University of Applied Sciences and Arts Northwestern Switzerland
School of Life Sciences

Scale-Up of the agitator speed based on computational fluid dynamics

Bachelor thesis

Autor:	*Christian David Maier*
Supervisor:	*Prof. Dr. Andreas Zogg*
Expert:	*Lukas Moser*
Duration:	*8.3.21 -6.8.21*

Declaration of originality

I hereby affirm that I have independently written the work presented by me, that I have given full details of the sources, internet sources and aids used, and that I have indicated the places of the work, including tables, maps and illustrations taken from other works or the internet, either verbatim or in spirit, as borrowed, in any case indicating the source.

_____

Signature

Inhaltsverzeichnis

1 Abbreviation

Abkürzung	Beschreibung
LES	Large Eddy Simulation
RAS/RANS	Reynolds Average Navier Stoke
MRF	Multiple Reference Frame
AMI	Cyclic Arbitrary Mesh Interface

For more information refer to Schwarz [5]

2 Symbols

Symbol	Einheit	
$\boldsymbol{g}$	m/s^2	Gravitationen 9.81 m/sec^2
$\boldsymbol{p}$	Pa	Pressure
$\boldsymbol{u}$	m/s	Velocity
$\boldsymbol{\rho}$	kg/m^3	Density
$\boldsymbol{\eta}$	Pa/s	Dynamic viscosity
$\boldsymbol{\delta}$	MPa	Stress tensor
$\boldsymbol{h}$	J	Enthalpie
$\boldsymbol{d}$	m	Diameter
$\boldsymbol{V}$	m^3	Volume
$\boldsymbol{r}$	m	radius

3 Abstract

Siegfried in Zofingen is currently working on a hydrogenation reaction. To mimic the production condition for the production, the reaction is carried out on a scale-down reactor laboratory scale (approx. 1 liter). One important process parameter that should be similar during the experiments of the scale-down reactor compared to the later production scale reactor is the mixing behaviour of the agitator. The goal of this work is to find a concept for the appropriate scale-down of the agitator speed based on computational fluid dynamics. This concept shall be compared to conventional scale up methods. To validate this concept different experiment with water and glycerol, as well with and without baffles, were carried out in a 5-litre scale reactor. For the simulation without baffles, the size of the vortex has been compared to the real experiment as well as conventional calculation. In general, the simulation predicted only a vortex 63.6 % of the measured size. In further experiments with suspending of catalyst particles, simulation with baffles were validated with catalyst particle behaviour. The particle behaviour in the experiments could not be measured accurately due to the small size of the particles and the insufficient camera quality. The simulation with particles had a critical error where particles got stuck on the wall which made the results unusable for particle movement prediction. Even with the simulation working correctly, it would not be beneficial for this project, due to the large amount of particles and the simplification done for the simulation. A second method was uses, where the mean velocity of cells in the steady state were compared to each other. Out of 3 different agitator speeds, the agitator speed calculated with keeping P/V constant (571 rpm) was most similar to the agitator speed of the reactor MZA1 (120 rpm). Due to the low sample size this only supports the result but does not confirm it. As a consequence, the results of this thesis give an indication for a possible solution to the stirring behaviour but conventional ways of finding the agitator speed are still necessary. It can be used as steppingstone, but it is still required to refine these simulations. Despite the dissatisfying result with the catalyst particles it could be shown that scale-down of agitator speed can still be supported by the mean of computational fluid dynamics.

Acknowledgments

Throughout the writing of this thesis, I have received a great deal of support and assistance from multiple people without whom I would have not been able to complete this thesis.

I would first like to thank my supervisor, Prof. Dr. Andreas Zogg, whose feedback and assistance help me massively with trying to achieve better results and working in such a project environment. While it was new for me to work in a practice-oriented project and have more independence in decision making, he was always there to guide me and give me helping hand with problems that came up and were completely new to me. I'm sure that the experiences taken from working with Andreas will help me even in my future endeavours.

I would also like to thank my expert, Lukas Moser, for the patient support with everything concerning simulation work, even within a short notice. From the basic setup to the complex simulation, every question was answered. Even with a busy schedule he had time for multiple talks at each state of the thesis.

In addition, I would like to thank my parent. My father for providing additional computer for simulation work and even staying up till the middle of the night, just to send me the result. My mother for the mental support in the last stretch of writing this thesis.

4 Introduction

4.1 Aim of the thesis

For the company Siegfried AG in Zofingen, Switzerland a hydrogenation reaction is to be tested on a scale-down reactor of the original production reactor.

The aim of this work is to find a concept for the appropriate scale down of the agitator speed based on computational fluid dynamics.

To achieve the target result, computational fluid dynamic simulation should be tested as a new method instead of the historical scale down formula. The calculations were performed using the open-source program OpenFOAM. The approach is to use this software to simulate a simple experiment that can be practically verified in the laboratory. In a next step the same mixing tasks were simulated in a

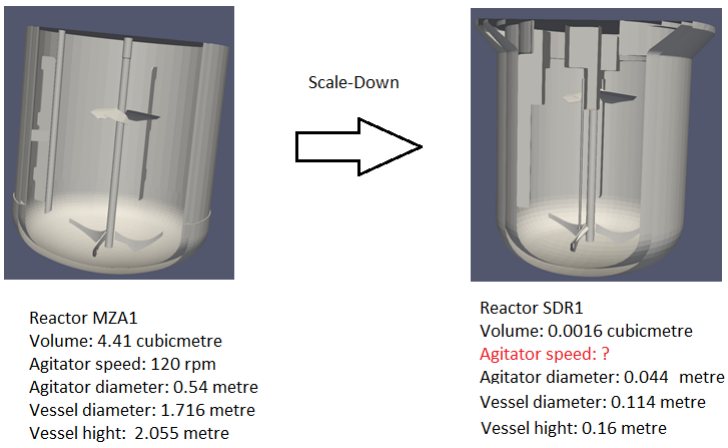


Figure 1 Production reactor MZA1 on the left, scale down reactor SDR1 on the right

production scale reactor (MZA1, see figure 1) and in a scale-down reactor (SDR1, see figure 1). The scale-down reactor was designed geometrically similar to the production scale reactor. The goal was to find the appropriate agitator speed that results in the same mixing behaviour of the MZA1 reactor compared to the SDR1 reactor. The verification of the simulation is done with optical analysis like the size of the vortex or particle distribution.

The ultimate goal of this work is to find an optimal stirring speed to achieve similar mixing of behaviour between the SDR1 reactor and the MZA1 reactor. To achieve this goal the thesis is divided into two phases. In the first phase, the simulation work will be validated. In the second phase, the stirring speed for reactor SDR1 will be determined. 120 rpm was the stirring speed usually used in the production reactor MZA1 (see fig. 1). In the end a stirring speed that creates a similar mixing behaviour in the scale down reactor SDR1 was the target.

The reason for choosing this topic is that conventional methods sometimes don't give the best results and are dependent on the calculating method (see app. 10.6). Since the technology of computational fluid dynamics has no defined pathway for the scale down of agitator speed and not a lot of people have worked with such a program, this thesis also serves as a short introduction into the field of fluid simulations and should give people with interest an insight and starting point for this technology. For this reason this thesis also goes into a standard case setup.

4.2 Theory

4.2.1 Conventional Scale-up

The scale-up method has long been used to test the properties and safety of reactions in large reactors. Reactor conditions are validated on miniature versions of the original reactor. One problem with this is that often not all the properties of the production scale reactor can be scaled down one to one. Examples of this are the specific heat transfer for the reaction mass to the reactor wall or the mixing behaviour. The down-scaling of the mixing behaviour is very important, especially for suspension tasks. In these cases, the focus is on ensuring that mixing behaviour is similar to the production reactor, as well as the quality of the mixing. To evaluate this, one of two different criteria is used in most scientific work. These criteria are on the one hand the 1-s criterion and the 90% criterion (Mattthias & Zehner, 1995). For the 1-s criterion, none of the particles may remain on the bottom of the reactor for longer than 1 second. This criterion is most frequently used in scientific work. The second criterion requires that the solid particles reach 90% of the liquid height. (EKATO, 2000)

Scale down suspension task

Over the last 70 years, several studies have been carried out to find a good rule for the scale-up of suspension processes. However, these have not led to any clear result (see app. 10.6). Most procedures are based on the rule $P/V \sim (dB/dM)^x$, where P is the power of the stirrer, V the volume of the reactor and d the diameter of the reactor. The problem with this rule is that the studies found different x values for different cases. For example, the x values fluctuate between -1 and 0.5 (Geisler et al., 1993). The large differences are due to the different suspension criteria, i.e., different requirements that should remain constant during scale-up (EKATO, 2000). Another problem is the different variables that must be considered, such as suspension properties, reactor size, stirrer tip speed and particle properties (Geisler et al., 1993). For example, keeping the volume specific power P/V constant can lead

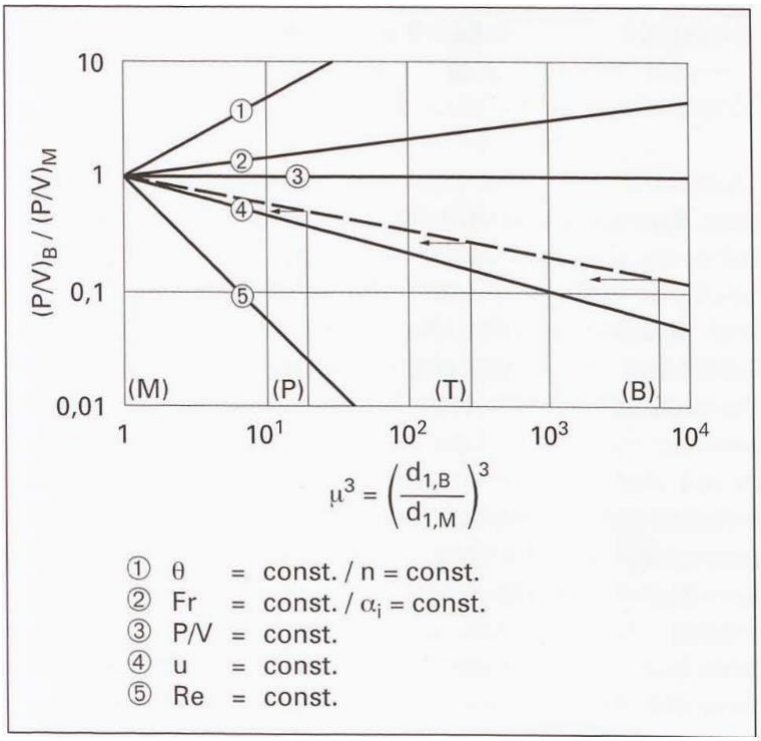


Figure 2 Penny diagram. Shows diameter ratio per volume depening stirring power. Lines are different stirring parameter staying constant. Source: (EKATO,2000)

to a value difference of the heat transfer or mixing time between the production reactor and the scale down reactor. (EKATO, 2000). Therefore, these rules only apply in certain sub-areas, but are not generally valid. The different laws found can be in the so-called Penny diagram which is shown in figure 2.

This means there is no simple generally applicable rule that describes the suspension process (Geisler et al., 1993). A good scale-up rule is only possible on a case-by-case situation and precise information on stirrer, particles and medium is required (EKATO, 2000).

4.2.2 Computational Fluid Dynamics

Since the advances of computer technology in recent years, another method has become increasingly important: The so-called Computational Fluid Dynamics simulations (CFD), which use computing power to simulate the behaviour in the reactor to varying degrees of accuracy. This makes it for example possible to simulate turbulent flows of non-compressible Newtonian fluids in various ways. The two main turbulence models are the Large Eddy Simulation (LES) and Reynolds Average Navier Stokes (RANS). Both models are based on the Navier-Stoke equations.

Impulse equation

$$\rho \left[\frac{\partial \underline{u}}{\partial t} + \nabla \cdot (\underline{u}\,\underline{u}) \right] = -\nabla p + \eta \nabla \cdot (\nabla \underline{u}) + \rho\, \underline{g}$$

Change of velocity with time — Movement of the field — Pressure Gradient — Viscosity Diffusion — External Forces: In this case gravity

(1)

The first part of the Impulse equation describes the speed of particles at a determined point. This is done by the partial derivative of the flow speed u. The second part is used to calculate the speed of a particle tint the entire fluid. So instead of focusing on a single point it follows a single particle. This is expressed by the material derivative of the speed u. This part corresponds to the pressure change, the gravitational force and the viscosity diffusion. This equation is the simplified version since the simulation work with an incompressible fluid. This is because it can be assumed that the density stays the same at every part of the equation which means the density can be seen as a constant value and be left out of the derivatives. This makes the later calculations with a much easier problem to solve.

Continuity equation / Mass conservation

$$\nabla \cdot \mathbf{u} = 0$$

Volume in flow and exit

(2)

This equation describes the conservation of mass within the field. As with the previous formula, it is the simplified version. The formula states that the gradient of the flow speed u is 0 in total. This basically means that the same amount of fluid volume enters the observed field as leaves it. Since

there is no density difference in the field, there is also no mass accumulation at any point within the field. For this reason, the partial derivative of density has been omitted.

Energy conservation equation

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot (\rho \underline{u} h) = \nabla \cdot (k \nabla T) + \frac{\partial p}{\partial t} + \cancel{\Phi}$$

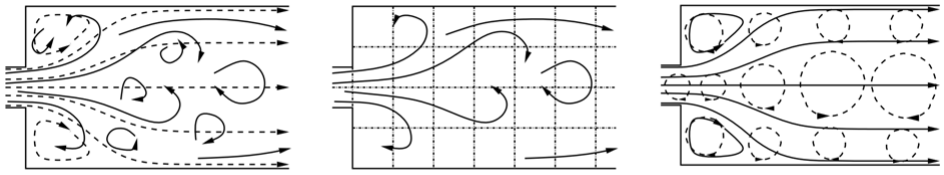
Local enthalpie change with time | Enthalpie change with the movement of the field | Heat flux | Pressure work | Heat dissipation term

(3)

The energy conservation equation describes on one side the change of enthalpy with both the material and the partial derivative. On the right side we have the change in energy heat flux, pressure work and heat dissipation.

Turbulence model

These equations are solved to a certain extent. In LES, turbulence is calculated numerically up to a certain size. Turbulences is vortices generated by fluctuations in state variables. The turbulence absorbs energy from the main flow and then decays into smaller turbulence. Smaller turbulences is then only approximated using only a model such as k-epsilon or k-omega (Schwarze, 2013). With RANS, only the mean values of the turbulences in one region are calculated with the equations and the rest is approximated with a turbulence model. The choice between RANS and LES usually depends heavily on the available time and computing power. LES provides a significantly more accurate result, because the equations for certain turbulence are solved directly, but it also requires more equations to be solved. For this reason, RANS is sufficient for most simple task. Because of this reason RANS was chosen for all simulation in this project.



Orignal | Large eddy simulation | Raynolds average navier stokes simulation

Simulation mesh

Another important element of CFD simulations is the construction of a suitable computational grid for the task at hand. A mesh consists of different cells for which the state must be calculated at each time step. This state is calculated using the three spatial directions of the Navier-Stoke equations. A very fine mesh would result in a high resolution, but also require a lot of computing power. A mesh that is too coarse needs only little computing power but will lead to an inaccurate or wrong result. To obtain the best possible mesh with sufficient size, we try to create a structured mesh that has a very uniform structure and thus making it easier to calculate the neighbouring cells in the mesh. A mesh of hexahedra is best suited for this purpose. However, this can lead to problems with more complex shapes, such as a stirrer. In this case, a mixed form is suitable, which consists of a structured mesh with

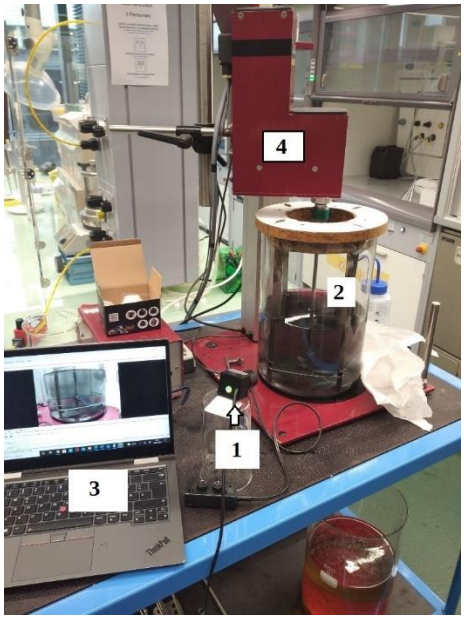
hexahedra in free space and adapted cells, i.e., different polyhedral. The polyhedral mesh would be used for the difficult geometry. The bottom line to try to create a mesh that, if refined, would not give a better result (Schwarze, 2013). In this thesis a mix of both mesh forms are used.

Simulation with movement

Another problem for CFD simulations arises when moving geometries, such as a rotating stirrer, are encountered. There are various approaches to solve this problem. One method is the so-called Cyclic Arbitrary Mesh Interface (AMI), where another cylindrical mesh is placed around the moving body, which is moved by a certain degree after each time step. For a very accurate simulation, the mesh must be moved in such small steps that no cell is skipped at the boundary of the inner and outer mesh. This approach results in small time steps. Consequently, a lot of computational effort and time is required. Another solution is the Multiple Reference Frame Method (MRF). In this method, a cylindrical mesh is first placed around the size of the object. Now, however, the mesh is not rotated, but a constant impulse is assigned to the cells, which corresponds to the movement of the object. A disadvantage here is that only constant movements can be considered, but not start-up and shut-down processes of a stirrer. In addition, the force always starts from the same point in the mesh and spreads only from this constant point(Delacroix et al., 2021). The agitator in this thesis was simulated with a MRF simulation.

