

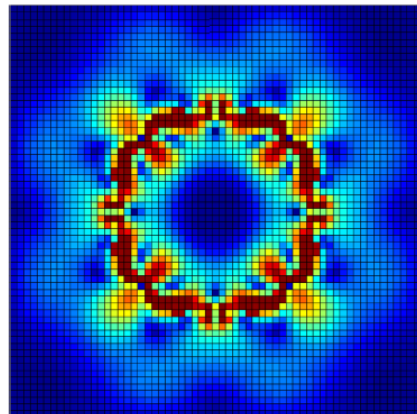
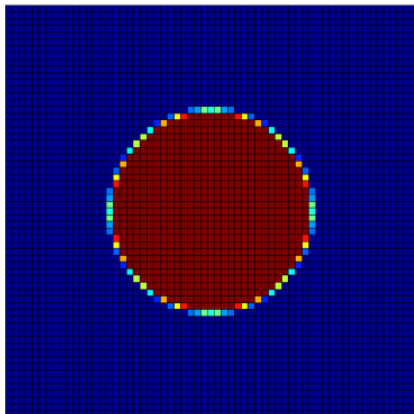
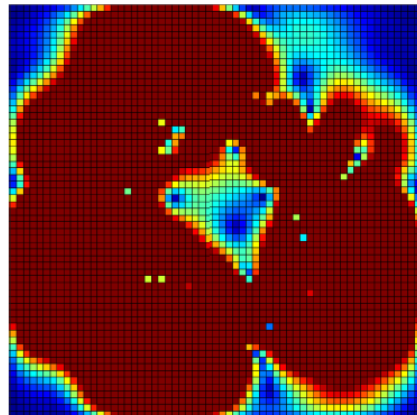
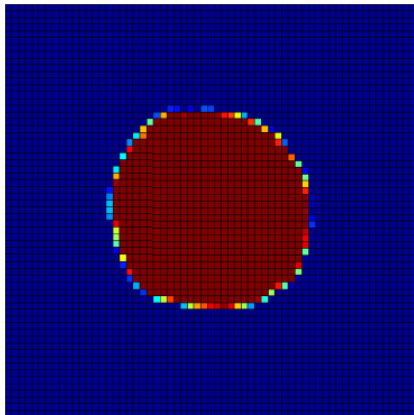


university of
 groningen

faculty of mathematics
 and natural sciences

A Numerical Surface Tension Model for Two-Phase Flow Simulations

K.W. Lam



Master Thesis in Applied Mathematics

August 2009

A Numerical Surface Tension Model for Two-Phase Flow Simulations

Summary

In two-phase flow simulations (for example air and water) the description of the fluid interface is important. For these simulations we have used a surface tension model. From literature we know that the approximation of the surface curvature is important, for badly approximated surface curvatures will lead to spurious (unphysical) velocities.

To track the surface interface we use a Volume of Fluid (VoF) method. Some researchers (Brackbill et al, Williams et al) say that this method will not work properly, so they modified it.

By looking at the structure of our model, we thought that VoF would also work unmodified. We have tested our methods on a stationary bubble case, because then spurious currents are best seen. The goal is to reduce these spurious currents, because they should not occur. During the talk we will compare results of our methods with results of the method that is being used in ComFlo.

Master Thesis in Applied Mathematics

Author: K.W. Lam

Supervisor(s): A.E.P. Veldman and R. Luppens

Date: August 2009

Institute of Mathematics and Computing Science

P.O. Box 407

9700 AK Groningen

The Netherlands

*I would like to dedicate this thesis to my parents:
To my mother, Cheung Pik Lin, who always is there for us.
And to the memories of my father, Lam Tung Sing. 爸爸, 我对你不起*

Contents

1	Introduction	1
2	The model	3
3	Literature review	5
3.1	A continuum method for modeling surface tension	5
3.2	Accuracy and convergence of continuum surface tension models	6
3.3	A novel technique for including surface tension in PLIC-VOF methods	8
3.4	A front-tracking algorithm for accurate representation of surface tension	10
3.5	Method used at RuG (program: ComFlo)	11
3.6	Height functions for applying contact angles to 3D VOF simulations	12
3.7	Summary of methods	12
4	Discretization	15
4.1	The delta function $\delta_\Gamma = \nabla C $	15
4.2	Discretization of κ in 5 cells	15
4.3	Discretization of κ in > 5 cells	16
4.3.1	9 cells discretization	16
4.3.2	25 cells discretization based on Shirani <i>et al.</i>	18
4.4	3D discretizations	20
4.4.1	Discretization of κ in 27 cells (3D)	20
4.4.2	3D discretization based on Shirani <i>et al.</i> (125 cells)	23
4.5	Free-surface displacement	27
5	Results	31
5.1	Simulation done with the delta function $4C(1 - C)$	31
5.2	5 cells vs. 9 cells discretization	33
5.3	Comparison with the height function method	35
5.3.1	The height function method vs. 9 cells discretization method	35
5.3.2	The height function method vs. method based on Shirani's unit normal	39
5.4	Results of 3D methods	43
6	Conclusion	47

Chapter 1

Introduction

Two-phase flow problems, for example of a liquid and a gas, separated by a moving, deforming interface, are of interest in many research fields. Ranging from environmental sciences to nuclear industries (power plants). It is desirable for researchers to be able to predict the behaviour. To get more knowledge of such flows, they simulate these flows.

In two-phase flow simulations, approximating the surface tension force term accurately is important, for not approximating this term properly will lead to spurious (non-physical) velocities. These spurious velocities are caused by the way the surface tension force is discretized and the way the surface curvature κ is approximated [11]. Researchers all over the world are trying to reduce these spurious velocities. Especially the curvature part is the subject of study by many.

To locate and track the surface interface the *Volume of Fluid method* (VoF) is widely used, because of its ability to conserve mass, even when the topology changes. (As in joining or breaking up of the fluids.)

Since the curvature $\kappa = \nabla \cdot \frac{\nabla C}{|\nabla C|}$, one might think that it can be approximated well by a finite difference discretization method (FDM), but VoF turns out to be too abrupt for a finite difference discretization. We can see that the values of the volume fractions function¹ in the cells are changing abruptly. To be able to use a FDM, Brackbill *et al.* [1] are creating a different volume fraction function by mollifying (smoothing) the abrupt one. The alternative volume fraction function is smooth enough, and Brackbill *et al.* are applying FDM to the mollified volume fraction function to approximate κ . This method to approximate κ is used by many researchers. Alternatively at the Groningen University the height function method is used to approximate κ [8].

Looking at the Navier-Stokes equations we wondered if it is really necessary to mollify first, and then use a FDM, since the pressure gradient shows the exact same abruptness. In this project we have looked at several finite difference discretizations, and we have compared these discretizations with the height function method. The discretizations will be given in chapter 4. What we do looks like smoothing, since we use more and more cells, but the main difference with Brackbill *et al.* [1] is that we do not change the volume fraction function, we use the original abrupt one.

First we will start with a literature review, where some other methods for approximating κ will pass by. Then we will give the discretizations of the methods that we have tried, and

¹There will be an explanation in the next chapter what volume fractions are.

we will compare the results of these methods with the height function method. To test these methods, we have made use of a stationary bubble test-case, for it is easy to determine the spurious currents. More about the stationary bubble we can see in chapter 5. At the end some conclusions will be given.

Chapter 2

The model

A two-phase flow is modelled with the Navier-Stokes equation

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \right) = -\nabla p + \nabla \cdot (\mu \mathbf{S}) + \mathbf{F} \quad (2.1)$$

where ρ is the density (which is assumed to be piecewise constant in this thesis, therefore $\nabla \cdot \mathbf{u} = 0$), μ the viscosity and \mathbf{S} the rate of strain tensor

$$\mathbf{S}_{ij} = \frac{1}{2} \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right)$$

and $\mathbf{F} = \sigma \kappa \mathbf{n} \delta_\Gamma + (\nabla \sigma) \delta_\Gamma$ is the surface tension force, with σ the surface tension coefficient and δ_Γ a delta function concentrated on the surface interface Γ . The surface tension coefficient σ is assumed to be constant, therefore the second term $(\nabla \sigma) \delta_\Gamma$ is neglected. So

$$\mathbf{F} = \sigma \kappa \mathbf{n} \delta_\Gamma, \quad (2.2)$$

where δ_Γ is a delta function. For δ_Γ there are many options.

To model a two-phase flow problem we need to track the interface. For this the **Volume of Fluid (VoF) method** is used. A short explanation about this method will be given now.

Volume of Fluid method

In a VoF-method for a two-phase system a color function c , defined as

$$c(x) = \begin{cases} 1 & \text{if } x \text{ lies in fluid 1,} \\ 0 < c(x) < 1 & \text{if } x \text{ lies in interface,} \\ 0 & \text{if } x \text{ lies in fluid 2} \end{cases}$$

is used. The volume fraction function C is defined as

$$C_i = \frac{\int_{\Omega_i} c(x) d\Omega_i}{|\Omega_i|}, \quad (2.3)$$

where Ω_i is the corresponding cell. In 2D simulations $|\Omega_i|$ is the area of the cell Ω_i , in 3D simulations $|\Omega_i|$ is the volume of Ω_i .

It then follows that a cell filled with fluid 1 has value $C = 1$, and a cell filled with fluid 2 has

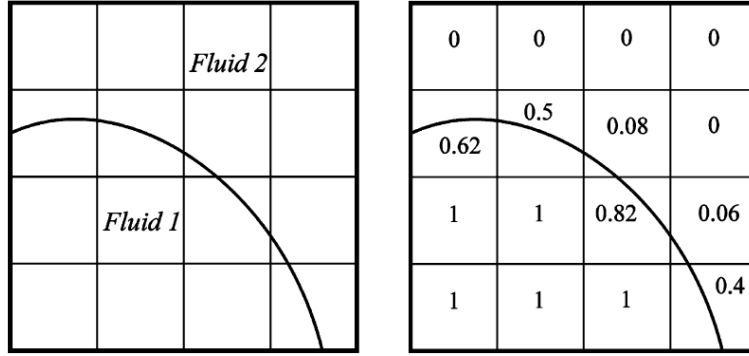


Figure 2.1: left: picture of the interface, right: corresponding volume fraction function

value $C = 0$. So when $0 < C < 1$ (fluid 1 and fluid 2 mixed) holds for a given cell, this is an interface cell. (See figure 2.1 for an example.)

In this study we have done simulations with two discrete formulations of the delta functions δ_Γ . We have also looked at how the curvature of the surface is determined numerically. For the curvature κ we use the following formulation

$$\kappa = \nabla \cdot \mathbf{n} \text{ with } \mathbf{n} = \frac{\nabla C}{|\nabla C|} \quad (2.4)$$

where C is the volume fraction function, and \mathbf{n} is the unit normal to the surface interface Γ . First a few methods of reseachers around the world are summarized in the next chapter, then the method that is being used at RuG will be explained.

Chapter 3

Literature review

As we have seen in the introduction, one of the reasons for the existence of spurious currents is the bad approximations of interface curvature. Therefore to reduce spurious currents, researchers around the world have tried to improve the approximations of the curvature of the interface. During a literature study we have studied several methods. Some of these methods are summarized in this chapter.

3.1 A continuum method for modeling surface tension

J. U. Brackbill, D. B. Kothe and C. Zemach

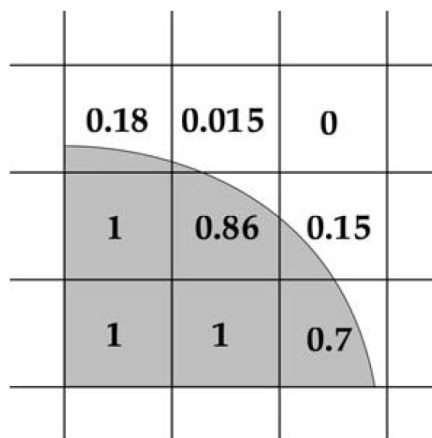


Figure 3.1: A surface interface with the corresponding volume fraction function.

In figure 3.1 we can see an example of a volume fraction function. We can see that it changes abruptly across the interface. Since we need the first and second derivative of the volume fraction function C to determine an approximation to \mathbf{n} and κ respectively, this abruptness of C might cause a problem.

Brackbill *et al.* proposed a solution to this problem by first convolving C with a smooth

kernel \mathbf{K} to construct a smoothed or *mollified* function \tilde{C} .

$$\tilde{C}(x) = \mathbf{K} * C(x) = \int_{\Omega_k} C(x') \mathbf{K}(x' - x) dx' \quad (3.1)$$

Now \mathbf{n} and κ follows from $\mathbf{n} = \frac{\nabla \tilde{C}}{|\nabla \tilde{C}|}$ and $\kappa = -\nabla \cdot \mathbf{n}$

The effect of a volume fraction function that is convolved with a smooth kernel \mathbf{K} is shown in figure 3.2. This example can be seen as a circular drop, $C = 1$ inside the drop (where the fluid is) and $C = 0$ outside the drop (for example air). The left part of the figure shows that near the interface C is too abrupt for standard finite differences. In the right part we see the mollified \tilde{C} . It is obvious that this one is much smoother than the left one, and because of this it is possible to apply a standard finite difference method to \tilde{C} [1].

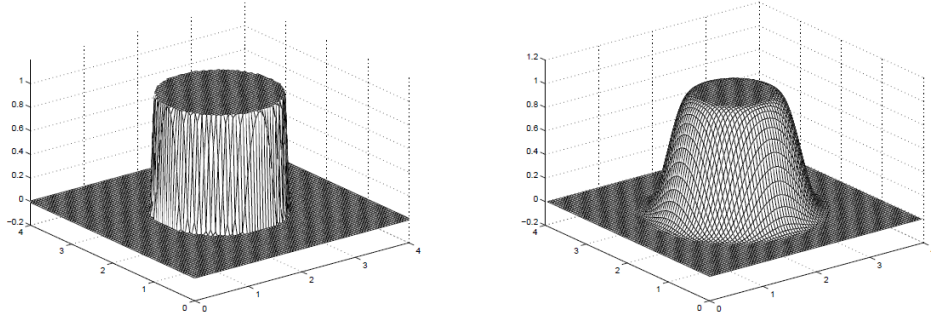


Figure 3.2: Here we can see the effect of the mollifying process. The right volume fraction function is the mollified version of the left one. One can see that the right one is a smooth volume fraction function.

For δ_Γ the authors uses $\delta_\Gamma = |\nabla \tilde{C}|$ (also mollified). To obtain the surface tension force one can substitute δ_Γ in (2.2).

3.2 Accuracy and convergence of continuum surface tension models

M.W. Williams, D.B. Kothe, E.G. Puckett

The authors of this article are also using the mollifying technique to smooth the volume fraction function. But instead of mollifying the volume fraction function, and using standard finite differences to approximate the first and second derivative (see Brackbill *et al.*), they have developed a hybrid method. Although in general the algorithm with a standard finite difference method is easy to implement, it is hard (and sometimes impossible) to show convergence under mesh refinement.

