




Universitetet  
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

## MASTER'S THESIS

Study program/specialization: <b>Engineering structures and Materials w/specialization in Mechanical Systems</b>	<b>Spring semester, 2019</b> <b>Open/Confidential</b>
Author: <b>Adnan Khalid</b>	 ..... (signature of author)
Program coordinator: Supervisor(s):	<b>Dr. Knut Erik T. Giljarhus</b> <b>Dr. Knut Erik T. Giljarhus</b> <b>Dr. Hirpa G. Lemu</b> <b>Mr. Jakob Trydal</b>
Title of master's thesis:	<b>Optimization of 2K-mixer for Paint</b>
Credits (ECT): <b>30</b>	
Keywords: <b>2K-Mixer</b> <b>Static Mixers</b> <b>OpenFOAM</b> <b>Coefficient of Variation (COV)</b> <b>Computational Fluid Dynamics (CFD)</b>	Number of pages: <b>49</b>  + supplemental material/other: <b>18</b>  Stavanger, 15.06.2019 date/year



Universitetet  
i Stavanger

MSC ENGINEERING STRUCTURES AND MATERIALS

MASTER'S THESIS

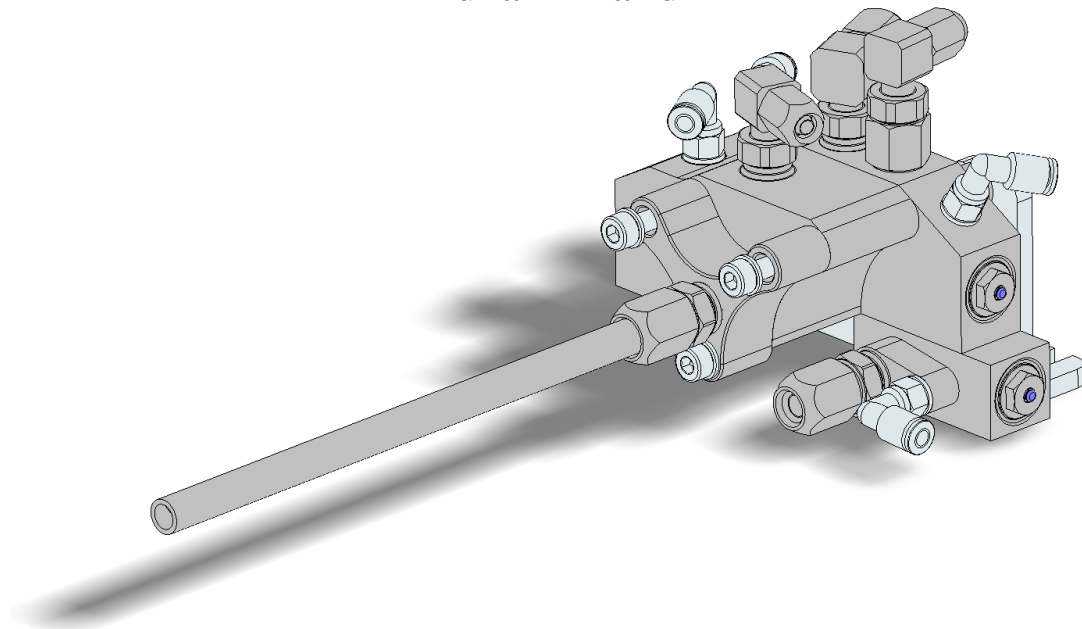
---

# Optimization of 2K-mixer for Paint

---

**AUTHOR:**

Adnan Khalid



University of Stavanger

In association with ABB AS.

Compiled in L<sup>A</sup>T<sub>E</sub>X

15th June 2019

*Supervisors:*

Dr. Knut Erik Giljarhus

Dr. Hirpa G. Lemu

Mr. Jakob Trydal

# Acknowledgement

This thesis concludes a two-year master degree in Engineering structures and Materials. I would like to express my endless gratitude to my supervisor Dr. Knut Erik Giljarhus and my external supervisor from ABB AS Mr. Jakob Trygdal, for introducing me to this interesting topic and helping me with all the different concepts involving calculations and research in optimization of a mixer for paint applications. I value the time Dr. Giljarhus spent guiding me through OpenFOAM and solving endless problems and issued throughout the last six months. I would also like to thank the University of Stavanger for being generous by giving me access to the UiS supercomputers. This allowed me to run detailed simulations that would require large computation power and therefore only possible on supercomputers such as theirs or similar. I would like to thank Robel Amar, Prashanth Sivaganesh and Rakul Inthrakumar for taking the time for proofreading the thesis. The most important thing in the course of the two-year study, and especially during writing process, was the support I received from my friends and family

# Abstract

This work presents a parametric study of a 2K-mixer for paint application that utilizes static mixers for mixing two component paint applied on a car body. The mixers is installed in a specialized valve block giving it the ability to mix together two component paint by cutting, folding, twisting and recombining the the fluid elements in the direction transverse to the main flow. The specialized block valve has the ability to quickly change the color if the next car body requires a different paint color. It is done by a cleaning process, where the mixed paint inside the mixer is wasted. To limit the waste of paint, this paper investigates the improvements that can be done to the block valve to ensure satisfactory mixing quality and less paint waste

# Contents

<b>List of Figures</b>	<b>III</b>
<b>List of Tables</b>	<b>III</b>
<b>Nomenclatures</b>	<b>V</b>
Abbreviations . . . . .	V
Greek Letters . . . . .	V
Roman Letters . . . . .	VI
<b>1 Introduction</b>	<b>1</b>
<b>2 Static mixers</b>	<b>3</b>
2.1 Mixing of miscible liquids . . . . .	5
2.1.1 Striation thickness model . . . . .	6
2.2 Reynolds Number in Static Mixers . . . . .	8
2.3 Assessing mixing homogeneity . . . . .	9
2.3.1 Coefficient of Variation . . . . .	9
2.3.2 Homogeneous mixing criteria . . . . .	11
<b>3 Governing Equations</b>	<b>12</b>
3.1 Mass conservation . . . . .	12
3.2 Momentum conservation . . . . .	14
3.3 Navier-Stokes equations for Newtonian fluids . . . . .	15
3.4 Scalar Transport Equation . . . . .	16
<b>4 2K-Mixer</b>	<b>17</b>
4.1 Stadelmaier static mixing element . . . . .	18
4.2 Simplified CAD model . . . . .	19
4.3 Paint Viscosity . . . . .	21
<b>5 OpenFOAM</b>	<b>22</b>
5.1 Mesh Setup . . . . .	23
5.1.1 blockMesh . . . . .	23
5.1.2 snappyHexMesh . . . . .	25
5.2 Computational Setup . . . . .	27
5.2.1 Boundary Conditions . . . . .	28
5.2.2 Time and data input/output control . . . . .	29
5.2.3 Added function Objects . . . . .	29
5.2.4 Numerical schemes . . . . .	31
5.2.5 Solution and algorithm control . . . . .	32

<b>6</b>	<b>Results &amp; discussion</b>	<b>33</b>
6.1	Meshing results . . . . .	33
6.2	Mesh convergence test . . . . .	34
6.3	Mixing results . . . . .	37
6.3.1	Varying viscosity . . . . .	39
6.3.2	Varying velocity . . . . .	40
6.3.3	Overall effect of viscosity and velocity . . . . .	41
6.3.4	Mixing result at varying tube diameter . . . . .	42
6.4	Modified design . . . . .	43
<b>7</b>	<b>Conclusion</b>	<b>46</b>
	<b>References</b>	<b>47</b>
	<b>Appendix</b>	<b>i</b>
A	Technical specifications of the 2K-mixer . . . . .	ii
B	CAD drawings . . . . .	iv
B.1	Static mixing tube 6.4 mm . . . . .	iv
B.2	Static mixing tube 6.4 mm modified design . . . . .	v
C	Viscosity conversion table . . . . .	vi
D	OpenFOAM base case . . . . .	vii
D.1	0 folder . . . . .	vii
D.2	Constant folder . . . . .	ix
D.3	System folder . . . . .	x
E	Simulation results . . . . .	xvii

# List of Figures

1	ABB IRB 5500-22 Paint Robot [5]	2
2	Static mixers sequential operations [8]	3
3	Most important issues to consider when selecting type of mixers	4
4	Spatial inhomogeneities [7]	5
5	Response of striation thickness [7]	6
6	Mixing mechanisms [7]	6
7	Schematical representation of the mixing of two fluids [11]	7
8	Mixing of black and white epoxy in a helical static mixer [11]	7
9	Mass flow in and out of a fluid element [14]	13
10	Stress components on a fluid element [14]	14
11	2K-mixer	17
12	Stadelmaier mixing elements	18
13	Half section view of the 2K-mixer	19
14	Simplified CAD model	20
15	Basic file structure in OpenFOAM	22
16	Block structure for the geometry [24]	24
17	blockMesh	24
18	snappyHexMesh process [25]	25
19	Sampled surfaces	30
20	Mesh quality	34
21	Base mesh residuals	35
22	COV at Plane 0	36
23	Volumetric flow rate	36
24	COV along the static mixing elements	36
25	Top view of mixing result	37
26	Mixing pattern	38
27	COV at varying viscosities	39
28	COV at varying volumetric flow rate	40
29	Number of elements necessary for sufficient mixing	41
30	COV at varying diameter	42
31	Modified design	43
32	Top view of modified design	43
33	Original design vs. modified design	44
34	initial component distribution	45
35	fig: COV for different designs	45

# List of Tables

1	Theoretical number of striation layers generated by commercial mixers . . . . .	7
2	Stadelmaier static mixer w/12 elements . . . . .	18
3	Paint Viscosities to evaluate . . . . .	21
4	Boundary conditions . . . . .	28
5	Sampled surfaces . . . . .	30
6	Grid sensitivity analysis . . . . .	34



# Nomenclature

## *List of Abbreviations*

<b>CFD</b>	Computational Fluid Dynamics
<b>CLI</b>	Command Line Interface
<b>COV</b>	Coefficient of Variation
<b>CPI</b>	Chemical Process Industry
<b>FEM</b>	Finite Element Method
<b>FOAM</b>	Field Operation And Manipulation
<b>FVM</b>	Finite Volume Method
<b>GAMG</b>	Generalised Geometric-Algebraic Multi-Grid
<b>GUI</b>	Graphical User Interface
<b>IPS</b>	Integrated Process System
<b>LPD</b>	Low Pressure Drop
<b>MATLAB</b>	Matrix Laboratory
<b>OS</b>	Operating System
<b>PI</b>	Process Industry
<b>SIMPLE</b>	Semi-Implicit Method for Pressure Linked Equations
<b>STL</b>	Stereo Lithography
<b>STR</b>	Stirred Tank Reactors

## *Greek Letters*

$\Gamma$	Diffusion coefficient	$[\text{m}^2/\text{s}]$
$\mu$	Dynamic viscosity	$[\text{Pa} \cdot \text{s}]$
$\nu$	Kinematic Viscosity	$[\text{m}^2/\text{s}]$
$\phi_a$	Ratio of added flow rate to total flow rate	$[-]$
$\rho$	Fluid density	$[\text{kg}/\text{m}^3]$
$\sigma$	Standard deviation	$[-]$
$\tau$	Viscous stress tensor	$[\text{Pa} \cdot \text{s}]$
$\varepsilon$	Strain rate tensor	$[-]$

## ***Roman Letters***

$C_f$	Friction coefficient	[–]
$c_i$	Point value composition	[–]
$c_m$	Mean composition	[–]
$D$	Conduit Diameter	[m]
$f$	Darcy friction factor	[–]
$K$	Number of data points in a sample	[–]
$L$	Mixing length	[m]
$N$	Number of elements	[–]
$n$	Number of striation layers	[–]
$p$	Pressure	[Pa]
$Q_a$	Volumetric flow rate of added material	[m <sup>3</sup> /s]
$Q_t$	Total volumetric flow rate	[m <sup>3</sup> /s]
$S$	Striation thickness	[m]
$S_0$	Initial striation thickness	[m]
$S_M$	Source term	[–]
$S_{max}$	Max. striation thickness after n elements	[m]
$V_{avg}$	Average flow velocity	[m/s]

# 1 Introduction

Mixing of substances has over the years been an essential operation in numerous engineering applications. It has had a significant impact in a wide range of industries such as pharmaceutical, biomedical, consumer product, petrochemical and paint industry. These industries utilize either active or passive methods to blend together two or more substances. In active mixing an additional moving structure or external source is used to stir the substances by the use of turbulent flow. Such sources are most commonly ultrasonic vibration, dielectrophoresis, electrohydrodynamic, electroosmosis, or magnetic force. Passive mixers on the other hand do not require any moving structures or energy, the process relies solely on diffusion where the flow is most often laminar. Obstacles are mounted inside a tube to cut, fold, twist, and re-combine fluid elements. [1].

It is well established since late 19th century that fluids which flow in a horizontal tube either are laminar or turbulent. Osborne Reynolds experimented and concluded this in 1883 [2]. In 2007, a review of his papers was conducted by Derek Jackson and Brian Launder at School of Mechanical, Aerospace, and Civil Engineering, University of Manchester [3]. Mixing in laminar flows is characterized by particle trajectories being parallel to the tube axis, low pressure drop per unit length, and no mixing in a given cross-section. In turbulent flows, mixing is present in a given cross-section perpendicular to the tube axis and the pressure drop per unit length is considerably higher [4].

ABB AS, a pioneering technology leader in industrial automation and robotics motion, has adopted the static mixing design in their paint robots. These are installed in automotive factories all over the world, where a broad range of different paint types and technologies are used to get the best paint quality on each car body. The applied paint can either be water-borne or solvent-borne and may be considered as one-component or two-component paint. Two-component paint is blended together in the process arm before it is applied to the car body. Two-component paint is composed by resin, which constitute the color of the paint compound, and a catalyst, which hardens the paint. Both components are fed into the process arm in two separate channels. Then, the components are mixed by a special valve block before it is applied to the car body by an atomizer. This type of block is known as a 2k-mixer, which is short for two-component mixer. Figure 1 illustrates one of ABB's IRB 5500-22 paint robot with the process arm and the atomizer.



Figure 1: ABB IRB 5500-22 Paint Robot [5]

The mixing process is essential to achieve a satisfactory paint result. It is important that the resin and the catalyst have been completely mixed before the resin-catalyst blend reaches the atomizer. The 2K-mixer must be able to rapidly alternate between the color of the fed mixes if the next object, e.g. a car body, needs a different color. Before the 2k-mixer is fed with another color resin, the tube must be completely cleaned. Thus, the residue in the tube from the previous mix is wasted. To limit the waste of paint, and consequently reduce the cost, the internal cavity of the 2k-mixer should be at its minimum.

This thesis concerns with a series of CFD simulations of an existing 2k-mixer. A parametric study is performed to investigate the mixing performances and to optimize the design. A range of viscosities are studied in addition to several design changes to the mixer itself. In short, this paper will try to answer the following questions:

- What is the optimal length of the mixing section?
- What is the optimal geometry of the mixing selection, to facilitate effective mixing and easy cleaning?
- Does the tube diameter affect the mixing result?
- How can residual paint before cleaning be minimized without affecting the mixing quality?

## 2 Static mixers

Motionless mixers, commonly known as static mixers, have over the last 50 years matured into an essential unit operation in the industry, especially in the chemical process industry (CPI) [6]. Its application is an attractive alternative mostly due to similar performance and sometimes greater results at lower costs compared to stirred-tank reactors (STR). Static mixers have lower energy consumption and requires less maintenance due to the lack of moving parts. Also, such mixers offer more controlled and scaleable rate of dilution in fed batch systems. Other advantages are the lower residence time for feed streams and that static mixers can be manufactured in a wide range of structural materials [7].

A static mixer's efficiency of mixing miscible fluids depends on its ability to radially mix and bring fluid elements together. These processes accelerate the effect of diffusion. Static mixers divide and redistribute streamlines sequentially in laminar flows by using the energy from the flowing fluid. The redistribution of flow is illustrated in Figure 2. In turbulent flows, the turbulence is enhanced in order to increase the mixing, up until the wall boundaries [7]. This thesis concerns primarily on the blending of miscible liquids in laminar flows. Static mixers can also be used to blend gases or immiscible fluids in laminar and turbulent flows, and in some cases, particulate solids, but this is not a part of this study.

The main principle of static mixing is the following: one places a series of identical motionless inserts called *elements* in a pipe to alter the flow field of the flowing fluid. The purpose of these elements is to cut, fold, twist, and re-combine fluid elements in the direction transverse to the main flow [6] [4]. The performance of these alterations is determined by the element design and number of elements in placed in series in the tube. Commercially available mixers consists of a wide variety of basic geometries with many adjustable parameters. These parameters can be optimized for each specific application. The most adjusted parameters is the number of elements in series. The aspect ratio, the ratio between the element length and diameter, is also an important parameter which is heavily adjusted. Commercial static mixers typically use standard values for various parameters that provides generally good mixing throughout a range of applications. Standard designs are based on available experimental data [7].

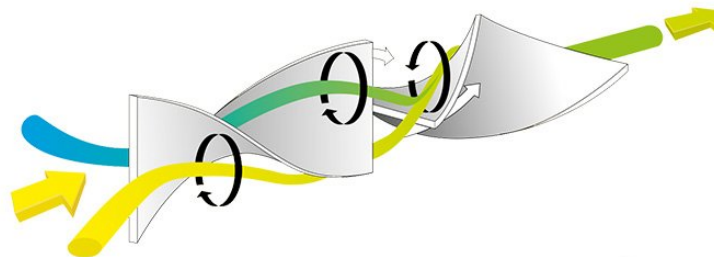


Figure 2: Static mixers sequential operations [8]

Initially, static mixers or dynamic mixers are selected before the performance and design process. For some cases, the choice is easily taken by the flow being continuous, but for more complicated processes this question becomes much more complicated. Myers et al (1997) tried simplifying this by creating a flowchart with the most important issues to consider when selecting the mixer [6]. A slightly edited version of this is presented in Figure 3.

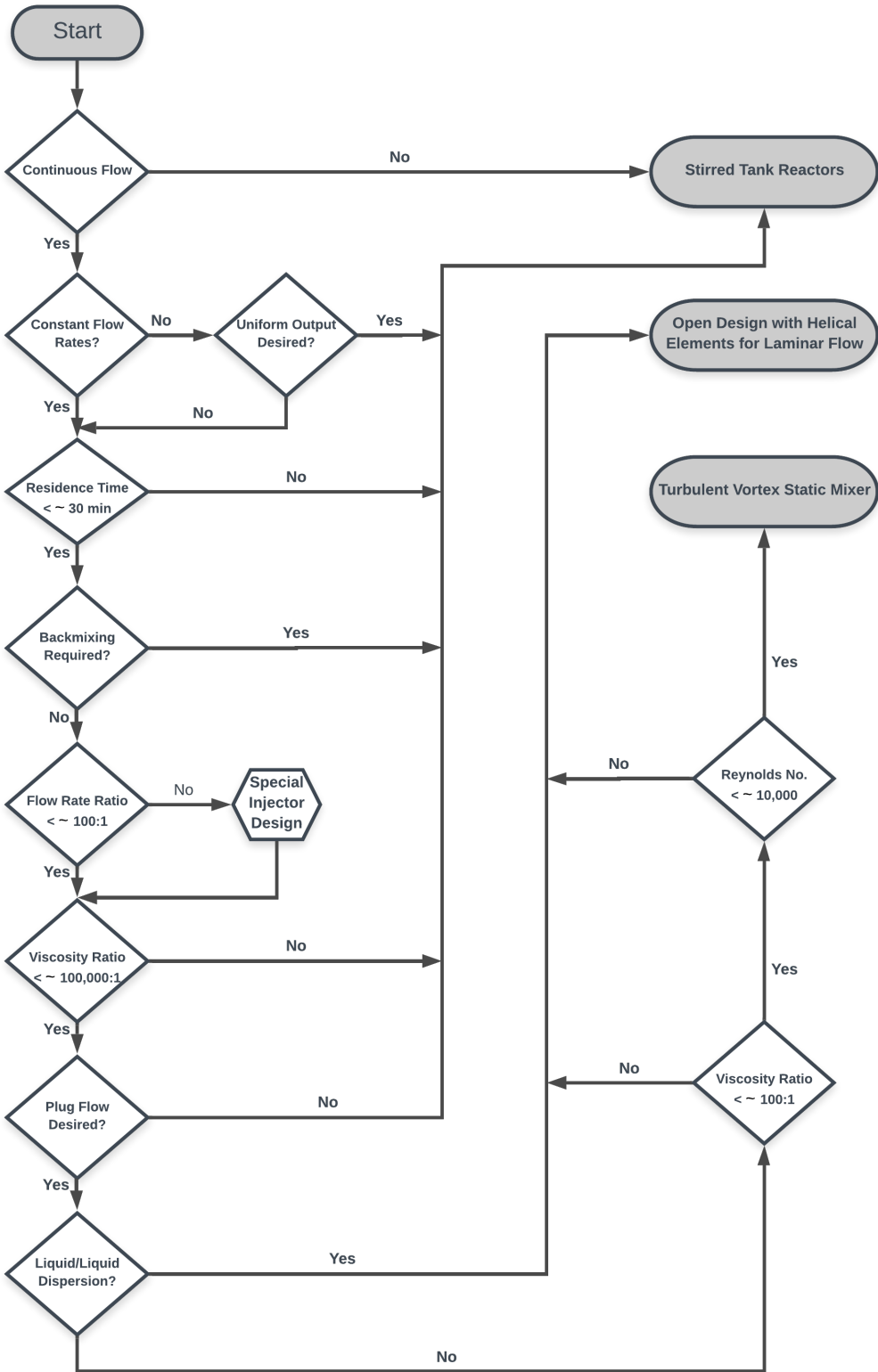


Figure 3: Most important issues to consider when selecting type of mixers

As mentioned previously, and illustrated in Figure 3, continuous flow is one of the main criteria for selecting motionless mixers. Dynamic mixers are more ideal for batch or semi-batch mixing. Dynamic mixers can be utilized for continuous process, but needs to satisfy a lot of requirements. Static mixers are ideal for use in processes that require short residence time ( $< 30$  min) while agitated vessels are more favourable for longer residence times, e.g. mixing of concrete. The absence of back-mixing in motionless mixers requires much more concentrated flow rates to achieve constant quality over time, on the other hand STR provides high rates of back-mixing which allows them to deliver constant quality over time even when the flow rates fluctuates over time. Even though static mixers can be used for a lot of applications, they are not very suitable for mixing fluids with extremely high viscosity ratio between the mixing substances ( $< \text{approx. } 100\,000:1$ ). In general, static mixers are used in application for where plug flow is desired [6].

