C_\mu k^2/\varepsilon$ is the eddy viscosity and C_μ is a dimensionless constant. $P_k = \mu_t 2S_{\text{ij}}S_{\text{ij}}$ is the production of k and $S_{\text{ij}} = \frac{1}{2}\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_i}\right)$ is the mean rate-of-strain tensor.

3.3 Non-Dimensional Numbers

3.3.1 Courant Number [10]

The Courant number is a non-dimensional number representing the movement in the grid for each time step. A $Co \leq 1$ is a necessity for the stability of a solution, but it's often required to have an even lower magnitude than 1. If $Co \leq 1$ the maximum distance of fluid particles is from one cell to another in one time step. Consequently, if $Co > 1$ fluid particles move through two cells or more in one increment of time and is not stable.

$$Co = \left|\frac{u_{cell} \cdot \Delta\text{t}}{\Delta x}\right| \tag{10}$$

where u_{cell} is the average velocity in grid cell, Δt the maximum time step and Δx the size of the local grid cell

3.3.2 Reynolds Number

The Reynolds number offers a measure of turbulence. It represents the ratio of inertial to viscous forces.

$$Re = \frac{UL}{\nu} \tag{11}$$

where ν is the kinematic viscosity, L is a characteristic length (equal to the diameter in circular pipes) and U is the mean velocity.

3.3.3 Schmidt Number

A third dimensionless number worth introducing in this study is the Schmidt number. It quantifies the relationship between viscous and molecular diffusion rate.

$$Sc = \frac{\nu}{D} \tag{12}$$

$$Sc_t = \frac{\nu_t}{K} \tag{13}$$

where ν_t is the eddy viscosity and K is the eddy diffusivity. Note: A unity Schmidt number implies $\nu = D$, and in turbulence $\nu_t = K$.

3.4 Turbulence

Lewis Fry Richardson, 1922:

"Big whirls have little whirls that feed on their velocity. Little whirls have lesser whirls, and so on to viscosity."

This verse is not only written for poetic purposes, it also describes the behavior of the energy cascade in turbulence. Why this chapter begins with quoting a well know mathematician describing the phenomenon of turbulence, is mainly due to its dominating role in the mixing process of fluids. The energy cascade brings together important mechanisms of blending. They can be found at the extremes of the cascade: Turbulent convection at the large scale and molecular dispersion at the small scale, bonded by turbulent diffusion.

3.4.1 Understanding the Concept of Turbulence [11]

The majority of flows can be described as turbulent. Thus, it's important to explain the behavior of this phenomenon. The behavior of turbulence can be described as complex. The given motion has been researched thoroughly through the years. Today's theory of turbulence is still principally undeveloped. Predictions of this complex flow must therefore rely on experiments and empirical correlations associated with the simulation situation.

In turbulent flows there are fluctuations in velocity, pressure, temperature and density, even when the average flow is steady. The fluctuations are random and can happen rapidly. In the CFD (computational fluid dynamics) world there exists another word for fluctuation in motion, namely eddies. Eddies can be found in regions where the flow is swirling. Such motion causes mass interchange in both the radial and axial direction in pipes.

3.4.2 Turbulent Scales [12]

At the large scale one can find the largest eddies. At the other end of the scale the smallest eddies are present. The smallest scales are known as Kolmogorov scales. The largest eddies ensures the biggest portion of the transport of momentum and energy. Their size is restricted by the outer boundaries of the geometry. Whereas the size of the smallest eddies are determined by viscosity. Turbulent kinetic energy is passed down, from the largest to the smallest scale, through the energy cascade.

3.4.3 Wall bounded turbulent flows [13]

For explanatory purposes a Reynolds number dependent on a wall distance y is defined:

$$Re_y = \frac{Uy}{\nu} \tag{14}$$

The magnitude of Re_y goes towards zero as the wall approaches. The definition of Re is given by inertial per viscous forces. Thus, one can conclude that while moving towards a solid surface viscous forces become increasingly dominant. One can argue that the thin layers connected to solid surfaces are independent on free stream parameters. To tackle the difference in fluid behavior a dimensionless parameter needs to be defined. Specifically a non-dimensional wall distance y^+. The parameter is defined as the distance from the wall to the closest cell's center point. The scalar is introduced by the following equation:

$$y^+ = \frac{u_* y}{\nu} \tag{15}$$

where u_* is the friction velcocity at the nearest wall, y is the distance to the nearest wall and ν the local kinematic viscosity.

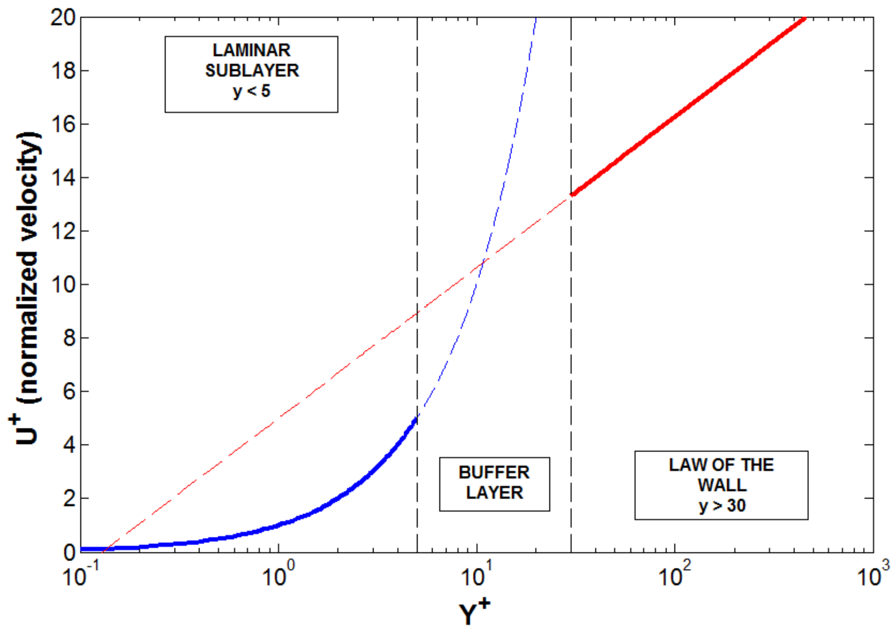

Figure 5 Normalized velocity versus normalized wall distance, distinguishing boundary layers [14]

The y^+-magnitude has an important application of determining the locations of different close to wall layers. Figure 5 shows a rough description of the different layers near solid surfaces (in reality there are more layers). The layers are:

1. Laminar or viscous sub-layer: In this layer the fluid is under strong influence of viscous forces. The fluid is in contact with the wall. The velocity profile is almost linear. There is an independency of free stream parameters.
2. Buffer layer: The region is a mix of the two neighbor layers. The magnitudes of turbulent and viscous stresses are similar.

3. Law of the wall layer: A boundary layer further from the wall. The layer assumes the average velocity is proportional to the logarithm of the wall distance.
4. Outer layer: This region is dominated by inertial forces. It's a turbulent region far from the wall.

The distinction of layers by y^+-values differs in various literatures. The definition of the location of the layers used in this thesis is:

1. Laminar or viscous sub-layer: $y^+ < 5$ [15]
2. Buffer layer: $5 < y^+ < 30$ [16]
3. Law of the wall layer: $30 < y^+ < 300$, preferably close to 30 [17]

In CFD-applications the different layers may be handled in the following manner [18]:

1. No wall functions needed if the mesh consists of values of $y^+ < 5$, preferably close to 1 for insurance of motion in the laminar sub-layer. If needed low-Re wall functions can be applicable.
2. Most turbulence models completely ignore the transitional area – namely the buffer layer. The simplification can be traced back to the difficulties of predicting the complex velocity behavior in this region.
3. The layer is handled by standard wall treatment.

3.5 Mixing Mechanisms

3.5.1 General Overview of Turbulent Mixing [19]

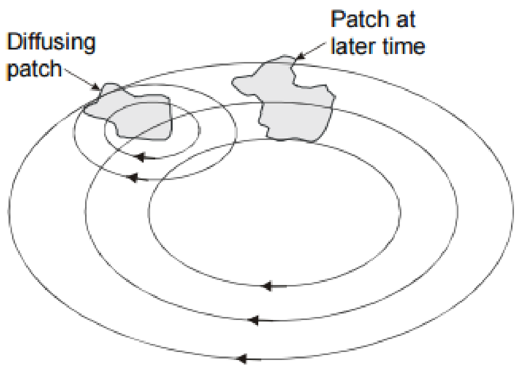

Figure 6 Illustration of mixing in turbulent flows [19]

Turbulent flows contain a broad array of length scales. Length scales classify eddy sizes. Consider the diffusing patch depicted in figure 6. The size of the diffusing patch is intermediate. Consequently one could argue that there exist both comparatively larger and smaller eddy sizes. The role of the smaller eddies is continuous deformation of the diffusing patch. The initial volume of the patch is distributed in a random manner. The distortion, stretching and convolution of the patch result in a larger volumetric region at a later time. The effect of the smaller eddies is steep concentration gradients. Consequently, the concentration of a species in turbulent flows will decrease over a given finite volume. The molecular diffusion, which will be described in more detail in consecutive sections, ensures smooth transitions of the concentration gradients.

3.5.2 Turbulent Diffusion [20]

In turbulent flows fluctuations in velocity are a common occurrence. Eddy motion causes turbulent mixing. This type of turbulent mixing goes under the name of turbulent diffusion. The diffusivity rate in these regions is local and thus differs from the bulk flow. The largest eddies contain the highest amount of turbulent kinetic energy. While moving down the energy cascade to the smaller scales, the turbulent kinetic energy content reduces with reduction in eddy size. The smaller eddies provide the largest concentration gradients, and the regions contain higher diffusivity.

3.5.3 Turbulent Convection [21]

In high Reynolds number flows viscous forces are small compared to inertial forces. Turbulence causes transfer of momentum. The random fluctuations are usually the major cause for mixing. In pipes such an occurrence leads to a movement of flow from the center of the pipe to the inner walls. A notable difference from laminar to turbulent pipe flow is the bigger region in the center of the pipe with more or less constant velocity. Turbulent convection is a bulk flow phenomenon.

3.5.4 Molecular Diffusion [21]

Molecular dispersion occurs at the molecular level. It’s essentially caused by regions with different levels of concentrations. The concentration gradients trigger a movement of molecules, from high to low concentration. Eventually the concentration levels will even out, and dynamic equilibrium is achieved. It’s called dynamic equilibrium due to a continued random motion of molecules. Such motion ensures a complete mix between two constituents. However, to achieve a complete mix by molecular diffusion alone, requires a substantially long flow length, and is quite time consuming.

Without molecular diffusion there cannot exist intermixing between two species. Thus, turbulent convection and diffusion cannot occur without molecular diffusion.

3.5.5 Coefficient of Variance [22]

The coefficient of variance (CoV) can represent quantification of the level of mixing. This thesis operates under the assumption that CoV < 0.05 [23] represents a uniform blend.

The coefficient of variance is defined by the following equation:

$$CoV = \frac{\sqrt{\frac{\sum_{i=1}^{N}(C_i - \langle C\rangle_A)^2}{N-1}}}{\langle C\rangle_A} \tag{16}$$

$\langle C\rangle_A$ is the cross sectional average of concentration for a given specie, C_i is the concentration at a measurement point and N is the number of measurement points.

4. Simulation

The simulations are run in the open source software called OpenFOAM.

4.1 OpenFOAM [24]

OpenFOAM is an open source coded CFD software, and is run in a Linux environment. OpenFOAM is a quite difficult CFD software. Since it’s an open source software, it offers the possibility of control over most simulation parameters. OpenFOAM delivers the tools to enable users to predict situations in a realistic manner.

4.1.1 Brief History

Outdated codes and lack of support associated with CFD software were several reasons for the motivation of creating the first version of OpenFOAM. A cooperation between Henry Weller and Hrvoje Jasak, the latter writing his PhD thesis (1996), ended up in the making of FOAM (1993).

At first FOAM was a closed source software. Its applications were mainly commercial. FOAM became an open source software in December 2004, hence the name change to OpenFOAM. Since the release date of OpenFOAM 1.0 there has been loads of updates. The latest version released, the version used in this thesis, is OpenFOAM 2.3. Today's applications of OpenFOAM cover a broad range of industries and institutions, from the usage in R&D teams to an integrated role in courses at universities.

4.1.2 Content

A rough description of OpenFOAM splits the software in two. The first part is a library. The second portion represents solvers and utilities. Solvers enable the user to solve specific situations using engineering mechanics. Utilities are pre-/post-processing tools such as meshing, visualization of data and others. OpenFOAM offers a wide range of solvers - Compressible and incompressible, transient and steady, even solvers that handle chemical reactions. The latter solver classifies the one used in this thesis.

4.1.3 Rough Case Structure

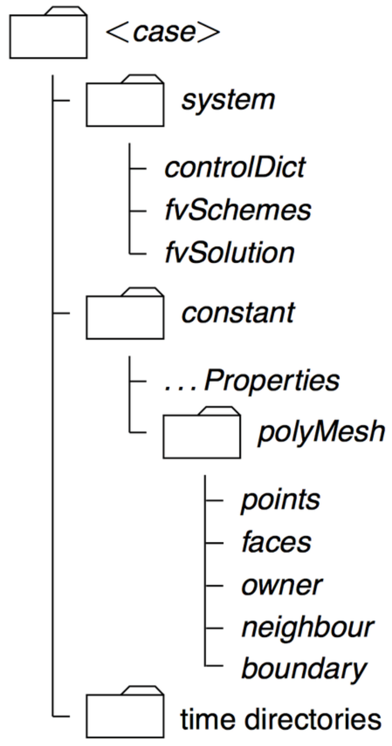

Figure 7 Basic File Structure in OpenFOAM [25]

Each case directory in OpenFOAM has a standard structure of three folders, namely `0`, `constant` and `system`, figure 7. The `0`-folder contains boundary conditions for appropriate field parameters, and is the first time directory. The `constant`-directory contains the `polyMesh`-folder along with appropriate files containing physical properties. The `polyMesh`-folder includes mesh description.The `system`-folder contains numerical control files, definitions of post-processing functions, discretisation schemes, solvers, definition of algorithms, and can include various dictionaries with different areas of application such as mesh tampering and parallel processing. Additional time directories are generated when running simulations. The folders contain data of the specific fields included in the case.

4.2 Geometry

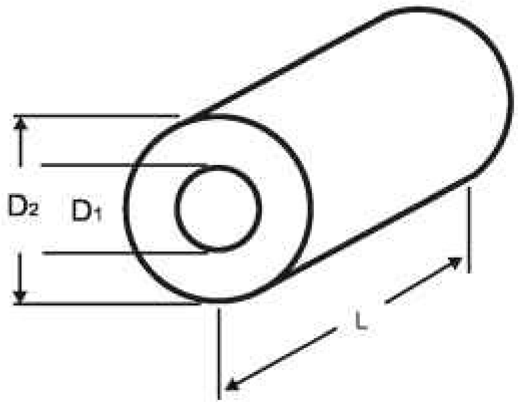

Figure 8 Complete geometry [26]

The geometry in this thesis is described by the use of a cylinder limited to a length of $\mathrm{L} = 0.5\mathrm{m}$, in the flow direction, figure 8. The entrance of helium and nitrogen occurs at one end, and the exit of the mix at the other end. The additional cases are all based on a standard geometry defined as case 1. The inner diameter delimits the inlet for helium gas, while the remaining cross sectional area constrains the inlet for nitrogen gas. Minor manipulations on the geometry distinguish the different cases. The changes are reduction, increase of inlet sizes and introduction of baffles. To summarize, D_1 varies while $D_2 = 0.018\mathrm{m}$ and $\mathrm{L} = 0.5\mathrm{m}$ is constant.

There are some simplifications of the geometry in the simulations. The helium inlet requires an additional pipe to penetrate the bulk pipe. Consequently, the helium pipe becomes an obstacle in the flow direction. This region is not included in the simulations.

4.2.1 Case 1 – Standard Geometry
The standard case is a pipe with inner diameter, $D_1 = 4\mathrm{mm}$.

4.2.2 Case 2 – Velocity Ratio Manipulation
The description of the manipulation of geometry has case 1 as reference.

Case 2A
A reduction of the inner diameter by 2mm, $D_1 = 2\mathrm{mm}$.