The hybrid method

Instead of using a finite difference method to approximate the first and second derivative,

3.2. ACCURACY AND CONVERGENCE OF CONTINUUM SURFACE TENSION MODELS 7

the authors make use of a property of the convolution product. Derivatives of $\tilde{C} = \mathbf{K} * C$ can be calculated by convolving C with the derivatives of the kernel \mathbf{K} , i.e.

$$\begin{aligned} \frac{\partial^n \tilde{C}(\mathbf{x})}{\partial x^n} &= \frac{\partial^n (\mathbf{K} * C)}{\partial x^n}(\mathbf{x}) = \left(\frac{\partial^n \mathbf{K}}{\partial x^n} * C \right)(\mathbf{x}) \\ &= \int_{\Omega_K} \frac{\partial^n \mathbf{K}}{\partial x^n}(\mathbf{x}' - \mathbf{x}) C(\mathbf{x}') d\mathbf{x}' \end{aligned} \quad (3.2)$$

To approximate the unit normal \mathbf{n} we can use (3.2) for $n = 1$ and for the curvature κ we can use $n = 2$.

There are some requirements for the kernel \mathbf{K} , to let this model work well. These requirements are

1. have compact support
2. be monotonically decreasing with respect to $r = |x|$
3. be radially-symmetric
4. be sufficiently smooth, i.e. for some $k \geq 3$, \mathbf{K} should be k times continuously differentiable
5. have a normality property, i.e. $\int_{\Omega_K} \mathbf{K}(x, \epsilon) dx = 1$
6. approach the Dirac δ function in the limit $|\Omega_K| \rightarrow 0$

where Ω_K is the support (Dutch: drager) of \mathbf{K} , and ϵ represents the size of the support of \mathbf{K} . So when \mathbf{K} is radially-symmetric, ϵ is equal to the radius of \mathbf{K} .

Requirements 1, 2 and 3 are desirable, but not necessary. For example, a kernel with the entire domain Ω as its support is possible, but since the surface tension term is a local problem this is not appropriate. Further requirement 2 is useful to prevent singularities to occur in kernel derivatives. Using radially-symmetric kernels will produce more uniformly accurate results.

Requirement 4 is necessary since then a property of the convolution product can be used (3.2). Requirement 6 is necessary to converge to the exact solution as $h \rightarrow 0$, where h is the meshwidth.

The following polynomial kernel \mathbf{K}_8 is introduced in this article

$$\mathbf{K}_8(r, \epsilon) = \begin{cases} A[1 - (r/\epsilon)^2]^4 & \text{if } r < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where A is a constant, such that $\int_{\Omega_K} \mathbf{K}_8(r, \epsilon) dr = 1$. For the δ_Γ in (2.2) they have chosen to use $\delta_\Gamma = 4\tilde{C}(1 - \tilde{C})$. (The same form that is used at the Groningen university, but these authors mollify the volume fraction function C).

3.3 A novel technique for including surface tension in PLIC-VOF methods

Markus Meier, George Yadigaroglu, Brian L. Smith

The authors of this article are approaching the curvature estimation in a different way than using discretized equations. They are looking for a function f_κ that gives the best estimation for the curvature.

$$k_{i,j} = f_\kappa(C_{i\pm 1,j\pm 1}, C_{i\pm 1,j\mp 1}, C_{i,j\pm 1}, C_{i\pm 1,j}, C_{i,j}) \quad (3.3)$$

where the $C_{a,b}$'s are volume fractions around a cell (i, j) . The coefficients of f_κ are determined only once and in advance, using a least-squares method against reference data.

Since a function of nine variables is not convenient, the authors have defined only three parameters for each cell (i, j) , which should contain all the information of the curvature.

$$\Sigma_{i,j} = \left\{ \sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} C_{i',j'} \right\} - A'_{i,j} \quad (3.4)$$

$$\Xi_{i,j} = C_{i,j} \quad (3.5)$$

$$\Omega_{i,j} = \arctan \left\{ \frac{\min(|n_x|, |n_y|)}{\max(|n_x|, |n_y|)} \right\} \quad (3.6)$$

The authors have constructed these parameters such that they strongly correlate with the curvature. The first parameter is the difference between the liquid content and the case where the interface is flat. The situation can be seen in the left picture of figure 3.3. The Ξ 's are just the volume fractions in the cell itself, and the Ω 's are the tilt angles of the interface-normal vector \mathbf{n} to the nearest horizontal or vertical.

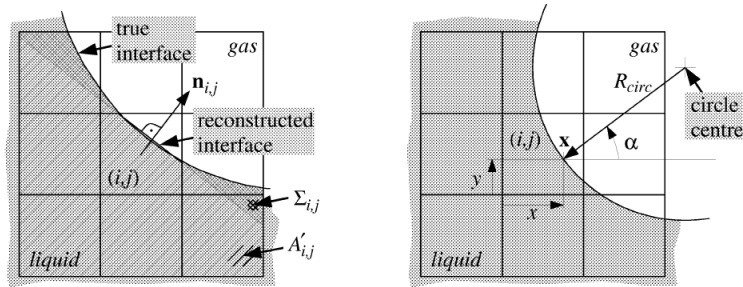


Figure 3.3: Here we can see the parameters that are described in (3.4), (3.5) and (3.6).

In figure 3.4 the authors have created situations where only one of the three parameters varies, and how this influences κ . The authors postulate that Σ , Ξ and Ω contain all the essential information for estimating the curvature, and that for any smooth interface with a radius of curvature greater than the mesh-width Δx , the curvature $\kappa_{i,j}$ can be estimated from these parameters with a unique function

$$\kappa = f_\kappa(\Sigma, \Xi, \Omega) \quad (3.7)$$

3.3. A NOVEL TECHNIQUE FOR INCLUDING SURFACE TENSION IN PLIC-VOF METHODS 9

For f_κ , they use a third-order polynomial

$$\kappa = \frac{1}{\Delta x} (c_1 + c_2 \Xi + c_3 \Omega + c_4 \Sigma + c_5 \Xi^2 + c_6 \Omega^2 + \dots + c_{20} \Xi \Omega \Sigma) \quad (3.8)$$

where the coefficients $c_1 \dots c_{20}$ will be determined only once by using a least square method against reference data. The determined function is then used for all flow simulations, which means that the computational expense is small.

The reference data is generated using a large set of calibration interfaces, for which Ξ , Ω , Σ and the true curvature κ_{true} are known. Circles are natural choices because their true curvature is $\kappa_{\text{true}} = 1/R_{\text{circ}}$, and the volume fractions and the parameters can be determined easily. A square grid is assumed so that the situation is geometrically identical for all cells. In figure 3.3 an example is given where cell (i, j) is surrounded by eight cells, and cell (i, j) is intersected by a circular interface. Now a point $\mathbf{x} = (x, y)$ on the circle, within cell (i, j) is chosen. So an angle α and a radius R_{circ} is uniquely defined. Now the four parameters x , y , α and κ_{true} are varied in discrete steps. In this way a large set of calibration interfaces will be obtained. The discrete steps they used are:

$$\begin{aligned} \frac{x}{\Delta x} &= 0.05, 0.15, \dots, 0.95 \\ \frac{y}{\Delta y} &= 0.05, 0.15, \dots, 0.95 \\ \alpha &= 0, 0.025\pi, \dots, 0.25\pi \\ \Delta x \cdot \kappa_{\text{true}} &= \pm 1, 15, \pm \frac{1}{2}, \pm \frac{1}{3}, \dots, \pm \frac{1}{10}, \pm 0.08, \pm 0.06, \dots, 0 \end{aligned} \quad (3.9)$$

In total there are 31900 calibration interfaces. For each calibration interface the volume fractions in each cell will be computed by a "pixel counting method", and then Σ , Ξ and Ω are computed from (3.4), (3.5) and (3.6).

Using standard least-squares fitting, the coefficients $c_1 - c_{20}$ are determined by minimizing $\sum (\kappa_{\text{true}} - \kappa)^2$, with κ from (3.8). So actually the estimator for $\kappa_{i,j}$ is obtained from (3.4), (3.5), (3.6), (3.8) and (3.9).

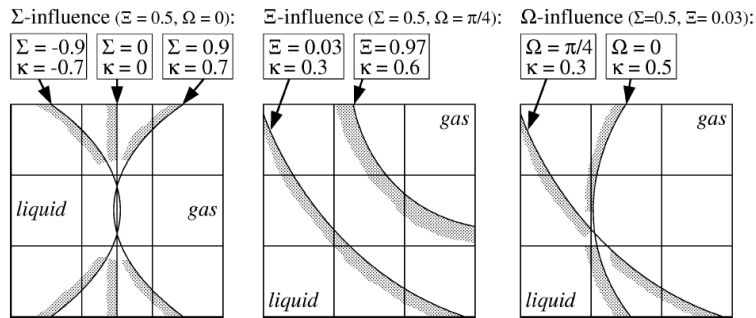


Figure 3.4: To see how κ is influenced by one parameter, the authors leave 2 of the 3 parameters unchanged.

3.4 A front-tracking algorithm for accurate representation of surface tension

Stéphane Popinet and Stéphane Zaleski

Another different approach for attacking the term with the curvature is proposed by these authors. They are using the conservative form of (2.1)

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot (2\mu \mathbf{D}) + \sigma \kappa \delta_{\Gamma} \mathbf{n} \quad (3.10)$$

The authors are using a momentum-conserving formulation of the Navier-Stokes equation. The pressure, volume fraction and velocity components are discretized on a uniform grid. Now the integral of the momentum equation is written as

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{u} dx = L(\mathbf{u}, \chi) - \oint_{\partial\Omega} p d\Gamma$$

where

$$L(\mathbf{u}, \chi) = - \oint_{\partial\Omega} \rho \mathbf{u} \otimes \mathbf{u} \cdot d\Gamma + \oint_{\partial\Omega} 2\mu \mathbf{D} \cdot d\Gamma + \int_{\Omega} \sigma \kappa \delta_{\Gamma} \mathbf{n} dx + \int_{\Omega} \rho g dx. \quad (3.11)$$

Here Ω is the control volume and $\partial\Omega$ is the boundary of the control volume. The article concentrates on the term $\int_{\Omega} \sigma \kappa \delta_{\Gamma} \mathbf{n} dx$.

Given a parametrization $(x(s), y(s))$ of the interface, where s is the arc length, the term due to the surface tension in the momentum equation can be calculated. The authors are using cubic splines to determine a parameterization of the interface. (Because they have a continuous first and second derivative). In figure 3.5 we see a situation where an interface segment AB is inside a control volume Ω . The term due to the surface tension can be written as

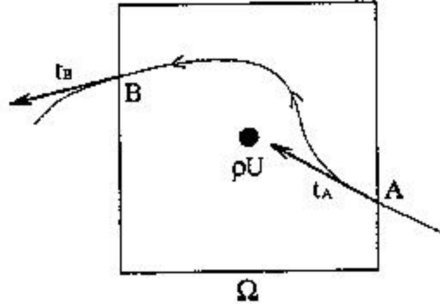
$$\int_{\Omega} \sigma \kappa \delta_{\Gamma} \mathbf{n} dx = \sigma \int_A^B \kappa \mathbf{n} ds.$$

Using the first Frenet-Serret equation for parametric curves this can be rewritten as

$$\sigma \int_A^B \kappa \mathbf{n} ds = \sigma \int_A^B \mathbf{t} dt = \sigma (\mathbf{t}_B - \mathbf{t}_A), \quad (3.12)$$

where \mathbf{t}_A and \mathbf{t}_B are the unit tangents to the curve in A and B .

¹the first Frenet-Serret equation is $(\nabla \cdot \mathbf{n})\mathbf{n} = \frac{dt}{ds} \Rightarrow \kappa \mathbf{n} ds = dt$

Figure 3.5: Control volume with unit tangents \mathbf{t}_a and \mathbf{t}_b .

So actually the authors are not approximating κ here, but they evaluate the surface tension term in (3.11), by means of (3.12).

3.5 Method used at RuG (program: ComFlo)

In ComFlo also a surface tension model is used. Hence (2.2) is also used in the code. The delta function δ_Γ is of the form $4C(1 - C)$, so the surface tension term has the form

$$\mathbf{F} = \sigma \kappa \mathbf{n} (4C(1 - C)) \quad (3.13)$$

For determining the curvature the height function method is used. This height function method is a technique that is developed at Groningen University. A short explanation will be given now.

The height function method

The height function method is based on the VoF method. A local height function is defined as the summation of volume fractions in the most dominant direction of the normal to the interface. For example, if we look at figure 3.6 the vertical direction is the dominant. This is because if we add the volume fractions in the left column, and the right column, and take the absolute value of the difference

$$\begin{aligned} f_{i-1} &= 1 + 0.97 + 0.06 = 2.03 \\ f_{i+1} &= 0.50 + 0.01 + 0 = 0.51 \\ \Delta f_{\text{horizontal}} &= |f_{i-1} - f_{i+1}| = 1.52 \end{aligned}$$

and we do the same for vertical direction

$$\begin{aligned} f_{j-1} &= 1 + 1 + 0.50 = 2.50 \\ f_{j+1} &= 0.06 + 0 + 0 = 0.06 \\ \Delta f_{\text{vertical}} &= |f_{j-1} - f_{j+1}| = 2.44 \end{aligned}$$

then we can see that $\Delta f_{\text{vertical}} > \Delta f_{\text{horizontal}}$.

In a 2D case where the vertical direction (y-direction) is the dominant normal direction, the three height functions are defined as

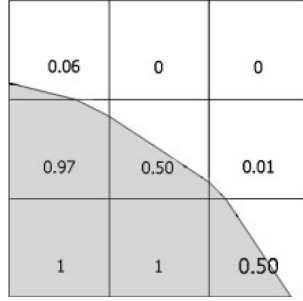


Figure 3.6: The vertical normal direction is dominant.

$$h_{i,j} = \sum_{j-1}^{j+1} f_{i,j} \Delta y_i \quad (3.14)$$

and at the center of cell (i, j) the curvature and normal are given by

$$\kappa = \frac{h_{xx}}{(1 + h_x^2)^{3/2}} \text{ and } \mathbf{n}(x, y) = \begin{bmatrix} h_x \\ -1 \end{bmatrix} \quad (3.15)$$

where h_x and h_{xx} are approximated with finite difference discretization

$$h_{xx} = \frac{h_{i+1,j} + 2h_{i,j} + h_{i-1,j}}{\Delta x_i^2} \text{ and } h_x = \frac{h_{i+1,j} - h_{i-1,j}}{2\Delta x_i} \quad (3.16)$$

and so (3.16) then can be substituted in (3.15).

3.6 Height functions for applying contact angles to 3D VOF simulations

S. Afkhami and M. Bussmann

These authors are also using a CSF model to discretize the surface tension force $F_{st} = \sigma \kappa \mathbf{n} \delta_\Gamma$. They approximate $\mathbf{n} \delta_\Gamma$ as ∇C . (So here $\delta_\Gamma = |\nabla C|$). The authors refer to Brackbill *et al.* for the discretization of F_{st} , which means that they mollify C . So they work with $F_{st} = \sigma \kappa \nabla \tilde{C}$. But for approximating the curvature κ they prefer the height function method to the mollifying method, because the height function method shows convergence with mesh refinement.

