Christoph Bauinger, BSc

An enhanced PCISPH Method for Engineering Applications and Multiphase Flows

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mathematics

submitted to

Graz University of Technology

Supervisor

Prof. Dr. O. Steinbach

Institute of Applied Mathematics

Graz, July 2018

Abstract English

The predictive-corrective incompressible smoothed particle hydrodynamics method (PCISPH) is a variant of smoothed particle hydrodynamics (SPH) introduced in 2009 for incompressible fluid flow simulations in computer graphics. It already showed promising improvements with respect to the computation time compared to the currently common weakly compressible SPH (WCSPH) method. But neither the original paper from 2009, in which PCISPH was presented to the public, nor any other publication so far investigated its accuracy and stability for engineering applications and physics based simulations.

This work first gives a general introduction to SPH, which includes a derivation of SPH and its application to the governing equations of fluid dynamics. Afterwards, novel enhancements for PCISPH are presented to adapt the method for physics based flows in engineering. These enhancements include 1) numerical diffusion and pressure smoothing to reduce the noise in the pressure field; 2) an adaption of the Adami Boundary Condition to PCISPH; 3) a second order time-stepping scheme; 4) a more general derivation to also cover multiphase flows with PCISPH. This enhanced PCISPH method is then tested against the original PCISPH method, WCSPH, other numerical solvers, and experiments, and it is shown that the enhanced PCISPH is significantly faster than WCSPH while preserving the accuracy.

Abstract German

Predictive Corrective Incompressible Smoothed Particle Hydrodynamics (PCISPH) ist eine Variante von Smoothed Particle Hydrodynamics (SPH), die erstmals in 2009 für inkompressible Fluidsimulationen in der Computergraphik vorgestellt wurde. Sie hat damals vielversprechende Resultate hinsichtlich der Laufzeit, im Vergleich zu der aktuell üblichen Weakly Compressible SPH Methode (WCSPH), geliefert. Aber weder das Originalpaper aus 2009, in welchem PCISPH vorgestellt wurde, noch irgendeine andere Publikation haben die Methode auf ihre Genauigkeit und Stabilität in Anwendungen der Ingenieurswissenschaften und in Physik basierten Simulationen untersucht.

Diese Arbeit gibt zu Beginn eine allgemeine Einleitung in SPH, die die Herleitung von SPH und dessen Anwendung of die Grundgleichungen der Fluiddynamik beinhaltet. Anschließend werden neue Entwicklungen von PCISPH, um die Methode für physikalische Strömungen in den Ingenieursdisziplinen anzupassen, präsentiert. Diese Entwicklungen beinhalten 1) numerische Diffusion und Glättung des Drucks, um das Rauschen im Druckfeld zu reduzieren; 2) eine Anpassung der Adami Randbedingung an PCISPH; 3) ein Zeitschrittverfahren zweiter Ordnung; 4) eine allgemeinere Herleitung, um auch Mehrphasenströmungen mit PCISPH behandeln zu können. Diese verbesserte PCISPH Methode wird dann gegen die originale Methode, WCSPH, andere numerische Löser und Experimente getestet, wobei gezeigt wird, dass die verbesserte PCISPH Methode signifikant schneller ist als WCSPH, während die Genauigkeit erhalten bleibt.

Acknowledgments

For many students, the master thesis marks the last hurdle towards finishing their studies and starting a life outside the university. For others, it is the start of new studies. Independent of the individual students' future plans, all needed to reach this point first. In my experience, this often required dedication, concentration, sleepless nights and coffee. A lot of coffee. But the most important thing on my personal way to this point were the people around me. Those who supported, helped and motivated me, and made my studies and this thesis possible.

One of those people was Dr. Martin Schifko, founder and CEO of the company ESS - Engineering Software Steyr GmbH, who allowed me to write this thesis as part of my work for ESS and gave me the possibility to work on something new, interesting and useful. I was part of a great team in ESS, which supported me to the best of their ability. Most notably Dr. Kamil Szewc and Dr. Chong Peng with whom this thesis would not have been possible at all.

Many thanks also to my supervisor Professor Dr. Olaf Steinbach, who guided me through this thesis with patience and great insights, whose numerical analysis lecture was one of the reasons I decided to specialize in numerical and applied mathematics, and who introduced me to teaching at the university in the last years.

I also want to thank my Mother, Renate Bauinger, and my Sister, Sabine Bauinger, for their great support, their open mindedness and their clear language when necessary.

Last, I want to thank all my colleagues and friends who made my studies worthwhile, fun, and a remarkable experience and who shared the long nights trying to solve the exercises with me. Without them, it would not have been the same. I wish them all the best for their future.

Contents

Introduction **7**

1 Derivation of SPH **9**
1.1 Interpolation . . . 9
1.2 Derivatives . . . 10
1.2.1 Gradient of $\langle A \rangle_c$ . . . 10
1.2.2 Divergence of $\langle \vec{A} \rangle_c$ . . . 11
1.3 Discretization . . . 11
1.4 SPH Operators . . . 13
1.5 Kernel Functions . . . 14
1.5.1 Properties . . . 14
1.5.2 Kernel Examples . . . 15

2 Applying SPH **18**
2.1 Continuity Equation . . . 18
2.2 Equation of Momentum . . . 19
2.2.1 Pressure Term . . . 20
2.2.2 Viscous Term . . . 21
2.3 Time Stepping . . . 21
2.4 Incompressibility . . . 23
2.4.1 Non-iterative EOS solvers . . . 24
2.4.2 Non-iterative EOS solvers with splitting . . . 25
2.4.3 Iterative EOS solvers with splitting . . . 26
2.4.4 Pressure Projection . . . 27
2.5 Boundaries . . . 27
2.5.1 Dummy Particles . . . 28
2.5.2 Boundary Forces . . . 31
2.6 Implementation . . . 32
2.6.1 Neighbor Search . . . 32
2.6.2 GPU Computing . . . 33

3 PCISPH **35**
3.1 Derivation of Pressure Correction f . . . 37
3.2 PCISPH in Engineering . . . 39
3.2.1 Numerical Diffusion and Pressure Smoothing . . . 40

3.2.2 Time Stepping 40
3.2.3 Boundary Handling 40
3.2.4 Multiphase PCISPH 41

4 Other SPH Variants **43**
4.1 WCSPH 43
4.2 IISPH 43

5 Numerical Tests **47**
5.1 Dam Break 47
5.2 Lid-Driven Cavity 49
5.3 Hydrostatic Equilibrium 53
5.4 Rayleigh-Taylor Instability 55

6 Conclusion and Future Work **63**

List of Figures **64**

Bibliography **65**

Introduction

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian particle based method to solve complex problems from fluid dynamics. It was first introduced in 1977 by Gingold and Monaghan [13] and, independent from each other, also by Lucy [24] to solve problems in astrophysics. Over the last four decades, the method was further developed and adapted for many different applications and was also introduced as a method for computational fluid dynamics (CFD) in engineering competing with Finite Difference (FDM), Finite Element (FEM) and, especially, Finite Volume Method (FVM).

According to my knowledge, it is one of the most used and widely spread mesh-free methods due to its great versatility, easy implementation, accurate results and further advantages compared to common mesh-based methods like the above mentioned FEM, FVM or FDM. Those advantages include easy ways to handle moving boundaries (forced and two-way coupled, [33]), multiphase flows, free surface flows, and - probably the most noteworthy of all - the expandability of meshes since in modern industrial applications the meshing process is one of the most cost and work intense tasks, when performing simulations.

The disadvantages of SPH are that there is no deep mathematical background yet. In addition, numerical experiments show that the number of elements necessary to reach a sufficient accuracy is quite high compared to mesh based methods like FVM and that the computation times can be significantly higher than with mesh based methods.

To counter the last disadvantage, the higher computation times, [7] presented a new variant of SPH in 2009, called *predictive-corrective incompressible SPH*, for applications in computer graphics. It already showed promising improvements with respect to the computation time compared to the currently common weakly compressible SPH (WCSPH) method. But neither the original paper [7], in which PCISPH was presented to the public, nor any other publication so far investigated its accuracy and stability for engineering applications and physics based simulations.

This work should give an introduction to some fundamentals of SPH in fluid dynamics and push the limits of the method to make it more versatile, accurate, faster, and worthwhile to use in engineering. For this purpose, a mathematical derivation of SPH is given in Chapter 1, where the method is derived using convolution and some basic properties of it. The chapter also includes information about the kernel functions used.

Chapter 2 shows how to apply SPH to the governing equations of fluid dynamics. This includes the continuity equation in Section 2.1, the equation of momentum in

Section 2.2, some thoughts on time-stepping in Section 2.3, enforcing incompressibility in Section 2.4 and how to handle boundary conditions in Section 2.5.

Chapter 3 then covers PCISPH, including the original derivation and the main topic of this thesis: the enhancement of the method to be able to handle physical correct fluid flows. The enhancements consist of 1) numerical diffusion and pressure smoothing to reduce the noise in the pressure field; 2) an adaption of the Adami Boundary Condition to PCISPH; 3) a second order time-stepping scheme; 4) a more general derivation to also cover multiphase flows with PCISPH.

In Chapter 4, other SPH variants of importance to this thesis are briefly presented. Namely WCSPH, since it is a widely used variant of SPH, ISPH, since it is the basis of the IISPH, which is the third alternative SPH method presented in the chapter and the logical development of PCISPH.

The final Chapter 5 then concludes with numerical tests to compare the enhanced PCISPH with the original PCISPH, WCSPH, other numerical solvers and analytical solutions.

It is assumed that the reader has a good understanding of the underlying physics, especially fluid mechanics, since there is no section dedicated to it. For an introduction to fluid mechanics, one can refer to [44] or [14].

1 Derivation of SPH

This chapter covers the derivation of the basic SPH formalism. It is loosely based on [44].

First, in Section 1.1, the interpolation of physical quantities is outlined, using convolution and approximations of the δ-distribution, so-called *kernels*. Section 1.2 explains the computation of derivatives with respect to the space coordinates of the interpolated physical quantities. The discretization is then covered in Section 1.3, while Section 1.4 presents alternative ways to represent the derivatives using common identities for the derivatives, called *SPH Operators*. Finally, Section 1.5 gives an overview of some kernel properties and shows some kernel examples.

1.1 Interpolation

The basic idea of SPH is to represent physical quantities at a given point $\vec{x}$ through an integral over some test volume and then discretize the resulting integral using interpolation points. These interpolation points are not fixed in space, but can be advected with the flow, and are therefore called *particles*.

Beginning with continuous integration, let $\Omega \subset \mathbb{R}^3$ be an open, bounded domain. Let $A(\vec{x}, t) : \Omega \times \mathbb{R}_+ \longrightarrow \mathbb{R}$ be some physical scalar valued field at a given point $\vec{x} \in \Omega$ in space and $t \geq 0$ in time (e.g. pressure, density, mass, etc.). The convolution with the δ-distribution yields

$$\begin{aligned} A(\vec{x}_0, t) &= (A * \delta)(\vec{x}_0, t) \\ &= \int_\Omega A(\vec{x}, t)\delta(\vec{x}_0 - \vec{x})d\vec{x}. \end{aligned}$$

For numerical purposes, the δ-distribution is an obvious problem. We get rid of it by approximation. Let $W_h : \Omega \longrightarrow \mathbb{R}$ be an approximation of δ in the sense

$$(A * W_h)(\vec{x}_0, t) \to A(\vec{x}_0, t) \text{ as } h \to 0,$$

for all $(\vec{x}_0, t) \in \Omega \times \mathbb{R}_+$. We call such a function *kernel* and the parameter h *smoothing length* or *smoothing radius*. Section 1.5 covers them in more detail and also shows some of the commonly used kernels. Using a similar notation as in [44], one can define the following continuous approximation.