5 Experimental Part

5.1 Experiment Setup



1: Camera

2: Vessel with stirrer

3: Laptop with recording program Kinovea

4: Stirring motor

Figure 3 Experiment setup used for experiment 1 to 12

5.2 Overview of the experiments

	Medium	Buffels	Pd-Catalyst	PE-particle	Stirringspeed	Simulated
1	Water	No	No	No	250	Yes
2	Water	No	No	No	325	Yes
3	Water	No	No	No	400	Yes
4	Water	Yes	No	No	250	Yes
5	Water	Yes	No	No	325	Yes
6	Water	Yes	No	No	400	Yes
7	Glycerol	No	No	Yes	250	No
8	Glycerol	No	No	Yes	325	No
9	Glycerol	Yes	No	Yes	250	Yes
10	Glycerol	Yes	No	Yes	325	No
11	Water	Yes	Yes	No	250	Yes
12	Water	Yes	Yes	No	400	Yes

Table 1: Experiments done with the simple geometry

Experiments 1 through 12 were all filmed at 120 frames per second at a resolution of 542x406. The parameters that were changed during the experiments are listed in Table 1. They are the type of medium used, whether baffles were used or not, whether a catalyst or PE particles were present, and finally what stirring speed was used.

To get a first comparison of the vortex size with the simulation, experiments were performed with water only. Experiments 1, 2, and 3 were performed without baffles to ensure that vortexing would occur. To obtain different vortex sizes, the stirrer speed was increased from 250 to 325 and 400 revolutions per minute (rpm). Once a steady state was observed, the vessel was videotaped for a few seconds. The stirrer was then turned off to determine the actual level of the water. Once the water had a velocity close to zero, the camera was turned off.

Experiments 4, 5, and 6 were performed with the same stirring speed as the first three experiments. The key difference was the use of 4 baffles attached to the edges of the vessel. These were taped in the same manner as the first 3 experiments.

For experiments 7 and 8, the same type of vessel was used as in the previous experiments. Glycerol was chosen as the medium. The glycerol was also filled with 2000 PE particles, possibly mixed with glass, as indicated by the measured density. Since the particles were difficult to record with a speed of 400 rpm, only speeds 250 and 325 were used. Experiments 7 and 8 were initially conducted without baffles, while experiments 9 and 10 were conducted with baffles.

The last two experiments 11 and 12 were performed with water, baffles and PE particles. The same camera setting and two different stirring speeds were used. The main purpose of this experiment was to determine if the catalyst particles could be observed during the process and if a difference in particle distribution could be seen at the two different speeds.

5.3 Standard Simulation Procedure

All simulations with OpenFOAM were performed according to a standard template. The steps were always the same and only different parameters and starting conditions were changed to fit the case being simulated. The following sections describe the different elements of the template based on experiment number 5.

The first step of each case is the preparation of a 3D-model. In some cases, the model has already prepared before the bachelor thesis. The models that have not been prepared are created using the CAD software Solidworks. Since OpenFOAM only work with stl-files, the model has to be saved as such (see Figure 4 and 5). For later parts of the simulation different parts of the model are required. This mostly includes parts of the model that have a finer surface structure. Each of these different parts must be stored separately in addition to the overall model.

Figure 4 Geometry-files used in template simulation. (/simulation/constant/triSurface)

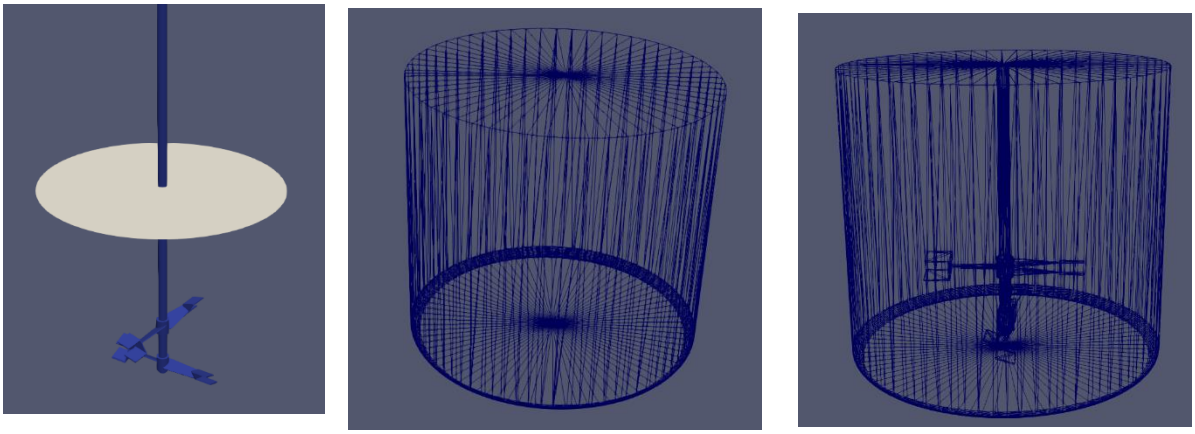


Figure 5 3D-geometry used in the template simulation. Left picture shows unten (stirrer - blue) and deckel (open part of the reactor, limiting the top part of the mesh - white). Middle picture shows vessel (reactor walls). Right picture shows all geometrys combined

The OpenFOAM standard cases consist of 3 different folders. These are 0, Constant and System (see Figure 6). The 0 folder contains all the starting conditions of the case, the Constant folder contains all the parts of the simulation that do not change, including the mesh of the 3D case, and the System folder contains the various active commands and functions for the simulation.

0	21.06.2021 16:35	Dateiordner
constant	21.06.2021 16:30	Dateiordner
system	21.06.2021 16:29	Dateiordner

Figure 6 Standard OpenFOAM simulation setup

Mesh creation

In all simulations the OpenFOAM tool snappyHexMesh is used. This tool also requires a specific folder structure. All stl-files created with Solidworks are stored in a folder under constant named triSurface. To use snappyHexMesh, there must to be a mesh that encloses all the geometry of the stl-files. The

```
convertToMeters 1; //unit for the coordinates

vertices // defines all 8 corner points of the hexahedron
(
    (-0.12 -0.4 -0.12) // Point 0
    (0.12 -0.4 -0.12)  // Point 1
    (0.12 0.4 -0.12)   // Point 2
    (-0.12 0.4 -0.12)  // Point 3
    (-0.12 -0.4 0.12)  // Point 4
    (0.12 -0.4 0.12)   // Point 5
    (0.12 0.4 0.12)    // Point 6
    (-0.12 0.4 0.12)   // Point 7
);

blocks // defines the hexahedron by naming the corner point order
       // second part gives the amount of cells in each coordinate direction
(
    hex (0 1 2 3 4 5 6 7) (150 500 150) simpleGrading (1 1 1) //
);

edges
```

Figure 7 creating a Mesh for the snapping process, point creation (/system/blockMeshDict)

blockMesh command helps to create such a mesh. The file for this command must be located in the

system-folder. In the first part of blockMesh the user specifies the coordinates of the 8 vertices and then in what order they form such a cuboid. This line of the file also specifies the number of cells inside the mesh in the 3 different directions. In order to achieve an optimal result, the number of cells should be chosen in such a way that the size of a cell edge is the same in each coordinate direction (see Figure 7).

```
patches // defines all 6 surfaces of the hexahedron
(
    patch maxY
    (
        (3 7 6 2)
    )

    patch minX
    (
        (0 4 7 3)
    )

    patch maxX
    (
        (2 6 5 1)
    )

    patch minY
    (
        (1 5 4 0)
    )

    patch minZ
    (
        (0 3 2 1)
    )

    patch maxZ
    (
        (4 5 6 7)
    )
);
```

Figure 8 Naming the 6 surfaces of the hexahedron (/system/blockMeshDict)

As a last step, each side of the box must be named with the 4 vertices (see Figure 8). The cube created based on these vertices can be seen in Figure 9.

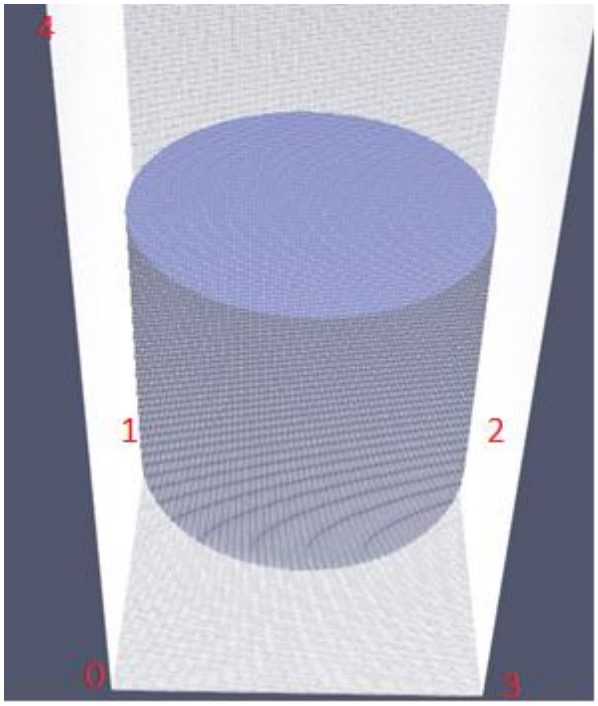


Figure 9 With OpenFoam generated blockMesh Mesh around the experiment geometry.

The next step to prepare **snappyHexMesh** is to extract the edge properties from the stl-files. For that the command **surfaceFeatureExtract** is used. The dictionary-file is also located in the system-folder. In the file it is specified which stl file should be used. Furthermore, the edges are selected that are smaller than the chosen refinement angle. Once this command is executed, files are created in constant containing various information for the selected surface edges (see Figure 10).

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

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

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

Figure 10 Extracting surface information from geometry that needs more defined snapping (/system/surfaceFeatureExtractDict)

snappyHexMesh

The next step in this case is to define **snappyHexMesh** bahaviour. The dictionary can again be found under system as **snappyHexMeshDict**. At the beginning of the file, you have to define which of the three steps, castellating, snappying and layering, should be executed. In the default template, the first

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


geometry // loads stl file geometry
{


    unten.stl {type triSurfaceMesh; name unten;}
    vessel.stl {type triSurfaceMesh; name vessel;}
    deckel.stl {type triSurfaceMesh; name deckel;}
    gesamt.stl {type triSurfaceMesh; name gesamt;}


    MRF // creats the mrf zone for agitator movement
    {
        type searchableCylinder;
        point1 (0 0.006 0);
        point2 (0 0.07 0);
        radius 0.060;
    }
};
```

Figure 11 Deciding the snapping method, loading geometry and creating the MRF zone (/system/snappyHexMeshDict)

two are set to true and the last one to false. Then the geometry files are loaded from the triSurface folder and given names that will be referenced later in the file. At this point, the MRF one is also defined. Since the stirrer fits nicely inside a cylinder, the searchableCylinder command is used. For the cylinder, the center of the top and bottom and the radius of the round side are needed (see Figure 11).

The next part of the files contains the information which geometry should be refined to which level. With maxGlobalCells the number of cells in the final mesh can be controlled, which is an important point for the calculation speed later on. The surfaceFeature created in a previous step is now loaded. After that, further details of the mesh are defined. Level 0 describes surface cells with the same size as the background mesh and with each level the size gets smaller. With refinementSurfaces is selected on which level the mesh should be refined on the surface. The first number describes the global level of the surface and the second number describes all surfaces that do not fit into the resolveFeatureAngle. Since an MRF zone is used in this case, the surface between the MRF zone and the mesh outside this zone must also be defined (see Figure 12).

```
castellatedMeshControls
{
    maxLocalCells 500000;  //max cells per CPU core
    maxGlobalCells 2000000; //max cells to use before mesh deletion step
    minRefinementCells 0;  //Amount of bad cells  allowed during refinement stages
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 2;  // expansion factor between each high & low refinement zone

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

    features // taken from STL from each .eMesh file created by "SurfaceFeatureExtract" command
    (
        {file "unten.eMesh"; level 4;}
        {file "vessel.eMesh"; level 1;}
        {file "deckel.eMesh"; level 1;}
    );

    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~
    refinementSurfaces // Surface-wise min and max refinement level
    {
        MRF
        {
            level (2 2); // refinment level global and local
            cellZone cellMRFzone; //naming the inside mesh part
            faceZone faceMRFzone; // naming the surface of the mesh
            cellZoneInside inside; // declaring which side is choosen
        }
    vessel {level (1 1);}
    deckel {level (1 1);}
    unten {level (4 4);} // was 4 4
```

Figure 12 Chosing the mesh quality, loading the surface information and deciding on the quality of refinement. (/system/snappyHexMeshDict)

Since part of the original mesh is being cut away, the program needs to know what to keep. This is controlled by the locationInMesh parameter, using a point inside the reactor (see Figure 13). The rest of the dictionary comes from the tutorial cases at tutorial/mesh/snappyHexMesh, which are included

```
    resolveFeatureAngle 70;  // Resolve sharp angles // Default 30
    refinementRegions        // In descending levels of fine-ness
    {



    MRF
        {
        mode inside;
        levels ((1E15 1));
        }



    }

    locationInMesh (0 0.15 0.03);  //to decide which side of mesh to keep **
    allowFreeStandingZoneFaces true;
```

Figure 13 Overall refinement of the mesh and which part of the mesh has to be kept (/system/snappyHexMeshDict)

in the standard version of OpenFoam. The mesh created based on the above description is shown in Figure 14.

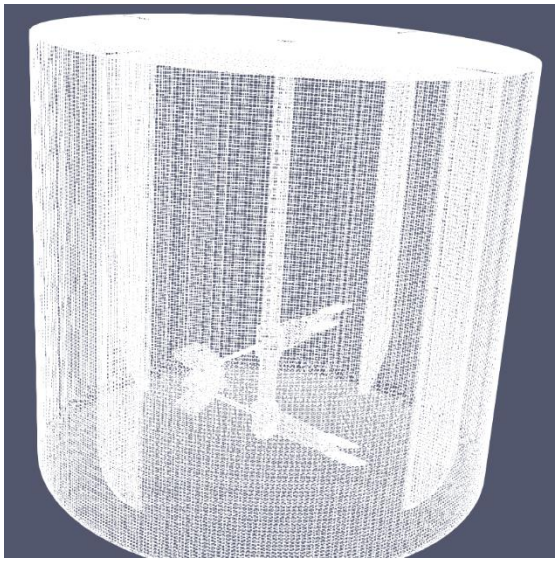


Figure 14 Finish mesh of the experiment geometry for validating simulations created by snappyHexMesh

"Constant" folder

Once the mesh is completed, all files in the "constant"-folder have to be edited. Since gravity is used to calculate the hydrostatic pressure, it must be defined in which direction the gravitational acceleration of the earth is. This is done with the file **g**. It has the dimensions with m and s^{-2} as well as the direction like -9.81 to Y (see Figure 15).

```
dimensions      [0 1 -2 0 0 0 0]; // units: [kg m s K mol A cd]
value           (0 -9.81 0); // coordinates [x y z]
```

Figure 15 Gravity constant (/constant/g)

Also, in the constant folder are **turbulenceProperties** of the simulation. It indicates what type of simulation you are running. In this template, the simulation is a Reynolds Average Stoke simulation with the k-Omega Shear Stress Transport turbulence model (see Figure 16).

```
simulationType RAS;

RAS
{
    RASModel        kOmegaSST; //turbulence model

    turbulence      on; //turbulent or laminar

    printCoeffs     off;
}
```

Figure 16 Turbulence model used: Raynolds Average Navier Stokes (/constant/turbulenceProperties)

Another file is the **transportProperties** file which contains the physical characteristics of the fluid or gas which are used in the simulation. In this case we got density and viscosity of water an air. The last parameter is sigma which stands for the surface tension (see Figure 17).

```
phases (water air); // defines the two phases

water //phase 1
{
    transportModel  Newtonian; // Newtonian fluid
    nu              1e-06;     // viscosity
    rho             998.2;     // density
}

air //phase 2
{
    transportModel  Newtonian; // Newtonian fluid
    nu              1.52e-5;   // viscosity
    rho             1.2;       // density
}

sigma           0.07; // surface tension
```

Figure 17 Information about the two fluids in the simulation (/constant/transportProperties)

The last part of the constant folder is the **mrfProperties**. This file contains the parameters for the rotation of the stirrer. With cellZone it is chosen which part of the mesh should include the movement. After that the center of the rotation axis is written to the origin line. The last line has the rotation speed of the mrfZone with the angular velocity omega. This velocity is calculated as OMEGA = 2 Pi / rotation speed (see Figure 18).

```
MRF1
{
    cellZone    cellMRFzone; // named in snappyHexMesh
    active      yes;

    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches ();

    origin    (0 0.038 0); // Middle of the mrf zone
    axis      (0 1 0); // Rotation axis
    omega     constant -34.034; // Angular velocity omega=2*Pi/T, T=Rotationtime
}
```

Figure 18 Defines the MRF zone (constant/mrfProperties)

"0" - folder

After the mesh is ready and the constant folder is set, the 0 folder can be created. The solver interFoam needs a few starting parameters. These are the turbulence properties **k** and **omega**, the calculated hydrostatic pressure **p_rgh** and the velocity **U**. Each starting condition must be defined for the entire free-standing mesh as well as for the patches that form the edges of the simulation.

The starting velocity in a cell is zero at the beginning for each direction. This is represented by the line internalField. Since the velocities at the walls are assumed to remain zero throughout the simulation, the value for them is also set to zero. The only exception is the top of the reactor, to assume an open top geometry (see Figure 19).

```
dimensions    [0 1 -1 0 0 0 0]; // m/s

internalField uniform (0 0 0); // velocity in the free standing cells

boundaryField

{
    deckel //cells at the top edge of the mesh
    {
    type            pressureInletOutletVelocity; //not a wall but a "open surface"
    phi             phi;
    value           uniform (0 0 0); // 0 in all directions
    }

    vessel
    {
    type      fixedValue; // value is fixed for the whole simulation
    value     uniform (0 0 0); // value stays zero the whole time
    }

    strom
    {
    type      fixedValue;
    value     uniform (0 0 0);
    }

    oben
    {
    type      fixedValue;
    value     uniform (0 0 0);
    }

    unten
    {
    type      fixedValue;
    value     uniform (0 0 0);
    }
```

Figure 19 Defines the starting condition of the velocity (/0/U)

The parameters for the turbulence properties omega and k have a similar value. Since there is no motion at the beginning of the simulation, there also is no turbulence. Therefore, like U, both parameters can be set to zero (see Figure 20).