3.7 Summary of methods

In this chapter we have seen several methods, that researchers around the world have developed, to determine the curvature of a interface. Since the surface tension force $\mathbf{F} = \sigma \kappa \mathbf{n} \delta_\Gamma$ also depends on the choice of δ_Γ , the combinations of the researchers are put in a summary table to get a good overview.

authors	curvature κ	δ function
Brackbill <i>et al.</i> [1]	mollify with a smooth kernel	$ \nabla\tilde{C} $
Williams <i>et al.</i> [7]	hybrid method	$4\tilde{C}(1 - \tilde{C})$
Meier <i>et al.</i> [2]	using least square method to determine function	$\delta_{\Gamma}\mathbf{n} = \frac{\nabla\rho(\mathbf{x})}{(\rho_l - \rho_g)} \frac{2\rho(\mathbf{x})}{(\rho_l + \rho_g)}$
Popinet & Zaleski [4]	using momentum preserving formulation of NS (unit tangents)	1st Frenet-Serret eqn. used to evaluate the σ term
M. ten Caat <i>et al.</i> [8]	height function	$4C(1 - C)$
Afkhami & Bussmann [9]	height function	$ \nabla\tilde{C} $

To approximate $\mathbf{n} = \frac{\nabla C}{|\nabla C|}$ and $\kappa = \nabla \cdot \mathbf{n}$, where C is the volume fraction function, it is straightforward to first try a finite difference method, but it turns out that finite difference methods cannot handle the abruptness of the volume fraction function. To attack this problem, Brackbill *et al.* [1] mollify the volume fraction function with a smooth kernel \mathbf{K} to obtain a new function that is smooth enough for finite difference methods (see figure 3.2).

To approximate κ Williams *et al.* are also using the mollifying technique, but the main difference with Brackbill *et al.* [1] is that they make use of a property of the convolution product. Instead of using a finite difference method to approximate the first and second derivative of the mollified volume fraction function, they convolve the volume fraction function with the first and second derivative of the smooth kernel \mathbf{K} . This hybrid method will give better results.

Meier *et al.* [2] approach the curvature approximation in a different way. With a least square estimation they determine a function κ in terms of the surrounding volume fractions. (9 in 2D and 27 in 3D). Once this function is known they can fill in the volume fractions to get the local κ .

Popinet and Zaleski [4] are making use of a momentum preserving formulation of the Navier-Stokes equation. With the first Frenet-Serret equation they are able to rewrite the surface tension term in terms of unit tangents to the surface in certain points (see figure 3.5).

For approximation of κ Veldman *et al.* have developed the *height function method*. Unlike the finite difference methods the height function does converge with mesh-refinement. This method is used by many researchers, e.g. Afkhami and Bussmann [9] are using the height function to approximate κ .

During the project we thought that it would be interesting to try a discretization of $\mathbf{n} = \frac{\nabla C}{|\nabla C|}$, with unmollified C , anyway. The idea is that finite differences should be able to handle the abruptness of C , since the pressure gradient possesses the same abruptness. For δ_{Γ} , $|\nabla C|$ is chosen, which we will also leave unmollified. Our main reason is that it would be inter-

esting to be able to compare different choices of δ_Γ . But later we found out that choosing $\delta_\Gamma = 4C(1 - C)$ is not appropriate. This can be shown by looking at the units. If we look at (2.1) we see that $-\nabla p$ should have the same unit as $F = \sigma\kappa\mathbf{n}\delta_\Gamma$. Since

$$[p] = \frac{\text{N}}{\text{m}^2} \Rightarrow [\nabla p] = \frac{\text{N}}{\text{m}^3}$$

also $[F] = \frac{\text{N}}{\text{m}^3}$ must hold for $\delta_\Gamma = 4C(1 - C)$. But now we have $[\sigma] = \frac{\text{N}}{\text{m}}, [\kappa] = \frac{1}{\text{m}}, \mathbf{n} =$ unitless and so is $4C(1 - C)$, so we have $[F] = \frac{\text{N}}{\text{m}^2}$. Hence we can see that there is missing a factor $\frac{1}{\text{m}}$, when $\delta_\Gamma = 4C(1 - C)$ is used.

On the other hand if we look at $\delta_\Gamma = |\nabla C|$, we then have $F = \sigma\kappa\nabla C$ (see (4.1)). Since $[\nabla C] = \frac{1}{\text{m}}$ we are not missing the factor $\frac{1}{\text{m}}$. In chapter 4 we will see some results with both delta functions.

First we will look at the discretizations, which can be found in the next chapter.

Chapter 4

Discretization

In this chapter the delta function $\delta_\Gamma = |\nabla C|$, the unit normal $\mathbf{n} = \frac{\nabla C}{|\nabla C|}$ and the surface curvature $\kappa = \nabla \cdot \mathbf{n}$ will be discretized. This will be done with a finite difference method, both over 5 and 9 cells in 2D (and a 3D version of the 9 cells discretization over 27 cells). Also will we make use of the unit normal, that we have found in an article by Shirani *et al.* [6], to discretize $\kappa = \nabla \cdot \mathbf{n}$ in 2D (over 25 cells) and in 3D (over 125 cells).

4.1 The delta function $\delta_\Gamma = |\nabla C|$

First of all, we take a look at $\mathbf{F} = \sigma \kappa \mathbf{n} \delta_\Gamma$. Since $|\nabla C|$ is chosen as the delta function, we have $\mathbf{F} = \sigma \kappa \mathbf{n} |\nabla C|$. Since $\mathbf{n} = \frac{\nabla C}{|\nabla C|} \Rightarrow \nabla C = \mathbf{n} |\nabla C|$. Therefore we get

$$\mathbf{F} = \sigma \kappa \nabla C \quad (4.1)$$

The discretization of ∇C (in 2 dimensions) is straightforward

$$\begin{aligned} \text{in } i \text{ direction} \quad \nabla C_x &= \frac{C(i+1, j) - C(i, j)}{dx} \\ \text{in } j \text{ direction} \quad \nabla C_y &= \frac{C(i, j+1) - C(i, j)}{dy} \end{aligned}$$

Now we will proceed with the discretizations of κ .

4.2 Discretization of κ in 5 cells

The idea is to use standard finite difference discretization of $\mathbf{n} = \frac{-\nabla C}{|\nabla C|}$ and $\kappa = \nabla \cdot \mathbf{n}$ in (i, j) , $(i \pm 1, j)$ and $(i, j \pm 1)$ (see figure 4.1).

If we divide it in upper right, upper left, down right and down left we can approximate \mathbf{n} for each quadrant (divided by the broken lines, see figure 4.1). For example, for the upper right corner the cells (i, j) , $(i+1, j)$ and $(i, j+1)$ are used:

$$\begin{aligned} \text{in } (i+1/2, j) \quad n_{i+1/2, j} &= \frac{C(i+1, j) - C(i, j)}{dx |\nabla C|} \\ \text{in } (i, j+1/2) \quad n_{i, j+1/2} &= \frac{C(i, j+1) - C(i, j)}{dy |\nabla C|} \end{aligned}$$

where

$$\nabla C = \left(\frac{C(i+1, j) - C(i, j)}{dx}, \frac{C(i, j+1) - C(i, j)}{dy} \right)$$

If this is done for all four quadrants, $\kappa = \nabla \cdot \mathbf{n}$ in (i, j) can be approximated. The arrows in figure 4.1 and 4.2 indicate the place where ∇C is determined and the dots indicate the place for the normal \mathbf{n} .

Since $\kappa = \frac{\partial}{\partial x} n_x + \frac{\partial}{\partial y} n_y$ we get

$$\kappa = \frac{n_{i+1/2, j} - n_{i-1/2, j}}{dx} + \frac{n_{i, j+1/2} - n_{i, j-1/2}}{dy} \quad (4.2)$$

And this is the approximation of curvature κ in (i, j) .

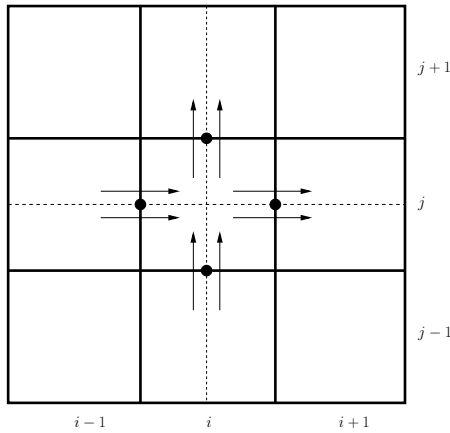


Figure 4.1: 5 cells discretization

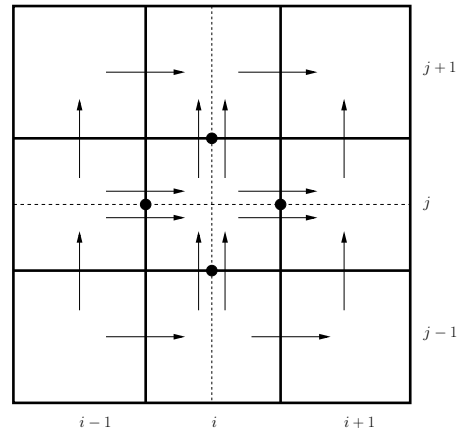


Figure 4.2: 9 cells discretization

4.3 Discretization of κ in > 5 cells

4.3.1 9 cells discretization

From the figures in the next chapter we will see that the results with a discretization over 5 cells are not very good. A reason might be that there is not enough information to be accurate. If we look at the left situation in figure 4.3, we see a situation with 3 full cells and 3 surface cells. In the right situation in figure 4.3, we see the cells that are being used, with a 5 cells discretization, to approximate the situation on the left. In figure 4.4 we see another example.

Therefore a discretization over 9 cells is tried (now there is extra information). The idea is to use (i, j) , $(i+1, j)$ and $(i, j+1)$ with $(i \pm 1, j \pm 1)$ and $(i \mp 1, j \pm 1)$ added. The same four quadrants will be used. (see figure 4.2). Then again (4.2) can be used to get κ .

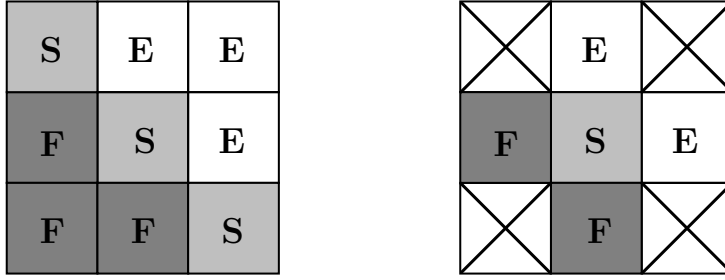


Figure 4.3: on the left we see the real situation, on the right we see the cells that are used in a 5 cells discretization. The crossed cells are not used.

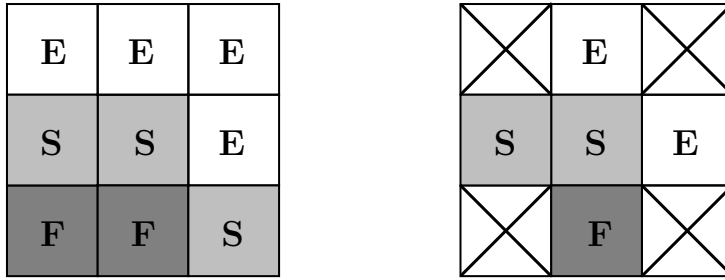


Figure 4.4: on the left we see the real situation, on the right we see the cells that are used in a 5 cells discretization. The crossed cells are not used.

For example, for the upper right corner now the cells (i, j) , $(i + 1, j)$, $(i, j + 1)$ and $(i + 1, j + 1)$ are used. (see figure 4.2.)

horizontal

$$\begin{aligned} \text{in } (i + 1/2, j + 1) \quad \nabla C_U &= \frac{C(i + 1, j + 1) - C(i, j + 1)}{dx} \\ \text{in } (i + 1/2, j) \quad \nabla C_D &= \frac{C(i + 1, j) - C(i, j)}{dy} \\ \text{in } (i + 1/2, j + 1/2) \quad \nabla C_{av} &= \frac{1}{2}(\nabla C_U + \nabla C_D) \end{aligned}$$

vertical

$$\begin{aligned} \text{in } (i, j + 1/2) \quad \nabla C_L &= \frac{C(i, j + 1) - C(i, j)}{dx} \\ \text{in } (i + 1, j + 1/2) \quad \nabla C_R &= \frac{C(i + 1, j + 1) - C(i + 1, j)}{dy} \\ \text{in } (i + 1/2, j + 1/2) \quad \nabla C_{av} &= \frac{1}{2}(\nabla C_L + \nabla C_R) \end{aligned}$$

from this we get $\nabla C = (\frac{1}{2}(\nabla C_U + \nabla C_D), \frac{1}{2}(\nabla C_L + \nabla C_R))$. This is for the upper right quadrant. If we do the same for the remaining three quadrants we can use the same formula (4.2) as in the previous section to approximate the curvature κ in (i, j) .

4.3.2 25 cells discretization based on Shirani *et al.*

This discretization is based on the way Shirani *et al.* [6] discretize the unit normal $\mathbf{n} = -\frac{\nabla C}{|\nabla C|}$.

The following equations are from their article

$$\nabla C_x = (C_{i+1,j+1} - C_{i-1,j+1} + 2(C_{i+1,j} - C_{i-1,j}) + C_{i+1,j-1} - C_{i-1,j-1})/\Delta x$$

$$\nabla C_y = (C_{i+1,j+1} - C_{i+1,j-1} + 2(C_{i,j+1} - C_{i,j-1}) + C_{i-1,j+1} - C_{i-1,j-1})/\Delta y$$

so the x and y components of the unit normal vector are

$$\mathbf{n}_x = -\frac{\nabla C_x}{\sqrt{(\nabla C_x)^2 + (\nabla C_y)^2}} \quad (4.3)$$

$$\mathbf{n}_y = -\frac{\nabla C_y}{\sqrt{(\nabla C_x)^2 + (\nabla C_y)^2}} \quad (4.4)$$

In their article they only use this for approximating the unit normal. We thought that it would be nice to see how it works in a discretization for approximating the curvature as well, since $\kappa = \nabla \cdot \mathbf{n}$.

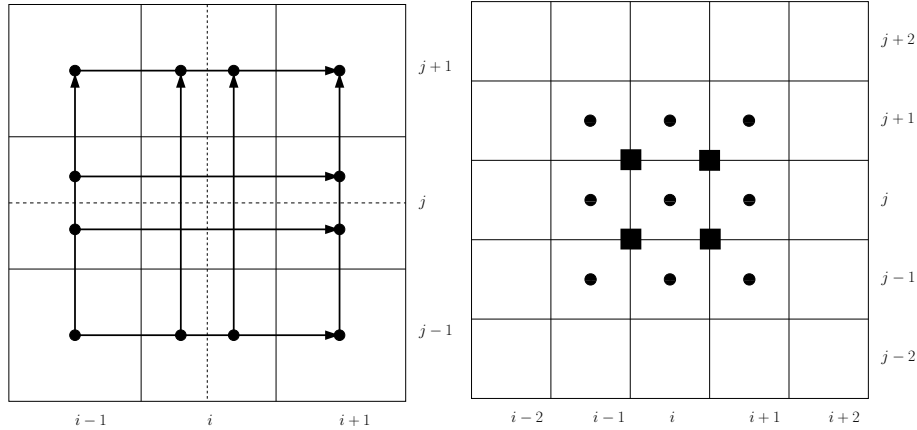


Figure 4.5: The cells (dotted) used for approximating \mathbf{n} in (i, j) , where \mathbf{n} is the unit normal to the interface. Figure 4.6: \mathbf{n} is determined in dotted cells. The means are taken in the squares.

The idea can be seen in figure 4.6. In the 9 dotted cells the unit normal is determined. So actually 25 cells are used in this discretization. Also we see 4 squares, where each square lies in 4 cells (figure 4.7). In such a square the mean value of the unit normal is taken from the 4 cells where it lies in.