Definition 1.1 (Continuous Approximation)**.** *As in the setting of above, let $\Omega \subset \mathbb{R}^3$ be an open, bounded domain. Let $A(\vec{x}, t) : \Omega \times \mathbb{R}_+ \longrightarrow \mathbb{R}$ and let $W_h : \Omega \longrightarrow \mathbb{R}$ be a kernel function. The continuous approximation of A is defined as*

$$\langle A \rangle_c(\vec{x}_0, t) := \int_\Omega A(\vec{x}, t) W_h(\vec{x}_0 - \vec{x}) d\vec{x} = (A * W_h)(\vec{x}_0, t).$$

Under sufficiently strong assumptions, one can show that this approximation is of second order with respect to h. This is proved in Theorem 1.5.

1.2 Derivatives

This section covers the computation of derivatives of the previously defined $\langle A \rangle_c$ with respect to the space variables. Of course, it is possible to numerically compute the derivatives, using for example finite differences. This is not preferably for multiple reasons. First, finite differences introduce another approximation error. Secondly, evaluating finite differences requires in general multiple evaluations of the function which can be time consuming. It would therefore be advantageous for an alternative to exist.

One will see in the following that for SPH derivatives can be formed using exact derivatives of the kernel W_h.

1.2.1 Gradient of $\langle A \rangle_c$

First, the gradient of a scalar valued function is investigated. Let $W_h \in \mathcal{C}^1\left(\overline{\Omega}\right)$ be a kernel function. Then the following identity holds true

$$\frac{\partial}{\partial x_i}(A * W_h)(\vec{x}_0, t) = \left(A * \frac{\partial}{\partial x_i} W_h\right)(\vec{x}_0, t),$$

where $\frac{\partial}{\partial \vec{x}_i}$ denotes the derivative with respect to the i-th space coordinate $\vec{x}_{0,i}$. The identity implies

$$\begin{aligned}
\nabla_{\vec{x}} \langle A \rangle_c(\vec{x}, t) &= \nabla_{\vec{x}} (A * W_h)(\vec{x}, t) \\
&= \int_\Omega A(\vec{x'}, t) \nabla_{\vec{x}} W_h(\vec{x} - \vec{x'}) d\vec{x'} \\
&= -\int_\Omega A(\vec{x'}, t) \nabla_{\vec{x'}} W_h(\vec{x} - \vec{x'}) d\vec{x'} \\
&= \int_\Omega \nabla_{\vec{x'}} A(\vec{x'}, t) W_h(\vec{x} - \vec{x'}) d\vec{x'} \\
&= \langle \nabla A \rangle_c(\vec{x}, t).
\end{aligned}$$

$\nabla_{\vec{x}} W_h(\vec{x} - \vec{x'})$ denotes the gradient with respect to $\vec{x}$, i.e.

$$\left(\frac{\partial W_h}{\partial \vec{x}_1}(\vec{x} - \vec{x'}), \frac{\partial W_h}{\partial \vec{x}_2}(\vec{x} - \vec{x'}), \frac{\partial W_h}{\partial \vec{x}_3}(\vec{x} - \vec{x'}) \right)^T,$$

and $\nabla_{\vec{x'}} W_h(\vec{x} - \vec{x'})$ denotes the gradient with respect to $\vec{x'}$, i.e.

$$\left(\frac{\partial W_h}{\partial \vec{x'}_1}(\vec{x} - \vec{x'}), \frac{\partial W_h}{\partial \vec{x'}_2}(\vec{x} - \vec{x'}), \frac{\partial W_h}{\partial \vec{x'}_3}(\vec{x} - \vec{x'}) \right)^T.$$

Obviously, $\nabla_{\vec{x}} W_h(\vec{x} - \vec{x'}) = -\nabla_{\vec{x'}} W_h(\vec{x} - \vec{x'})$ holds true. The one to last equality follows from the Gauss's Divergence Theorem [5] assuming that $\operatorname{supp} W_h(\vec{x} - \cdot) \subset \Omega = \emptyset$. This means, that the equality is in general wrong. Assuming the support of the kernel W_h is compact, it is still wrong close to free surfaces or the boundary. How to handle these cases is not covered here. It is discussed for example in [44, Section 5.2.5]. How to handle boundaries in general is presented in Section 2.5.

The above result can be summarized in the following lemma.

Lemma 1.2. *Let $A : \Omega \times \mathbb{R}_+ \longrightarrow \mathbb{R}$ be a scalar valued, continuously differentiable function. Let $\Omega \subset \mathbb{R}^3$ and let $W_h(\vec{x} - \cdot) \in \mathcal{C}^1\left(\overline{\Omega}\right)$ with $\operatorname{supp} W_h(\vec{x} - \cdot) \subset \Omega = \emptyset$ for some $\vec{x} \in \Omega$. Then*

$$\nabla_{\vec{x}} \langle A \rangle_c(\vec{x}, t) = \langle \nabla A \rangle_c(\vec{x}, t).$$

1.2.2 Divergence of $\langle \vec{A} \rangle_c$

Similar to the computation of the gradient in the previous Section 1.2.1, one can analyze the divergence of a vector valued function. Following the same arguments as before, one gets the following lemma.

Lemma 1.3. *Let $\vec{A} : \Omega \times \mathbb{R} \longrightarrow \mathbb{R}^3$ be a vector valued function. Let $W_h(\vec{x} - \cdot) \in \mathcal{C}^1\left(\overline{\Omega}\right)$ with $\operatorname{supp} W_h(\vec{x} - \cdot) \subset \Omega = \emptyset$ for some $\vec{x} \in \Omega$. Then*

$$\nabla \cdot \langle \vec{A} \rangle_c(\vec{x}_0, t) = \langle \nabla \cdot \vec{A} \rangle_c(\vec{x}_0, t) = \int_\Omega \vec{A}(\vec{x}, t) \cdot \nabla W_h(\vec{x}_0 - \vec{x}) d\vec{x}.$$

1.3 Discretization

The next step is the discretization of the integral from Definition 1.1.

Let $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$ be a finite set of points in Ω. These points are called *particles*. We can assign each particle $i = 1, \ldots, n$, a mass m_i such that the sum over all masses equals the total mass in Ω. I.e.:

$$\sum_{i=1}^{n} m_i = \int_\Omega \rho(\vec{x}, t_0) d\vec{x},$$

where $\rho(\vec{x}, t_0)$ denotes the initial density (the density at time t_0) at a point $\vec{x}$. It is now possible to assign a volume

$$V_i = \frac{m_i}{\rho_i} \tag{1.1}$$

to each particle, where $\rho_i := \rho_i(t) := \rho(\vec{x}_i, t)$ and $V_i := V_i(t) := V(\vec{x}_i, t)$ for a fixed time t. In practice, the initial distribution of the particles in Ω is in general not random but equidistant with some predefined distance Δr. The volume of each particle is then given by

$$V_i = (\Delta r)^3.$$

The computation of the mass m_i then follows from equation (1.1).

Using the particles as interpolation points for a numerical integration yields

$$\langle A \rangle_c(\vec{x}_i, t) = \int_\Omega A(\vec{x}, t) W_h(\vec{x}_i - \vec{x}) d\vec{x} \approx \sum_{j=1}^{n} V_j A(\vec{x}_j, t) W_h(\vec{x}_i - \vec{x}_j).$$

Using the shorter notation for fixed t and h with $A_j := A(\vec{x}_j, t)$ and $W_{ij} := W_h(\vec{x}_i - \vec{x}_j)$, one can define

Definition 1.4 (Discrete Approximation)**.** *As in the setting of above, let $\Omega \subset \mathbb{R}^3$ be an open, bounded domain. Let $A(\vec{x}, t) : \Omega \times \mathbb{R}_+ \longrightarrow \mathbb{R}$, let $W_h : \Omega \longrightarrow \mathbb{R}$ be a kernel function and let $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$ be a finite set of points in Ω. The discrete approximation of A is defined as*

$$\langle A_i \rangle := \sum_{j=1}^{n} V_j A_j W_{ij}, \tag{1.2}$$

where the subscripts denote the evaluation at the corresponding discretization point as described above.

The accuracy of $\langle A_i \rangle$ obviously depends on the distribution and the number of the supporting points, i.e. the particles, $\vec{x}_j$, $j = 1, \ldots, n$. The number of supporting points considered in the approximation (1.2) depends on W_h, especially if the support of W_h is bounded. But as the number of particles admitted by the support of W_h increases, the δ-distribution is in general approximated worse by W_h. So for $h \to 0$, the accuracy of the discretization (1.2) decreases while the accuracy of the approximation of the δ-distribution by the kernel W_h increases. In [44, Section 5.2.3], an error estimate for an equidistant distribution of the particles was presented. It was shown that in the 1D case the error of the discrete approximation $|\langle A_i \rangle - A_i|$ behaves like $h^2 + \left(\frac{\Delta r}{h}\right)^{\beta_W}$ as $h \to 0$ with Δr denoting the placement distance of the particles and β_W a constant depending on the kernel W. Since the particles are moving, the assumption of an equidistant particle distribution is in general too strong. A general error estimate for disordered particles as they occur in SPH simulations is still an unsolved problem [44].

Analogously, the derivatives from Section 1.2 can be discretized which results in

$$\nabla\langle A_i\rangle = \sum_{j=1}^{n} V_j A_j \nabla W_{ij} \tag{1.3}$$

and

$$\nabla\cdot\langle \vec{A}_i\rangle = \sum_{j=1}^{n} V_j \vec{A}_j \cdot \nabla W_{ij}. \tag{1.4}$$

1.4 SPH Operators

As one can see, the identities (1.3) and (1.4) are not symmetrical in the sense that the contribution from particle i to the value of particle j, A_j, is in general not the same as the contribution from particle j to the value of particle i, A_i. As we will see later (Section 2), this lack in symmetry can pose a problem. Therefore, it is necessary to use different but equivalent representations of (1.3) and (1.4). For that purpose, the following two identities are used, which hold true for every $k \in \mathbb{R}$ and every function $\rho \in \mathcal{C}^1(\Omega)$, as simple computations show

$$\nabla A = \rho^k \nabla\left(\frac{A}{\rho^k}\right) + \frac{A}{\rho^k}\nabla\left(\rho^k\right)$$

and

$$\nabla\cdot\vec{A} = \rho^k \nabla\cdot\left(\frac{\vec{A}}{\rho^k}\right) + \frac{\vec{A}}{\rho^k}\cdot\nabla\left(\rho^k\right).$$

For practical purposes, we take ρ as the density. Note that these identities are problematic for $k \neq 0$ for multiphase flows or free surfaces, since in that case $\nabla(\rho^k)$ does not exist in general on the interface due to the jumps in the density occurring there. Using the previously derived approximations for the right hand-side then yields the following classes of *SPH operators*

$$\langle\nabla A_i\rangle = \rho_i^k \sum_j V_j \frac{A_j}{\rho_j^k}\nabla W_{ij} + \frac{A_i}{\rho_i^k}\sum_j V_j \rho_j^k \nabla W_{ij} = \sum_j V_j \frac{\rho_i^{2k} A_j + \rho_j^{2k} A_i}{\rho_j^k \rho_i^k}\nabla W_{ij} \tag{1.5}$$

and analogously

$$\langle\nabla\cdot\vec{A}_i\rangle = \sum_j V_j \frac{\rho_i^{2k}\vec{A}_j + \rho_j^{2k}\vec{A}_i}{\rho_j^k\rho_i^k}\cdot\nabla W_{ij}. \tag{1.6}$$

Similarly, with the identities

$$\nabla A = \frac{1}{\rho^k}\nabla\left(A\rho^k\right) - \frac{A}{\rho^k}\nabla\left(\rho^k\right)$$

and

$$\nabla \cdot \vec{A} = \frac{1}{\rho^k} \nabla \cdot \left(\frac{\vec{A}}{\rho^k} \right) - \frac{\vec{A}}{\rho^k} \cdot \nabla \left(\rho^k \right),$$

one gets different classes of SPH operators

$$\langle \nabla A_i \rangle = -\frac{1}{\rho_i^{2k}} \sum_j V_j \rho_j^k \rho_i^k A_{ij} \nabla W_{ij} \tag{1.7}$$

and

$$\langle \nabla \cdot \vec{A}_i \rangle = -\frac{1}{\rho_i^{2k}} \sum_j V_j \rho_j^k \rho_i^k \vec{A}_{ij} \cdot \nabla W_{ij}, \tag{1.8}$$

where $\vec{A}_{ij}$, in accordance with the notation so far, denotes the difference $\vec{A}_i - \vec{A}_j = \vec{A}(\vec{x}_i) - \vec{A}(\vec{x}_j)$. One can see that these SPH operators are symmetrical.