```
dimensions      [0 2 -2 0 0 0 0]; // m^2/s^2

internalField uniform 1e-08; // free standing mesh, close to 0

boundaryField

{
    deckel
    {
    type      zeroGradient; // no change between at the surface of the mesh
    }

    vessel
    {
    type      kqRWallFunction; // special wall function for k
    value     $internalField; // calculates from the field
    }

    strom
    {
    type      kqRWallFunction;
    value     $internalField;
    }

    oben
    {
    type      kqRWallFunction;
    value     $internalField;
    }

    unten
    {
    type      kqRWallFunction;
    value     $internalField;
    }
}
```

```
dimensions      [0 0 -1 0 0 0 0]; // 1/s

internalField uniform 0.0001; // free standing mesh close to 0

boundaryField

{
    deckel
    {
    type      zeroGradient;
    }

    vessel
    {
    type      omegaWallFunction;
    value     $internalField;
    }

    strom
    {
    type      omegaWallFunction;
    value     $internalField;
    }

    oben
    {
    type      omegaWallFunction;
    value     $internalField;
    }

    unten
    {
    type      omegaWallFunction;
    value     $internalField;
    }
```

Figure 20 paramter k and omega defined at the start of the simulation (0/k) and (0/omega)

Another parameter that is calculated at each step is the hydrostatic pressure effect. This describes the pressure created by the water level above a certain point. All that is needed is the density, which is constant in this case anyway, the gravitational force defined earlier, and the height of the water level, which can be easily calculated from the simulation. That is, the parameter starts with a default value of zero and takes its initial value in the rest of the field from the initial water level.

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

internalField uniform 0;

boundaryField

{
    deckel
    {
    type            totalPressure;
    p0              uniform 0;
    U               U;
    phi             phi;
    rho             rho;
    psi             none;
    gamma           1;
    value           uniform 0;
    }

    vessel
    {
     type         fixedFluxPressure;
     value        $internalField;
    }

    strom
    {
     type         fixedFluxPressure;
     value        $internalField;
    }

    oben
    {
     type         fixedFluxPressure;
     value        $internalField;
    }

    unten
    {
     type         fixedFluxPressure;
     value        $internalField;
    }
```

“System” - folder

The preparation for the simulation is completed with the system files. One of the basic files is the **setFieldsDicts**. This file is used to set the filling level of the geometry for one fluid. The first step is to set alpha.water = 1 for each individual cell, i.e. it takes the water parameter from the alpha.water file in folder 0. This means that the entire geometry is filled with water. The next step is to cut out the part that is not to be filled. To do this, create a box with the 2 diagonal vertices and set everything in this box to alpha.water = 0. In this case it would be replaced by air (see Figure 22).

```
defaultFieldValues
(
 volScalarFieldValue alpha.water 1 // fills the entire mesh with water

);
regions
(
 boxToCell
 {

 box (-0.12 0.16 -0.12) (0.12 0.4 0.12);  // creates a box, all cells in these
 fieldValues                              // boxes have the value water = 0
        (                                 // means that this area is filled with air
            volScalarFieldValue alpha.water 0
        );
 }
);
```

Figure 22 defines the area which are filled with air and water (/system/setFieldsDict)

Another dictionary is the **controlDict** which is used to run the whole simulation.The first line is the solver type that will be used for the simulation. In this case it is interFoam. Afterwards, the start and stop time for the simulation are set, as well as the preferred time increments between each calculation. The simulation will only save the data at times set in writeInterval. The last important item in this file is that runTimeModifable is enabled. This means that the time step between calculations can be changed depending on the case without testing different deltaT. DeltaT in this case, is the time

```
application       interFoam; // solver

startFrom         latestTime; // Start time, either: startTime or latestTime

startTime         0; // if startFrom startTime, startTime defines start in sec

stopAt            endTime; // end of simulation

deltaT            0.001; // timesteps

endTime           30; // end of simulation time

writeControl      adjustableRunTime;

writeInterval     0.25; // results gets written down, in sec

purgeWrite        0;

writeFormat       binary;

writePrecision    9;

compressed        no;

timeFormat        general;

timePrecision     6;

runTimeModifiable yes; // deltT can be changed

maxCo             0.7; // max Courant number

maxAlphaCo        0.5; // max Alpha Courant number

adjustTimeStep    yes;

maxDeltaT         1;
```

Figure 21 controls the run of the simulation (/system/controlDict)

step calculated in the previous calculation. In these simulations, deltaT would be approxmateley 0.0001 seconds. This change is made with the maxCo and maxAlphaCo values, both of which indicate

the maximum degree of uncertainty allowed. In the simulation, the selected deltaT cannot be larger if this means that the maxCo number exceedss the number selected at the beginning (see Figure 21).

T The last file used in the simulation setup is the **decomposeParDict**. Simulations take a lot of time and are usually run on one CPU core. To increase the calculation speed, it is possible to divide the network into different sections and have them executed by different CPU cores. In this template, six cores are used. In the first part of the dictionary, the number of cores is specified as numberOfSubdomains. The next step is to choose how the network will be split. In the Scotch method, it is calculated so that each core has approximately the same amount of work during the simulation (see Figure 23).

```
numberOfSubdomains 6; // # of CPU cores

method          scotch;

simpleCoeffs
{
    n               ( 1 1 6 ); // needs to multiply to = # cores
    delta           0.001;
}

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

manualCoeffs
{
    dataFile        "cellDecomposition";
}
```

Figure 23 describes how the mesh gets devived for calculation with multiple cpu (/system/decomposeParDict)

5.4 Particle Simulations

The previous template in 5.3 was applied to a simple fluid simulation. Since the goal of this thesis is to predict catalyst distribution, particles had to be added to the simulations. For this purpose, a new solver called icoUncoupledKinematicParcelFoam was chosen. This solver works different from the solver used in 5.3. For this solver, defined particles follow the velocity field of the simulation. It is assumed that the particles have no impact on the velocity field. This means the velocity field is stationary for the entire simulation. The basic structure with the 3 different folders remained the same. One of the biggest changes from the last case is that the parameters in folder "0" have been reduced to just U. This is the velocity field taken from the simulation describes in 5.3. In practice this means, that the parameter U in the last time step of the simulations done in 5.3 get copied to the "0"-folder of the simulation in 5.4. Parameter k, omega and pressure are not necessary.

The biggest change from the previous template is the addition of the kinematicCloudPositions and kinematicCloudProperties files. The first of these two files describe the starting position for each particle at the beginning of the simulation. This is done using the three coordinates. The second file contains more information about the parameter of the particles. In constant properties the physical properties of th particles are definied. Necessary points are the density, the modulus of elasticity via the Young's modulus [6] and the deformation of the particle via the Poisson's ratio (see Figure 24).

```
solution
{
    active          true; //file used for simulation
    coupled         false;
    transient       yes;
    cellValueSourceCorrection off;
    maxCo           0.3;
    interpolationSchemes
    {
        rho             cell;
        U               cellPoint;
        mu              cell;
    }
    integrationSchemes
    {
        U               Euler;
    }
}
constantProperties
{
    rho0            1495; // density particles
    youngsModulus   6e8; // elasticity
    poissonsRatio   0.35; //deformation
}
```

Figure 24 describes particle parameters (/constant/particlesProperties)

Another part of this file is the injectionmodel where the time and the size of the particle are defined. In this case, it is not necessary to define the mass of the particles, since the number of particles is read from the positionsFile. The mass is then calculated using the number particles, the density and the diameter of the particles. It is still necessary to specify the size of the particles, either with a fixed value for the diameter as in this example or with a distribution model (see Figure 25).

```
subModels
{
    particleForces
    {
        sphereDrag; //slowing down due fluid
        gravity;    // gravity forces
    }
    injectionModels //adding particles to the simulation
    {
        model1
        {
            type            manualInjection;
            massTotal       0; // not relevant for this case
            parcelBasisType fixed;
            nParticle       1;
            SOI             0;
            positionsFile   "kinematicCloudPositions"; // file with particle xyz
            U0              (0 0 0);
            sizeDistribution // size of the particles
            {
                type        fixedValue; // all same size
                fixedValueDistribution
                {
                    value   0.0028319; // diameter in metre
                }
            }
        }
```

Figure 25 describes the adding of particles into the simulation (/constant/particleProperties)

The last part of this file defines the interaction between particles and patches/walls in the simulation. Since interaction between particles take calculation time, they are not simulated. The interaction between particles and the wall is simulated with the wall type “rebound”. This type is defined by the coefficient of restitution and elasticity, which describe with what velocity particles are moving away from the wall after hitting it (see Figure 26).

```
    localInteractionCoeffs
    {
        patches
        (
            vessel
            {
                type rebound;
                e    0.97; // coefficient of restitution, 1 no lose of velocity
                mu   0.09; // elasticity coefficent
            }
            unten
            {
                type rebound;
                e    0.97;
                mu   0.09;
            }
            strom
            {
                type rebound;
                e    0.97;
                mu   0.09;
            }
            deckel
            {
                type escape;
            }
        );
    }
```

Figure 26 describes the wall-particle interaction (/constant/particleProperties)

6 Results and discussion

6.1 Vortex experiments

In this section of the thesis, the simulation used for the validation are presented. These simulations correspond to experiment 1, 2 and 3 in table 1. Since there was no reference data available for simulations with an agitator system, they had to be validated. For this reason, a simple geometry was chosen, to reduce errors that could affect the outcome of these simulations. A agitator in a cylindrical vessel was chosen as the simplest simulation. It is expected that an agitator creates a vortex. Furthermore, it is expected that the size of the vortex depends on the agitator speed (see app. 10.5). This magnitude could be measured. For this reason, simulations were performed at 3 different stirrer speed, 250, 325 and 400 rpm, to obtain different vortices. Water and glycerol were chosen as media for this experiment. Due to high viscosity of glycerol, no significant vortices were measured at the selected speeds. With water, there was a clear result.

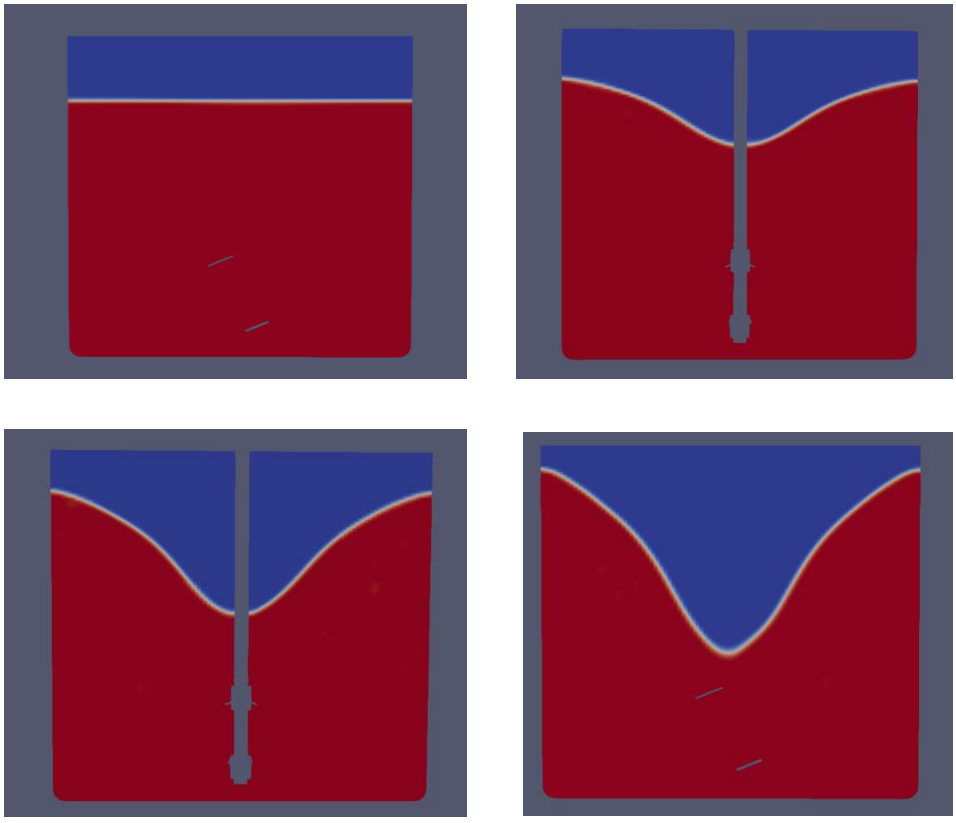


Figure 27 All 4 diagrams show a slice through the middle of the vessel. The red part represents the vessel parts filled with water and the blue stands for parts filled with air. Upper Left: Starting water level for all 3 simulations. Upper right: Water level for the steady state with 250 rpm. Lower left: Water level for the steady state with 325 rpm. Lower right: Water level for the steady state with 400 rpm

Figure 27 shows a section through the centre of the vessel and the stirrer. The red parts of each diagram represent the water and the blue parts represent for the air in the simulation. The first graph shows the initial state of the simulation, and each subsequent graph shows the level for a stirring speed of 250, 325 and then 400 rpm. For the different stirring speeds not the same time step was chosen. The reason for this is that each simulation did not require the same amount of time to reach a steady state where the vortex size remains approximately the same over a long period of time. The faster stirring speeds reach the desired steady state much faster. The run time selected for the simulations was 65.75 seconds for 250 rounds per minute, 37.75 seconds for 325 rounds per minute and 45.75 seconds for 400 rounds per minute. The size of each vortex was then measured with respect to the initial start condition. At 250 rpm, the vortex was 2.95 cm deep, while it was 16 cm high at the start. To test this result two different methods were chosen. The first was to estimate the vortex depth using calculations. For this purpose, the formula used in Zlokarnik [7] was used (see app. 10.5). This gives a result for general propeller stirrer problems. In this case, the solution of the formula was 4.96 cm. The second method was to test the problem in a real experiment. For this water was poured into the same vessel as in the simulation and filled up to the same level. As in the simulation, the measurement was started when a visible

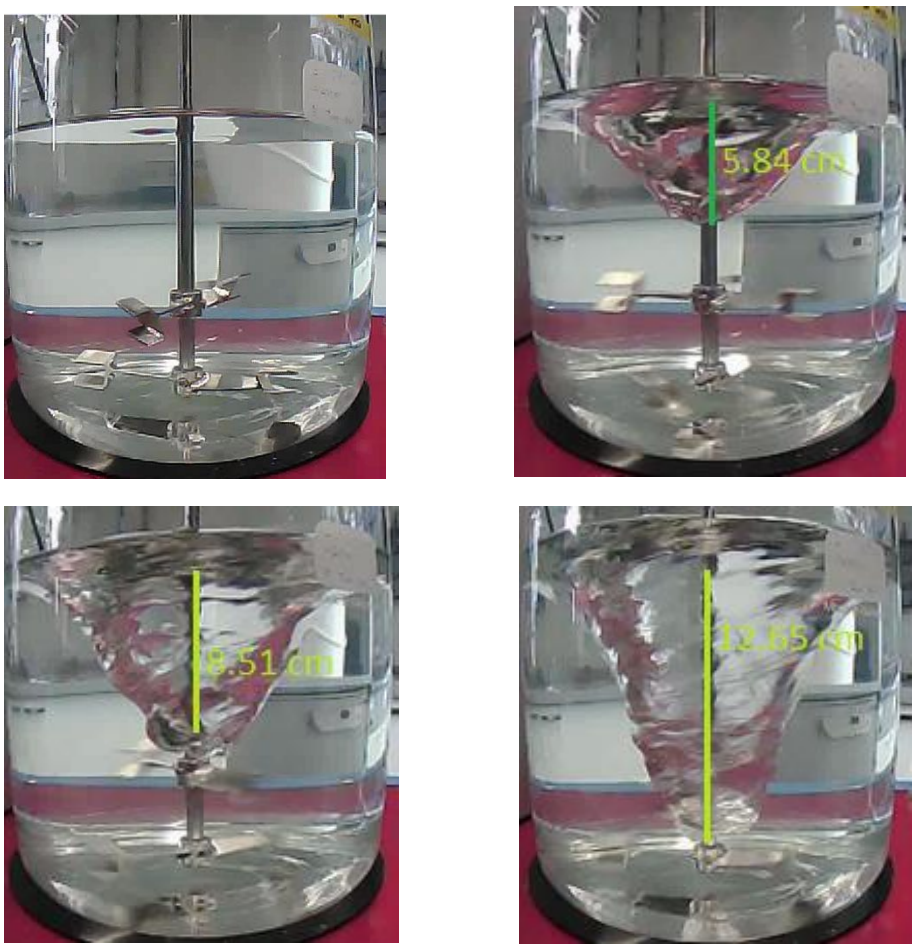


Figure 28 All 4 pictures show the simple experiment setup filled with water 16 cm high. Upper left: The starting condition of the experiment. Upper right: Water level at a steady state with 250 rpm. Lower left: Water level at a steady state with 325 rpm. Lower right: Water level at a steady state with 400 rpm

steady state was reached. The result of these experiments is shown in Figure 28. At 250 rpm the vortex had a size of 5.84 cm depth. The same was done for the other two stirring speeds. At 350 rpm, the vortex had a calculated size of 8.38 cm and measured depth of 8.51 cm. The simulation gave a result of 5.3 cm. At 400 rpm, the measured size was 12.65 cm, the formula gave 12.7 cm and the simulation gave a vortex with 8.4 cm (see table 2). This means that the simulated vortex was on average only 63.6 % of the real height.

Agitator speed in rpm	400	325	250
Measured size in cm	12.65	8.51	5.84
Simulated size in cm	8.4	5.3	2.95
Calculated size in cm	12.7	8.38	4.96

Table 2 Vortex size of the validation experiment, Result of calculation, measuring and simulating

A clear trend can be seen in these results. While the calculated approximations are close to the real depth, the simulations are always predicting a smaller vortex. There are a several explanations for the large difference between reality and simulations.