For example, for the upper right corner (in figure 4.7) it will be

$$\mathbf{n}_{ur} = \frac{1}{4}(\mathbf{n}_{i,j} + \mathbf{n}_{i,j+1} + \mathbf{n}_{i+1,j} + \mathbf{n}_{i+1,j+1})$$

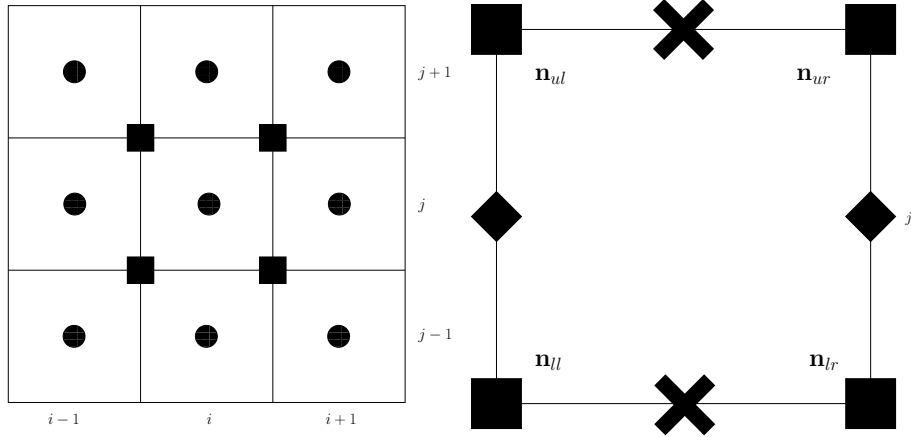


Figure 4.7: Each square is the mean of 4 corresponding dots.

Figure 4.8: The location of the squares, diamonds and crosses.

In figure 4.8, cell (i, j) is zoomed in. In the crosses the mean of the corresponding j -components of the normals are taken, and in the diamonds the mean of the corresponding i -components. To determine $\kappa = \nabla \cdot \mathbf{n}$ we can use (4.2) again.

So we get

$$\kappa = \frac{(\mathbf{n}_{ur} + \mathbf{n}_{lr})_x - (\mathbf{n}_{ul} + \mathbf{n}_{ll})_x}{2dx} + \frac{(\mathbf{n}_{ul} + \mathbf{n}_{ur})_y - (\mathbf{n}_{ll} + \mathbf{n}_{lr})_y}{2dy}$$

where the subscripts x and y indicate the x and y component of the averaged unit normals. The location of the averaged unit normals can be found in figure 4.8.

4.4 3D discretizations

In this section 3D versions of previous discretizations will be given. We first will start with the extension of the 9-cells discretization (to 27 cells), and then the extension of the 25 cells discretization (Shirani *et al.*), to 125 cells.

4.4.1 Discretization of κ in 27 cells (3D)

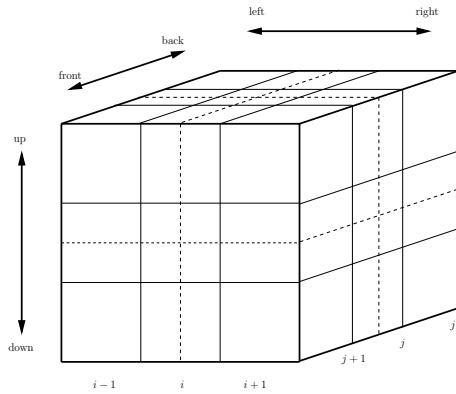


Figure 4.9: A 3D cube.

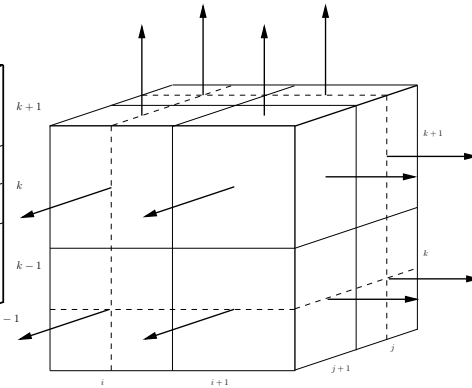


Figure 4.10: Right upper front octant.

The discretizations in the previous sections are for 2 dimensions. In this section the discretization will be extended to 3 dimensions. It is based on the same idea as for the 9 cells discretization. This discretization is done over 27 cells (see figure 4.9). Like in the 2 dimensional case where it was divided in quadrants, it will now be divided into octants (done with the broken lines in figure 4.9).

These octants are

$$\{\text{left, right}\} \times \{\text{down, up}\} \times \{\text{back, front}\}$$

To show the idea of the discretization, discretization of octant $\{\text{right, upper, front}\}$ will be described in detail now. (see figure 4.10). As we can see in the figure, for an octant we can determine a direction component (i, j or k) of ∇C by first determining the "arrows" in a direction and then take the average of the "arrows".

In the discretization we will see subscripts. For example ∇C_{uf} , where uf stands for "upper front". If we look in figure 4.10 the involved cells are $C(i+1, j+1, k+1)$ and $C(i, j+1, k+1)$, since they are in the upperlayer when seen from the k -direction, and in the front layer when seen from the j -direction.

Now we will go to the discretization.

In i -direction

Here we calculate the "arrows" in i -direction

$$\begin{aligned}
\nabla C_{uf} &= \frac{C(i+1, j+1, k+1) - C(i, j+1, k+1)}{dx} \\
\nabla C_{ub} &= \frac{C(i+1, j, k+1) - C(i, j, k+1)}{dx} \\
\nabla C_{df} &= \frac{C(i+1, j+1, k) - C(i, j+1, k)}{dx} \\
\nabla C_{db} &= \frac{C(i+1, j, k) - C(i, j, k)}{dx}
\end{aligned}$$

Here we take the average of the arrows to obtain the average value of ∇C in i -direction. The same is also done for j and k direction.

$$\nabla C_x = (\nabla C_{uf} + \nabla C_{ub} + \nabla C_{df} + \nabla C_{db})/4 \quad (4.5)$$

In j -direction

$$\begin{aligned}
\nabla C_{ru} &= \frac{C(i+1, j+1, k+1) - C(i+1, j, k+1)}{dy} \\
\nabla C_{lu} &= \frac{C(i, j+1, k+1) - C(i, j, k+1)}{dy} \\
\nabla C_{rd} &= \frac{C(i+1, j+1, k) - C(i+1, j, k)}{dy} \\
\nabla C_{ld} &= \frac{C(i, j+1, k) - C(i, j, k)}{dy}
\end{aligned}$$

$$\nabla C_y = (\nabla C_{ru} + \nabla C_{lu} + \nabla C_{rd} + \nabla C_{ld})/4 \quad (4.6)$$

In k -direction

$$\begin{aligned}
 \nabla C_{rf} &= \frac{C(i+1, j+1, k+1) - C(i+1, j+1, k)}{dz} \\
 \nabla C_{lf} &= \frac{C(i, j+1, k+1) - C(i, j+1, k)}{dz} \\
 \nabla C_{rb} &= \frac{C(i+1, j, k+1) - C(i+1, j, k)}{dz} \\
 \nabla C_{lb} &= \frac{C(i, j, k+1) - C(i, j, k)}{dz} \\
 \nabla C_z &= (\nabla C_{rf} + \nabla C_{lf} + \nabla C_{rb} + \nabla C_{lb})/4
 \end{aligned} \tag{4.7}$$

So with (4.5),(4.6) and (4.7) we get

$$\nabla C_{ruf} = (\nabla C_x, \nabla C_y, \nabla C_z) \Rightarrow |\nabla C_{ruf}| = \sqrt{(\nabla C_x)^2 + (\nabla C_y)^2 + (\nabla C_z)^2}$$

And for the unit normal in octant {right, upper, front} we get

$$\mathbf{n}_{ruf} = \left(\frac{\nabla C_x}{|\nabla C_{ruf}|}, \frac{\nabla C_y}{|\nabla C_{ruf}|}, \frac{\nabla C_z}{|\nabla C_{ruf}|} \right) \tag{4.8}$$

In the same way we can determine the direction of the normals for the other 7 octants. For every octant we then have normal component in i, j and k direction. If we look at the components in i -direction (see figure 4.11) we can determine the average of these components. (The big arrows in the figure).

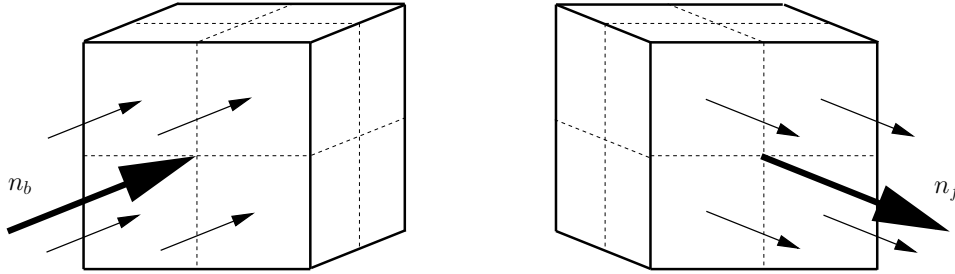


Figure 4.11: The 8 normal components in i -direction, where the big arrows are the average of 4 corresponding small ones.

When this is also done for j and k direction we can use

$$\kappa = \frac{n_f - n_b}{dx} + \frac{n_r - n_l}{dy} + \frac{n_u - n_d}{dz} \tag{4.9}$$

to obtain κ .

4.4.2 3D discretization based on Shirani *et al.* (125 cells)

Here the discretization based on Shirani's unit normal¹ will be extended to 3 dimensions. Only the i -direction will be done in detail here, the directions j and k can be done in exactly the same way.

Since the unit normal is $\mathbf{n} = \frac{\nabla C}{|\nabla C|}$ we first must approximate ∇C . For the i -direction we have to discretize in i, j -plane and i, k -plane using (4.3). (see figure 4.12 and 4.13.)

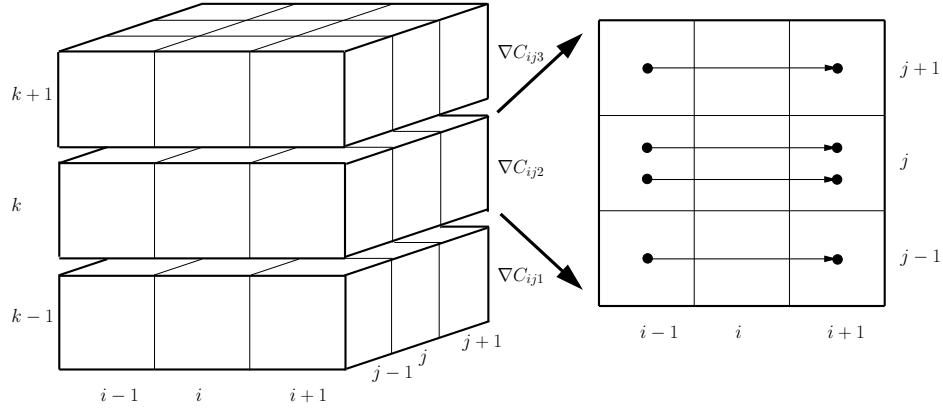


Figure 4.12: Discretizing in i, j -planes.

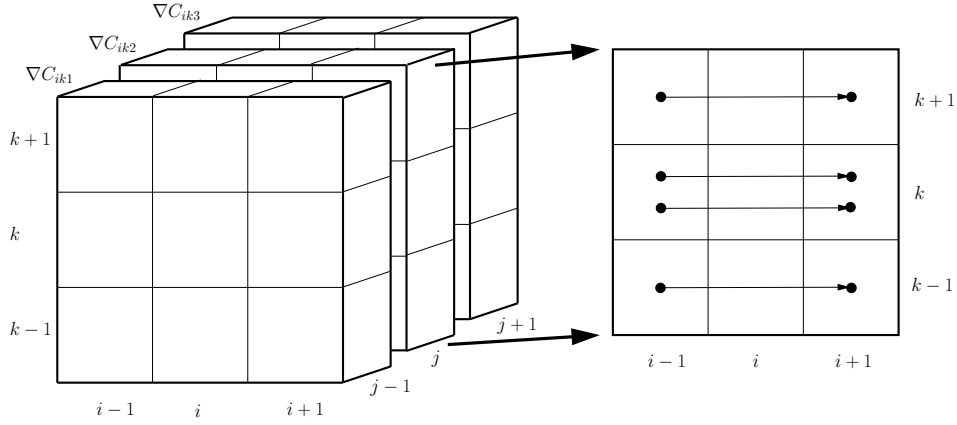
Now for every plane in figure 4.12, ∇C will be approximated.

$$\begin{aligned}\nabla C_{ij1} &= (C(i+1, j-1, k-1) - C(i-1, j-1, k-1) + 2(C(i+1, j, k-1) \\ &\quad - C(i-1, j, k-1)) + C(i+1, j+1, k-1) - C(i-1, j+1, k-1))/\Delta x \\ \nabla C_{ij2} &= (C(i+1, j-1, k) - C(i-1, j-1, k) + 2(C(i+1, j, k) \\ &\quad - C(i-1, j, k)) + C(i+1, j+1, k) - C(i-1, j+1, k))/\Delta x \\ \nabla C_{ij3} &= (C(i+1, j-1, k+1) - C(i-1, j-1, k+1) + 2(C(i+1, j, k+1) \\ &\quad - C(i-1, j, k+1)) + C(i+1, j+1, k+1) - C(i-1, j+1, k+1))/\Delta x\end{aligned}$$

The mean is

$$\nabla C_{ij} = \nabla C_{ij1} + \nabla C_{ij2} + \nabla C_{ij3} \quad (4.10)$$

¹Throughout this thesis, when 'Shirani's unit normal' is mentioned, the unit normal that Shirani *et al.* [6] are using is meant.

Figure 4.13: Discretizing in i, k -planes.

Now the same will be done for the planes in figure 4.13

$$\begin{aligned}\nabla C_{ik1} &= (C(i+1, j-1, k-1) - C(i-1, j-1, k-1) + 2(C(i+1, j-1, k) \\ &\quad - C(i-1, j-1, k)) + C(i+1, j-1, k+1) - C(i-1, j-1, k+1))/\Delta x \\ \nabla C_{ik2} &= (C(i+1, j, k-1) - C(i-1, j, k-1) + 2(C(i+1, j, k) \\ &\quad - C(i-1, j, k)) + C(i+1, j, k+1) - C(i-1, j, k+1))/\Delta x \\ \nabla C_{ik3} &= (C(i+1, j+1, k-1) - C(i-1, j+1, k-1) + 2(C(i+1, j+1, k) \\ &\quad - C(i-1, j+1, k)) + C(i+1, j+1, k+1) - C(i-1, j+1, k+1))/\Delta x\end{aligned}$$

So the mean is.

$$\nabla C_{ik} = \nabla C_{ik1} + \nabla C_{ik2} + \nabla C_{ik3} \quad (4.11)$$

Now the i -component of ∇C is the sum of the 2 means

$$\nabla C_i = \nabla C_{ij} + \nabla C_{ik} \quad (4.12)$$

In the same way ∇C_j and ∇C_k can be obtained, and with $\nabla C = (\nabla C_i, \nabla C_j, \nabla C_k)$ known, the unit normal \mathbf{n} can be calculated.