1.5 Kernel Functions

The kernel functions were introduced in Section 1.1 as an approximation of the δ-distribution. They have to fulfill some requirements to actually be usable for SPH, which are discussed in some detail in Section 1.5.1. In short, they need to be normalized, symmetric and sufficiently smooth. Some examples of kernels employed in practice, fulfilling the requirements, are given in Section 1.5.2.

1.5.1 Properties

This section describes the necessary properties the kernel functions need to have according to [44]. For that, we have a look at the accuracy of the spatial approximation as in Definition 1.1.

Theorem 1.5. *Let $\Omega \subset \mathbb{R}^3$, let $A : \Omega \times \mathbb{R}_+ \to \mathbb{R}$ be a $\mathcal{C}^2$ function with respect to the first three variables and let the kernel W_h, as introduced above, have the following properties*

1. *W_h is continuous,*
2. *$\operatorname{supp} W_h$ is compact and convex and it exists a constant $C > 0$ such that for all $h < h_0$, for some h_0, the support of W_h is included in a sphere with radius Ch, i.e. $\operatorname{supp} W_h \subset B(Ch)$,*
3. *W_h is symmetric, i.e. $W_h(\vec{x}) = W_h(-\vec{x})$,*
4. *$\int_\Omega W_h(\vec{x} - \vec{x'}) d\vec{x'} = 1$ for all $\vec{x} \in \Omega$.*

Then the error of the approximation $|A(\vec{x},t) - \langle A\rangle_c(\vec{x},t)|$*, with* $\langle A\rangle_c(\vec{x},t)$ *from Definition 1.1, is quadratic in* h *for all points* $\vec{x} \in \Omega$ *sufficiently far from the boundary* $\partial\Omega$*.*

Proof. Assume $\vec{x} \in \Omega$ is sufficiently far from the boundary such that

$$\{\vec{x'} \in \Omega \mid W_h(\vec{x} - \vec{x'}) \neq 0\} \cap \partial\Omega = \emptyset.$$

Using a second order Taylor approximation [23] yields

$$A(\vec{x'},t) = A(\vec{x},t) + \left(\vec{x'} - \vec{x}\right)^T DA(\vec{x},t) + \frac{1}{2}\left(\vec{x'} - \vec{x}\right)^T D^2A(\vec{\zeta},t)\left(\vec{x'} - \vec{x}\right),$$

where $DA(\vec{x},t)$ and $D^2A(\vec{x},t)$ denote the gradient and the Hessian respectively and ζ is a point on the line $\left[\vec{x};\vec{x'}\right]$ which lies completely in Ω if $\vec{x} - \vec{x'} \in \operatorname{supp} W_h$ due to the convexity of the support. Then,

$$\begin{aligned}
\langle A\rangle_c(\vec{x},t) &= \int_\Omega A(\vec{x'},t)W_h(\vec{x} - \vec{x'})d\vec{x'} \\
&= \int_\Omega \left(A(\vec{x},t) + \left(\vec{x'} - \vec{x}\right)^T DA(\vec{x},t) + \dots\right. \\
&\quad \left. \dots \frac{1}{2}\left(\vec{x'} - \vec{x}\right)^T D^2A(\vec{\zeta},t)\left(\vec{x'} - \vec{x}\right)\right) W_h(\vec{x} - \vec{x'})d\vec{x'} \\
&= A(\vec{x},t) + O(h^2).
\end{aligned}$$

The last equality uses the normalization, the symmetry of the kernel and therefore anti-symmetry of the first order term, the assumptions to the support of the kernel function and the continuity of the second derivatives of A. □

Note that assumption 2 is not necessarily required but stated here since it is useful in practice to increase the performance and for the sake of simplicity of the proof. The assumption basically means that the support of the kernel goes to 0 as $h \to 0$ which makes the argument in the last equality, why the second order term behaves like $O(h^2)$, obvious. This is not a strong limitation for practical purposes. A compact supported kernel is often used in practice anyway since otherwise the complexity to evaluate the sums (1.2) would be quadratic. The easiest and most common way to fulfill assumption 3 in practice, is to set the kernel as a function of the distance $W_h(\vec{x}) = W_h(\|\vec{x}\|)$.

Another noteworthy fact is that the symmetry of the kernel W_h directly implies the anti-symmetry of the gradient $-\nabla W_h(\vec{r}) = \nabla W_h(-\vec{r})$ for a differentiable kernel W_h.

1.5.2 Kernel Examples

In the following, some of the most common kernels are presented. The normalization constants given are for the 3D case. In the case of 2D simulations, the constants might change and need to be adjusted accordingly.

Gaussian

The Gaussian kernel [13, 25] is defined as

$$W_h(\vec{r}) := \frac{1}{h^3 \pi^{\frac{3}{2}}} \exp\left(-\left(\frac{|\vec{r}|}{h}\right)^2\right).$$

In modern SPH codes, this kernel is rarely used since it leads to quadratic complexity when evaluating (1.2) for every particle and the problems occurring due to the non-empty intersection with the boundary as discussed in Section 1.2.

Wendland

The Wendland kernel [44, 47] is given as

$$W_h(\vec{r}) := \begin{cases} \frac{21}{16\pi h^3}\left(1-\frac{|\vec{r}|}{2h}\right)^4\left(1+2\frac{|\vec{r}|}{h}\right) & \text{if } 0 \le |\vec{r}| \le 2h \\ 0 & \text{else} \end{cases}.$$

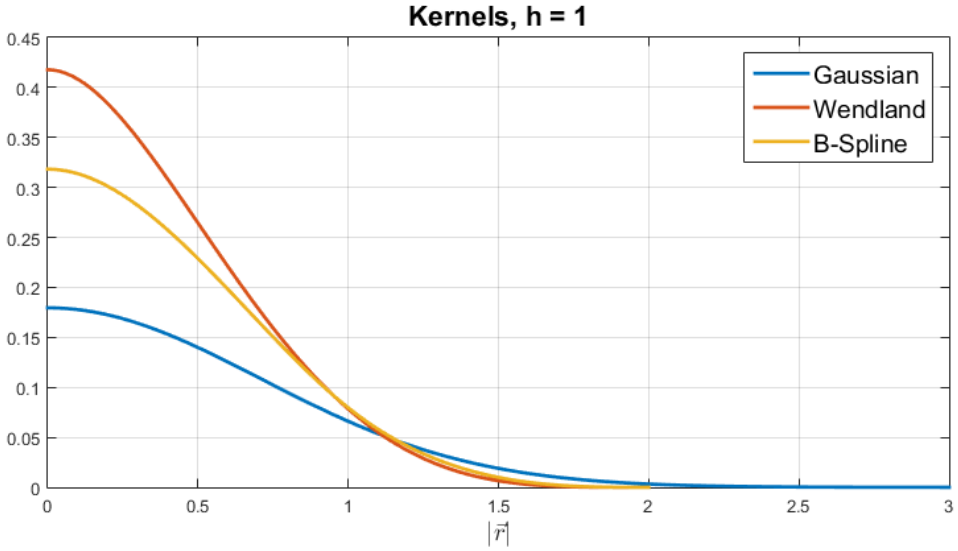
The Wendland kernel is well-suited as an SPH kernel since it is compactly supported and based on only one algebraic equation. The compact support decreases the number of neighbors necessary for the evaluation of (1.2) which, consequently, reduces the overall complexity of the computation. The simple representation avoids heavy branching, which might lead to performance loss when using for example Graphic Processing Units (GPUs) for computation.

3-degree B-spline

B-splines are widely used kernel functions for the SPH method [28, 29, 44]. An example for such a B-spline is the 3-degree B-spline defined as follows

$$W_h(\vec{r}) := \frac{1}{\pi h^3} \begin{cases} 1-\frac{3}{2}\left(\frac{|\vec{r}|}{h}\right)^2+\frac{3}{4}\left(\frac{|\vec{r}|}{h}\right)^3 & \text{if } 0 \le |\vec{r}| \le h, \\ \frac{1}{4}\left(2-\left(\frac{|\vec{r}|}{h}\right)\right)^3 & \text{if } h < |\vec{r}| \le 2h, \\ 0 & \text{else.} \end{cases}$$

Figure 1.1: Plot of different kernel functions for $h = 1$.



2 Applying SPH

This chapter is about the application of the SPH formalism derived in Chapter 1 to the basic equations of fluid mechanics, i.e. the continuity equation (Section 2.1) and the equation of momentum (Section 2.2). Those equations are supposed to be granted and neither derived nor discussed in any detail in this thesis. For more information regarding these equations, see [14] or [44]. Section 2.3 then briefly discusses time stepping in SPH. In Section 2.4, different ways to enforce incompressibility and the resulting variants of SPH are presented. The last two sections in this chapter, Sections 2.5 and 2.6, cover some ways to handle boundaries with SPH and briefly discuss the implementation respectively.

2.1 Continuity Equation

Let $\Omega \subset \mathbb{R}^3$. Let $\rho : \Omega \times \mathbb{R}_+ \to \mathbb{R}$ be the density of the fluid and $\vec{v} : \Omega \times \mathbb{R}_+ \to \mathbb{R}^3$ the velocity. The *continuity equation* is then given by

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \vec{v}.$$

Using the results from Chapter 1, to discretize the right hand-side of the equation at an arbitrary but fixed point $\vec{x}_i \in \Omega$, yields

$$\frac{d\rho_i}{dt} \approx \left\langle \frac{d\rho_i}{dt} \right\rangle = -\rho_i \sum_{j=1}^{n} V_j \vec{v}_j \cdot \nabla W_{ij}. \tag{2.1}$$

In accordance with Chapter 1, the notations $\rho_i = \rho_i(t) = \rho(\vec{x}_i, t)$ and $\vec{v}_i = \vec{v}_i(t) = \vec{v}(\vec{x}_i, t)$ are used.

Equation (2.1) is rarely used in practice due to lack of symmetry, but a different approximation coming from the SPH operator (1.8) for $k = 0$. Namely,

$$\left\langle \frac{d\rho_i}{dt} \right\rangle = \sum_{j=1}^{n} m_j \vec{v}_{ij} \cdot \nabla W_{ij}, \tag{2.2}$$

where $m_j = m(\vec{x}_j)$ denotes the mass of particle j and $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$ the difference in velocity. Approximation of the left hand-side in (2.2) with forward differences yields

$$\rho_i(t + \Delta t) \approx \rho_i + \Delta t \sum_{j=1}^{n} m_j \vec{v}_{ij} \cdot \nabla W_{ij}. \tag{2.3}$$

This is a convenient formula to update the density in a time stepping algorithm with step size Δt. A more detailed introduction to the time stepping is given in Section 2.3.