## 2.1 *Mixing of miscible liquids*

High viscosity fluids struggle to blend together in a pipe due to pressure limitations which prevent turbulence. Imagine an empty pipe with two liquids, one black and one white where both occupies the same amount of space in a circular pipe. An illustration of this is visualized in Figure 4(a). In undistributed laminar flows, both liquids exits the pipe approximately the same as they entered. The only exception being close to the interface where a gray color can be observed. This gray color is due to molecular diffusion effects. There is no convective mixing in tangential or radial direction. The same concept applies when two liquids are separated radially, illustrated in Figure 4(b). In both cases, there is poor interaction between the liquids which will yield spatial inhomogeneties. Stand-alone diffusion can provide mixing in capillary tubes, but the effects radically lessen upon scaling which is rarely sufficient in any industrial-scale application. This is why conventional static mixers are designed to homogenize the fluids by redistributing the fluid in radial and tangential direction [7].

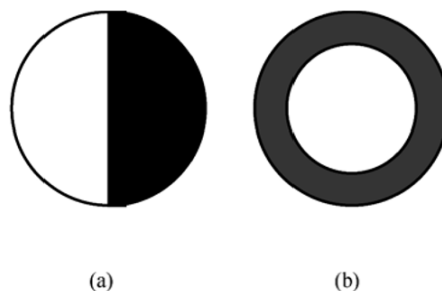


Figure 4: Spatial inhomogeneities [7]

Plug flow, also known as piston flow, is in fluid mechanics one of the models of a velocity profile for a fluid flow in a empty pipe. In plug flows, it is assumed that the velocity profile is constant across any cross-section perpendicular to the pipe axis. This model assumes that the boundary layer adjacent to the inner wall of the pipe does not exist [9]. Ideally, the black and white liquids

from Figure 4 are transformed into one uniformly gray liquid when exiting the pipe, and all the molecules emerging from the tube would have entered it together [7].

### 2.1.1 Striation thickness model

Striation thickness is used to quantify the mixing efficiency of static mixers. Figure 5 illustrates this by showing the striation thickness,  $S$ , decreases when the fluid is sheared perpendicular to the its initial striations. This causes the area between the black and white fluid to increase. When the displacement of the upper surface becomes large enough, the striation thickness,  $S$ , decreases below the human eyes visibility. When this happens the molecular diffusion will eliminate local concentrations. This mixing effects from simple shear becomes relative small when the striation has the same orientation as the shear. Greater efficiency is achieved when the shear direction changed accordingly, this ensures that the shear is always perpendicular to the striations. In a static mixer the flow is split, stretched, sliced and recombined when passing through one element. These mechanics further improves the efficiency. Figure 6 illustrates these mechanisms for a  $2^N$  mixer [7].



Figure 5: Response of striation thickness [7]

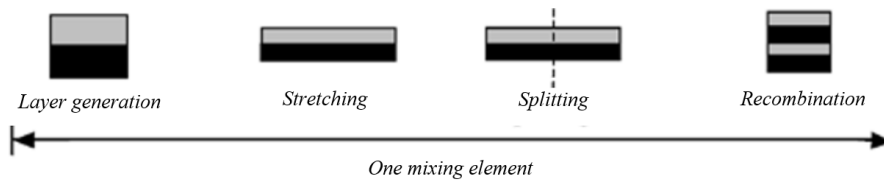


Figure 6: Mixing mechanisms [7]

The theoretical efficiency of static mixers can simply be quantified by the number of striation layers,  $n$ , static mixers generates. A simple helical static mixer, illustrated in Figure 2, increases the number of layers by a factor of 2 per element, for  $N$  elements the number of striation layers is given by  $2^N$ . Some commercially static mixers, i.e. ISG mixers, slices the incoming fluid into 4 layers per element. Due to this the mixer is classified as  $4^N$  for  $N$  elements. This higher number of striation layers comes at the cost of greater pressure drop. More complex mixers with multilayer design are also available such as CBMiM which also splits the fluid into a number of spirals in a tube,  $n_s$ , and number of layers in a multilayer design,  $n_p$ , in SMV mixers [10]. The theoretical numbers of striation layers generated by commercial static mixers are tabulated in Table 1 below.



Table 1: Theoretical number of striation layers generated by commercial mixers

Mixer type	Kenics Ross LPD Cleveland Komax	Bran and Lübbe Ross ISG KAM	Lightnin Inliner 50	CBMiM	SMV
Number of layers, $n$	$2^N$	$4^N$	$3 \cdot 2^{(N-1)}$	$> n_s \cdot 6^{3N}$	$n_p \cdot (2n_p)^{N-1}$

Even though mixers such as Ross low pressure drop (LPD) or Kenics (Figure 2) mixer has a lower theoretical mixing efficiency compared to some multilayer mixers such as Inliner 50, the number of layer growth is quite rapid. Figure 7 illustrates two miscible fluids entering a mixer classified as  $2^N$ . Both fluids occupy the same amount of space as described in Figure 4. Assuming that the tube diameter is 8 mm, after passing 5 elements, the fluids striation thickness,  $S$ , has decreased almost beyond human eyes visibility, where molecular diffusivity will eliminate concentration differences and the mixture becomes homogeneous. Note that this is not the actual behavior. In helical static mixers, the layers are more twisted due to the swirling motion in the mixing elements. This is represented in Figure 8 [11].

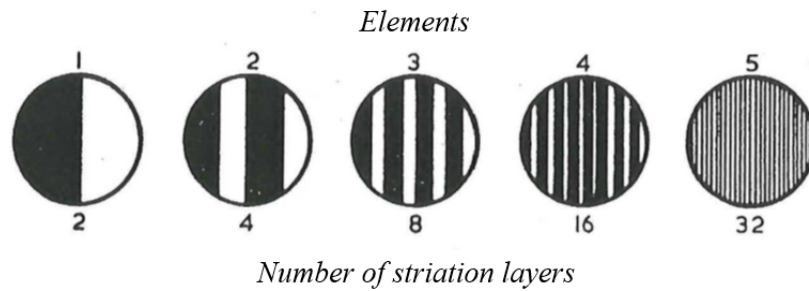


Figure 7: Schematical representation of the mixing of two fluids [11]

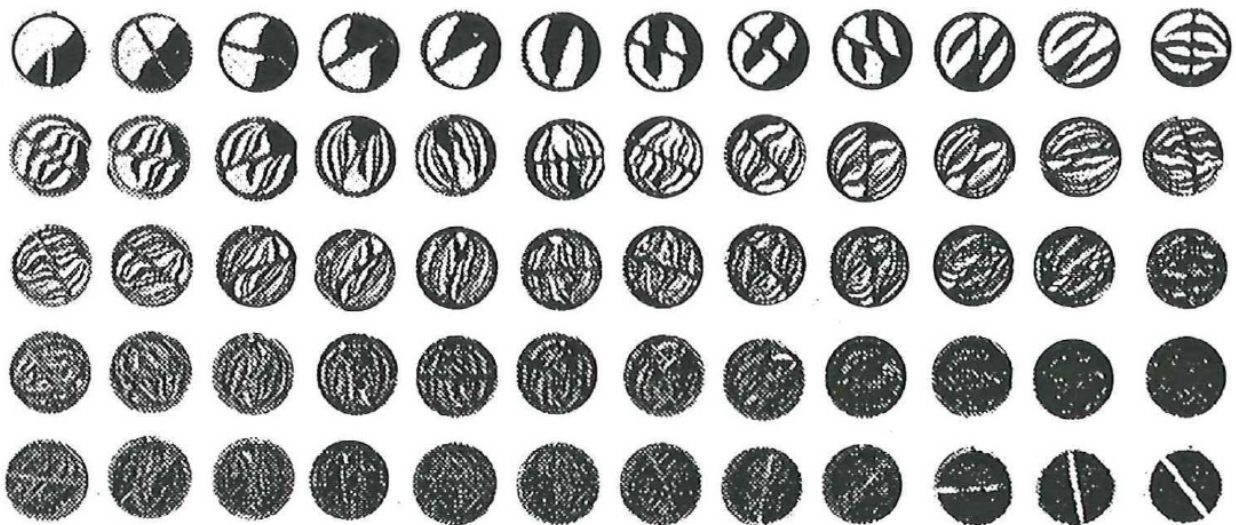


Figure 8: Mixing of black and white epoxy in a helical static mixer [11]

## 2.2 Reynolds Number in Static Mixers

Osborn Reynolds discovered after exhaustive experiments in the late 19th century, that the flow regime depends mainly on the ratio between the internal and viscous forces in fluids. This ratio is named after the man himself and referred to as the Reynolds number, this number can be expressed as:

$$Re = \frac{\text{Inertia forces}}{\text{Viscous forces}} = \frac{V_{avg}D}{\nu} \quad (1)$$

Where  $U_{avg}$  is the average flow velocity,  $D$  is the tube diameter for pipe, and  $\nu$  is the kinematic viscosity. The fluid flow is laminar, when the viscous forces are large enough to prevent the random and rapid fluctuation of the fluid. At large Reynolds numbers, the inertia forces are large relative to the viscous forces. This result in the viscous forces not being large enough to suppress the random and rapid fluid fluctuations to keep the fluid in line, thus the flow becoming turbulent [12].

Precise values of Reynolds number is desired for laminar, transitional and turbulent flows, Unfortunately this is not the case in practice. The transition from laminar to turbulent flow turns out to also be dependent on the degree of disturbance of the flow by surface roughness, pipe vibrations and fluctuations in the upstream flow[12]. Typically, fluids having viscosities greater than 0.1 Pa · s will be in laminar flow under most conditions in the process industry (PI). For flows in empty tubes,  $Re \leq 100$  is laminar with negligible contribution from the momentum term in the equation of motion. For  $Re \leq 2100$ , the flow is assumed to be laminar, but can have small disturbance that can lead to wake shedding and other oscillatory behavior. For  $Re \geq 2100$ , the flow is unstable and an assumption of turbulence is conservative for pressure drop calculations. Pressure drop in laminar flow is smaller than for turbulent flow. An assumption of turbulence is non conservative for  $2100 \leq Re \leq 5000$  when mixing is involved [7].

The same concept used for an open pipe can also be applied for static mixers of the open helical type with one exception. The transition values of the Reynolds number (Re) is lowered with a factor of 2. This means that the flow in static mixers are generally laminar for  $Re \leq 50$  and turbulent for  $Re \geq 1000$ . The reason for this is due to the element inserts causing systematic disturbance to the flow field. For  $50 \leq Re \leq 1000$ , the flow is in its transitional phase where complex but fairly reproducible flow behavior is expected [7].

## 2.3 Assessing mixing homogeneity

In the early days, mixing homogeneity were assessed mainly on the maximum striation thickness produce,  $S_{max}$ , after the flow has proceeded through  $N$  number of elements. This relationship can be determined by the following equation:

$$S_{max} = \frac{D}{n} \quad (2)$$

Where  $n$  is the number of striation created by the specific mixers tabulated in Table 1, and  $D$  is the conduit diameter. In recent years more sophisticated theories for evaluating mixing has emerged e.g. Danckwerts' scale of mixing performance of static mixers mentioned in [6]. There are also a great variety of parameters that can be used to evaluate mixing homogeneity, but these parameters are not defined properly and nor converted between each other. The striation thickness model given in Eq. 2 is still highly satisfactory from a theoretical point of view. This is primarily due to its independence of molecular diffusion and problems concerning sample size. Imagine two fluids that are identical except for some measurable characteristics e.g. color. In laminar flow, the mixing performance should exclusively depend on the initial distribution of the fluid at the mixing tube inlet, the geometry of the mixing element and number of elements in series. In transitional flow, mixing homogeneity may be dependent on the Reynolds number. Different physical properties like viscosity and volume fraction will also influence the mixing efficiency for miscible fluids. Even considering these changes the striation thickness models remains a well defined concept. If accurate tracking of striations and residence time can be achieved, calculations can be superimposed on a numerical solution. Unfortunately this is difficult for striations since they are difficult to measure. CFD calculations even struggle due to numerical diffusion [7].

### 2.3.1 Coefficient of Variation

Today, the more accepted approach for quantifying mixing homogeneity stems from taking simultaneous samples at various points over the conduit cross-sectional area at a fixed axial location. These measurements are used to calculate the coefficient of variation (COV), which is a more statistical approach [6]. This measure of uniformity takes the ratio between the standard deviation in composition,  $\sigma$ , and the mean composition  $c_m$ :

$$COV = \frac{\sigma}{c_m} \quad (3)$$

Where the lower COV indicates greater mixing. The standard deviation in composition,  $\sigma$ , and the

mean composition,  $x_m$  is given by the following:

$$\sigma = \sqrt{\frac{1}{K-1} \sum_{i=1}^K (c_i - c_m)^2} \quad (4a)$$

$$x_m = \frac{1}{K} \sum_{j=1}^K C_{m,j} \quad (4b)$$

where  $K$  is the number of measurement over the pipe cross-section, usually  $K \geq 9$ , and  $c_i$  is the point value composition of the  $i$ -th probe. Grosz-Röll stated a more general correlation for COV in his technology report in 1980 ([13]). He concluded that for a two-component laminar system that the COV becomes a function of three dimensionless variables:

$$\text{COV} = f\left(\frac{\mu_1}{\mu_2}; C_m; \frac{L}{D}\right) \quad (5)$$

where  $\mu_1/\mu_2$  is the viscosity ratio between the respected components and  $L$  is the total length of the static mixer [7]. For  $\mu_1/\mu_2 \approx 1$ , the COV becomes a function of the following variables:

$$\text{COV} = b \exp\left(-c \frac{L}{D}\right) \quad (6)$$

where both  $c$  and  $b$  are adjustable coefficients.  $b$  depends on  $c_m$  in laminar flows, but also on the flow velocity in turbulent flows. The coefficient  $c$  depends solely on the geometry of the mixer, and represent the rate of decrease of COV per unit of mixers length [7][10].

Experimental results usually present the COV relative to the feed COV at the mixer inlet,  $\text{COV}_0$ . This feed COV can be determined as:

$$\text{COV}_0 = \sqrt{\frac{1 - \phi_a}{\phi_a}} \quad (7)$$

here  $\phi_a$  represents the ratio of the volumetric flow rate of the added material,  $Q_a$ , and the total volumetric flow rate,  $Q_t$  [6]:

$$\phi_a = \frac{Q_a}{Q_t} \quad (8)$$

### 2.3.2 *Homogeneous mixing criteria*

Myers et al. (1997) writes in their report that most industrial blending applications can be satisfied with a  $COV \approx 0.05$ . However, blending of visual uniformity may require  $COV \leq 0.01$  e.g blending of color. Keep in mind that  $COV$  of 5% does not mean that all concentrations are within the range of 5% of the mean concentration. Rather, the standard deviation of the concentration is equal to 5% of the mean concentration ( $\sigma = 0.05x_m$ ). Assuming that the point concentrations are distributed normally about the mean, the  $COV$  can be related to the distribution of concentrations [6].

# 3 Governing Equations

In this chapter, the mathematical basis for a comprehensive general-purpose model of the fluid flow is presented. The governing equations of the CFD simulations are the following sets of equations known as some of the conservation laws of physics [14]:

- Conservation of (fluid) mass
- The rate of momentum change is equal to the sum of the forces on a fluid particle (Newton's 2nd law)
- The rate of change of energy equals the sum of the rate of heat addition to and the rate of work done on a fluid particle (1st law of thermodynamics)

The following equations are presented as from H. K. Versteeg's & Malalasekera's book in CFD [14]. For this 2k-mixer, the internal fluid is at all times liquid, therefore the fluid is perceived as incompressible. The equation derived from the 1st law of thermodynamics is not used when simulating the mixing performance of the 2K-mixer as heat and work exerted by the fluid are not pertinent results for the mixing performance. Therefore, it is not derived in this thesis.

## 3.1 Mass conservation

Mass conservation equation describes the mass balance for the fluid element. According to Versteeg & Malalasekera "Rate of increase of mass in fluid element = Net rate of flow of mass into fluid element" ([14], p.11). The rate of increase of mass in the fluid element is described by:

$$\frac{\partial}{\partial t}(\rho\delta x\delta y\delta z) = \frac{\partial\rho}{\partial t}\delta x\delta y\delta z \quad (9)$$

Imagine the simplest form of a hexahedral fluid element, as depicted in Figure 9. From the figure, one observes the expressions of the mass flow rate across a face of the element. This is the product of the velocity component normal to the face, area of the face, and density of the fluid. Incoming flow is signed positively and outgoing flow is signed negatively. The net rate flow of mass into the element is equal to the sum of incoming mass flow rates, minus the sum of outgoing mass flow rates, across its faces. From Figure 9, the net mass flow rate into an element is expressed mathematically as:

$$\begin{aligned} & \left(\rho u - \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y\delta z - \left(\rho u + \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y\delta z - \\ & \left(\rho v - \frac{\partial(\rho v)}{\partial y}\frac{1}{2}\delta y\right)\delta x\delta z - \left(\rho v + \frac{\partial(\rho v)}{\partial y}\frac{1}{2}\delta y\right)\delta x\delta z - \\ & \left(\rho w - \frac{\partial(\rho w)}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y - \left(\rho w + \frac{\partial(\rho w)}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y \end{aligned} \quad (10)$$

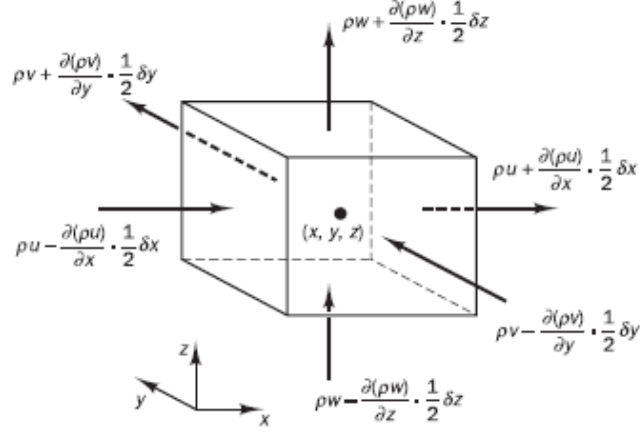


Figure 9: Mass flow in and out of a fluid element [14]

By arranging all the terms of mass balance to the left hand side and divide the element by its volume  $\delta x \delta y \delta z$  we get the unsteady, three dimensional mass conservation equation known as the continuity equation, given by the following equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0 \quad (11)$$

In a more compact vector notation where  $\vec{U} = (u, v, w)$ , the continuity equation can be written as:

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \vec{U} = 0 \quad (12)$$

where the first term is the transient, describing the rate of change of density. The second term is the convective term, describing the net flow of mass out of the fluid element [15]. For liquids, which are assumed to be incompressible, the density  $\rho$  does not change and the transient term is neglected. Hence the continuity equation for incompressible fluids is described by the following:

$$\vec{\nabla} \cdot \vec{U} = 0 \quad (13)$$

Since static mixers in most cases are placed in pipe structures, the continuity equation for incompressible fluids can be expressed in a cylindrical coordinate system. The continuity equation can be rewritten as [12]:

$$\frac{1}{r} \frac{\partial(r u_r)}{\partial r} + \frac{1}{r} \frac{\partial u_\theta}{\partial \theta} + \frac{\partial u_z}{\partial z} = 0 \quad (14)$$

## 3.2 Momentum conservation

The momentum conservation equation is based on Newton's 2nd law. Stated by Versteeg & Malalasekera; "the rate of change of momentum of a fluid particle equals the sum of the forces on the particle" ([14], p.14). One distinguishes between two types of forces acting on fluid elements, namely surface forces and body forces. Surface forces consist of pressure forces, viscous forces and gravity forces, whereas body forces consist of centrifugal forces, Coriolis forces and electromagnetic forces. Stress action on a fluid element is commonly defined in terms of pressure and the nine viscous stress components as visualized in Figure 10a. The normal pressure is denoted by  $p$  and the viscous stresses are denoted by  $\tau$ . The suffix notation is used to describe the direction. The suffix  $i$  and  $j$  indicates that the stress component acts in the  $j$ -direction on a surface normal to the  $i$ -direction.