1. The simulation is performed with a multiple reference frame instead of a cyclic arbitrary mesh interface. If an AMI had been used, the mesh would have moved with the time step. Instead, the simulation gives everything in the MRF zone a specific torque to achieve the desired rotation speed. This can be clearly seen in the graph in Figure 29 and Figure 30 below. It shows the velocity at the beginning and at the end of the simulation for the case of 400 rpm. The holes in the diagrams are the positions of the stirrer. It shows that the position always remains the same and therefore the position from which all force is applied to the water medium. This also means that there is no force behind the stirring blades. As the second diagram shows, there is a lack of velocity in the middle part of the simulation behind the blades. It would clearly have been more accurate to use the AMI method, but as mentioned in the first part of this thesis, this would also take a lot of computational time. And this is also the main reason for all these small simplifications in the simulation. Since time is limited, the computation time must be shortened to get the desired result before a deadline.
2. Another simplification used was the Reynolds Average Simulation where the small turbulences are only approximated. With the turbulence model k-omega Shear Stress Transport a good result is still possible, but the results are significantly less accurate compared to a large eddy simulation where turbulences get calculated to much finer detail.
3. Another thing that could contribute to the difference in the simulation are the choice of time steps. Since these simulations are based on differential equation, you get to the next time step from the previous one as a starting condition, but the amount of time between these steps can be chosen according to the simulation. The smaller these timesteps are the more precise is the result. In this simulation the time steps get regulated with a variable called courant number which is calculated using the mean of the volumetric flux. In literature a courant number under 0.3 or 0.4 is recommended. With the limited time left, this variable had to be raised in most cases to 0.7. While this is still under the starting value of 1 it influences the end result and could also be part of the difference measured.

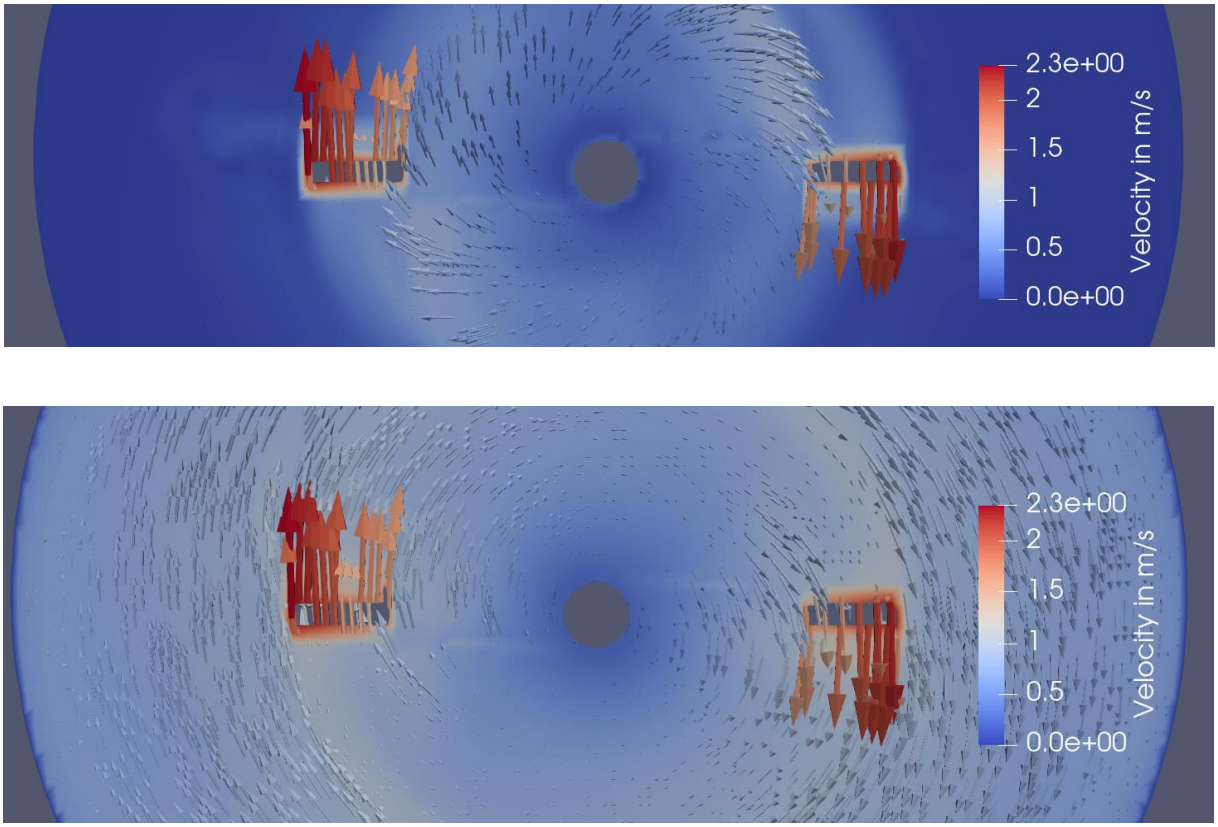


Figure 29 Both diagrams show a horizontal slice through the vessel at the height of the agitator. The figure show the velocity field with velocity vectors at certain points of the simulation with 400 rpm. Both holes with the high velocity vectors are the location of the stirring blades. Upper: Velocity field after 0.5 seconds being simulated. Lower: Velocity field after reaching steady state (45.75 seconds)

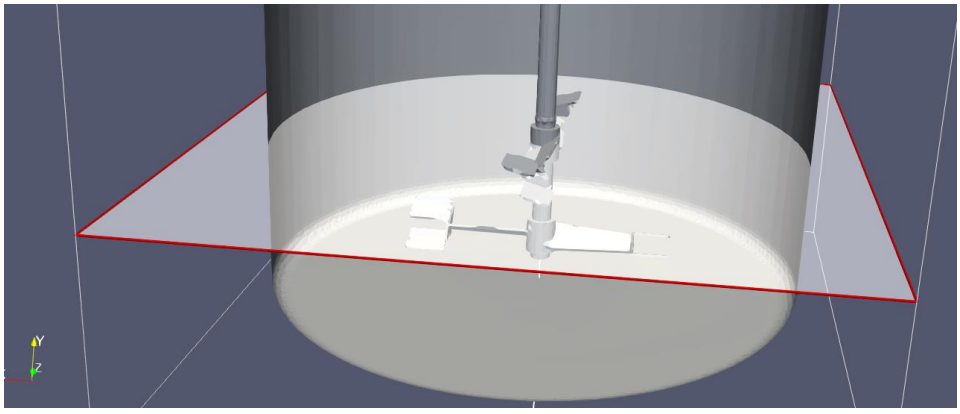


Figure 30 shows the slice throught the simulation at the agitator. Sections are shown in Figure 29

6.2 Experiment with water and baffles

The next step of the experiments was to investigate the influence of baffles on the fluid flow. For this purpose, the same geometry as in the previous simulation in chapter 6.1 was used but this time with 4 symmetrically opposed baffles. For the rotation speed, 400 and 250 rpm were chosen. Since there was no vortex size to measure, the steady state was chosen by keeping the speed approximately constant at different points and time steps. A first comparison between 400 rpm with and without baffles already shows a significant difference between the flow in the medium. Figure 31 shows the velocity field in a section through the middle part of the vessel for the simulation with and without baffles. The left diagram shows the vessel without baffles and it is clear that all velocity vectors in the

direction of rotation are in an almost straight line, which also explains the vortex. The vessel with baffles, on the other hand, shows a much stronger upward and downward motion. Because of the stirrer used in this experiment, the flow goes down inside the vessel and rises on the outside of the vessel. There is still motion in the direction of rotation, but most of the velocity is concentrated in the axial flow. It is also noticeable that the overall velocity in the vessel with the baffles is slower than that in the vessel without the baffles. This simulation behaves as expected, at least to the naked eye. The best method to validate this would be to measure the velocity inside the vessel, which was not possible in the short time available. Another method would be to have some particles inside the vessel and track the flow of the particles. This strategy will be discussed in the next part of this thesis.

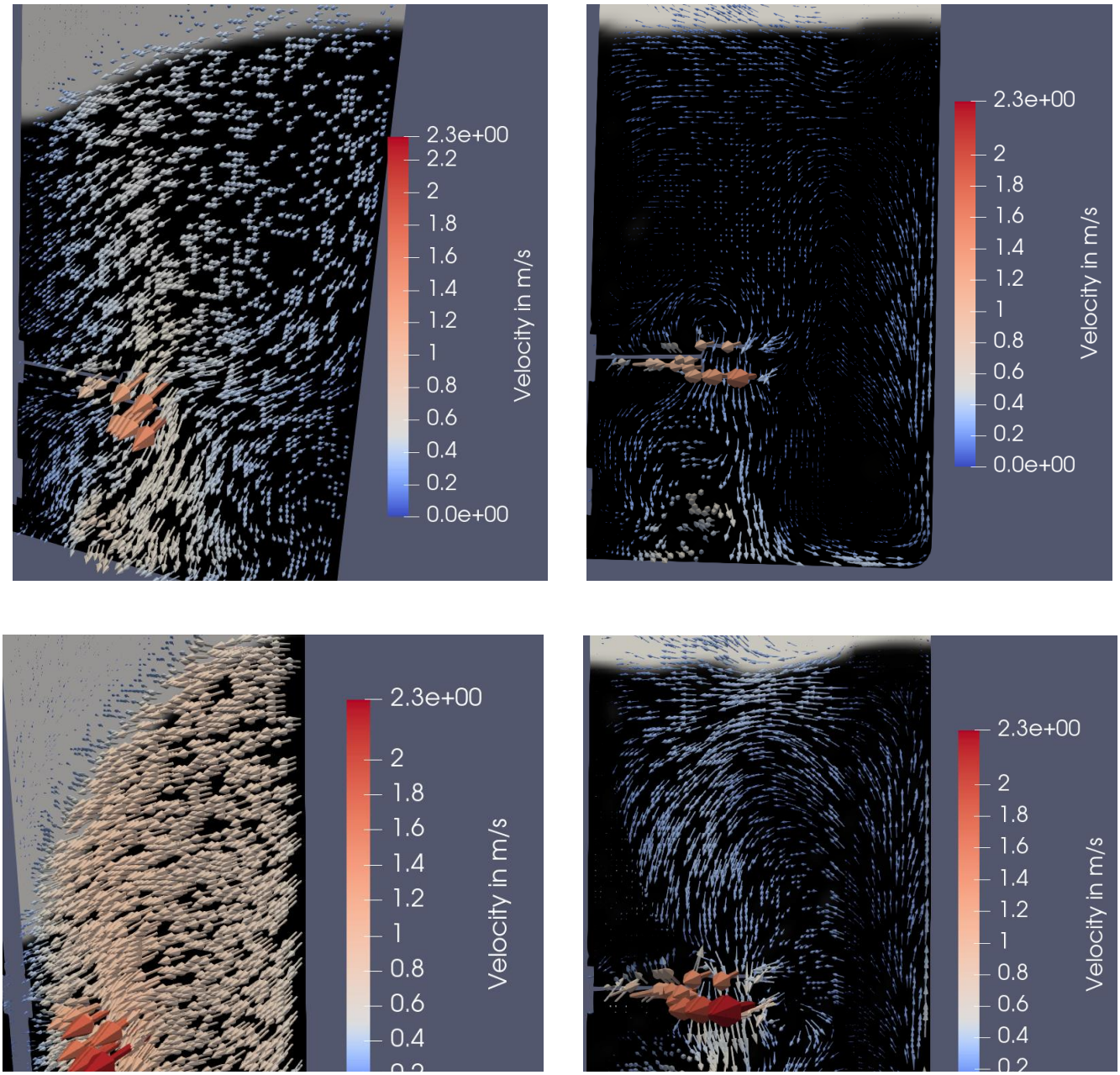


Figure 31 The 4 diagrams show a vertical slice through the middle of the vessel. Shown is only one side of the slice. All diagrams have the velocity vectors coloured after the value. The black background represents the water level at the time of the simulation. The grey parts are air. Upper left:Velocity field at the steady state, taken from the simulation without baffles and 250 rpm. Upper right: Velocity field at the steady state, taken from the simulation with baffles and 250 rpm. Lower left: Velocity field at the steady state, taken from the simulation without baffles and 400 rpm. Lower right: Velocity field at the steady state, taken from the simulation with baffles and 400 rpm.

6.3 Particle simulation with water

The next part of the thesis was to test how the catalyst from the main project would behave in the simulation and compare this to reality. In the original experiment with the scale down reactor, 0.1 g of the catalyst was used with about 1 litre of fluid. Since the reactor was filled with 5 litres of water 0.5 g of catalyst would be the appropriate amount in the simulation. A scanning electron microscope was used to determine the average size of the catalyst particles (see app.10.3). Since the particles had a variety of sizes, 3 groups were formed. 2 µm, 15 µm and 100 µm and an average occurrence was given for each group. Using the density and mass, the number of particles could now be calculated. This resulted in a particle count of around 80 million. A simulation with this number would not have been possible in the time limit. For this reason, the particle count was scaled down to a reasonable size by a factor of 100. This should still show a trend in the simulation result.

The next step was to find a reasonable analysis of the simulation result. For this reason, the mesh was divided into 6 different parts of interest, as shown in Figure 32. The first part is 1 cm above the bottom of the vessel, the second and third parts are a cylinder around the stirrer from the bottom 1 cm to the top 15 cm. They are divided in the middle at 7.5 cm. The fourth and fifth parts are the outside of the cylinder, divided in the same way as the previous parts, and the last part is the top 1 cm of the water level. There are several reasons for this choice. One of the most important points for good mixing is that most of the particles leave the bottom of the vessel. Another reason is how well the particles are distributed in the vessel.

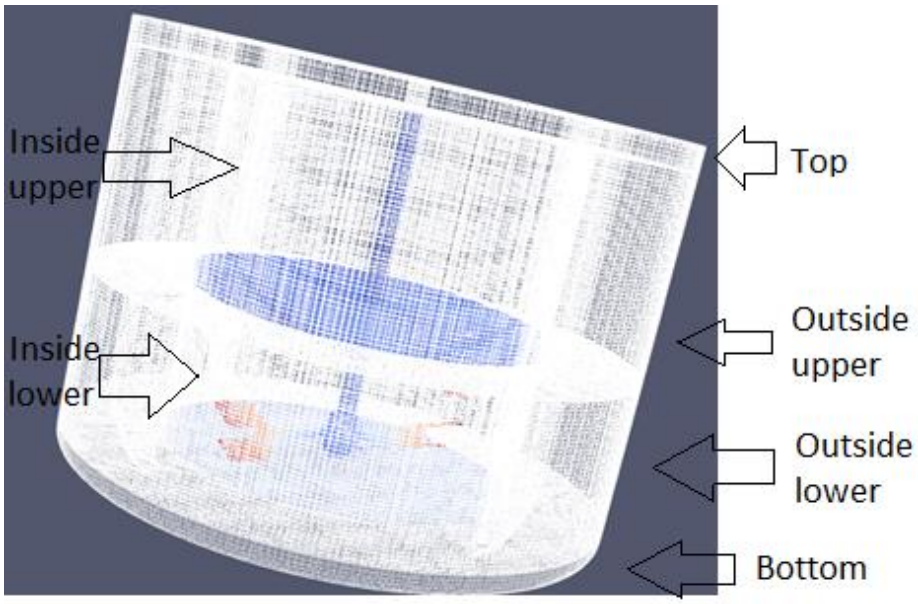


Figure 32 Diagram shows the mesh used in the analysis of the simulation. It is divided into 6 parts. Bottom, a cut at 6% of the height, inside lower, the area around the stirring blades up to 50 % of the height, inside upper, the area above the inside lower part up to 94 % of the height, outside lower same hight as inside lower but the remaining mesh, outside higher, the part above inside lower up to 94 % and top, a slice of the last 6 % height.

To make the different parts comparable, the amount of particles is calculated as a function of volume. Speed of 400 and 250 rpm were chosen for the simulation. Figure 34 shows the number of particles in each part as a function of volume. In both tables, it can be clearly seen that a large portion of the particles remain at the bottom or return there. Overall, 36.5 % of the particles remained at the bottom of the vessel at 400 rpm and 35.2% at 250 rpm. It can be clearly seen that most of the particles are still in the bottom half of the vessel at 85% and 81% respectively. It can also be seen that the axial flow brings more particles to the centre (23.8 particles per cm^3) than to the outside (15.5 particles per cm^3). However, both figures do not give the whole picture, since a respectable number of particles remain attached either to the baffles or directly to the stirring blades, which lowers the distribution rate inside the medium, as it can be seen in figure 33.

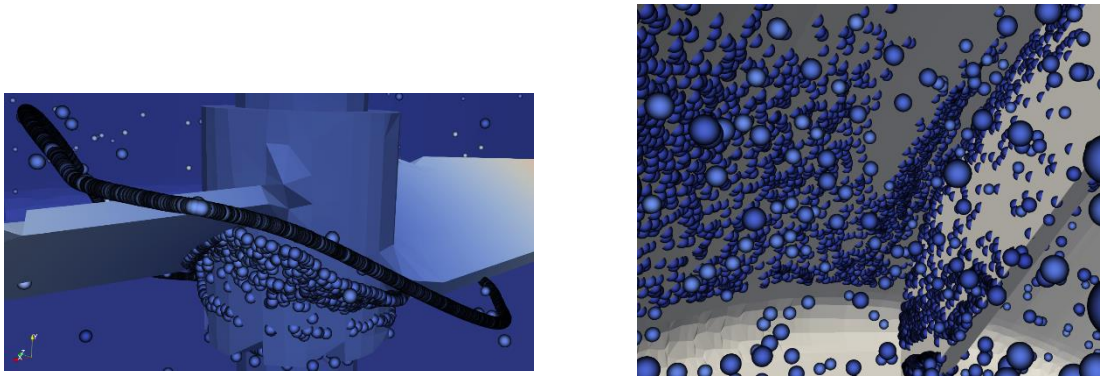


Figure 33 Left: Particles being stuck at the stirring blade at 400 rpm. Right: Large amount of particles stuck on the outside wall at 400 rpm

Particles being stuck

After all the simulation were run, a major flaw was found in the concept of simulation work. To start a simulation, all the initial conditions of the mesh must be defined. This includes the edge region of the mesh ,also known as patches. These patches must know the interaction between the outside of the mesh and the inside. As described in the standard templete cases, normal fluid simulations use either the noslip function or the fixedValue function. The fixedValue function is used to set the velocity to a uniform value of 0 in all 3 directions in normal cases. This is done because it is normally assumed that directly on the wall the adhesion is greater than the cohesion of the fluid. OpenFoam accomplishes this by setting the closest cell near the solid surface to a velocity of zero. This constant velocity is maintained throughout the simulation and in the particle simulation. In the particle simulation, the patches are given what is known as a rebound condition, in which particles that hit a surface bounce off with a small loss of energy. To reach the surface, they must traverse the 0 velocity field. In doing so, they do not gain any new energy, but they still lose some due to friction with the liquid medium. Particles that thus lose all their energy and come to a stop are now trapped in the 0 velocity cells and receive no further momentum throughout the simulation. This phenomenon can be seen in Figure 34 where both particles are on the wall and near the stirring blades. They physically stick to the wall, and the number of stuck particles only increases during the simulation. This leads to a contradiction, where a long simulation time is beneficial to get a more accurate result, but at the same time increases the inaccuracy due to the stuck particles. There are some solutions to this problem, such as using a different interpolation scheme instead of giving the wall a constant value of zero, it would be just close

to zero. Another solution would be add a small layer above the surfaces to get a preferably small cell. This would then again lead to a increase in calculation time.