Case 2B
An increase of the inner diameter of 2mm, $D_1 = 6\mathrm{mm}$.

Case 2C
An increment of 4mm on the inner diameter, $D_1 = 8\mathrm{mm}$.

Case 2D
A raise of 6mm of the inner diameter, $D_1 = 10\mathrm{mm}$.

4.2.3 Case 3 – Introduction of Baffles
The pictures describing case 3's geometry are made with the use of axisymmetric simplification. Thus, they are only showing a 3 degree slice of the entire pipe. The simplification will be explained in more detail in the consecutive section.

Case 3A

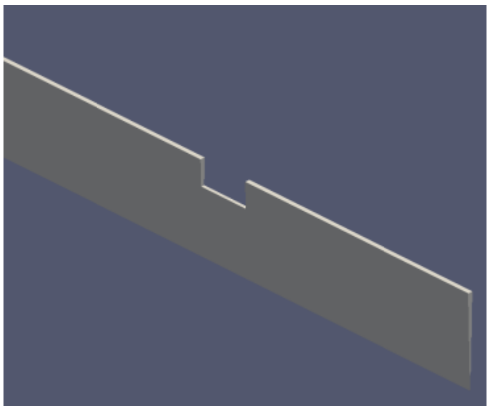

Figure 9 Geometry of one baffle

The introduction of one baffle is shown in figure 9. Its start point in the flow direction is at 25mm, and is 5mm long and 2.5mm tall. Case 3A has the same inlet sizes as case 1.

Case 3B

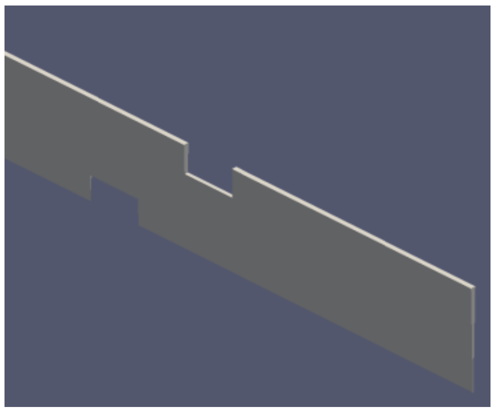

Figure 10 Geometry of two baffles

The introduction of two baffles is shown in figure 10. The first baffle's dimensions and location is identical to that of case 3A. The second baffle has the same dimensions but its location is in the middle of the pipe starting at 35mm downstream and ending at a length of 40mm in the flow direction. Case 3B has the same inlet sizes as case 1.

4.3 Mesh

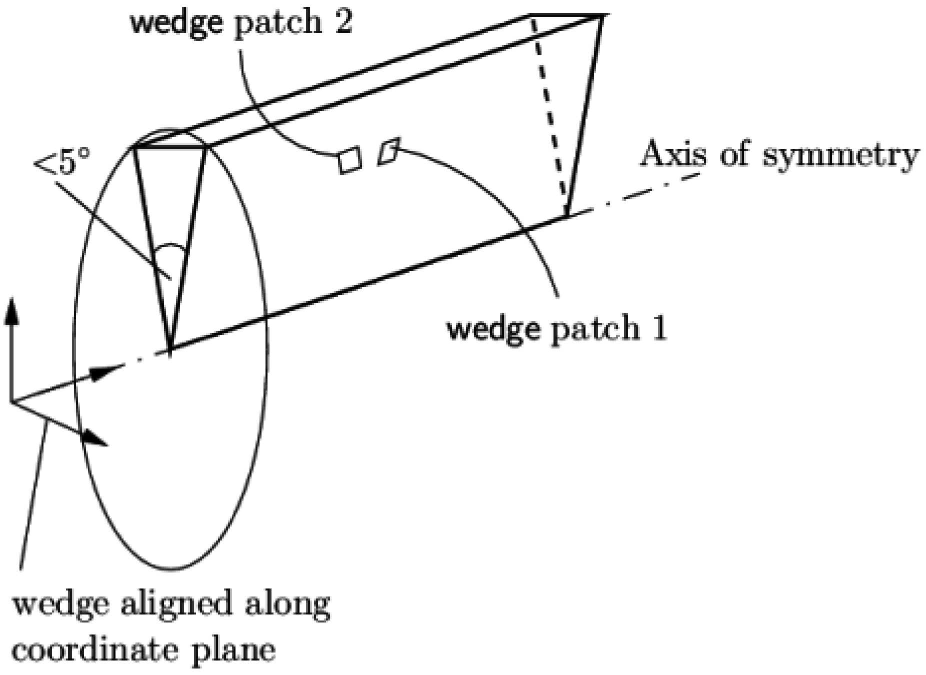

Figure 11 Axisymmetric simplification of a cylinder [27]

The geometry in this study is simplified by axisymmetry. Axisymmetry assumes no gradients in the theta direction (the rotational direction about the axis in the axial direction). The simplifications are made by the creation of a 3 degree wedge, and a thickness of 1 cell. The appearance of such geometry is depicted in figure 11.

Geometry code in OpenFOAM is defined in 3-dimensional space. Thus, both 1- and 2-dimensional geometries must be defined in the x-, y- and z-direction. The reduction in dimensions is achieved by 3-dimensional mesh definition and appropriate boundary conditions. Specifically, such boundaries are namely empty for 1- and 2-dimensional problems and wedge for axisymmetric cases. [28]

4.3.1 Mesh Generation

The standard meshing tool in the OpenFOAM environment is called `blockMesh`. It enables the user to create geometry by the definition of vertices, lines and blocks. The meshing utility is usually reserved for problems depicted by simple geometries - i.e. squares, triangles and cylinders. One can even combine such shapes to create a compound object. However, cylinders should be avoided when combining different shapes. This is due to the requirement of having common vertices ensuring equally sized blocks, at the faces of intersection. For more complex geometries such as cars and other objects with three dimensional curvatures, using `blockMesh` proves to be a time consuming endeavor, and in some cases impossible.

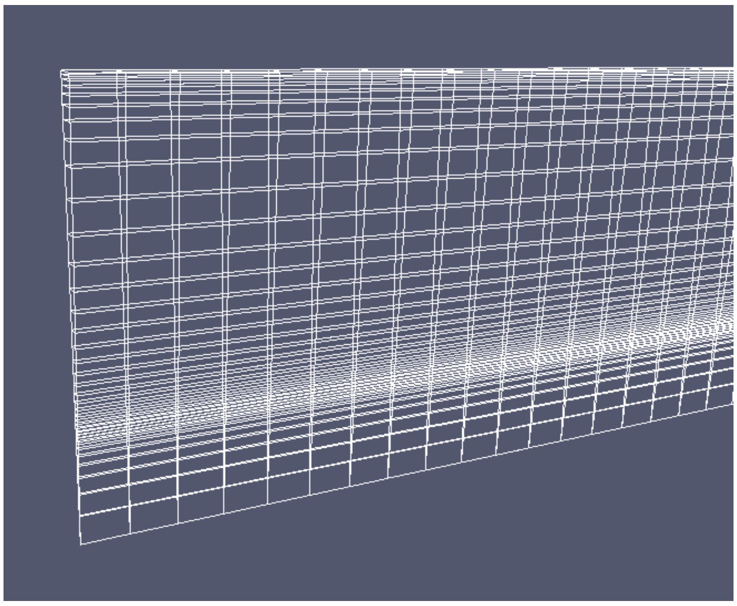

Figure 12 Depiction of the standard mesh

The creation of the wedge is done by the use of three simple shapes stacked on top of one another - a triangle at the bottom followed by two trapezoidal blocks. By describing geometry by multiple blocks the user gains versatility in the matter of mesh tampering. Three blocks enables triple grading. One can clearly see from figure 12, mesh grading in three directions – at the wall and the intertwining of the inlets.

The wedge is spanned out by 36 blocks in the radial direction, 500 blocks in the flow direction and a thickness of 1 block.

4.4 Boundary Conditions

There are two inlets and one outlet – one inlet for helium and one for nitrogen gas. These faces are defined as patches due to the entrance and exit of fluid. There are two faces defined as walls. The first one delimits the pipe in the radial direction, and the second one is a thin wall between the two inlets. The two last boundaries are wedges.

4.4.1 Inlets

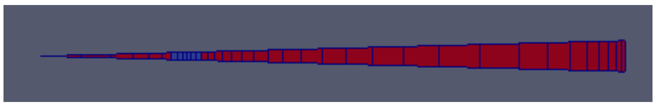

Figure 13 Compound Inlet

Figure 13 shows the inlet face which is spanned out by three boundary faces. They are separated by colors. The leftmost area denotes the inlet for helium gas while the rightmost blocks delimit the inlet patch for nitrogen gas. The dark blue blocks depict a thin wall between the two inlets. This compound inlet face is created by the use of the command `topoSet` followed by `createPatch –overwrite`. The commands are run after `blockMesh` has been run. The dictionaries `topoSetDict` and `createPatchDict` determining the compound inlet can be found in appendix D.

4.4.2 Outlet

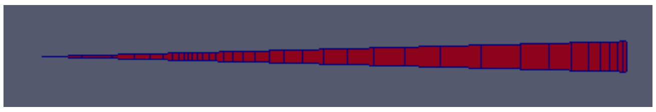

Figure 14 Outlet

The outlet is combined by the three previously mentioned shapes. The entire face depicted in figure 14 spans out the outlet.

4.4.3 Field Boundaries

The specification of field boundaries is needed in the `0`-folder. This section will systematically present the definition of such boundaries. Note: the wedge patches are always defined as wedges. Thus, they are excluded in the patch conditions table.

Table 1 Definition of types of patches

Name	Type
inletN2	Patch
inletHe	Patch
outlet	Patch
thinWall	Wall
wall	Wall
front	Wedge
back	Wedge

To summarize the types of patches, table 1 is created.

Table 2 Definition of patch conditions

	inletN2	inletHe	outlet	thinWall	Wall
epsilon	inletOutlet	inletOutlet	zeroGradient	Compressible:: epsilonWall Function	Compressible:: epsilonLowRe WallFunction
He	fixedValue	fixedValue	inletOutlet	zeroGradient	zeroGradient
k	inletOutlet	inletOutlet	zeroGradient	Compressible:: kqrWall Function	Compressible:: kLowReWall Function
N2	fixedValue	fixedValue	inletOutlet	zeroGradient	zeroGradient
p	zeroGradient	zeroGradient	outletInlet	zeroGradient	zeroGradient
T	fixedValue	fixedValue	inletOutlet	zeroGradient	zeroGradient
U	inletOutlet	inletOutlet	zeroGradient	fixedValue	fixedValue

The boundary conditions specified in table 2 apply to all cases. Initial values are not included due to different magnitudes across cases. The initial boundary values of helium gas concentration He, nitrogen gas concentration N2, pressure p and temperature T, does not vary from one case to another. The initial boundary values of turbulent dissipation epsilon, turbulent kinetic energy k and velocity U, respectively, do vary amongst cases.

4.4.4 Field Constants

Table 3 Presentation of input parameters

Symbol	Description	Magnitude	Units
D_2	Inner diameter of pipe	0.018	[m]
L	Length of pipe	0.5	[m]
p	Pressure	8	[bar]
$\dot{V}$	Mean flow rate (100%)	8.333E-03	[m^3/s]
$\dot{V}_{N_2}$	Flow rate nitrogen (99%)	8.250E-03	[m^3/s]
$\dot{V}_{He}$	Flow rate helium (1%)	8.333E-05	[m^3/s]
T	Temperature	353	[K]
μ_{N_2}	Dynamic viscosity nitrogen	1.93E-05	[kg/(m·s)]
μ_{He}	Dynamic viscosity helium	2.10E-05	[kg/(m·s)]
ν_{N_2}	Kinematic viscosity nitrogen	1.657E-05	[m^2/s]
ν_{He}	Kinematic viscosity helium	1.262E-04	[m^2/s]
ρ_{N_2}	Density of mass nitrogen	1.165	[kg/m^3]
ρ_{He}	Density of mass helium	0.1664	[kg/m^3]

There are several parameters which are input parameters for calculations of inlet conditions. The parameters are given in table 3 in alphabetical succession.

Note: All the simulations are conducted at the nitrogen generators maximum volumetric flow rate given by: $\dot{V} = 8.333 \cdot 10^{-3} m^3/s$.

Table 4 Magnitudes of various calculations and initial conditions

-	Case 2A	Case 1	Case 2B	Case 2C	Case 2D	-
D_1	**0.002m**	**0.004m**	**0.006m**	**0.008m**	**0.010m**	**Units**
A_{He}	3.142E-06	1.257E-05	2.827E-05	5.027E-05	7.854E-05	[m^2]
A_{N_2}	2.474E-04	2.348E-04	2.160E-04	1.909E-04	1.594E-04	[m^2]
A	2.545E-04	2.545E-04	2.545E-04	2.545E-04	2.545E-04	[m^2]
U_{He}	26.526	6.631	2.947	1.658	1.061	[m/s]
U_{N_2}	33.347	35.131	38.197	43.227	51.745	[m/s]
U	32.748	32.748	32.748	32.748	32.748	[m/s]
U_{N_2}/U_{He}	1.257	5.298	12.960	26.074	48.768	[ ]
Re_{He}	420.37	210.19	140.12	105.09	84.074	[ ]
Re_{N_2}	30193.50	27567.97	25362.54	23483.83	21864.26	[ ]
Re	35581.56	35581.56	35581.56	35581.56	35581.56	[ ]
l_{He}	0.000140	0.000280	0.000420	0.000560	0.000700	[m]

l_{N_2}	0.001050	0.000910	0.000770	0.000630	0.000490	[m]
l	0.001260	0.001260	0.001260	0.001260	0.001260	[m]
I_{He}	0.0752	0.0820	0.0863	0.0894	0.0919	[ ]
I_{N_2}	0.0441	0.0446	0.0450	0.0455	0.0459	[ ]
I	0.0432	0.0432	0.0432	0.0432	0.0432	[ ]
k_{He}	5.967	0.444	0.097	0.033	0.014	[m^2/s^2]
k_{N_2}	3.239	3.678	4.440	5.796	8.455	[m^2/s^2]
k	2.998	2.998	2.998	2.998	2.998	[m^2/s^2]
ε_{He}	17107.832	173.329	11.810	1.756	0.400	[m^2/s^3]
ε_{N_2}	912.394	1273.691	1996.227	3639.764	8244.916	[m^2/s^3]
ε	677.096	677.096	677.096	677.096	677.096	[m^2/s^3]

Table 4 presents the magnitudes of inlet conditions in the succession on which they should be calculated. Subscript He represents ownership of the patch inletHe, subscript N_2 affiliates initial values to the patch inletN2, and parameters with no subscripts belongs to the internal field. The field conditions vary for the parameters epsilon, k and U. The dictionaries in appendix A are updated using the correct values for the different cases.

4.5 Numerical method

4.5.1 `reactingFoam` [30]

The solver `reactingFoam` is an unsteady and uncoupled solver. It predicts the phenomenon of combustion with chemical reactions. The combustion process in `reactingFoam` handles the thermophysical modeling in simulations. The turbulent transport properties are calculated based on the thermophysical model. [31]

Due to the presence of two inert gases there aren't any reactions. Consequently, the chemistry part in `reactingFoam` is toggled off, while the combustion remains on.

The solver `reactingFoam` is known to handle the diffusion of gas [2]. It enables the definition of unmixed individual species.

Some of the assumptions in this thesis' setup of `reactingFoam` are:

- Unity Schmidt number: $Sc_t = 1$, eddy diffusivity is equal to eddy diffusivity [32]
- Simplified diffusion laws: Usually Fick's law. A suitable assumption when it comes to binary gas mixing [33]
- Equal C_p across species [34]
- Constant C_p: For dominance of nitrogen gas the assumption is valid. [35]

The solver is run by the command `reactingFoam | tee log`, while in the case folder. The two latter terms ensures the generation of a log-file.

4.5.2 Turbulence Model [36]

The turbulence is modeled with the standard k-ε model equations. One for turbulent kinetic energy k, and one for turbulent dissipation ε. The model calculates a single length scale. Based on velocity and length scales the turbulent viscosity can be calculated. The standard k-ε model applies wall treatment thus the buffer region is not resolved. The model offers comparatively higher convergence rates at the cost of low computational power, and provides reasonable predictions for most situations. It is widely used in combustion investigations [37]. However, it struggles in the prediction of more complex flows – i.e. swirling, rotating and separating flows. Its application is mainly for fully turbulent flows.