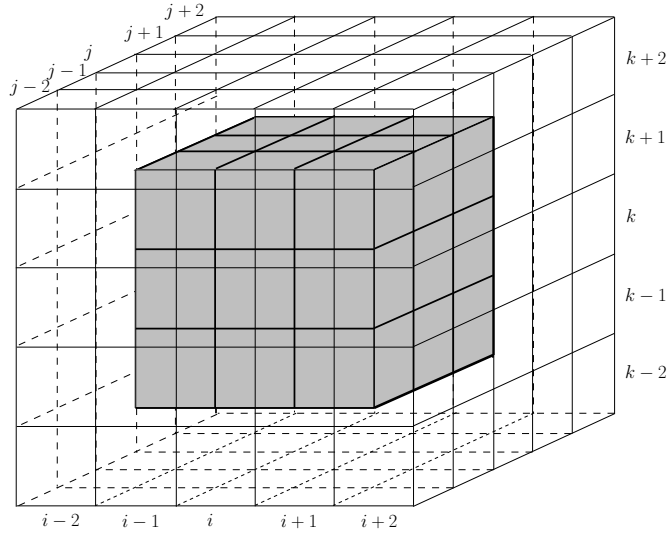


Figure 4.14: The location of a 3×3 cube (the dark one) in a 5×5 cube.

Now the divergence of the unit normal \mathbf{n} will be determined, since $\kappa = \nabla \cdot \mathbf{n}$. To determine κ in (i, j, k) , all cells in figure 4.14 will be used. In the 5^3 cube there is a dark 3^3 cube (see also figure 4.15a).

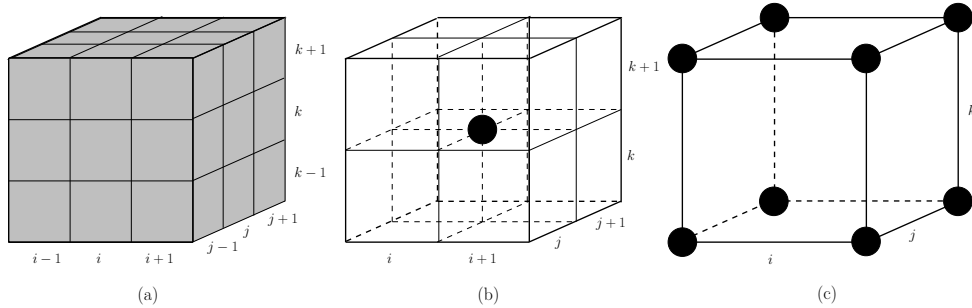


Figure 4.15: Different zoom-ins of the cube.

First in all the cells in figure 4.15a the unit normal will be determined. So 27 unit normals will be approximated. Then 8 means of the normals will be taken (in 8 corners of the cube in figure 4.15c). In figure 4.15b we see the corner

$$\{i, i + 1\} \times \{j, j + 1\} \times \{k, k + 1\}$$

In the big black dot in figure 4.15b the mean is located. In figure 4.15c we can see that the 8 dots are in the 8 corners of cell (i, j, k) . And with the 8 mean unit normals from figure 4.15c known we can determine the curvature with

$$\kappa = \frac{\partial}{\partial x} \mathbf{n}_x + \frac{\partial}{\partial y} \mathbf{n}_y + \frac{\partial}{\partial z} \mathbf{n}_z \tag{4.13}$$

For a direction we use the corresponding component of the unit normals. For the ordering of the dots see figure 4.16.

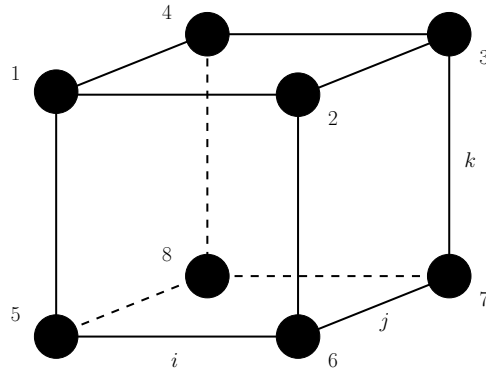


Figure 4.16: The ordering of the dots.

So we get:

$$\frac{\partial}{\partial x} \mathbf{n}_x = \mathbf{n}_{x2} + \mathbf{n}_{x3} + \mathbf{n}_{x6} + \mathbf{n}_{x7} - \mathbf{n}_{x1} - \mathbf{n}_{x4} - \mathbf{n}_{x5} - \mathbf{n}_{x8} \quad (4.14)$$

$$\frac{\partial}{\partial y} \mathbf{n}_y = \mathbf{n}_{y1} + \mathbf{n}_{y2} + \mathbf{n}_{y5} + \mathbf{n}_{y6} - \mathbf{n}_{y3} - \mathbf{n}_{y4} - \mathbf{n}_{y7} - \mathbf{n}_{y8} \quad (4.15)$$

$$\frac{\partial}{\partial z} \mathbf{n}_z = \mathbf{n}_{z1} + \mathbf{n}_{z2} + \mathbf{n}_{z3} + \mathbf{n}_{z4} - \mathbf{n}_{z5} - \mathbf{n}_{z6} - \mathbf{n}_{z7} - \mathbf{n}_{z8} \quad (4.16)$$

and to get κ we substitute (4.14), (4.15) and (4.16) into (4.13).

4.5 Free-surface displacement

What also turned out to be important in our simulations is the free-surface displacement algorithm. In the ComFlo code there is an adapted version of the VOF method first introduced by Hirt and Nichols [12]. A piecewise constant reconstruction of the free-surface is used, where in 2D, the free surface is displaced by changing the VOF value in a cell using calculated fluxes through cell faces.

$$\delta F^s = \mathbf{u} \cdot \mathbf{n} A \delta t$$

Here \mathbf{u} is the velocity of the fluid, \mathbf{n} is the unit normal to the free-surface, A the area and δt the time step.

When all fluxes have been calculated, the VOF function is updated from time level n to $n+1$ using

$$(F^s)^{n+1} = (F^s)^n - \frac{\delta F_e^s + \delta F_n^s - \delta F_w^s - \delta F_s^s}{\delta x \delta y}$$

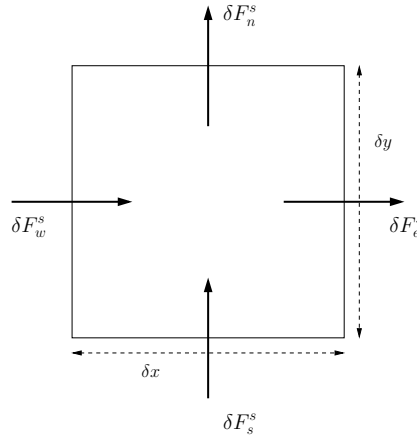


Figure 4.17: The fluxes.

The original VOF method has two main drawbacks. The first is that flotsam and jetsam can appear, which are small droplets disconnecting from the free surface. The second drawback is the gain or loss of fluid due to rounding the VOF function when $F^s > 1$ or $F^s < 0$.

By combining the VOF method with a local height function, these problems do not appear anymore. The local height function is used in the following way. For every surface cell a local height function is defined in the most normal direction. After calculating the fluxes across the cell boundaries of the three cells (as in the original VOF method), not the volume fractions of the cells in the three columns are updated but the local height function. Then the volume fractions of the cells are calculated from the height of the fluid column.

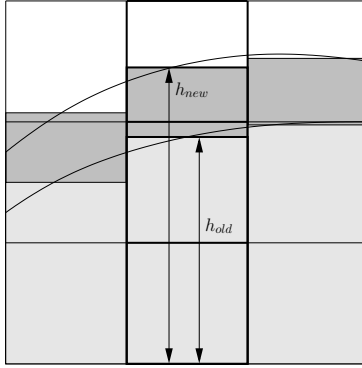


Figure 4.18: The old height function (h_{old}) with the updated one (h_{new}).

With this adapted method, the two main drawbacks in the original method do not occur anymore. But as we will see, with this adapted method, there is now loss of symmetry in the velocity field, see figure 4.19. With this loss of symmetry in the velocity field, the bubble is not expected to stay symmetric. To avoid this problem R. Luppens came with his displacement algorithm VFLUP. As we can see in figure 4.20 the velocity field is symmetric now.

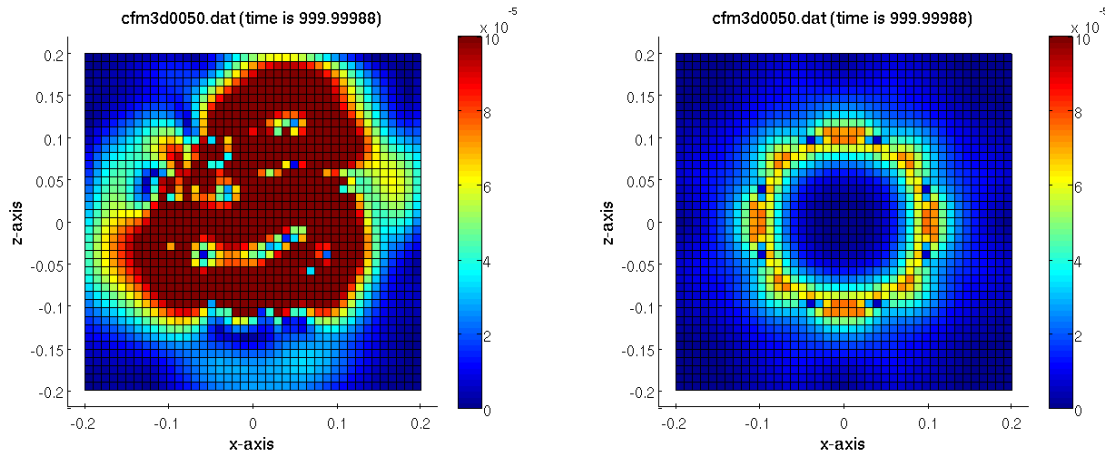


Figure 4.19: Velocity field with displacement algorithm VFCONV. Figure 4.20: Velocity field with displacement algorithm VFLUP.

The main difference between the new displacement method VFLUP and the original 'RUG-method' is the symmetric approach in the correction phase, after the free surface advection. In both methods, first the free surface is advected, as described above. When the new FS values are computed, the FS-field is subsequently corrected, based on new local height functions. In the original method, the corrections are computed through a simple kji -loop through the field and corrections are carried out immediately.

As a result, a corrected FS-value in a point (i, j, k) is used in the following (neighbouring) points, thus inducing an accumulation of corrections. Hence, the final FS-field will not be symmetric in general, even when the underlying flow problem is symmetric. In the new method the field is first scanned through a kji -loop, but the corrections are not yet carried

out. The effect of possible corrections is monitored and stored and all possible corrections are gathered. When the full field is scanned, the corrections are carried out, still based on the height function approach. This results in a symmetric correction of the FS-field. When the underlying flow problem is symmetric, the final FS-field will also be symmetric.

Chapter 5

Results

In this chapter some results of simulations will be shown. First we will look at the original method, i.e. the method with the delta function of the form $4C(1 - C)$. As we have seen at the end of the second chapter, the units do not match. Therefore it was not expected that this method works well, as we will see. So further simulations are done with the delta function $|\nabla C|$.

The other results are of simulations that are done with κ 's that are discretized in the previous chapter, and will be compared to the height function method. Finally some 3-dimensional results will be shown.

For the simulations a test case with a the stationary bubble is used. The initial velocity field is zero. This should stay zero in time, as no external forces are added in the simulations, but we've learned that (spurious) velocities may show up.

For the simulations we used the following settings. We have set the density of the fluid ρ_w to $1.0 \cdot 10^3 \text{ kg/m}^3$, the density of air ρ_a to 1.0 kg/m^3 , the surface tension coefficient to $\sigma = 73 \cdot 10^{-3} \text{ N/m}$, and the radius of the bubble to $r = 0.1 \text{ m}$.

From the Laplace-Young equation we know that the pressure jump Δp over the free surface, must be 0.73 (in 2 dimensions), since $\Delta p = \frac{\sigma}{r}$. In 3 dimensions $\Delta p = 2\frac{\sigma}{r}$ holds, which gives $\Delta p = 1,46$.

5.1 Simulation done with the delta function $4C(1 - C)$

In this section, results of a simulation with the delta function $4C(1 - C)$ will be shown. The height function method is used for curvature calculations.

The results confirm what was said at the end of chapter 2, with regard to the missing factor $\frac{1}{m}$ when looking at the units. Therefore it is not expected that this method will work well. As we can see in figure 5.1 the bubble has moved a bit after 1000 sec, and figure 5.2 shows that there is hardly a pressure jump. Δp should be 0.73, but it is 0.0032 instead.

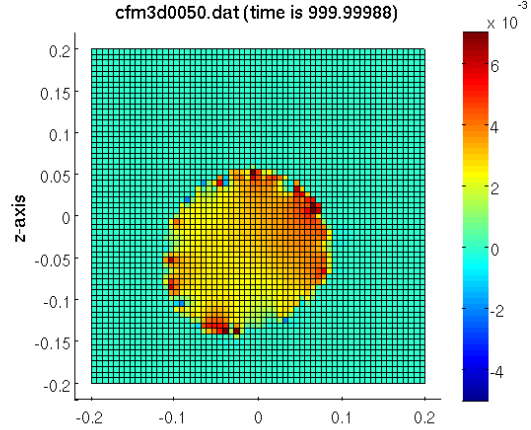
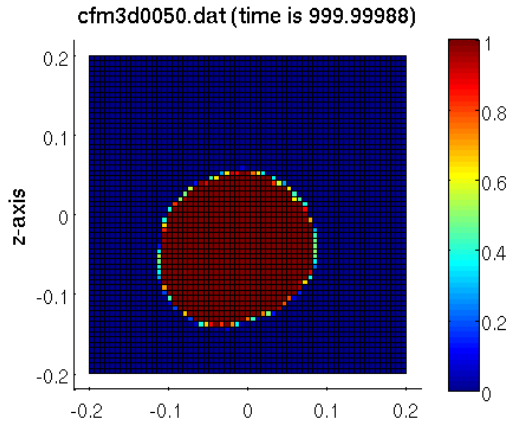


Figure 5.1: There is obvious movement when $\delta_\Gamma = 4C(1 - C)$ is used. Figure 5.2: There is hardly a pressure jump when $\delta_\Gamma = 4C(1 - C)$ is used.

In figure 5.3 and 5.4 the 2norm of absolute velocities is plotted against time (in figure 5.4 a semi-logarithmic¹ scale is used.) We can see that the absolute velocity is not decreasing, so it is not expected that the bubble will come to rest, and eventually will crash into the boundary instead.

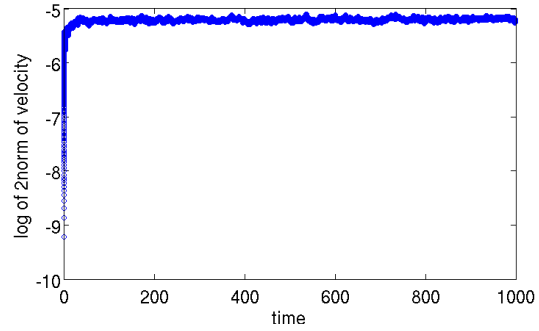
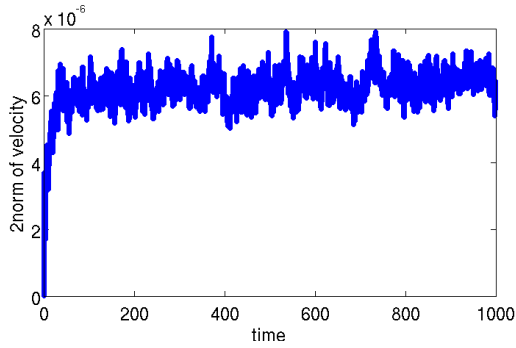


Figure 5.3: Mean 2norm of the absolute velocity against time. Figure 5.4: The same as figure 5.3, but now in logarithmic scale.

