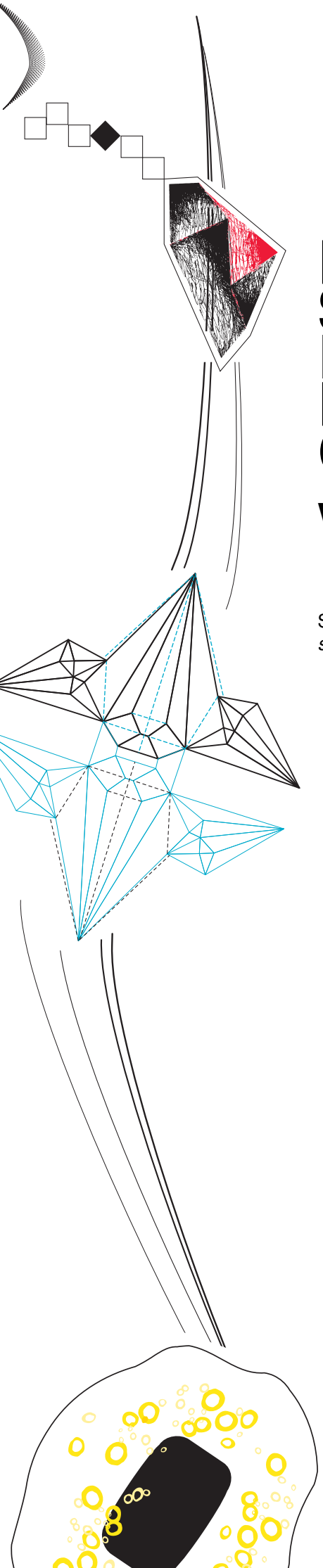
December 1, 2017



DIRECT NUMERICAL SIMULATION OF BUBBLE/WALL AND DROPLET/DROPLET COLLISION

WITH A MOTIVATION FOR LUBRICATION

Sietske van der Honing
supervised by Prof. Dr. Ir. Bernard Geurts

UNIVERSITY OF TWENTE.

Abstract

A Volume of fluid method is used to solve a one particle bubble flow, and a two droplet flow. For systems with large number of bubbles and/or droplets, direct numerical simulation is necessary, since there are no analytical solutions and empirical results are often muddled. To achieve good models for particle interaction it is best to start with simple cases. Two particle-interaction cases have been simulated and are presented in this paper. The bubble colliding with a wall and a head on droplet/droplet collision. These simulations have been done with a Volume of fluid multi marker method with a piecwise linear interface calculation scheme (VofMMRK), which has been implemented in OpenFOAM ®. The 2D scheme has been held up to a benchmark [15] and gave accurate results, with at least first order convergence and reasonable computation cost. The 3D case has been compared to mphBox [9] and shows good overlap. Parallelization has been applied and speedup is investigated, with simple domain decomposition computing time is nearly halved when using double the amount of processors, however this speedup stagnates when using 32 processors. This may be due to the solver, the decomposition or the (small) size of the problem. Some background and application information on Lubrication is provided, which could be used to examine new and old particle interaction models.

Contents

Chapter 1

Introduction

There are many different types of fluid problems, the two fluid flow, which is presented in this paper, is only one of them.Two fluid flow is a common occurence in industry, for example for steam production or in nuclear reactors [4], a more commonly recognised two fluid flow is rain. Like for most fluid problems the governing equations are given by a certain form of the Navier-Stokes equations. The equations used in this paper are not a simple form of the Navier-Stokes equations, and are therefor not analytically solvable.
Luckily more and more accurate and efficient schemes are made to solve these flows. We will use a volume of fluid method to solve a one particle bubble flow, and a two droplet flow. Extra challenges can be present in two fluid flow as well: turbulence, special boundary condition or just sheer size. This makes it computationally expensive to solve these flows accurately. On top of that empirical results for large bubbly flows are not accurate due to vision problems or influence of probes, making direct numerical simulation (DNS) not only convienent, but also necessary. It is important to find computationally cheaper ways to solve multi bubble flows with near contact accurately. The direction we chose to start this process is by looking into lubrication theory, we did a literature study into the history and scientific relevance of lubrication theory. Using this theory could lead to cost efficient schemes on coarser grids, while still staying accurate. However this theory is not accurate for the whole flow field, so models need to be found to implement this theory to practice. Applying this theory is left for future research.
In this paper we laid the groundwork of the afore mentioned solution by delivering a DNS which is accurate and introducing the steps which could be taken to decide whether the lubrication theory is applicable in this case. The cases laid out in this paper, Bubble/Wall and Droplet/Droplet, were chosen to compare with existing results as well as the relevancy to the near-contact question. Comparing results with already existing results is the only benchmark for problems for which no analytical solution or empirical result is available.
For the DNS we will use a Volume of Fluid (VOF) method. This method will be discussed in the first Chapter [2]. The platform in which it will be programmed, OpenFOAM®, will then shortly be discussed in Chapter [3]. We will compare the cases to the results of other papers, namely Hysing [15] and Cifani [9], in Chapter [4] and Chapter [5]. In Chapter [6] you can read about Lubrication theory and its possible application. The conclusions and recommendations for future research can be found in the last Chapter [7]. To improve reading of the equations a nomenclature is included in appendix [D].

Chapter 2

The Model

2.1 Mathematical Formulation

In this paper we will assume a viscous, incompressible, isothermal, immiscible two-fluid flow. This will be described by the Navier-Stokes equations. Isothermal means that the temperature will not influence the flow, making the energy equation superfluous. The flow is thus solely based on the continuity and momentum equations. The immiscibility, non mixability, of the two fluids will be modeled by a source term, acting locally around the interfaces between the two fluids. This source term ensures that we can solve the equations over the entire domain. The governing equations can be found in equations 2.1 and 2.2.

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot (2\mu \boldsymbol{D}) + \rho \boldsymbol{g} + \boldsymbol{S}, \tag{2.1}$$

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2.2}$$

Here $\boldsymbol{u}$ is the velocity field, t the time, ρ the density, p the static pressure, μ the dynamic viscosity, and $\boldsymbol{g}$ is the gravity. The deformation tensor, $\boldsymbol{D}$, is given by $\boldsymbol{D}_{i,j} = \frac{\partial_i u_j + \partial_j u_i}{2}$. Lastly $\boldsymbol{S}$ is the local source term for the interaction between the two phases. This interfacial tension force is given by $\boldsymbol{S} = \sigma \kappa \boldsymbol{n} \delta(n)$, where σ is the surface tension and κ the curvature of the surface area, $\boldsymbol{n}$ is the local unit normal to the interface and $\delta(n)$ the dirac delta function on the surface area.
Since we want to solve the equations over the entire domain, we need to define a density and dynamic viscosity, which account for the particles in the fluid. These are respectively defined by equations 2.3 and 2.4.

$$\rho = \alpha \rho_p + (1 - \alpha)\rho_f. \tag{2.3}$$

$$\mu = \alpha \mu_p + (1 - \alpha)\mu_f. \tag{2.4}$$

Here α is the volume fraction field, the subscript p is used for the particle values, whereas the subscript f stands for the surrounding fluid values. The formulation now slightly changes from the method mentioned in Cifani [8], since we will now use a multiple marker approach [20]. This means that each particle has its own volume fraction field (α_i), with $0 \leq \alpha_i \leq 1$. This is to prevent numerical coalescence. The volume fraction field shows the location of particles, inside the particle the field equals 1, outside it equals 0. The volume fraction field used in equations 2.3 and 2.4 is now defined as $\alpha = \max_i(\alpha_i)$. To follow the spatial distribution inside the flow a transport equation for each of the volume fraction fields is solved, which can be found in equation 2.5.

$$\frac{\partial \alpha_i}{\partial t} + \boldsymbol{u} \cdot \nabla \alpha_i = 0 \qquad \forall\, i \in \mathbb{N} \tag{2.5}$$

2.2 Numerical Algorithm

We will look into the discretization of equations 2.1, 2.2 and 2.5, where we will spend some extra attention on the source term $\boldsymbol{S}$ from equation 2.1. We have five unknowns in these equations, namely $\boldsymbol{u} = (u\ v\ w), p$ and α. From this unknowns the velocity field and the pressure are coupled by equation 2.1, leading to four unknowns for three equations. We will use the continuity equation 2.2 as the fourth equation, however this does not contain the pressure. We will use a special coupling technique, in which we apply the divergence operator onto the momentum equation 2.1 [13].
We will first apply a semi-discretization on the momentum equation 2.1 [16]. For this semi-discretization we only discretized the time derivative, while keeping the spatial derivative in partial differential forms, so we can use the mass conservation to eliminate terms and end up with the Poisson equation for

the pressure [13]. After we semi-discretize the equations we will use the PISO, Pressure-Implicit-Split-Operator, of OpenFOAM®, which is used for transient calculations, but is limited in time step by the Courant number [13]. The Courant number, also called Courant-Friedrichs-Lewy condition, is a necessary condition for convergence. Keeping to this condition makes sure information does not reach second or third neighbor cells, which are not taken into account in the explicit time stepping scheme. The semi-discretized equation can be found in equations 2.6 - 2.9.

$$a_c \boldsymbol{u}_c = H(\boldsymbol{u}) - \nabla \hat{p} + \boldsymbol{F}, \tag{2.6}$$

$$\text{with} \qquad H(\boldsymbol{u}) = \sum_{nb} a_{nb} \boldsymbol{u}_{nb}^{n+1} + \frac{1}{\partial t}[2(\rho \boldsymbol{u})_c^n - \frac{1}{2}(\rho \boldsymbol{u}_c^{n-1})], \tag{2.7}$$

$$\hat{p} = p - \rho \boldsymbol{g} \cdot \boldsymbol{x}, \tag{2.8}$$

$$\text{and} \qquad \boldsymbol{F} = \boldsymbol{S}^{n+1} - \boldsymbol{g} \cdot \boldsymbol{x} \nabla \rho. \tag{2.9}$$

This form is derived from a second order backward differencing scheme in time [8]. In these equation the subscript c is the center of cell, whereas the subscript nb means all the nearest neighboring cells. The coefficients a_p and a_{nb} result from the discretization of the time derivative, the diffusion term and the convective term for all the neigbouring cells. A full description on how to find these coefficients can be found in [25]. The term F from equations 2.6 and 2.9 contains the discretization of the remainder of the gravity term, after we used the modified pressure in equation 2.6, given by equation 2.8.
From the semi descrete form of the equation 2.6 you can attain the discrete velocity field at the cell faces (subscript cf) by using interpolation and a face-centred gradient operator, this can be seen in equation 2.10.

$$\boldsymbol{u}_{cf} = \left(\frac{H(\boldsymbol{u})}{a_c} \right)_{cf} - (\frac{1}{a_c})_{cf} [\nabla_{cf} \hat{p} - (F)_{cf}]. \tag{2.10}$$

After applying the continuity constraint 2.2 to equation 2.10 we get the well known Poisson equation for the pressure, equation 2.11.

$$\nabla \cdot \left[(\frac{1}{a_c})_{cf} \nabla_{cf} \hat{p} \right] = \nabla \cdot \left(\frac{H(\boldsymbol{u})}{a_c} \right)_{cf} + \nabla \cdot \left[(\frac{1}{a_c})_{cf} (F)_{cf} \right]. \tag{2.11}$$

The PISO algorithm can be summed up as follows:

1. Set the boundary conditions related to the case.
2. Compute volume fraction fields. (Equation 2.13)
3. Compute interfacial force. (Equations 2.14)
4. Compute mean density and mean viscosity. (Equations 2.3 and 2.4)
5. Solve the discretized momentum equation to compute an intermediate velocity field. (Equation 2.6)
6. Compute the mass fluxes at the cell faces.
7. Solve the pressure equation. (Equation 2.11)
8. Correct the mass fluxes at the cell faces.
9. Correct the velocities on the basis of the new pressure field.
10. Update the boundary conditions.
11. Repeat from 6 for the prescribed number of times.
12. Increase the time step and repeat from 1.

For step 2 we will look at the transport equation for the interface (Equation 2.5). This is done by a time marching line advection, which is done sequentially along each coordinate with the algorithm proposed by Puckett [23]. The interface in a grid cell can be defined as in equation 2.12.

$$\boldsymbol{n} \times \boldsymbol{x} = q. \tag{2.12}$$

Here $\boldsymbol{n}$ is the normal vector to the interface, x the position vector and q a parameter related to the intersection points of the interface with the coordinate axes. Details about the relation between the

volume fraction (α) and the parameter q can be found in the appendix of [8] and in Scardovelli [26]. This reconstruction method is programmed in OpenFoam and will give a sharp representation of the interface. We will then separately advance the volume fraction field of each particle in time by an operator split method, sequentially along each coordinate direction. The scheme, proposed by Puckett [23], for the 3D algorithm can be found in equation 2.13.

$$\begin{aligned}
\alpha_{i,x} &= \alpha_i^n - \Delta t \frac{\partial(u\alpha_i^n)}{\partial x} + \Delta t \alpha_{i,x} \frac{\partial u}{\partial x}, \\
\alpha_{i,y} &= \alpha_{i,x} - \Delta t \frac{\partial(v\alpha_{i,x})}{\partial y} + \Delta t \alpha_{i,y} \frac{\partial v}{\partial y}, \\
\alpha_{i,z} &= \alpha_{i,y} - \Delta t \frac{\partial(w\alpha_{i,y})}{\partial z} + \Delta t \alpha_{i,z} \frac{\partial w}{\partial z}, \\
\alpha_i^{n+1} &= \alpha_{i,z} - \Delta t (\alpha_{i,x} \frac{\partial u}{\partial x} + \alpha_{i,y} \frac{\partial v}{\partial y} + \alpha_{i,z} \frac{\partial w}{\partial z}).
\end{aligned} \tag{2.13}$$

The last term in each of the intermediate equations of 2.13 is to correct for over- or undershoots, which could occur after a single direction sweep. However this term vanishes in the total change, provided that the velocity is divergence free, equation 2.2. The order of the direction sweeps are alternated to reduce assymetries, where all permutations of direction sweeps are used.
For step 3, the interfacial force, we will use the continuum-surface-force (CSF) approach of Brackbill [6], with an extension to a two marker approach, as done by Kwakkel [20]. We will use an interface value smoothened over multiple cells, for the computation of the curvature. This technique is based on the convolution of the volume fraction field α with a kernel Φ, to get to the smoothened field $\tilde{\alpha}$. The details of this technique can be found in [8] and [12].
The interfacial force in equation 2.9 is then given by equation 2.14.

$$(\boldsymbol{S}^{n+1})_{cf} = \sum_i \sigma_i \kappa_{cf,i} \nabla_{cf} \tilde{\alpha}_i^{n+1} \tag{2.14}$$

$$\text{with} \qquad \kappa_{cf,i} = \nabla \cdot \tilde{\boldsymbol{n}} = \nabla \cdot \frac{\nabla \tilde{\alpha}_i^{n+1}}{|\nabla \tilde{\alpha}_i^{n+1}|} \tag{2.15}$$

Where σ_i is the surface tension of the ith particle, $\kappa_{cf,i}$ the curvature at the cell face, and $\tilde{\boldsymbol{n}}$ is the approximation to the normal.
In step 6 and 8 we calculated the mass fluxes at the cell faces. This means we calculate the outflow and inflow of mass at every face of a cell. When we combine this outflow and inflow, a mass conservation should hold, so no mass loss or mass gain. If this is not the case we have to make a change in the flow field and thus the pressure. By iterating over this we will eventually reach a threshold value.
Note: in step 4 we will use the smoothened fraction field, $\tilde{\alpha} = \max_i \tilde{\alpha}_i$. And numerically bound this value between 0 and 1, by setting values higher than 1 to 1 and likewise for lower than 0.