In the SPH community, a different way to update the density is commonly seen and used. It can be derived by applying (1.2) directly to the density ρ. It yields

$$\rho_i \approx \langle \rho_i \rangle = \sum_{j=1}^{n} V_j \rho_j W_{ij} = \sum_{j=1}^{n} m_j W_{ij}. \tag{2.4}$$

The advantage of this formulation is that it does not depend on the velocity $\vec{v}$. On the other hand, it is not applicable to multiphase or free-surface flows since the number of particles considered in the sum on the right hand-side decreases close to the interface and the density is thus underestimated there.

2.2 Equation of Momentum

Similar to Section 2.1, this section shows how the SPH formalism can be used to discretize the equation of momentum in fluid mechanics. The equation is given in Lagrangian formulation by

$$\frac{d\vec{v}}{dt} = -\frac{1}{\rho}\nabla p + \nu \Delta \vec{v} + \vec{g},$$

where $\vec{v} = \vec{v}(\vec{x}, t)$ denotes the velocity, $\rho = \rho(\vec{x}, t)$ the density, $p = p(\vec{x}, t)$ the pressure, ν the kinematic viscosity, and $\vec{g}$ external accelerations like, for example, gravitation. The gradient ∇p is to be understood with respect to the space variable $\vec{x}$ and the Laplacian of the vector valued velocity $\Delta \vec{v}$ is meant to be taken component wise. For a proper derivation of this equation and more information regarding it, see [44] or [14].

The first step is to spatially discretize the equation similar as already presented on the basis of the continuity equation. This can by done by evaluating the equation at a given discretization point $\vec{x}_i$ and applying the SPH discretization (1.2) to the right hand-side, which yields

$$\left\langle \frac{d\vec{v}_i}{dt} \right\rangle = \vec{g}_i - \frac{1}{\rho_i} \sum_{j=1}^{n} V_j p_j \nabla W_{ij} + \nu_i \sum_{j=1}^{n} V_j \vec{v}_j \Delta W_{ij}.$$

The next step is the discretization of the time derivative on the left hand-side. The simplest way to do this is by forward differences, resulting in

$$\vec{v}_i(t + \Delta t) \approx v_i + \Delta t \left(\vec{g}_i - \frac{1}{\rho_i} \sum_{j=1}^{n} V_j p_j \nabla W_{ij} + \nu_i \sum_{j=1}^{n} V_j \vec{v}_j \Delta W_{ij} \right). \tag{2.5}$$

While this is a proper discretization, it is not recommended for practical usage since the force terms are not symmetric and therefore violate Newton's third law of motion [12, Section 2.1]. This can be seen more clearly by multiplication with m_i and using Newton's second law of motion [12, Section 2.1]. Skipping the brackets $\langle\cdot\rangle$, indicating the approximation, to increase readability, results in the following

$$\begin{aligned}
\vec{F}_i &= m_i \vec{a}_i \\
&= m_i \frac{d\vec{v}_i}{dt} \\
&= m_i \vec{g}_i - m_i \frac{1}{\rho_i} \sum_{j=1}^{n} V_j p_j \nabla W_{ij} + m_i \nu_i \sum_{j=1}^{n} V_j \vec{v}_j \Delta W_{ij} \\
&= \vec{F}_i^{ext} - \sum_{j=1}^{n} V_i V_j p_j \nabla W_{ij} + \sum_{j=1}^{n} m_i \nu_i V_j \vec{v}_j \Delta W_{ij} \\
&= \vec{F}_i^{ext} - \sum_{j=1}^{n} \vec{F}_{i\leftarrow j}^{p} + \sum_{j=1}^{n} \vec{F}_{i\leftarrow j}^{\nu},
\end{aligned}$$

where $\vec{F}_i^{ext} := m_i \vec{g}_i$ denotes the external forces acting on particle i,

$$\vec{F}_{i\leftarrow j}^{p} := V_i V_j p_j \nabla W_{ij},$$

denotes the pressure force acting on particle i induced by particle j, and

$$\vec{F}_{i\leftarrow j}^{\nu} := m_i \nu_i V_j \vec{v}_j \Delta W_{ij},$$

the viscous forces acting on particle i induced by particle j. According to Newton's third law of motion, one expects the anti-symmetry $\vec{F}_{i\leftarrow j} = -\vec{F}_{j\leftarrow i}$ for all particles i and j to hold true for all forces $\vec{F}$. Obviously, this is in general not the case in the stated approximation. It makes it necessary to modify it which is discussed in the following.

2.2.1 Pressure Term

Using (1.5) for $k = 1$ results in an anti-symmetrical approximation for the pressure term

$$\left\langle \frac{\nabla p_i}{\rho_i} \right\rangle = \sum_{j=1}^{n} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}.$$

While this formulation is widely used in the SPH community, it is not very well-suited for multiphase flows as it was shown in [6]. The reason is that the used SPH operator (1.5) uses the gradient of the density $\nabla\rho$ which is not defined at interfaces for multiphase flows. This problem occurs for all $k \neq 0$ in (1.5). In [6], it was

therefore proposed to use (1.5) for $k = 0$ which yields the following anti-symmetrical approximation which is also well-suited for multiphase flows

$$\left\langle \frac{\nabla p_i}{\rho_i} \right\rangle = \sum_j m_j \frac{p_i + p_j}{\rho_i \rho_j} \nabla W_{ij}.$$

2.2.2 Viscous Term

There are two problems with the straight-forward viscous term used in (2.5). The first is the above-mentioned lack of anti-symmetry. The second problem is the second derivative of the kernel function W. The problem does not lie in the computation of the second derivative, since it can be performed analytically, but rather in the numerical stability of the resulting method, because the second derivative of the kernel can lead to a high sensitivity to particle disorder [29]. Without going into further details, the following two approximations were suggested in [10] and [30] respectively.

$$\nu \Delta \vec{v} = \sum_j 8 m_j \frac{\nu_i + \nu_j}{\rho_i + \rho_j} \frac{\vec{v}_{ij} \cdot \vec{x}_{ij}}{\|\vec{x}_{ij}\|^2 + \eta^2} \nabla W_{ij},$$

$$\nu \Delta \vec{v} = \sum_j \alpha c h m_j \frac{\vec{v}_{ij} \cdot \vec{x}_{ij}}{\|\vec{x}_{ij}\|^2 + \eta^2} \nabla W_{ij},$$

where η is a regularization constant to avoid the singularity for $\|\vec{x}_{ij}\| = 0$ often chosen to be $0.1h$, h is the smoothing length, α is a tuning parameter and c the speed of sound.

One of the big disadvantages of SPH is that even for inviscid flows one needs to add an artificial viscosity term or dissipative term to the equation of momentum, similar to the viscous terms above, to stabilize the numerical scheme.

2.3 Time Stepping

In the application of the SPH formalism to the governing equations above, a simple forward difference scheme was used or, to be more precise, an explicit Euler scheme [45] to discretize the time derivatives on a single equation basis. It remains to discuss the time integration with respect to all the variables as a system. Namely, the positions $\vec{x}$, velocities $\vec{v}$ and densities ρ, which are the three parameters with time derivatives occurring in the governing equations. For a general discussion, let $A = A(t)$ be a scalar or vector valued function in a single variable t which solves the initial value problem

$$\begin{aligned} \dot{A} &= f(t, A), \\ A(t_0) &= A_0, \end{aligned}$$

for some function f. Let $t_0 < t_1 < \ldots < t_N$ be a discretization of the interval $[t_0, t_N]$ and $\Delta t_k := t_{k+1} - t_k$ for $k = 0, \ldots, N-1$. Then, the explicit Euler scheme is given by

$$A_{k+1} = A_k + \Delta t_k f_k,$$

where $A_k \approx A(t_k)$ denotes the computed approximation of $A(t_k)$ and $f_k := f(t_k, A_k)$ the evaluation of f at t_k and A_k. Using this scheme on all three aforementioned variables of interest ($\vec{x}$, $\vec{v}$, ρ) yields in total a scheme of the following design

$$\begin{aligned}
\vec{v}_{k+1} &= \vec{v}_k + \frac{\Delta t_k}{m} \vec{F}_k, \\
\vec{x}_{k+1} &= \vec{x}_k + \Delta t_k \vec{v}_k, \\
\rho_{k+1} &= \rho_k + \Delta t_k f_k^\rho,
\end{aligned}$$

where f_k^ρ is the density update of ones choice, for example (2.2) or (2.4), at time t_k. One can see that the three equations are independent of each other and the order in which they are solved does not matter. It is a so-called *fully explicit scheme*.

Another approach would be to use an implicit time stepping scheme like the implicit Euler scheme

$$A_{k+1} = A_k + \Delta t_k f_{k+1}$$

for time integration. Again, applying this to the three variables of interest yields

$$\begin{aligned}
\vec{v}_{k+1} &= \vec{v}_k + \frac{\Delta t_k}{m} \vec{F}_{k+1}, \\
\vec{x}_{k+1} &= \vec{x}_k + \Delta t_k \vec{v}_{k+1}, \\
\rho_{k+1} &= \rho_k + \Delta t_k f_{k+1}^\rho.
\end{aligned}$$

In this formulation, all parameters depend on each other, since F_{k+1} and f_{k+1}^ρ depend on the positions $\vec{x}_{k+1}$. Updating the values requires to solve a nonlinear system in each time step which is not recommended due to numerical efficiency. It is a so-called *fully implicit scheme*.

The last possibility is to use a mixture of explicit and implicit schemes on the different equations. Applying the explicit scheme to the velocity equation and the implicit scheme to the positions and the densities results in the following so-called first order *semi-implicit scheme*

$$\begin{aligned}
\vec{v}_{k+1} &= \vec{v}_k + \frac{\Delta t_k}{m} \vec{F}_k, \\
\vec{x}_{k+1} &= \vec{x}_k + \Delta t_k \vec{v}_{k+1}, \\
\rho_{k+1} &= \rho_k + \Delta t_k f_{k+1}^\rho.
\end{aligned}$$

It was shown in [44] that this last scheme is to prefer since it is time reversible, which can be seen by swapping the stepping indices k with $k+1$, and therefore conserves the energy.

So far, only first order schemes were used. Obviously, one can follow the same thoughts using higher order schemes which would then result in, for example, the second order *Störmer-Verlet scheme* [46, Section 17.4]

$$\begin{aligned}
\vec{x}_{k+\frac{1}{2}} &= \vec{x}_k + \frac{\Delta t_k}{2}\vec{v}_k, \\
\vec{v}_{k+1} &= \vec{v}_k + \frac{\Delta t_k}{m}\vec{F}_{k+\frac{1}{2}}, \\
\vec{x}_{k+1} &= \vec{x}_{k+\frac{1}{2}} + \frac{\Delta t_k}{2}\vec{v}_{k+1}, \\
\rho_{k+1} &= \rho_k + \Delta t_k f^{\rho}_{k+1},
\end{aligned}$$

which was shown to be more stable [31, 44] but obviously requires a higher computational effort compared to the first order schemes due to the additional computation of an intermediate position $\vec{x}_{k+\frac{1}{2}}$.

2.4 Incompressibility

Enforcing incompressibility is an essential part of realistic and physical correct simulations of incompressible flows. There are different approaches to achieve incompressibility and, depending on the chosen method, it is one of the most expensive steps in a SPH simulation. The common basic idea of all methods is to compute the pressure of each particle, necessary to enforce incompressibility, from the density. To increase readability, the brackets $\langle\cdot\rangle$, indicating the approximations, are dropped in the following.