Besides this problem, these results can also be partially explained by the simplifications made to perform this simulation. This simulation performed with the icoUncoupledKinematicParcelFoam, which is used to simulate simple particle motion. In this case, we need to simulate the fluid flow already without the particles,as

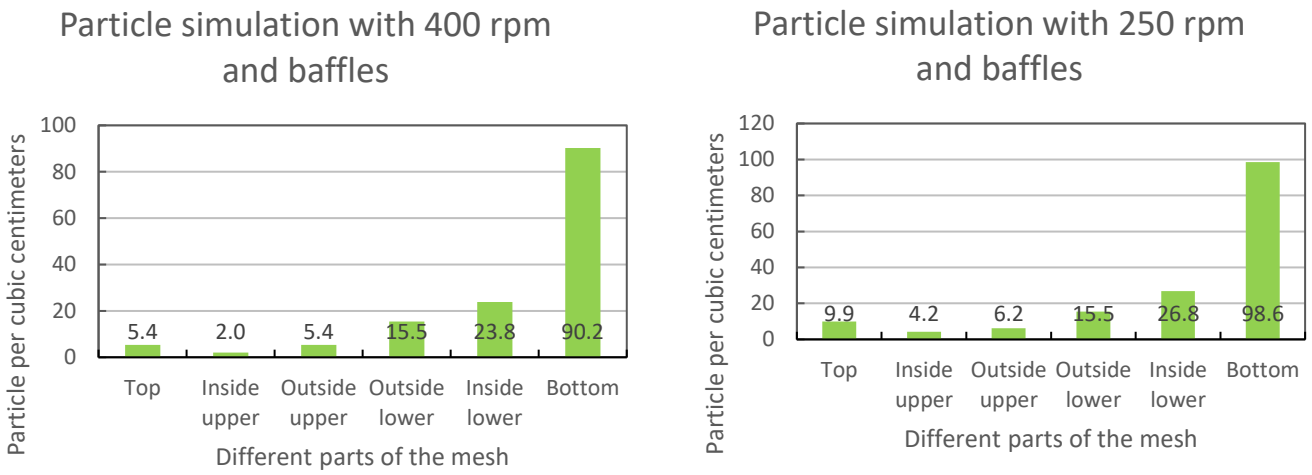


Figure 34 Both graphs show the particle distribution at the steady state of the simulation. This is done with particle amount depending on volume in the chosen area. The unit used is particles per cubic centimetres of volume. Left: Particle distribution with baffles and 400 rpm. Right Particle distribution with baffles and 250 rpm

it was done in the previous chapter 6.2. This also means that erors and simplifications from the previous step are applied to this case. In the next step, the velocity field is assumed to be stable and particles just follow this velocity field. Then the particles are released in this velocity field and simply follow their path. There are no interactions between the fluid flow and the particles or between the particles and other particles. This option is available but would drastially increase the computational time required. The only interaction used in this simulation is between walls, stirrer and particles, where particles bounce from the surfaces with a small amount of energy lost. Another problem that is noticeable in the plots is that the velocity fields in the first simulation differ between air and medium, but since we are only copying the velocity fields from the first case, this information gets lost. This leads to the fact that the vessel in the second simulation is filled only with water.For this reason, some particles leave the simulation at an early stage as they are deflected upwards out by the flow. This phenomenon is maximized if all particles are still at the bottom of the vessel at the beginning of the simulation. Not all particles are captured up by the flow at the beginning and some of the particles that do go with the flow shoot out the top. This would normally be minimized by starting the agitator slowly. This leaving of the particles also explains the lower numbers in the graph at 400 rpm, as more particles left the simulation due to a higher velocity that brought more particles to the top of the vessel.

All these solutions were not possible for the project because the error was found too close to the deadline. The solutions would require all simulations to be run again, which would take several weeks.

For this reason, the analysis of the result was done anyway to find patterns in the simulation that could be useful for future work in this area or on this topic.

There are two main reasons why this simulation was performed. First, is to validate this type of simulation using a simple experiment and second, to see if it is possible to see the particles in the scale down reactor at a later time. For both, a 120 frame per second camera was used to capture snapshots of the stirring process. This was not possible with the camera chosen, as can be seen in the Figure 35. As the image shows, the particles only form a gray mass with no discernible particle. This is mainly due to the small size of the active coal particles. The graph in Figure 35 also shows the total amount of particles in the simulation. While the size of the particles is not to scale since they are not visible from this distance, they still give an idea of the general location of the particles. Due to the simulation time the entire time limit was not tested and the simulation only reaches 21.4 seconds. It is possible that with more time, the overall distribution in the vessel could change, but with more particles adhering to the wall This would require enhanced testing after this thesis. Another experiment for validation was performed with larger particles in glycerol.

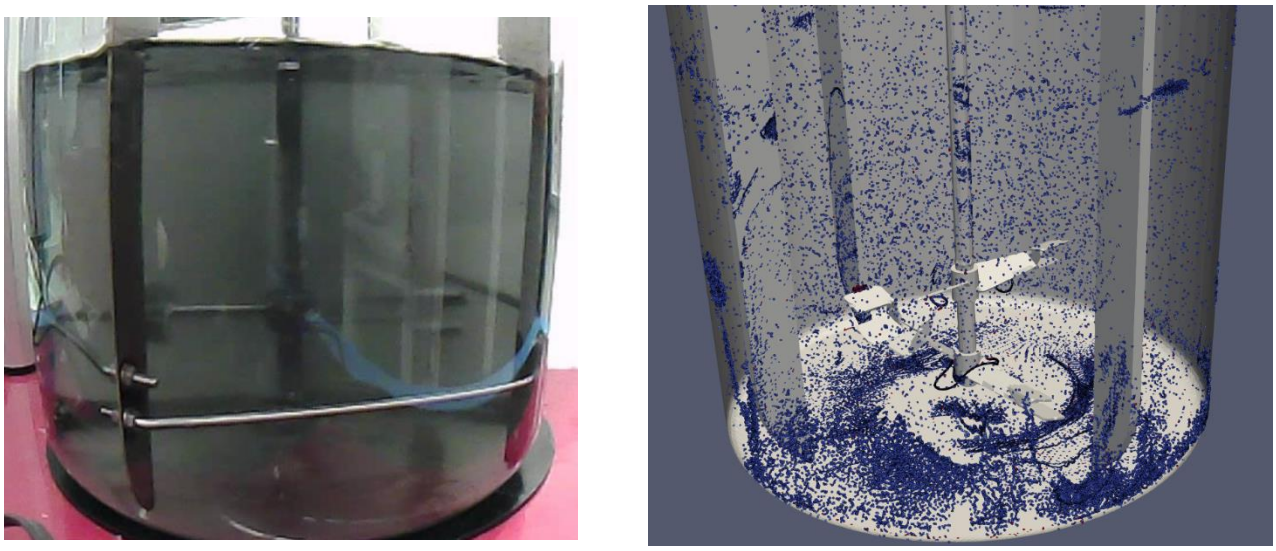


Figure 35 Left: Picture of the experiment run with 250 mg of particles at 400 rpm. Right: Diagram showing the particle distribution at 20 seconds into the simulation. Particles are scaled up to a visible range and do not represent the real size.

6.4 Simulation with glycerol and baffles

Since the last experiment did not lead to a good validation, the experiment was repeated with glycerol and PE-particles. These particles were larger than the catalyst used in the previous experiment. The

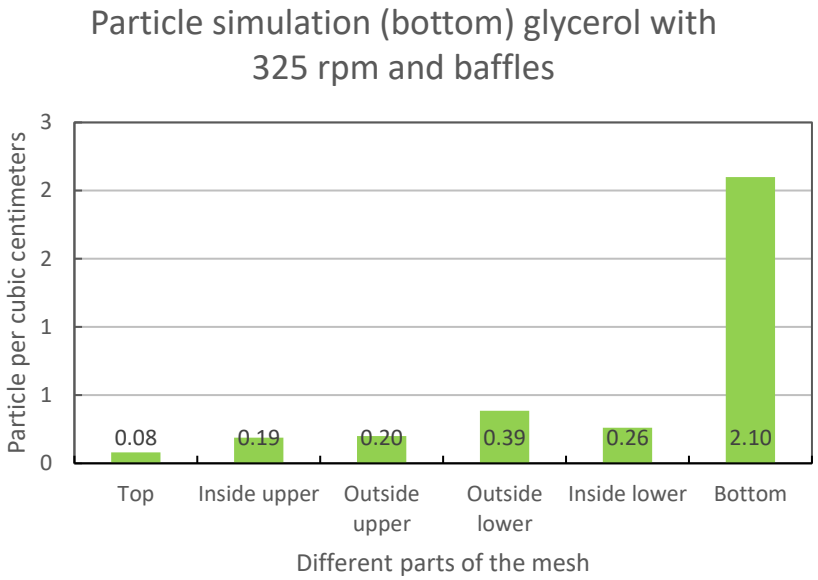


Figure 36 The graph shows the particle distribution at the steady state of the simulation. This is done with particle amount depending on volume in the chosen area. The unit used is particles per cubic centimetres of volume. The simulation was run with 325 rpm and baffles.

particles had an equivalent diameter of 2.83 mm.The number of particles was also reduced from the in chapter 6.2 used 80,000 to 2,000. Again, the simulation started with all particles at the bottom of the vessel, and the velocity was taken from the steady state of the previous liquid simulation. The simulation ran long enough to assume that a steady state has been reached. The geometry was like chapter 6.3 divided into 6 different parts and the average particles per volume were calculated. These numbers can be seen in Figure 36, which shows the amount of particles per cubic centimetres in different areas of the vessel. This graph shows what can also be seen in Figure 37, most of the particles stay on the bottom of the vessel and don't move up, probably due to falling on the bottom of the vessel where the velocity equals zero. Only about 25.9 % of the particles leave the bottom 6% of the vessel. Compared to the actual experiment, this does not fit at all. As can be seen in the picture in Figure 37, most of the particles leave the bottom of the vessel, and there is some distribution of particles inside the medium. As in chapter 6.3, the velocity field was taken from the fluid simulation, which was not very accurate, as can be seen ´from the size of the vortices. Another problem is that all particles start at the bottom of the simulation. To test this theory, another simulation was run with different starting positions.

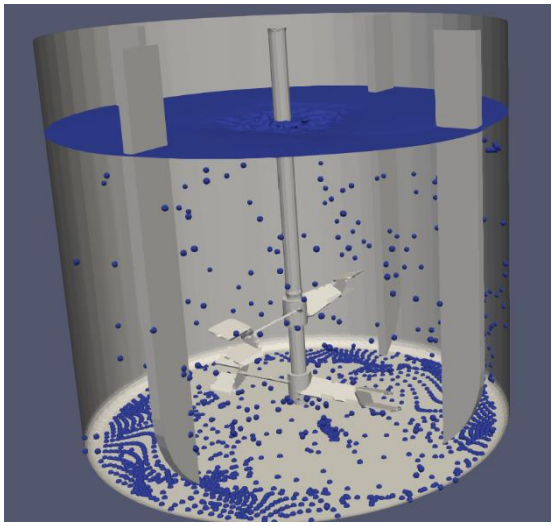
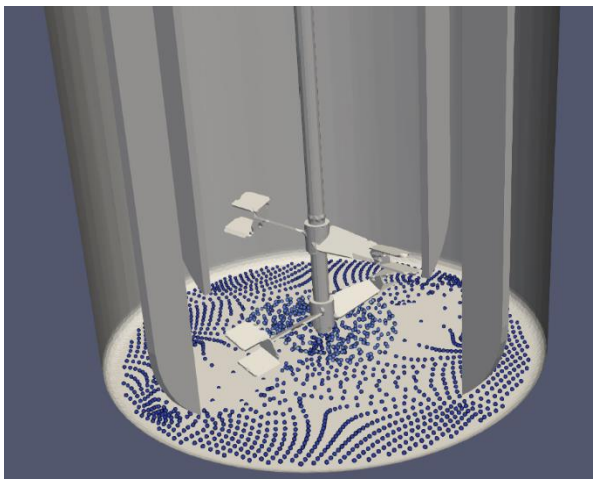


Figure 37 Left: Shows a diagram of the end of the simulation after 120 seconds. Simulation was run with 325 rpm. All particles started at the bottom of the reactor Right: Shows the experiment setup with glycerol and particles run with 325 rpm. Picture of the steady state.

While in the first simulation all particles started at the bottom of the vessel, they now started in the middle part on 2 sides of vessel, as it can be seen in Figure 38.



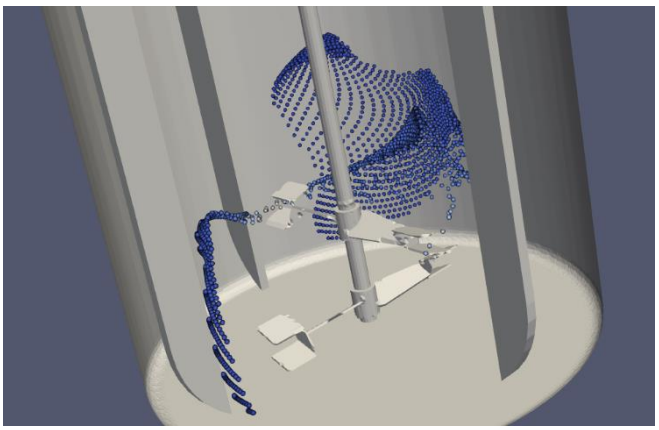


Figure 38 Right: 0.5 seconds of the simulation with particles being at the bottom at the beginning. Right: 0.5 seconds simulated with the particles starting in the middle of the vessel.

This led to a different result than in the first simulation. As Figure 39 shows, although there are still many particles in the bottom of the simulation, there are significantly fewer compared to the previous case. In this simulation, only 36 % of all particles are in the bottom 6 % instead of 74.1%. This is still not identical to the real experiment, but as you can see in Figure 40 below, it is closer to reality. The image on the left shows the particle distribution in the reactor after 75 seconds with the initial position at the bottom of the vessel. The image on the right shows the position of the particles at the same time, but with the initial position decribed above. This unsweaxoewa what Figure 16 already shows,

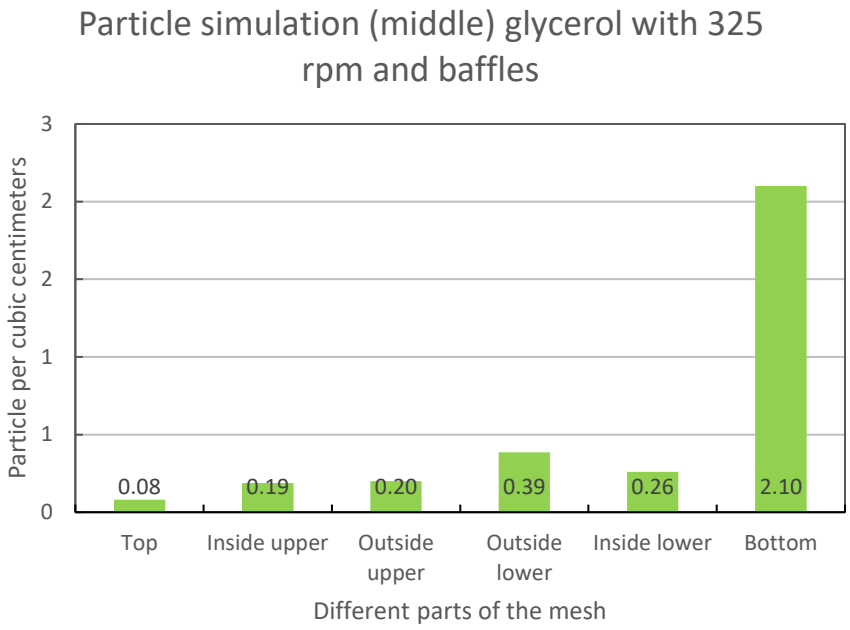


Figure 39 The graph shows the particle distribution at the steady state of the simulation. This is done with particle amount depending on volume in the chosen area. The unit used is particles per cubic centimetres of volume. The simulation was run with 325 rpm and baffles. The particles started in the middle part of the vessel.

that such a simple change in the starting position of the particles has a significant effect even over a period more than a minute. It could still be that over a longer period of time both states reach a similar distribution. Having done all the simulations to this point, it must be said that the simulation chosen here would have to be simulated at a higher quality to achieve an acceptable result. This mainly concerns the simulations in 6.1 and 6.2 since they could of use for this project.

All in all, for a high quality comparison it would be necessary to count the particles in each part of the experiment, but due to the quality of the camera and the viscosity of the glycerol, which made it difficult to focus the whole vessel with the camera, this was not possible (see figure 37). Visual comparison shows that the simulation is still not accurate enough to predict the experiment and would need to be further refined, which would mean investing more time in the simulation. This was not possible at this time and the mixing behaviour of the scale down reactor was simulated to show the principle for such work (see 6.5).