Chapter 3

OpenFoam

The method (VofMMRK) explained in the previous chapter has been implemented in OpenFOAM® [22]. OpenFOAM ® stands for Open Source Field Operation And Manipulation, it is a programme used for a lot of Computational Fluid Dynamics (CFD) problems. This programme has an object based C++ library, with already exsting classes and its own notation. It provides a number of solvers, designed to solve specific problems, and utilities, performing data manipulation tasks. You can think of mesh building or decomposing as a utility. On top of the existing solvers and utilities you can add your own or modify existing ones to fit your purpose.
To get to know more about OpenFOAM® you can check the User Guide, available at:
https://cfd.direct/openfoam/user-guide/

3.1 The Case

The OpenFOAM® case file is set up like any other case in OpenFOAM®. An overview of this can be found in figure 3.1. A sample of the files are attached in the appendix for clarity.

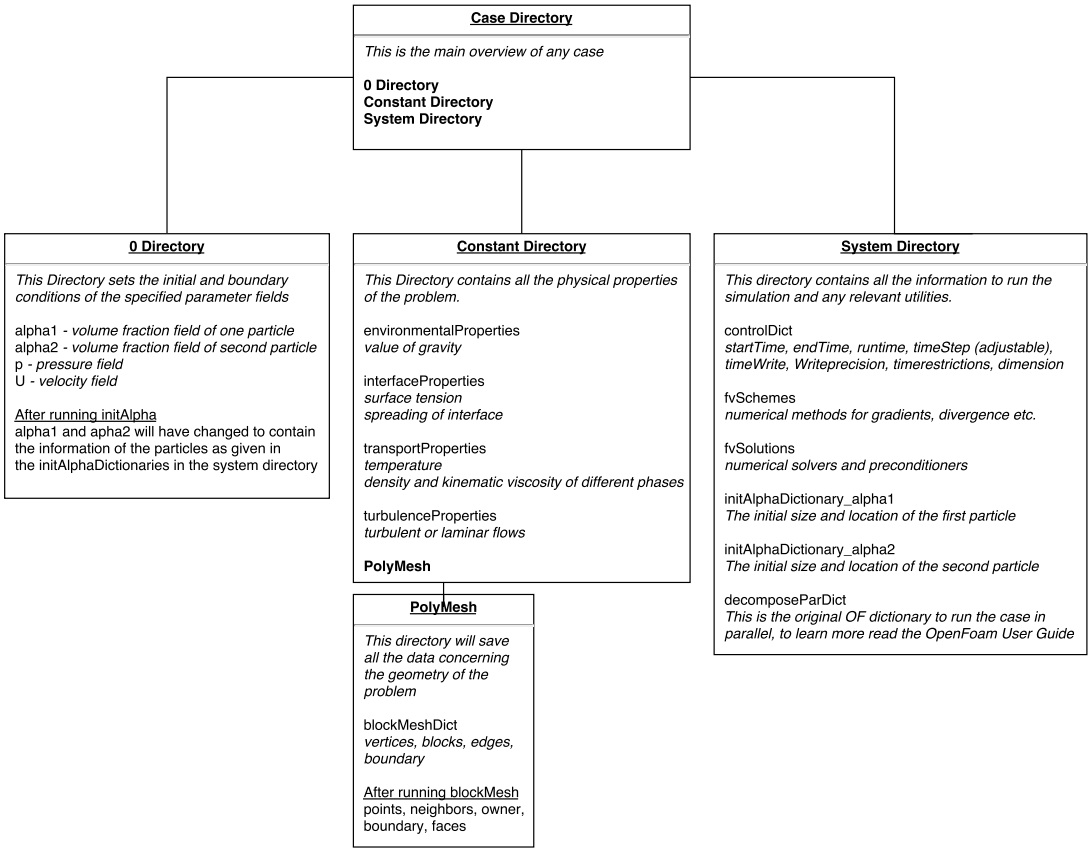


Figure 3.1: Flowchart of the structure of an OpenFOAM® case for VofMMRK. In this figure the directories are bold, the files are plain, and the explanations are cursive.

Chapter 4

2D DNS

In this chapter we will use the VofMMRK solver described in Chapter [2]. We will simulate three different cases, examine the spatial convergence and compare this to results of other methods.

4.1 Benchmark

A good way to validate a new numerical method for a complex problem is using a benchmark. Since there is no analytical solution to compare your results with, it would be good if different methods obtain similar results. Therefor we will simulate the first testcase of the benchmark of Hysing [15]. In this benchmark they apply both visual comparison of the results, as well as quantitative comparison. We will only work on the first test case, a bubble undergoing moderate shape deformation, and not on the second more challenging test case. This is because the results in the paper for this second testcase were rather inconclusive on a number of points, among which the point of break up and the final form of the bubble [15]. The used parameter values can be found in table 4.1.
The testcase we will be working on has a Reynolds number of 35 and an Eötvös number of 10 with density and viscosity ratios equal to 10. The Reynolds number is a dimensionless quantity, a high Reynolds number shows turbulent flows, whereas a low Reynolds number flow is a laminar flow. The Reynolds number is here defined as $Re = \frac{\rho_f U_g L}{\mu_f}$. The Eötvös number is defined as $Eo = \frac{\rho_f U_g^2 L}{\sigma}$, and gives the ratio of gravitational forces to surface tension effects [15]. For these definitions we used $U_g = \sqrt{g 2 r_0}$, the gravitational velocity.
Experimental studies on this type of bubble have been done by Clift et al. [10] and the bubble should end up in the ellipsoidal regime, however we will be doing a 2D simulation. The visual comparison of the result will show that in the 2D case it will also end up in the ellipsoidal regime. We will compare the results on circularity, the center of mass, and the mean rise velocity of the bubble.

	symbol	value
Density of the fluid	ρ_f	1000
Density of the bubble	ρ_b	100
Viscosity of the fluid	μ_f	10
Viscosity of the bubble	μ_b	1
Surface tension of the bubble	σ	24.5
Gravity	$\boldsymbol{g}$	$(0\ -0.98\ 0)$
Initial location of the bubble	(x, y)	$(0.5, 0.5)$
Initial bubble radius	r_0	0.25
Dimensions of the box	X, Y	$[0, 1], [0, 2]$
Gridsize	$(n_x$ x $n_y)$	40x80, 80x160, 160x320, 320x640
Maximum Courant number	C	0.45
Maximum Interface Courant number	C_I	0.45

Table 4.1: Physical quantities of the first test case of the Hysing Benchmark [15].

We will calculate the norms and rate of convergence as in the paper [15], for ease repeated in equations 4.1 - 4.7.

$$\text{Center of Mass} \qquad \boldsymbol{X}_c = (x_c, y_c) = \frac{\int_{\Omega_b} \boldsymbol{x} dx}{\int_{\Omega_b} dx}. \tag{4.1}$$

$$\text{Circularity} \qquad \gamma = \frac{2\pi r}{P_b}. \tag{4.2}$$

$$\text{Rise velocity} \qquad \boldsymbol{U}_c = \frac{\int_{\Omega_b} \boldsymbol{u} dx}{\int_{\Omega_b} dx}. \tag{4.3}$$

$$l_1 \text{ error} \qquad ||e||_1 = \frac{\sum_{t=1}^{NTS} |q_{t,ref} - q_t|}{\sum_{t=1}^{NTS} |q_{t,ref}|}. \tag{4.4}$$

$$l_2 \text{ error} \qquad ||e||_2 = \left(\frac{\sum_{t=1}^{NTS} |q_{t,ref} - q_t|^2}{\sum_{t=1}^{NTS} |q_{t,ref}|^2} \right)^{1/2}. \tag{4.5}$$

$$l_\infty \text{ error} \qquad ||e||_\infty = \frac{\max_t |q_{t,ref} - q_t|}{\max_t |q_{t,ref}|}. \tag{4.6}$$

$$\text{Rate of Convergence} \qquad ROC = \frac{\log_{10}(||e^{l-1}||/||e^l||)}{\log_{10}(h^{l-1}/h^l)}. \tag{4.7}$$

Here the subscript c means the center of an object, the subscript b means it is from the bubble. P stands for perimeter and q for the different quantities, namely Center of Mass, Circularity and Rise velocity. NTS is the number of time steps, and ref is the reference solution, the solution from the finest grid, with $\frac{1}{h} = 320$. Lastly the supscript l is the grid refinement level. We were supposed to calculate these values at every time step, we however only saved the data of certain moments in time, this may influence our results slightly.
The statistics from the simulation can be found in table 4.2 and figure 4.2. Were we can see that the computation time is scaling quadratically with $\frac{1}{h}$, so linearly with the number of grid cells.
We made the figure of the bubble shapes for all the grid sizes at time $t = 3$ to visually compare with the

1/h	NTS	CPU in s
40	174	6.01
80	285	29.17
160	512	174.85
320	997	872.39

Table 4.2: Statistics from the simulation of Hysing testcase1, where NTS is number of time steps.

benchmark as well. We can see that the results are in agreement with the benchmark, since they are converging to the black line. This can be found in figure 4.1. We can also clearly see that the bubble did end up in the ellipsoidal regime as expected.
For the error norms and the rate of convergence we attained the values found in table 4.3.

$1/h$	$\|\|e\|\|_1$	ROC_1	$\|\|e\|\|_2$	ROC_2	$\|\|e\|\|_\infty$	ROC_∞
Center of mass						
40	1.09E-01		1.10E-01		1.30E-01	
80	5.19E-02	1.07	5.23E-02	1.07	5.94E-02	1.13
160	1.82E-02	1.51	1.84E-02	1.50	2.06E-02	1.53
Rise velocity						
40	7.89E-02		8.75E-02		9.05E-02	
80	3.85E-02	1.03	3.89E-02	1.17	3.82E-02	1.24
160	1.11E-02	1.80	1.14E-02	1.77	1.35E-02	1.50
Circularity						
40	9.78E-03		1.22E-02		2.28E-02	
80	1.62E-03	2.60	1.94E-03	2.65	3.43E-03	2.73
160	4.66E-04	1.79	6.60E-04	1.56	1.29E-03	1.42

Table 4.3: Relative error norms and convergence rates for testcase 1 of the Hysing Benchmark [15].

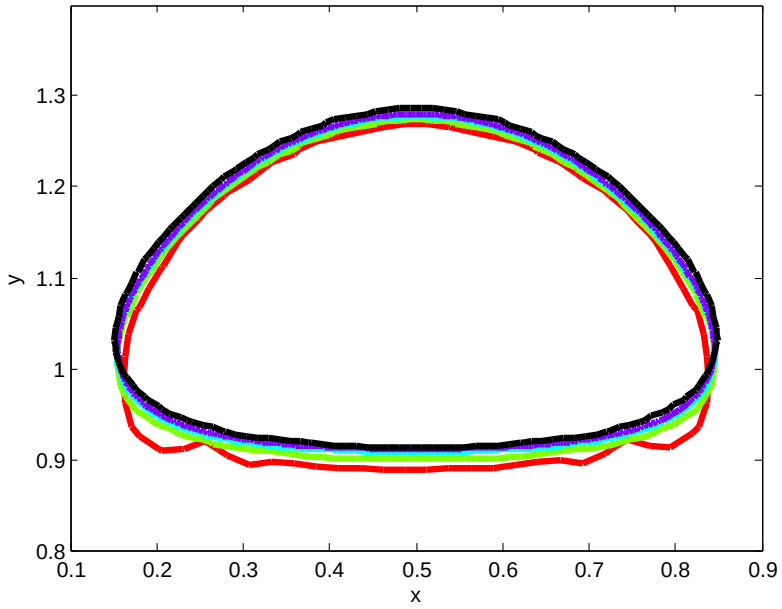


Figure 4.1: The contour, $c = 0,5$, of the bubble at time $t = 3$ for testcase 1 of the Hysing Benchmark [15]. Here red is for gridsize 40 x 80, green for 80x160, blue for 160x320 and purple for 320x640. The result in black is the result of mphBox [9], which coincides with the reference result of the Hysing Benchmark [15].