Before discussing the incompressibility in more detail, the following exemplary SPH Algorithm 1, describing a single time step with step size Δt, sums up the work so far. Please note that Algorithm 1 is just an example for a basic SPH algorithm. It can be modified by using a different approximation for the density computation as described in Section 2.1 or a different time stepping scheme to compute $\vec{v}_i(t+\Delta t)$ and $\vec{x}_i(t+\Delta t)$ as described in Section 2.3. Depending on the way the pressure is computed, the order in which the quantities are computed might change or additional loops need to be introduced.

In the following, the different algorithms describing the methods to enforce incompressibility are based on Algorithm 1. The pseudo codes that are shown here in the following for the different ways to enforce incompressibility are just examples and can be modified similarly as just described for the basic SPH algorithm. According to [16], the approaches to enforce incompressibility can be classified into four categories. Namely

- Non-iterative equation of state (EOS) solvers,
- Non-iterative EOS solvers with splitting,

Algorithm 1 Basic SPH

for all particle i **do**
  $\rho_i = \sum_j m_j W_{ij}$ *//as in* (2.4)
  Compute pressure p_i *//Enforce incompressibility*
end for
for all particle i **do**
  $\vec{F}_i^{press} = -\frac{m_i}{\rho_i}\nabla p_i$ *//see Section 2.2*
  $\vec{F}_i^{\nu} = -\nu\Delta\vec{v}_i$ *//see Section 2.2*
  $\vec{F}_i^{ext} = m_i\vec{g}$
end for
for all particle i **do**
  $\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{\Delta t}{m_i}\left(\vec{F}_i^{press} + \vec{F}_i^{\nu} + \vec{F}_i^{ext}\right)$
  $\vec{x}_i(t+\Delta t) = \vec{x}_i(t) + \Delta t\vec{v}_i(t+\Delta t)$
end for

- Iterative EOS solvers with splitting,
- Pressure projection solvers.

Be aware, that there are many ways to classify the variants of SPH and that there is no common classification system available in the literature. For example, [40] only separates the different SPH methods into weakly compressible and truly incompressible SPH methods. The classification proposed in [16], which is also used here, is restrictive and not all SPH methods can be classified with it. In addition, it only considers some very rough classification depending on the way the pressure is computed and does not consider other possible time stepping schemes, density computations, boundary representations, etc. Depending on how these things are employed, the basic Algorithm 1 might change significantly. Nevertheless, this classification system is a good first step to show the wide variety of different SPH methods and gives a good introduction to them.

2.4.1 Non-iterative EOS solvers

Non-iterative EOS solvers are based on computing the pressure p_i of particle i from an equation of state (EOS) relating the pressure with the density. There are different equations of state used, like

$$p_i = c_s^2\rho_i$$

proposed in [21], where c_s denotes some artificial speed of sound, or

$$p_i = k\left(\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1\right), \tag{2.6}$$

as in [30], where k and γ are constant parameters and ρ_0 is the rest density of the fluid. Algorithm 1 is an example for such a non-iterative EOS solver, if the pressure in the algorithm is computed with an equation of state. According to [16], this class of SPH solvers is in general outperformed by iterative or pressure projection solvers. Nevertheless, due to their simple nature, they are easy to implement and also admit a wide range of modifications to introduce and model different physical properties.

2.4.2 Non-iterative EOS solvers with splitting

This class of SPH solvers is also based on an EOS as described in the previous Section 2.4.1. The difference is that the concept of *splitting* is in addition introduced. Splitting means that the forces acting on a particle are split into known forces (i.e. external forces like gravitation and force fields, surface tension, viscous forces, etc.) and the unknown pressure forces. From the known forces, the new velocities, positions, and the new densities are predicted and, based on those values, the necessary pressures to resolve the density errors, are computed. Algorithm 2 is an example for such a non-iterative EOS solver with splitting.

Algorithm 2 Non-iterative EOS with splitting

for all particle i **do**
  Compute all non-pressure forces $\vec{F}_i^{ext}$
  predict velocity $\vec{v}_i^* = \vec{v}_i + \frac{\Delta t}{m_i}\vec{F}_i^{ext}$
end for
for all particle i **do**
  predict density $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j m_j \left(\vec{v}_i^* - \vec{v}_j^*\right) \cdot \nabla W_{ij}$ *//see Section 2.1*
  Compute p_i from ρ_i^* using an EOS *//see Section 2.4.1*
end for
for all particle i **do**
  compute pressure force $\vec{F}_i^p = -\frac{m_i}{\rho_i}\nabla p_i$ *//see Section 2.2*
end for
for all particle i **do**
  $\vec{v}_i(t+\Delta t) = \vec{v}_i^* + \frac{\Delta t}{m_i}\vec{F}_i^{press}$
  $\vec{x}_i(t+\Delta t) = \vec{x}_i(t) + \Delta t \vec{v}_i(t+\Delta t)$
end for

According to [16], there is no comparison between non-iterative EOS solver with and without splitting in literature yet (2014). As of my knowledge, this is still true (2018). The reason might be - to the best of my knowledge - that nobody actually uses non-iterative EOS solvers with splitting in practice. Nevertheless, it is a good way to introduce the idea of splitting which is also used in the following categories of SPH methods.

2.4.3 Iterative EOS solvers with splitting

As the name suggests, the class of iterative EOS solvers with splitting also use the idea of splitting the pressure and the non-pressure forces as proposed in Section 2.4.2. In contrast to the methods presented so far, this class iteratively uses the computed pressures to predict the positions and velocities again and recompute the pressures until a certain maximum density fluctuation is reached. Algorithm 3 is an example of such a procedure. An obvious disadvantage of these SPH methods is the higher

Algorithm 3 Iterative EOS with splitting

for all particle i **do**
  Compute all non-pressure forces $\vec{F}_i^{ext}$
  predict velocity $\vec{v}_i^* = \vec{v}_i + \frac{\Delta t}{m_i}\vec{F}_i^{ext}$
  predict position $\vec{x}_i^* = \vec{x}_i + \Delta t\vec{v}_i^*$
end for
while Density fluctuations too big **do**
  for all particle i **do**
    predict density $\rho_i^* = \sum_j m_j W_{ij}^*$ *//as in* (2.4), $W_{ij}^* := W(\vec{x}_i^* - \vec{x}_j^*)$
    Compute p_i from ρ_i^* using an EOS *//see Section 2.4.1*
  end for
  for all particle i **do**
    compute pressure force $\vec{F}_i^p = -\frac{m_i}{\rho_i}\nabla p_i$ *//see Section 2.2*
    update predicted velocity $\vec{v}_i^* = \vec{v}_i^* + \frac{\Delta t}{m_i}\vec{F}_i^p$
    update predicted position $\vec{x}_i^* = \vec{x}_i^* + \frac{\Delta t^2}{m_i}\vec{F}_i^p$
  end for
end while
for all particle i **do**
  $\vec{v}_i(t + \Delta t) = \vec{v}_i^*$
  $\vec{x}_i(t + \Delta t) = \vec{x}_i^*$
end for

computational effort necessary to compute each time step. In addition, the code is more difficult to parallelize than for non-iterative solvers. Another drawback of this class is that there is no convergence result on the inner iteration for determining the pressures, at least to my knowledge. Nevertheless, the advantages of iterative solvers include bigger time steps than for non-iterative solvers, which outweigh the higher computational costs for each time step and therefore results in a possible speed up of multiple magnitudes compared to non-iterative solvers [7]. In addition, due to the stricter enforcement of the incompressibility, the resulting density field is closer to the rest density than in non-iterative solvers.

2.4.4 Pressure Projection

The idea of pressure projection methods is to solve a Pressure Poisson Equation (PPE) resulting from the non-pressure forces, i.e.

$$\frac{d}{dt}\vec{v}^* = \vec{g} + \nu\Delta\vec{v}$$

and

$$\frac{d}{dt}\vec{v} = \vec{g} + \nu\Delta\vec{v} - \frac{1}{\rho}\nabla p = \frac{d}{dt}\vec{v}^* - \frac{1}{\rho}\nabla p.$$

Discretizing the time derivative on the left hand-side and applying the divergence on both sides, using $\nabla \cdot \vec{v} = 0$ from the continuity equation for incompressible fluids and substituting the predicted velocitiy $\vec{v}^*$, yields

$$\frac{\rho_0}{\Delta t}\nabla \cdot \vec{v}_i^* = \nabla^2 p_i. \tag{2.7}$$

Or, when substituting the left hand-side with the continuity equation,

$$\begin{aligned} \frac{\rho_0 - \rho_i^*}{\Delta t^2} &= \nabla^2 p_i && \text{in } \Omega, \\ \frac{\partial p_i}{\partial \vec{n}} &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{2.8}$$

where $\frac{\partial}{\partial \vec{n}}$ denotes the normal derivative. The solution of (2.8) is then the pressure necessary to resolve the density fluctuations. Solving (2.8) is the biggest drawback of pressure projection methods since it takes in general considerable computational effort. In addition, it is not that easy to parallelize and, as far as I know, there are no convergence results regarding a discretization of (2.8) with the SPH formalism. In practice, when discretizing (2.8), the resulting linear system of equations is represented by a sparse matrix due to the compact supports of the kernel functions used, and the solver can be parallelized effectively as was shown in [15]. Algorithm 4 is an example algorithm for such a pressure projection method.

2.5 Boundaries

This section is about the representation of boundaries in SPH which is in general not an easy topic and many different ways to do this were developed in the past couple of decades. It is not far fetched to claim that basically every publication covering SPH dedicates a section to handling the boundaries. In Section 1.5.1 it was shown that there is a problem when the support of the kernel function intersects the boundary. The discretization performed in SPH shown in Section 1.3 might be inaccurate close to the boundary due to missing particles, and therefore supporting points, for the

Algorithm 4 Pressure Projection

for all particle i **do**
  Compute all non-pressure forces $\vec{F}_i^{ext}$
  predict velocity $\vec{v}_i^* = \vec{v}_i + \frac{\Delta t}{m_i}\vec{F}_i^{ext}$
end for
for all particle i **do**
  predict density $\rho_i^* = \sum_j m_j W_{ij} + \Delta t \sum_j m_j \left(\vec{v}_i^* - \vec{v}_j^*\right) \cdot \nabla W_{ij}$ *//see Section 2.1*
end for
solve PPE (2.8)
for all particle i **do**
  compute pressure force $\vec{F}_i^p = -\frac{m_i}{\rho_i}\nabla p_i$ *//see Section 2.2*
end for
for all particle i **do**
  $\vec{v}_i(t+\Delta t) = \vec{v}_i^* + \frac{\Delta t}{m_i}\vec{F}_i^p$
  $\vec{x}_i(t+\Delta t) = \vec{x}_i + \Delta t \vec{v}_i(t+\Delta t)$
end for

numerical integration. In addition to the aforementioned difficulties, another crucial problem arises. Namely, how to ensure that the walls are not penetrated by fluid particles. In the following, to reduce the complexity of the topic, only the treatment of solid wall boundaries is considered.