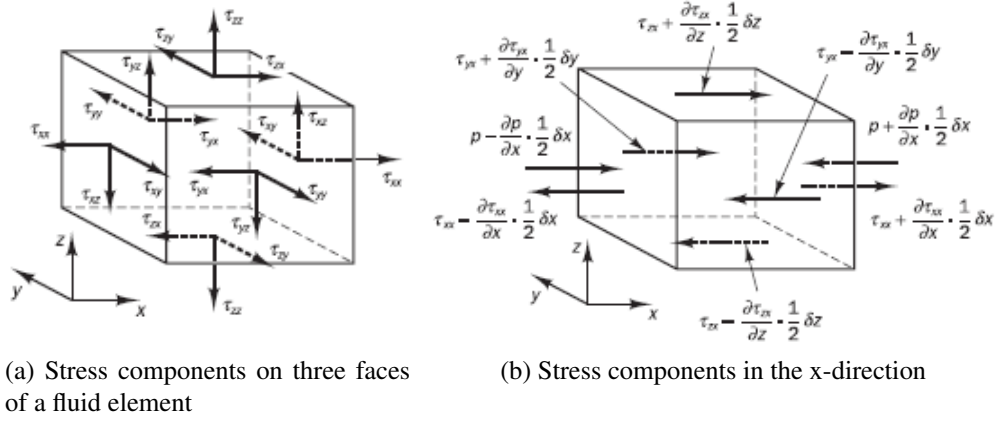


Figure 10: Stress components on a fluid element [14]

Considering all the components acting in the  $x$ -direction, namely the forces due to pressure  $p$  and the stress components  $\tau_{xx}$ ,  $\tau_{xy}$  and  $\tau_{xz}$  shown in Figure 10b above. The force from a surface stress action on a fluid element is a product of stress and area. Forces acting in the direction of a coordinate axis are given a positive sign and forces acting in the opposite direction of a coordinate axis are given a negative sign. The net force acting in the  $x$ -direction is expressed by taking the sum of the force components in this direction. The resultant net force in the  $x$ -direction from the stress components acting on the left and right face is given by:

$$\begin{aligned} & \left[ \left( p - \frac{\partial p}{\partial x} \frac{1}{2} \delta x \right) - \left( \tau_{xx} - \frac{\partial \tau_{xx}}{\partial x} \frac{1}{2} \delta x \right) \right] \delta y \delta z - \left( p + \frac{\partial p}{\partial x} \frac{1}{2} \delta x \right) \delta y \delta z \\ & + \left( \tau_{xx} + \frac{\partial \tau_{xx}}{\partial x} \frac{1}{2} \delta x \right) \delta y \delta z = \left( -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} \right) \delta x \delta y \delta z \end{aligned} \quad (15)$$

Similarly, the net force in the  $x$ -direction from the stress components acting on the front and back faces is:

$$- \left( \tau_{yx} - \frac{\partial \tau_{yx}}{\partial y} \frac{1}{2} \delta y \right) \delta x \delta z + \left( \tau_{yx} + \frac{\partial \tau_{yx}}{\partial y} \frac{1}{2} \delta y \right) \delta x \delta z = \frac{\partial \tau_{yx}}{\partial y} \delta x \delta y \delta z \quad (16)$$



The net force in the  $x$ -direction from the stress components acting on the top and bottom surfaces is:

$$-\left(\tau_{zx} - \frac{\partial\tau_{zx}}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y + \left(\tau_{zx} + \frac{\partial\tau_{zx}}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y = \frac{\partial\tau_{zx}}{\partial z}\delta x\delta y\delta z \quad (17)$$

By summing Eq. 15, Eq. 16 and Eq. 17, and dividing by the volume  $\delta x\delta y\delta z$ , the total force per unit volume on the fluid in  $x$ -direction is expressed as:

$$\frac{\partial(-p + \tau_{xx})}{\partial x} + \frac{\partial\tau_{yx}}{\partial y} + \frac{\partial\tau_{zx}}{\partial z} \quad (18)$$

The overall effect of the body forces can be introduced by adding a source term,  $S_{Mx}$ , at the end of Eq. 18. Similarly, the momentum conservation, i.e. the total force per unit volume, in  $y$ - and  $z$ -direction can be derived in the same manner as for  $x$ -direction. The momentum conservation in  $x$ -,  $y$ - and  $z$ -direction, including the source term, is written in full as the following set of equations. However, in this study, the overall effects of the body forces are neglected in the simulations and is therefore not discussed further in this thesis.

$$\rho \frac{Du}{Dt} = \frac{\partial(-p + \tau_{xx})}{\partial x} + \frac{\partial\tau_{yx}}{\partial y} + \frac{\partial\tau_{zx}}{\partial z} + S_{Mx} \quad (19a)$$

$$\rho \frac{Dv}{Dt} = \frac{\partial\tau_{xy}}{\partial x} + \frac{\partial(-p + \tau_{yy})}{\partial y} + \frac{\partial\tau_{zy}}{\partial z} + S_{My} \quad (19b)$$

$$\rho \frac{Dw}{Dt} = \frac{\partial\tau_{xz}}{\partial x} + \frac{\partial\tau_{yz}}{\partial y} + \frac{\partial(-p + \tau_{zz})}{\partial z} + S_{Mz} \quad (19c)$$

### 3.3 Navier-Stokes equations for Newtonian fluids

The governing equations still contain further unknowns, namely the viscous stress tensor,  $\tau_{ij}$ . This tensor in many fluid flows can be expressed as functions of the local deformation rate or strain rate. The fluids, which are assumed to be both incompressible ( $\rho = \text{constant}$ ) and isothermal (local changes in temperature are small or nonexistent) in this thesis [14]. Further, the dynamic viscosity ( $\mu$ ) and kinematic viscosity ( $\nu$ ) are set to be constant. Hence it can be shown that the viscous stress tensor for an incompressible Newtonian fluid with constant properties reduces to [12]:

$$\tau_{ij} = 2\mu\varepsilon_{ij} \quad (20)$$

where  $\varepsilon_{ij}$  is the strain rate tensor which will not be discussed any further. The algebra this composition introduces is thoroughly worked in [12] Chap. 4.4. In three dimensions, the nine viscous stress tensor yields:

$$\tau_{ij} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} = \begin{bmatrix} 2\mu \frac{\partial u}{\partial x} & \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & \mu \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & 2\mu \frac{\partial v}{\partial y} & \mu \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ \mu \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) & \mu \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) & 2\mu \frac{\partial w}{\partial z} \end{bmatrix} \quad (21)$$

Substituting viscous stress tensors (Eq. 21) into Eq. 19a, Eq. 19b and Eq. 19c yields after some rearrangements the so-called Navier-Stokes equations for incompressible and isothermal flow.

$$\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \mu \nabla^2 u + S_{Mx} \quad (22a)$$

$$\rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \mu \nabla^2 v + S_{My} \quad (22b)$$

$$\rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \mu \nabla^2 w + S_{Mz} \quad (22c)$$

In a more compact vector notation where  $\vec{U} = (u, v, w)$  and  $\vec{S}_M = (S_{Mx}, S_{My}, S_{Mz})$ , the Navier-Stokes equations for incompressible, isothermal flow can be written as:

$$\rho \frac{D\vec{U}}{Dt} = -\vec{\nabla} p + \mu \nabla^2 \vec{U} + \vec{S}_M \quad (23)$$

although this equation was derived using Cartesian coordinates, the Navier-Stokes equation is valid in any orthogonal coordinate system, i.e. cylindrical coordinate system.

### 3.4 Scalar Transport Equation

A close relation can be established by looking closer into the Navier-Stokes equation derived from the momentum equations and the continuity equation, which describes the mass balance. Introducing a scalar variable  $\Phi$  the conservative form of all fluid flow equations, also including the equations for scalar quantities can be described by the following expression for incompressible flow:

$$\frac{\partial \Phi}{\partial t} + \nabla \cdot (\Phi \vec{U}) = \frac{1}{\rho} \nabla \cdot (\Gamma \nabla \Phi) + S_\Phi \quad (24)$$

Where  $\Gamma$  represents the diffusion coefficient of diffusivity [16]. The first term of Eq. 24 is the transient term describing the rate of increase of  $\Phi$  of a fluid element. The second term is the convection term, describing the net rate of flow of  $\Phi$  out of the fluid element. The third term is the diffusion term which term describes the rate of increase of  $\Phi$  due to diffusion. The final term is the source term which describes the rate of increase of  $\Phi$  due to the source. It can be noted that in order to express Eq. 24, terms that are not common between the equations are hidden inside the source term. The scalar transport equation is also applicable for the energy equation.

## 4 2K-Mixer

ABB 2K-mixer valve block with membrane valve is specifically designed to provide constant and consistent mixing of two-component fluids for automatic coating applications. Most often, two types of fluids are brought together and mixed, e.g. resin containing the color and catalyst, and the hardening process will start when the fluids come in contact with each other. During this process, the fluids are pressurized through a static mixing tube accommodating 12 mixing elements or more. A precise mixing ratio between fluids and color elements is required to get a satisfactory result. Integrated process system (IPS) will monitor and control the mixture ratio and fluid rate parameters to ensure a precise and stable fluid regulation by making adjustments to the gear-pump system connected to the 2K-mixer. Note that the gear-pump will not assure a desired mixing efficiency.

A 2K-mixer is flexible regarding the number of color and catalyst inlets due to its modular design. The standard version comprise of three modules; one module containing the cleaning agent valve, one module with the catalyst membrane valve and one module with the color valve and dump valve. The modules and the inlets are illustrated in figure 11a with the 2K-mixer. Figure 11b illustrates the process diagram for the standard design with 1 color and 1 catalyst, furthermore exhibiting the possibility that multiple objects can successfully be painted with different colors if arranged in a painting line. This means the mixer must be cleaned for contamination in advance and between every color change. The cleaning agent inlet is therefore positioned at the rear end of the mixer to provide fast cleaning.

The schematics for the valves are represented in figure 11b. Notice the M-PAC 2/2 fluid valves are used to control the color inlet, cleaning agent inlet and dump outlet and membrane valve M-PAC 2/2 fluid controls the catalyst inlet. The mixer is rated for 25-100 mL/ min as minimum fluid flow, however, maximum fluid flow is depended on fluid viscosity, system pressure and the needed accuracy on the coated object and wear. Normal operating flow rate is typically between 400-500 mL/ min. For more, see appendix chapter A.

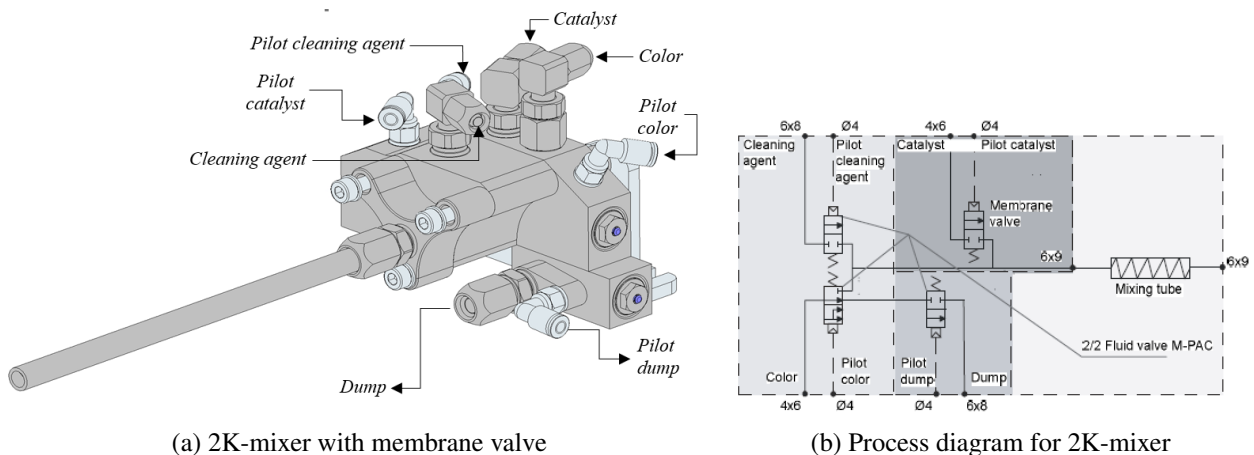


Figure 11: 2K-mixer

## 4.1 Stadelmaier static mixing element

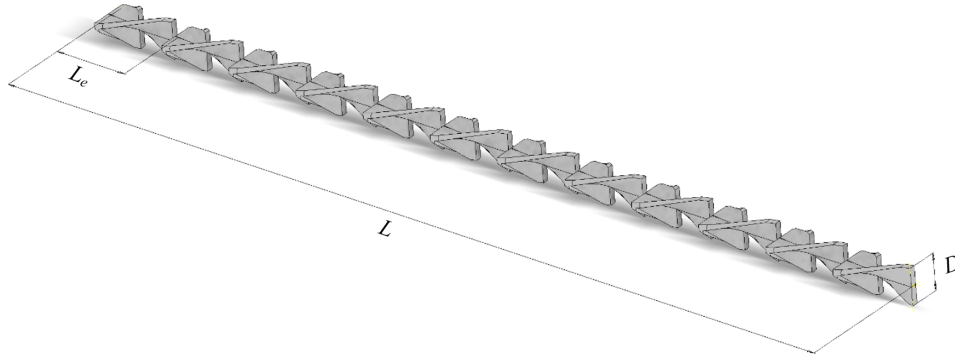


Figure 12: Stadelmaier mixing elements

The static mixer examined in this paper is provided by Stadelmaier GmbH and consists of an open pipe in which helical elements are mounted. The exact type is unknown and therefore not easily available, but products with similar design are widely used in industrial applications which incorporates laminar flow processes. The best known product of this design is believed to be the Chemineer's static mixer with a helical design. The exact model that is implemented in the 2K-mixer valve block is represented in 12, the mixer is composed by a series of elements where one edge is twisted 90 degrees over an element length, and the other edge is twisted 90 degrees in the opposite direction over a half element length. Each element is placed identically along the main axis. These mixing elements will henceforth be referred to as Stadelmaier static mixing elements.

The exact dimensions of Stadelmaier mixing elements placed in the 2K-mixer has an element length  $L_e$  of 12.6 mm with a conduit diameter  $D$  of 6.4 mm. This gives a ratio of approximately 1.97. The element design increases the fluid layers by a factor of two per element. According to Table 1, this results to the number of striation layers,  $n$ , to be 4096 as the numbers of elements,  $N$ , in use are 12 during the process. Subsequently, this means the maximum striation thickness expected according to Eq. 2 would be 1.6  $\mu\text{m}$  and not visible to the naked eye. The expected outcome will therefore be a homogeneous mixture. A summary is tabulated in 2 below.

Table 2: Stadelmaier static mixer w/12 elements

$L_e$ [mm]	$D$ [mm]	$N$ [-]	Ratio [-]	$n$ [-]	$S_{\max}$ [ $\mu\text{m}$ ]
12.6	6.4	12	1.97	4096	1.6

## 4.2 Simplified CAD model

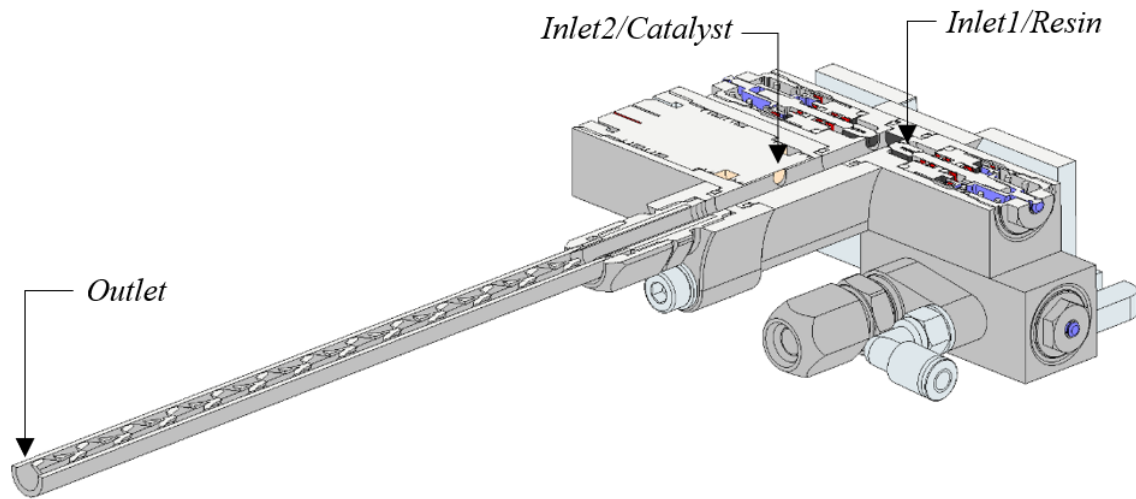


Figure 13: Half section view of the 2K-mixer

Figure 13 illustrates a half section view of the 2K-mixer with placement of the two main inlets for mixing marked. The resin module can be located at the rear end attached underneath the dump module. When cleaning the 2K-mixer, the valve that regulates the resin flow will be closed and cleaned separate from the 2K-mixer to ensure that there is no residue from the previous color in the valve during color change. The cleaning agent is drained through the dump module. The valve and inlet opposite of the resin inlet is the cleaning module. The main purpose of this component is to clean the main tube before any color change, therefore positioned at the back to assure the cleanliness of entire pipe during cleanup. This process will clean from the far rear of the pipe to the outlet in front making sure that no previous color residue will contaminate the following color. The cleaning module is kept closed during the main mixing procedure.

The module containing the catalyst is placed alongside the cleaning module, although further down the pipe. This design choice was to prevent the chance of getting any resin inside the catalyst line. Note, there are other option if the design version does not include said choice. One applicable method to reduce the chance of getting resin inside the catalyst line is to open the line seconds before the resin line. This ensures that the catalyst flow is still liquid when the resin module is opened

Stadelmaier mixing elements can be observed lying further down the main channel in Figure 13. This leaves additional space between the location where the catalyst enters the main tube and the point where the actual mixing starts. Some molecular diffusion is expected in this part of the process, just before entering the mixing elements. Generally, all static mixers are evaluated with the fluids occupying the same amount of space with the same velocity before entering the mixing elements. They are also shielded from each other with a thin wall to prevent molecular diffusion between the mixers before entering the mixing elements. Any diffusion could yield uncertainty in results.

The primary objective of this thesis is to optimize the mixing process in hopes of reducing the internal volume. This will reduce paint waste during cleaning, being one of the reasons for the simplified CAD model generated in Autodesk Inventor Professional 2018 (Figure 14). Observe from the half-section view how the main functionality is preserved to quantify the mixing performance of 2K mixer with the exclusion of the valves and the entire cleaning agent module. All other dimensions are identical compared to the original Stadelmaier design, and the Stadelmaier mixing elements is imported as a STL file from the 2K-mixers CAD model in Autodesk Inventor to ensure the exact geometry. Same mixing performance as the 2K-mixer, and less computation time for the mesh generation by only taking into account the desirable sections of the model rather than generating mesh for the entire 2K-mixer are other reasons that contributed to the design. Additional length to the main tube is added in the CAD file for simulation purposes.

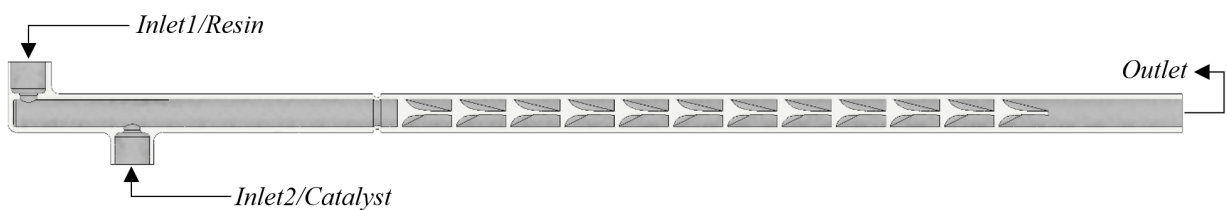


Figure 14: Simplified CAD model

The model is engineered with 3 different tube diameter, 4.5 mm, 5.5 mm and the original standard design of 6.4 mm. The 4.5 mm and 5.5 mm model is designed using Autodesk Fusion 360 by scaling the model down on the axis perpendicular to the main tube while keeping the last axis constant. This maintains the geometry of the Stadelmaier mixing elements similar to the original. All necessary dimensions in use are from the original standard model of the 2K-mixer. Technical drawing for this model can be found in appendix section B.

A Stereo Lithography (STL) file is generated from the CAD model for use in simulations. This file format represents a 3D object by breaking down the model into a logical series of triangular surfaces. The triangles are uniquely defined by its normal and three points representing its vertices. The orientation of the triangle normal gives the information about what is inside and what is outside of the solid body. The normal vector for each triangles points out of the solid model. Another important aspect about a STL file is that the adjacent triangles must have two vertices in common. Problems such as gaps in the STL file can exist if the normal vector points towards the interior of the solid model, or the adjacent triangles for some reason have not two common vertices. A complete listing of the x,y and z coordinates of the triangles and normal vectors represents the final geometry [17].