Comparison to Benchmark

When looking at the results a couple of things come to mind. First of all our error norms seem higher than that of any of the other codes. And the order of convergence of the VofMMRK scheme does not seem to reach second order, which seems also true for the FreeLIFE group, but the TP2D comes close to second order and the MooNMD even goes to forth order when looking at the Center of Mass. For this simulation we used the adjustable time step from OpenFOAM ®, which keeps the Courant number automatically below a certain value yet takes the largest time step possible. This might have caused the scheme to be less than second order, due to a temporal error. Therefor we also did this simulation with a stationary time step, which can be found in Appendix A.
When looking at statistics we see a comparable result of CPU per time step with the TP2D group, but a much smaller number of time steps taken. This could lead to a temporal error and may explain the discrepancy between the results from the VofMMRK code compared to the others. The FreeLIFE group seems to take a lot more computation time, which makes finer resolutions more expensive. When looking at the figure we see that the CPU is scaling with second order with respect to gridsize. This is as expected, since we double the amount of cells in two directions, thus squaring the amount of work. The number of time steps seems to influence it a little as well, eventough it is scaling close to linearly with respect to the grid size.
All in all this method seems to get quite good results with relative low computation cost and has the possibility to improve results with even higher resolution.

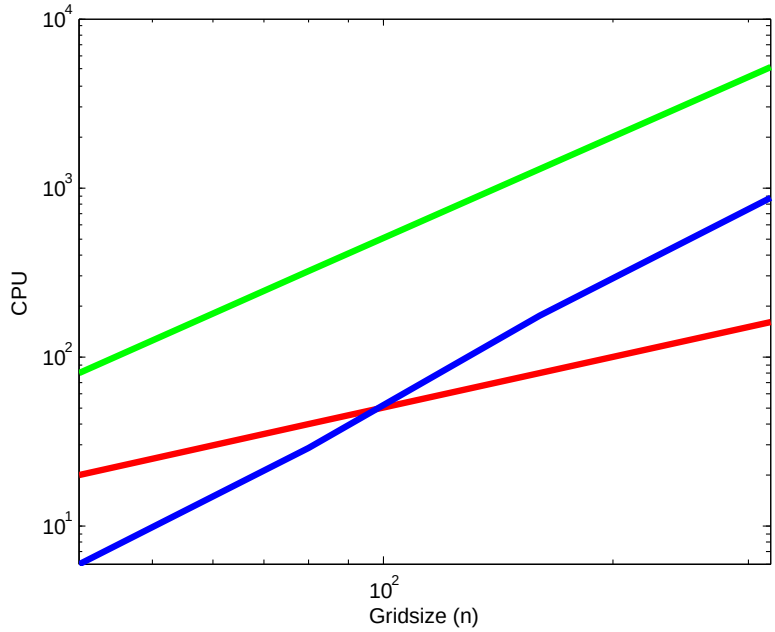


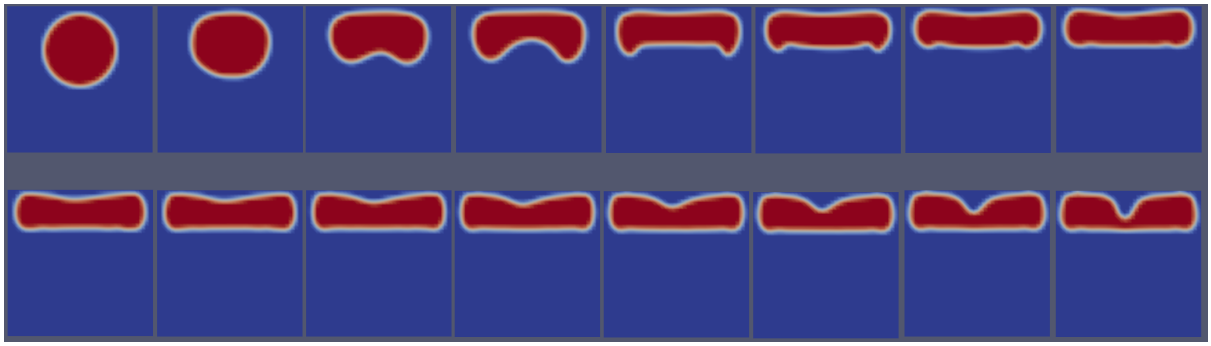
Figure 4.2: The CPU time in a loglog plot against the gridsize. The CPU time is given in blue, first order is given in red and second order is given in green.

4.2 Bubble/Wall

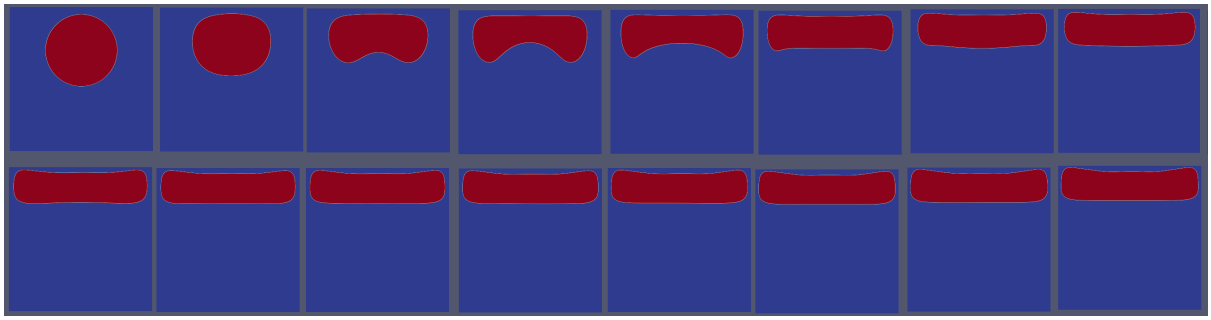
We did a direct numerical simulation of this case to see how the solver performed with more challenging parameters. The values can be found in table 4.4. We also wanted to see the modeling of near contact with the wall. To see how the simulation dealt with the boundary condintions, and if convergence could once again be obtained. Lastly this simulation is a good reference when looking at near-contact situations. It could be investigated if the contact near a wall is similar to a head on collision, and could be used for simulating collisions on coarser grids, by using a reflection method. Lastly we can see if a lubrication correction is needed for contact with a wall as well.

	symbol	value
Density of the fluid	ρ_f	1000
Density of the bubble	ρ_b	1
Viscosity of the fluid	μ_f	10
Viscosity of the bubble	μ_b	0.1
Surface tension of the bubble	σ	1.96
Gravity	$\boldsymbol{g}$	$(0\ -0.98\ 0)$
Initial location of the bubble	(x, y)	$(0.5, 0.7)$
Initial bubble radius	r_0	0.25
Dimensions of the box	X, Y	$[0, 1], [0, 1]$
Gridsize	$(n_x$ x $n_y)$	40x40, 80x80, 160x160, 320x320, 640x640
Maximum Courant number	C	0.45
Maximum Interface Courant number	C_I	0.45

Table 4.4: Physical quantities for the Bubble/Wall case.



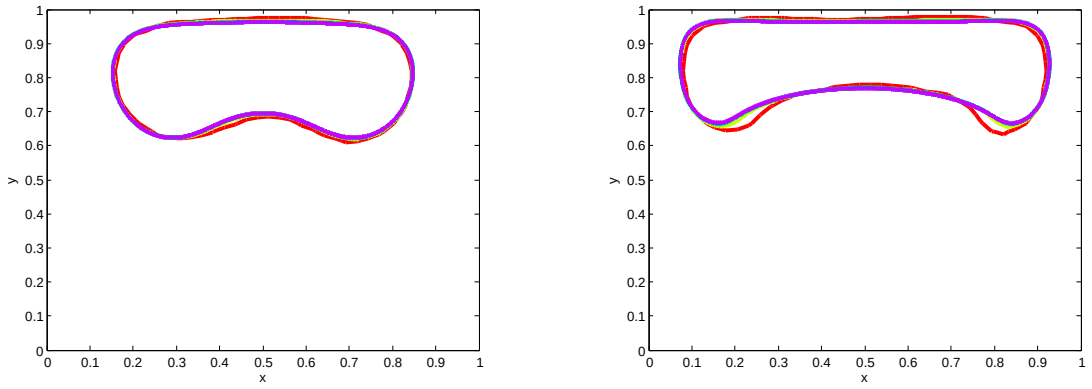
(a) gridsize is 40x40



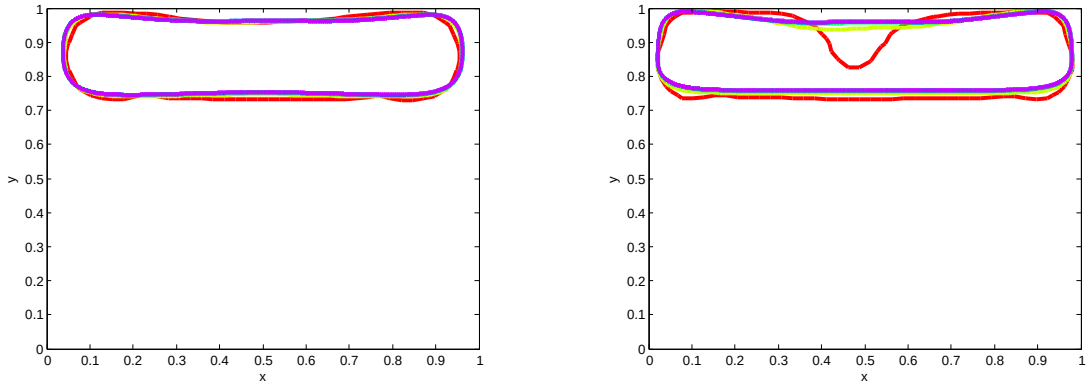
(b) gridsize is 640x640

Figure 4.3: The volume fraction field, α, for the different grids at different time steps, starting at $0s$ with intervals of $0.5s$ till $8s$. Here the inside of the bubble is red, while the outer fluid is blue.

From all the relevant data, such as the volume fraction field (α), the pressure (p), and the velocity (U), we decided to mainly focus our attention on the volume fraction field. In figure 4.3 we can see the time evolution of this simulation at the coarsest and the finest grid. There is a big difference between these two, which shows that a convergence study is relevant. For example at time $t = 4$, top right situation, we see some extra attributes in the coarsest grid, and at time $t = 8$ a whole extra gap appears. We will look into these times and two earlier times, namely $t = 1$ and $t = 2$, and make a contour plot of the bubbles on all the grid sizes, this can be found in figure 4.4.



(a) $t = 1$ (b) $t = 2$



(c) $t = 4$ (d) $t = 8$

Figure 4.4: The contour,$c = 0.5$, of the bubble on the different grids at different time steps. Grid 40x40 is in red, 80x80 is in yellow, 160x160 is in green, 320x320 is in blue, and 640x640 is in purple.

We also looked at mass conservation of this method, the results can be found in figure 4.5. The formulation of the relative area difference is found in equation 4.8.

$$\text{relative area distance} = \frac{\sum_{n_x,n_y} \alpha(t=0) - \sum_{n_x,n_y} \alpha(t)}{\sum_{n_x,n_y} \alpha(t=0)}. \tag{4.8}$$

From the figure we can conclude that there is a mass loss, since the volume of the bubble reduces over time, while the density stays constant. We can see that this mass loss takes place when there is a lot of movement, whereas it slows when the bubble is barely moving. The error is of the order $O(10^-5)$ and is not dominant.

We will use the same quantities, error, and convergence rate calculations as in the previous section, equations 4.1 - 4.7. The results for this simulation can be found in table 4.8.

Conclusion

We looked at this case to see how the method would be dealing with more challenging parameters. It seems that we get accurate results and that the computation time per time step has gone down since the previous simulation. We can either assume that the method is capable of more challenging parameters or scrutinize the set-up a bit more. It seems that some parameters are indeed more challenging, but the critical velocity is probably smaller, leading to smaller Reynolds numbers. Nevertheless we once again see convergence and still have an interesting case for the near contact question. It is not possible to answer if it is similar to a head-on collision, since we don't have that type of data here. The boundary conditions do influence the solution largely, we now see a bigger difference between the grids, especially at $t = 8$. This is not only for the coarsest grid, but also for the grid with $\frac{1}{h} = 80$. This shows that lubrication correction might also be necessary for the wall collision case, where we see the thin film geometry. Lastly we conclude that the mass loss of this method was not dominant for this case, since the order ($O(10^-5)$) was smaller than the other errors.