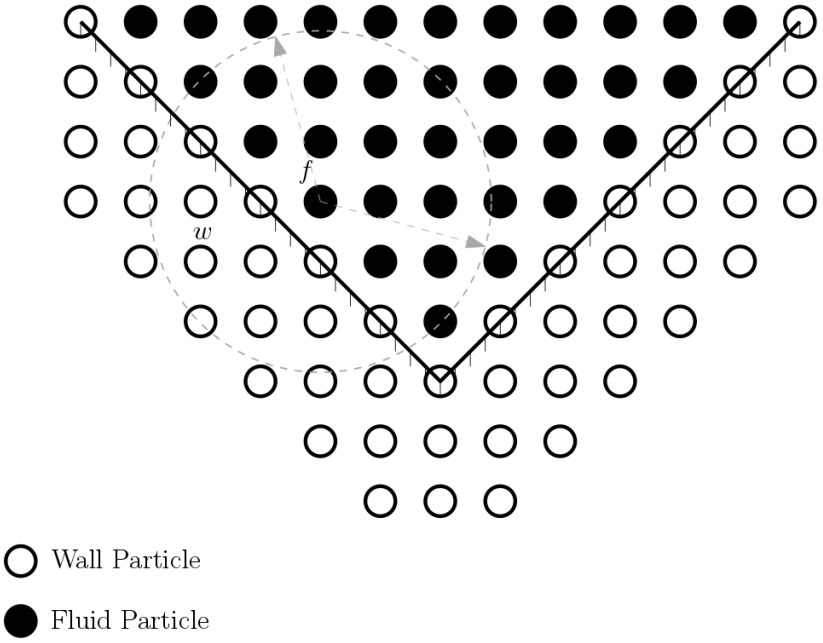
According to [36], the different methods to treat these wall boundaries can be separated into two groups. First, filling the walls with boundary or wall particles which are used to ensure a covered support of the kernel functions close to the boundary. These particles are often referred to as *dummy* or *ghost* particles [22, 33, 36, 37]. Second, introducing artificial forces to cover for the surface integral (compare Section 1.2) which does not vanish and to guarantee non-penetrability of the wall [18].

2.5.1 Dummy Particles

The dummy particle approach is based upon the idea to not just sample the fluids with particles but also the solids. This can be done either on the fly during the computation or in advance, before the computation. Sampling particles on the fly has in general the advantages that non-penetrability of the boundaries can be guaranteed more easily and that boundary conditions (e.g. no-slip or free-slip conditions) can be enforced more accurately. The disadvantages of this approach include comparably higher computational costs and that it is difficult to implement for complex shaped walls. The advantages of the pre-sampled methods are that it is fairly simple to implement, that it is considerably fast to compute and that even complex structures can be represented with it. The main disadvantages of this approach is that one needs

in general multiple layers of dummy particles (in general 2 or 3 layers) to be able to guarantee non-penetration of the boundary by fluid particles and to get correct approximations of the physical quantities. The consequence of this is that thin structures are problematic since they can only be represented with a sufficiently high resolution. Figure 2.1 shows a sketch of a typical dummy particle wall.

Figure 2.1: Wall sampled with dummy particles.



Adami Boundary Condition

There are many ways how to use dummy particles to enforce impenetrability of the walls, slip conditions, no-slip conditions, moving boundaries, etc. One method, which uses dummy particles to represent the boundary, was presented in [36] by Adami et al. It is therefore called the *Adami Boundary Condition*. It is explained in the following since it is the method of choice for the simulations performed in Section 5.

First, the solids are sampled with multiple layers of equidistantly placed particles, with placement distance Δr as depicted in Figure 2.1. Each particle gets an initial mass according to the rest density ρ_0 and the placement distance Δr. The idea of the Adami Boundary Condition is then to extrapolate the pressure p_w and the density ρ_w of wall particles w from the surrounding fluid particles after the initial state. For that, an equilibrium of forces is used on the boundary, assuming that the solid is not moving,

$$0 \stackrel{!}{=} \frac{d\vec{v}_f}{dt} = -\frac{\nabla p_f}{\rho_f} + \vec{g},$$

where $\vec{v}$, p, ρ and $\vec{g}$ denote the velocity, pressure, density and external accelerations respectively and the subscript f indicates that all the quantities are for fluid particles. Reordering the terms and integrating along the line between a fluid and wall particle yields

$$\int \nabla p \cdot d\vec{l} = \rho_f \int \vec{g} \cdot d\vec{l}$$

and consequently

$$p_w = p_f + \rho_f \vec{g} \cdot \vec{x}_{wf},$$

where the subscript w denotes the wall particles and $\vec{x}_{wf} = \vec{x}_w - \vec{x}_f$ denotes the distance vector between w and f. Note that the equation is for a single fluid particle. Summation over all fluid particles and normalization with the kernel then results in

$$p_w = \frac{\sum\limits_f p_f W_{wf} + \vec{g} \cdot \sum\limits_f \rho_f \vec{x}_{wf} W_{wf}}{\sum\limits_f W_{wf}}$$

for the extrapolation of the wall pressure from the fluid pressure.

The next step is the extrapolation of the wall particle density ρ_w from the fluid particles' densities. The method proposed in [36] was originally for the usage with an equation of state, like (2.6). It is therefore possible to compute the density from this equation of state similar to

$$\rho_w = \rho_{0,f} \left(\frac{p_w}{k} + 1 \right)^{\frac{1}{\gamma}}.$$

In [36] both, free-slip and no-slip boundary conditions, were discussed. To impose a free-slip boundary condition is considerably easily. It can be achieved by simply omitting the viscous interactions (compare equation (2.5)) with wall particles. Imposing a no-slip condition, which means $\vec{v} = 0$ at the boundary, is slightly more complex due to (1.2), which shows that the velocity $\vec{v}$ at any given point is the weighted average over the neighboring velocities. If all fluid particles close to the boundary have a non-zero velocity in the same direction and, according to the prior assumption, the boundary is not moving, then the weighted average over all velocities on the boundary, which includes fluid and boundary particles alike, cannot be 0. It is for this reason necessary to assign an artificial velocity $\tilde{\vec{v}}$ to the boundary particles used for the computation of the viscous forces. In [36], the following artificial velocity $\tilde{\vec{v}}_w$ was proposed

$$\tilde{\vec{v}}_w = -\frac{\sum_f \vec{v}_f W_{wf}}{\sum_f W_{wf}},$$

which corresponds to the negative average velocity of the neighboring fluid particles. Note that, for the sake of simplicity, this formulation is only valid for non-moving boundaries. In [36], a general approach for moving boundaries was covered.

Boundary Condition according to [33]

One of the main disadvantages of the Adami Boundary Condition is that thin structures pose a problem since multiple layers of particles are necessary to guarantee non-penetration of the wall and a fully sampled support of fluid particles close to the boundary. In addition, it was assumed that the initial particle placement is equidistant, which is, together with the necessity of having multiple particle layers, not always easy to guarantee, especially for complex shapes. There are some approaches how to solve this problem. One is the method presented in [33], where the density of the boundary particles is not fixed but extrapolated from the fluid particle and the density summation used does not depend on the mass of the neighboring particles.

$$\rho_{f,i} = m_{f,i} \sum_j W_{ij} + m_{f,i} \sum_k W_{ik},$$

where the subscript f indicates a fluid particle, the sum over j the sum over the neighboring fluid particles and the sum over k the sum over the neighboring boundary particles. The volume of a particle can be expressed in the following way

$$V_{b,i} = \frac{m_{b,i}}{\rho_{b,i}} = \frac{1}{\sum_k W_{ik}},$$

where the subscript b indicates a boundary particle. The final fluid particle density is then computed in the following way

$$\rho_{f,i} = m_{f,i} \sum_j W_{ij} + \sum_k \psi_{b,k}(\rho_{0,i}) W_{ik},$$

with $\psi_{b,k}(\rho_{0,i}) = \rho_{0,i} V_{b,k}$, where $\rho_{0,i}$ denotes the rest density of particle i. In [33], it was shown that this approach is applicable for thin structures and even fluid structure interactions with adequate force term formulations.

2.5.2 Boundary Forces

The second approach to enforce boundary conditions, beside the dummy particle approach presented in Section 2.5.1, is the usage of artificial forces. The advantage of this approach compared to the dummy particles is that, in general, one only requires a single layer of particles to represent the boundary [36, 44]. Disadvantages include that the numerical integration is inaccurate due to missing particles close to the boundary, that particles tend to stick to the surface because of it and that comparably large forces can occur on the boundary to ensure non-penetration which, consequently, leads to much smaller time steps [33]. There are many different ways [26, 31, 42] to enforce boundary conditions with this method, but they shall not be subject of this thesis.

2.6 Implementation

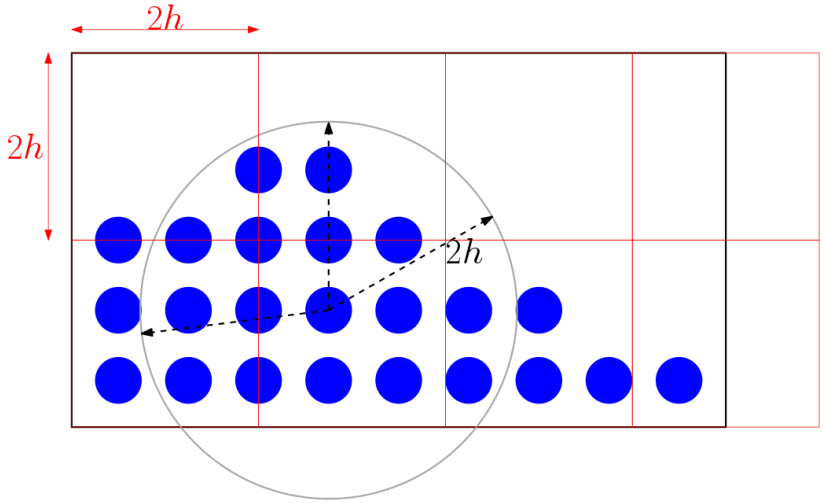
On of the biggest advantages of SPH compared to common mesh-based methods, such as Finite Volume Method (FVM) or Finite Element Method (FEM), is that it is comparably simple and, consequently, easy to implement. Nevertheless, there are some details which require special attention and were not mentioned so far. One is how to find neighbors. From Sections 1 and 2, one can see that SPH requires the evaluation of sums over neighboring particles. Since an evaluation against every particle would lead to a quadratic complexity, it is thus necessary to have a closer look at the *neighborhood search*. Hence, the following Section 2.6.1 introduces the concepts used for that.

To be able to use SPH for bigger simulations and more and more elements to increase the accuracy and resolution, it is necessary to parallelize the code and harness the full computational power of modern computer systems. In the last decade, GPU (graphics processing unit) computing became popular and, due to the simple basic algorithm and comparably small interactions in terms of memory access, SPH is well-suited for it. Section 2.6.2 covers this in detail.

2.6.1 Neighbor Search

Many computations in SPH are based on summations over neighboring particles. This means that for each particle i one needs to find all neighboring particles. Of course, one can follow a straight forward approach and just test every particle against every particle. The obvious disadvantage of this method would be the resulting quadratic complexity. To avoid this, kernel functions with compact support were introduced in Section 1.5. Additionally, it is also necessary to find all the particles within the compact support fast to avoid a quadratic complexity. In SPH this is achieved with a so-called *search grid*. It is a regular three-dimensional grid which covers the whole computational domain, where each grid cell has edge length of the kernel support radius. Each grid cell needs to know the neighboring grid cells. One can then perform a bucket sort algorithm [41, Section 8], or some other linear sorting algorithm, to sort all particles according to the grid cells they are in. According to the choice of the size of the grid cells, one can then find all neighbors within the kernel support radius just looking through the cell the particle i is in and all the neighboring cells of it. The great advantage of this method is that the number of neighbors, one needs to check for each particle, is independent of the total number of the particles in the system. In addition, the sorting can be performed in linear time and therefore does not impede the overall complexity. The disadvantage is the slightly higher memory consumption due to the necessity of storing the search grid. Figure 2.2 illustrates this search grid in 2D. The red quads indicate the grid cells, the black rectangle the domain, the blue spheres the particles and the gray circle the kernel support. One can also see that the search grid does not necessarily need to coincide with the domain, but needs to at least include the domain for this approach to work.

Figure 2.2: 2D sketch of neighbor search grid.