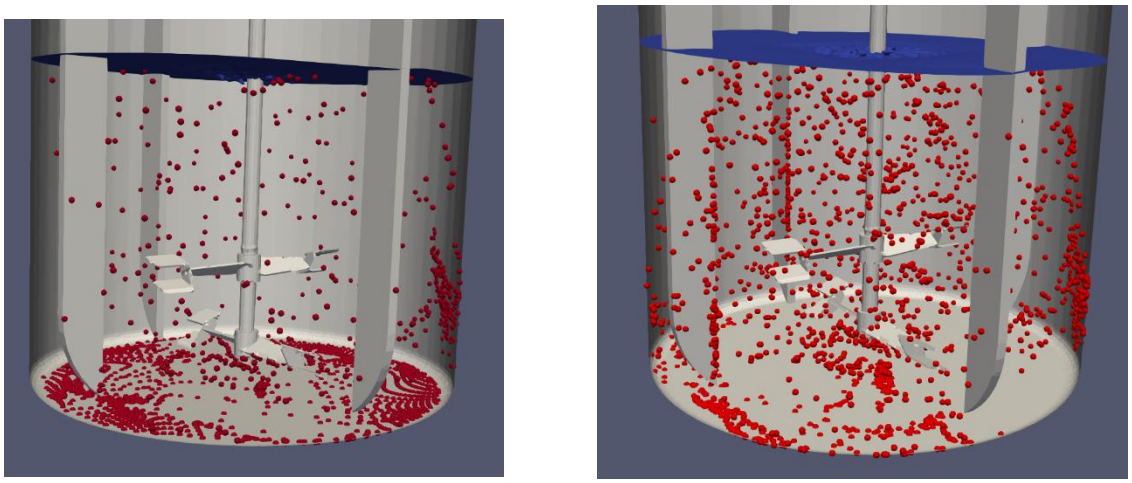
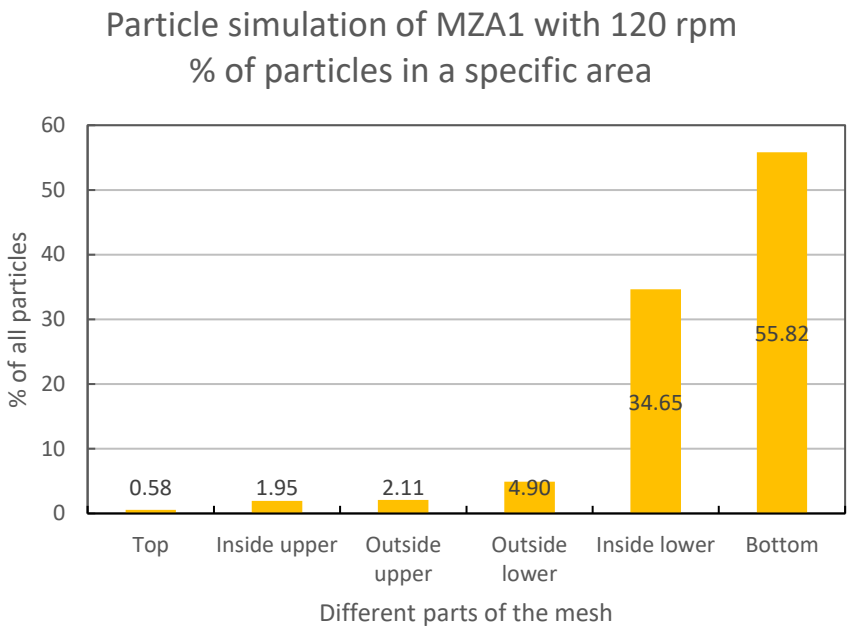


Figure 40 Left: particle distribution after 75 seconds with particles starting at the bottom of the vessel. Right: Particle distribution after 75 seconds with particles starting in the middle part of the vessel.

6.5 Simulation MZA1 and SDR1

In this part of the thesis, the results of the two reactors used for scale-down are presented. These reactors are the production reactor MZA1 used by Siegfried and the scale-down reactor SDR1, which was used in the laboratory to test the reaction. The goal of this part was to find a good relationship between the rotational speeds of the two reactors.

For this purpose, a simulation of the main reactor MZA1 was performed with the typically used rotation speed of 120 rpm. This was used as a basis for comparison. For the reactor SDR1 2 different speeds were chosen. The first speed was 400 rpm, which was used in previous experiments with this reactor. The second speed was calculated using the classical P/V rule for the scale up of reactors. This resulted in to a rotation speed of 571 rpm (see app. 10.6). Both rotational speeds were then simulated. To get a good comparison between all simulations, the catalyst particles from the first part were used again. Due to the difference in size between the reactors, a different number of particles were used for the simulation. The number of particles in the reactor is significantly lower than in the experiment. To simulate the actual number of particles around $5.4*10^{10}$ entries would be required in the simulation. This necessary computing power is not available for this project. Therefore, the number of particles was reduced to about $8.8*10^{4}$. Since the primary purpose of the results analysis is to look at the percentage distribution of particles in a given area of the reactor, this was considered acceptable. To get a good comparison between the two reactors, the same method of dividing the reactor into different parts was used again. In the previous simulation, the results were analysed based on particles per volume. This gives a good indication of the distribution between the different parts of the vessel, and comparisons between simulations in the same reactor are understandable. But with two different reactors, this method is not perfect. A better method of comparison is to plot the percentage of particles in an region compared to the total amount of particles. As can be seen from the graph in Figure 41, most of the particles (95.37 %) are in the bottom half of the reactor, although the image next to it seems to show a good distribution. 55.82 % of the particles did not even leave the bottom 6 % of the reactor. Most of these particles got stuck at the bottom of the reactor when they were pushed down by the lower stirring blades. The reason for the large number of particles in the middle of the reactor is mainly the agitator, into which the particles are pulled.



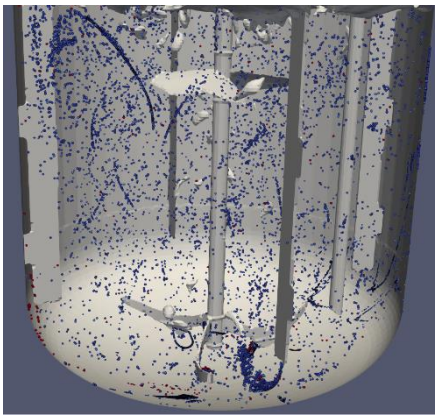


Figure 41 Left: Graph of the % of particles in the chosen areas for the simulation in the reactor MZA1 with 120 rpm after 12 seconds. Right: Diagram of the simulation in the on the left shown state. Particle size is scaled up and does not represent the real size. Particles at the bottom of the reactor rolled down to the middle of bottom, directly under the agitator.

Comparing this with the reactor SDR1 gives a different result. As Figure 42 shows, there is a better distribution between the different parts of the reactor. While there is still 63.25 % of all particles in the bottom half of the reactor, this is an acceptable result considering that the particles started at the bottom, and a few got stuck to the bottom or the agitator. The original plan for this phase of the project would have been, to now find the agitator rate that would best match the MZA1 reactor to the simulation. The results of the simulation with reactor MZA1 are inadequate, due to the nature of the simulation with a large vessel. Because of the volume of the vessel, the particle movement motion must be simulated for a longer time, but because of the velocity problem near the walls, the simulation is not representative of the actual reactor. Trying to match the reactor SDR1 with this simulation result would lead to useless result that would be worse in real practical use.

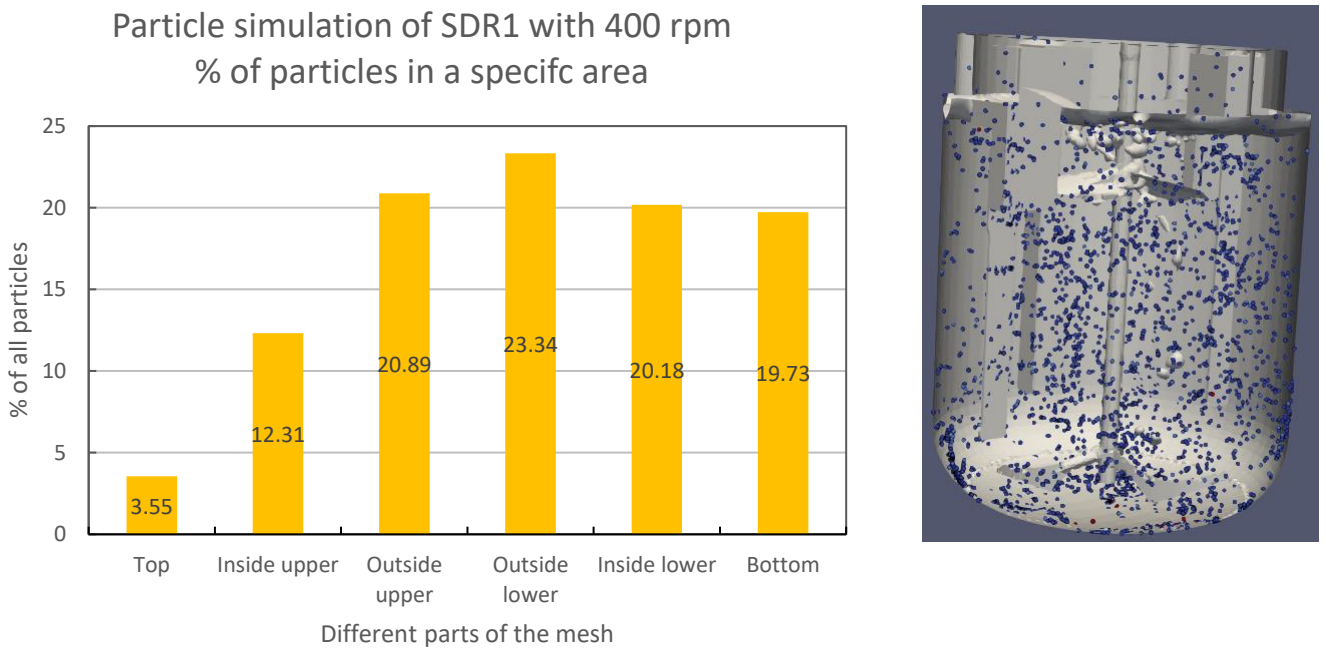


Figure 42 Left: Graph of the % of catalyst particles in the chosen areas for the simulation in the reactor SDR1 with 400 rpm after 3.5 seconds. Right: Diagram of the simulation in the on the left shown state. Particle size is scaled up and does not represent the real size.

For this reason, it makes sense to compare both reactor SDR1 simulations. Since they have the same

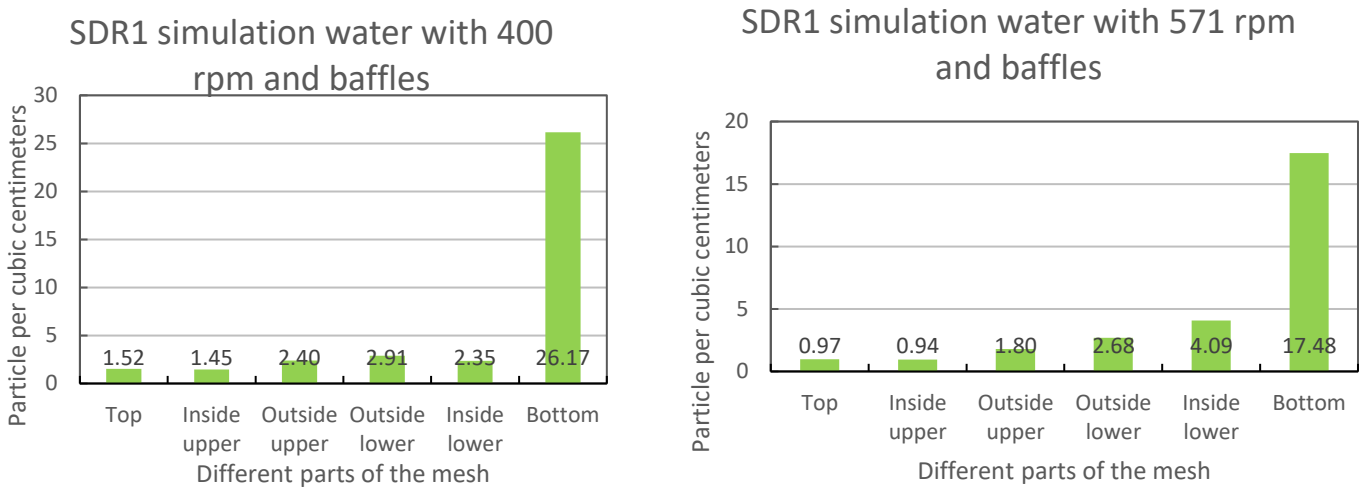


Figure 43 Both graphs show the particle distribution at the chosen state of the simulation. This is done with particle amount depending on volume in the chosen area. The unit used is particles per cubic centimetres of volume. Left: Particle distribution in reactor SDR1 and 400 rpm at 3.5 second. Right Particle distribution in reactor SDR1 and 571 rpm at 2.5 seconds

geometric structure, the more intuitive comparison method with particles per cubic centimetre can be used. Figure 43 shows both the 400 rpm and 571 rpm simulations. As in the previous parts, the amount of particles are calculated for each area of the reactor. While in the above graph the percentages appear to be evenly distributed, both new graphs show that the bottom has less volume than the other areas. Comparing this it to the rule stated in the indroduction, both meet the 90 % rule, meaning that more than 90 % of the volume of the reactor is filled with particles. The other rule was the 1-s rule.

This is clearly not satisfied because a large amount of particles remain at the bottom of the reactor. While the distribution is more uniform in the 400 rpm simulation, more particles remain at the bottom. In the 571 simulation, fewer particles are in the top 50 % of the reactor, but more particles leave the bottom 6 %.

6.6 Mean velocity

In chapter 6.5, it was explored if it is possible to find the agitator speed for the reactor SDR1. Since there was a problem with particles being stuck at walls and the particle amount hat to be reduced, this idea was deemed not feasible. A second attempt to find the agitator speed was done using the flow velocity of the fluid simulation. For this reason, 4 different simulations were carried out. The first simulation was done with the reactor MZA1 at 120 rpm. It was assumed that at after 10 seconds of simulation time there would a steady flow. For the next 10 seconds the mean velocity of every cell was calculated. The same was done with the 3 other simulations. These were with the reactor SDR1 and at 120, 400 and 571 rpm. 120 rpm as a direct conversion from the reactor MZA1, 400 as it was the standard agitator speed used so far in experiment with the reactor SDR1 and 571 because it was calculated from P/V. It was planned to also simulate 1471 rpm where both agitators have the same tip speed but the result wasn't available in time.

Aim of these simulations was it to find a connection between the mean velocity of the cells and with that find a possible answer for the agitator speed. Figure 44 shows the mean velocity of the reactor MZA1 between 10 and 20 seconds. The x-axis shows velocity intervals of 0.005 seconds. Every cell was sorted into one of these intervals. The amount of cells in each interval is depicted on the y-axis. There are two maxima in this graph at 0.2525 m/s and 0.4275 m/s. The middle of these 2 is at 0.34 m/s.

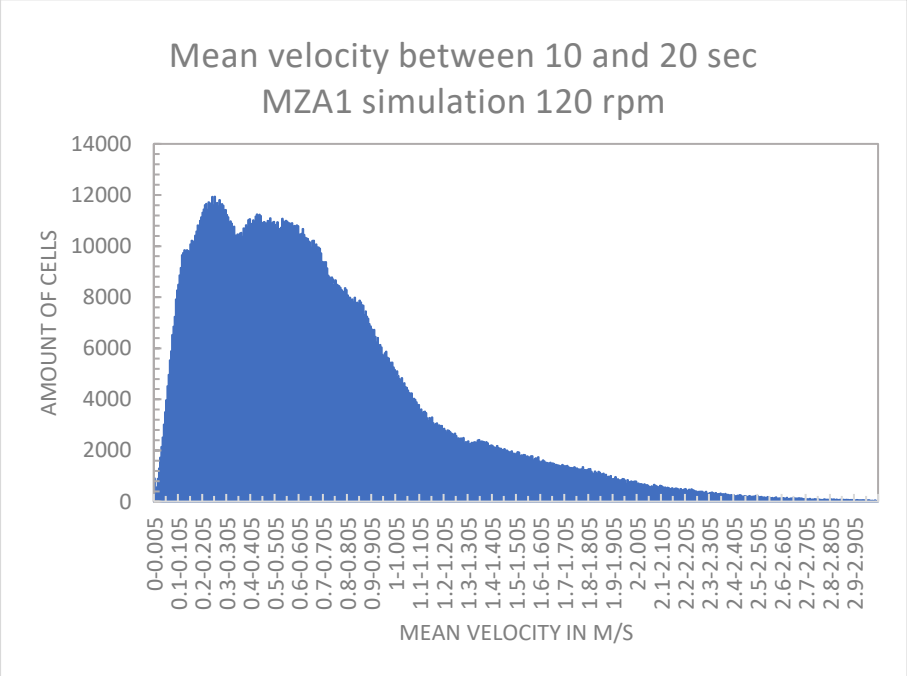


Figure 44 show the mean velocity between 10 and 20 seconds for the simulation MZA1 with 120 rpm. The y-axis is the amount of cells in the velocity bracket

Figure 45 shows the same information as the previous graph, but the simulation was done with the reactor SDR1. As expected, due to the size difference in agitators, the velocity is drastically different with the same agitator speed. Almost no cell has a velocity higher than 0.3 m/s and the maximum cells

have a velocity of 0.1 m/s. This simulation was done as a control so if large differences in result would be easily visible, since it was obvious that the agitator speed of 120 would be to slow for the reactor SDR1.