### 4.3 *Paint Viscosity*

The paint utilized in the 2K-mixer depends on the industrial application, in this case the car manufacturer. This means that the 2K-mixer has to deliver satisfactory mixing results over a broad range of different viscosities. Paint are usually measured using standard flow cups, which are designed to accurately measure the viscosity of liquids such as paint, inks, varnishes and similar products. The viscosity is measured by the time duration for the flow of a specific amount of fluid through an orifice. Time duration can also be used as a relative measurement and classification of viscosity. The physical measurement is then converted into dynamic viscosity using a viscosity calculator or available tables [18].

The flow cup type used to determine the different viscosities in this thesis is called DIN Cup #4. Only fluids that drains in 20 to 40 seconds from DIN Cup #4 is considered in this paper. Viscosity conversion table supplied by Norcross Corporation is used to convert the values into dynamic viscosity. The table is available at [19]. A copy of the conversion table can be found in Appendix C. The viscosity tested for the 2K-mixer is tabulated in the Table 3. Kinematic velocity can be derived by converting the kinematic viscosity to  $\text{Pa} \cdot \text{s}$ , followed by a division calculation by its mass density, assuming resin and catalyst components in the 2K-mixer have a mass density of  $1300 \text{ kg/m}^3$ . It is presumed that both components contains the same viscosity.

Table 3: Paint Viscosities to evaluate

<b>Din Cup #4</b>	<i>20s</i>	<i>25s</i>	<i>30s</i>	<i>35s</i>	<i>40s</i>
$\mu[\text{cP}]$	50	75	100	125	150
$\nu[\text{m}^2/\text{s}]$	$3.846e^{-5}$	$5.769e^{-5}$	$7.692e^{-5}$	$9.615e^{-5}$	$11.538e^{-5}$

# 5 OpenFOAM

Simulations of the static mixer are run in the open source CFD software called OpenFOAM. Open source Field Operation And Manipulation (OpenFOAM) is a free, open source CFD software primarily developed by OpenCFD Ltd. since 2004. The software has a large user base across a wide range of both engineering and scientific application. Since its launch back in 2004, OpenFOAM has grown to have an extensive range of features and solvers. The software is capable of solving almost anything from complex fluid flows with chemical reactions, turbulence, heat transfer, to acoustics, solid mechanics and electromagnetics [20].

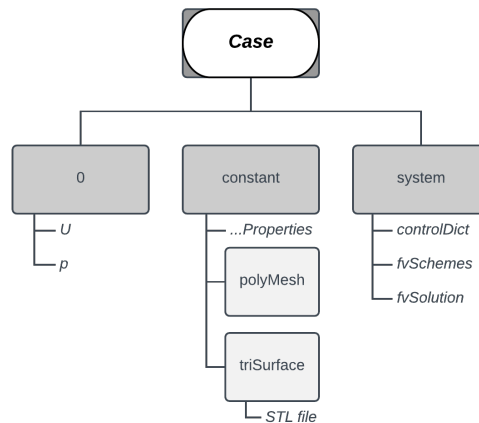


Figure 15: Basic file structure in OpenFOAM

Compared to other software, OpenFOAM does not contain a traditional graphical user interface (GUI). The software is primarily text-driven, written with the computer language C++ and runs in Linux based operating system (OS) from the command-line interface called terminal. The program is built up by a folder structure, which contains the sets of files which are built up by pieces of codes that is used by the solvers and utilities. For each case, a new folder structure is constructed containing a minimum of 3 folders, containing sets of files and sub-folders required to run an application. This structure is shown in Figure 15.

The system folder contains the setting parameters associated with the solution procedure itself. At least three files are included, namely controlDict, fvSchemes and fvSolution. Where controlDict contains run control parameters and parameters for data output. fvSchemes and fvSolutions contains the discretization schemes and solver controls for each field [21].

The constant folder holds files specifying physical properties for the application concerned, such as transportProperties or turbulenceProperties. It also includes polyMesh sub folder containing all the information needed for the case mesh, and triSurface sub-folder which includes a STL file if the mesh is generated using a physical CAD model. Various meshing commands is included in OpenFOAM, they are for most cases used to generate the polyMesh sub-folder. If these commands are used, separate files has to be added in the system folder that holds information about various parameters needed to generate a full description of the case mesh [21].



The 0 folder holds separate files which contains initial values and boundary conditions for the variables needed to solve the problem, e.g velocity and pressure as depicted in Figure 15. The files vary depending on the chosen solver and turbulence model defined in the constant sub-folder. As the simulation run, multiple folder is generated containing the solved fields solution for the fixed time or iteration dependent on the solver [21].

All the input files required to setup a base case for the simulation of the mixing performance of the 2K-mixer is presented in Appendix D.

## 5.1 Mesh Setup

Generating a good mesh is essential to achieve an accurate result from a CFD simulation. Mesh quality issues can have a substantial impact on the final result, and is one of the main challenges in modern CFD. Quality issues in worst case scenario impact the solution to a point where the solver diverges, which results in no solution at all. A mesh that is too refined increases the total number of cells resulting in increased computational work (e.g. number of iterations) adding to the overall cost of the simulation. Numerical inaccuracy can also occur with meshes having a lot of cells. Meshes can be generated using various pre-processing programs such as Pointwise, Gmsh, CENTAUR or Harpoon which are mesh generation software able to create meshes that can be imported into OpenFOAM. These programs tend to use arbitrary unstructured or polyhedral meshes, allowing for a wide variety of mesh problems; non-orthogonality cells and face skewness cells being the most common. The detail of modelling being used for the simulations decides the acceptable level of mesh quality [22].

Given its importance, a lot of effort work has been put into developing metrics to quantify mesh quality and methods to improve the overall quality. Numerous individual metrics have been proposed with reference to finite element method (FEM) meshes. Commercial and open source practice has tended to utilise metrics like non-orthogonality and cell skewness, when generating meshes using finite volume method (FVM). Exactly which meshing strategy to adopt, depends on the exact problem and it is possible that no universal meshing solution is possible. However, in many areas of CFD analysis there is an interest to generate automated meshing of existing CAD geometries. An example of an automated meshing system is *snappyHexMesh* which is a part of the OpenFOAM software package [22]. This automated meshing tool is used to generate the mesh needed to simulate the simplified CAD model discussed in chapter 4.2. The user provide a STL file containing the desired geometry to be meshed and a background mesh. The background mesh, typically contains a simple hexahedral block mesh.

### 5.1.1 *blockMesh*

The background mesh is generated using the mesh generating utility supplied with OpenFOAM called *blockMesh*. Mesh is generated from a file named *blockMeshDict* placed in the system folder

of a case. The *blockMesh* command generates a set of files to the polyMesh folder located in the constant folder. Principles that lies behind the *blockMesh* command is to decompose the main domain geometry into a set 1 or more three dimensional hexahedral blocks. To do so, the block is defined by 8 vertices specified by the local coordinate system  $(x, y, z)$  that be ordered counterclockwise as depicted in Figure 16. The defined blocks definition is contained in a list named blocks. Another important aspect defined in blocks list is a vector which contains the number of cells that is to be generated in each direction. This is what defines the number of cells that is to be generated in the block domain. The cell expansion ratios are also defined in the block list [23].

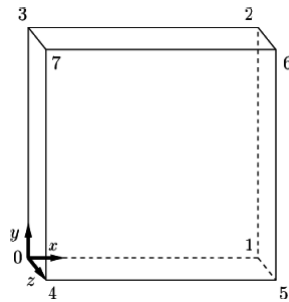


Figure 16: Block structure for the geometry [24]

Boundary patches can also be defined in the *blockMeshDict* file. The boundary of the generated mesh is defined in a list named boundary. The boundaries is defined into regions, where each patch in the list are defined by a keyword as a reference for other files. Each boundary patch contains information on the type of patch on which some boundary conditions are applied or a particular geometric condition, and the face bounded by 4 vertices [23]. In our case three patches is defined, two for the inlets and one for the outlet. Rest of the faces are undefined as they in later stage gets deleted while using the *snappyHexMesh* for generating the desired mesh. It is important that faces are flush with the geometry to create an inner volume mesh, which is the case for our geometry. The background mesh is placed just slightly inside the two inlets and outlet, this is done so that the boundary patches that are defined, they do not get deleted when creating the final mesh and have a closed volume inside the pipe *snappyHexMesh* to operate in. If this is not done, *snappyHexMesh* is going to generate a volume mesh containing the box shape created by the *blockMesh*. An overall results of the background mesh with its named patches are shown in Figure 17.

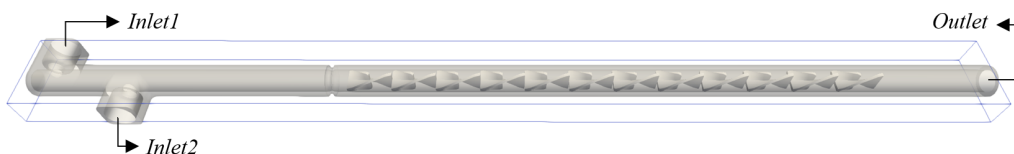


Figure 17: blockMesh

### 5.1.2 *snappyHexMesh*

As briefly mentioned in chapter 5.1, the *snappyHexMesh* utility generates automatically a mesh provided that the user provide a STL file and a background mesh. This automated utility generates 3-dimensional meshes containing hexahedral and split-hexahedral domains from a STL file or a OBJ file located in the triSurface sub-folder in the constant folder [25]. *snappyHexMesh* performs a three step meshing operation of castellation, snapping and boundary layer refinement.

In the castellation step, some cells are intersecting with the geometry surfaces. These cells are then refined by splitting them into smaller refined cells. This refinement process is controllable by the user defining a minimum and maximum level of refinement. The refinement process is done by defining a point in the area for where the mesh is going to be created. When the refinement process is done, all cells that lies outside of the desired geometric domains are deleted e.g. for a car this would be cells placed on the interior of the STL geometry. In the snapping process, cells are snapped to the STL surfaces using an iterative process of mesh movement. The number of iterations and specific mesh quality constraints are input parameters for the user, these parameters controls the cell refinement and face merging. In the final but optional step, cell layers can be added to grow a boundary layer close to the surface by specifying in advance number of input parameters. The process as a whole is a robust and automated process. The downside being that the user needs to specify a large number of input parameters provided in advance in a file called *snappyHexMeshDict* in the system folder. All the main steps in *snappyHexMesh* is visualized for a simple car model in Figure 18. Additionally a *meshQualityDict* file can be added to the system folder with meshing criteria for the mesh quality.

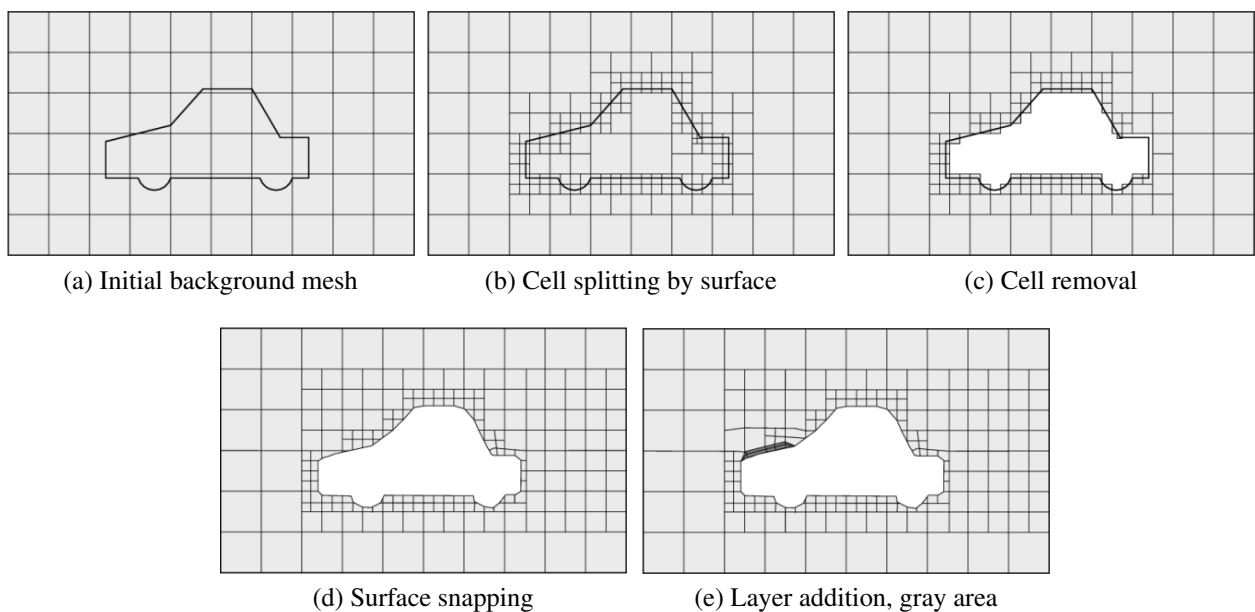


Figure 18: *snappyHexMesh* process [25]

For meshing the simplified CAD geometry, the STL file is loaded into the triSurface sub-folder for the *snappyHexMeshDict* file to be able to read the geometry. All the geometry surfaces are defined as wall patch for which boundary conditions are going to be applied. The minimum and maximum refinement level are set at level two. The point for where we want the mesh to be created is set to be inside the pipe, since the volume inside the pipe is bounded by the patches at the inlets and outlet, this is going to create a volume mesh of the inside structure of the STL file and delete everything that lies outside the desired region. The snapping process is set to implicit, and standard values for the iterative process and provided by OpenFOAM software. The snapping process however can also be expressed explicitly by including the surfaceFeatureExtract feature in OpenFOAM. This is more common to enhance the snapping process for extremely complex geometries. Finally the boundary layer section is not utilized as its believed to have a really small impact on the final result due to the low velocities that are simulated.

## 5.2 *Computational Setup*

Liquids mixed together in the 2K-mixer is assumed to be Newtonian, incompressible and have the same viscosity. With all these assumptions in mind, the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) solver is employed in this thesis. This is a steady-state solver for incompressible flows with turbulence modeling [26]. This solvers employs the SIMPLE algorithm to solve the continuity equation (see Eq. 13) and the momentum equation (see Eq. 19 a-c) [27]. It does so by generating the pressure from the velocity components by applying the Navier-Stokes equations (see 23) combined with an iterative procedure [28]. The algebra behind the SIMPLE algorithm is not presented in this thesis, but are thoroughly worked in H. K. Vetsteeg's & Malalasekera's book in CFD ([14]) in chapter 6.4 for two dimensional laminar steady flow equation in Cartesian coordinates.

Maximum flow velocity inside the 2K-mixer is assumed to be 1000 mL/min, thus the maximum Reynolds number for the mixed liquid being smaller than 56 and therefore the flow is assumed to have laminar behavior. Reynolds number can be calculated using Eq. 1. To be able to specify the turbulence model in OpenFOAM a keyword of the selected model is employed. In this thesis the laminar flow model is employed, the keyword for this model is simply "laminar". This is defined as the keyword for the simulationType in a file called turbulenceProperties located in the constant folder.

In OpenFOAM, the kinematic viscosity is defined in a file named transportProperties located in the same folder as the turbulence model. Additionally to the kinematic viscosity, the keyword for Newtonian fluids, which is "Newtonian" is set as the the transport model.

## 5.2.1 Boundary Conditions

In OpenFOAM, all the initial field values and boundary conditions are stored in the 0 folder as separate text files. It is important to remember from the meshing process which boundary entities are defined as patches and which are defined as walls. The internal field values are all set to be zero. Boundary conditions applied in this case are listed in Table 4 below for a flow velocity of 440 mL/min. Due to the fluids being incompressible, The velocity components in m/s is derived by assuming that there is 10 times more resin than catalyst in the final mixture. Hence, 400 mL/min for the resin inlet (inlet 1) and 40 mL/min for the catalyst inlet (inlet 2). By converting the volumetric flow rate to SI units and divide by the inlets cross-section area, which has a diameter of 8 mm, yields the velocity components at the inlets.

The type zeroGradient boundary condition is applied for the inlet patches and pipe walls for the pressure field, outlet patch for the velocity field, and pipe walls patch for both the scalar fields resin and catalyst. This boundary condition uses a zero-gradient condition the internal field onto the patch faces. The fixedValue boundary condition is applied to the outlet patch for the pressure field, both inlets for the velocity field and both inlets for the scalar fields with specified uniform values. The fixedValue boundary condition supplies fixed value constraints at the patches. The no-slip condition is applied to all the walls for the velocity gradient. This is done by defining the boundary condition type noSlip to the pipe walls patches for the velocity field. Finally the inletOutlet boundary condition is utilized for the outlet patch for both scalar fields. This boundary condition provides a generic outflow condition, with specified inflow for the case of return flow [29].

Table 4: Boundary conditions

	<b>Inlet 1</b>	<b>Inlet 2</b>	<b>Outlet</b>	<b>Pipe walls</b>
<b><i>p</i></b>	type zeroGradient	type zeroGradient	type fixedValue value uniform 0	type zeroGradient
<b><i>U</i></b>	type fixedValue value uniform (0,0,0.3316)	type fixedValue value uniform (0,0,-0.03316)	type zeroGradient	type noSlip
<b><i>Resin</i></b>	type fixedValue value uniform 1	type fixedValue value uniform 0	type inletOutlet inletValue uniform 0 value uniform 0	type zeroGradient
<b><i>Catalyst</i></b>	type fixedValue value uniform 0	type fixedValue value uniform 1	type inletOutlet inletValue uniform 0 value uniform 0	type zeroGradient

## 5.2.2 Time and data input/output control

All solvers in OpenFOAM begin their runs by setting up a database. The database controls I/O, where time is an inextricable part of the database considering the time intervals during each run when requesting output data [30]. This information is stored in a separate text file called *controlDict* located in the system folder. Settings set in *controlDict* is essential in the creation of said database. With the inclusion of simpleFOAM, the solver becomes a steady state solver and therefore allow an iterative approach. Note, time step must be set to 1 in order to run the adapted solver.

A lot of settings can be specified in this file, but since only small modifications were done, this section covers only the changes made from the tutorial case. Firstly we change the number of iterations to 500, this is done by changing the *endTime* settings. *writeControl* is changed to 20, this writes out the field data for every 20th iteration. At last, the *purgeWrite* is set to be 5. This settings only keeps the last 6 saved iteration and delete the rest. By doing this, the simulation size do not require a huge storage space.

Functions Objects can also be added in the *controlDict* file additional to the database controls I/O and time settings. These are looked into and specified in the following section.

## 5.2.3 Added function Objects

Ever since the OpenFOAM version 4.0, the "non-GUI" post-processing tools have been combined within a single command line interface (CLI) [31]. This functionality gives the user the ability to define numerous post-processing functionality such as writing out the residuals of every iteration step or even running an entire different solver in parallel during the running of a simulation, e.g. scalar transport solver.

In our case, three different functions are defined for our case. The *residuals*, *scalarTransport* and *surfaceFieldValue* functions. These functions are defined in the *controlDict* file under the functions panel.

The function object *residuals* allows the user to print out solvers performance for a list of fields [32], i.e. velocity field . For the current case, velocity (U), pressure (p) and both scalar fields (Resin and catalyst) is specified to be printed out. By monitoring the information written out the user can deduce whether or not the solution has converged. The residual should approach zero when the solution converges. The number of iterations indicates how many time the matrix is solved for the current equation [15]. The residuals can be monitored simultaneously by plotting out the result using *foamMonitor* command in the terminal windows as the simulation runs to easily deduce when the solution has converged.

The *scalarTransport* function object enables the *scalarTransportFoam* solver to run in conjunction with the used solver [33]. By adding this functionality, Resin and Catalyst fields can be separated

to analyse the effects if mixing of the two components. To do so, the user needs to specify the boundary conditions, numerical scheme and the diffusion coefficient for each scalar field. In the current case boundary conditions for both resin and catalyst is specified in separate text files in the 0 folder, numerical scheme used for the scalars are defined in the text file named *fvSchemes* and the diffusion coefficient is specified alongside the function in the *controlDict* file. The diffusion coefficient is set to be 1.0E-9 for both scalar fields.

The last function object that is added is the *surfaceFieldValue* function. This function object provides options to manipulate the surface field data into derived forms, such as summation, minimum, maximum or coefficient of variation (COV) [34]. *surfaceFieldValue* enables the user to define a plane using a base point and its normal vector, on which calculations that can be made. A total number of 13 planes is defined on which COV calculations is calculated. Each plane is placed after each element and before the first element. All the defined planes from where the COV calculation is calculated from is defined in Table 5 and is also depicted in Figure 19 below. Note that the base point for the sampled surfaces is given in *mm* and in the *controlDict* under functions is written in *m* in Appendix. This is because OpenFOAM operates in SI units.