4.5.3 Thermophysical models

Table 5 Definition of thermophysical models

Entry	Thermophysical model	Description
type	hpsiThermo	Basic thermophysical model. Calculations are based on compressibility
mixture	reactingMixture	Combustion mixture using thermodynamics and reaction schemes
transport	const	Enthalpy h and entropy s calculated from constant specific heat c_p
thermo	hConst	Constant c_p and heat of fusion H_f
equationOfState	perfectGas	Ideal gas law
specie	Specie	Specie properties calculated from c_p, h and/or s
energy	sensibleEnthalpy	Split evaluation of enthalpy for easier treatment of energy changes

Table 5 offers a presentation of the selection of thermophysical models in the simulations. A brief explanation of their resulting effects is given.

4.5.4 Schemes

Table 6 Definition of numerical schemes

Numerical scheme	Entry	Terms	Set-up
ddtSchemes	default	All	Euler
gradSchemes	default	All	Gauss linear
divSchemes	default none	U	Gauss limitedLinearV 1
		Yi_h	Gauss limitedLinear 1
		K	Gauss limitedLinear 1
		p	Gauss limitedLinear 1
		epsilon	Gauss limitedLinear 1
		k	Gauss limitedLinear 1
		muEff*dev2(T(grad(U)))	Gauss linear
laplacianSchemes	default	All	Gauss linear orthogonal
interpolationSchemes	default	All	Gauss linear
snGradSchemes	default	All	Orthogonal
fluxRequired	default no	p	N/A

This section offers a brief explanation of keywords in table 6. The first order time scheme `ddtSchemes`, is a discretisation scheme for transient problems. The sub-dictionaries gradSchemes, divSchemes, laplacianSchemes, interpolationSchemes and snGradSchemes, are reserved for gradient, divergence, Laplacian, interpolations of values and surface normal gradient terms, respectively. The Euler Scheme is a first order, bounded and implicit scheme. The standard finite volume discretisation of Gaussian integration is applied by the use of the keyword Gauss. The interpolation schemes are specified due to the requirement of Gauss having values transferred from cell centers to face centers. The linear scheme is general linear interpolation. The limitedLinear scheme requires calculations to be based on the flux of the flow. The interpolations are completed by the method of limited linear differencing. The value of 1 in the setup usually gives the best condition for convergence, while values closer to zero improves accuracy. The limitedLinearV scheme is an improved scheme which takes the direction of the field into account. Orthogonal specification implies orthogonal mesh treatment. The fields which requires generation of flux is defined in the fluxRequired sub-ditionary. [38]

4.5.5 Solvers

Pre-conditioned conjugate gradient (PCG) and pre-conditioned bi-conjugate gradient (PBiCG) are iterative solvers of system of equations A·x = b. PCG requires the matrix A to be symmetric [39] while PBiCG is a solver for asymmetric matrices [40]. The tolerance of the solution is specified in the fvSolution-file.

The pre-conditioners diagonal incomplete-Cholesky (DIC) and the pre-conditioner diagonal incomplete LU (DILU) transform a given set of equations into a form which is more suitable for numerical solving methods. Their function is to reach the solution using fewer iterations. [41]

The pressure is solved by the use of the PCG solver, and the pre-conditioner DIC. The velocity, enthalpy, turbulent kinetic energy and turbulent dissipation are solved by the PBiCG solver, and the pre-conditioner DILU.

The density of mass is solved by the diagonal-solver. It is a diagonal solver for explicit systems.

The PIMPLE (PISO-SIMPLE) algorithm is a combination of the algorithms Semi-Implicit Method for Pressure Linked Equations (SIMPLE) and Pressure Implicit Splitting of Operators (PISO).

The SIMPLE algorithm solves steady-state problems. When running the solver the non-linear effects of velocity is given more weight than an accurate determination of the pressure field. Properties are under relaxed to ensure stability and the reach of convergence. [42]

The PISO algorithm solves transient and compressible simulations. There is a necessity of solving the velocity-pressure coupling in each time step. To handle the non-linear effects of velocity, small time steps are needed. PISO introduces the simulation time controlling Courant number. [43]

The PIMPLE algorithm consists of an outer PIMPLE loop and an inner PISO loop. The outer loop calculates the momentum equations and the total energy equation. The inner loop handles the pressure equations and provides velocity corrections. [44]

4.5.6 Numerical Control

The numerical settings require a thorough investigation, especially for transient problems. A selection of parameters given in the controlDict in Appendix D, are presented.

Some of the time control definition is done by the key words `startTime`, `endTime` and `deltaT`. The start and end time specifies the duration of the simulation. A rule of thumb for transient simulations is to ensure enough time for at least three passages of flow. The defined run time is 0.05s. Diffusion related simulations require small time steps. Trial and error has proven that the time steps need to be as low as $1 \cdot 10^{-6}$ s. The `adjustableTimestep` entry specified to `yes`, adjusts the time step where it's needed, and is controlled by the courant number. The `maxCo` entry is put 0.2 for case 1 and 2, and 0.1 for case 3, to ensure stability in the simulations.

By the use of `functions` in the `controldict`, `sampledSurface` is defined. It enables the user to define a plane, by a base point and its normal, on which calculations can be made. There are two operations defined in the `controlDict`, namely `areaAverage` and `CoV`. After the simulation, the data can be gathered and analyzed.

4.6 Experimental Design

The experiments will investigate the scale of mixing needed to achieve a sufficiently uniform blend between the two gases helium and nitrogen. The simulations are going to be run on geometry that fits the N2Work nitrogen generator's standard dimensions. Thus, one avoids the need for additional components, such as connectors, fittings and pipes. The goal is attempting to reach the requirement for a uniform mix, CoV < 0.05.

The investigation commences at an end which will most likely not fulfill the requirement. Further analysis is done by simulations on different velocity ratios r, followed by the introduction of baffles in the flow direction. The enhancement of the mixing mechanisms is the leading factor in the manipulation of initial conditions and geometry. Hopefully a uniform mix will be achieved.

4.6.1 Case 1 – Standard Case

Case 1 is a clean pipe. It's a starting point for the gas mixing analysis. The simulations will give an indication of the scale of mixing needed. Grid convergence and mesh quality tests will be done on case 1. Additional cases will be run on mesh resolutions determined by the clean pipe case.

4.6.2 Case 2 – Velocity Ratio Manipulation

Case 2 describes the first step on the road to a uniform gas mix. In case 2 the cross-sections of the inlets are both increased and reduced to obtain different inlet velocities. Higher velocities cause more turbulence which is essential in the gas blending process. The effects of the different velocity ratios r, on the level of mixing will be tested. One could argue that the simulations will result in an enhanced gas mix. An indication of further enhancement of mixing mechanics will be given.

To summarize case 2, and present their velocity ratios:

• Case 2A:	$r = 1.25$
• Case 1:	$r = 5.30$
• Case 2B:	$r = 12.96$
• Case 2C:	$r = 26.07$
• Case 2D:	$r = 48.77$

4.6.3 Case 3 – Introduction of Baffles

The introduction of turbulence inducing geometry in the form of baffles, defines case 3. Simulations are first run on a pipe with a single baffle, then two baffles. Hopefully obstructions in the flow direction will ensure a sufficient mix.

5. Results

The results chapter presents graphs and distribution plots. The graphs are made in Microsoft Excel. The data is collected from post-processing functions in OpenFOAM. The distribution plots are created in the post-processing tool paraView. Views of pipes are cut in half to increase resolution. There isn't much change in flow behavior in the last section.

5.1 Mesh

This section will present data and graphs determining the quality of the generated mesh.

5.1.1 Grid Convergence

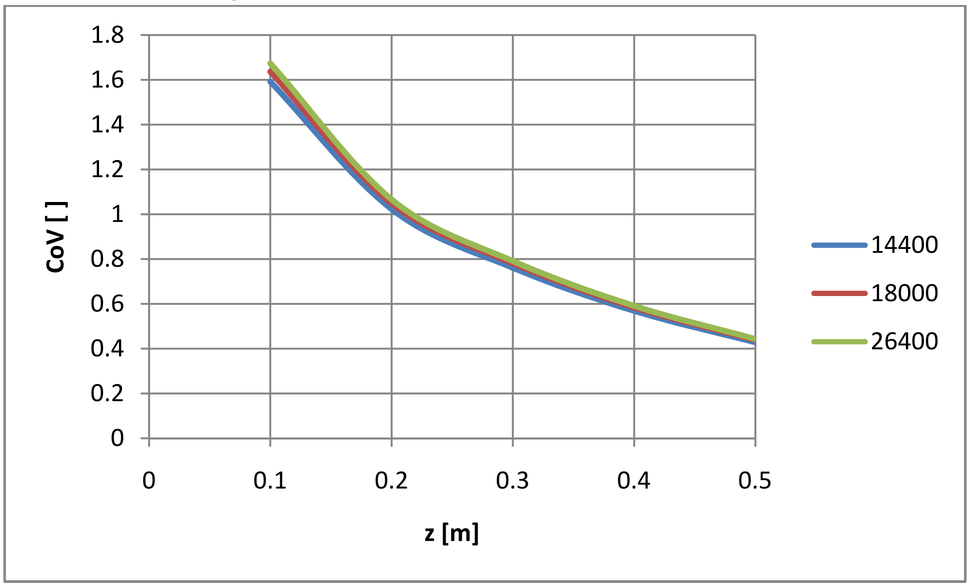

Figure 15 Graph of CoV versus axial distance z

To ensure the grid independence of a case study, a mesh convergence test is conducted. The interesting parameter in this thesis is the CoV. Thus this scalar is the controlling parameter in the test. Figure 15 shows three types of mesh split by the number of cells. One can notice small differences in the magnitude of CoV between the refinement levels.

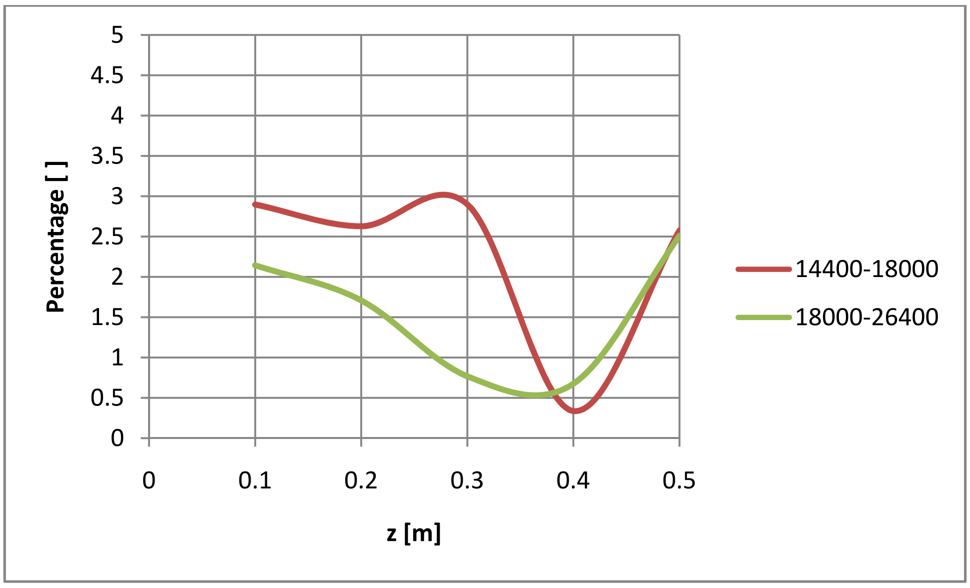

Figure 16 Graph of percentage difference in CoV versus axial distance z

The increase in number of cells for the 14400-18000 refinement is 25.0%, while the increase from 18000-26400 is 46.7%. The one cell thickness remained constant. The 14400-18000 refinement was accomplished by increasing the number of cells in the axial and radial direction by 10%. Similarly, the 18000-26400 refinement was completed by raising the amount of cells in the two directions, by a factor of 1.21 (two consecutive 10%-increases). Figure 16 shows the percentage difference in CoV for each mesh refinement. Upstream the difference is slightly above 2.0% for the last refinement. The difference shrinks with motion downstream, and rises at the last section of the pipe. The difference between these values at the 18000-26400 refinement is around 2.5% at the outlet. The 14400-18000 refinement shows slightly higher differences at major portions of the pipe.

5.1.2 y+-values

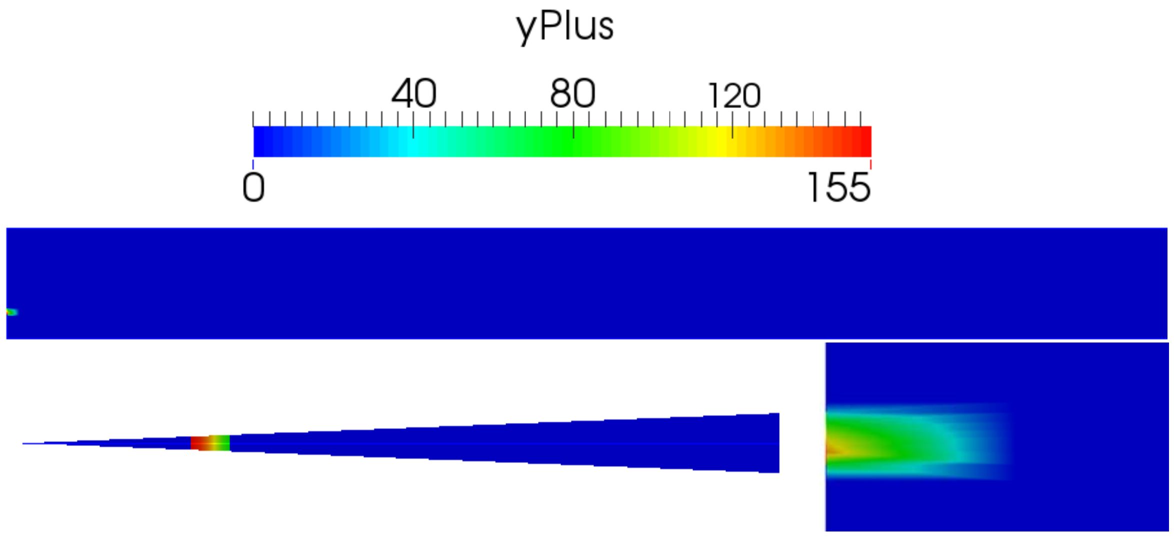

Figure 17 y^+-values plots on the standard grid

Figure 17 provides two views of the distribution of y^+-values – a side view of the pipe and of the cross section. At the bottom right corner is a zoomed in view of the side view. One can notice that the higher concentrations of y^+-values are located at the thin wall between the two inlets. Whereas the bulk region of the grid contains lower y^+-values.

By running the command `yPlusRAS` on the finished simulation of the standard grid, the following output is given in table 7:

Table 7 y^+-values for the patches defined as walls

Patch	y^+-min	y^+-max	y^+-avg
wall	2.135	2.902	2.204
thinWall	76.01	154.7	119.7

,

Running `yPlusRAS` on the other cases' mesh gives similar outputs.

5.2 Cases

5.2.1 Case 1 – Standard Case

Case 1 is, as mentioned in previous sections, a reference case. The meshes of the other cases are similar to case 1. Results associated with case 1 are presented along with case 2 due to reasons of comparison. A presentation and analysis of case 1's residuals is given in this section. The other cases' residuals have similar characteristics.