¹Throughout this thesis when 'plotting on semi-logarithmic scale' is mentioned, logarithmic to the base 10 is meant.

5.2 5 cells vs. 9 cells discretization

In the previous chapter we expected (according to the figures 4.3 and 4.4) that a finite difference discretization of $\kappa = \nabla \cdot \mathbf{n}$ over 5 cells might be a bit inaccurate, as a lot of cells were omitted in the 5 cells discretization. Therefore 4 cells were added to the discretization to include more information. Some results of simulations with both discretizations will be compared to each other in this section.

First we look at results of simulations done on a 41×41 grid. As we can see in figure 5.5, the method done with the 9 cells discretization shows smaller (spurious) velocities.

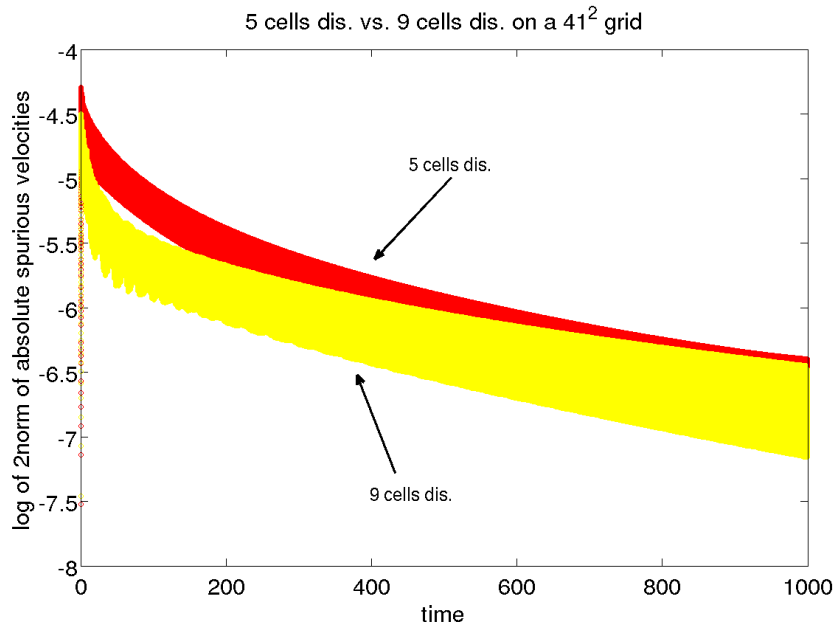


Figure 5.5: Comparison on a 41×41 grid.

If we look at simulations done on a 61×61 grid, we again can see the same phenomena.

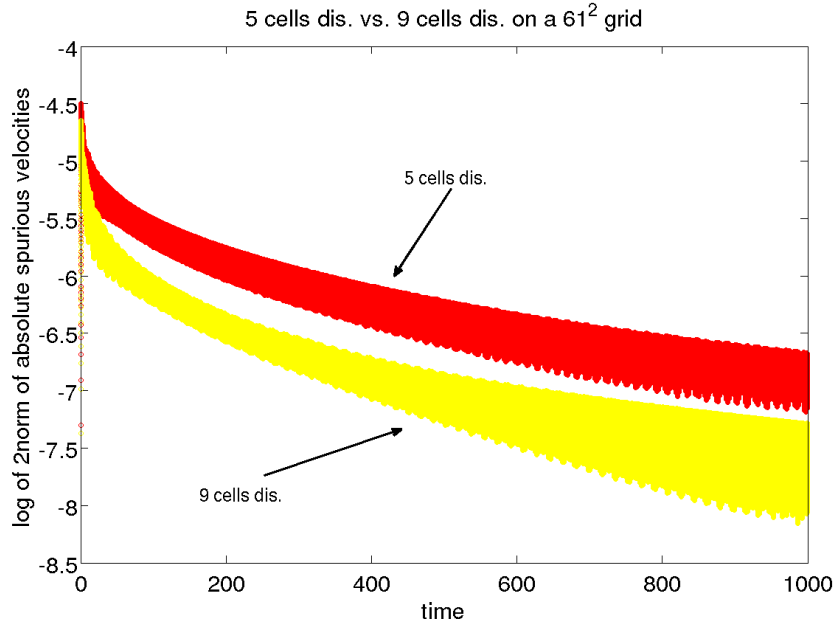


Figure 5.6: Comparison on a 61×61 grid.

In figure 5.6 we can clearly see that the velocities with the 9 cells discretization are smaller than with the 5 cells discretization on a 61×61 grid.

If we look at the values of the curvature κ , we see that the values of the 5 cells discretization are poor, but values of the 9 cells discretization method are not very good either. (Since the correct value is $\kappa = 10$).

grid	5 cells dis.	9 cells dis.
41×41	4,96	8,20
61×61	4,37	10,67

The table with curvature κ .

5.3 Comparison with the height function method

In the previous section we have seen the accuracy of both the 5 cells discretization method and the 9 cells discretization method. We have seen that both methods are not good with regard to spurious velocities. Albeit the poor results we like to compare the best of these two methods with the height function method. So in this section the height function method will be compared with the 9 cells discretization of $\kappa = \nabla \cdot \mathbf{n}$. Further a comparison will be done with the height function method and the method of the discretization of κ based on Shirani's discretization of the unit normal [6].

5.3.1 The height function method vs. 9 cells discretization method

In this section the height function method will be compared to the finite difference discretization of $\kappa = \nabla \cdot \mathbf{n}$ discussed in the previous chapter. The comparison is done on a few different grids.

First we will start with the height function method on a 40×40 grid.

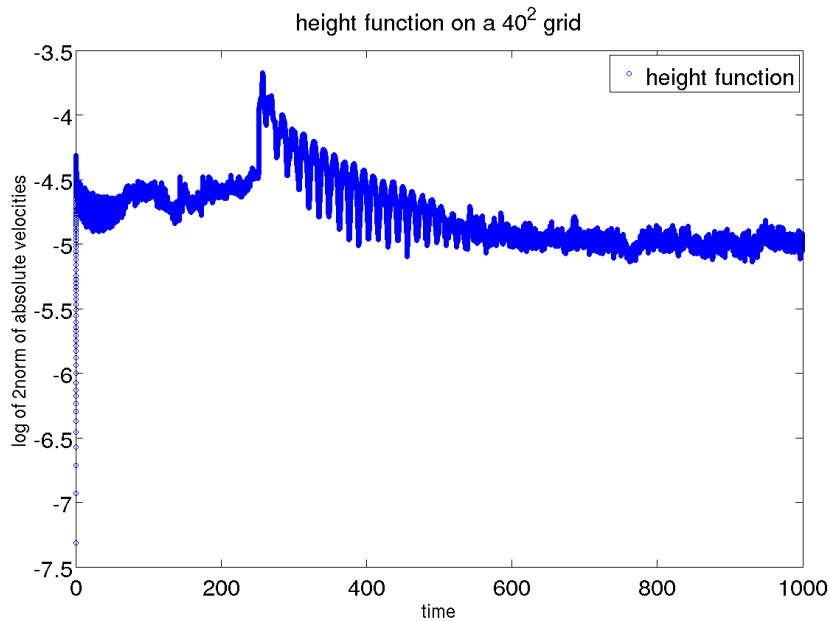


Figure 5.7: The peak indicates the rapid increasing of the spurious velocity.

In figure 5.7 the spurious velocities of this simulation are shown. We can see that the absolute velocity acts strange. The absolute velocity increases rapidly (around 250 sec.). The bubble crashes into the boundary, and then comes to rest (see figure 5.7 and 5.8). On most other 'even' grids this problem also occurs for the height function method. To avoid this problem, further 2-dimensional simulations are done on "odd" grids.

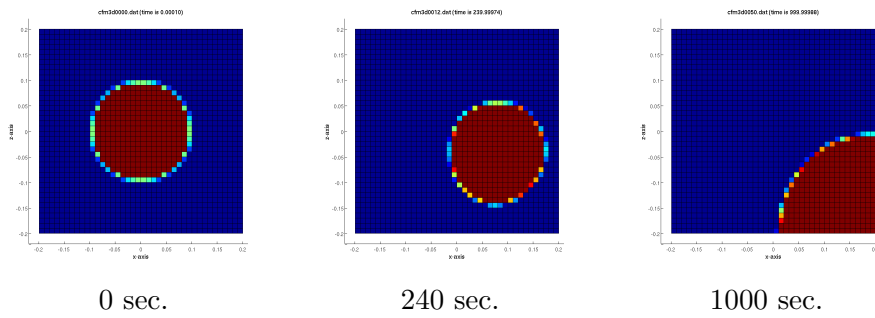
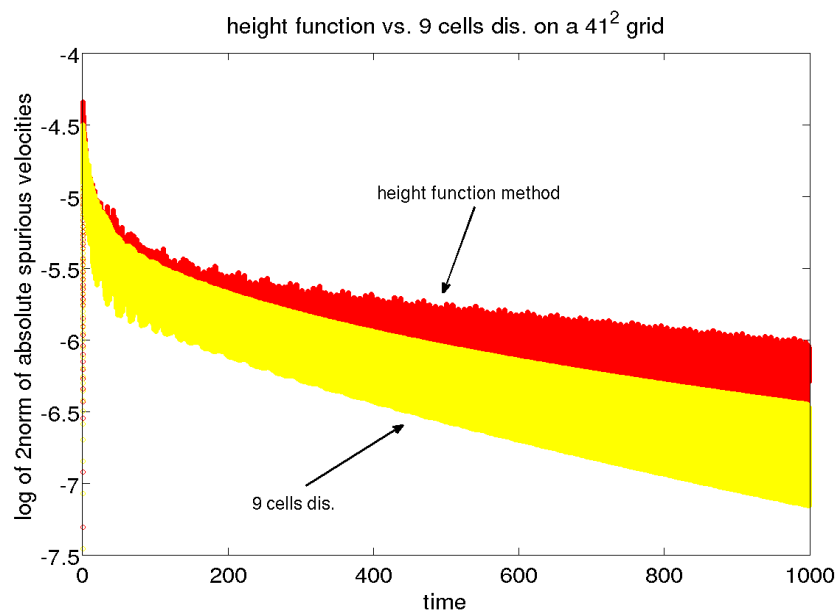


Figure 5.8: The bubble crashes into wall.

The figures below are results of simulations of different methods that are done on different grids. The pictures are plotted with semi-logarithmic scale.

Figure 5.9: The comparison on a 41×41 grid.

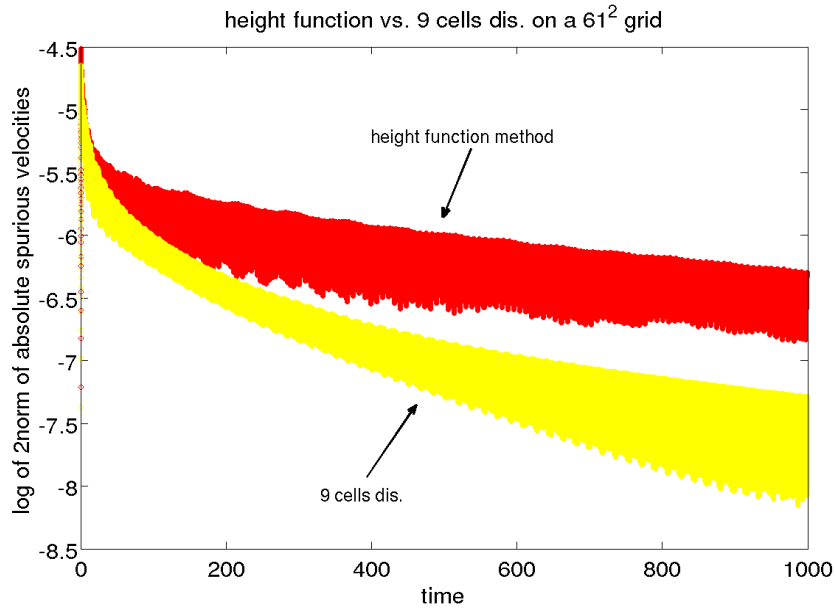


Figure 5.10: The comparison on a 61×61 grid.

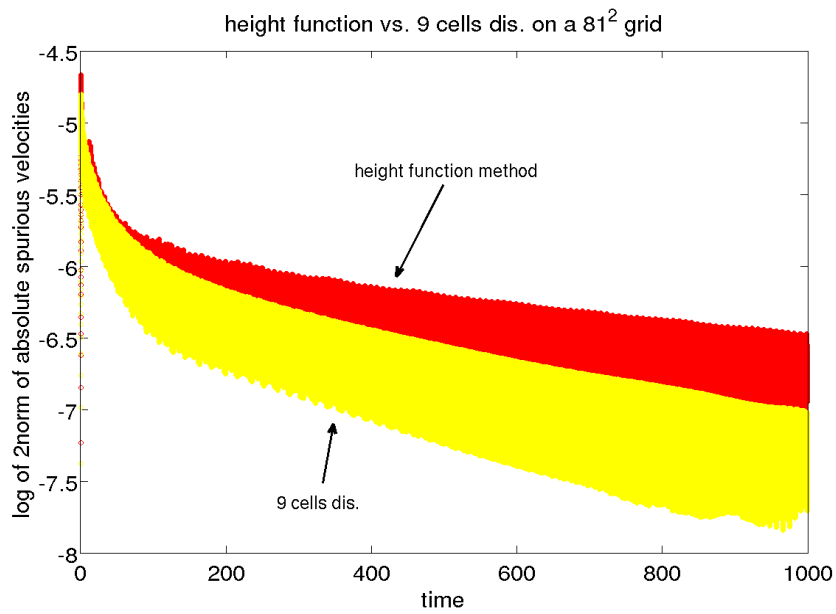


Figure 5.11: The comparison on a 81×81 grid.

According to the figures 5.9, 5.10 and 5.11 one might think that the 9 cells discretization method is better than the height function method. Simply because the 2norm of the absolute velocities is smaller. But if we look at the following table we see that the height function method gives much better approximations for the curvature κ . Since the values of the absolute velocity of the height function method are relatively small, we can say that the height function method is better than the 9 cells discretization method. If we look at the values for κ that are obtained with the 9 cells discretization method, we cannot see that the value is close to 10, only around 10.

grid	height function	$\kappa = \nabla \cdot \mathbf{n}$
40×40	crashes into wall	-0,952 - 13,4
41×41	9,75	8,20
61×61	9,78	10,67
81×81	9,80	9,13

The combination of this table and the figures 5.9, 5.10 and 5.11 is strange. Since bad approximations of κ are one of the main reasons for spurious velocities, one might expect that the height function method gives smaller spurious velocities, as κ is approximated more accurately. A problem might be the averaging of the fluid densities over the free surface. To avoid this problem we have set both densities equal ($\rho_{\text{water}} = \rho_{\text{air}} = 1.0 \cdot 10^3$). The results of these simulations are shown in figure 5.12 and figure 5.13. Here we see that the lowest values of the velocity of the height function method are indeed smaller than the lowest values of the velocity of the 9 cells discretization. This indicates that there is a problem with the density averaging indeed. We can conclude that the height function method is a better method than the 9 cells discretization method.