Table 5: Sampled surfaces

Plane #	Plane 0	Plane 1	Plane 2	Plane 3	Plane 4	Plane 5	Plane 6
$(x,y,z)$ [mm]	(34.6, 0, 0)	(47.2, 0, 0)	(59.8, 0, 0)	(72.4, 0, 0)	(85.0, 0, 0)	(97.6, 0, 0)	(110.2, 0, 0)
Plane #	Plane 7	Plane 8	Plane 9	Plane 10	Plane 11	Plane 12	
$(x,y,z)$ [mm]	(122.8, 0, 0)	(135.4, 0, 0)	(148.0, 0, 0)	(160.6, 0, 0)	(173.2, 0, 0)	(186.0, 0, 0)	

At last, a pre-defined function for calculating the volumetric flow rate at the outlet is defined. This is just added to ensure that the initial calculations of the inlet velocity is done correctly, and to see how many iterations the solvers has to calculate before getting a steady output. The pre-defined function object is just the *surfaceFieldValue* function object defined to calculate the volumetric flow rate of the velocity field on a patch. Since the flow is incompressible, the outlet flow should be the same as adding together the volumetric flow rate of both inlets.

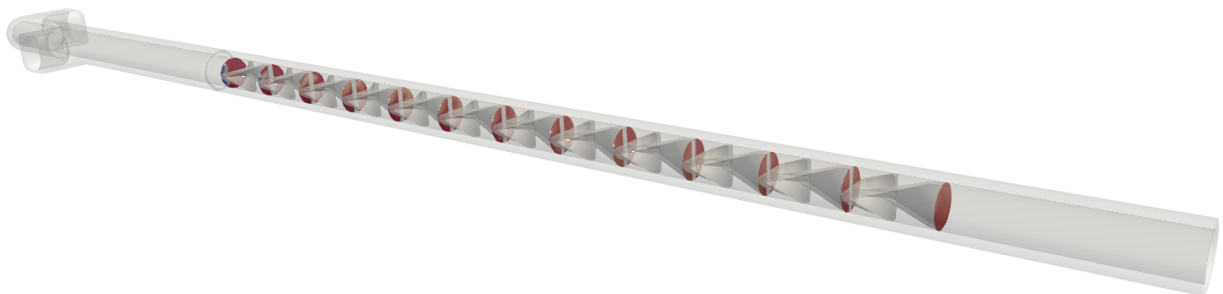


Figure 19: Sampled surfaces



## 5.2.4 Numerical schemes

In OpenFOAM, the numerical schemes are defined in a text file called *fvSchemes* placed in the system folder. This text file sets the numerical schemes for terms, e.g. derivatives in equations, which are calculated during a simulation such as time scheme, gradient, divergence, laplacian, interpolation, component of gradient normal to cell faces and distance to wall calculation [35].

Firstly, the time derivative scheme is specified as steady state as this is not applicable for this case. This sets all the time derivatives in the governing equation to be zero [35]. In other words, the flow does not change over time. Remember that the SIMPLE solver is a steady state solver.

The default gradient scheme is specified as Gauss linear, which is a method of discretization of the divergence terms. Gauss entry specifies the standard finite volume discretization of Gaussian integration. This entry requires an interpolation scheme which is specified with the keyword linear, this means linear interpolation or central differencing. However, the velocity gradient scheme is overwritten to cellLimited Gauss linear 1. This particular scheme limits the gradient such that when the cell values extrapolated to faces using the calculated gradient. This ensures that the face values do not fall outside the bounds of values in surrounding cells. The number at the end specifies the limit coefficient, in this case 1. This is done to improve boundedness and stability. This has been done in some tutorial cases, especially for cases involving poor mesh quality [35].

No default convective scheme is specified for this case, however the bounded Gauss linearUpwind limited convective scheme is used for the velocity field and both scalar fields. The interpolation scheme used is linearUpwind, which is second order upwind biased scheme. Since we are working on a steady state case, bounded is added in front of the discretization scheme as this helps maintain boundedness of the solution variable and promotes better convergence [35].  $\text{div}((\mu_{\text{Eff}} * \text{dev2}(T(\text{grad}(U))))))$  which is a part of the momentum equation works only with Gauss linear [15].

Gauss linear corrected scheme is set as the default for the laplacian scheme and not overwritten by any field like the gradient scheme. The word corrected after Gauss linear ensures an explicit non-orthogonal correction [15].

The interpolation scheme is set as linear, and component of gradient normal to cell faces is set as corrected.

### 5.2.5 *Solution and algorithm control*

Equation solvers, tolerance and algorithms are controlled from a text file named *fvSolution*. This file can be found under the system folder. Equations solved by CFD results usually appear in large matrices, mostly built by zeros so traditional algebraic methods become inefficient and iterative methods is favoured. Generalised geometric-algebraic multi-grid (GAMG) solver is utilized for pressure. For velocity and both the scalar field (Resin and Catalyst), smoothSolver is used. GAMG requires a positive definite, diagonally dominant matrix to operate and smoothSolver requires a smoother. symGaussSeidel smoother is applied. Additionally to the solvers and smoother, tolerance for how precise the solution is based on the initial residuals. A relative tolerance is also specified, this setting specifies how accurate the solution is for each iteration step [15]. Settings such as convergence criteria can also be defined in this file so that the solver automatically stops when certain residual criteria is met, but since we want to run the simulation until steady-state no convergence criteria is specified.

# 6 Results & discussion

The following chapter will summarize the results of the OpenFOAM simulations. As mentioned in the introduction, the different simulations explore the effects of velocity and viscosity on the mixing efficiency of the current design. Suggestions on how to optimize the design for greater efficiency is also discussed.

The results will be presented and discussed simultaneously with cross-reference throughout this thesis. Note that all the graphs and figures represented in this chapter is generated for a viscosity of 125 cP, standard diameter of 6.4 mm and a mean flow rate of 440 mL/min if nothing else is specified. Flow rate of 440 mL/min is within the mean flow range of the IRB 5500-22 paint robot. All raw COV values used to generate the plots are presented in appendix chapter E.

All the figures represented is produced using ParaView, which is a multi-platform post-processing data analysis and visualization application. The graphs and distribution plot is generated in Matrix Laboratory (MATLAB) R2018b by extracting raw data produced by the function objects discussed in chapter 5.2.3.

## 6.1 Meshing results

As discussed in chapter 5.1 Mesh Setup, a good mesh is essential to acquire accurate and sufficient results as poor mesh equals bad results. The automated meshing algorithm *snappyHexMesh* is used to generate a structured mesh. This process involves a trial and error approach to get a desirable mesh quality. The main goal here is to create a mesh that represents the internal volume of the tube geometry accurately.

Unfortunately *snappyHexMesh* was not able to snap to the geometry surface properly due to corrupt STL files generated by Autodesk Inventor. Another complication that occurred was OpenFOAM creating 17.000 patches for unknown reasons, resulting in the mesh taking close to 19 hours to generate. This was later solved by generating the STL file in Autodesk Fusion 360 instead. However, issues with snapping performance of the *snappyHexMesh* was not resolved and therefore creating a mesh with jagged surfaces instead of smooth surface close to the walls. The result is depicted in Figure 20a.

From Figure 20b, it can be observed that the mesh refinement is high close to the edges. An increase to the number of snap iterations or to the refinement lvl will in theory enable work around the jagged mesh, but both attempts resulted in more computational time with minimal changes to the overall quality of the mesh. The creation of jagged surface can be a result of gaps in the STL file when exported from Autodesk Inventor or Fusion 360. Jagged surfaces could also be the result of improper importation of original STEP file of the 2K-mixer into Autodesk Inventor.

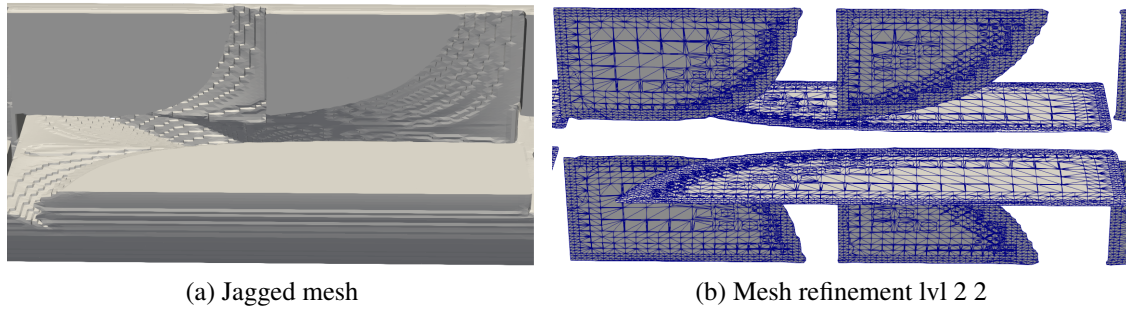


Figure 20: Mesh quality

Furthermore, generating a new STL file with less quality in Autodesk Inventor did not improve the snapping process either. A work around could be to generate a parasodic text file and use pre-processing softwares such as Pointwise, mentioned in chapter 5.1, to produce a sufficient mesh which can be imported to OpenFOAM or other CFD applications. As for this thesis, the number of cells in the background mesh was increased to minimize the jagged surfaces which leads to increased meshing time.

## 6.2 Mesh convergence test

Despite all efforts to keep the mesh size small as possible, the final mesh requires a significant amount of computational power and time to be solved. To help speed up this process, the simulations was ran on the University of Stavanger's cluster. All simulations used varying grid sizes to ensure that simulation results are independent from the number of grid cells, in addition to viscosity value of 125 cP with a flow rate of 440 mL/ min. The grid convergence test is performed by changing the number of cells in x,y and z directions in the background mesh, allowing the use as a reference for the plots. A total number of 4 different mesh are studied with the simulation marked as 100 % in Table 6 is the base or a foundation for the grid convergence test. The different grids are calculated by increasing or decreasing the number of cells by 10 % of its original value from each axis.

Table 6: Grid sensitivity analysis

Percentage variation og grid resolution	Number of cells in (x,y,z)	Number of cells in background mesh	Refinement lvl (min. max.)	Number of cells after meshing
80 %	416x24x48	479 232	(2 2)	1 608 278
90 %	468x27x54	682 344	(2 2)	2 085 832
100 %	520x30x60	936 000	(2 2)	2 630 880
110 %	570x33x68	1 241 460	(2 2)	3 143 132

A total number of 500 iterations is set to be the maximum number of iterations for the solver as this is believed to be enough for the solution to converge. The residual is closely monitored to see when the solution has converged. The residuals for the base mesh (100 %) is presented in Figure 21, although the residuals for both scalar fields were unfortunately not noted. However, they were closely monitored on the screen and had equally low values as the pressure and velocity field were presented.

Reason for why the scalar fields residuals were not printed may be that they are calculated using the function object instead of a dedicated solver for scalar fields. Note that the pressure field has a higher value than the velocity at the start, but after approximately 120 iterations, they are the same and converges around 150 iterations. After 120 iterations the pressure residual starts to oscillates, but the mean value remains the same. The velocity residuals keep declining, but since the residuals is as low as  $10^{-6}$  the simulation is stopped after 200 iterations.

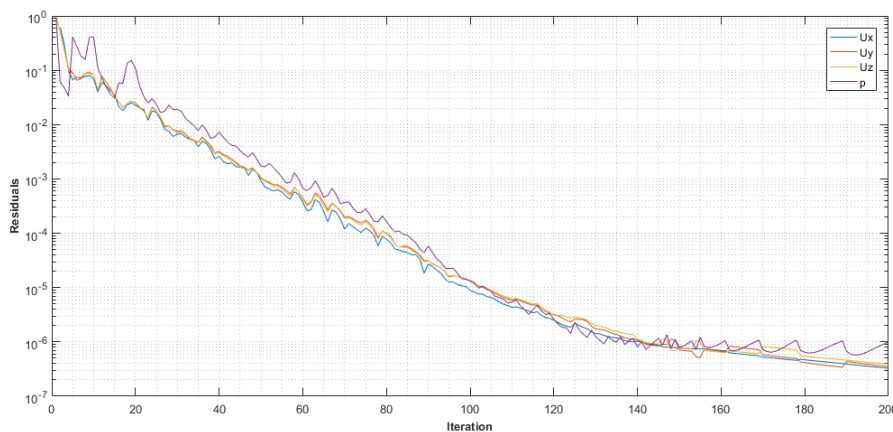


Figure 21: Base mesh residuals

The COV values, which is the most important value in this case converges for all the mesh tested just after approximately 50 iteration for the COV value at plane 0, depicted in Figure 22. This is the plane right before the mixture enters the mixing element. One interesting to notice is that the mesh marked as 100 % has huge oscillations compared to the other mesh, but converges the same iteration as the other mesh. All the planes calculating COV and its placement is discussed and visualized in chapter 5.2.3, but are not presented to conserve space. The volumetric flow rate also converges after few iteration and all the tested mesh converges to a constant value of 440 mL/ min, illustrated in Figure 23, which is the expected from its initial boundary at the inlet and the flow being incompressible. The fact that the simulations converges that fast is due to the flow being laminar and do not require as many iterations to converge compared to turbulent flow.

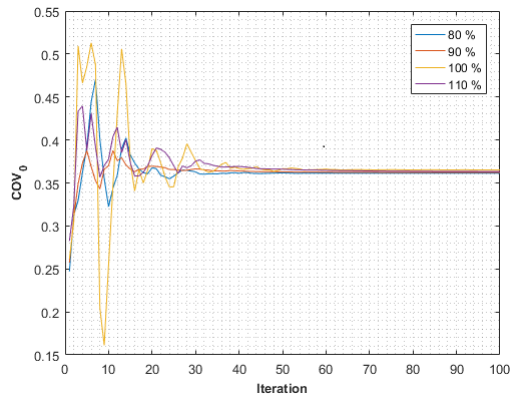


Figure 22: COV at Plane 0

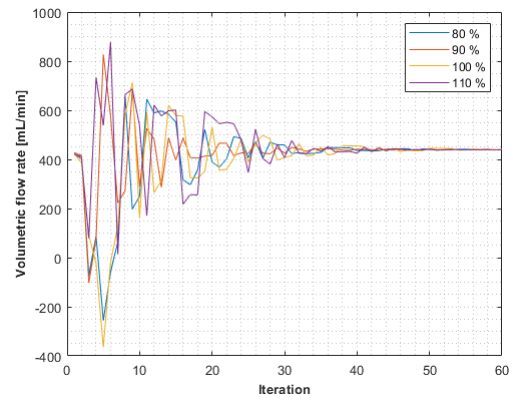


Figure 23: Volumetric flow rate

It can be seen from both Figure 22 and Figure 23 indicates that all the mesh tested in this convergence test are quite fine and the final result do not depend on the final mesh quality tested. The COV values converges almost at the same iteration with the same final value. This perhaps is becomes more visible when plotting the COV as a function of planes defined in chapter 5.2.3, see Figure 24. The lime green line represents the following COV function. Due to the viscosity ratio between the component assumed to be 1 the COV function can be fitted as a exponential function discussed in chapter 2.3.1.

$$\text{COV}(N) = 0.330 \exp\left(-0.137 \frac{NL_e}{D}\right)$$

It can be observed that the final results barely changes varying grid size, hence the the simulations is independent of the mesh tested. A fine mesh is necessary preserve the overall geometry and reduce the jagged surfaces created due the the snapping process in *snappyHexMesh* did not snap to the STL surface properly. For all the following cases the mesh marked as 100%, represented with the yellow line in Figure 24 is used. It is also possible to use the 80 % mesh since this test shows that the COV do not change.

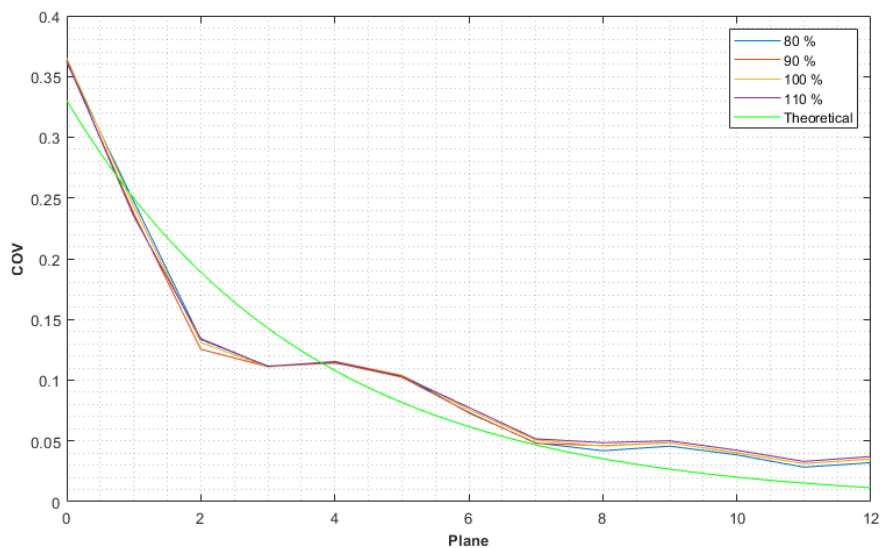


Figure 24: COV along the static mixing elements

## 6.3 *Mixing results*

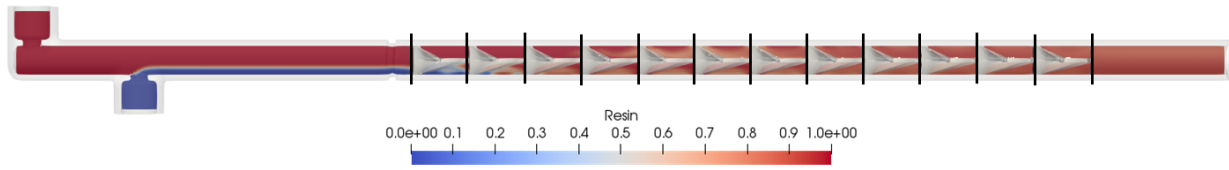


Figure 25: Top view of mixing result

Figure 25 illustrates the two mixing component and how their concentration changes as they are mixed along the pipe. The red color represents 100 % Resin entering in from the inlet at the rear. Blue represents 0 % Resin, hence 100 % Catalyst. The black vertical lines represents the plane placement for the COV calculation, Plane 0 in the far left to plane 12 in the fare right. One interesting fact from Figure 25. One interesting discovery from Figure 25 is that there seem to be very little molecular diffusion from where the catalyst enters to the first mixing elements. This can be due to the high viscosity (125 cP) of the fluid components that prevents the molecular diffusion. This means that the space between the Catalyst inlet to the first mixing element can be reduced without effecting the mixing efficiency of the Stadelmaier mixing elements. Doing this is going to reduce a significant amount of the internal cavity of the mixer, resulting in less paint wasted during the color changing process. Another important aspect is the Resin concentration to the far right of Figure 25 that is roughly 90 % with slighter less concentration on the top left. 90 % resin concentration is due to the fact that there is 10 times more resin than catalyst. A hard 90 % concentration at the end results in a COV of 0, indicating that the mixture is completely mixed together and can not be further mixed!

The mixing pattern for the Stadelmaier mixing elements is presented in Figure 26 below. The first plane (plane 0) visualizes the initial component distribution with the the catalyst in blue to the far left side of the cross section and resin occupying most of the space. In between small molecular diffusion can be observed by the white color in between the blue and red. The main mixing starts as the mixture starts moving trough the mixing elements. The catalyst starts to move clockwise as it progresses and starts blending in to the resin. One interesting thing to notice by this mixing pattern is that the mixing generally starts happening close to the the tube wall, most obvious in Figure 26c, leaving mostly pure resin in the middle. If the initial distribution of catalyst was different, occupying more space in the middle and less wide this may result in much better mixing in the middle at the same time as the outside reducing the number of elements needed to get satisfactory mixing results. A rough conclusion can be made by closely inspecting 26h, After passing 7 elements the catalyst and resin is mostly mixed and should give a low COV. This is closely investigated in the upcoming chapters for more accurate representation of the mixing.