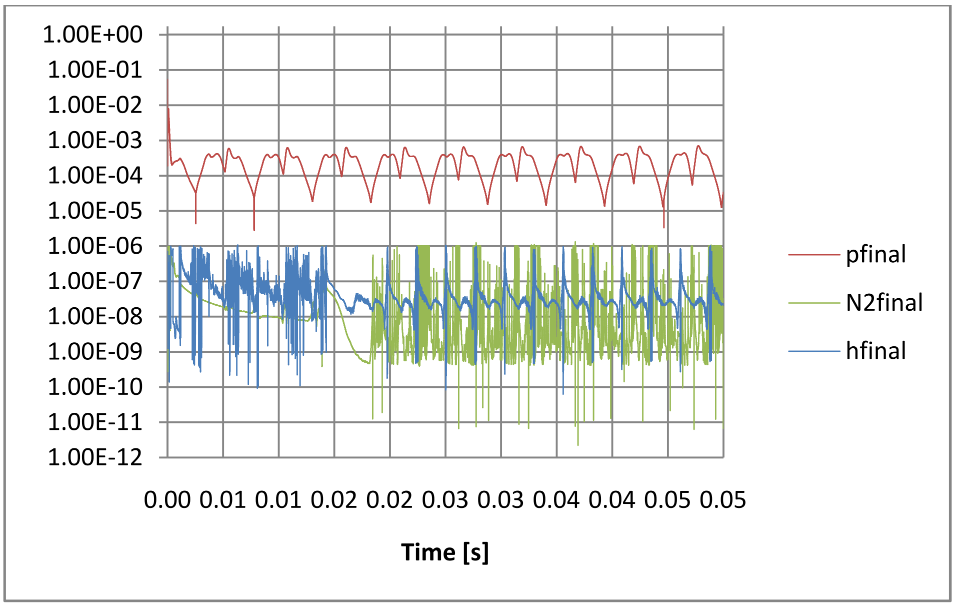

Figure 18 Graph of final residuals for selected parameters versus time

Figure 18 shows the final residuals. The residuals for enthalpy h, pressure p and nitrogen gas concentration N2 are included. The solution must converge at every time step for unsteady simulations, but not in the first time steps. The residuals seem to be sufficiently low, thus a converged solution is assumed.

5.2.2 Case 2 – Velocity Ratio Manipulation

This section will present CoV-curves, helium concentration plots, turbulent kinetic energy plots and turbulent viscosity plots for case 2. Velocity vector fields and temperature plots are included for a selection of cases.

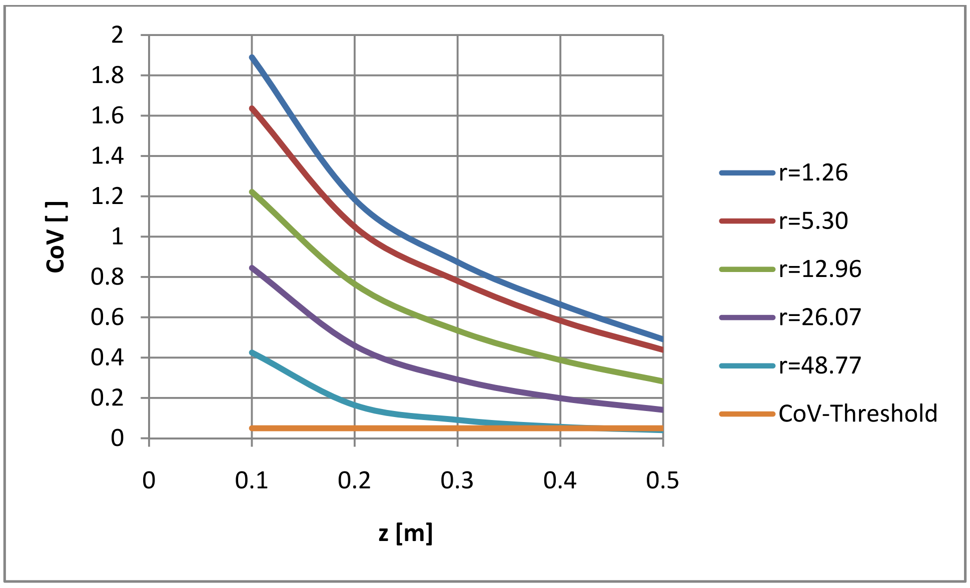

Figure 19 Plots of CoV at different velocity ratios versus axial distance z

Figure 19 presents the case 2 CoV-curves for different axial distances. Note, the high CoV values (>1) are caused by the occurrence of huge local concentrations of helium.

- Case 2A, r = 1.26: The CoV-curve is decreasing. The outlet magnitude has been reduced to CoV = 0.49.
- Case 1, r = 5.30: The velocity ratio r, has been increased by a factor of 4.21. The decrease in CoV is small. The outlet value is CoV = 0.44.
- Case 2B, r = 12.96: The velocity ratio has been raised by a factor of 2.45. The shrinkage in CoV is increasing with the velocity ratio. At the outlet the magnitude of CoV is 0.28.
- Case 2C, r = 26.07: The velocity ratio has been doubled. The CoV-curve has shifted downwards. The outlet value is CoV = 0.14.
- Case 2D, r = 48.77: The velocity ratio has been increased by a factor of 1.87. The outlet value of CoV is below the threshold of CoV = 0.05.

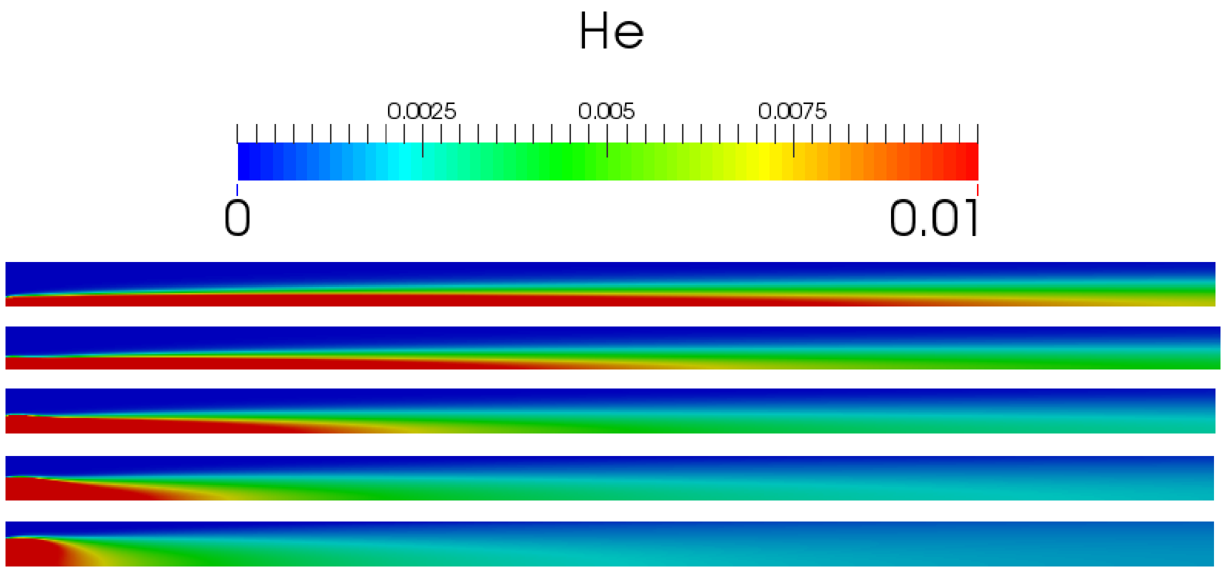

Figure 20 Helium concentration plots for increasing velocity ratios

Figure 20 shows the distribution of helium concentration for case 2. From top to bottom, case 2A, case 1, case 2B, case 2C and case 2D are displayed. At the top one can notice distinct layers of helium concentration. As the velocity ratios increase the length of the layer motion is decreasing.

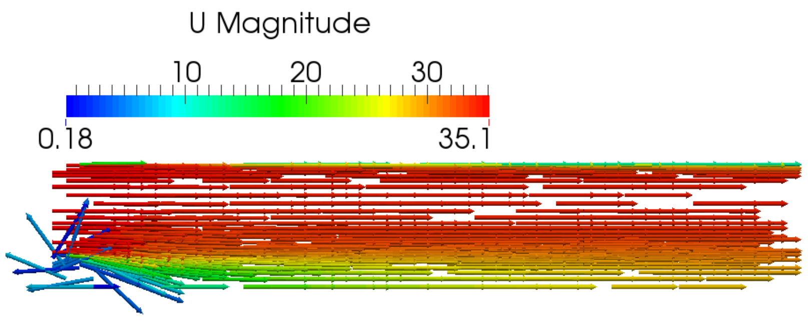

Figure 21 Velocity vector field plot, case 1

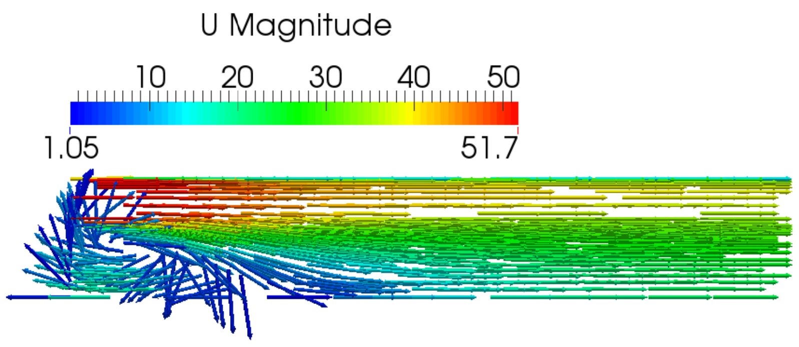

Figure 22 Velocity vector field plot, case 2D

By the comparison of figure 21 and 22 one can claim that there is an increase in swirling motion at the inlet.

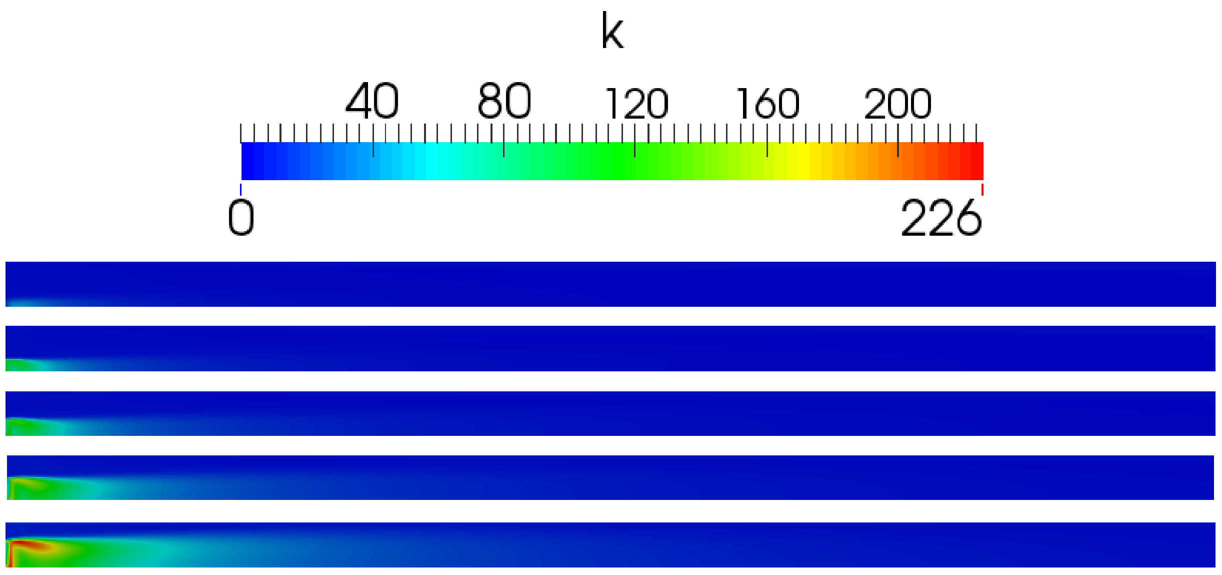

Figure 23 Turbulent kinetic energy plots for increasing velocity ratios

Figure 23 shows distributions of turbulent kinetic energy for case 2. From top to bottom, case 2A, case 1, case 2B, case 2C and case 2D are displayed. As the velocity ratios increase the amount of turbulent kinetic energy at the inlets rises.

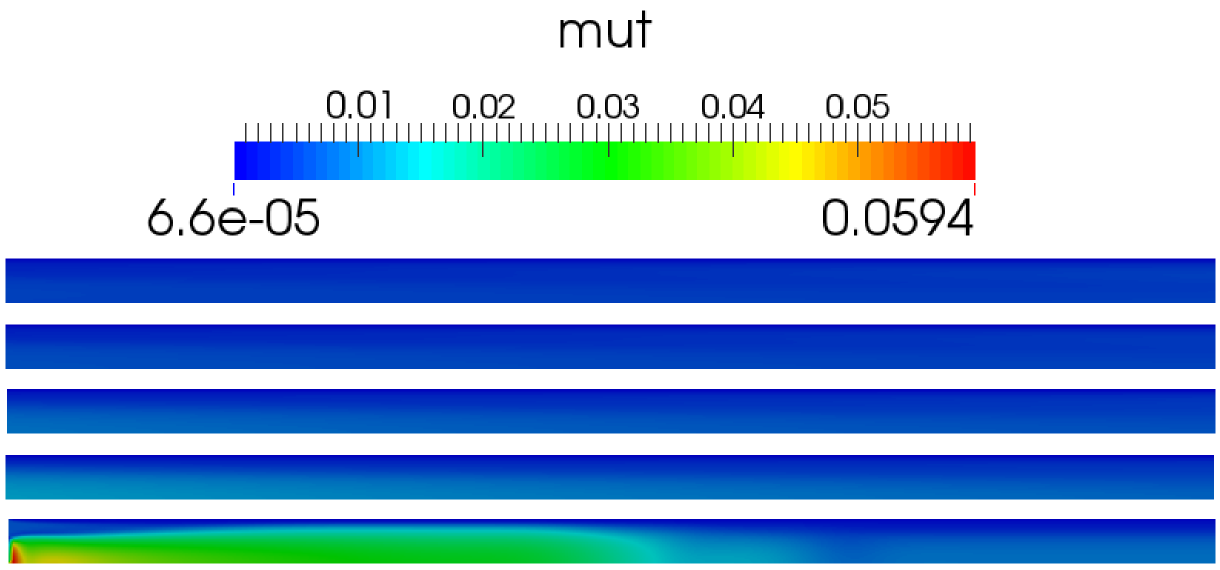

Figure 24 Turbulent viscosity for increasing velocity ratios

A unity Schmidt number states that turbulent viscosity equals diffusivity (3.3.3). Figure 24 displays increasing diffusivity with rising velocity ratios. From top to bottom, case 2A, case 1, case 2B, case 2C and case 2D are displayed.

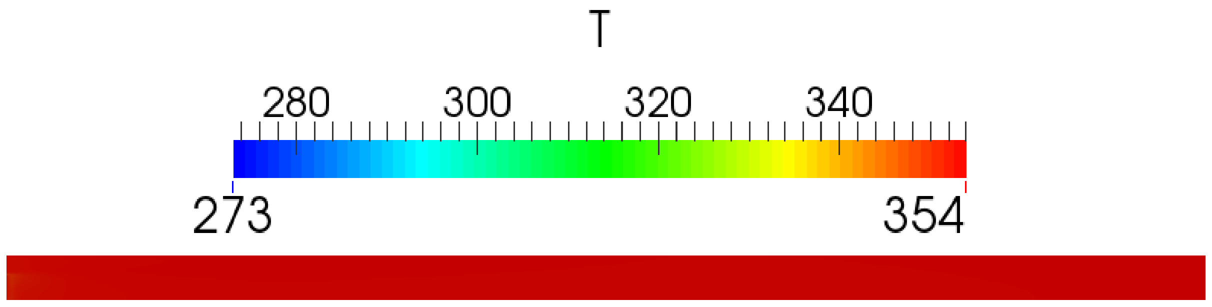

Figure 25 Temperature plot, case 2D

The temperature distribution for case 2D, shown in figure 25, is the only one included amongst the different velocity ratio cases. The other cases have less variation in temperature.

5.2.3 Case 3 – Introduction of Baffles

The presentation of the results for case 3 follows the same pattern as case 2.