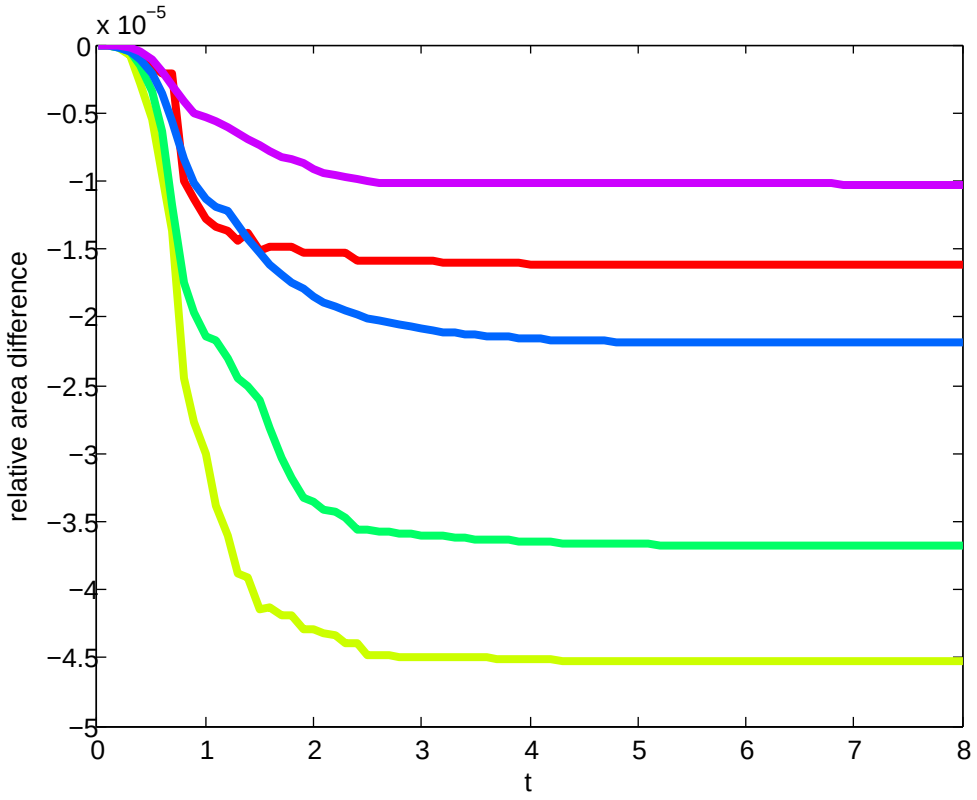


Figure 4.5: The relative area error compared to the numerical area at $t = 0$, which shows some mass loss over time. Here grid 40 x 40 is in red, 80x80 is in yellow, 160x160 is in green, 320x320 is in blue, and 640x640 is in purple.

n_x	$\|\|e\|\|_1$	ROC_1	$\|\|e\|\|_2$	ROC_2	$\|\|e\|\|_\infty$	ROC_∞
Center of mass						
40	1.62E-01		1.65E-01		1.95E-01	
80	7.17E-02	1.18	7.22E-02	1.19	8.66E-02	1.17
160	3.03E-02	1.24	3.05E-02	1.24	3.49E-02	1.31
320	9.48E-03	1.67	9.67E-03	1.66	1.36E-02	1.36
Rise velocity						
40	3.05E-01		2.63E-01		4.18E-01	
80	1.77E-01	0.78	1.46E-01	0.85	1.40E-01	1.57
160	1.02E-01	0.80	7.39E-02	0.98	6.72E-02	1.06
320	4.77E-02	1.09	2.91E-02	1.35	2.47E-02	1.45
Circularity						
40	1.60E-02		1.92E-02		3.55E-02	
80	5.37E-03	1.57	6.37E-03	1.59	9.10E-03	1.96
160	1.71E-03	1.65	1.94E-03	1.71	2.69E-03	1.76
320	4.34E-04	1.98	4.72E-04	2.04	6.36E-04	2.08

Table 4.5: Error norms and convergence rates for the bubble/wall collision.

n_x	NTS	CPU in s
40	429	11.37
80	479	53.64
160	596	304.55
320	10005	2814.94
640	2316	30234.51

Table 4.6: Statistics from the simulation of the case of the bubble/wall collision.

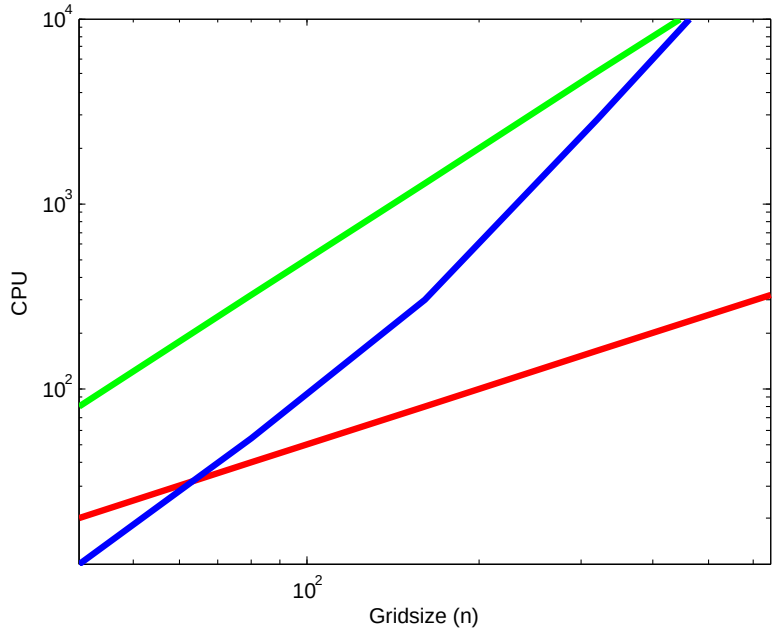


Figure 4.6: The CPU time in a loglog plot against the gridsize. The CPU time is given in blue, first order is given in red and second order is given in green.

4.3 Droplet/Droplet

For particle-particle interaction a head on collision of two equally sized particles is the simplest start. That is why we did a DNS of a droplet/droplet collision. We simulated this case first with physical parameters, however the simulation did not seem stable and too much was happening. That is why we choose to use the mathematical parameters, used for the droplet/droplet collision in the paper of Cifani [9]. The values of these parameters can be found in table 4.7.

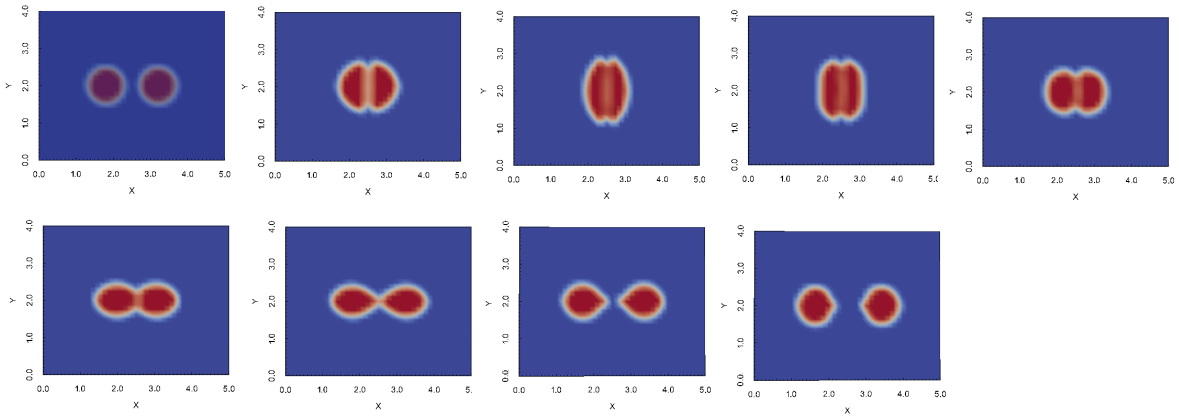
	symbol	value
Density of the fluid	ρ_f	0.001
Density of the droplets	ρ_b	1
Viscosity of the fluid	μ_f	0.0005
Viscosity of the droplets	μ_b	0.05
Surface tension of the droplets	σ	0.24
Gravity	$\boldsymbol{g}$	$(0\ 0\ 0)$
Initial location of the droplet 1	$(x, y)_1$	$(1.8, 2)$
Initial location of the droplet 2	$(x, y)_2$	$(3.2, 2)$
Droplet radius	r	0.5
Dimensions of the box	X, Y	$[0, 5], [0, 4]$
Gridsize	$(n_x$ x $n_y)$	40x32, 80x64, 160x128, 320x256, 640x512
Maximum Courant number	C	0.45
Maximum Interface Courant number	C_I	0.45

Table 4.7: Physical quantities used for the head on collision of two droplets.

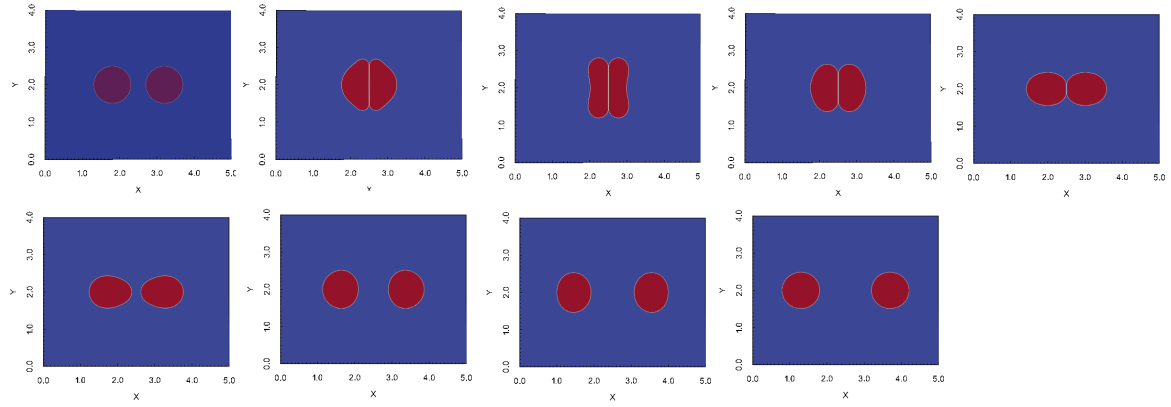
To show what we are simulating we made another time loop, for two different grids. This shows once again that grid refinement is very important. These time loops can be found in figure 4.7. From this time loop it seems that it takes approximately 1 second to get to the height of the colision but it takes almost twice as long to get back to the original shape. To look into the difference between the different grid sizes more clearly we made a contour plot at six interesting times. These plots can be found in figure 4.8. In the contour plot we can once again see convergence. We did the convergence study with the Hysing quantities for both the droplets separately, where the rise velocity is now the droplet velocity. The equations used for the calculation can be found in section [4.1], equations 4.1 - 4.7. In Table 4.8 and Table 4.9 you can find the results for the convergence, respectively the statistics of the simulation. We also like to make sure that the two droplets in the collision are symmetric, that is why we decided to compare the circularity of both the droplets. This can be found in figure 4.9. In this figure we can see that on a coarse grids there is a slight dissymmetry, however this disappears with finer resolution.

Conclusion

This is the first simulation in which two particles, and thus the first simulation in which the multi marker approach is used. We get good results with the solver, with a symmetric setup we get symmetric results on fine resolutions. We see once again the sligthly higher than first order convergence. We see that the computing cost are similar to those of the first simulation (Section [4.1]). We do however have less grid cells, the difference in cost is probably due to the fact that we have two particles. We chose this case due to its relevance of the near contact question. It is hard to answer if lubrication theory is needed, but we can clearly see that the collision on finer grids leads to a slightly higher rebound velocity. Which could indicate that the pressure between the particles is underestimated on coarse grids.

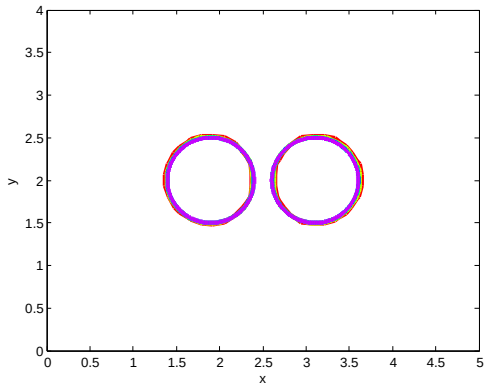


(a) grid 40x32

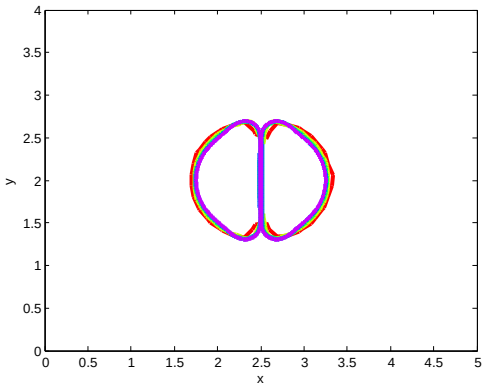


(b) grid 640x512

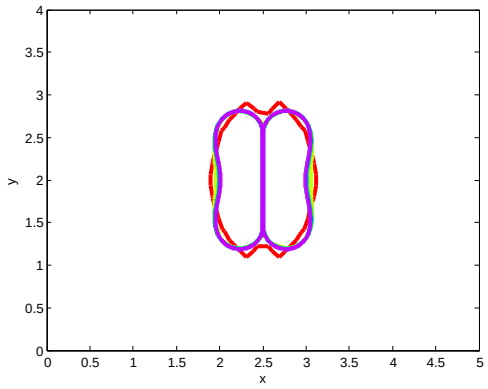
Figure 4.7: The volume fraction field, α, for the different grids at different time steps, starting at $0s$ with intervals of $0.5s$ till $4s$. Here the inside of the droplet is red, while the outer fluid is blue.