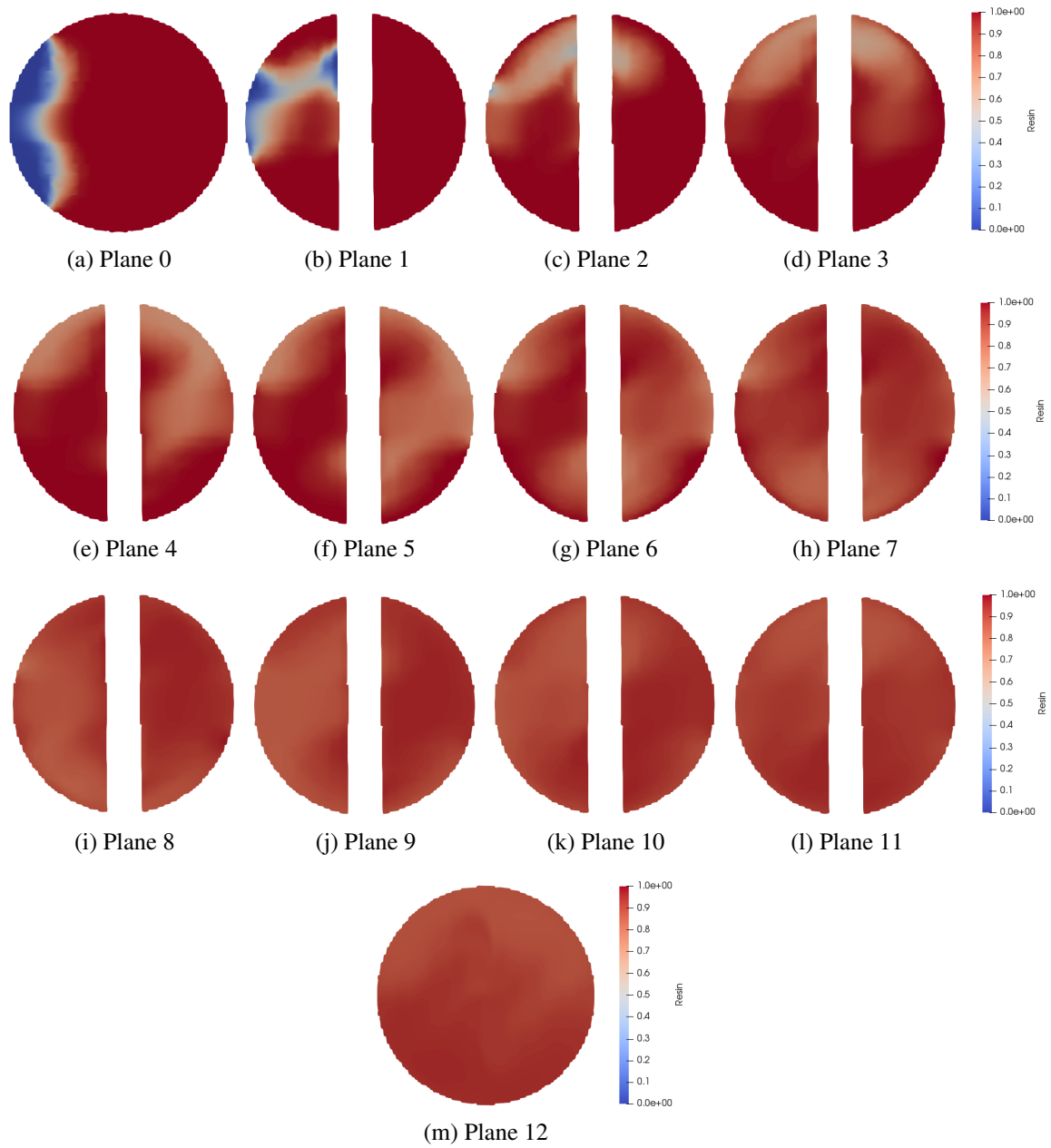


Figure 26: Mixing pattern



### 6.3.1 Varying viscosity

Visualized in Figure 27 is the effect of different viscosities has on the mixing performance. Due to the catalyst being a colorless liquid the COV threshold for satisfied mixing result is set to 5 %, indicated by the blue dotted line. Everything that has a low value than 5 % is accounted as a completely mixed paint that can be applied on the car body. The initial COV value barely changes for all 5 viscosities, strengthening the theory that the space from the catalyst inlet and the first mixing element can be eliminated to reduce the internal cavity without sacrificing the performance of the 2K-mixer. As the mixture progresses through the static mixers the paint with lower viscosity performs better and better, except for the higher velocity paint. The paint carrying a viscosity of 50 cP has a staggering COV value close to 0 %, and passes the COV threshold just after 6 elements. This is great for vendors applying paint to cars with low viscosities since they can reduce the number of elements and get an even smaller internal cavity reducing the paint wasted during color change. For paint with high viscosity, at least one more element is necessary to get a satisfied mixing.

One interesting fact discovered by this test is that even though the paint with 150 cP has a 20 % higher value than paint with 125 cP they both perform almost identical. Indicated by the green and purple line in Figure 27. This means that for a constant flow rate of 440 mL/min the number of mixing elements can be reduced to at least 7 and still get a satisfied paint mixture that can be applied on the car body, even with different viscosities. Also interesting is that the two first elements are those who carry most of the mixing load compared to the rest of the element, reducing the COV value to less than 15 %. The mixing velocity gradually slows down for increasing numbers of elements and even almost converges for the mixture with the lowest viscosity.

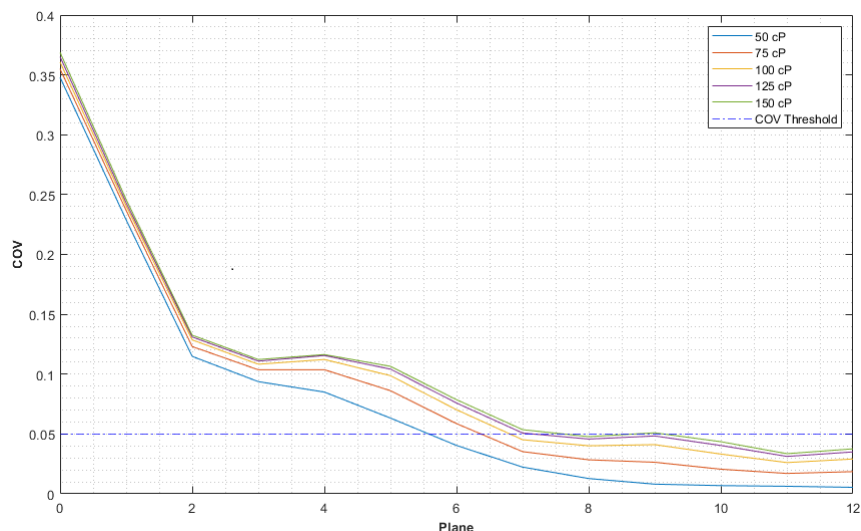


Figure 27: COV at varying viscosities

### 6.3.2 Varying velocity

Due to the fact that the volumetric flow rate changes depending on which part of the car body is painted, a test with varying velocity is conducted to find out if the flow rate has any impact on the mixing quality. The result is conducted with a viscosity of 125 cP. The final result is presented in Figure 28 below.

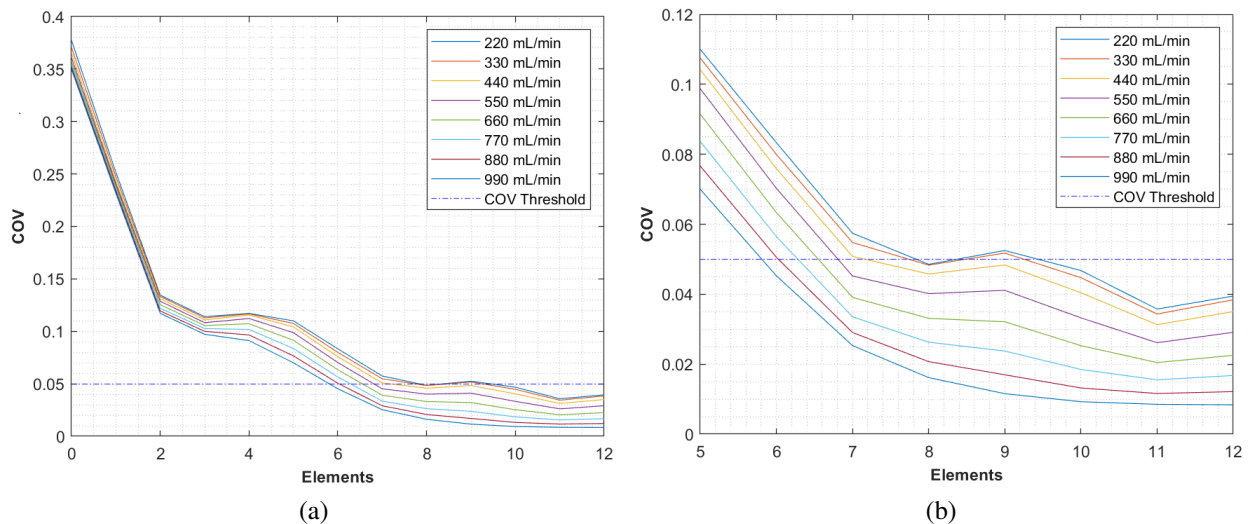


Figure 28: COV at varying volumetric flow rate

The mixer is tested for 8 different volumetric flow rate, covering a wide range of the mixers capacity. From Figure 28a it can be noticed that the initial COV value do barely change 2 % for the lowest to the highest flow rate. this small change however do hardly have any major impact on the final result. As the fluid progresses trough the mixing element this incremental change starts to increase. The reason for this is due to the fluid having a higher momentum when folded into each other, resulting more violent mixing increasing the mixing performance.

Due to the fact that the COV decreases rapidly in the beginning an we are only interested in how many mixing elements that is needed to have a satisfied mixture Figure 28b only contains the COV values for the last 7 mixing elements. Figure 28b illustrates more clearly the effect the velocity has on the mixing quality. A higher flow rate do not require as many mixing elements as for flow with a low flow rate. A satisfied mixing quality is reach with only 6 mixing elements for high flow rate compared to 8, depending on how you interpret the COV value increasing between the 8th and 9th element. This incremental increase may be due to some numerical error in the simulation, but this increase is minimal and just above the 5 % COV threshold for two of the lowest flow rates. Similar to the test done with different viscosities, most of the mixing is carried out by the first two mixing elements, but in this case higher flow causes the even more mixing and after 10 elements the COV is as low as 1 % for if the flow rate is at its highest value (990 mL/ min).

### 6.3.3 Overall effect of viscosity and velocity

As previously discussed in chapter 6.3.1 and 6.3.2, most of the mixing is carried out by the first two mixing elements resulting in a  $COV \leq 15\%$  and the mixing velocity gradually converging as the flow progresses through more mixing elements. This means that adding more mixing elements is not going to increase the mixing quality much, instead this just increases the internal volume in the mixer resulting in that more paint is wasted during the color changing process. This again adds to the overall cost of paint.

The main goal in this thesis is to optimize the 2K-mixer for paint to be applicable for a wide variety of paint with different viscosities. Because companies utilizing painting tools such as this typically desire to be efficient for various paint jobs. Due to this reason a total number of 40 simulations is run for various types of viscosities with various range of flow rate to easily conclude the maximum number of mixing element needed to get a good mixing result and reduce the extra elements that is not needed to minimize the paint wasted. The result is presented in Figure 29. In addition all the raw COV values for the different viscosity and flow velocity is presented in appendix chapter E.

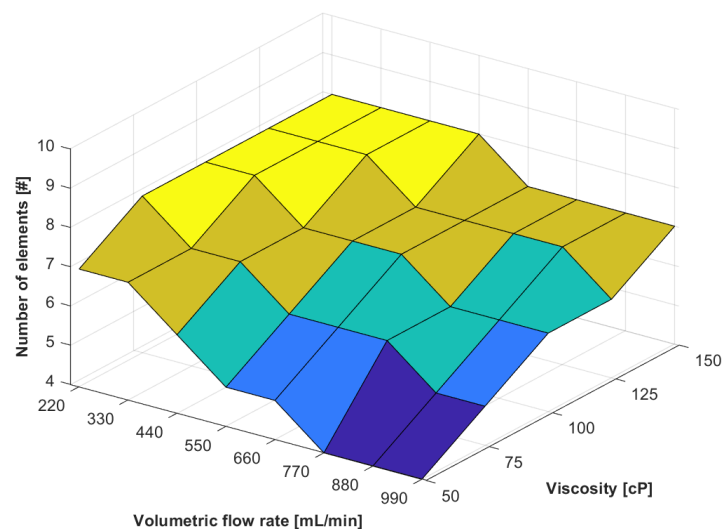


Figure 29: Number of elements necessary for sufficient mixing

As previously discussed, low viscosity and high volumetric flow rate results in better mixing quality. This can also be seen from Figure 29 as for the lowest viscosity and highest flow rate gives the best mixing result and passes the COV threshold just after 4 mixing elements. For higher viscosity paint the COV value increases and a total number of 6 elements are needed to reach the desired mixing at the same flow rate. What's really interesting in the number of elements needed for a high viscosity paint with low velocity, as this is the bottleneck for selecting the number of elements for ABB. According to the simulations conducted the maximum number of elements needed to ensure the satisfied blending result is 8, visualized with yellow surfaces in the surface plot.

One thing to have in mind is that the simulation is run by assuming that both resin and catalyst have the same viscosity, resulting in a single phase simulation. This means that the presented results only applicable for viscosity ratio equal to 1 between the two components, according to the theory presented in chapter 2.3.1. When the viscosity ratio of the components is not equal to one, the COV becomes a function of three dimensionless variables; the viscosity ratio between the mixed components, the mean concentration at the sampling area, and the ratio between the total mixing length and the tube diameter. This function can be any function and not necessarily an exponential function, which is the case for this thesis.

### 6.3.4 *Mixing result at varying tube diameter*

According to the obsolete method for evaluating mixing efficiency mentioned in chapter 2.1.1, reducing the conduit diameter results in smaller maximum striation thickness, which then results in better performance. This is mostly valid and the mixer with a conduit diameter of 4.5 mm performs better than the mixers with higher diameters. The result is presented in Figure 30, but the difference is marginal. This may be due to the diameters tested being really small. Notice that the initial COV is the same, and the difference can firstly be noticed after the mixture has passed 3 elements. Changing the diameter from its initial 6.4 mm to value of 4.5 mm is going to reduce the overall COV by 2 % and kept constant from element 4 and throughout element number 12.

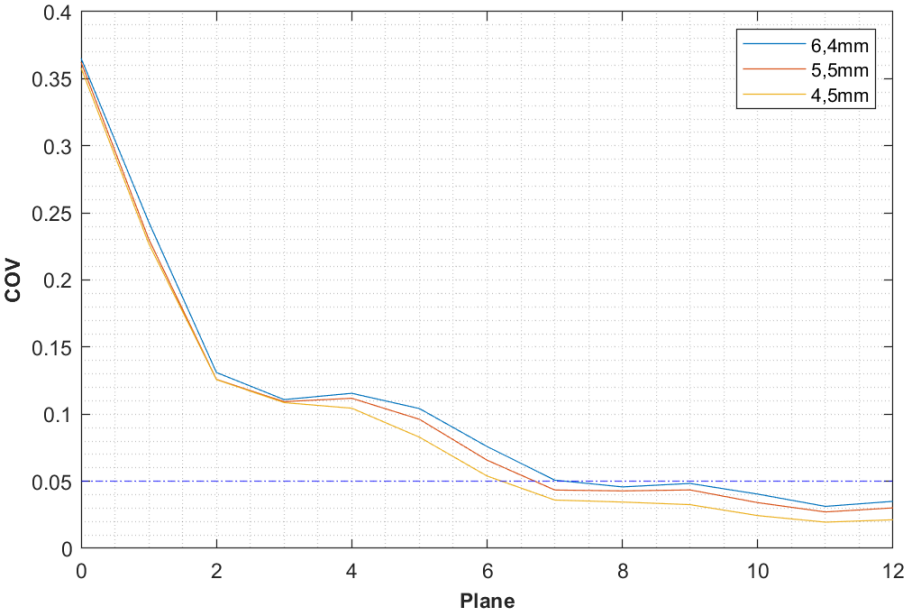


Figure 30: COV at varying diameter

## 6.4 Modified design

Slightly changed geometry to the original design is evaluated according to the finding in section 6.3. In this design a 45 ° angle is added to the catalyst line relative to the flowing fluid in the main pipe instead of having it perpendicular like in the existing model (see Figure 14). This design change is visualized in Figure 31a. Additionally, Figure 31b represents the same design with one minor difference. This design has an additional extruded wall that shields the catalyst inlet from the resin flow preventing the resin to penetrate into the catalyst line and cause subsequent mixing inside the catalyst line. Also by a decrease to the angle of the catalyst line relative to the flowing fluid in the main pipe, instead of having it perpendicular like in the existing model, also reduces the chance of getting resin in inside the catalyst line. Remember that the catalyst line does not contain an additional dump valve underneath, causality being the paint hardening inside the catalyst line and never get cleaned out properly. Consequently by adding a wall that shields the catalyst line, also prevents the cleaning quality on the wall side closed to the catalyst line. Remember that the cleaning agent line is placed at the back end just on the opposite side of the resin line that cleans the main pipe.

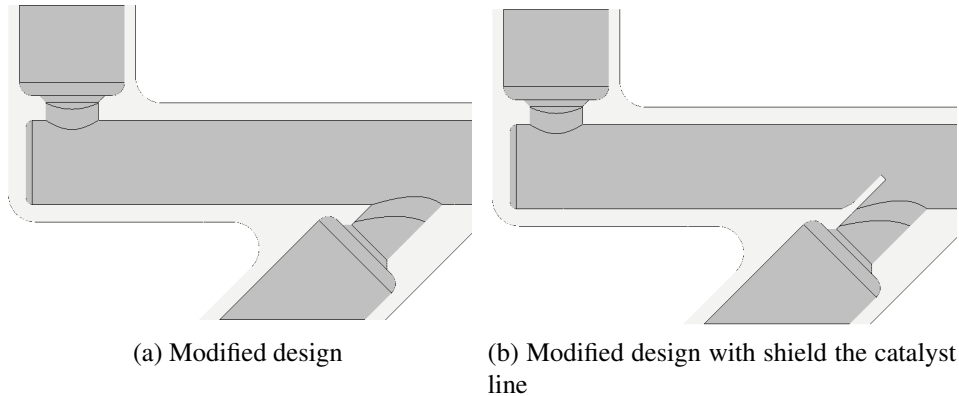


Figure 31: Modified design

Also the space between the location where the catalyst enters the main tube and the point where the actual mixing start is entirely excluded, as previous simulation concluded that this space is unnecessary. This means that the entire length of the tube gets reduced. The modified design is represented in Figure 32. Technical drawings for this model is also presented in appendix section B.2.

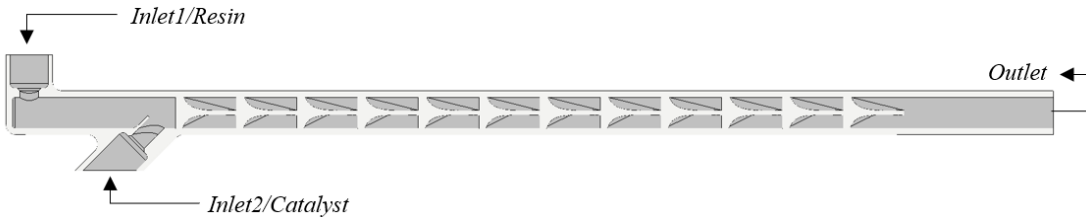


Figure 32: Top view of modified design

Unfortunately Due to problems with the generation of STL files using Autodesk Inventor, the modified geometry with the wall shielding the catalyst line were not properly meshed. The snapping process ended up excluding the additional wall geometry, resulting in almost identical meshes between the modified design and the modified design containing the wall. Hence only the modified design without the wall shield is compared to the original design.

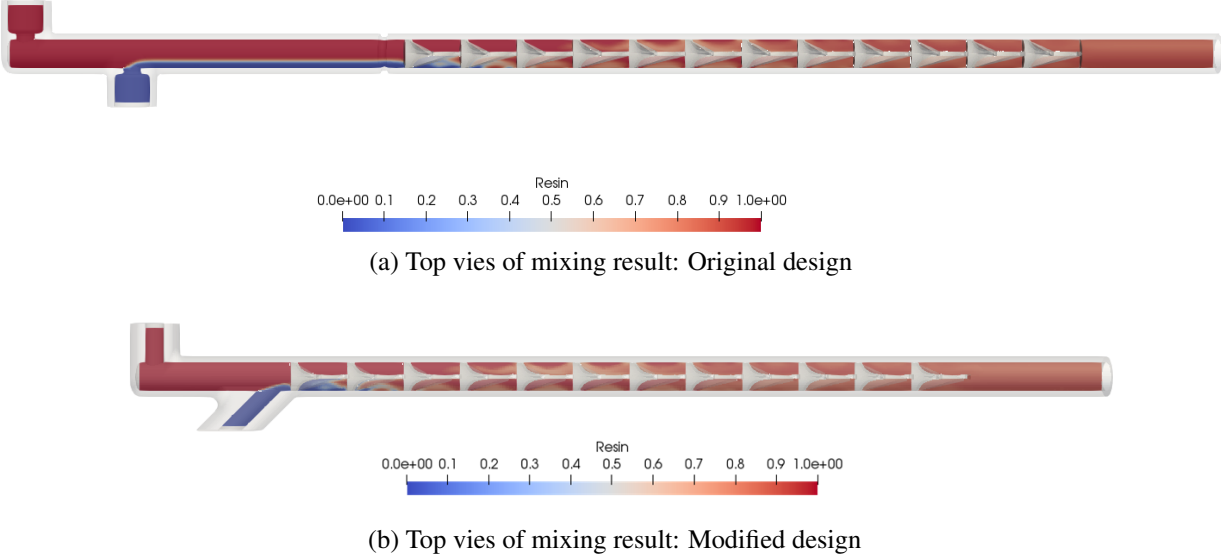


Figure 33: Original design vs. modified design

Figure 33 illustrates the component distribution for both models looking from above. One interesting discovery from this figure is that the component distribution looks relatively similar to each other. One of the reasons to angle the catalyst inlet was to intentionally align the catalyst flow more with the resin flow, to get a different initial distribution of the component and investigate how different distributions changes the final COV value. As depicted in Figure 34, this was not the case as both designs resulted in having almost identical component distribution, with the original design having slightly higher molecular diffusion due to resin and catalyst have had more time to interact with each other due to the space between the catalyst line and the first mixing elements on the original design. This further means the COV value is going to be almost identical. This is concluded by Figure 35 below. Although the design changes did not give the result initially predicted, this test further conclude that the space between the catalyst inlet and the first mixing element can be reduced without impacting the mixing quality of the paint. Also to implement these design changes the cleaning agent valve, which has to be moved to make space for the catalyst line that is placed at an angle.