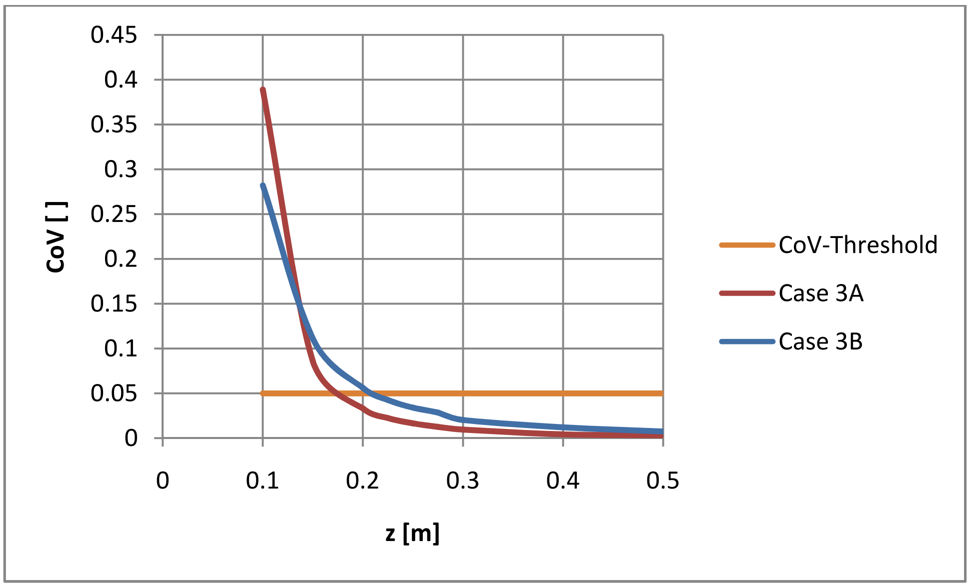

Figure 26 Graph of CoV versus axial distance z

Case 3A and 3B's CoV-curves are given in figure 26. They have a steeper character in the first portion of the pipe compared to case 2. Both curves end up below the CoV-threshold of 0.05. Case 3A reaches the threshold at an axial distance of ~0.17m, while case 3B reaches the limit at an axial distance slightly above 0.2m. A flattening character describes the curve when reaching the half point of the pipe. The outlet CoV-values are CoV = 0.03 for case 3A, and CoV = 0.07 for case 3B.

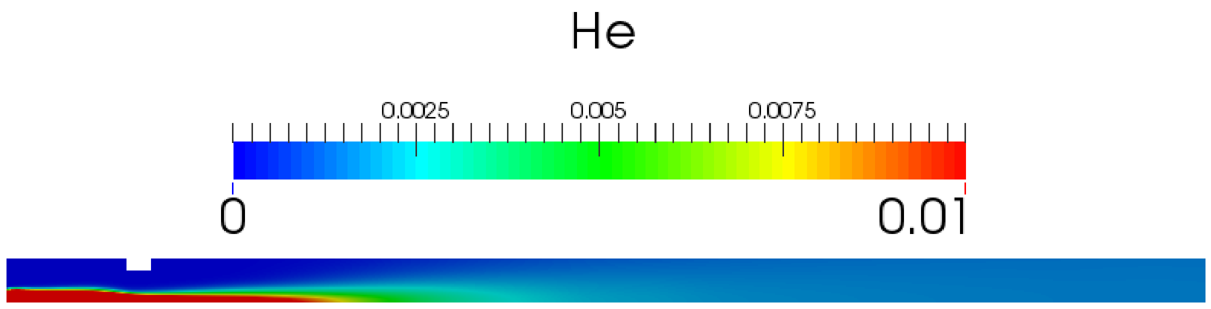

Figure 27 Helium concentration plot, case 3A

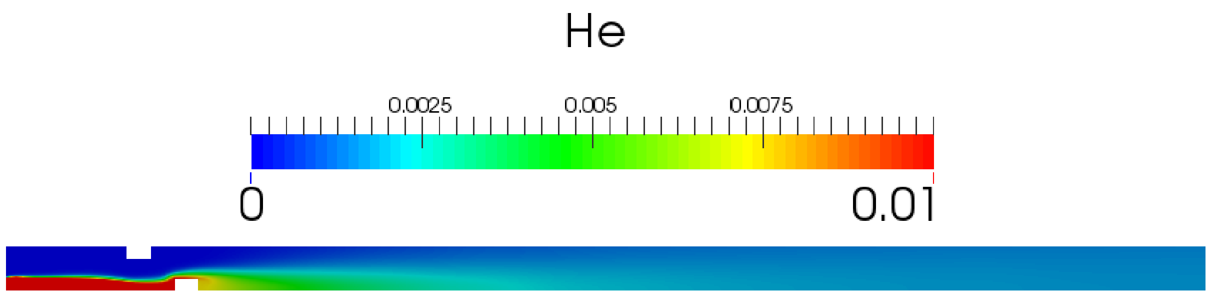

Figure 28 Helium concentration plot, case 3B

When comparing figure 27 and figure 28 one can notice that the end of the first figure is a shade darker blue. This observation corresponds well with figure 26. The CoV-graph for Case 3A does show a slightly lower magnitude of CoV compared to case 3B. The layer like motion can be noticed in figure 27 and 28.

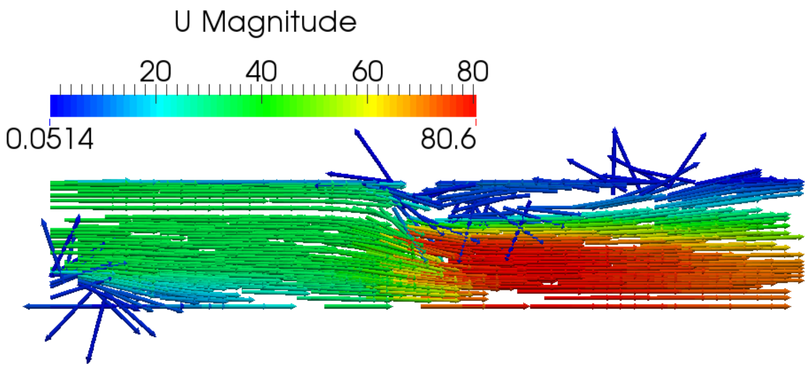

Figure 29 Velocity vector field plot, case 3A

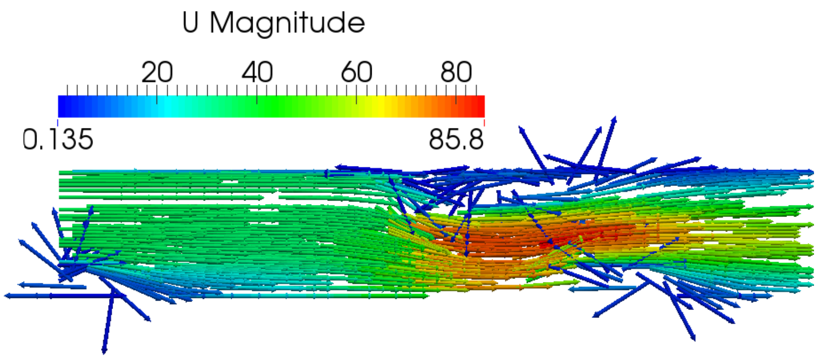

Figure 30 Velocity vector field plot, case 3B

Figure 29 and figure 30 are included to prove evidence of eddy motion. One can notice 2 fields of swirling motion in figure 29, and 3 regions of swirls in figure 30. There is one at the intertwining of the inlet flows and one at each baffle.

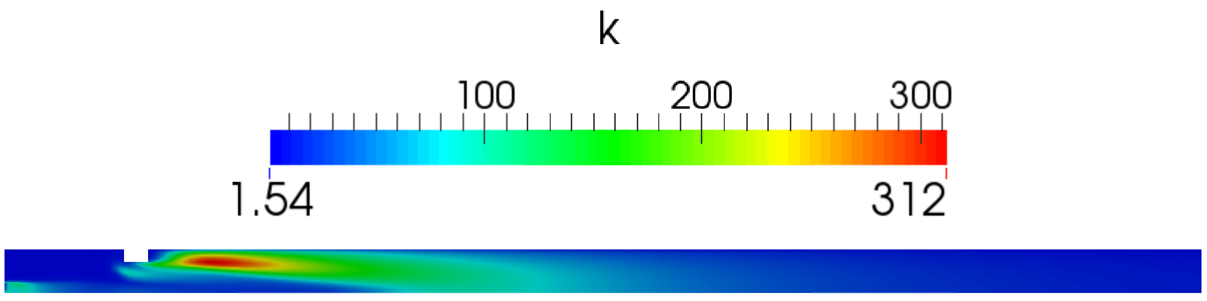

Figure 31 Turbulent kinetic energy plot, case 3A

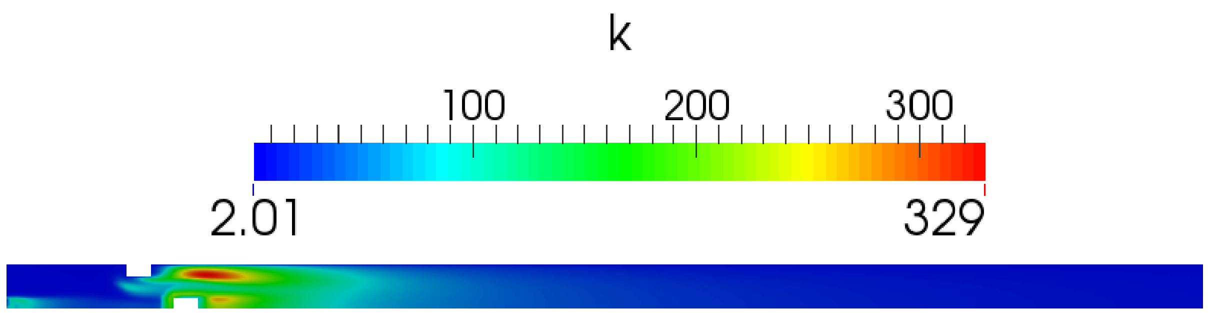

Figure 32 Turbulent kinetic energy plot, case 3B

The turbulent kinetic energy max value is larger in figure 32 than figure 31. However, the region of greater kinetic energy is larger for case 3A. Case 3B has three regions of higher energy, while Case 3A has two. This observation corresponds well with figures 29 and 30.

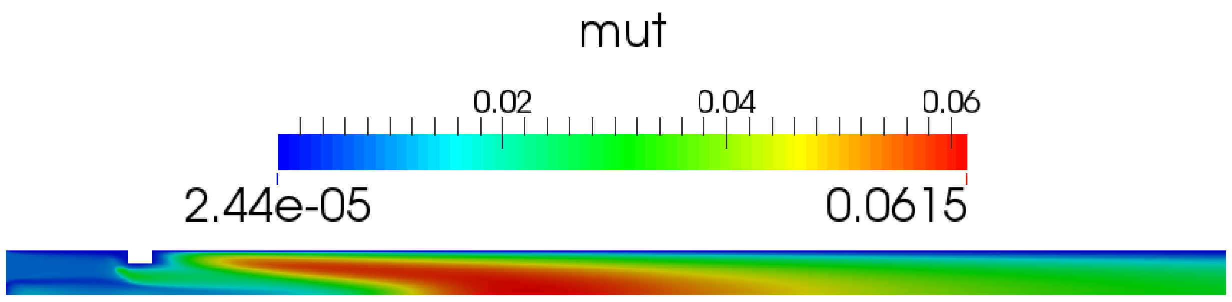

Figure 33 Turbulent viscosity plot, case 3A

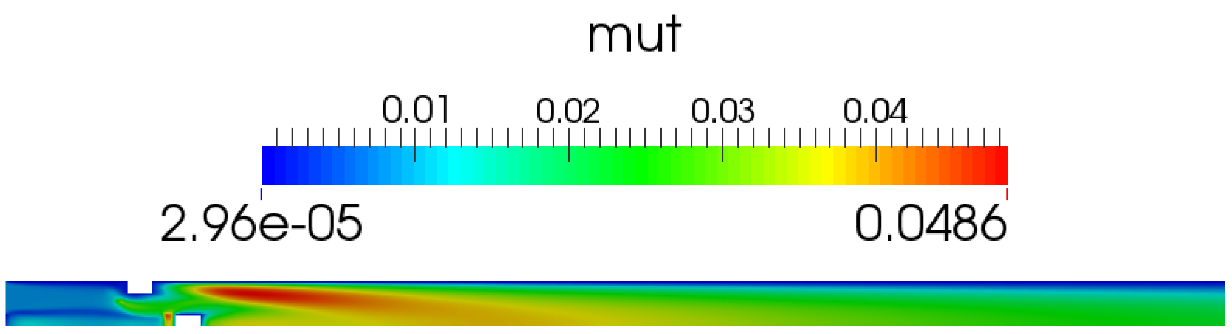

Figure 34 Turbulent viscosity plot, case 3B

The turbulent viscosity plots, figure 33 and 34, show the distribution of turbulent viscosity. There is a distinct difference in diffusivity. Case 3A has a greater region of high viscosity, in addition to a greater maximum value. The larger magnitudes of turbulent viscosity can be found in the wake of the swirls.

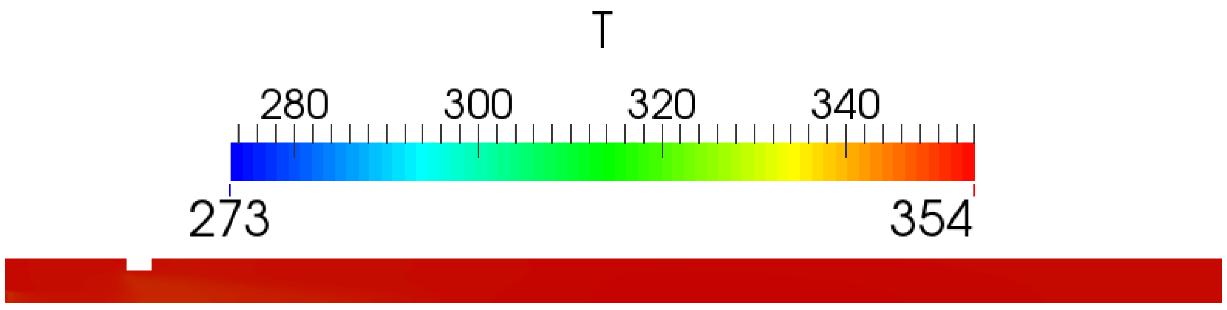

Figure 35 Distribution of temperature, case 3A

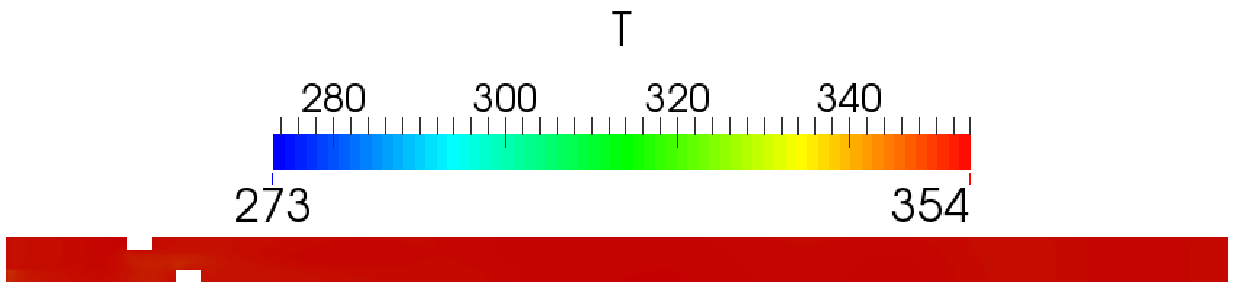

Figure 36 Distribution of temperature, case 3B

Figure 35 and 36 shows the variation in temperature for one and two baffles. One can sense shades of orange in regions with contractions in cross sections. The temperature variation is small. In both cases the temperature varies from 349K to 354K.

6. Analysis

6.1 Nitrogas HeliNite System

A presentation of required components, considerations and a walkthrough of the system has hopefully been given. The feasibility of the depicted system will be determined when prototype construction and testing commences. Time will tell if the proposition has been successful.

The author cannot guarantee that the system will be exactly as described. However, it should provide a solid foundation for the prototype construction. The only statement which can be made with certainty is that there will be changes to the setup. Which ones and their scale can only be determined when the construction process has commenced.

There are uncertainties when it comes to the coverage of the system. It might be lent out to leak detection firms. In this case the required components are most likely the N2Work nitrogen generator, flow meter, flow controller and the gas mixing unit. On the other hand, there is a possibility of Nitrogas AS conducting the entire leak detection process. Consequently, the entire system is needed. For the reason of not being able to see in to the future, one can only make a qualified guess of the coverage of the Nitrogas HeliNite system.