2.6.2 GPU Computing

In the last years, GPU computing got quite popular for high performance computing (HPC). This is due to the much better price to computation power ratio as shown in figure 2.3, where a modern Intel eon E5-2698-v4 CPU is compared to a NVidia GeForce GTX 1080 GPU in terms of price and computation power. The data for figure 2.3 were retrieved from [3] and [4]. It can easily be seen that in terms of raw computation power, GPUs outperform CPUs vastly while maintaining a much better price to computation power ratio. The disadvantages of GPU computing include the

	Intel Xeon E5-2698v4	**NVidia GeForce GTX 1080**
Price	3145.21€[a]	599.00€
Cores	20	2560
Frequency	2.2 GHz	1.607 GHz
GFLOPS	352	8873
Memory	up to 1.54 TB	8 GB
Price/GFLOP	8.94€	0.0675€

[a]Price taken from [2]

Figure 2.3: Comparison between high-end CPU and GPU.

difficult implementation compared to CPU computing, which stems from the difficulty of parallelizing an algorithm efficiently for thousands of independent threads and the more complex memory management, and the significantly smaller memory available. Additionally, the performance of GPUs is only that high for working with single precision. It drops significantly using double precision. Despite these disadvantages, GPU

computing got popular in the SPH community [42]. Due to its simplicity and low memory requirements, SPH is well-suited for GPU computing and the aforementioned disadvantages are not that severe with SPH. The difficulty of the implementation can be reduced using for example the NVidia parallel computing platform for GPUs, called CUDA [1].

Nevertheless, there are some difficulties with SPH on GPUs, especially when one wants to use multiple GPUs or even multiple nodes. While this is a challenging problem, there are solutions already available and presented to the public [11, 20].

3 PCISPH

A variant of SPH is the so-called *predictive-corrective incompressible smoothed particle hydrodynamics* (PCISPH) method. It was first introduced in 2009 by Solenthaler and Parajola [7] and, basically, belongs to the iterative EOS solvers with splitting, as described in Section 2.4.3. The method already proves that the classification system is by far not perfect since PCISPH can also be seen as a Pressure Projection Method as in Section 2.4.4. But due to the strong assumptions made during the derivation, which is presented in the following, it is not really solving the PPE anymore and therefore categorized as an iterative EOS solver with splitting.

The idea of the method is to predict the pressures and the densities based on the known forces and then correct the pressures until a given incompressibility condition is fulfilled. In Algorithm 5, one can see that in the while-loop the pressure p_i of each particle i is corrected according to the predicted positions $\vec{x}_i$. This correction depends on the function f which is defined as

$$f(\rho^*_{err,i}) := \frac{\rho^*_{err,i}\rho_0^2}{\Delta t^2 m^2 \left(\sum_j \nabla W_{0j} \cdot \sum_j \nabla W_{0j} + \sum_j (\nabla W_{0j} \cdot \nabla W_{0j}) \right)},$$

where W_{0j} is the kernel function for a prototype particle with fully filled neighborhood. This means that the sums in the denominator are constant and can therfore be precomputed. Note that this formula is only valid for constant mass $m = m_i$, for all particles i. For a proper derivation of f, see [7] or Section 3.1.

While a single time step in PCISPH is much more expensive to compute than a time step for a EOS based SPH variant, like *weakly compressible SPH* (WCSPH), it can admit a much bigger time step. Tests show, that this outweighs the additional computational effort per time step and that PCISPH is in many cases much faster than WCSPH. Another drawback of PCISPH is that adaptive time stepping is only conditionally possible due to the pressure depending on the time step size Δt. This is especially problematic for simulations where shocks occur.

As far as I know, there is no work on the convergence of the above described predictive-corrective scheme so far. A thorough investigation of it might lead to new insights regarding PCISPH and SPH in general. Additionally, the derivation of the function f as proposed in [7] is based on some assumptions like constant mass m for all particles and introduces some approximation errors which are not investigated further in [7] and, according to my knowledge, in no other publication.

Algorithm 5 PCISPH

for all particle i **do**
  Compute all non-pressure forces $\vec{F}_i^{ext}$
  Initialize pressure $p_i = 0$
  Initialize pressure force $\vec{F}_i^p = 0$
end for
while Density fluctuations too big **do**
  for all particle i **do**
    predict velocity $\vec{v}_i^*(t + \Delta t) = \vec{v}_i(t) + \frac{\Delta t}{m_i}\left(\vec{F}_i^p + \vec{F}_i^{ext}\right)$
    predict position $\vec{x}_i^*(t + \Delta t) = \vec{x}_i(t) + \Delta t \vec{v}_i^*(t + \Delta t)$
  end for
  for all particle i **do**
    predict density $\rho_i^* = \sum_j m_j W_{ij}^*$ *//as in* (2.4), $W_{ij}^* := W(\vec{x}_i^* - \vec{x}_j^*)$
    predict density fluctuation $\rho_{err,i}^* = \rho_i^* - \rho_0$
    update pressure $p_i += f(\rho_{err,i}^*)$
  end for
  for all particle i **do**
    compute pressure force $\vec{F}_i^p = -\frac{m_i}{\rho_i}\nabla p_i$ *//see Section 2.2*
  end for
end while
for all particle i **do**
  $\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + \frac{\Delta t}{m_i}\left(\vec{F}_i^p + \vec{F}_i^{ext}\right)$
  $\vec{x}_i(t + \Delta t) = \vec{x}_i(t) + \Delta t \vec{v}_i(t + \Delta t)$
end for

3.1 Derivation of Pressure Correction f

This section is about the derivation of the pressure correction scheme used in PCISPH. The derivation, which is presented in the following, is based closely on the derivation presented in the original publication [7].

The derivation starts from the mass summation approach (2.4) for the computation of the density. In addition, it is assumed that the masses m_j of all particles j are equal and constant $m_j = m$, which corresponds to a single-phase flow. Equation (2.4) therefore simplifies to

$$\rho_i(t) = m \sum_{j=1}^{n} W(\vec{x}_i(t) - \vec{x}_j(t)), \tag{3.1}$$

where the time dependency is stated explicitly. The goal, when enforcing incompressibility, is to enforce the density to be always the same or close to the rest density ρ_0 of the fluid. Consequently, the logically next step is to look at the future density $\rho_i(t + \Delta t)$ of each particle i. The application of equation (3.1) to $\rho_i(t + \Delta t)$,

$$\rho_i(t + \Delta t) = m \sum_{j=1}^{n} W(\vec{x}_i(t + \Delta t) - \vec{x}_j(t + \Delta t)),$$

shows that for the computation of the next density, knowledge about the next positions $\vec{x}_i(t + \Delta t)$ is required, which is an obvious problem. To solve that problem, we first split $\vec{x}_i(t + \Delta t)$ into the current position $\vec{x}_i$ and the displacement $\Delta\vec{x}_i$

$$\vec{x}_i(t + \Delta t) =: \vec{x}_i + \Delta\vec{x}_i,$$

using a notation where an evaluation at time t is not stated explicitly. Substituting this back into the above equation yields

$$\rho_i(t + \Delta t) = m \sum_{j=1}^{n} W(\vec{x}_i - \vec{x}_j + \Delta\vec{x}_i - \Delta\vec{x}_j). \tag{3.2}$$

The next step is to use a first order Taylor expansion of the kernel

$$W(\vec{x}_i - \vec{x}_j + \Delta\vec{x}_i - \Delta\vec{x}_j)$$

in the point $\vec{x}_i - \vec{x}_j$ which yields

$$W(\vec{x}_i - \vec{x}_j + \Delta\vec{x}_i - \Delta\vec{x}_j) \approx W(\vec{x}_i - \vec{x}_j) + \nabla W(\vec{x}_i - \vec{x}_j) \cdot (\Delta\vec{x}_i - \Delta\vec{x}_j)\,.$$

Note that this approximation is more accurate for small changes in the positions $\Delta\vec{x}_i$. Plugging the Taylor expansion back into (3.2) results in

$$\rho_i(t + \Delta t) = m \sum_{j=1}^{n} \left(W(\vec{x}_i - \vec{x}_j) + \nabla W(\vec{x}_i - \vec{x}_j) \cdot (\Delta\vec{x}_i - \Delta\vec{x}_j)\right).$$

One can see that the first term in the sum, together with the mass m, equals the density ρ_i. One can therefore define the change in density $\Delta\rho_i$ of particle i

$$\rho_i(t + \Delta t) - \rho_i(t) =: \Delta\rho_i = m \sum_{j=1}^{n} \nabla W(\vec{x}_i - \vec{x}_j) \cdot (\Delta\vec{x}_i - \Delta\vec{x}_j) \,.$$

Using the usual shortened notation for the kernel and rearranging terms results in

$$\Delta\rho_i = m \sum_{j=1}^{n} \nabla W(\vec{x}_i - \vec{x}_j) \cdot (\Delta\vec{x}_i - \Delta\vec{x}_j) = m\Delta\vec{x}_i \sum_{j=1}^{n} \nabla W_{ij} - m \sum_{j=1}^{n} \nabla W_{ij}\Delta\vec{x}_j. \quad (3.3)$$

The problem of finding the new density was reduced to finding the displacements $\Delta\vec{x}_i$ of all particles i, which can be computed knowing the forces acting on the particles. The only forces unknown are the pressure forces F_i^p, since they depend on the unknown pressures. So, one can assume that the particle position $\vec{x}_i$ are already known according to the known forces and that the particle displacements $\Delta\vec{x}_i$ are only induced by the pressure p. Using a Leap Frog integration scheme shows that the particle displacements $\Delta\vec{x}_i$ can be computed as follows from the pressure forces F_i^p

$$\Delta\vec{x}_i = \frac{\Delta t^2}{2} \frac{F_i^p}{m}. \quad (3.4)$$

With that, the problem changes to finding the pressure forces F_i^p for all particles i. In [7], the same pressure force formulation as described in Section 2.2.1 was used, namely

$$F_i^p = -m^2 \sum_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}.$$

Now, assuming that the densities ρ_i are approximately the rest density ρ_0 of the fluid for all particles i and that the pressures of the neighboring particles of particles i are approximately equal to the same pressure $\tilde{p}_i$, one gets

$$F_i^p = -m^2 \frac{2\tilde{p}_i}{\rho_0^2} \sum_j \nabla W_{ij}.$$

Substituting back to (3.4) yields

$$\Delta\vec{x}_i = -\frac{\Delta t^2 m \tilde{p}_i}{\rho_0^2} \sum_j \nabla W_{ij}.$$

With this, the unknown displacement $\Delta\vec{x}_i$ in (3.3) can be computed. The computation of the neighboring particles' displacement $\Delta\vec{x}_j$ in (3.3) can be done as described in the following.