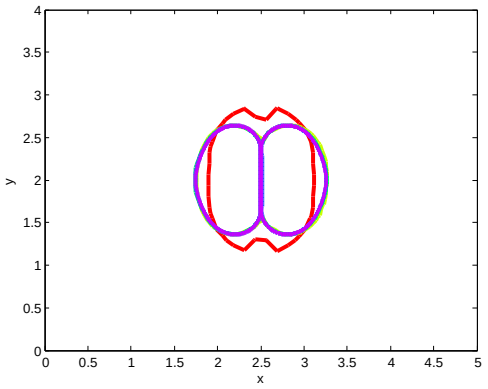
(a) $t = 0.1$



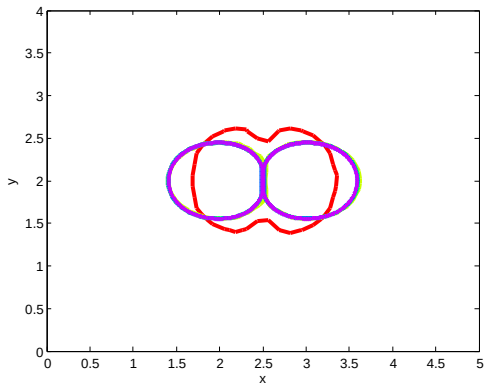
(b) $t = 0.5$



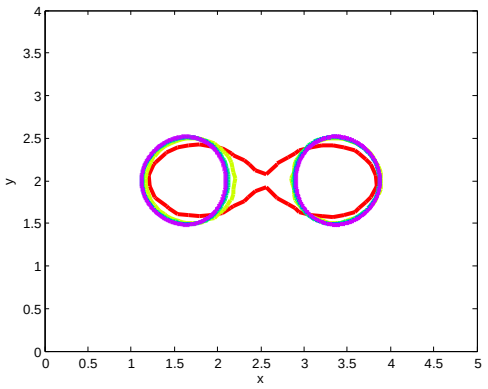
(c) $t = 1$



(d) $t = 1.5$



(e) $t = 2$



(f) $t = 3$

Figure 4.8: The contour, $c = 0.5$, of the two droplets on the different grids at different time steps. Grid 40x40 is in red, 80x80 is in yellow, 160x160 is in green, 320x320 is in blue, and 640x640 is in purple.

n_x	$\|\|e\|\|_1$	ROC_1	$\|\|e\|\|_2$	ROC_2	$\|\|e\|\|_\infty$	ROC_∞
$\boldsymbol{\alpha}_1$						
Center of mass						
40	3.15E-01		3.24E-01		4.65E-01	
80	1.43E-01	1.14	1.46E-01	1.15	2.04E-01	1.19
160	6.73E-02	1.08	6.79E-02	1.10	7.97E-02	1.36
320	2.19E-02	1.62	2.21E-02	1.62	2.65E-02	1.59
Droplet velocity						
40	5.41E-01		5.21E-01		4.97E-01	
80	3.26E-01	0.73	3.01E-01	0.79	2.99E-01	0.73
160	1.58E-01	1.04	1.55E-01	0.96	1.72E-01	0.79
320	6.49E-02	1.28	6.22E-02	1.32	6.74E-02	1.35
Circularity						
40	2.90E-02		3.88E-02		9.24E-02	
80	6.39E-03	2.18	8.63E-03	2.17	1.74E-02	2.41
160	2.91E-03	1.14	4.04E-03	1.09	8.09E-03	1.11
320	9.44E-04	1.62	1.25E-03	1.69	2.81E-03	1.53
$\boldsymbol{\alpha}_2$						
Center of mass						
40	3.35E-01		3.39E-01		4.21E-01	
80	1.44E-01	1.22	1.45E-01	1.23	1.73E-01	1.29
160	6.28E-02	1.19	6.36E-02	1.19	7.48E-02	1.21
320	2.13E-02	1.56	2.17E-02	1.55	2.77E-02	1.43
Droplet velocity						
40	5.07E-01		5.07E-01		5.03E-01	
80	2.78E-01	0.86	2.81E-01	0.82	2.92E-01	0.77
160	1.58E-01	0.81	1.53E-01	0.88	1.67E-01	0.84
320	6.46E-02	1.29	6.03E-02	1.34	6.57E-02	1.39
Circularity						
40	2.98E-02		4.09E-02		9.93E-02	
80	6.65E-03	2.16	9.25E-03	2.14	1.86E-02	2.41
160	3.04E-03	1.13	4.31E-03	1.10	9.07E-03	1.04
320	9.51E-04	1.68	1.32E-03	1.71	2.91E-03	1.64

Table 4.8: Error norms and convergence rates for the case of the head on collision between two droplets.

n_x	NTS	CPU in s
40	226	6.9
80	241	24.86
160	353	136.62
320	588	1118.1
640	1116	18407.65

Table 4.9: Statistics from the simulation of the case of the head on collision between two droplets.

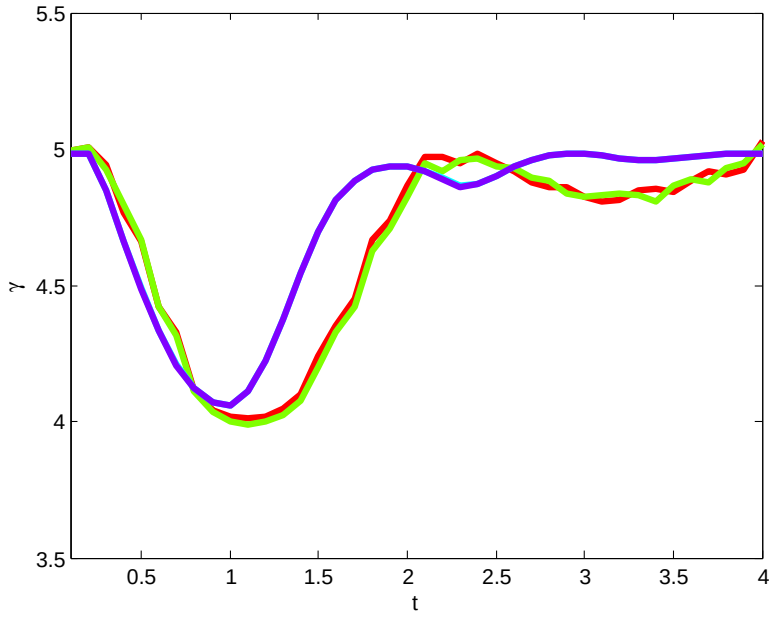


Figure 4.9: The circularity of the two droplets on diferent grids. Here the circularity of the left bubble is given in red for the coarsest grid, and in blue for the finest grid. The circularity of the right bubble is given in green for the coarsest grid and in purple for the finest grid.

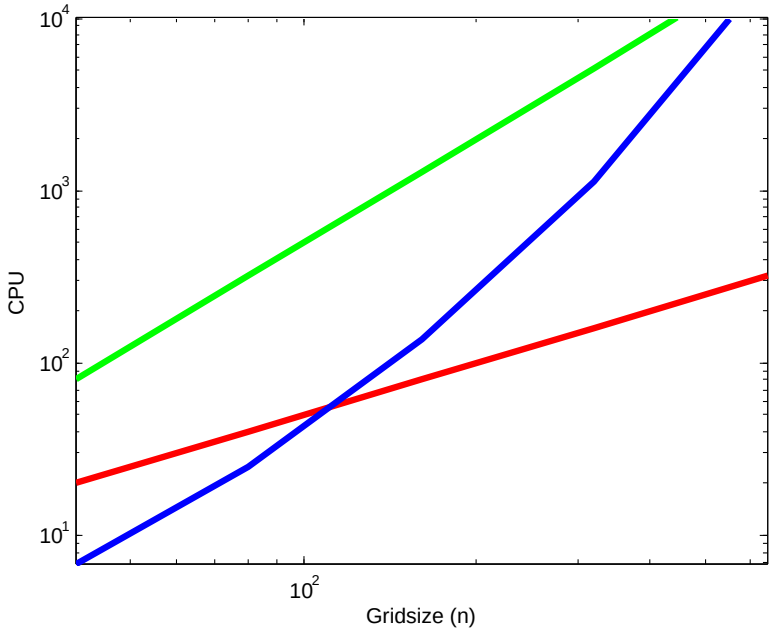


Figure 4.10: The CPU time in a loglog plot against the gridsize. The CPU time is given in blue, first order is given in red and second order is given in green.

Chapter 5

3D comparison

In this chapter we will use the VofMMRK solver in a 3D setting. We will simulate the droplet/droplet collision again and compare it to the results of Cifani [9]. This case is simulated in parallel, so we also added a section on parallelization and speedup.

5.1 3D Droplet/Droplet Collision

We also did the simulation of a droplet/droplet collision in 3D. We added this DNS to make another comparison and see the work of the solver for 3D cases. This could also be used to test if the models produced in 2D would obtain similar results in 3D. The parameters for this case can be found in table 5.1. This case was also done with the code mphBox of Cifani [9]. A visual comparison of the results of these codes is done. We were planning on doing a finer resolution case as well to see how the code evolved compared to the results gained by Cifani [9]. However due to some problems with the initial velocity and high computational cost this could not be achieved in the time frame.
To show what we are simulating we made another time loop, which can be found in figure 5.1. For the

	symbol	value
Density of the fluid	ρ_f	0.001
Density of the particles	ρ_b	1
Viscosity of the fluid	μ_f	0.0005
Viscosity of the particles	μ_b	0.05
Surface tension of the particle	σ	0.24
Gravity	g	0
Initial location of the particle 1	$(x, y, z)_1$	$(1.3,\ 1.5,\ 1.5)$
Initial location of the particle 2	$(x, y, z)_2$	$(2.7,\ 1.5,\ 1.5)$
Bubble radius	r	0.5
Dimensions of the box	X, Y, Z	$[0, 4], [0, 3], [0, 3]$
Gridsize	$(n_x$ x n_y x $n_z)$	128 x 96 x 96
Timestep	dt	0.0004

Table 5.1: Physical quantities used for a head on collision of two droplets in 3D.

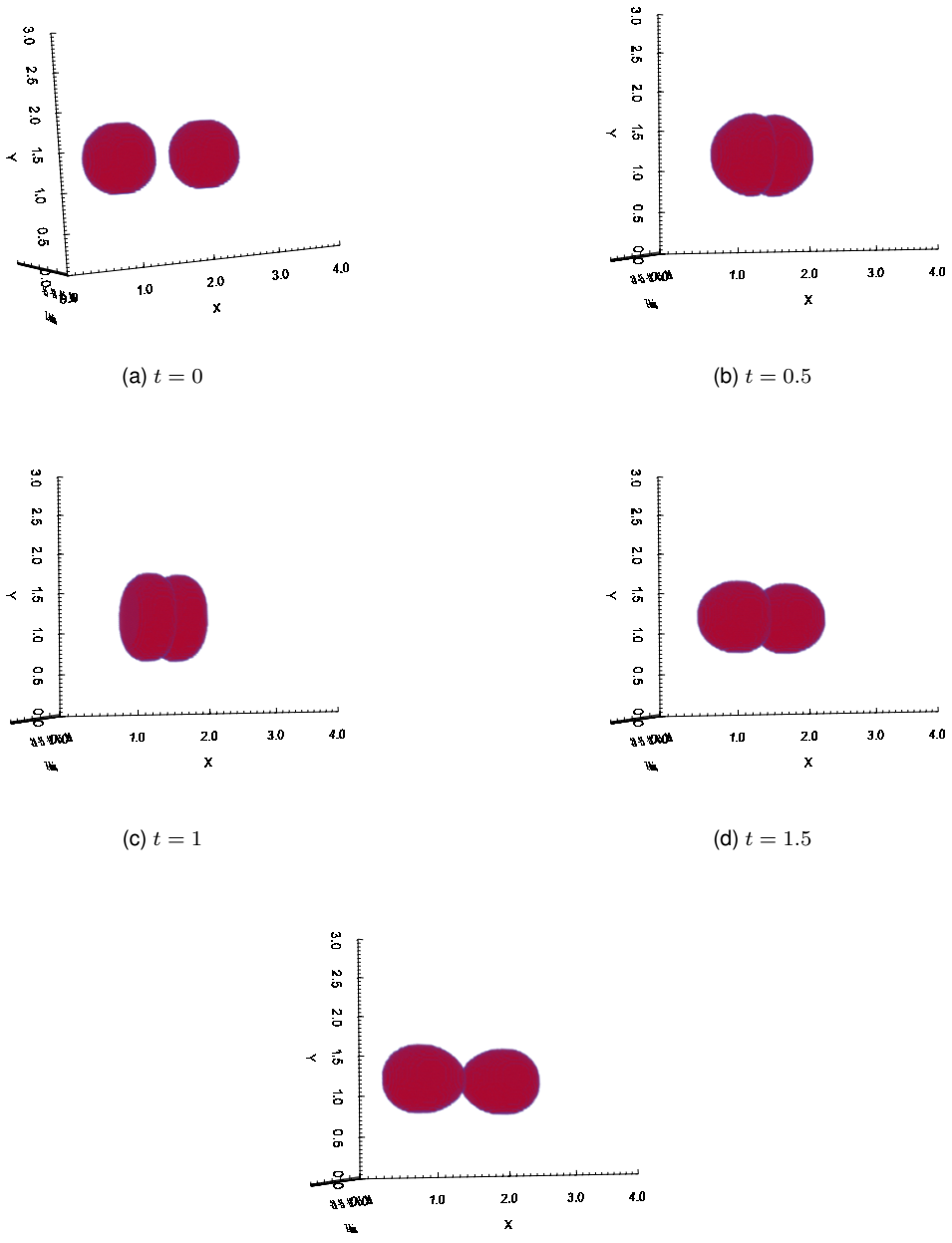
comparison of the VofMMRK method with the mphBox, we made a slice right down the middle of the z-axis and made a vector field view of the velocity at different time steps, which can be found in figure 5.2. We see that the results almost overlap, though the differences grow a little with time.
The statistics can be found in table 5.2.

n_x	NPU	NTS	execution Time in s
128	16	10000	131381.88

Table 5.2: Statistics from the simulation of the case of the head on collision between two particle in 3D, NPU is number of processors used

Conclusion

The solver is able to simulate 3D simulations, which compare quite accurately with the solver from Cifani [9]. It seems however that the VofMMRK solver is slightly delayed with respect to the droplet movement compared to mphBox [9]. When looking at section [4.3] this might improve with higher resolution.
It would be good to do some grid refinement, or even coarsening, to check the convergence of the 3D solver. However doubling the grid in every direction will probably take at least eight times as long, and probably slightly more looking at 2D results. We do not have computation cost from the mphBox solver, so the only comparison for this are the 2D cases. It is however hard to make a comparison between these cases. When we look at the computation time per timestep per grid cell, we see that this is of the same order with the 2D case. However for the 3D case we are using 16 processors whereas for the 2D case we only used 1.