6.2 Gas Mixing

There were conducted simulations on 3 different cases. The results are analyzed in the following succession: CoV-curves, helium concentration plots, velocity vector field plots, turbulent kinetic energy plots and turbulent viscosity plots. The viscosity plots are probably the most important displays due to representing diffusivity (unity Schmidt number). At the end, there is an analysis of assumptions and possible errors.

Case 2

Case 2's CoV-curves, figure 19, shift downwards with increasing velocity ratios. Cased 2D is the only one below the CoV-threshold at the outlet. One could argue that the increasing velocity ratios have induced turbulence, and boosted the mixing process. Figure 20 shows the difference in concentration in a clean pipe. The plots show that the concentration gradients are increasing with rising velocity ratios. With the increasing velocity ratios one can notice a distinct reduction in the length of the layers of different concentration levels. A comparison of figure 21 and 22 displays an increase in fluctuating motion at the inlets. One can argue that this observation has resulted in the rise in concentration gradients. Theory state that eddies contain high turbulent kinetic energy (3.5.3). One can claim that the bottom pipes in figure 23 give evidence of the occurrence of such swirling motion. Figures 21, 22 and 23 correspond well in proving the eddy behavior. Figure 24 displays the increasing diffusivity for rising velocity ratios. One can assume that the greater eddy at the inlets has produced the increase the turbulent viscosity. The bottom pipe in figure 24 depicts a large region of higher diffusivity in the wake of the eddy. By relying on the stated theory (3.5.2) this is the region providing the majority of mixing.

Case 3

Case 3's CoV-curves, depicted in figure 26 are much steeper and initially lower, compared to case 2. The difference is the introduction of baffles. Thus, one could claim that these geometrical changes cause the extra mixing. Case 3B's CoV curve is initially lower than Case 3A's. However, the 1 baffle case's CoV-curve is steeper and catches up around at around 0.14m downstream. The plots of the concentration distribution of helium, figure 27 and 28, display a more rapid smoothing of concentration layers than case 2. By studying figure 29 one notices two regions which describe fluctuating motion. Figure 30 depicts three of these regions. One can claim that these swirling motions have caused the additional mixing. An interesting observation is that the outlet magnitude of CoV is lower for case 3A than 3B. An analysis of the turbulent kinetic energy and viscosity plots, figure 31 32, 33 and 34, has offered the following explanation: Case 3B has one additional eddy and a greater value of maximum kinetic energy. However, the turbulent kinetic energy plot of case 3A depicts a larger energy containing region. Theory states that smaller eddies have larger concentration gradients. The majority of the mixing process occurs in these local regions (3.5.3). The larger eddy region should generate more of the smaller eddies ensuring enhanced mixing mechanisms. Figure 33 and 34 validates that the majority of the mixing has occurred in the wake of the larger eddies, where the turbulent energy cascade has generated smaller eddies. Case 3A has a greater region of high turbulent viscosity and a greater maximum magnitude than Case 3B. Thus 3A's superior mixing seems plausible.

Sources of error can be traced back to the simplifications of the geometry, assumptions integrated in the solver `reactingFoam`, selection of thermophysical models, turbulence models etc.

The helium inlet in the middle of the pipe requires penetration of the larger pipe. Consequently the vertical section of the pipe acts as an obstruction in the flow direction. It ought to produce some extra turbulence. One could assume that the additional turbulence would boost the mixing the process.

The inlet velocities are defined as uniform. One could claim that the extra obstruction causes the inlet conditions to be erroneous. There is also some backwards facing motion at the inlets. The pre-inlet region is not included in the simulations, and could lead to falsely predictions.

Turbulent motion has a three dimensional character. Thus, the wedge simplification could produce some errors.

Some of the assumptions in the solver `reactingFoam` associated with gas mixing are the unity Schmidt number and Fick's law (4.5.1). As mentioned earlier, $Sc = 1$ conditions equality of turbulent viscosity and diffusivity. Some studies claim that there is strong dependency of the non-dimensional number, Sc [45]. By correctly adjusting the magnitude of the Schmidt number some models may give improved correspondence with experimental results. Fick's law is applied due to the complexity of handling multispecies diffusion. It's usually a valid approximation for a binary mixture. [46]

By studying figures 25, 35 and 36 one can argue that the constant specific heat assumption is valid. The temperature gradients are minor. The temperature changes are in the range of 5K. The major nitrogen content also validates the assumption [35].

The standard k-epsilon turbulence model calculates eddy viscosity based on a single length scale. In reality there is a range of length scales. Thus, the eddy viscosity prediction is simplified. The eddy viscosity is a leading factor in the gas mixing process. The k-epsilon model's prediction of this parameter might have an influential impact on the results. [47]

7. Conclusion

7.1 Nitrogas HeliNite System

A complete HeliNite system has been presented. The system's components and requirements have been defined, and their functionality described. Important considerations have been investigated. A guide to the Nitrogas HeliNite system has been given.

Components and requirements

1. Clean air: Optimal nitrogen gas generation.
2. N2Work nitrogen generator: Generation of nitrogen gas only need power supply and clean air.
3. Flow meter: Measures and restricts the flow of nitrogen gas.
4. Helium bottle: Access to helium gas.
5. Flow controller: Limits the helium gas flow. Receives signals from the flow meter.
6. Gas mixing unit: Ensures a uniform gas mix (CoV<0.05).
7. Test domain: Either the inside domain of the test object, or a boundary for the test object's outside region.
8. Helium spectrometer: Detection of increments in helium concentration.
9. Exhaust: Release or recycle of the substance HeliNite.
10. Vacuum pump: Evacuation of test domain.

Note: Numbering has figure 4 as reference.

Necessary pre-requisitions and considerations

- Initial guess of potential leak size
- Clean air and power supply
- Analysis of test object
- Determination of suitable leak detection method
- Investigation of environmental influences

Nitrogas HeliNite walkthrough

1. Construct the Nitrogas HeliNite system
2. Perform a reference leak test
3. Determine suitable leak test
4. Flush system with nitrogen gas
5. Pressurize test domain with HeliNite
6. Execute leak test

The major benefits by the use of a Nitrogas HeliNite system are versatility and customization. There is opportunity of changing operational parameters. The flow rate can be increased up to 500l/min, and the pressure level can be adjusted up to 350bar [1].

If more time was given, the results of the construction and the testing processes would be included in the thesis. However, the writing of the thesis occurred simultaneously with the construction of the N2Work nitrogen generator prototypes 4 and 5, and the N2Work high pressure compressors 1 and 2. The Nitrogas HeliNite system is next in line. Hopefully, in the foreseeable future, the system described in this thesis will be found in the leak detection market.

7.2 Gas Mixing

The comments of the results are given under the assumption of relying on a correct analysis.

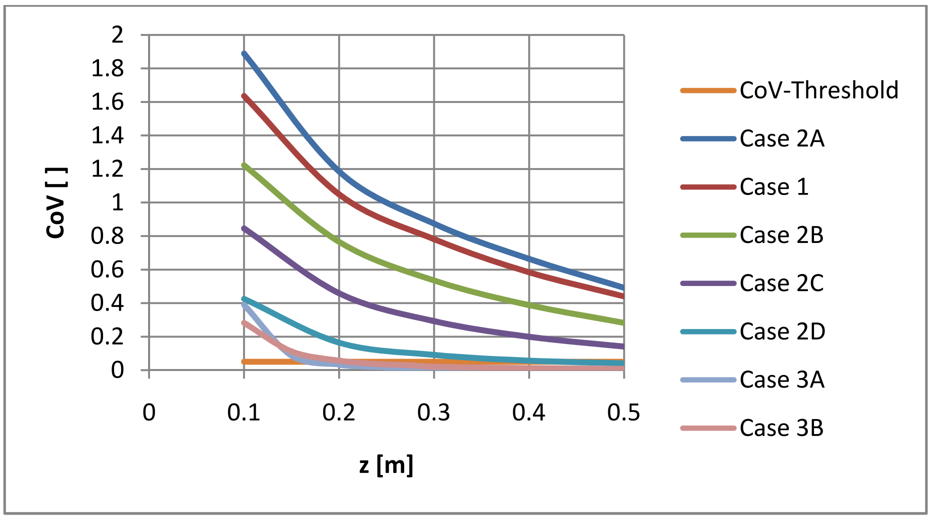

Figure 37 Graph of Cov versus axial distance z, all cases

Table 8 Summary of results

-	Case 2A	Case 1	Case 2B	Case 2C	Case 2D	Case 3A	Case 3B
$\boldsymbol{r}$	1.26	5.30	12.96	26.07	48.77	5.30	5.30
$\boldsymbol{CoV}$	0.491	0.439	0.282	0.141	0.040	0.003	0.007

Figure 37 and table 8 provides a summary of all the simulations.

A range of experiments using the software OpenFOAM has been completed. Blending in a clean pipe (case 2) reached a uniform mix for a velocity ratio $r = 48.8$. The introduction of 1 baffle gave a value of $CoV = 0.003$, while two baffles resulted in a magnitude of $CoV = 0.007$. Thus, Case 3A is the winner in the CoV-race. Case 2D and case 3 ensured mixing below the threshold of $CoV = 0.05$. The simulations have shown that an adequate mix between the two constituents helium and nitrogen gas, can be obtained either by a sufficiently high velocity ratio or minor manipulations of the geometry.

The blending process of two constituents is often done by the use of a jet. The jet would need to be designed such that the penetration of the mean flow stops at the center line of the pipe, ensuring symmetrically distribution of the tracer gas. Accomplishing the accuracy of hitting the center while being restricted by geometry and flow conditions, could prove to be a difficult endeavor.

There is also the possibility of the use of a static mixer. One could argue that the results of the experiments show that there isn't a need for the acquisition of a static gas mixer.

If time was not a factor, the size and location of the needed baffles could be investigated. The CoV-values are well below the threshold. One could decrease their size to come closer to the CoV-limit, and analyze the effects of changing their locations. One could tamper with the baffle shape. A ski jump, helical obstructions and cross-sectional grids are some of the geometries that could replace the simple square baffle. An analysis of the pre-inlet region should be considered. There is some backward motion at the inlets in which additional mixing could happen. There is also the penetration of the nitrogen pipe which will most likely alter the flow behavior.

8. References

[1] Nitrogas.no, 'Nitrogas - Complete nitrogen production', 2015. [Online]. Available: http://www.nitrogas.no.

[2] Cfd-online.com, 'Tracing a gas in a gas -- CFD Online Discussion Forums', 2015. [Online]. Available: http://www.cfd-online.com/Forums/openfoam-solving/59147-tracing-gas-gas.html.

[3] Stephen P. Hansen, 'Leaks: The good, the Bad and the Ugly', The Bell Jar, 1998. [Online]. Available:http://www.belljar.net/71leaks.pdf.

[4] S. User, 'Permeation', *Leakdetection-technology.com*, 2015. [Online]. Available: http://www.leakdetection-technology.com/science/the-flow-of-gases-in-leaks/permeation.

[5] Ralstoninst.com, 'The Benefits of Pressure Calibration with Nitrogen', 2015. [Online]. Available: http://www.ralstoninst.com/news/story/the-benefits-of-pressure-calibration-with-nitrogen/.

[6] HVS Leak Detection provides helium leak testing for vacuum, pressure & hermetic systems. We also rent, sell, rebuild and repair helium leak detectors., 'Detection Methods - HVS Leak Detection', 2015. [Online]. Available: http://www.heliumleakdetection.net/Helium-Leak-Testing/what-is-helium-leak-detection.html.

[7] Leaktesting.co.uk, 'Leak Testing Basic Principles - Background Information on Leaks and Leak Testing from TQC', 2015. [Online]. Available: http://www.leaktesting.co.uk/leak_basic_principles.htm.

[8] 2015. [Online]. Available: http://mtm-inc.com/reduce_project_risk/why_is_helium_used_in_vacuum_leak_detection/.

[9] Cfd-online.com, 'Standard k-epsilon model -- CFD-Wiki, the free CFD reference', 2015. [Online]. Available: http://www.cfd-online.com/Wiki/Standard_k-epsilon_model.

[10] Cfd-online.com, 'Courant–Friedrichs–Lewy condition -- CFD-Wiki, the free CFD reference', 2015. [Online]. Available: http://www.cfd-online.com/Wiki/Courant%E2%80%93Friedrichs%E2%80%93Lewy_condition.

[11] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007, pp. 113-114.

[12] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007, pp. 40-44.

[13] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007, pp. 57-59.

[14] L. Team, 'Computational Fluid Dynamics (CFD) Blog - LEAP Australia & New Zealand | Turbulence Part 3 – Selection of wall functions and Y+ to best capture the Turbulent Boundary Layer', *Computationalfluiddynamics.com.au*, 2015. [Online]. Available: http://www.computationalfluiddynamics.com.au/turbulence-part-3-selection-of-wall-functions-and-y-to-best-capture-the-turbulent-boundary-layer/.

[15] M. Chmielewski and M. Gieras, 'Three-zonal Wall Function for k-ε Turbulence Models', Warsaw University of Technology, Institute of Heat Engineering, Warsaw, 2013, pp. 2.

[16] G. Kalitzin, G. Medic, G. Iaccarino and P. Durbin, 'Near-wall behavior of RANS turbulence models and implications for wall functions', Flow Physics and Computation Division, Department of Mechanical Engineering, Stanford University, Stanford, 2004. [Online]. Available: http://web.stanford.edu/group/uq/pdfs/journals/jcp_wf.pdf

[17] M. Chmielewski and M. Gieras, 'Three-zonal Wall Function for k-ε Turbulence Models', Warsaw University of Technology, Institute of Heat Engineering, Warsaw, 2013, pp. 1.

[18] COMSOL Blog, 'Which Turbulence Model Should I Choose for my CFD Application?', 2013. [Online]. Available: http://www.comsol.com/blogs/which-turbulence-model-should-choose-cfd-application/.

[19] P. Roberts and D. Webster, 'TURBULENT DIFFUSION', School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, Georgia, 2010.

[20] E. Paul, V. Atiemo-Obeng and S. Kresta, *Handbook of industrial mixing*. Hoboken, N.J.: Wiley-Interscience, 2004, p. 22.

[21] E. Paul, V. Atiemo-Obeng and S. Kresta, *Handbook of industrial mixing*. Hoboken, N.J.: Wiley-Interscience, 2004, p. 22-24.

[22] S. Westgard, 'Z-4: Mean, Standard Deviation, And Coefficient Of Variation - Westgard', *Westgard.com*, 2015. [Online]. Available: https://www.westgard.com/lesson34.htm.

[23] E. Paul, V. Atiemo-Obeng and S. Kresta, *Handbook of industrial mixing*. Hoboken, N.J.: Wiley-Interscience, 2004, p. 339.

[24] O. Foundation, 'Features of OpenFOAM', *Openfoam.org*, 2015. [Online]. Available: http://www.openfoam.org/features/.

[25] CFD Direct, 'OpenFOAM User Guide: 4.1 OpenFOAM case directory', 2015. [Online]. Available: http://cfd.direct/openfoam/user-guide/case-file-structure/.

[26] Imgbuddy.com, 'Imgs For > Moment Of Inertia Hollow Cylinder', 2015. [Online]. Available: http://imgbuddy.com/moment-of-inertia-hollow-cylinder.asp.

[27] CFD Direct, 'OpenFOAM User Guide: 5.2 Boundaries', 2015. [Online]. Available: http://cfd.direct/OpenFOAM/user-guide/boundaries/.

[28] CFD Direct, 'OpenFOAM User Guide: 5.1 Mesh description', 2015. [Online]. Available: http://cfd.direct/OpenFOAM/user-guide/mesh-description/.

[29] Jullio.pe.kr, '6.2.2 Determining Turbulence Parameters', 2015. [Online]. Available: http://jullio.pe.kr/fluent6.1/help/html/ug/node178.htm.

[30] Scialert.net, 'Stabilization of Non-premixed Turbulent Combustion by a Swirling Air Flow', 2015. [Online]. Available: http://www.scialert.net/fulltext/?doi=ajaps.2012.445.459&org=12.

[31] CFD Direct, 'OpenFOAM User Guide: 7.1 Thermophysical models', 2015. [Online]. Available: http://cfd.direct/openfoam/user-guide/thermophysical.

[32] A. Lundstöm, 'Simple Gas Phase Reaction', Chalmers University of Technology, 2008.

[33] T. Poinsot and D. Veynante, *Theoretical and numerical combustion*. Philadelphia: Edwards, 2005, p. 7.