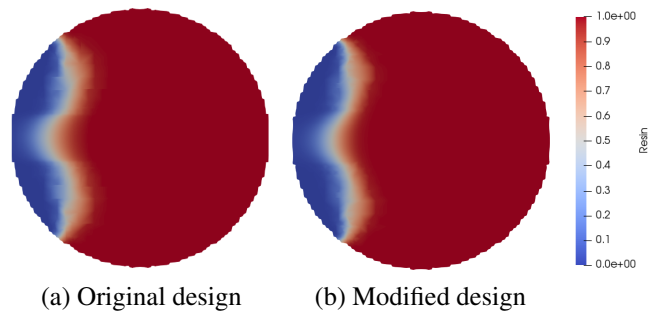


Figure 34: initial component distribution

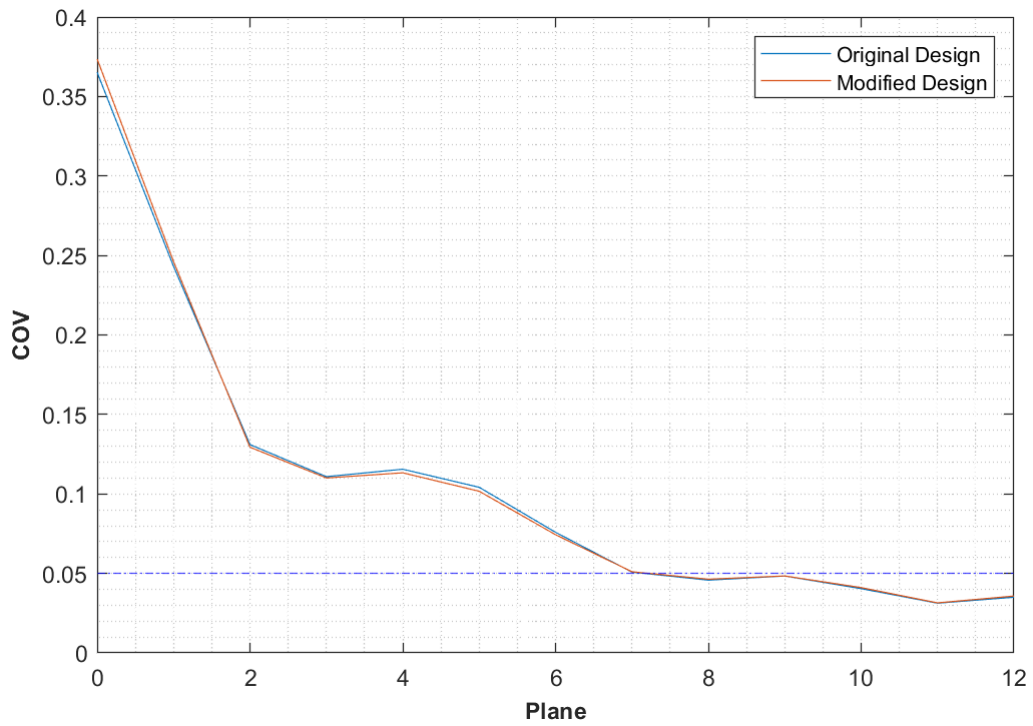


Figure 35: fig: COV for different designs

## 7 Conclusion

The primary objective of this thesis is to optimize the mixing process and reducing the internal volume of the 2K-mixer, with the aim of reducing paint waste during cleaning. Therefore it is essential that the 2K-mixer is optimized, giving a paint mixture that can be applied onto a car body, and have minimal paint waste during a color changing process. To accomplish this, static mixing elements that are installed in the 2K-mixer are imported into a CAD program where a simplified model is generated representing the section of mixing as accurately as possible.

The model is then exported into a STL file for CFD simulations for evaluating the mixing efficiency using OpenFOAM. First and foremost, the STL file has to be of a high quality in order to be able to generate a mesh, that represent the internal cavity of the mixing tube as accurately as possible. Having a corrupted STL file, which may be the case in this thesis, can cause numerous problems with the meshing process. This is solved by increasing the number of cells in the base mesh, such that the mesh represents the geometry as accurately as possible. To ensure that the simulation results are independent from the number of grid cells, several simulations used varying grid sizes. The grid convergence test is conducted by changing the number of cells in x,y and z direction in the background mesh.

The mixing performance is evaluated using the simpleFOAM solver in OpenFOAM in conjunction with the scalarTransport and surfaceFieldValue function objects to calculate the coefficient of variation (COV) after each elements and before the first element to represent how the COV value changes as the paint flow through the mixing elements. Simulations reveal that the the mixer utilized in the 2K-mixer block valve performs better when the flow rate is high. It is also revealed that the high viscosity paint requires more mixing elements compared to low viscosity paint. The diameter is also tested, and results showed that by keeping the element length constant and reducing the diameter increases the mixing quality, but effects is small due to the already diameter being small. Simulations show that the number of elements can be reduced to at least 8 elements from its initial 12 elements, and get a satisfactory mixing result for two component paint with a viscosity ratio close to 1. Mixing of paint with viscosity ratio not close to 1 is not investigated in this thesis, and is therefore something that can be researched further to conclude if more than 8 mixing elements is necessary.

Some design changes is suggested to further reduce the internal cavity of the mixing tube, such as reducing the space between the catalyst inlet and the first mixing element. This research shows that the distance has little to nothing to say for the final COV value. The mixing quality is also tested by placing the catalyst inlet at a lower angle relative to the main flow velocity, which did not do anything with the mixing performance of the static mixing elements.



# References

- [1] S. Hossain, M. Ansari and K.-Y. Kim, ‘Evaluation of the mixing performance of three passive micromixers’, *Chem. Eng. J.*, vol. 150, no. 2-3, pp. 492–501, Aug. 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1385894709001508> (visited on 24/01/2019).
- [2] O. Reynolds, ‘An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels’, *Philos. Trans. R. Soc.*, vol. 174, pp. 935–982, Mar. 1883. [Online]. Available: <http://www.jstor.org/stable/109431> (visited on 12/03/2019).
- [3] D. Jackson and B. Launder, ‘Osborne reynolds and the publication of his papers on turbulent flow’, *Annu. Rev. Fluid Mech.*, vol. 39, no. 1, pp. 19–35, Jan. 2007. [Online]. Available: <https://doi.org/10.1146/annurev.fluid.39.050905.110241> (visited on 12/03/2019).
- [4] E. Saadjan, A. Rodrigo and J. Mota, ‘On chaotic advection in a static mixer’, *Chem. Eng. J.*, vol. 187, pp. 289–298, Apr. 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1385894712001647> (visited on 29/01/2019).
- [5] RobotWorx, *ABB IRB 5500-22*, 2019. [Online]. Available: <https://www.robots.com/robots/abb-irb-5500-22> (visited on 15/02/2019).
- [6] K. Myers, A. Bakker and D. Ryan, ‘Avoid agitation by selecting static mixers’, *Chem. Eng. Prog.*, vol. 93, no. 6, pp. 28–38, Jun. 1997. [Online]. Available: [https://www.researchgate.net/publication/286362262\\_Avoid\\_agitation\\_by\\_selecting\\_static\\_mixers](https://www.researchgate.net/publication/286362262_Avoid_agitation_by_selecting_static_mixers) (visited on 01/04/2019).
- [7] R. Thakur, C. Vial, K. Nigam, E. Nauman and G. Djelveh, ‘Static mixers in the process industries—a review’, *Trans. IChemE*, vol. 81, no. 7, pp. 787–826, Aug. 2003. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0263876203723685> (visited on 13/03/2019).
- [8] Primix B.V., *Operating principle of static mixer*, 2018. [Online]. Available: <https://www.primix.com/products/static-mixers-custom-made/operational-principles.html> (visited on 02/04/2019).
- [9] Wikipedia contributors, *Plug flow*, 2019. [Online]. Available: [https://en.wikipedia.org/wiki/Plug\\_flow](https://en.wikipedia.org/wiki/Plug_flow) (visited on 09/04/2019).
- [10] A. Cybulski, *Static mixers*, 2007. [Online]. Available: [https://traxxys.com/wp-content/uploads/2017/05/1.1.4\\_Technology\\_Report\\_Static\\_Mixers\\_Cybulski-1.pdf](https://traxxys.com/wp-content/uploads/2017/05/1.1.4_Technology_Report_Static_Mixers_Cybulski-1.pdf) (visited on 09/04/2019).
- [11] C. van Pijpen, ‘Laminar flow in a primix static mixer. simulation and experiments’, Master’s Thesis, Department of Applied Physics, Delft University of Technology, Netherlands, Jan. 2002. [Online]. Available: <http://resolver.tudelft.nl/uuid:408bf886-ebed-4b4a-9d08-f0a957dc9408> (visited on 19/04/2019).
- [12] Y. A. Çengel and J. M. Cimbala, *Fluid mechanics : fundamentals and applications*, 3rd ed. in SI units. Boston: McGraw-Hill, 2014.

- [13] F. Grosz-Röll, ‘Assessing homogeneity in motionless mixers’, *Int. Chem. Eng.*, vol. 20, pp. 542–549, 1980.
- [14] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics : the finite volume method*, 2nd ed. Harlow: Pearson/Prentice Hall, 2007.
- [15] O. Sirevaag, ‘Cfd simulation of an offshore air intake and exhaust system’, Master’s Thesis, Department of Mechanical and Structural Engineering and Materials Science, University of Stavanger, Norway, Jun. 2015. [Online]. Available: <https://uis.brage.unit.no/uis-xmlui/handle/11250/2411432> (visited on 14/05/2019).
- [16] CFD Online, *Generic scalar transport equation*, 2006. [Online]. Available: [https://www.cfd-online.com/Wiki/Generic\\_scalar\\_transport\\_equation](https://www.cfd-online.com/Wiki/Generic_scalar_transport_equation) (visited on 11/06/2019).
- [17] H. G. Lemu, ‘Additive manufacturing (rapid prototyping & manufacturing)’, University lecture presentation in MSK550 at the University of Stavanger, Oct. 2018.
- [18] Wikipedia contributors, *Flow cups*, 2016. [Online]. Available: [https://en.wikipedia.org/wiki/Flow\\_cups](https://en.wikipedia.org/wiki/Flow_cups) (visited on 29/05/2019).
- [19] C. Greenshields, *Viscosity conversion table*, 2019. [Online]. Available: [http://cdn2.hubspot.net/hub/219243/file-2484811014-pdf/Downloads/Conversion\\_Table\\_eGuide.pdf](http://cdn2.hubspot.net/hub/219243/file-2484811014-pdf/Downloads/Conversion_Table_eGuide.pdf) (visited on 20/03/2019).
- [20] OpenFOAM, *About openfoam*, 2019. [Online]. Available: <https://www.openfoam.com/> (visited on 15/05/2019).
- [21] C. Greenshields, *File structure of openfoam cases*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-case-file-str%20ucture/#x16-1220004.1> (visited on 15/05/2019).
- [22] B. Fabritius and G. Tabor, ‘Improving the quality of finite volume meshes through genetic optimisation’, *Engineering with Computers*, vol. 32, no. 3, pp. 425–440, Jul. 2016. [Online]. Available: <https://link.springer.com/article/10.1007/s00366-015-0423-0> (visited on 25/05/2019).
- [23] C. Greenshields, *Mesh generation with the blockmesh utility*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-blockmesh/> (visited on 26/05/2019).
- [24] K. E. Giljarhus, ‘OpenFOAM tutorials’, University tutorials in MSK600 at the University of Stavanger, Jan. 2019.
- [25] C. Greenshields, *Mesh generation with the snappyhexmesh utility*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-snappyhexmesh/> (visited on 26/05/2019).
- [26] C. Greenshields, *Standard solvers*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-standard-solvers/> (visited on 30/05/2019).
- [27] OpenFOAM, *simpleFoam*, 2019. [Online]. Available: <https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-incompressible-simpleFoam.html> (visited on 30/05/2019).
- [28] U. C. Liestyarini, ‘Cfd analysis of internal pipe flows’, Master’s Thesis, Department of Mechanical and Structural Engineering and Materials Science, University of Stavanger,

- Norway, Jun. 2016. [Online]. Available: <https://uis.brage.unit.no/uis-xmlui/handle/11250/2411432> (visited on 15/05/2019).
- [29] OpenFOAM, *Standard boundary conditions*, 2019. [Online]. Available: <https://www.openfoam.com/documentation/user-guide/standard-boundaryconditions.php> (visited on 30/05/2019).
- [30] C. Greenshields, *Time and data input/output control*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-controldict/> (visited on 01/06/2019).
- [31] C. Greenshields, *Post-processing command line interface*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-post-processing-cli/> (visited on 30/05/2019).
- [32] OpenFOAM, *Residuals*, 2019. [Online]. Available: <https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-utilities-residuals.html> (visited on 02/03/2019).
- [33] OpenFOAM, *Scalar transport*, 2019. [Online]. Available: <https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-solvers-scalar-transport.html> (visited on 28/03/2019).
- [34] OpenFOAM, *Surface field value*, 2019. [Online]. Available: <https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-field-surface-field-value.html> (visited on 18/04/2019).
- [35] C. Greenshields, *Numerical schemes*, 2018. [Online]. Available: <https://cfd.direct/openfoam/user-guide/v6-fvschemes/> (visited on 03/06/2019).

---

# APPENDIX

---

# A Technical specifications of the 2K-mixer

## 7 Fluid Components

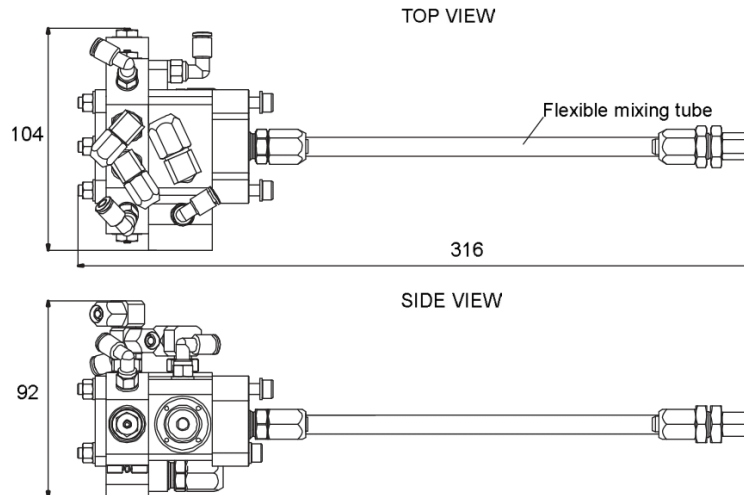
### 7.5 2K-Mixer with membrane valve, M-PAC

#### 7.5.5 Technical specifications

Dimensions

2K-mixer dimensions are shown in the illustration below.

Figure 79 2K-mixer with membrane valve, dimensions



General

2K-mixer type:	Static mixer
MTTR (Mean Time To Repair), 2K-mixer:	10 min. (Time to exchange mixing elements)
MTTR (Mean Time To Repair), 2/2 Fluid valve:	< 2 min. (Exchange complete valve)
MTTR (Mean Time To Repair), 2/2 Fluid membrane valve:	< 2 min. (Exchange complete valve)

Material

2K-mixer:	Acid proof stainless steel
Mixing elements:	Polypropylene - PP
Mixing tube:	Fluor polymer - FEP

Fluid Flow

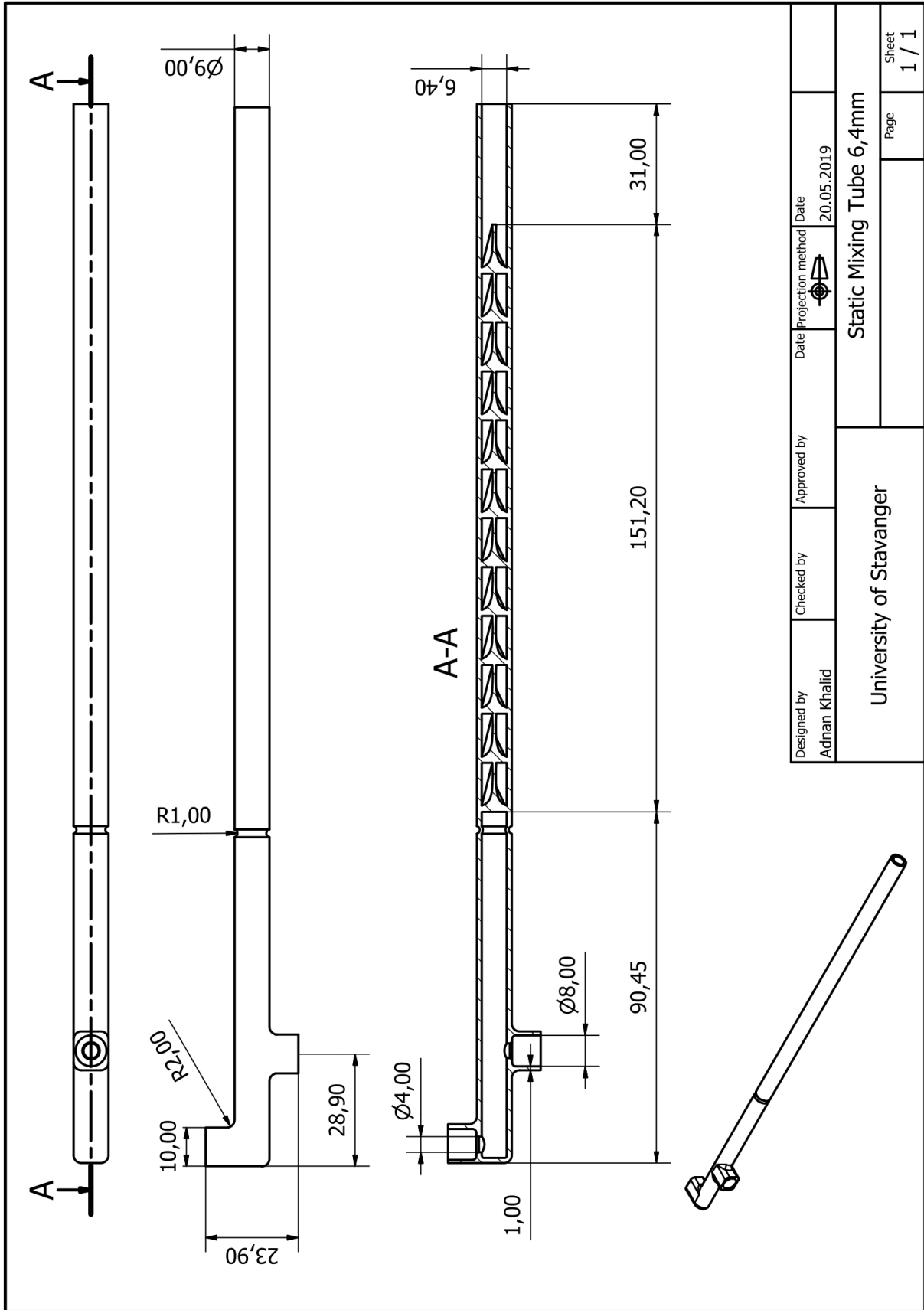
Maximum flow:	*1
Minimum flow:	100ml/min.*1/*2
Volumetric accuracy:	±2%, accuracy depends on fluid regulation
Max. permitted operating fluid pressure:	20bar
Recommended operating fluid pressure:	2 - 6bar
Max. permitted fluid temperature:	60 °C
Recommended pilot air pressure range:	
- M-PAC valve:	8 - 10bar
- Membrane valve:	4 - 7bar

- \*1 Fluid flow depends on viscosity, system pressure, needed accuracy on coated object and wear. Actual flow should be verified with current fluid.
- \*2 A special mixing tube for flow rates between 25-100 ml/min. is available. Please contact your local ABB organization.