(a) $t = 0$

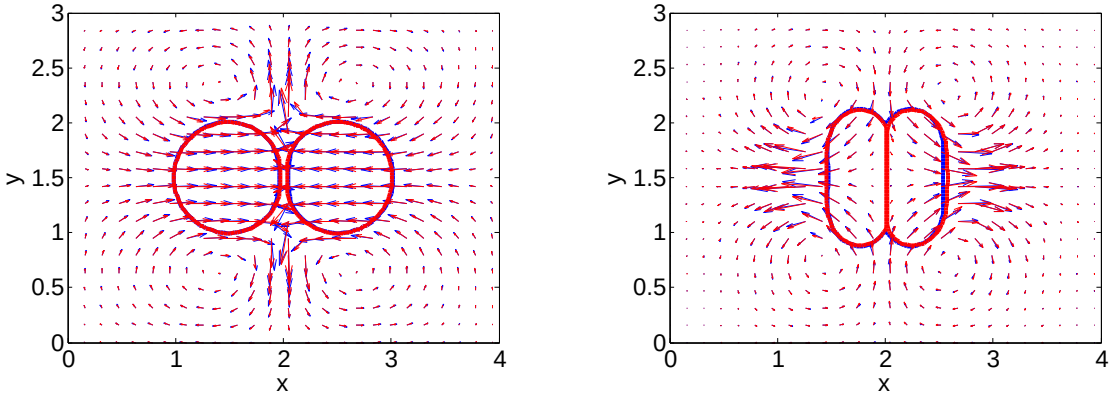
(b) $t = 0.5$

(c) $t = 1$

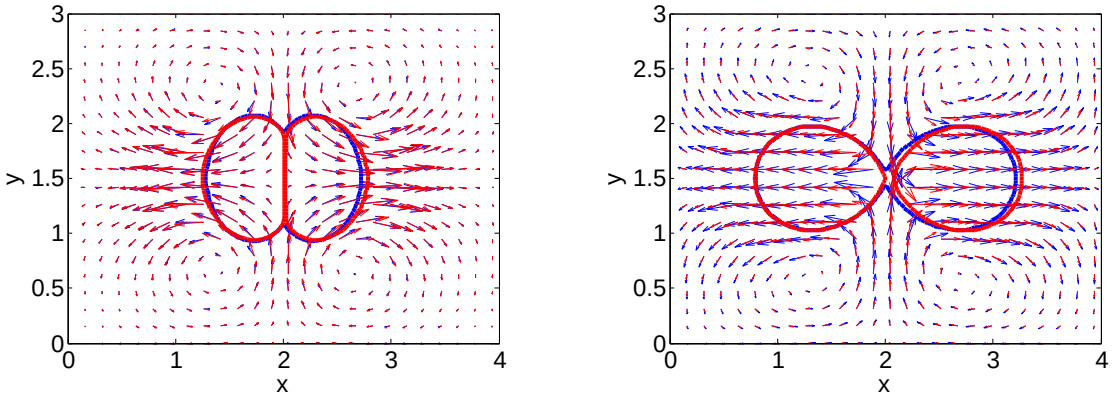
(d) $t = 1.5$

(e) $t = 2$

Figure 5.1: The volume fraction, α, at different time steps, starting at $0s$ with intervals of $0.5s$ till $2s$. Where the droplets are red.



(a) $t = 0.2$ (b) $t = 1$



(c) $t = 1.2$ (d) $t = 2$

Figure 5.2: The velocity vector field (U) and the droplet location at the different times. The values of vofMMRK are given in blue and the results of mphBox are given in red.

5.2 Parallelization speedup

In section [5.1] the abbreviation NPU, number of processors used, was already mentioned. By using multiple processors we can reduce the total computation time, this is called parallelization. OpenFOAM has built-in scripts for parallelization, the method of parallel computing used by OpenFOAM® is based on domain decomposition [22]. This method is done in three phases, decomposing in seperate domains, using the decomposePar utility, running the application in parallel, with the MPI (Message passing interface) Run command, lastly you can reconstruct the entire domain, using the reconstructPar utility, or post-process each of the domains seperately.
The mesh decomposition can be done in multiple ways, here we used simple geometric decompositon which splits the domain into the number of pieces in each direction given by the vector. For example for 16 processors we used the decomposition $(4\ 2\ 2)$ dividing the $x-$direction in four parts, the $y-$direction in two, and the $z-$direction in two.
Parallelization would ideally reduce the computation time by half if double the number of processers are used, this is however usually not the case. This is due to the communication needed between the processors on the boundaries of the decomposed domains, such that there are no discontinuities. Other overhead cost could be in the tracking of particles, which could be necessary for the expansion to particle-particle interaction [28]. The way in which you decompose your domain could also be of influence.
In table 5.3 you can see the results of the speedup for the finest grid of the 3D case mentioned in section 5. We only simulated a short time (till $t = 0.001$), to get an idea of the speedup. Where we used the following decompositions $(1\ 1\ 1), (2\ 1\ 1), (4\ 1\ 1), (2\ 2\ 2), (4\ 2\ 2)$, and $(8\ 2\ 2)$. The speedup is then given by the following calculation Speedup$(NPU) = \frac{\text{CPU}(NPU/2)}{\text{CPU}(NPU)}$. We used the parameters found in table 5.1, except that the grid size was larger, namely 256x192x192. We used both the adjustable time step function from OpenFOAM® as a stationary time step of d$t = 0.000025$.

NPU	CPU in s	Mean CPU per timeStep in s	Speedup
ATS			
1	5611	401	
2	3393	242	1.65
4	2192	157	1.55
8	1342	96	1.63
16	708	51	1.90
32	623	45	1.14
SDT			
1	12911	323	
2	9571	239	1.35
4	9086	227	1.05
8	6712	168	1.35
16	4601	115	1.46
32	5877	147	0.78

Table 5.3: Speedup values for the fine 3D droplet/droplet case, where ATS stands for Adjustable Time Step, and SDT stands for stationary dt. We simulated the fine 3D case from section 5.1 till $t = 0.001$ for a different number of processors.

Conclusion

Ideally we wish the speedup to be equal to 2 since we double the number of processors, however you can see that it is slightly less, as expected. We can see that the adjustable time step is more computational expensive when using just a few processors but gets cheaper as the number of processors increases. Also notable is the fact that the speedup is doing welll up to 16 processors for this problem, but at 32 processors the values decrease, for the stationary time step it even gets slower. This might be because at 32 processors the communication between the processors about the boundary conditions has higher computational cost than the actually algorithm itself. It is also possible that a different decomposition leads to a better speedup or that the case simulated is simply not big enough for that number of processors, leading to a decrease in efficiency.
The overall lower computational cost of the adjustable time step is easy too explain, since only 14 time steps were taken compared to the 40 of the stationary time step. We looked into the CPU time a bit more, and saw it could be composed into four groups, The VOF algorithm (2-4), the U equations (5), the first P equations (6-10) and the second P equations (6-10). Where between brackets we mentioned the steps of the PISO algorithm seen in section [2.2]. Where the VOF algorithm and the U equations were

comparable, the P equations stood out. Why the difference in CPU time for these equations is so much higher for the stationary time step than for the adjustable time step, I cannot explain. The calculations are added in appendix B.
This results may vary for different cases, but we like to recommend using if possible at least 16 processors when simulating 3D data, since it will save a lot of time.

Chapter 6

Lubrication

6.1 History and scientific relevance of lubrication

So far we have been dealing with the description of a two fluid flow system. In such a system different types of interaction take place. The particles are moved by the fluid, but at the same time influence the fluid itself. The walls influence both the flow and the location of the particles. These interactions are already taken into account by the transport equation 2.5, the interfacial force source term in the Navier stokes equations 2.1 and the boundary conditions on the field. However there is another type of interaction, namely the particle-particle interaction. This interaction consists of a number of different phenomena, like collisions, resulting in coalescence, bouncing or deformation, breakups and repulsive forces. Each of these phenomena is an intricate problem for simulation. For example the Volume of Fluid method contains the problem of numerical collision [20], which happens when particles are within one grid cell distance of each other. The extension to the multiple marker approach resolves this problem, but prohibits collisions completely.
The accurate simulation of particle-particle interaction is more important when more particles are involved, Yao et al. [30] mentiones that the particle-particle interaction will even dominate the hydrodynamic behaviors in high concentration cases. The research done in this field is extensive, however most of the papers target only one specific problem, like two rigid spheres in an unbounded fluid of equal size ([17]), or two particles freely moving in a cylinder [21] and [30]. Based on this research different interaction models are also investigated, like the following collision models: critical velocity models ([19]), energy models ([14], [27]), film drainage models ([20], [11], [7]) or damper-spring models ([18]).
Many of these methods are based on Lubrication theory. Lattice Boltzman and Stokes multipole simulatians incorperarte the lubrication force when the particles are in near contact. While in the distributed Lagrange multiplier method they are applied to the particles instead of refining the grid [1].
Yao et al. [30] mentions that their analysis reveals that particle-particle interaction can be neglected when the separation distance is three times larger than the sum of particles radii when the two particles are identical, while Kwakkel [20] mentions that coalescence typically takes place if the film thickness h is of $O(100)\ nm$, at this scale the van der Waals force becomes active. However a lot of interaction is still possible between those occurences. And making a grid where each cell is of order $O(100)\ nm$ makes it nearly impossible to simulate even small problems. Local refinement methods could be a solution to this problem[29]. However we choose to look into the possibility of applying the thin-film lubrication theory on a coarse grid, to get more accurate solutions. How this could be achieved can be found in section 6.3.
The motivation for this approach is that the thin film equation leads to the loss of a dimension, making a 3D problem a 2D problem, and making the equations analytically solveable. The classical theory, as expanded below, states that the lubrication force goes to infinity as the distance between the particles goes to zero. This would mean that the particles would be prevented from touching. The theory of lubrication originates from O. Reynolds, with his wel known Reynolds equations, and his heuristic way of finding these equations to apply to industrial purposes, like sliding bearings or circular bearings [24]. Since 1886 extensive research has been done on these equations. The use has been expanded to model all different kind of lubricant layers, and multiple generalisations have been done to adhere to the shape of the problem [2].

6.2 Theory

We will now look at the classical Lubrication theory. We will show a heuristic approach of deriving the Reynolds equations from the Navier Stokes equations. We will get the simplification by assuming a thin film geometry between the surfaces (i.e., one dimension in the fluid domain is far smaller than the

other two)[2]. A more rigorous mathematical approach, based on assymptotic expansion can be found in [3]. First we consider an incompressible Newtonian lubricant in an isothermal and isoviscous regime, exactly like the assumption made in section 2.1 except that instead of a two-fluid system we only have one fluid. The Navier-Stokes equations for this case are given by equations 6.1 and 6.2.

$$\rho\left[\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right] = -\nabla p + \nu\Delta\boldsymbol{u} + \rho\boldsymbol{f}, \tag{6.1}$$

$$\nabla\cdot\boldsymbol{u} = 0. \tag{6.2}$$

Once again $\boldsymbol{u} = (u\ v\ w)$ and p denote respectively the fluid velocity field and the pressure, ρ, ν and $\boldsymbol{f}$ the density, kinematic viscosity and external volume forces. To reduce the complexity of this equations in this geometry, thin film, we get the simplification from the fact that the length scale (L_{xy}) in the xy plane of the film is significantly larger than the length scale L_z in the $z-$direction. When the fraction is small, typically $\epsilon = \frac{L_z}{L_{xy}} \leq O(10^1)$ [2], we can apply scaling of the original equations to reduce the dimension of the problem.
We will define the new spatial coordinates and velocity components as follows:

$$X = \frac{x}{L_{xy}}, \quad Y = \frac{y}{L_{xy}}, \quad Z = \frac{z}{L_z}, \tag{6.3}$$

$$U = \frac{u}{U_{xy}}, \quad V = \frac{v}{U_{xy}}, \quad W = \frac{w}{Uz}. \tag{6.4}$$

From equation 6.2 it is then easy to see that $Uz = \epsilon Uxy$. And the scaled pressure and scale time can be defined as in equation 6.5 and equation 6.6 respectively.

$$P = p\frac{\epsilon Re}{\rho U_{xy}^2}, \tag{6.5}$$

$$T = t\frac{U_{xy}}{L_{xy}}. \tag{6.6}$$

Here the Reynolds number is defined as $Re = \frac{U_{xy}L_{xy}}{\nu}$.
We can now introduce all the new variables (of equations 6.3, 6.4, 6.5, and 6.6) into equations 6.1 and 6.2. Neglecting the body forces ($\boldsymbol{f}$), and terms smaller then $O(\epsilon^2)$, we will obtain the 3D-Reynolds equations given in equation 6.7.

$$\begin{aligned}
\epsilon^2 Re(\frac{\partial U}{\partial T} + U\frac{\partial U}{\partial X} + V\frac{\partial U}{\partial Y} + W\frac{\partial U}{\partial Z}) &= -\frac{\partial P}{\partial X} + \frac{\partial^2 U}{\partial Z^2},\\
\epsilon^2 Re(\frac{\partial V}{\partial T} + U\frac{\partial V}{\partial X} + V\frac{\partial V}{\partial Y} + W\frac{\partial V}{\partial Z}) &= -\frac{\partial P}{\partial X} + \frac{\partial^2 V}{\partial Z^2},\\
\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} + \frac{\partial W}{\partial Z} &= 0.
\end{aligned} \tag{6.7}$$

To be noticed is that the missing equation in the momentum conservation implies that the pressure is constant across the lubricant film. Moreover, passing to the limit when ϵ tends to zero and coming back to the original variables, we get the simplified model of equation 6.8.

$$\begin{aligned}
\frac{\partial p}{\partial x} &= \mu\frac{\partial^2 u}{\partial z^2},\\
\frac{\partial p}{\partial y} &= \mu\frac{\partial^2 v}{\partial z^2},\\
\nabla\cdot\boldsymbol{u} &= 0.
\end{aligned} \tag{6.8}$$