[34] T. Poinsot and D. Veynante, *Theoretical and numerical combustion*. Philadelphia: Edwards, 2005, p. 24.

[35] T. Poinsot and D. Veynante, *Theoretical and numerical combustion*. Philadelphia: Edwards, 2005, p. 6.

[36] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007, pp. 75.

[37] Rodi, W., "Turbulence Models and their Application in Hydraulics," State of the Art Paper, Presented by the IAHR-Section on Fundamentals of Division II: Exp. and Math. Fluid Dynamics, 1979.

[38] CFD Direct, 'OpenFOAM User Guide: 4.4 Numerical schemes', 2015. [Online]. Available: http://cfd.direct/openfoam/user-guide/fvschemes/.

[39] Se.mathworks.com, 'Preconditioned conjugate gradients method - MATLAB pcg', 2015. [Online]. Available: http://se.mathworks.com/help/matlab/ref/pcg.html.

[40] Netlib.org, 'BiConjugate Gradient (BiCG)', 2015. [Online]. Available: http://www.netlib.org/linalg/html_templates/node32.html.

[41] T. Behrens, 'OpenFOAM's basic solvers for linear systems of equations', Technical University of Denmark, 2009.

[42] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007, pp. 191-192.

[43] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007, pp. 193-196.

[44] H. Aguerre, S. Damián, J. Gimenez and N. Nigro, *MODELING OF COMPRESSIBLE FLUID PROBLEMS WITH OPENFOAM USING DYNAMIC MESH TECHNOLOGY*. Mendoza: Asociacion Argentia de Mecanica Computacional, 2013, pp. 1001-1002.

[45] C. Gualtieri and F. Bombardelli, 'Parameterization of the turbulent Schmidt number in the numerical simulation of open-channel flows', IAHR Congress, Tsinghua University Press, Beijing, 2013.

[46] T. Poinsot and D. Veynante, *Theoretical and numerical combustion*. Philadelphia: Edwards, 2005, p. 14-15.

[47] Cfd-online.com, 'Use of k-epsilon and k-omega Models -- CFD Online Discussion Forums', 2015. [Online]. Available: http://www.cfd-online.com/Forums/main/75554-use-k-epsilon-k-omega-models.html.

9. Appendices

9.1 Appendix A: `0`-Folder – Field Initial Conditions

1st note: The `alphat`-, `He`-, `N2`-, `p`- and `T`-field initial conditions are common for each case. The remaining `epsilon`-, `k`- and `U`-field initial conditions vary from one case to another.

2nd note: The Ydefault-file is used in simulations where there is a large number of species. It enables the user to define common boundary conditions for a range of specie. The alphat-file contains turbulence thermal diffusivity parameters. Both files are, in this thesis' simulations, unused. Their content is only zero-values.

9.1.1 `alphat`

```
/*--------------------------------*- 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
{
    inletN2
    {
        type            fixedValue;
        value           uniform 0;
    }
    inletHe
    {
        type            fixedValue;
        value           uniform 0;
    }
    outlet
    {
        type            zeroGradient;
    }
    wall
    {
        type            zeroGradient;
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            zeroGradient;
    }
}


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

9.1.2 epsilon

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.x                                 |
|   \\  /    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 677.1;

boundaryField
{
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    wall
    {
        type            compressible::epsilonLowReWallFunction;
        value           uniform 677.1;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;
    }
    inletN2
    {
        type            inletOutlet;
        inletValue      uniform 1274;
        value           uniform 1274;
    }
    outlet
    {
        type            zeroGradient;
    }
    inletHe
    {
        type            inletOutlet;
        inletValue      uniform 173.3;
        value           uniform 173.3;
    }
    thinWall
    {
        type            compressible::epsilonWallFunction;
        value           uniform 677.1;
        Cmu             0.09;
        kappa           0.41;
        E               9.8;

    }
}


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

9.1.3 He

```
/*--------------------------------*- 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      O2;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inletN2
    {
        type            fixedValue;
        value           uniform 0.0;
    }
    inletHe
    {
        type            fixedValue;
        value           uniform 1.0;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
    wall
    {
        type            zeroGradient;
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            zeroGradient;
    }
}


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

9.1.4 k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.x                                 |
|   \\  /    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 2.998;

boundaryField
{
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    wall
    {
        type            kLowReWallFunction;
        value           uniform 2.998;
    }
    inletN2
    {
        type            inletOutlet;
        inletValue      uniform 3.678;
        value           uniform 3.678;
    }
    outlet
    {
        type            zeroGradient;
    }
    inletHe
    {
        type            inletOutlet;
        inletValue      uniform 0.444;
        value           uniform 0.444;
    }
    thinWall
    {
        type            kqRWallFunction;
        value           uniform 2.998;
    }
}


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

9.1.5 N2

```
/*--------------------------------*- 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      O2;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 1;

boundaryField
{
    inletN2
    {
        type            fixedValue;
        value           uniform 1.0;
    }
    inletHe
    {
        type            fixedValue;
        value           uniform 0.0;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 1;
        value           uniform 1;

    }
    wall
    {
        type            zeroGradient;
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            zeroGradient;
    }
}


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

9.1.6 p

```
/*--------------------------------*- 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      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 8.106e5; // 8bar=1.01325e5*8Pa

boundaryField
{
    inletN2
    {
        type            zeroGradient;
    }
    inletHe
    {
        type            zeroGradient;
    }
    outlet
    {
        type            outletInlet;
        outletValue     uniform 8.106e5;
        value           uniform 1.013e5;
    }
    wall
    {
        type            zeroGradient;
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            zeroGradient;
    }
}


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

9.1.7 T

```
/*--------------------------------*- 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      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 353;

boundaryField
{
    inletN2
    {
        type            fixedValue;
        value           uniform 353; //60 degrees celsius
    }
    inletHe
    {
        type            fixedValue;
        value           uniform 353; //Room temperature
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 353;
        value           uniform 353;
    }
    wall
    {
        type            zeroGradient;
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            zeroGradient;
    }
}


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

9.1.8 U

```
/*--------------------------------*- 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;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 32.75);

boundaryField
{
   inletN2
    {
        type            inletOutlet;
        inletValue      uniform (0 0 35.13);
        value           (0 0 0.01);
    }
    inletHe
    {
        type            inletOutlet;
        inletValue      uniform (0 0 6.631);
        value           uniform (0 0 0.01);
    }
    outlet
    {
        type            zeroGradient;
    }
    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}


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

9.1.9 Ydefault

```
/*--------------------------------*- 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      Ydefault;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0.0;

boundaryField
{
    inletN2
    {
        type            fixedValue;
        value           uniform 0.0;
    }
    inletHe
    {
        type            fixedValue;
        value           uniform 0.0;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 0.0;
        value           uniform 0.0;

    }
    wall
    {
        type            zeroGradient;
    }
    front
    {
        type            wedge;
    }
    back
    {
        type            wedge;
    }
    bluff
    {
        type            zeroGradient;
    }
}


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

9.2 Appendix B: `Constant`-Folder – Case Mesh

9.2.1 `blockMeshDict – coarserStandard`

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                 //0
(0.002249228981 0.000058898134 0)       //1
(0.002249228981 0.000058898134 0.5)     //2
(0 0 0.5)                               //3
(0.002249228981 -0.000058898134 0)      //4
(0.002249228981 -0.000058898134 0.5)    //5

(0.006747686944 0.000176694401 0)       //6
(0.006747686944 0.000176694401 0.5)     //7
(0.006747686944 -0.000176694401 0)      //8
(0.006747686944 -0.000176694401 0.5)    //9

(0.008996915925 0.000235592535 0)       //10
(0.008996915925 0.000235592535 0.5)     //11
(0.008996915925 -0.000235592535 0)      //12
(0.008996915925 -0.000235592535 0.5)    //13
);

blocks
(
hex (0 4 1 0 3 5 2 3) (8 1 450) simpleGrading (0.1 1 1)
hex (1 4 8 6 2 5 9 7) (1 16 450) simpleGrading (1 10 1)
hex (6 8 12 10 7 9 13 11) (1 8 450) simpleGrading (1 0.1 1)
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                 (0 1 2 3)
                (1 6 7 2)
               (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
               (0 3 5 4)
               (4 5 9 8)
               (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
               (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
               (0 4 1 0)
               (1 4 8 6)
               (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
               (3 5 2 3)
               (2 5 9 7)
               (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.2 blockMeshDict – Case 1 – standard

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                 //0
(0.002249228981 0.000058898134 0)      //1
(0.002249228981 0.000058898134 0.5)    //2
(0 0 0.5)                               //3
(0.002249228981 -0.000058898134 0)     //4
(0.002249228981 -0.000058898134 0.5)   //5

(0.006747686944 0.000176694401 0)      //6
(0.006747686944 0.000176694401 0.5)    //7
(0.006747686944 -0.000176694401 0)     //8
(0.006747686944 -0.000176694401 0.5)   //9

(0.008996915925 0.000235592535 0)      //10
(0.008996915925 0.000235592535 0.5)    //11
(0.008996915925 -0.000235592535 0)     //12
(0.008996915925 -0.000235592535 0.5)   //13
);
```

```
blocks
(
hex (0 4 1 0 3 5 2 3) (9 1 500) simpleGrading (0.1 1 1) //2.25 => 9
hex (1 4 8 6 2 5 9 7) (1 18 500) simpleGrading (1 10 1) //4.5 => 18
hex (6 8 12 10 7 9 13 11) (1 9 500) simpleGrading (1 0.1 1) //2.25 => 9
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
               (1 6 7 2)
                (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (4 5 9 8)
                (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 8 6)
                (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (3 5 2 3)
                (2 5 9 7)
                (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.3 blockMeshDict – finerStandard

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                 //0
(0.002249228981 0.000058898134 0)       //1
(0.002249228981 0.000058898134 0.5)     //2
(0 0 0.5)                               //3
(0.002249228981 -0.000058898134 0)      //4
(0.002249228981 -0.000058898134 0.5)    //5

(0.006747686944 0.000176694401 0)       //6
(0.006747686944 0.000176694401 0.5)     //7
(0.006747686944 -0.000176694401 0)      //8
(0.006747686944 -0.000176694401 0.5)    //9

(0.008996915925 0.000235592535 0)       //10
(0.008996915925 0.000235592535 0.5)     //11
(0.008996915925 -0.000235592535 0)      //12
(0.008996915925 -0.000235592535 0.5)    //13
);

blocks
(
hex (0 4 1 0 3 5 2 3) (11 1 600) simpleGrading (0.1 1 1) //2.25 => 9
hex (1 4 8 6 2 5 9 7) (1 22 600) simpleGrading (1 10 1) //4.5 => 18
hex (6 8 12 10 7 9 13 11) (1 11 600) simpleGrading (1 0.1 1) //2.25 => 9
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
               (1 6 7 2)
                (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (4 5 9 8)
                (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 8 6)
                (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (3 5 2 3)
                (2 5 9 7)
                (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.4 blockMeshDict – Case 2A

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
```

```
(
(0 0 0)                               //0
(0.003248886306 0.000085075082 0)     //1
(0.003248886306 0.000085075082 0.5)   //2
(0 0 0.5)                             //3
(0.003248886306 -0.000085075082 0)    //4
(0.003248886306 -0.000085075082 0.5)  //5

(0.005748029619 0.000150517453 0)     //6
(0.005748029619 0.000150517453 0.5)   //7
(0.005748029619 -0.000150517453 0)    //8
(0.005748029619 -0.000150517453 0.5)  //9

(0.008996915925 0.000235592535 0)     //10
(0.008996915925 0.000235592535 0.5)   //11
(0.008996915925 -0.000235592535 0)    //12
(0.008996915925 -0.000235592535 0.5)  //13
);

blocks
(
hex (0 4 1 0 3 5 2 3) (13 1 500) simpleGrading (0.1 1 1) //3.25 => 13
hex (1 4 8 6 2 5 9 7) (1 10 500) simpleGrading (1 10 1) //2.5 => 10
hex (6 8 12 10 7 9 13 11) (1 13 500) simpleGrading (1 0.1 1) //3.25 => 13
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
                 (1 6 7 2)
                (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (4 5 9 8)
                (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 8 6)
                (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (3 5 2 3)
                (2 5 9 7)
                (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.5 `blockMeshDict` – Case 2B

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                 //0
(0.003248886306 0.000085075082 0)       //1
(0.003248886306 0.000085075082 0.5)     //2
(0 0 0.5)                               //3
(0.003248886306 -0.000085075082 0)      //4
(0.003248886306 -0.000085075082 0.5)    //5

(0.005748029619 0.000150517453 0)       //6
(0.005748029619 0.000150517453 0.5)     //7
(0.005748029619 -0.000150517453 0)      //8
(0.005748029619 -0.000150517453 0.5)    //9

(0.008996915925 0.000235592535 0)       //10
(0.008996915925 0.000235592535 0.5)     //11
(0.008996915925 -0.000235592535 0)      //12
(0.008996915925 -0.000235592535 0.5)    //13
);

blocks
(
hex (0 4 1 0 3 5 2 3) (13 1 500) simpleGrading (0.1 1 1) //3.25 => 13
hex (1 4 8 6 2 5 9 7) (1 10 500) simpleGrading (1 10 1) //2.5 => 10
hex (6 8 12 10 7 9 13 11) (1 13 500) simpleGrading (1 0.1 1) //3.25 => 13
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
                (1 6 7 2)
                (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (4 5 9 8)
                (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 8 6)
                (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (3 5 2 3)
                (2 5 9 7)
                (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.6 blockMeshDict – Case 2C

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                 //0
(0.004248543631 0.000111252030 0)       //1
(0.004248543631 0.000111252030 0.5)     //2
(0 0 0.5)                               //3
(0.004248543631 -0.000111252030 0)      //4
(0.004248543631 -0.000111252030 0.5)    //5

(0.006497772612 0.000170150164 0)       //6
(0.006497772612 0.000170150164 0.5)     //7
(0.006497772612 -0.000170150164 0)      //8
(0.006497772612 -0.000170150164 0.5)    //9

(0.008996915925 0.000235592535 0)       //10
(0.008996915925 0.000235592535 0.5)     //11
(0.008996915925 -0.000235592535 0)      //12
(0.008996915925 -0.000235592535 0.5)    //13
);

blocks
(
hex (0 4 1 0 3 5 2 3) (17 1 500) simpleGrading (0.1 1 1) //4.25 => 17
hex (1 4 8 6 2 5 9 7) (1 9 500) simpleGrading (1 10 1) //2.25 => 9
hex (6 8 12 10 7 9 13 11) (1 10 500) simpleGrading (1 0.1 1) //2.5 => 10
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
               (1 6 7 2)
                (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (4 5 9 8)
                (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 8 6)
                (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (3 5 2 3)
                (2 5 9 7)
                (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.7 `blockMeshDict` – Case 2D

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                //0
(0.005248200956 0.000137428979 0)      //1
(0.005248200956 0.000137428979 0.5)    //2
```

```
(0 0 0.5)                              //3
(0.005248200956 -0.000137428979 0)     //4
(0.005248200956 -0.000137428979 0.5)   //5

(0.006997601275 0.000183238638 0)      //6
(0.006997601275 0.000183238638 0.5)    //7
(0.006997601275 -0.000183238638 0)     //8
(0.006997601275 -0.000183238638 0.5)   //9

(0.008996915925 0.000235592535 0)      //10
(0.008996915925 0.000235592535 0.5)    //11
(0.008996915925 -0.000235592535 0)     //12
(0.008996915925 -0.000235592535 0.5)   //13
);

blocks
(
hex (0 4 1 0 3 5 2 3) (21 1 500) simpleGrading (0.1 1 1) //5.25 => 21
hex (1 4 8 6 2 5 9 7) (1 7 500) simpleGrading (1 10 1) //1.75 => 7
hex (6 8 12 10 7 9 13 11) (1 8 500) simpleGrading (1 0.1 1) //2 => 8
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
                (1 6 7 2)
                (6 10 11 7)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (4 5 9 8)
                (8 9 13 12)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (10 12 13 11)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 8 6)
                (6 8 12 10)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (3 5 2 3)
                (2 5 9 7)
                (7 9 13 11)
           );
      }
);

mergePatchPairs
(
);

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

9.2.8 `blockMeshDict` – Case 3A

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)                                 //0
(0.002249228981 0.000058898134 0)       //1
(0.002249228981 0.000058898134 0.025)   //2
(0 0 0.025)                             //3
(0.002249228981 -0.000058898134 0)      //4
(0.002249228981 -0.000058898134 0.025)  //5
(0.002249228981 0.000058898134 0.03)    //6
(0 0 0.03)                              //7
(0.002249228981 -0.000058898134 0.03)   //8
(0.002249228981 0.000058898134 0.5)     //9
(0 0 0.5)                               //10
(0.002249228981 -0.000058898134 0.5)    //11

(0.006497772612 0.000170150164 0)       //12
(0.006497772612 0.000170150164 0.025)   //13
(0.006497772612 -0.000170150164 0)      //14
(0.006497772612 -0.000170150164 0.025)  //15
(0.006497772612 0.000170150164 0.03)    //16
(0.006497772612 -0.000170150164 0.03)   //17
(0.006497772612 0.000170150164 0.5)     //18
(0.006497772612 -0.000170150164 0.5)    //19

(0.008996915925 0.000235592535 0)       //20
(0.008996915925 0.000235592535 0.025)   //21
(0.008996915925 -0.000235592535 0)      //22
(0.008996915925 -0.000235592535 0.025)  //23
(0.008996915925 0.000235592535 0.03)    //24
```

```
(0.008996915925 0.000235592535 0.5)    //25
(0.008996915925 -0.000235592535 0.03)  //26
(0.008996915925 -0.000235592535 0.5)   //27
);

blocks
(
hex (0 4 1 0 3 5 2 3) (9 1 25) simpleGrading (0.1 1 1)
hex (3 5 2 3 7 8 6 7) (9 1 5) simpleGrading (0.1 1 1)
hex (7 8 6 7 10 11 9 10) (9 1 470) simpleGrading (0.1 1 1)

hex (1 4 14 12 2 5 15 13) (1 18 25) simpleGrading (1 1 1)
hex (2 5 15 13 6 8 17 16) (1 18 5) simpleGrading (1 1 1)
hex (6 8 17 16 9 11 19 18) (1 18 470) simpleGrading (1 1 1)

hex (12 14 22 20 13 15 23 21) (1 9 25) simpleGrading (1 0.1 1)
hex (16 17 26 24 18 19 27 25) (1 9 470) simpleGrading (1 0.1 1)
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
               (3 2 6 7)
                (7 6 9 10)
                (1 12 13 2)
                (2 13 16 6)
                (6 16 18 9)
                (12 20 21 13)
                (16 24 25 18)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (3 7 8 5)
                (7 10 11 8)
                (4 5 15 14)
                (5 8 17 15)
                (8 11 19 17)
                (14 15 23 22)
                (17 19 27 26)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (20 22 23 21)
                (24 26 27 25)
                (13 15 17 16)
                (16 17 26 24)
                (13 15 23 21)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 14 12)
                (12 14 22 20)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (10 11 9 10)
                (9 11 19 18)
                (18 19 27 25)
           );
      }
     axis
     {
           type empty;
           faces
           (
               (0 3 3 0)
           );
      }
);

mergePatchPairs
(
);

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

9.2.9 `blockMeshDict` – Case 3B

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
//baffle#1 vertices
(0 0 0)                                   //0
(0.002249228981 0.000058898134 0)       //1
(0.002249228981 0.000058898134 0.025)   //2
(0 0 0.025)                               //3
(0.002249228981 -0.000058898134 0)      //4
(0.002249228981 -0.000058898134 0.025)  //5
(0.002249228981 0.000058898134 0.03)    //6
```

```
(0 0 0.03)                              //7
(0.002249228981 -0.000058898134 0.03)   //8
(0.002249228981 0.000058898134 0.035)   //9
(0 0 0.035)                             //10
(0.002249228981 -0.000058898134 0.035)  //11

(0.006497772612 0.000170150164 0)       //12
(0.006497772612 0.000170150164 0.025)   //13
(0.006497772612 -0.000170150164 0)      //14
(0.006497772612 -0.000170150164 0.025)  //15
(0.006497772612 0.000170150164 0.03)    //16
(0.006497772612 -0.000170150164 0.03)   //17
(0.006497772612 0.000170150164 0.035)   //18
(0.006497772612 -0.000170150164 0.035)  //19

(0.008996915925 0.000235592535 0)       //20
(0.008996915925 0.000235592535 0.025)   //21
(0.008996915925 -0.000235592535 0)      //22
(0.008996915925 -0.000235592535 0.025)  //23
(0.008996915925 0.000235592535 0.03)    //24
(0.008996915925 0.000235592535 0.035)   //25
(0.008996915925 -0.000235592535 0.03)   //26
(0.008996915925 -0.000235592535 0.035)  //27

//baffle#2 vertices
(0.006497772612 0.000170150164 0.04)    //28
(0.002249228981 0.000058898134 0.04)    //29
(0.006497772612 -0.000170150164 0.04)   //30
(0.002249228981 -0.000058898134 0.04)   //31

(0.006497772612 0.000170150164 0.5)     //32
(0.002249228981 0.000058898134 0.5)     //33
(0.006497772612 -0.000170150164 0.5)    //34
(0.002249228981 -0.000058898134 0.5)    //35

(0 0 0.04)                              //36
(0 0 0.5)                               //37

(0.008996915925 0.000235592535 0.04)    //38
(0.008996915925 -0.000235592535 0.04)   //39

(0.008996915925 0.000235592535 0.5)     //40
(0.008996915925 -0.000235592535 0.5)    //41
);

blocks
(
hex (0 4 1 0 3 5 2 3) (9 1 25) simpleGrading (0.1 1 1)
hex (3 5 2 3 7 8 6 7) (9 1 5) simpleGrading (0.1 1 1)
hex (7 8 6 7 10 11 9 10) (9 1 5) simpleGrading (0.1 1 1)
hex (36 31 29 36 37 35 33 37) (9 1 460) simpleGrading (0.1 1 1)

hex (1 4 14 12 2 5 15 13) (1 18 25) simpleGrading (1 1 1)
hex (2 5 15 13 6 8 17 16) (1 18 5) simpleGrading (1 1 1)
hex (6 8 17 16 9 11 19 18) (1 18 5) simpleGrading (1 1 1)
hex (9 11 19 18 29 31 30 28) (1 18 5) simpleGrading (1 1 1)
hex (29 31 30 28 33 35 34 32) (1 18 460) simpleGrading (1 1 1)

hex (12 14 22 20 13 15 23 21) (1 9 25) simpleGrading (1 0.1 1)
hex (16 17 26 24 18 19 27 25) (1 9 5) simpleGrading (1 0.1 1)
hex (18 19 27 25 28 30 39 38) (1 9 5) simpleGrading (1 0.1 1)
hex (28 30 39 38 32 34 41 40) (1 9 460) simpleGrading (1 0.1 1)
);

edges
(
);

boundary
(
     front
     {
           type wedge;
           faces
           (
                (0 1 2 3)
               (3 2 6 7)
                (7 6 9 10)
                (1 12 13 2)
                (2 13 16 6)
                (6 16 18 9)
                (12 20 21 13)
                (16 24 25 18)

                (9 18 28 29)
                (29 28 32 33)
                (36 29 33 37)
                (18 25 38 28)
                (28 38 40 32)
           );
      }
     back
     {
           type wedge;
           faces
           (
                (0 3 5 4)
                (3 7 8 5)
                (7 10 11 8)
                (4 5 15 14)
                (5 8 17 15)
                (8 11 19 17)
                (14 15 23 22)
                (17 19 27 26)

                (11 31 30 19)
                (31 35 34 30)
                (36 37 35 31)
                (19 30 39 27)
                (30 34 41 39)
           );
      }
     wall
     {
           type wall;
           faces
           (
                (20 22 23 21)
                (24 26 27 25)
                (13 15 17 16)
                (16 17 26 24)
                (13 15 23 21)

                (9 11 31 29)
                (36 31 29 36)
                (10 11 9 10)
                (25 27 39 38)
                (38 39 41 40)
           );
      }
     inletN2
     {
           type patch;
           faces
           (
                (0 4 1 0)
                (1 4 14 12)
                (12 14 22 20)
           );
      }
     outlet
     {
           type patch;
           faces
           (
                (37 35 33 37)
                (33 35 34 32)
                (32 34 41 40)
           );
      }
);

mergePatchPairs
(
);

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

9.3 Appendix C: `constant`-Folder – Physical Properties

9.3.1 `chemistryProperties`