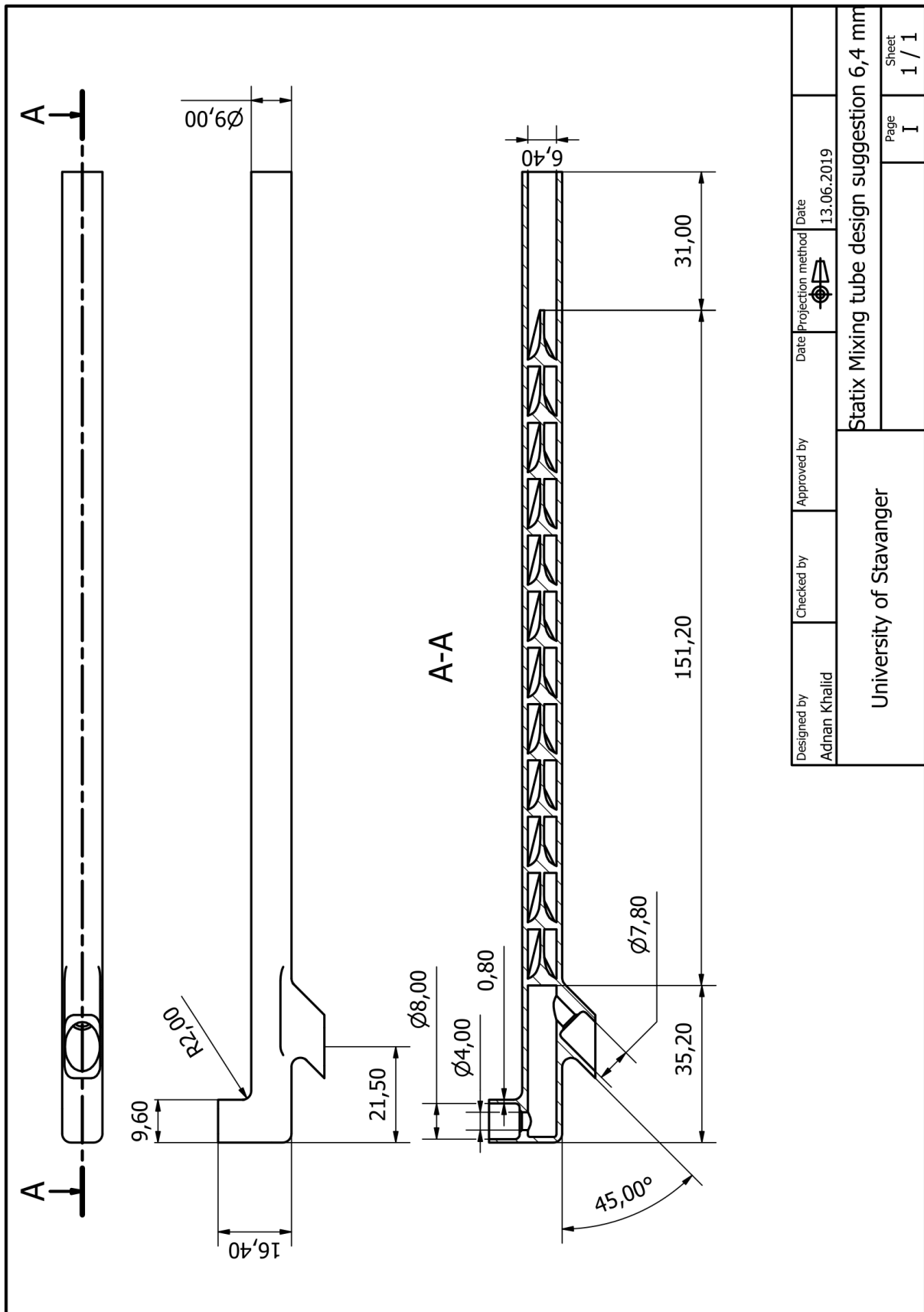
For further technical information, please refer to the parts lists in the technical documentation.

# B CAD drawings

## B.1 Static mixing tube 6.4 mm



## B.2 Static mixing tube 6.4 mm modified design







# D OpenFOAM base case

## D.1 0 folder

### Catalyst

```
/*-----*- C++ -*-----*/
=====
// / Field | OpenFOAM: The Open Source CFD Toolbox
// / Operation | Website: https://openfoam.org
// / And | Version: dev
// / Manipulation |
\*------*/
FoamFile
{
    version 2.0;
    format volScalarField;
    class Catalyst;
    object
}
// * * * * *
sInlet 1;
dimensions [0 0 0 0 0]; // s can represent any scalar
// with any dimensional units
internalField uniform 0;
boundaryField
{
    inlet1
    {
        type fixedValue;
        value uniform 0.0;
    }
    inlet2
    {
        type fixedValue;
        value uniform 1.0;
    }
    outlet
    {
        type inletOutlet;
        inletValue uniform 0;
        value uniform 0;
    }
    pipe_patch24311
    {
        type zeroGradient;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// * * * * *

```

### P

```
/*-----*- C++ -*-----*/
=====
// / Field | OpenFOAM: The Open Source CFD Toolbox
// / Operation | Website: https://openfoam.org
// / And | Version: dev
// / Manipulation |
\*------*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object
}
// * * * * *
dimensions [0 2 -2 0 0 0];
internalField uniform 0;
boundaryField
{
    inlet1
    {
        type zeroGradient;
    }
    inlet2
    {
        type zeroGradient;
    }
    outlet
    {
        type fixedValue;
        value uniform 0;
    }
    pipe_patch24311
    {
        type zeroGradient;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// * * * * *

```

# Resin

```
/*-----*- C++ -*-----*/
//
// Field
// Operation
// And
// Manipulation
//
FoamFile
{
    version      2.0;
    format       ascii;
    class        volScalarField;
    object       Resin;
}
// * * * * *
sInlet      1;
dimensions  [0 0 0 0 0]; // s can represent any scalar
// with any dimensional units
internalField uniform 0;
boundaryField
{
    inlet1
    {
        type       fixedValue;
        value       uniform 1.0;
    }
    inlet2
    {
        type       fixedValue;
        value       uniform 0.0;
    }
    outlet
    {
        type       inletOutlet;
        inletValue uniform 0;
        value       uniform 0;
    }
    pipe_patch24311
    {
        type       zeroGradient;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// *****
```

# U

```
/*-----*- C++ -*-----*/
//
// Field
// Operation
// And
// Manipulation
//
FoamFile
{
    version      2.0;
    format       ascii;
    class        volVectorField;
    object       U;
}
// * * * * *
Uinlet      (10 0 0);
dimensions  [0 1 -1 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
    inlet1
    {
        type       fixedValue;
        value       uniform (0 0 0.1326);
    }
    inlet2
    {
        type       fixedValue;
        value       uniform (0 0 -0.01326);
    }
    outlet
    {
        type       zeroGradient;
    }
    pipe_patch24311
    {
        type       noSlip;
    }
    #includeEtc "caseDicts/setConstraintTypes"
}
// *****
```

## D.2 Constant folder

```

transportProperties
/*-----*-- C++ -*-----*/
=====
\\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox
\\ / O p e r a t i o n | Website: https://openfoam.org
\\ / A n d | Version: dev
\\ / M a n i p u l a t i o n |
\\*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object        transportProperties;
}
// * * * * *
transportModel  Newtonian;
nu              [0 2 -1 0 0 0] 9.615e-05;
// *****

```

```

turbulenceProperties
/*-----*-- C++ -*-----*/
=====
\\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox
\\ / O p e r a t i o n | Website: https://openfoam.org
\\ / A n d | Version: dev
\\ / M a n i p u l a t i o n |
\\*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object        RASProperties;
}
// * * * * *
simulationType laminar;

RAS
{
    RASModel      kEpsilon;
    turbulence    on;
    printCoeffs   on;
}
// *****

```

## D.3 System folder

```

blockMeshDict
/*----- C++ -----*/
=====
// / F i e l d | OpenFOAM: The Open Source CFD Toolbox
// / O p e r a t i o n | Website: https://openfoam.org
// / A n d | Version: dev
// / M a n i p u l a t i o n |
/*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object blockMeshDict;
}
// *****
backgroundMesh
{
    xMin -0.05575;
    xMax 0.2159;
    yMin -0.0045;
    yMax 0.0045;
    zMin -0.01095;
    zMax 0.01095;
    xCells 520;
    yCells 30;
    zCells 60;
}
}
convertToMeters 1;
vertices
(
    ($:backgroundMesh.xMin $:backgroundMesh.yMin $:backgroundMesh.zMin)
    ($:backgroundMesh.xMax $:backgroundMesh.yMin $:backgroundMesh.zMin)
    ($:backgroundMesh.xMax $:backgroundMesh.yMax $:backgroundMesh.zMin)
    ($:backgroundMesh.xMin $:backgroundMesh.yMax $:backgroundMesh.zMin)
    ($:backgroundMesh.xMin $:backgroundMesh.yMin $:backgroundMesh.zMax)
    ($:backgroundMesh.xMax $:backgroundMesh.yMin $:backgroundMesh.zMax)
    ($:backgroundMesh.xMin $:backgroundMesh.yMax $:backgroundMesh.zMax)
    ($:backgroundMesh.xMax $:backgroundMesh.yMax $:backgroundMesh.zMax)
);
blocks
(
    hex (0 1 2 3 4 5 6 7)
    (
        $:backgroundMesh.xCells
        $:backgroundMesh.yCells
        $:backgroundMesh.zCells
    )
    simpleGrading (1 1 1)
);
edges
(
    boundary
    (
        inlet
    )

```

```

{
    type patch;
    faces
    (
        (0 1 2 3)
    );
    inlet2
    {
        type patch;
        faces
        (
            (4 5 6 7)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
};
mergePatchPairs
(
);
// *****
ControlDict
/*----- C++ -----*/
=====
// / F i e l d | OpenFOAM: The Open Source CFD Toolbox
// / O p e r a t i o n | Website: https://openfoam.org
// / A n d | Version: dev
// / M a n i p u l a t i o n |
/*-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object controlDict;
}
// *****
application simpleFoam;
startFrom latestTime;
startTime 0;
stopAt endTime;
endTime 500;
deltaT 1;
writeControl adjustableRunTime;
writeInterval 20;
purgeWrite 6;
writeFormat ascii;

```

```

writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable true;
adjustTimeStep yes;
//maxCo 5;
//maxDeltaT 1e-03;
// *****
functions
{
//RESIDUALS
residuals
{
type residuals;
libs ("libUtilityFunctionObjects.so");
fields
{
}
}
// *****
//Scalar Transport
Resin
{
type scalarTransport;
resetOnStartup false;
functionObjectLibs ("libsolverFunctionObjects.so" );
schemesField Resin;
// Diffusion coefficient
D 1.0E-9;
// Name of Field
field Resin;
fvOptions
{
}
}
Catalyst
{
type scalarTransport;
resetOnStartup false;
functionObjectLibs ("libsolverFunctionObjects.so" );
schemesField Catalyst;
// Diffusion coefficient
D 1.0E-9;
// Name of Field
field Catalyst;
fvOptions
{
}
}
// *****
// Coefficient of Variation
CoV_0
{
type surfaceFieldValue;
libs ("libfieldFunctionObjects.so");
fields
{
}
//writeControl writeTime;
log yes;
writeFields no;
regionType sampledSurface;
name plane0;
sampledSurfaceDict
{
name plane0;
type plane;
planeType pointAndNormal;
pointAndNormalDict
{
point (0.0346 0 0);
normal (1 0 0);
}
}
operation CoV;
}
CoV_1
{
type surfaceFieldValue;
libs ("libfieldFunctionObjects.so");
fields
{
}
//writeControl writeTime;
log yes;
writeFields no;
regionType sampledSurface;
name plane1;
sampledSurfaceDict
{
name plane1;
type plane;
planeType pointAndNormal;
pointAndNormalDict
{
point (0.0472 0 0);
normal (1 0 0);
}
}
operation CoV;
}
CoV_2
{
type surfaceFieldValue;
libs ("libfieldFunctionObjects.so");
fields
{
}
//writeControl writeTime;
log yes;
writeFields no;
regionType sampledSurface;
name plane2;
sampledSurfaceDict
{
name plane2;
type plane;
planeType pointAndNormal;
pointAndNormalDict

```

```

    {
        point (0.0598 0 0);
        normal (1 0 0);
    }
    operation
    {
        CoV;
    }
}
CoV_3
{
    type
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane3;
    sampledSurfaceDict
    {
        name
        plane3;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.0724 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_4
{
    type
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane4;
    sampledSurfaceDict
    {
        name
        plane4;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.0850 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_5
{
    type
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane5;
    sampledSurfaceDict
    {
        name
        plane5;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.0976 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_6
{
    type
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane6;
    sampledSurfaceDict
    {
        name
        plane6;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.1102 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_7
{
    type
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane7;
    sampledSurfaceDict
    {
        name
        plane7;
        type
        plane;
        planeType
        pointAndNormal;
    }
}

```

```

    pointAndNormalDict
    {
        point (0.1228 0 0);
        normal (1 0 0);
    }
    operation
    {
        CoV;
    }
}
CoV_8
{
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane8;
    sampledSurfaceDict
    {
        name
        plane8;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.1354 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_9
{
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane9;
    sampledSurfaceDict
    {
        name
        plane9;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.1480 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_10
{
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane10;
    sampledSurfaceDict
    {
        name
        plane10;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.1606 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_11
{
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane11;
    sampledSurfaceDict
    {
        name
        plane11;
        type
        plane;
        planeType
        pointAndNormal;
        pointAndNormalDict
        {
            point (0.1732 0 0);
            normal (1 0 0);
        }
    }
    operation
    {
        CoV;
    }
}
CoV_12
{
    surfaceFieldValue;
    libs
    ("libfieldFunctionObjects.so");
    fields
    (Resin Catalyst);
    //writeControl
    writeTime;
    log
    yes;
    writeFields
    no;
    regionType
    sampledSurface;
    name
    plane12;
    sampledSurfaceDict
    {
        name
        plane12;
        type
        plane;
    }
}

```



# fvSchemes

```

planeType      pointAndNormal;
pointAndNormalDict
{
    point (0.1860 0 0);
    normal (1 0 0);
}
operation      Cov;
}
// *****
// Flow_Rate
flowRate
{
    name        outlet;
    #includeEtc "caseDicts/postProcessing/flowRate/flowRatePatch.cfg"
}
}
// *****

=====
// Field | OpenFOAM: The Open Source CFD Toolbox
// Operation | Website: https://openfoam.org
// And | Version: dev
// Manipulation |
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       fvSchemes;
}
// *****
dtSchemes
{
    default      steadyState;
}
gradSchemes
{
    default      Gauss linear;
    cellLimited  Gauss linear 1;
    $limited;
    grad(k)      $limited;
    grad(epsilon) $limited;
}
divSchemes
{
    default      none;
    div(phi,U)   bounded Gauss linearUpwind limited;
    turbulence   bounded Gauss limitedLinear 1;
    div(phi,k)   $turbulence;
    div(phi,epsilon) $turbulence;
    div(phi,Resin) bounded Gauss linearUpwind limited;
    div(phi,Catalyst) bounded Gauss linearUpwind limited;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}
laplacianSchemes
{
    default      Gauss linear corrected;
}
interpolationSchemes
{
    default      linear;
}
snGradSchemes
{
    default      corrected;
}
wallDist
{
    method meshWave;
}
// *****

```

## fvSolution

```
/*-----*- C++ -*-----*\n\n=====\n\\ / F ield | OpenFOAM: The Open Source CFD Toolbox\n\\ / O peration | Website: https://openfoam.org\n\\ / A nd | Version: dev\n\\ / M anipulation |\n\nFoamFile\n{\n  version 2.0;\n  format ascii;\n  class dictionary;\n  object fvSolution;\n}\n\n// *****\nsolvers\n{\n  P\n  {\n    solver GAMG;\n    smoother GaussSeidel;\n    tolerance 1e-6;\n    relTol 0.1;\n  }\n\n  Phi\n  {\n    $P;\n  }\n\n  "(U|k|omega|epsilon)"\n  {\n    solver smoothSolver;\n    smoother symGaussSeidel;\n    tolerance 1e-6;\n    relTol 0.1;\n  }\n\n  "(Resin|Catalyst)"\n  {\n    solver smoothSolver;\n    smoother symGaussSeidel;\n    tolerance 1e-6;\n    relTol 0.1;\n  }\n\n  SIMPLE\n  {\n    //residualControl\n    //{\n    //  P 1e-4;\n    //  U 1e-4;\n    //  "(k|omega|epsilon)" 1e-4;\n    //}\n    nNonOrthogonalCorrectors 0;\n    pRefCell 0;\n    pRefValue 0;\n  }\n}
```

```
potentialFlow\n{\n  nNonOrthogonalCorrectors 10;\n}\n\nrelaxationFactors\n{\n  fields\n  {\n    P 0.3;\n  }\n  equations\n  {\n    U 0.7;\n    "(k|omega|epsilon)." 0.7;\n  }\n}\n\n// *****\n
```

## meshQualityDict

```
/*-----*- C++ -*-----*\n\n=====\n\\ / F ield | OpenFOAM: The Open Source CFD Toolbox\n\\ / O peration | Website: https://openfoam.org\n\\ / A nd | Version: dev\n\\ / M anipulation |\n\nFoamFile\n{\n  version 2.0;\n  format ascii;\n  class dictionary;\n  object meshQualityDict;\n}\n\n// *****\n#includeEtc "caseDicts/mesh/generation/meshQualityDict.cfg"\n// - minFaceWeight (0 -> 0.5)\n// minFaceWeight 0.02;\nmaxNonOrtho 45;\nmaxInternalSkewness 2;\n// *****\n
```

# snappyHexMeshDict

```
/*-----*- C++ -*/
=====
\\ // Field | OpenFOAM: The Open Source CFD Toolbox
\\ // Operation | Website: https://openfoam.org
\\ // A nd | Version: dev
\\ // M anipulation |
\\-----*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    object snappyHexMeshDict;
}
// * * * * *
#includeEtc "caseDicts/mesh/generation/snappyHexMeshDict.cfg"
castellatedMesh on;
snap on;
addLayers off;
geometry
{
    mix.stl
    {
        type triSurfaceMesh;
        name pipe;
    }
};
castellatedMeshControls
{
    refinementSurfaces
    {
        pipe
        {
            level (2 2);
            patchInfo{type wall;}
        }
    }
    refinementRegions
    {
    }
    locationInMesh (-0.002 0.001 0.0005);
}
snapControls
{
    explicitFeatureSnap false;
    implicitFeatureSnap true;
}
addLayersControls
{
    layers
    {
        "pipe"
        {
            nSurfaceLay 4;
        }
    }
}
```



Plane	COV 220 mL/min	COV 330 mL/min	COV 440 mL/min	COV 550 mL/min	COV 660 mL/min	COV 770 mL/min	COV 880 mL/min	COV 990 mL/min
0	3,8002E-01	3,7397E-01	3,6878E-01	3,6440E-01	3,6076E-01	3,5766E-01	3,5503E-01	3,5284E-01
1	2,5298E-01	2,4950E-01	2,4592E-01	2,4259E-01	2,4061E-01	2,3884E-01	2,3690E-01	2,3488E-01
2	1,3479E-01	1,3404E-01	1,3256E-01	1,3068E-01	1,2865E-01	1,2591E-01	1,2308E-01	1,2065E-01
3	1,1428E-01	1,1352E-01	1,1221E-01	1,1046E-01	1,0827E-01	1,0599E-01	1,0363E-01	1,0121E-01
4	1,1721E-01	1,1693E-01	1,1636E-01	1,1515E-01	1,1226E-01	1,0812E-01	1,0370E-01	9,9094E-02
5	1,1071E-01	1,0888E-01	1,0650E-01	1,0349E-01	9,8747E-02	9,2742E-02	8,6201E-02	8,0049E-02
6	8,4225E-02	8,1768E-02	7,8672E-02	7,5095E-02	7,0292E-02	6,4542E-02	5,8692E-02	5,3412E-02
7	5,8010E-02	5,6315E-02	5,3618E-02	5,0003E-02	4,5226E-02	4,0094E-02	3,5284E-02	3,1161E-02
8	4,8315E-02	4,8649E-02	4,7756E-02	4,5063E-02	4,0176E-02	3,4296E-02	2,8445E-02	2,3334E-02
9	5,2327E-02	5,2377E-02	5,0976E-02	4,7422E-02	4,1124E-02	3,3605E-02	2,6395E-02	2,0153E-02
10	4,7086E-02	4,5975E-02	4,3561E-02	3,9403E-02	3,3253E-02	2,6556E-02	2,0594E-02	1,5656E-02
11	3,5989E-02	3,5208E-02	3,3546E-02	3,0559E-02	2,6144E-02	2,1373E-02	1,7061E-02	1,3450E-02
12	3,9453E-02	3,9177E-02	3,7588E-02	3,4188E-02	2,9117E-02	2,3592E-02	1,8562E-02	1,4328E-02

### Diameter test with 125 cP and 440m/min

Plane	COV_6,4mm	COV_5,5mm	COV_4,5mm
0	3,6519E-01	3,6218E-01	3,5802E-01
1	2,4314E-01	2,3037E-01	2,2705E-01
2	1,3104E-01	1,2590E-01	1,2571E-01
3	1,1087E-01	1,0932E-01	1,0861E-01
4	1,1556E-01	1,1186E-01	1,0445E-01
5	1,0419E-01	9,6183E-02	8,2854E-02
6	7,5871E-02	6,5705E-02	5,3892E-02
7	5,0812E-02	4,3473E-02	3,6008E-02
8	4,5781E-02	4,2809E-02	3,4497E-02
9	4,8359E-02	4,3549E-02	3,2559E-02
10	4,0414E-02	3,4062E-02	2,4411E-02
11	3,1280E-02	2,7118E-02	1,9500E-02
12	3,5025E-02	3,0164E-02	2,1302E-02

### For 125 cP and 440m/min

Surface	2K-mixer design	Modified Design
0	3,6519E-01	3,7358E-01
1	2,4314E-01	2,4614E-01
2	1,3104E-01	1,2939E-01
3	1,1087E-01	1,1006E-01
4	1,1556E-01	1,1329E-01
5	1,0419E-01	1,0170E-01
6	7,5871E-02	7,4350E-02
7	5,0812E-02	5,1060E-02
8	4,5781E-02	4,6370E-02
9	4,8359E-02	4,8419E-02
10	4,0414E-02	4,1108E-02
11	3,1280E-02	3,1508E-02
12	3,5025E-02	3,5771E-02