6.3 Application

As seen in the Bubble/Wall and the Droplet/Droplet collision cases there is convergence happening between the grid sizes. The Bubble/Wall shows a clear gap in the coarser grids, which could be due to an underestimation in the outflow from the thin film, a lubrication correction might help solve this without going to finer resolutions. In the Droplet/Droplet case the motivation is a bit less clear, however the rebound velocity seems to be underestimated on coarser grids.
It is also good to notice that the computation costs gets huge for finer grids and 3D-simulations, using lubrication theory and a coarse grid could possibly reduce these cost while still leading to accurate solutions. When investigating the application of lubrication theory to this problem it is wise to start with

a 2D investigation, luckily a similar reduction to a 2D-Reynolds equation exists. As mentioned in section 6.2 some assumptions were made in the deduction of the Reynolds equations. One of them involved that the boundaries of the fluid film are solid walls. This is obviously not the case in this research, however from the motivation given in section 6.1 it seems a correction is needed on coarse grids, and this theory might improve results on coarse grids.
First we will look at the 2D equations, where $\epsilon = \frac{L_y}{L_x}$, and solve them according to the approach of [5].

$$\begin{aligned} \frac{\partial p}{\partial x} &= \mu \frac{\partial^2 u}{\partial y^2}, \\ \frac{\partial p}{\partial y} &= 0. \end{aligned} \tag{6.9}$$

We will combine this with the fact that the change in height ($h = h_1 - h_2$), should be equal to the outflow (Q) in the $x-$direction, or simply put

$$\begin{aligned} \frac{\partial h}{\partial t} + \frac{\partial Q}{\partial x} &= 0, \\ \text{where} \qquad Q(x) &= \int_{h_1(x)}^{h_2(x)} u(x,y)dy. \end{aligned} \tag{6.10}$$

Since in equation 6.9 the pressure is independent of y, we can integrate the velocity in the $x-$direction over y twice and get:

$$u(x,y) = \frac{1}{2\mu}\frac{\partial p}{\partial x}(y^2 - (h_2+h_1)y - \frac{1}{2}(h_1^2 + h_1 + h_2^2 + h_2)) + \frac{u_2 - u_1}{h_2 - h_1}y + \frac{u_1 h_2 - u_2 h_1}{h_2 - h_1}. \tag{6.11}$$

Here we used $u(x,h_1) = u_1, u(x,h_2) = u_2$. We will then use equation 6.10 to get:

$$\begin{aligned} &\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(\int_{h_1(x)}^{h_2(x)} (\frac{1}{2\mu}\frac{\partial p}{\partial x}(y^2 - (h_2+h_1)y - \frac{1}{2}(h_1^2 + h_1 + h_2^2 + h_2)) + \frac{u_2 - u_1}{h_2 - h_1}y + \frac{u_1 h_2 - u_2 h_1}{h_2 - h_1})dy) = 0, \\ &\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}(\frac{u_1 h_2 - u_2 h_1}{2} + \frac{\frac{2}{3}(h_1^3 + h_2^3) + \frac{1}{2}(h_1^2 - h_2^2)}{2\mu}\frac{\partial p}{\partial x}). \end{aligned} \tag{6.12}$$

Intergrating this twice over x for a given domain (x_0, x_1) we get:

$$p = \int_{x_0}^{x_1} \frac{2\mu}{\frac{2}{3}(h_1^3 + h_2^3) + \frac{1}{2}(h_1^2 - h_2^2)} (\int_{x_0}^{x_1} \frac{\partial h}{\partial t}dx - \frac{u_1 h_2 - u_2 h_1}{2})dx. \tag{6.13}$$

We now have an equation for the pressure in the thin film gap (equation 6.13), and an equation for the velocity (equation 6.11).
There are some ways in which this could be applied to the given method. First of all we could do a numerical implementation in which we do a lubrication correction for a certain domain. So we could first calculate the solution over the whole domain. Look at the locations where particles are within a certain distance and get a new computational domain, the 'thin' film between the particles. We could then calculate the solution to the Reynolds equations where we use the originally calculated velocity and pressure values at the boundaries as the boundary conditions. An obtain for that distance was mentioned in Yao et al. [30], particle-particle interaction becomes dominant if two particles come within three times the sum of the particle radii of each other. However a parameter study could be done on the best value of this distance, since the thin film equations are based on a thinner film than three times the sum of the particle radii.
A second option which could be interesting to examine is adding a source term in the transport equation of the volume fraction 2.5, to correct for the difference in flow between the Reynolds equations and the Navier Stokes solution.
One could try to apply these methods to the coarse grid solution with $n_x = 40$ and then compare the results with the fine grid solution $n_x = 640$. When implementing this method into a multi-particulate flow more problems may arise with parallelization and tracking collisions efficiently, though outside the scope of this paper some information can be found in [28].

Chapter 7

Conclusion and Recommendations

No analytical solution for multi-bubble systems exist, and experimental results are often muddled. Therefor it is important to have dependable simulation methods for applications in industry or nature. Multi-bubble system need to incorperate all types of fluid interaction. In this paper we look for ways to improve particle interaction, while still staying computationally cheap. This was done by a literature study into Lubrication theory and a DNS of two simple cases, Bubble/Wall and Droplet/Droplet.
The simulation was done by th VOF-PLIC method of Cifani [8] with the multiple marker extension. This approach, based on Kwakkel [20], prevents numerical coalescence and makes bouncing of particles possible. We examined the accuracy of this method by comparing it with the Hysing benchmark study [15], and doing a spatial convergence investigation. The VofMMRK method gave accurate results when comparing it to the benchmark. Good results were already attained on the coarsest grid, this combined with a slightly higher than first order convergence and low computation cost makes this method suitable for further investigation. The theoratical 2nd order convergence was not reached, changing tolerances or number of corrections in the PISO scheme might still make this 2nd order attainable. It would also be good to do an investigation into what Courant number should be used when using the adjustable time step, since using this did lead to different results.
For the analysis of the 3D case we had to look in a different direction. We compared the results of the VofMMRK method with mphBox from Cifani [9]. Once again we obtained a visually good overlap for the results. However it was difficult to get quantitative results about the performance. This was due to some time constraint and considerably larger computation cost. It would be good if some convergence study could still be done. A good tool for 3D simulations is parallelization, using more processors to speed up the computation time. Parallelization by mesh decomposition was investigated and showed a decrease of almost half of the computation time when using double the amount of processors.
We examined the cases Bubble/Wall and Droplet/Droplet to get a better idea of the application needed to get better results. One of the questions was if this cases are similar. This was impossible to answer due to different parameters, though the question still remains interesting and could be investigated in the future. The Bubble/Wall showed the thin film geometry described in Lubrication theory. In this simulation the finer grids clearly resolved the flow in this thin film better than the coarser grids. It seems as if this phenomena could certainly be approved by applying the lubrication theory. In the Droplet/Droplet case the necessity was less obvious. However the rebound speed on the coarser grids was lower. It is possible that applying lubrication theory could improve this.
Some background information on particle-particle interaction and lubrication theory has been given. Future research could be done on the cases simulated in this work. One could try to apply the theory in multiple ways, some initial ideas are suggested here, namely a partail domain replacement or a source term in the volume fraction advection equation. However a lot of other different methods for particle interaction do already exist. It would be interesting to get a clear overview of the existing methods, and a comparison of their performances. It would then also be interesting to see how different collisions are resolved as well, think of different radii or different angle of approach, since these are rarely discussed in papers.
Lastly I would like to make some recommendations with respect to the VofMMRK solver and its implementation. The solver seems to be working well for one and two particles and gets fairly accurate results with relatively coarse grids. These factors make this solver ideal for doing initial investigation, however the method is now only working for up to 2 particles. If one would like to simulate multi-particle solutions an extension of the initAlpha utility is needed. On top of that it would be wise to make the initial velocity a utility as well, so that you do not have to recompile the code for every different situation.

Bibliography

[1] Ardekani, A. M., and R. H. Rangel. ”Numerical investigation of particleparticle and particlewall collisions in a viscous fluid.” Journal of fluid mechanics 596 (2008): 437-466.

[2] Bayada, Guy, and Carlos Vzquez. ”A survey on mathematical aspects of lubrication problems.” Bol. Soc. Esp. Mat. Apl 39 (2007): 31-74.

[3] Bayada, Guy, and Michele Chambat. ”The transition between the Stokes equations and the Reynolds equation: a mathematical proof.” Applied mathematics & optimization 14.1 (1986): 73-93.

[4] Bendiksen, Kjell H., et al. ”The dynamic two-fluid model OLGA: Theory and application.” SPE production engineering 6.02 (1991): 171-180.

[5] Berthe, D., and M. Godet. ”A more general form of Reynolds equation-application to rough surfaces.” Wear 27.3 (1974): 345-357.

[6] Brackbill, J. U., Douglas B. Kothe, and Charles Zemach. ”A continuum method for modeling surface tension.” Journal of computational physics 100.2 (1992): 335-354.

[7] Chesters, A.K. ”The modelling of coalescence processes in fluid-liquid dispersions: a review of current understanding.” Chemical engineering research & design 69.A4 (1991): 259-270.

[8] Cifani, Paolo, et al. ”A comparison between the surface compression method and an interface reconstruction method for the VOF approach.” Computers & Fluids 136 (2016): 421-435.

[9] Cifani, Paolo. ”DNS of turbulent bubble-laden channel flows.” Ph.D. Thesis, University of Twente, 2017. ISBN : 978-90-365-4416-0 DOI: 0.3990/1.9789036544160

[10] Clift, Roland, John R. Grace, and Martin E. Weber. Bubbles, drops, and particles. Courier Corporation, 2005.

[11] Davis, Robert H., Jeffrey A. Schonberg, and John M. Rallison. ”The lubrication force between two viscous drops.” Physics of Fluids A: Fluid Dynamics 1.1 (1989): 77-81.

[12] Hoang, Duong A., et al. ”Benchmark numerical simulations of segmented two-phase flows in microchannels using the Volume of Fluid method.” Computers & Fluids 86 (2013): 28-36.

[13] Holzmann, Tobias. ”Mathematics, Numerics, Derivations and OpenFOAM.” (2016).

[14] Howarth, W. J. ”Coalescence of drops in a turbulent flow field.” Chemical Engineering Science 19.1 (1964): 33-38.

[15] Hysing, Shu-Ren, et al. ”Quantitative benchmark computations of twodimensional bubble dynamics.” International Journal for Numerical Methods in Fluids 60.11 (2009): 1259-1288.

[16] H. Jasak, Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows, Ph.D. Thesis, Imperial College, London, 1996.

[17] Jeffrey, D. J., and Y. Onishi. ”Calculation of the resistance and mobility functions for two unequal rigid spheres in low-Reynolds-number flow.” Journal of Fluid Mechanics 139 (1984): 261-290.

[18] Lankarani, H. M., and P. E. Nikravesh. ”A contact force model with hysteresis damping for impact analysis of multibody systems.” Journal of mechanical Design 112.3 (1990): 369-376.

[19] Lehr, F., and D. Mewes. ”A transport equation for the interfacial area density applied to bubble columns.” Chemical Engineering Science 56.3 (2001): 1159-1166.

[20] Kwakkel, Marcel, Wim-Paul Breugem, and Bendiks Jan Boersma. ”Extension of a CLSVOF method for droplet-laden flows with a coalescence/breakup model.” Journal of Computational Physics 253 (2013): 166-188.

[21] Navardi, Shahin, Sukalyan Bhattacharya, and Hanyan Wu. ”Stokesian simulation of two unequal spheres in a pressure-driven creeping flow through a cylinder.” Computers & Fluids 121 (2015): 145-163.

[22] OpenCFD Ltd. User and programmer’s guide; 2013. URL: http://www.openfoam.com

[23] Puckett, Elbridge Gerry, et al. ”A high-order projection method for tracking fluid interfaces in variable density incompressible flows.” Journal of Computational Physics 130.2 (1997): 269-282.

[24] Reynolds, Osborne. ”On the Theory of Lubrication and Its Application to Mr. Beauchamp Tower’s Experiments, Including an Experimental Determination of the Viscosity of Olive Oil.” Proceedings of the Royal Society of London 40.242-245 (1886): 191-203.

[25] Rusche H. Computational fluid dynamics of dispersed two-phase flows at high phase fractions. Imperial College London (University of London); 2003 Ph.D. thesis

[26] Scardovelli, Ruben, and Stephane Zaleski. ”Analytical relations connecting linear interfaces and volume fractions in rectangular grids.” Journal of Computational Physics 164.1 (2000): 228-237.

[27] Sovova, H. ”Breakage and coalescence of drops in a batch stirred vesselII comparison of model and experiments.” Chemical Engineering Science 36.9 (1981): 1567-1573.

[28] Traoré, Philippe, Jean-Charles Laurentie, and Lucian Dascalescu. ”An efficient 4 way coupling CFDDEM model for dense gassolid particulate flows simulations.” Computers & Fluids 113 (2015): 65-76.

[29] Wan, Decheng, and Stefan Turek. ”Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows.” Journal of Computational Physics 222.1 (2007): 28-56.

[30] Yao, Xin, and Teck Neng Wong. ”Slow viscous flow around two particles in a cylinder.” Microfluidics and Nanofluidics 21.10 (2017): 161.

Appendix A

Benchmark results with stationary time step

We redid the case from section [4.1] with a stationary time step. We started with a dt of 0.001 for the coarsest grid, and halved it for every refinement. Due to some time constraint we ran the finer cases in parallel. We hoped that using a stationary time step would lead to second order convergence. The parameters are repeated for ease in table A.1. We once again calculated the quantities, error norms and rate of convergence defined in equations 4.1 - 4.7. The results can be found in Table A.2.

	symbol	value
Density of the fluid	ρ_f	1000
Density of the bubble	ρ_b	100
Viscosity of the fluid	μ_f	10
Viscosity of the bubble	μ_b	1
Surface tension of the bubble	σ	24.5
Gravity	$\boldsymbol{g}$	$(0\ -0.98\ 0)$
Initial location of the bubble	(x, y)	$(0.5, 0.5)$
Bubble radius	r	0.25
Dimensions of the box	X, Y	$[0, 1], [0, 2]$
Gridsize	$(n_x$ x $n_y)$	40x80, 80x160, 160x320, 320x640
Time step	dt	0.001, 0.0005, 0.00025, 0.000125

Table A.1: Physical quantities of the first test case of the Hysing Benchmark [15].