```
/*--------------------------------*- 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      chemistryProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

chemistryType
{
    chemistrySolver   noChemistrySolver;
    chemistryThermo   psi;
}

chemistry           off;

initialChemicalTimeStep 1e-07;

EulerImplicitCoeffs
{
    cTauChem        1;
    equilibriumRateLimiter off;
}

odeCoeffs
{
    solver          Rosenbrock43;
    absTol          1e-12;
    relTol          0.01;
}

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

9.3.2 combustionProperties

```
/*--------------------------------*- 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      combustionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

combustionModel  laminar<psiChemistryCombustion>;

active  true;

laminarCoeffs
{
}

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

9.3.3 g

```
/*--------------------------------*- 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       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           ( 0 0 0 );

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

9.3.4 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;

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

9.3.5 reactions

```
species
(
    He
    N2
);

reactions
{
}
```

9.3.6 thermo.compressibleGas

```
/*--------------------------------*- 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      thermo.compressibleGas;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

N2
{
    specie
    {
        nMoles          1;
        molWeight       28.0134;
    }
    thermodynamics
    {
        Cp   1044;
        Hf   2.55e+04;
    }
    transport
    {
        mu              1.91e-06;
        Pr              0.69;
    }
}
```

```
He
{
    specie
    {
        nMoles          1;
        molWeight       4.02;
    }
    thermodynamics
    {
        Cp   5194;
        Hf   5.23e+03;
    }
    transport
    {
        mu              2.10e-06;
        Pr              0.71;
    }
}

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

9.3.7 thermophyscalProperties

```
/*--------------------------------*- 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         reactingMixture;
    transport       const;
    thermo          hConst;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}

inertSpecie N2;

inertSpecie He;

chemistryReader foamChemistryReader;

foamChemistryFile "$FOAM_CASE/constant/reactions";

foamChemistryThermoFile "$FOAM_CASE/constant/thermo.compressibleGas";


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

9.3.8 `transportProperties`

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;
nu              1.657e-05;

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

9.3.9 `turbulenceProperties`

```
/*--------------------------------*- 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      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//N2 is turbulent. How to solve reactingFoam with turbulence?

simulationType  RASModel;

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

9.4 Appendix D: `system`-Directory

9.4.1 `controlDict`

Note: there are two functions defined for data output, areaAverage and CoV. For the sake of not flooding the appendices there is only included one function of each. The method of extracting the wanted data from the simulations has thus been presented.

```
/*--------------------------------*- 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     reactingFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.05;

deltaT          1e-6;

writeControl    adjustableRunTime;

writeInterval   0.005;

purgeWrite      0;

writeFormat     ascii;

writePrecision  12;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           0.2;

functions
{

planeAreaAverage
      {
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;
        log             true;
        valueOutput     true;
        source          sampledSurface;
        surfaceFormat   raw;
        sampledSurfaceDict
        {
            type            plane;
            basePoint       (0 0 0.4999);
            normalVector    (0 0 1);
        }
        operation       areaAverage;
        fields
        (
            He
            N2
        );
      }

planeCoV0.500
      {
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;
        log             true;
        valueOutput     true;
        source          sampledSurface;
        surfaceFormat   raw;
        sampledSurfaceDict
        {
            type            plane;
            basePoint       (0 0 0.4999);
            normalVector    (0 0 1);
        }
        operation       CoV;
        fields
        (
            He
            N2
        );
      }
}


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

9.4.2 createPatchDict

```
/*--------------------------------*- 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      createPatchDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
pointSync false;

patches
(
    {
        name inletHe;
        patchInfo
        {
            type patch;
        }
        constructFrom set;
        set inletHePatchFaces;
    }

 {
        name thinWall;
        patchInfo
        {
            type wall;
        }
        constructFrom set;
        set bluffPatchFaces;
    }

);

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

9.4.3 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         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;

    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,Yi_h)   Gauss limitedLinear 1;
    div(phi,K)      Gauss limitedLinear 1;
    div(phid,p)     Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,k) Gauss limitedLinear 1;
    div((muEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear orthogonal;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         orthogonal;
}

fluxRequired
```

```
{
    default         no;
    p;
}

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

9.4.4 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
{
    "rho.*"
    {
        solver          diagonal;
    }

    p
    {
        solver           PCG;
        preconditioner   DIC;
        tolerance        1e-6;
        relTol           0.1;
    }

    pFinal
    {
        $p;
        tolerance        1e-6;
        relTol           0.0;
    }

    "(U|h|k|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0.1;
    }

    "(U|h|k|epsilon)Final"
    {
        $U;
        relTol          1e-10;
    }

    Yi
    {
        $hFinal;
    }
}
```

```
PIMPLE
{
    momentumPredictor no;
    nOuterCorrectors  1;
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
}


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

9.4.5 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;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//inletHe generation
//on inletN2 defined in blockMeshDict
actions
(
    {
        name    inletFaces;
        type    faceSet;
        action  new;
        source  patchToFace;
        sourceInfo
        {
            name "inletN2";
        }
    }

    {
        name    inletHeCells;
        type    cellSet;
        action  new;
        source  cylinderToCell;
        sourceInfo
        {
            p1      (0 0 -0.05);
          p2      (0 0 0.05);
            radius  0.002;
        }
    }

    {
        name    inletHeFaces;
        type    faceSet;
        action  new;
        source  cellToFace;
        sourceInfo
        {
            set     inletHeCells;
            option  all;
        }
    }

       {
        name    bluffCells;
        type    cellSet;
        action  new;
        source  cylinderAnnulusToCell;
        sourceInfo
        {
            p1      (0 0 -0.05);
            p2      (0 0 0.05);
            outerRadius  0.0025;
             innerRadius  0.002;

        }
       }

       {
        name    bluffFaces;
        type    faceSet;
        action  new;
        source  cellToFace;
        sourceInfo
        {
            set     bluffCells;
            option  all;
        }
    }
// Start with the entire inletN2-face
       {
        name    coFlowPatchFaces;
        type    faceSet;
        action  new;
        source  faceToFace;
        sourceInfo
        {
            set inletFaces;
        }
    }
// Remove the bluff-faces
       {
        name    coFlowPatchFaces;
        type    faceSet;
        action  delete;
        source  faceToFace;
        sourceInfo
        {
                    set bluffFaces;
        }
    }
// Remove the inletHe-faces
       {
        name    coFlowPatchFaces;
        type    faceSet;
        action  delete;
        source  faceToFace;
        sourceInfo
        {
                    set inletHeFaces;
        }
    }
// Start with the entire inletN2-face
       {
        name    bluffPatchFaces;
        type    faceSet;
        action  new;
        source  faceToFace;
        sourceInfo
        {
            set inletFaces;
        }
    }
// Remove the bluff-faces
       {
        name    bluffPatchFaces;
        type    faceSet;
        action  delete;
        source  faceToFace;
        sourceInfo
        {
                   set coFlowPatchFaces;
        }
    }
// Remove the inletHe-faces
       {
        name    bluffPatchFaces;
        type    faceSet;
        action  delete;
        source  faceToFace;
        sourceInfo
        {
                   set inletHeFaces;
        }
    }
// Start with the entire inletN2-face
       {
        name    inletHePatchFaces;
        type    faceSet;
        action  new;
        source  faceToFace;
        sourceInfo
        {
            set inletFaces;
        }
    }
// Remove the coflow-faces
       {
        name    inletHePatchFaces;
        type    faceSet;
        action  delete;
        source  faceToFace;
        sourceInfo
        {
                   set coFlowPatchFaces;
        }
    }
// Remove the bluff-faces
       {
        name    inletHePatchFaces;
        type    faceSet;
        action  delete;
        source  faceToFace;
        sourceInfo
        {
                   set bluffFaces;
        }
    }

);

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

9.5 Appendix E: Dropbox-Link

The following link contains the OpenFOAM simulations conducted in this thesis:

https://www.dropbox.com/sh/j4fvwy0sb06ieef/AABOYPsrke1flSoBqUNelIy7a?dl=0