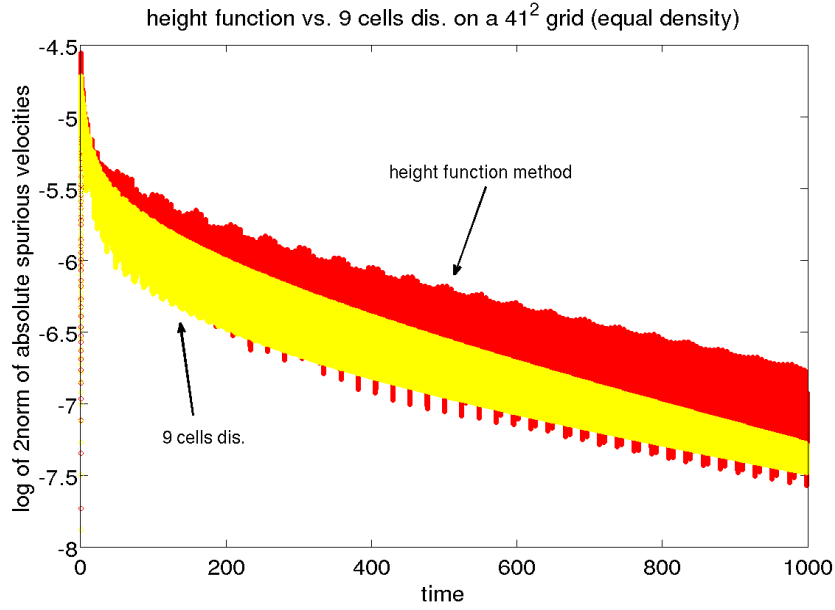


Figure 5.12: With $\rho_{\text{water}} = \rho_{\text{air}} = 1.0 \cdot 10^3$ on a 41×41 grid.

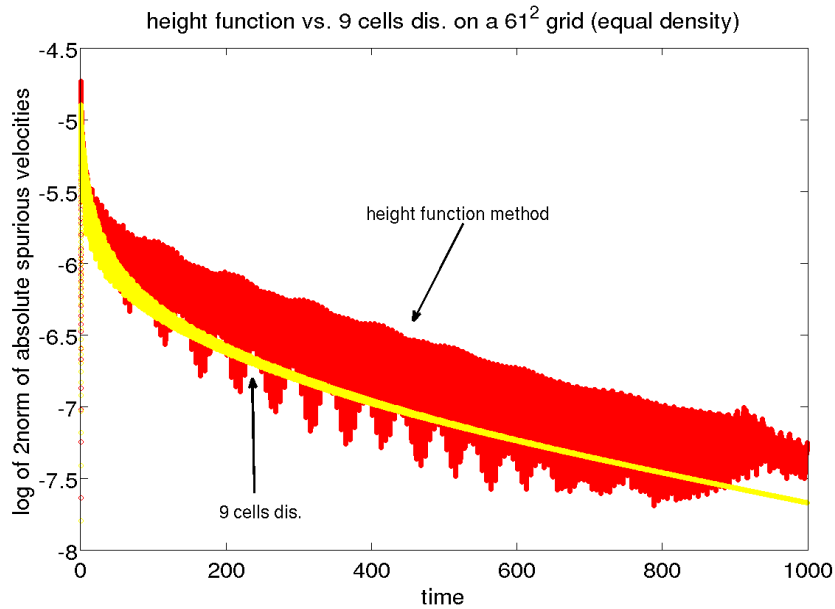
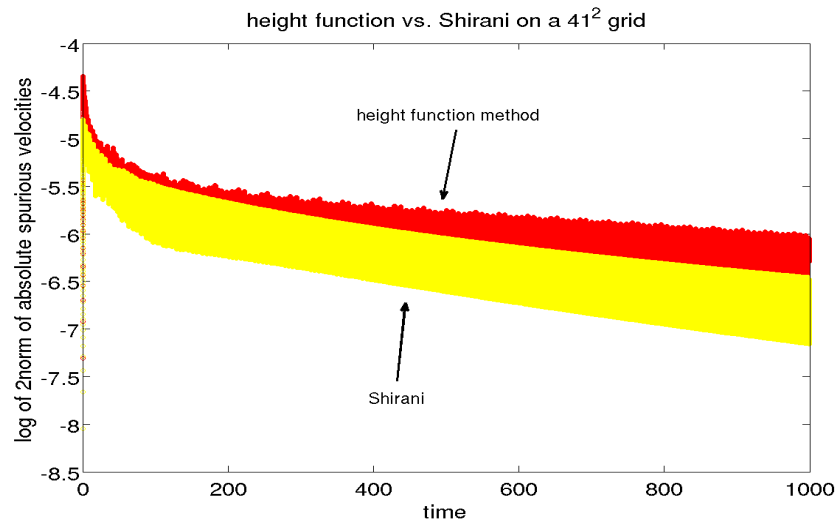
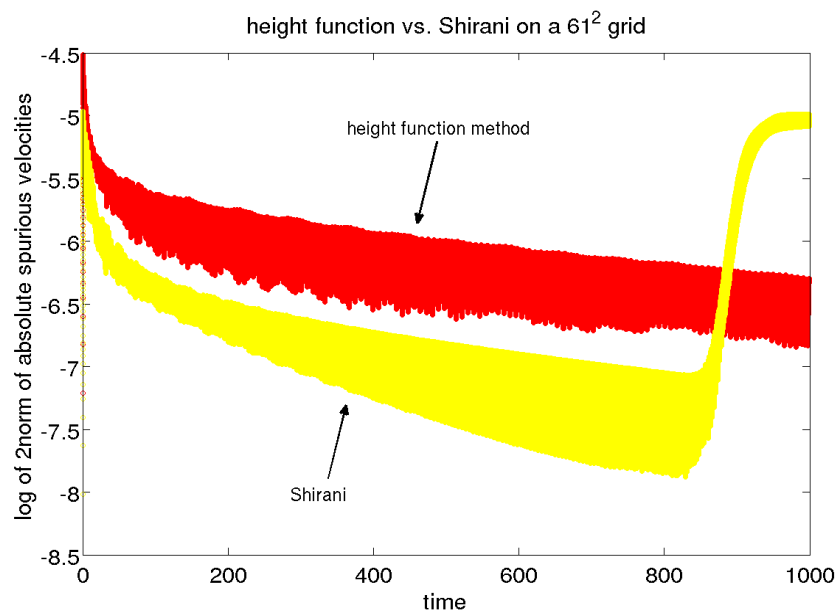


Figure 5.13: With $\rho_{\text{water}} = \rho_{\text{air}} = 1.0 \cdot 10^3$ on a 61×61 grid.

5.3.2 The height function method vs. method based on Shirani's unit normal

In this section the height function method will be compared to the method based on Shirani's unit normal [6]. The comparison is done on the same grids as the comparison between the height function method and the 9 cells discretization, i.e. on a 41×41 , 61×61 and 81×81

grid.

Figure 5.14: The comparison on a 41×41 grid.Figure 5.15: The comparison on a 61×61 grid.

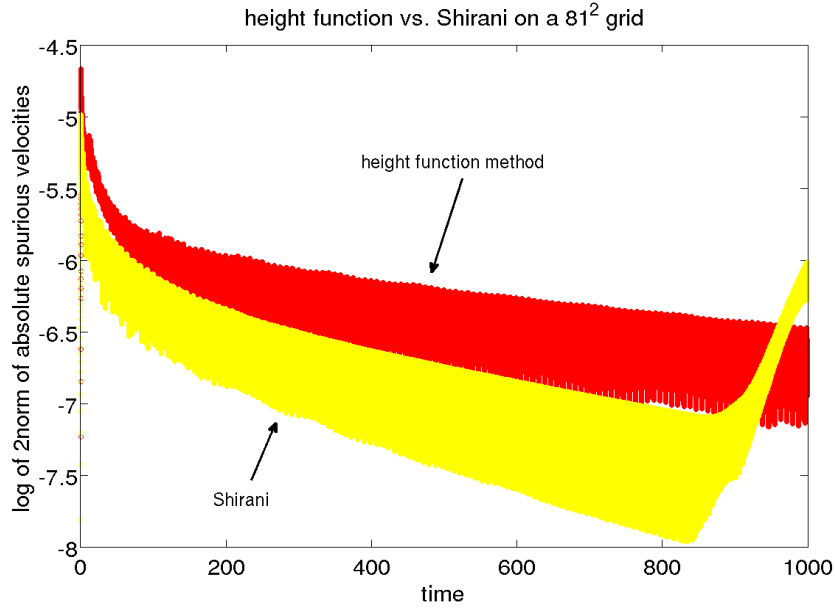


Figure 5.16: The comparison on a 81×81 grid.

If we look at the comparison on the 41×41 grid (figure 5.14) we do not see much happening (yet). What we see is that the absolute velocities of the method based on Shirani’s unit normal are smaller than the absolute velocities of the height function method. We also can see this in figure 5.15 and figure 5.16 (61×61 and 81×81 grid resp.)

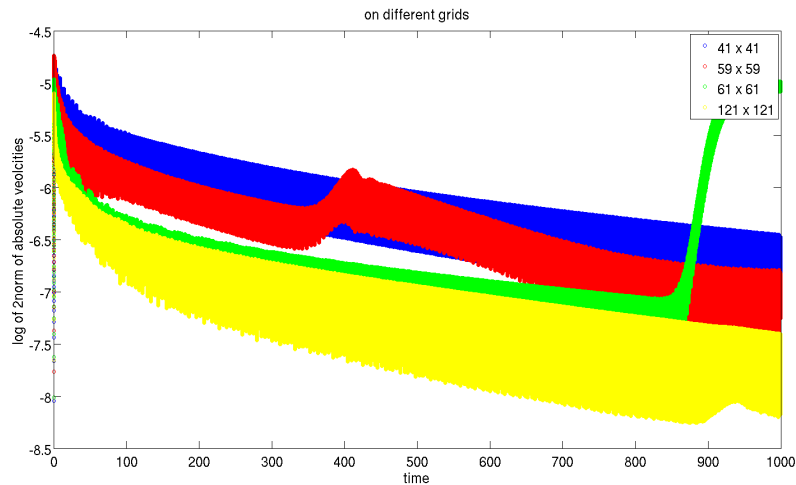


Figure 5.17: Comparison on different grids (Shirani).

grid	height function	Shirani
41×41	9,75	9,5205
61×61	9,78	10,22
81×81	9,80	10,27

In figure 5.15 and figure 5.16 we see that the velocities of the 'Shirani methods' are increasing rapidly around 820 seconds. We have looked at this problem. During our simulations, whether we have set the timestep δt such that the CFL-condition is obeyed, or the capillary timestep is used. The sudden increase of the spurious velocity still occurs. The reason of this remains unknown.

Symmetry

Another point of interest in this case, is the preservation of the symmetry of the velocity field when the simulation is done with the method based on Shirani's unit normal. If we look at the figures 5.18 and 5.20 (the height function method), we see a symmetry of the second order. If we look at the figures 5.19 and 5.21, we see a symmetry of the fourth order.

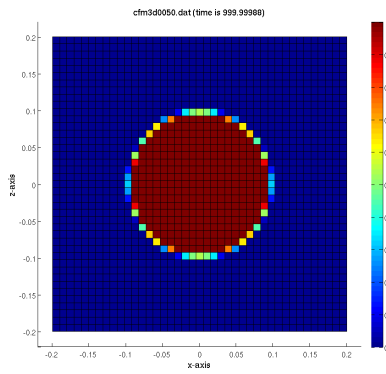


Figure 5.18: The volume fraction function of a simulation done with the height function method.

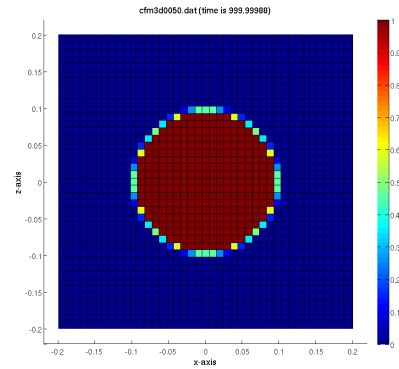


Figure 5.19: The volume fraction function of a simulation done with method based on Shirani's unit normal.

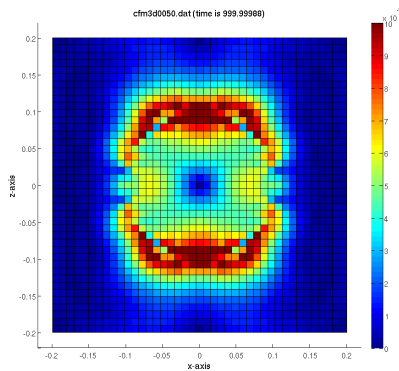


Figure 5.20: The velocity field of a simulation done with the height function method.

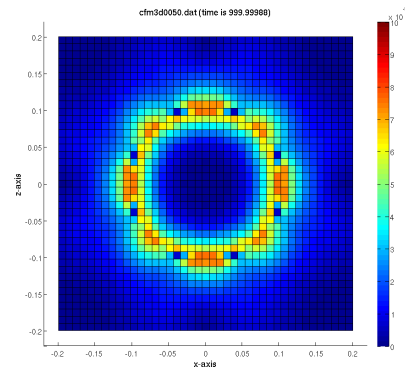


Figure 5.21: The velocity field of a simulation done with the method based on Shirani's unit normal.

5.4 Results of 3D methods

In this section we will show some results of a simulation that is done with the curvature discretized following § 4.4.2 (the method based on Shirani’s unit normal). This is done with the best combination tested in the two dimensional simulations, i.e. with $\delta_\Gamma = |\nabla C|$ as the delta function and the displacement algorithm VFLUP. Further, it is done on a 40^3 grid, problems that occur on ‘even’ grids in 2 dimensions do not seem to occur in 3 dimensions². By looking at the cross sections (figures 5.22, 5.23, 5.24, 5.25 and 5.26) in i, j and k directions, we can see that the method is nearly symmetric in the 3 directions.

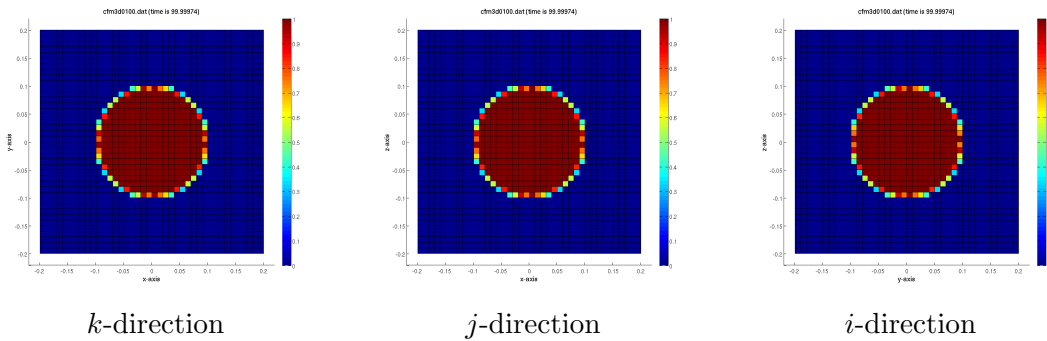


Figure 5.22: The volume fractions.