$1/h$	$\|\|e\|\|_1$	ROC_1	$\|\|e\|\|_2$	ROC_2	$\|\|e\|\|_{/inf}$	ROC_{inf}
Center of mass						
40	1.10E-01		1.12E-01		1.41E-01	
80	4.74E-02	1.22	4.77E-02	1.24	5.41E-02	1.39
160	1.60E-02	1.57	1.63E-02	1.55	2.16E-02	1.33
Rise velocity						
40	1.15E-01		1.23E-01		1.14E-01	
80	4.93E-02	1.23	4.87E-02	1.33	4.21E-02	1.43
160	1.63E-02	1.60	1.64E-02	1.57	1.77E-02	1.25
Circularity						
40	5.12E-02		1.45E-01		4.55E-01	
80	4.38E-02	0.22	1.46E-01	-0.01	4.61E-01	-0.02
160	4.30E-02	0.03	1.46E-01	0.00	4.62E-01	0.00

Table A.2: Relative error norms and convergence rates for testcase 1 of the Hysing Benchmark [15].

Conclusion

We can see that the scheme still does not reach its expected second order convergence. The only possibility I can think of for reaching its second order of convergence is setting the tolerances of the PISO scheme stricter.

$1/h$	NPU	NTS	execution Time in s
40	1	3000	170.03
80	1	6000	2010.42
160	4	12000	6221.44
320	24	24000	15357.59

Table A.3: Statistics from the simulation of the Hysing benchmark with a stationary time step, NPU is number of processors used

We did examine the difference between the different time stepping methods as well. On the finest grid there seemed to be almost no difference, however on the coarsest grid the difference was quite big. In figure A.1 we show the difference between this solutions. From this we can conclude that the maximum Courant number for the simulations with an adjustable time step was set too high. This is probably due to the transport equation for the volume fraction (equation 2.5). When looking at the courant numbers of this simulation we see a Courant number of the order $O(10^{-2})$ and an interface Courant number of the order $O(10^{-3})$, which is significantly lower than the number used in the ATS simulations.
Comparing the computation time is a bit harder due to the use of multiple processors for the finer grid. When looking at the two coarser grids we do however see that the computation time per time step is also slightly higher.

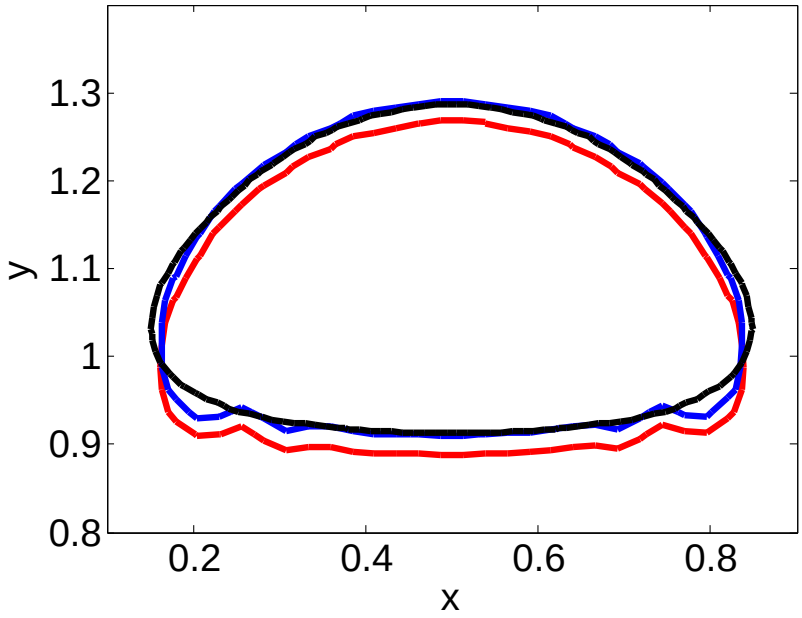


Figure A.1: The bubble at time $t = 3$ on the coarsest grid $\frac{1}{h} = 40$. Here red is the SDT, blue is the ATS and black is the reference solution of the benchmark [15].

Appendix B

CPU Data Speedup

Time	CPU VOF algorithm	CPU U equations	CPU first P equations	CPU second P equations	Total CPU per time step
2,94E-05	24,01	4,27	9,24	16,87	64,33
6,41E-05	24,83	4,48	10,69	16,52	57,51
0,000104768	23,38	4,24	9,25	21,68	59,47
0,000151885	23,49	3,95	5,82	15,34	49,57
0,000208426	23,62	4,01	4,97	12,04	45,6
0,000274391	24,52	3,96	4,54	13,23	47,24
0,000346952	23,86	4	4,76	12,76	46,31
0,000428583	23,97	4,32	5,68	12,29	47,28
0,000523819	24,19	4,18	5,41	13,46	48,23
0,000619055	24,59	4,2	4,75	12,69	47,29
0,000714291	24,6	4,15	4,79	14,15	48,66
0,000809528	24,93	4,22	4,7	14,33	49,21
0,000904764	24,24	4,07	4,39	13,01	46,74
0,001	23,58	4,32	4,75	12,91	46,8
Mean	24,13	4,17	5,98	14,38	50,30
Maximum	24,93	4,48	10,69	21,68	64,33
Minimum	23,38	3,95	4,39	12,04	45,6

Table B.1: Analysis of the speedup data with an adjustable time step for 16 processors.

Time	CPU VOF algorithm	CPU U equations	CPU first P equations	CPU second P equations	Total CPU per time step
2,50E-05	21,92	4,25	9,78	16,08	62,23
5,00E-05	21,7	4,48	9,99	16,19	53,22
7,50E-05	21,49	4,38	9,89	20,27	56,89
0,0001	21,75	4,01	5,5	17,31	49,46
0,000125	21,52	3,81	5,17	11,46	42,86
0,00015	21,81	3,98	4,78	10,65	42,17
0,000175	22,68	4,1	4,73	8,98	41,35
0,0002	21,73	4,05	5	12,08	43,74
0,000225	21,52	3,93	5,38	10,89	42,62
0,00025	22,17	4,22	5,32	9,87	42,5
0,000275	21,68	4,01	4,86	11,33	42,79
0,0003	21,58	3,99	5,21	12,47	44,13
0,000325	22,06	4,51	4,92	11,39	43,82
0,00035	22,34	4,21	5,09	11,68	44,19
0,000375	22,61	4,1	5,17	10,66	43,47
0,0004	22,9	4,47	4,91	8,97	42,1
0,000425	22,34	4,57	5,34	10,2	43,39
0,00045	22,69	3,87	5,46	9,46	42,37
0,000475	22,33	4,82	5,4	10,77	44,19
0,0005	23,32	4,38	5,33	10,79	44,78
0,000525	21,55	4,47	5,36	12,35	44,62
0,00055	21,89	4,21	5,39	9,91	42,41
0,000575	26,91	6,22	21,89	68,02	124,11
0,0006	20,32	4,22	17,61	48,58	91,72
0,000625	19,75	4,7	18,37	62	105,62
0,00065	22,01	4,54	21,69	61,17	110,15
0,000675	21	4,36	18,75	61,45	106,5
0,0007	20,33	4,36	20,32	68,16	114,15
0,000725	21,25	4,01	18,44	71,69	116,18
0,00075	25,58	5,05	23,62	73,54	128,91
0,000775	24,69	5,23	25,16	76,35	132,38
0,0008	33,68	7,4	26,45	80,76	149,67
0,000825	29,29	5,54	24,47	86,72	147,33
0,00085	31,72	5,8	24,06	84,8	147,64
0,000875	29,98	5,77	24,23	82,09	143,35
0,0009	37,11	5,74	24,24	78,75	147,28
0,000925	29,5	8,5	27,52	76,61	143,62
0,00095	27,77	5,21	23,44	81,37	138,64
0,000975	27,49	5,22	22,6	79,34	135,72
0,001	31,02	6,74	24,27	80,81	144,48
Mean	24,12	4,79	13,38	39,65	83,17
Maximum	37,11	8,5	27,52	86,72	149,67
Minimum	19,75	3,81	4,73	8,97	41,35

Table B.2: Analysis of the speedup data with a stationary time step for 16 processors.

Appendix C

OpenFOAM® files

We added the files which were typically different for the case and could be usefull to reproduce the results.

0 directory

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.3.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    class       volScalarField;
    location    "0";
    object      alpha1;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    right
    {
        type            zeroGradient;
    }
    left
    {
        type            zeroGradient;
    }
    back
    {
        type            empty;
    }
    front
    {
        type            empty;
    }
```

```
    top
    {
        type            zeroGradient;
    }
    bottom
    {
        type            zeroGradient;
    }
}

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

constant directory

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.4                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;

    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      interfaceProperties;

}

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


sCells_alpha1 6;
sCells_alpha2 6;

sigma           sigma [1 0 -2 0 0 0 0] 0.24;


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

/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.4                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
```

```
FoamFile
{
    version         2.0;
    format          ascii;

    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;

}


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

phase1
{
    transportModel  Newtonian;
      nu              nu [0 2 -1 0 0 0 0] 0.05;
      rho             rho [1 -3 0 0 0 0 0] 1;
}

phase2
{
    transportModel  Newtonian;
    nu              nu [0 2 -1 0 0 0 0] 0.5;
    rho             rho [1 -3 0 0 0 0 0] 0.001;
}


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

system

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

ddtSchemes
{
    default         backward;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)                 Gauss linear;
```

```
}

divSchemes
{
        default      Gauss linear;
    div(phi,U)   Gauss vanLeer;
    div(nHf,gTv) Gauss vanLeer;

}

laplacianSchemes
{
    default         Gauss linear uncorrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         uncorrected;
}

fluxRequired
{
    default         no;
    p;
}

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

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

solvers
{

    p
    {
        solver          PCG;
        preconditioner
        {
            preconditioner  GAMG;
            tolerance       1e-03;
```

```
            relTol          0;
            smoother        DICGaussSeidel;
            nPreSweeps      0;
            nPostSweeps     2;
            nFinestSweeps   2;
            cacheAgglomeration true;
            nCellsInCoarsestLevel 100;
            agglomerator    faceAreaPair;
            mergeLevels     1;
        }
        tolerance       1e-6;
        relTol          0;
    }


    pFinal
    {
        $p;
        tolerance       1e-12;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-8;
        relTol          0;
    }
    UFinal
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-8;
        relTol          0;
    }



}


PIMPLE
{

    momentumPredictor yes;
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;

   pRefCell 0;
   pRefValue        0;


}


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

/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.2.0
```

```
|
|    \\  /    A nd           | Web:      www.OpenFOAM.org
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      initAlphaDictionary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

method single;

level 16;

single
{
        R 0.5;
        xc      1.8;
        yc      2;
        zc      -0.005;
}

array
{
}



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

Appendix D

Nomenclature

Throughout this paper many symbols have been used. We tried to mention them here in certain sub-categories. When one of these symbols is bold in the text, it signifies that this symbol is a vector, tensor or matrix. Capital letters are mostly used to signify the non dimensional parameter of that quantity, however could also be a typical scale.

Field variables		
$\boldsymbol{u}$	$(u\ v\ w)$	Velocity field
t		Time
ρ		Density
p		Static pressure
$\hat{p}$	$p - \rho \boldsymbol{g} \cdot \boldsymbol{x}$	Modified pressure
μ		Dynamic viscosity
ν		Kinematic viscosity
$\boldsymbol{g}$		Gravity vector
σ		Surface tension
κ		Curvature
α	$\max_i \alpha_i$	Volume fraction field
$\boldsymbol{x}$	$(x,\ y,\ z)$	Location vector
Re	$\frac{U\ L}{\nu}$	Reynolds number, nondimensional parameter
Special conventions		
$\boldsymbol{D}$	$D_{i,j} = \frac{\partial_i u_j + \partial_j u_i}{2}$	Deformation tensor
$\boldsymbol{S}$	$\sigma \kappa \boldsymbol{n} \delta(n)$	Interfacial tension force
$\boldsymbol{n}$		Unit normal
$\delta(.)$		Dirac delta function
$\|\|e\|\|.$		Vector error
Ψ		Kernel operator for smoothening, details in [26]
$\boldsymbol{X}$	$\frac{\int_{\Omega_b} \boldsymbol{x} dx}{\int_{\Omega_b} dx}$	Center of Mass
γ	$\frac{2\pi r}{P_b}$	Circularity
$\boldsymbol{U}$	$\frac{\int_{\Omega_b} \boldsymbol{u} dx}{\int_{\Omega_b} dx}$	Rise velocity
ϵ		the fraction in the thin film
C	$\frac{\boldsymbol{u} \Delta t}{\Delta x}$	Maximum Courant number
General		
r		Radius of a particle
n		number of gridcells
Ω		Domain
P		Perimeter
$O(.)$		Big O notation, signifying order
nm		Nano meter
$\mathbb{N}$		Set of Natural numbers
Abbreviations		
VOF		Volume of fluid
NTS		Number of time steps
ref		reference value
ROC		Rate of convergence

NPU	number of processors used
ATS	Adjustable Time Step
SDT	Stationary delta t
PISO	Pressure-Implicit-Split-Operator

super- and subscripts

$\cdot_b$	Bubble
$\cdot_p$	Particle
$\cdot_f$	Surrounding fluid
$\cdot_{nb}$	Neabouring grid cells
$\cdot_c$	Center of grid cell
$\cdot_{cf}$	Cell faces
$\cdot_x$	in the $x-$direction
$\cdot_y$	in the $y-$direction
$\cdot_z$	in the $z-$direction
$\cdot_1$	Belonging to particle 1
$\cdot_2$	Belonging to particle 2
$\dot{\cdot}^n$	Time step indicator
$\tilde{\cdot}$	Smoothened

List of Figures

List of Tables