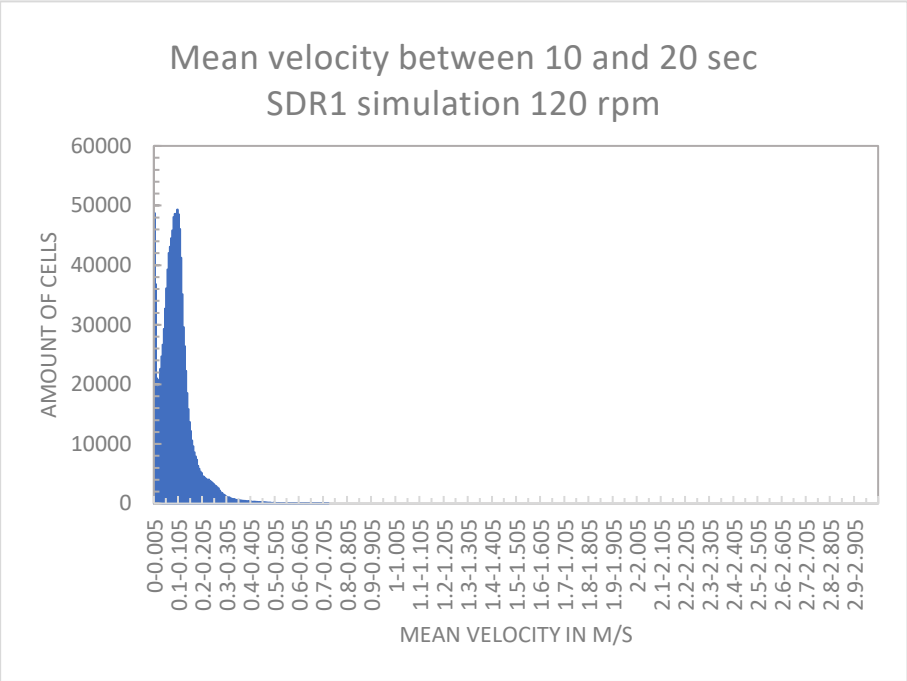


Figure 45 show the mean velocity between 10 and 20 seconds for the simulation SDR1 with 120 rpm. The y-axis is the amount of cells in the velocity bracket

The next simulation was done with 400 rpm and the reactor SDR1. Figure 46 shows the result of this simulation. Like the two graphs above, it shows the mean velocity of each cell between 10 and 20 seconds. To find an answer to the the agitator speed question, it is nesecarry to compare

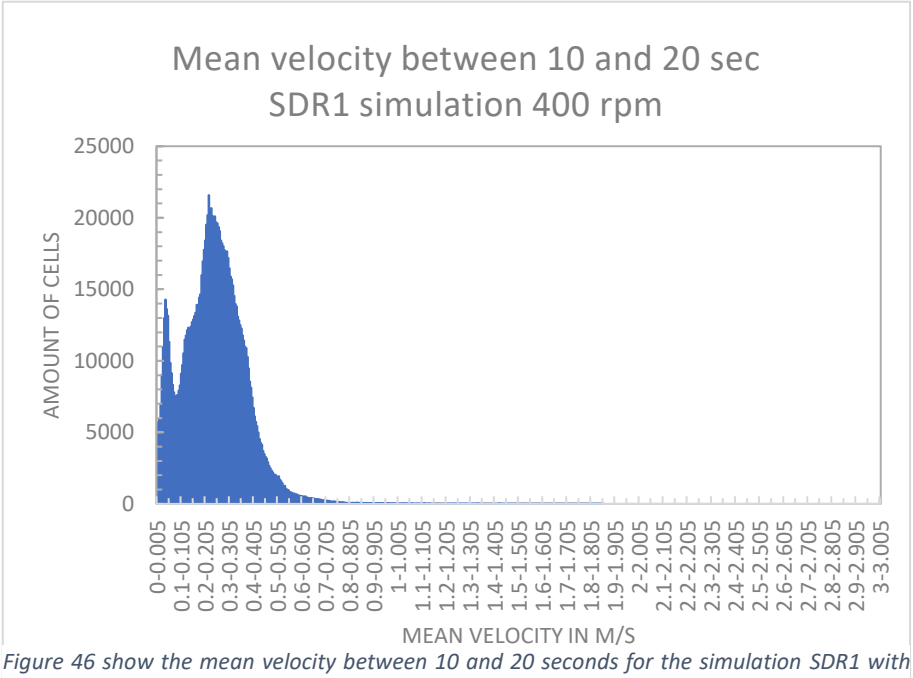


Figure 46 show the mean velocity between 10 and 20 seconds for the simulation SDR1 with 400 rpm. The y-axis is the amount of cells in the velocity bracket

characteristics of the distribution. While the figure 44 seems more like gauss distribution, it is not the case for this graph. There are two clearly visible peaks at 0.035 m/s and 0.225 m/s. The second peak is close to the maxima 0.25 of the MZA1 simulation. But unlike the simulation MZA1, there is a large drop of the velocity with almost no cells having a higher velocity than 0.7 m/s. The MZA1 simulation has a steady drop to around 2.5 m/s

The results of the last simulation are shown in Figure 47 . Like the previous 3 graphs, it depicts the mean velocity with the cell amount. The velocity with the maximum amount of cells was at 0.33 m/s. The graph still has 2 peaks but the right peak is close to a gauss form. The middle of the 2 maxima in the the MZA1 simulation was at 0.34 m/s. While the form is congruent between the 2 graphs and the gauss form is thinner in figure, it is the most comparable of the 3 simulations.

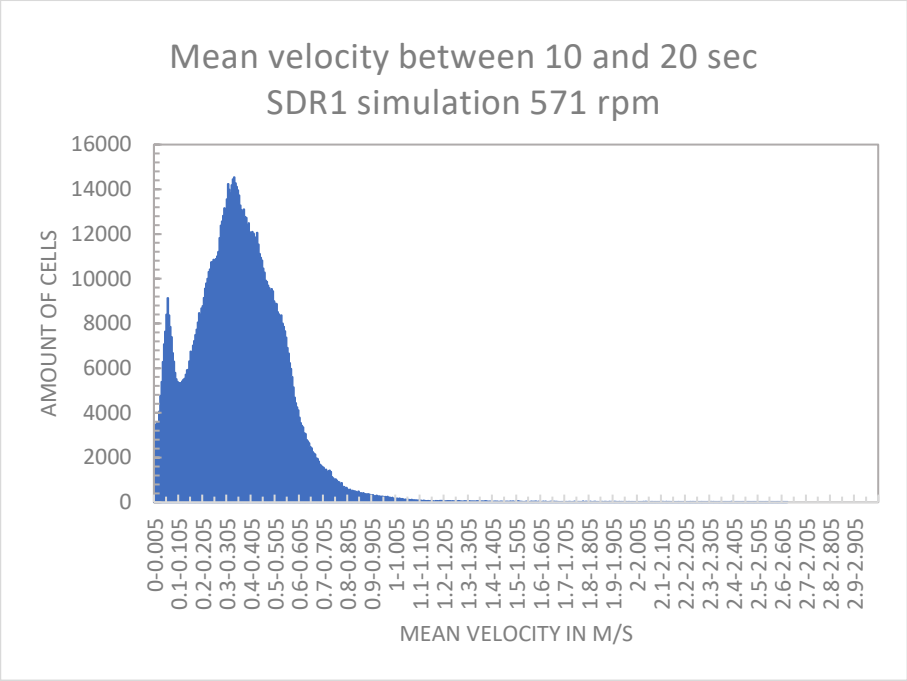


Figure 47 show the mean velocity between 10 and 20 seconds for the simulation SDR1 with 571 rpm. The y-axis is the amount of cells in the velocity bracket

There are a few reason for the difference in the graphs.

1. While the the structure of both reactors are similar, the number of baffles is not. In the construction of the baffles it was only possible to fit 3 inside the reactor SDR1. They were also attachted to a auxialliary construction which reached into the fluid. This means, while the flow is mostly the same, there is a difference. This is also confirmed by the first peak in each of the SDR1 simulations. Every simulation has a peak at low speeds with a drop off before the main gauss form. Since every simulation has this peak, it seems to be reactor dependend.
2. The agitator in the reactor MZA1 has a higher tip speed. This means, while the general velocity is the same between both reactors, the maximum mean velocity is higher in the reactor MZA1. This is shown by values going up to 2.5 m/s. This also explains the thinner gauss form in the simulations with the reactor SDR1

In the end it can be assumed that while the graphs are not 100 % matching, the simulation with 571 is the most similar to the simulation with the reactor SDR1. The biggest problems with having a conclusive statement is that the sample size is too small. While 571 rpm seems to be the most fitting

in these samples, it doesn't mean that this is the correct agitator speed. To confirm this more simulation sample, preferable with speed over 571, would be needed.

7 Conclusion

The main objective of this thesis was to find an optimal stirring speed for the reactor SDR that is equivalent to the agitator speed of 120 for the reactor MZA1. Up to this point it can safely be assumed that this target has an inconclusive result. With the conventional P/V method for scale down reactor the agitator speed of 571 was calculated. This agitator speed was tried to be verified in two ways. The first way was to use particle movement to compare the general flow of the fluids in the simulation. This was not feasible with the amount of particles and the problem with particles being stuck at wall. The second method was to look at the mean velocity in the steady state of the simulation. This lead to the conclusion that the most fitting agitator speed out of 3 choices, 120, 400 and 571 would be 571. This supports the agitator speed calculated at the start of this thesis. While it supports the the hypothetical agitator speed, it does not confirm it. The sample size of simulations is not big enough to confirm the result and would need more simulation to maybe give a definitive result.

Computational fluid dynamics is a useful tool that are error prone for beginner. A lot of time needs to be spent on corrections Most of the working time for this thesis went into the learning and correcting of OpenFoam simulations. Even so, unexpected errors can occur that can ruin weeks of simulation work.

Through the thesis it is apparent that the use of such a tool can be useful but must be considered on a case-by-case basis. For example, the vortex simulation at the beginning show, that while OpenFoam simulate the size of vortex, the difference between the result and the validation experiment shows the error of the simulation. But finding the error in the simulation is difficult since most of the error is only visible after the simulation is done and the results were analysed. Following that is the search for the mistake which can lead to one solution which after that also needs testing through another simulation. When a simple simulation in this thesis takes an average of 5 to 6 days, and error correction can be difficult. This also leads to the numbers of error discussed in the result part of this thesis. Mistakes like the 0 velocity are only noticed after two connecting simulations are done. These mistakes invalidate the whole simulation and sometimes are hard to fix. Another part of the poor result of simulation work, is the requirement of quality. Simulations are always a balance act between the quality and the time limit for calculation time. Estimating the time and quality required can be quite difficult for beginners in this field. This can clearly be seen on this thesis. While the results were generated in time, the quality of the simulation makes it almost unusable.

Even if the results are not quantitatively correct, they can still be analysed qualitatively. The simulation with water clearly shows the use of simulation work with the velocity field and in which direction the fluid is being moved. Even though the vortex size was not correct, it can be assumed that the general movement with and without baffles are like the real experiment. The difficulty with all simulations done in this thesis is that they are hard to validate. The result not being accurate has an advantage in this case because the difference is visible to the naked eye. But when the simulation come close to the reality a better validation method is needed. There were 3 different ideas in this thesis for simulation validation. The vortex size clearly worked and is easy to implement. But as we have seen with the glycerol where there was no vortex created, this is only possible with certain mediums. The next method was to use a camera to get a visual result that could be analysed. But with the small size of the catalyst and required quality for a clear picture at these speeds this was also difficult. This lead to the conclusion that particle simulations are not the right choice for this project. While simulations about the fluid flow can be used, it is better to find out parameter concerning the catalyst with conventional

methods. The last method mentioned was to try measuring the flow speed at certain points of the fluid. This was only intended in theory but would probably give the best result in baffle reactors.

8 Outlook

Altough this thesis has not given a conclusive result, it was not all for nothing. It gives a brief introduction to the subject of simulation work and is a good basis for someone who wants to continue this project or is working on a similar problem.

Simulation quality

The main problem with this work was not that it did not work, but that there was not enough time to try all the idea and work out as many bugs as possible. While the particle simulations are an interesting topic, they are based on the fluid simulation and until it gives a satisfactory result, it is hardly worth to improving. If someone tries to take up this project, the first step should be to improve the quality of the pure fluid simulation. Some things have been tried and have not resulted in a significant increase in quality. For example, one big factor that is always looked at is the overall mesh. This was identified early on and improved to a higher standard. This made the simulation run smoother, but the overall result did not change significantly. Instead of looking at the overall mesh, one should look at the surface mesh. To save time, a large portion of the stirring blades were simplified and given a coarse mesh. In particular, working on the mesh around the stirring blades could lead to a better result. This would be the simplest change to try. Another point that is necessary, is to solve the problem with the 0 velocity boundaries. There are two solutions discussed in the results analysis earlier in this thesis. The first and probably simpler solution, is to use the layer tool built into snappyHexMesh. This could help to form smaller cells around the surfaces and thus reduce the time that the particles have to move in a 0-velocty field. Since the size of this layer depends on the size of the solid material, this solution is not really recommended for later simulation with the catalyst. The diameter of the cell would have to be closer to the diameter of the particle, which would drastically increase the computation time for this catalyst. Instead, you could try the other method with the interpolation scheme called interpolationCellPointWallModified, which should prevent the velocity from reaching 0 entirely at the wall.

This last part is not important for simple fluid simulations but if particles are to be added, this problem cannot be ignored. Another way to to improve the quality of the simulation is to reduce the allowed courant number. In these simulations, the courant number of 0.7 was used, due to time constraints. While this is not bad, there seems to be a consensus in the literature that 0.3 is recommended for fine turbulent simulations. This should only increase the computation time by 1 or 2 days depending on the simulation duration but might give a better result. Since this has not been, the result cannot be guaranteed, but it is worth a try. If these simple solutions do not work, it may be necessary to change either the way the stirrer is moved or the way the turbulence is simulated. As shown in this thesis, the stationary power source may not be sufficient to simulate this reactor and a rotating mesh must be used to simulate the actual motion of the stirrer. The other idea would be to use a large eddy simulation instead of the Reynolds average simulation. Either solution would increase the quality of the simulation but would also require much more computational time. This would only be recommended if it is certain that the simplified version is not good enough.

RapidCFD

A major challenge in this work is running the simulations. The simulations have a significant runtime and lead to very time-consuming iterations. In order to achieve improvements in the results. It was always necessary to allow for a runtime of several days. Attempts to speed up the calculations by using

multiple CPU cores were successful to a small extent but failed beyond a certain number. The optimum for the simulation in this thesis seems to be in the range of up to eight cores. When more CPU cores are used, the speed does not increase further.

From this behaviour, it can be deduced that a reduction in computing time can only be achieved by increasing of the computing power of the individual CPU cores. Another way to make the simulation fast could be to use computational cores on the graphics card. The processors on these graphics cards are optimized to perform computational operations extremely fast. There is an offshoot of OpenFoam that has been optimized for computation by graphics cards. This project can be found on Github under the name RapidCFD. Unfortunately, it has not been extensively maintained in recent months. It is still based on an old OpenFoam version 2.3.1, so it is not possible to compute the current project directly. However, it would be interesting to adapt a simulation to the older OpenFoam format and test to what extent the use of a graphics card brings speed advantages. It should be noted that it would take some time to set up the necessary infrastructure for the calculation with RapidCFD. The drivers for the graphics cards have to be installed and the sources have to be compiled.

Future work with OpenFOAM

From this I also learned that it is important to start simulation work early. Setting up a simulation may not be the hardest part, but running the simulation is the most time consuming. Much of a project like this depends on the amount of simulation you can run, and that will ultimately be the limiting factor. It does take time to learn a new program and figure out all the little details, but it should be done as quickly as possible. The first simulations often contain errors that are fixed with each simulation cycle. This also means that there should be no downtime between simulations. Although it is helpful to run simulations early, you need to be able to compare them to real experiments if you are unsure of the quality of the simulations. To keep the simulation cycle going, it is imperative that, the real experiment can also be run and compared after the initial simulation. If this is not possible, the experiment must be run at the possible time. Subsequent validation may create time pressure that led to the poor results in this work.

If the results were acceptable, it would have been advisable to test these simulations on reactor SDR1 to see if they were valid. Since the ordered stirrer did not arrive in time this was not possible for this part of the project. Since the results were not that good, different approaches described above should be tested. If a reasonable result is obtained with the simple vessel and stirrer a test with the reactor SDR1 should be performed.

One final thing I would have done differently in this thesis is to become more familiar with the command and structure of OpenFoam. While I learned a lot by making mistakes and exploring them, mistakes could have been avoided. One such example is the size of the surrounding mesh before blockMesh. Although the basic commands were known, a simple error cell size resulted in led to a delay of almost a week to find the error.

While OpenFoam did not produce the preferred result in this project, it is a useful thing to learn and with a little refinement, can be useful in this and other projects.

9 Literature

[1] Matthias, K., & Zehner, P. (1995). Konzept zur Massstabsübertragung beim Suspendieren im Rührbehalter*. *Praxis*, *67*(3), 280–288.

[2] Geisler, R. K., Buurman, C., & Mersmann, A. B. (1993). Scale-up of the necessary power input in stirred vessels with suspensions. *The Chemical Engineering Journal*, *51*(1), 29–39. https://doi.org/10.1016/0300-9467(93)80005-9

[3] EKATO. (2000). *Handbuch der Rührtechnik: Grundlagen, Auslegung, Rührer und Rührsysteme, Mechanik, Konstruktion, Dichtungstechnik, Betriebssicherheit, Anwendungsgebiete*.