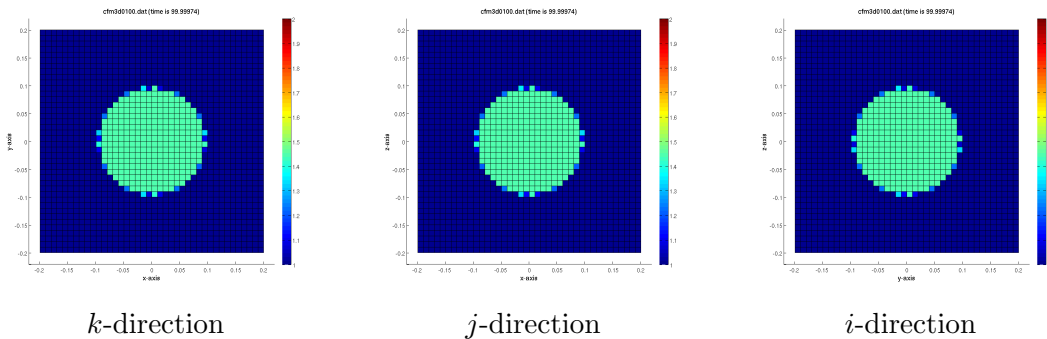


Figure 5.23: The pressure field.

²3D calculations are done on both 40^2 and 41^2 grids.

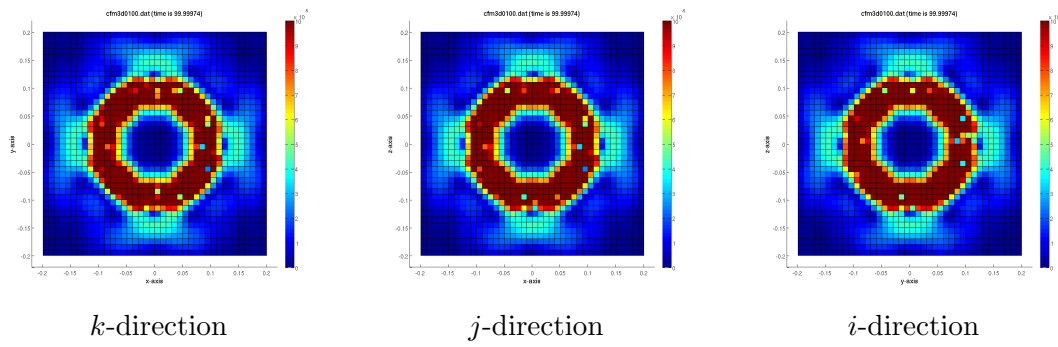


Figure 5.24: The absolute velocities.

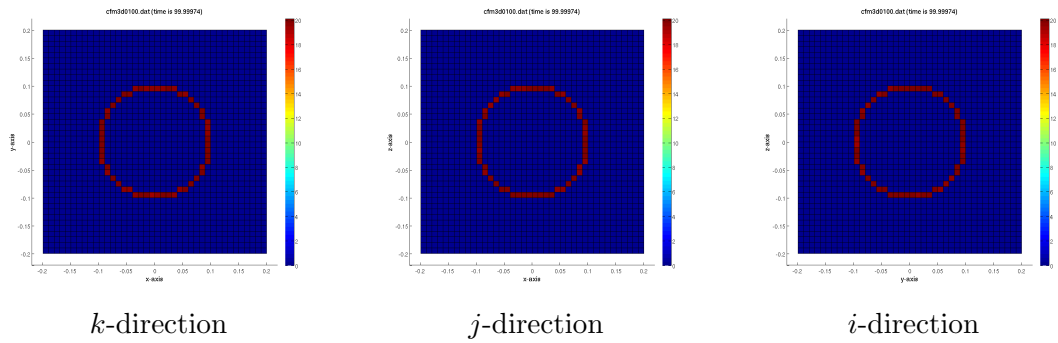
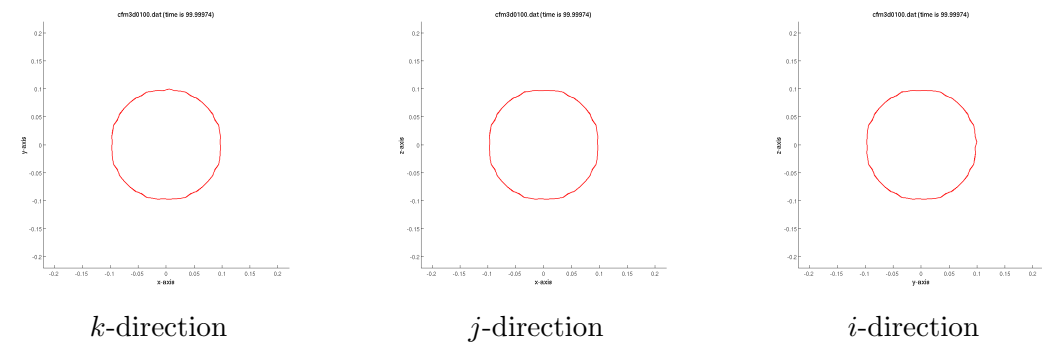
Figure 5.25: The curvature κ .

Figure 5.26: The interface.

In figure 5.25 we can see that the curvatures in the surface cells more or less have the same values. The mean value of the curvature $\bar{\kappa} = 19.8370$, which is close to the real value of 20, since in 3 dimensions $\kappa = \frac{2}{r}$ holds for a bubble, and the $r_{\text{bubble}} = 0.1$.

Here we see plots of the 2norm of the absolute velocity. As we can see in both figure 5.27 and figure 5.28 the values of the absolute velocities are decreasing. This means the bubble is coming to rest.

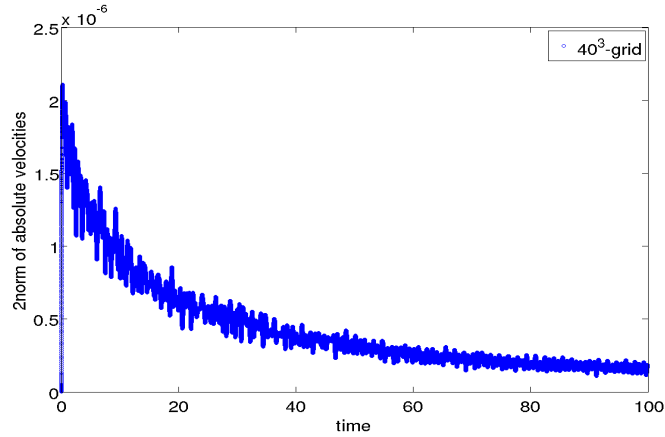


Figure 5.27: The absolute velocity of the 3D version of the method that is based on Shirani’s unit normal.

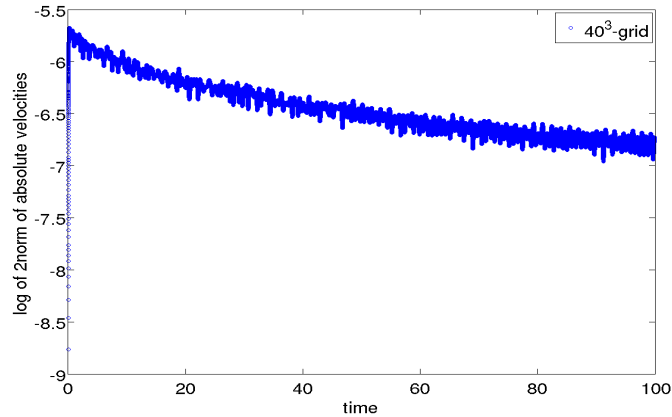


Figure 5.28: The semi-logarithmic version of figure 5.27.

In figure 5.29 we see some 3D methods compared to each other by absolute spurious velocities. We can see that the 2norm of the absolute spurious velocity of the method based on Shirani’s normal is obviously decreasing, while that of the height function method is not. The decreasing can also be seen in the 27 cells discretization method (the 3D version of the 9 cells discretization method).

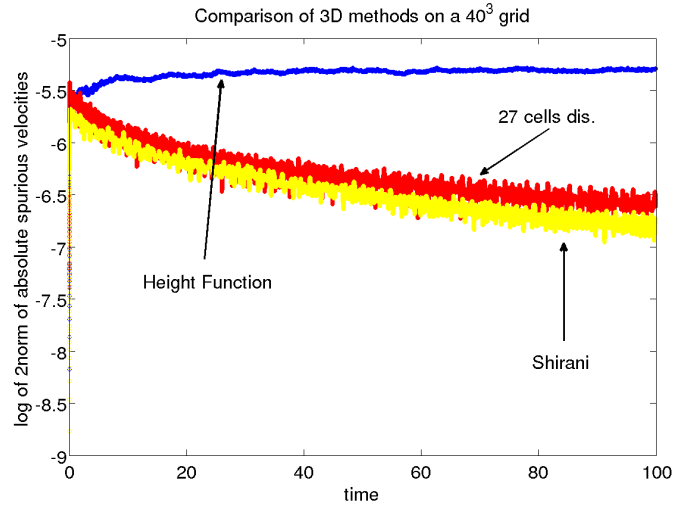


Figure 5.29: 3D methods comparison of the heightfunction method, 27 cells discretization and the method based on Shirani's unit normal

Here we see a table of the three 3D simulations. The values of κ of the height function method are widely spread, but the value for the pressure jump is good (the correct value is $\Delta p = 1.46$). By looking at this table the 'Shirani method' looks like the best one in 3D, since both the values for κ and the pressure jump Δp are the closest to the correct values.

used method	min. κ	max. κ	pressure jump Δp
Height function	0.18	43.76	1.4782
27 cells dis.	18.74	19.47	1.3951
Shirani	19.73	20.11	1.4491
Exact	20	20	1.46

Chapter 6

Conclusion

In this study we have done simulations with a surface tension model (chapter 2). It is known from literature that one of the main reasons for spurious currents to exist, is badly approximated surface curvatures κ . Brackbill *et al.* [1] first mollifies the volume fraction function C , and then discretize this mollified function \tilde{C} with a finite difference method. The mollification is necessary for the original volume fraction function C is too abrupt for a finite difference method (see figure 3.2). This method is used widely.

But because the pressure gradient in the Navier-Stokes equation shows exact the same abruptness as the volume fraction function C , we thought it would be interesting to see what will happen if we leave C unmollified. In this study we have investigated a few discretizations of the curvature κ . Also we have used another delta function, for as we have seen in § 3.7 there was missing a factor $\frac{1}{m}$ in the method using delta function $\delta_\Gamma = 4C(1 - C)$ when looking at the equations, so this method is not expected to work well. This is confirmed in § 5.1. We saw obvious movements of the bubble, and the spurious velocities were not decreasing, so the bubble will keep on moving and eventually crashes onto the boundary.

Because of this reason further simulations were done with another delta function. We have chosen $\delta_\Gamma = |\nabla C|$, as the units are matching when using this deltafunction. After many simulations we can conclude that VFLUP is a better displacement algorithm than VFCONV (see figure 4.19 and 4.20). So all simulations are done with deltafunction $|\nabla C|$ and displacement algorithm VFLUP unless stated otherwise.

First we have compared the 5 cells discretization of the curvature κ with the 9 cells discretization. The 5 cells discretization turned out to be inaccurate. A cause can be that too many cells are omitted in the 5 cells discretization (see figure 4.3 and figure 4.4). The four "obvious" cells were added to obtain a 9 cells discretization. In figure 5.5 and figure 5.6 we can see that the spurious velocities of the 9 cells discretization method are smaller. The values of the curvature κ of the 9 cells discretization are better, but unfortunately far from convincing.

To see how good the 9 cells discretization method actually is, we have compared it to the height function method, which is a well-known method for approximating κ [8]. In § 5.3.1 we have compared both methods. At first glance, the 9 cells discretization method looks like a better method than the height function method, just by looking at the spurious velocities of both methods. But if we look at the values of the curvature the height function method is much better. This is awkward, since the values of the spurious velocities are worse. We suspected that this is caused by the averaging of the densities of the fluid, and tested it with $\rho_{\text{water}} = \rho_{\text{air}} = 1.0 \cdot 10^3$. As a result the minimum values for the spurious velocities are smaller

with the height function method. We can say that the height function method is better than the 9 cells discretization method.

Another discretization that we've investigated is based on the way the unit normal is discretized by Shirani *et al.* [6]. We have also compared this method to the height function method (the figures of this comparison are in § 5.3.2). We saw that the method based on Shirani's unit normal has smaller spurious velocities than the height function method till a certain time, but then suddenly the spurious velocities increase rapidly. Until the sudden increase of the velocities the method based on Shirani's unit normal seems a better method than the height function method. If we look at the values of the curvature κ , they are not worse than the values that are obtained with the height function method. If we look at the symmetry of the velocity field and volume fraction function (see figure 5.18, 5.19, 5.20 and 5.21) we see that the method based on Shirani's unit normal produces much nicer symmetry. So until the strange rapid increasing of the spurious velocity the method based on Shirani's unit normal looks better. We do not have an idea what causes this problem.

We have also done some simulation in 3D. In § 5.4 we see that 'Shirani's method' is nicely symmetric in 3 directions. We have also compared several methods on a $40 \times 40 \times 40$ grid. On this grid 'Shirani's method' looks like to be the best method.

Bibliography

- [1] J.U. Brackbill, D.B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100: 335-354, 1992
- [2] B. Meier, G. Yadigaroglu, B.L. Smith. A novel technique for including surface tension in PLIC-VOF methods. *Eur. J. Mech. B/Fluids.*, 21: 61-73, 2002
- [3] A.E.P. Veldman, J. Gerrits, R. Luppens, J.A. Helder, J.P.B. Vreeburg. The numerical simulation of liquid sloshing on board spacecraft. *J. Comput. Phys.*, 224: 82-99, 2007
- [4] S. Popinet, S. Zaleski. A front-tracking algorithm for accurate representation of surface tension. *Int. J. Numer. Fluids*, 30: 75-793, 1999
- [5] K.M.T. Kleefsman, G. Fekken, A.E.P. Veldman, B. Iwanowski, B. Buchner. A volume-of-Fluid based simulation for wave impact problems. *J. Comput. Phys.*, 206: 363-393, 2005
- [6] E. Shirani, N. Ashgriz, J. Mostaghimi. Interface pressure calculation based on conservation of momentum for front capturing methods. *J. Comput. Phys.*, 203: 154-175, 2005
- [7] M.W. Williams, D.B. Kothe, E.G. Puckett. Accuracy of Convergence of Continuum Surface Tension Models, 1998
- [8] M. ten Caat. Numerical Simulation of Incompressible Two-Phase Flow, Master's Thesis. Groningen University, September 2002
- [9] S. Afkhami and M. Bussmann. Height functions for applying contact angles to 3D VOF simulations. *Int. J. Numer. Meth. Fluids*, 57: 453 - 472
- [10] M. Raessi, J. Mostaghimi, M. Bussmann. Advecting normal vectors. A new method for calculating interface normals and curvatures when modeling two-phase flows. *J. Comput. Phys.*, 226: 774-797, 2007
- [11] M.M. Francois, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, M.W. Williams. A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. *J. Comput. Phys.*, 213: 141-173, 2006
- [12] C.W. Hirt, D.B. Nichols. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. *J. Comput. Phys.*, 39: 201-225, 1981