[4] Delacroix, B., Rastoeix, J., Fradette, L., Bertrand, F., & Blais, B. (2021). CFD-DEM simulations of solid-liquid flow in stirred tanks using a non-inertial frame of reference. *Chemical Engineering Science*, *230*, 116137. https://doi.org/10.1016/j.ces.2020.116137

[5] Schwarze, R. (2013). CFD-Modellierung: Grundlagen und Anwendungen bei Strömungsprozessen. In *CFD-Modellierung*. http://link.springer.com/10.1007/978-3-642-24378-3%0Ahttp://link.springer.com/10.1007/978-3-642-24378-3_1

[6] Stanishevsky A. V. (2001). Handbook of Surfaces and Interfaces of Materials. ISBN: 978-0-12-513910-6

[7] Zlokarnik M. (1971) Trombentiefe beim Rühren in unbewehrten Behältern. *Chemie Ingenieur Technik, 43(18),* 1028-1030. **https://doi.org/10.1002/cite.330431807**

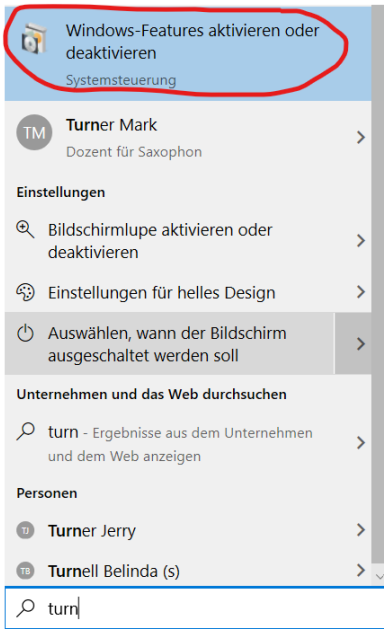
10 Apendix

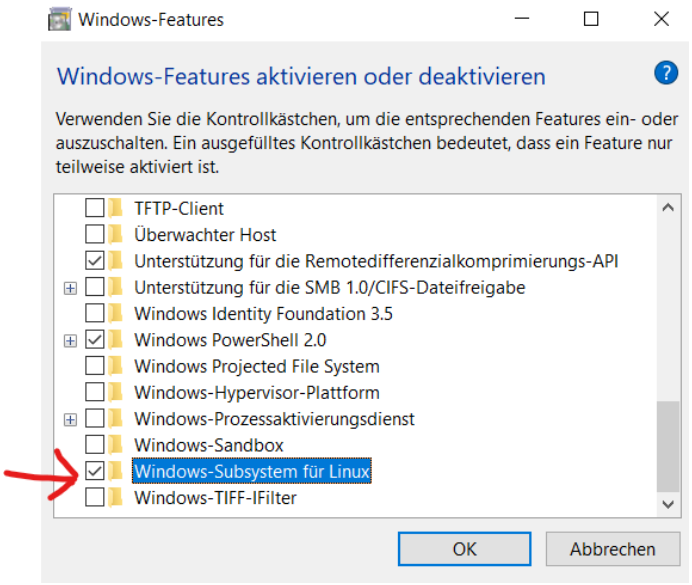
10.1 OpenFoam

All simulation done in this thesis where run on the OpenFoam v.2012. The version was installed on the Ubuntu Sub system v 18.04 Lite. All result analysis was done with the program v.5.9.1 of paraview. As text editor the program PSPad editor was used.

For people working first time with the program OpenFoam there is a short explanation how to install the program. The process of installing OpenFoam was done as followed:

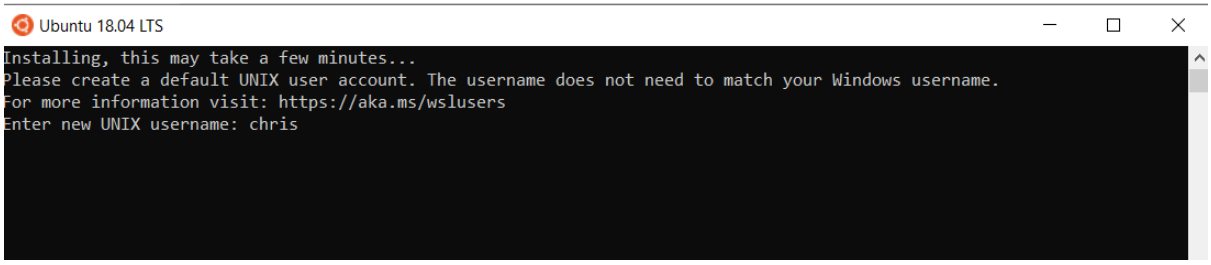
In a first step linux sub system were enabled to be installed. For this the option was checked in the widows feature menu.



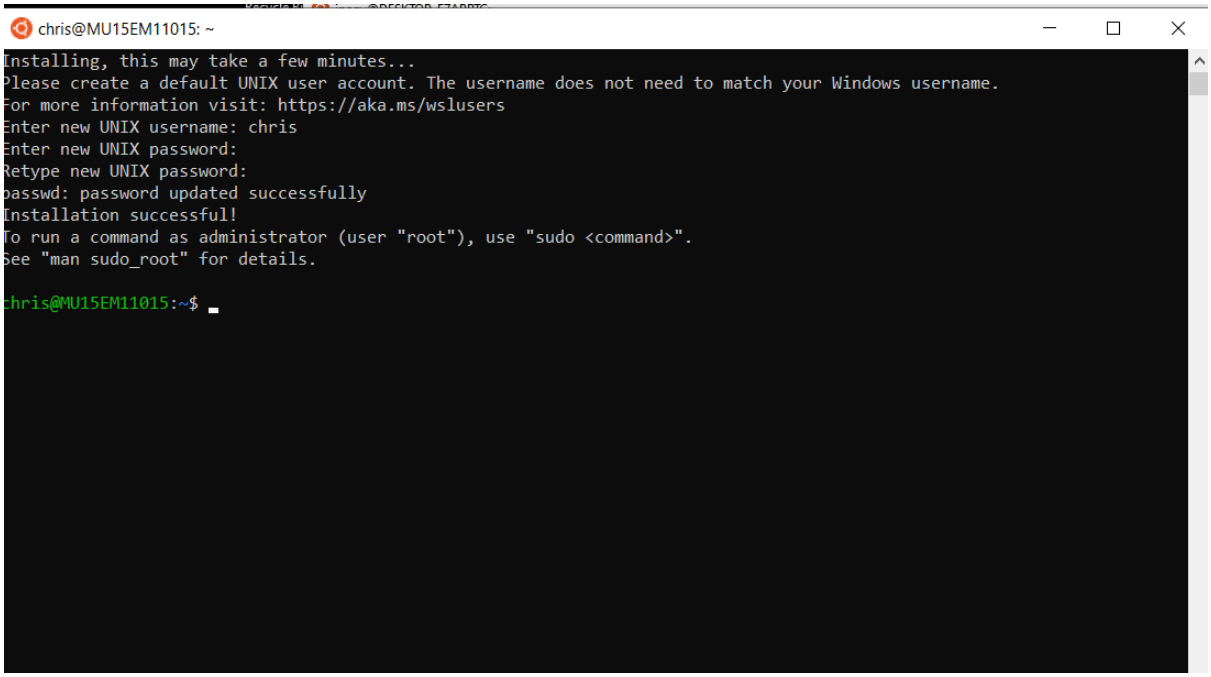


Following after that the ubuntu software was downloaded in the windows store.

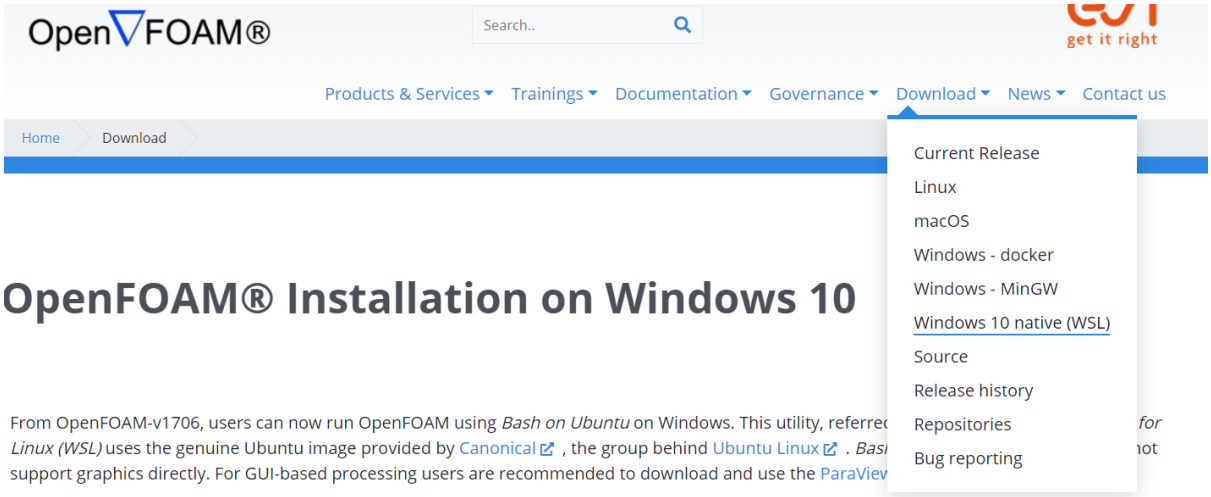
After the download the program was executed which led to the following command box.



A password had to be inserted. Important is in this case that no actual symbols will appear while typing but the password is still what one types. After that the following screen should appear.



With this step down, OpenFoam can be downloaded. There are a few version of OpenFoam available for download but for this method the correct choice is Windows 10 native (WLS).



The next step are all done in the ubunte command consol. In a first step the path needs to go the download folder. This is done with the command:

```
chris@MU15EM11015:~$ cp /mnt/c/Users/admin/Downloads/OpenFOAM-v2012-windows10.tgz .
```

The only thing that might need to be changed is the admin part and replaced with the account name of the local user. It is also important to not leave out the dot at the end of the command.

The program should now get decompressed with the following command and it might take some time.

```
chris@MU15EM11015:~$ tar xf OpenFOAM-v2012-windows10.tgz
```

In the next step the bashrc file needs to be changed. For that the file need to open with a text editor. This is done with nano.

```
chris@MU15EM11015:~$ nano .bashrc
```

With the arrow keys the file is navigate to the end and one line is written down.

```
source ~/OpenFOAM/OpenFOAM-v2012/etc/bashrc
```

This line is then saved with control + o and the file exited with control + x.

After restarting the ubuntu command box should be able to run. To test this out, the simple command blockMesh can be type into the line and executed. It should result in a response like this:

```
chris@MU15EM11015:~$ blockMesh
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2012                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : _7bdb509494-20201222 OPENFOAM=2012
Arch   : "LSB;label=32;scalar=64"
Exec   : blockMesh
Date   : Jun 08 2021
Time   : 16:59:43
```

To get the data calculated with OpenFoam the following command is useful for windows user

```
chris@MU15EM11015:~$ explorer.exe .
```

This opens a windows folder for the ubuntu filed and they are easier to access.

10.2 Difference in simulations

There were several different simulations run in this thesis. This part is trying to show in which way they were different.

While the simulations for the experiment were all done on the same geometry, a few parameters were changed from case to case. One of them was speed of the agitator. There were 3 different speeds chosen: 250, 325 and 400 rpm. To achieve this, the omega value in the MRFProperties file were changed.

For 250 :

```
    omega      constant -26.18;
```

325:

```
    omega      constant -34.034;
```

400

```
    omega      constant -41.888;
```

All these values are calculated after the formula omega = 2π / T while T is the time needed for one full rotation. The reason why they are all negative has to do with the rotation direction of the agitator. Since OpenFoam calculates the value from the bottom of the cylinder and the agitator is being moved counter-clockwise to have the correct flow movement.

Another parameter that changes over the simulation is the type of medium used. In this thesis both water and glycerol were used. This reflects in the simulation through the physical properties. In the file transportProperties are the both the density as well as the viscosity written down. Since air is also in the simulation it is also necessary to define the interaction on the surface of the medium. For water is would look like this:

```
water
{
    transportModel  Newtonian;
    nu              1e-06;
    rho             998.2;
}

sigma           0.07;
```

This would change for glycerol:

```
glycerol
{
    transportModel  Newtonian;
    nu              1.12e-03;
    rho             1261;
}

sigma           0.059;
```

The last big difference between the simulation is the use of baffles or not. For one, the stl file of the baffles must be included into triSurface folder. Other files that must be included, changed depending on the baffles are all starting parameter patch conditions, the snapping surfaces in snappyHexMesh and the surfaceFeatureExtract file.

10.3 Particle size catalyst

In the hydrogenation reaction uses a particle catalyst. To determine the size of the catalyst particles, multiple pictures with a scanning electron microscope were done. Since it was not possible to measure all particles with a computer a representative square was chosen, and particles were counted. The particles were then sorted into 3 size categorize to simplify it enough for a simulation.

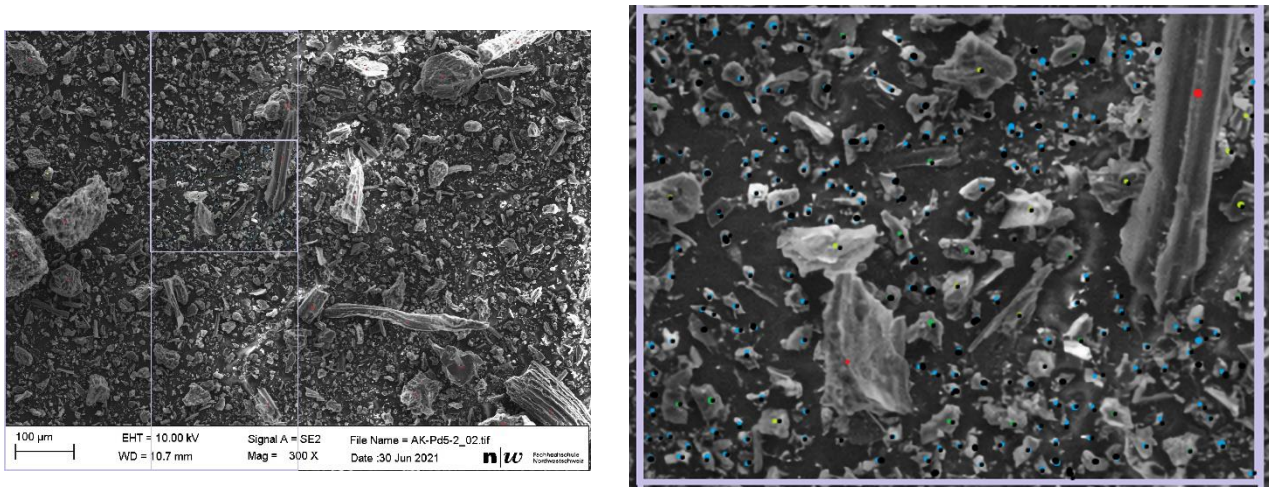


Figure 48 Left picture shows a part of the catalyst mass. The purple square is being counted. Right picture: shows a close up of the square with particles counted. Blue dots for particles witha size of 2 micrometer, green for 15 micrometer and red for 100 micrometer

Size in µ	Amount	Volume in m^3	Particles in 255 mg
100	2	5.236E-7	90970
15	29	1.767E-9	1319072
2	194	4.189E-12	8824138

Table 2: Calculation of the particle amount for the simulation with catalyst particles

10.4 Particle size PE

For the simulation with PE particles, both the volume and the density of the particles had to be known. For this reason, 10 particles were measured, and the average size was calculated. The particles had an average diameter of 0.001416 meter and a density of 1495 kg /m^3.

10.5 Calculating vortex size

In this thesis the vortex sizes get approximated with a conventional calculation. For this the formula used in Zlokarnik was applied.

$$h_T = 13.8 \times \left(\frac{n^2 \times d}{g}\right) \times \left(0.25 - \left(\frac{d^3 \times \rho^2 \times g}{\eta^2}\right)^{-0.10} \times \left(\frac{h_{L\ddot{U}}}{d}\right) \times d\right.$$

The formula calculates the depth of the vortex h_T, depending on the agitator speed n, the agitator diameter d, the gravitational force g, the density ρ, the kinematic viscosity η and the fluid level over the agitator blades $h_{L\ddot{U}}$. The validation experiment had the following parameter.

Density in kg/m^3	Viscosity	Agitator Diameter in m
998.2	0.001	0.112

The formula was used on all 3 agitator speeds.

Velocity in rpm	400	325	250
Vortex depth in cm	12.7	8.38	4.96

10.6 Scale up for agitator speed

P/V constant

The normal scale down of agitator speed is done with the formula P/V = constant. This includes the agitator power and reactor volume. The agitator power can be calculated with the following formula:

$$P = Ne \times \rho \times d^5 \times n^3$$

P stands for the agitator power, Ne for the newton number, d for the diameter of the agitator and n the rotation amount in one minute for the agitator. Since both agitators are the same only different in size it can be assumed that the Ne value is the same. For a stirrer of this form, it can be approximated to around 0.35. The two diameters are 0.54 m for reactor MZA1 and 0.0441 m for reactor SDR1. Since it was not known how full the reactor is normally filled, the volumes for the reactor MZA1 were assumed to be 3.606 m^3 and 0.0014m^3 for the reactor SDR1.

$P = 0.35 \times 997 \frac{kg}{m^3} \times 0.54^5\ m \times 120^3 = 27.69 \times 10^6\ W$ For the reactor MZA1.

With the conventional method that P/V = constant we get the following result.

$$P_{SDR1} = \frac{P_{MZA1}}{V_{MZA1}} \times V_{SDR1} = 10.77 \times 10^3 W$$

This then leads to the agitator speed of 571.

$$n = \sqrt[3]{\frac{P_{SDR1}}{Ne \times \rho \times d^5}} = 570.77$$

Tip speed constant

The tip speed of the agitator can also be constant in the scale down method. To calculate the tip speed the following formula is used.

$$U = \pi \times n \times d$$

In this formula U is the tip speed of the agitator, n the rotation amount and d the agitator diameter.

$$U_{MZA1} = \pi \times 120 \times 0.54\ m = 203.58\ \frac{m}{min}$$

$$n_{SDR1} = \frac{U_{MZA1}}{\pi \times 0.044\ m} = 1470.7$$

Both calculations keep one parameter constant and can be used for conventional scale up. The agitator speed for the two calculations is 571 and 1471, a difference of 900 rpm.

11 Electronic Apendix

- *3-D Model*
  - *SDR1*
  - *Experiment*
  - *MZA1*
- *Flash reports and labour journal*
- *Python scripts*
- *Simulations*
  - *Experiment Simulation*
    - *Glycerol*
      - *With baffles*
      - *Without baffles*
    - *Water*
      - *With baffles*
      - *Without baffles*
  - *MZA1 Simulation*
  - *SDR1 Simulation*