Assume that the displacement of a neighboring particle $\Delta\vec{x}_j$ only depends on the current particle i,

$$\Delta\vec{x}_j = \Delta\vec{x}_{j\leftarrow i},$$

where the subscript notation on the right hand-side denotes the contribution of particle i to the displacement of particle j. This assumption is probably the most critical since it is in general wrong but necessary to avoid a linear system of equations and have independent equations for each particle. Without this assumption, it is still possible to separate the equations and fully parallelize the method, as it was shown in [15], but this would lead to a different variant of SPH, namely *implicit incompressible SPH* (IISPH), which is briefly explained in Section 4.2. With this assumption and the anti-symmetry of the gradient of the kernel, $\nabla W_{ji} = -\nabla W_{ij}$, one gets for the displacement of a neighboring particle

$$\Delta\vec{x}_j = \frac{\Delta t^2 m\tilde{p}_i}{\rho_0^2}\nabla W_{ij}.$$

Substituting everything back to (3.3) results in

$$\begin{aligned}\Delta\rho_i &= m\Delta\vec{x}_i\sum_{j=1}^{n}\nabla W_{ij} - m\sum_{j=1}^{n}\nabla W_{ij}\Delta\vec{x}_j \\ &= -\frac{\Delta t^2 m^2\tilde{p}_i}{\rho_0^2}\left(\sum_j \nabla W_{ij}\cdot\sum_j \nabla W_{ij} + \sum_j \nabla W_{ij}\cdot\nabla W_{ij}\right).\end{aligned}$$

Solving for $\tilde{p}_i$, introducing the density deviation to the rest density $\rho^*_{err,i} := \rho^*_i - \rho_0$, and precomputing the sums over the gradients of the kernel in the denominator according to the particle distribution around a prototype particle and denoting them with W_{0j}, gives the final formula for f as in [7]

$$\tilde{p}_i = \frac{\rho_0^2\rho^*_{err,i}}{\Delta t^2 m^2\left(\sum_j \nabla W_{0j}\cdot\sum_j \nabla W_{0j} + \sum_j \nabla W_{0j}\cdot\nabla W_{0j}\right)} =: f(\rho^*_{err,i}). \tag{3.5}$$

3.2 PCISPH in Engineering

In [7], PCISPH was developed for applications in computer graphics and there was no investigation regarding the accuracy and its applicability to engineering problems. Due to the increased performance compared to other SPH variants, like WCSPH, my colleagues at ESS - Engineering Software Steyr GmbH and I decided to further investigate into PCISPH and evaluate if the method is applicable to engineering problems. Consequently, the results presented in this section are based on the work of my colleagues Dr. Chong Peng, Dr Kamil Szewc, Dr Hui Cao, and myself and until now not available to the public.

3.2.1 Numerical Diffusion and Pressure Smoothing

According to our investigations, the basic PCISPH algorithm suffers from strong pressure oscillations. There are two reasons for this. First, equation (3.5) is basically an EOS similar to equation (2.6) but with adaptive stiffness constants. PCISPH therefore suffers from the same so-called *short length-scale noise* [32] as other EOS based SPH variants. Second, PCISPH uses a much larger time step than other EOS based SPH variants which further contributes to the first problem. We introduced two solutions to this problem: *pressure smoothing* and *numerical diffusion*.

The numerical diffusion term was taken as in [9] which changes the density update to

$$\rho_i(t+\Delta t) = \rho_i + \Delta t \left(\sum_j m_j \vec{v}_{ij} \cdot \nabla W_{ij} + \delta h c \sum_j \frac{2m_j \left(\rho_i - \rho_j\right) \vec{x}_{ij}}{\rho_j \left(\left\| \vec{x}_{ij} \right\|^2 + \eta^2 \right)} \cdot \nabla W_{ij} \right),$$

where (2.4) is used as basic update for the density, δ is a constant parameter, usually chosen 0.1, c is the speed of sound, and η a parameter to avoid a singularity at $\|\vec{x}_{ij}\| = 0$.

The pressure smoothing was done by replacing the computed pressure p_i for each particle i by a smoothed pressure $\overline{p}_i$ which is computed the following way

$$\overline{p}_i = \chi p_i + (1-\chi) \sum_j \frac{m_j}{\rho_j} p_j W_{ij},$$

where χ denotes a smoothing parameter in $[0,1]$ with $\chi = 1$ being equivalent to no smoothing at all and $\chi = 0$ to full smoothing of the pressure. This approach allows to get rid of pressure spikes in single particles and thus stabilizes the simulation.

3.2.2 Time Stepping

In Section 2, the time derivatives were approximated using a simple first order semi-implicit scheme. We switched to a second order Störmer-Verlet scheme as presented in Section 2.3. The disadvantage of using the Störmer-Verlet scheme is an increased computational effort, since an intermediate step needs to be computed. Nevertheless, the advantages of increased stability and accuracy are significant.

3.2.3 Boundary Handling

As boundary representation, the Adami Boundary Condition as presented in Section 2.5.1 is used. The problem with the original Adami Boundary Condition is that it requires an EOS to compute the density ρ_w of the a particles from the pressure p_w of a the particle. Since PCISPH does not have such an EOS, another way to compute

the density is necessary. We use the following formula to extrapolate the density ρ_w of a wall particle w from the densities ρ_f of the neighboring fluid particles

$$\rho_w = \frac{1}{\sum_f W_{wf}} \sum_f \rho_f W_{wf}.$$

It corresponds to the weighted average of the neighboring fluid particles' densities. Otherwise, the Adami Boundary Condition was employed as described in Section 2.5.1.

3.2.4 Multiphase PCISPH

The derivation shown in Section 3.1 was only done for single phase flows. In engineering, there are many cases where multiple fluids play an essential role. Therefore, a general numerical method needs to be able to handle these multiphase flows. As a consequence, I investigated how to generalize PCISPH, as presented in [7] and Section 3.1, to be able to handle multiphase flows. The general methodology was to follow the derivation presented in Section 3.1, identify potential problems and assumptions that might be restricting for multiphase flows, and fix them using solutions known for other SPH variants.

The first problematic assumption in Section 3.1 was the uniformity of the masses m_i of the particles. For the derivation of a general multiphase PCISPH method, it is necessary to assume that each particle i holds a mass m_i, which can vary between particles. Next, the choice of the density formulation (3.1) is potentially problematic. The obvious reason is that the formulation gives different values inside a fluid, where the smoothing radius does not intersect any boundaries, free surfaces or interfaces to other fluids, and close to the boundary of the fluid. To get rid of that problem, (2.3) is employed to update the density which is well-known to be better suited for multiphase flows. Using a semi-implicit time stepping scheme one gets for the positions $\Delta\vec{x}_i = \Delta t\vec{v}_i(t + \Delta t)$ and with it from (2.3) the following formula for the density error

$$\Delta\rho_i = \Delta t \sum_j m_j \vec{v}_{ij}(t + \Delta t) \cdot \nabla W_{ij} = \sum_j m_j \Delta\vec{x}_i \cdot \nabla W_{ij} - \sum_j m_j \Delta\vec{x}_j \cdot \nabla W_{ij}.$$

One can see that this is basically the same result as (3.3). The difference is that there is much more flexibility in the derivation since a different time stepping scheme would lead to a different formulation here and only one of the most basic one, a semi-implicit first order scheme, led to the same formulation as before. Following the same pattern as in Section 3.1, one gets

$$\Delta\vec{x}_i = \frac{\Delta t^2}{2} \frac{F_i^p}{m_i}$$

for the displacement of particle i according to the unknown pressure forces F_i^p acting on particle i. Previously the next step was to express the pressure forces. As was

discussed in Section 2.2.1, [6] and [40], the formulation used before is not well-suited for multiphase flows. Therefore, the following formulation for the pressure force is used

$$F_i^p = -m_i \sum_j m_j \frac{p_i + p_j}{\rho_i \rho_j} \nabla W_{ij}.$$

From here on, the derivation is similar to Section 3.1, just without the assumption that the densities are equal to the rest densities ρ_0. It results in the following updating rule for the pressure in the case of multiphase flows

$$\tilde{p}_i = \frac{\rho_i \rho^*_{err,i}}{\Delta t^2 \left(\sum_j \frac{m_j}{\rho_j} \nabla W_{ij} \cdot \sum_j m_j \nabla W_{ij} + m_i \sum_j \frac{m_j}{\rho_j} \nabla W_{ij} \cdot \nabla W_{ij} \right)} =: f(\rho^*_{err,i}),$$

where the sums in the denominator cannot be precomputed anymore since they depend on the masses and densities of the neighboring particles which might change according to the changes of the interfaces. This concludes the derivation and shows the changes necessary in the main algorithm to enable the computation of multiphase flows with PCISPH.

4 Other SPH Variants

SPH is a versatile method which is constantly adapted to new purposes, to simulate more and more complex physical problems both, in fluid and solid mechanics, to increase accuracy and performance and make the method more attractive for industrial applications. This development led to many different variants of SPH which are recognized in the SPH community. While this thesis is mainly focused on further developing one of the variants, PCISPH, it is not unimportant to have some general knowledge of other SPH variants currently in use.

4.1 WCSPH

The so-called *weakly compressible smoothed particle hydrodynamics* (WCSPH) method is the variant of SPH which is most widely known and used and it is also the first method of SPH for handling incompressible flows and should therefore not be missing in any work related to SPH. It belongs to the non-iterative EOS solvers as described in Section 2.4.1. The pressure p_i in Algorithm 1 is computed from an equation of state of the form (2.6). The rest of Algorithm 1 can be taken over one to one for WCSPH which is why it is not repeated at this place.

In WCSPH, every time step can be computed comparably fast and easily since the pressure of each particle only depends on a single algebraic equation, which does not depend on any values of other particles. The problems include the "correct" choice of the parameters γ and k in the EOS (2.6) to reach a sufficient stiffness and the comparably small time step necessary for a sufficiently stiff EOS. Consequently, this leads in general to higher computation times compared to other SPH variants despite the fast computation of each time step. WCSPH has the advantage that it can be adapted quite easily to simulate different physical behaviors including heat transfer [19], solid mechanics [34], fluid-structure interactions [33], geophysics [27], fracture simulation [8], and many more.

4.2 IISPH

A sub-variant of the pressure projection SPH methods is the so-called *implicit incompressible smoothed particle hydrodynamics* (IISPH) method which was first presented in [15]. This SPH method is interesting in the context of this thesis since it is the logical development of PCISPH and therefore explained in greater detail in the following.

In the derivation of the pressure correction of PCISPH in Section 3, two assumptions were used to greatly simplify the equations. The first was that the pressure p_i of a particle i equals the pressure of all neighboring particles. The second was that the displacement $\Delta\vec{x}_j$ of a neighboring particle j only depends on the pressure p_i of the current particle. These two assumptions simplify the problem from a system of equations to a single equation for each particle, which can be explicitly evaluated to update the pressure. Starting from (3.4), using the same pressure force formulation as in the standard PCISPH approach but without the two simplifications, then yields

$$\begin{aligned}\Delta\vec{x}_i &= -\frac{\Delta t^2}{2}\sum_j m_j\left(\frac{p_i}{\rho_i^2}+\frac{p_j}{\rho_j^2}\right)\nabla W_{ij}\\ &= \underbrace{\left(-\frac{\Delta t^2}{2}\sum_j \frac{m_j}{\rho_i^2}\nabla W_{ij}\right)}_{\vec{d}_{ii}} p_i + \sum_j \underbrace{-\frac{\Delta t^2}{2}\frac{p_j}{\rho_j^2}}_{\vec{d}_{ij}}\nabla W_{ij}\\ &= \vec{d}_{ii}p_i + \sum_j \vec{d}_{ij}p_j.\end{aligned}$$

Substituting this back into equation (3.3) yields

$$\Delta\rho_i = \sum_j m_j\left(\vec{d}_{ii}p_i + \sum_{j'}\vec{d}_{ij'}p_{j'} - \vec{d}_{jj}p_j - \sum_k \vec{d}_{jk}p_k\right)\nabla W_{ij}, \tag{4.1}$$

where the summation indices j and j' denote summation over neighboring particles of particle i and the summation index k denotes summation over the neighbors' neighbors. Theoretically, one can stop here and use this update formula to compute the pressures p_i to guarantee incompressibility with any solver for linear systems of equations. In practice, however, it is not recommended to do so since the computational effort is significantly higher than for comparable methods like PCISPH or WCSPH due to the nested sums. In addition, setting up the system of equations is difficult, memory, and time consuming, especially in a parallel manner. It was therefore proposed in [15] to use a weighted Jacobi method to solve the linear system.

Let a linear system of equations be given by

$$\sum_j a_{ij}p_j = \Delta\rho_i \qquad \text{for } i = 1\dots n,$$

then the weighted Jacobi algorithm yields

$$p_i^{l+1} = (1-\omega)\,p_i^l + \omega\frac{\Delta\rho_i - \sum_{j\neq i} a_{ij}p_j^l}{a_{ii}},$$

where $\omega \in (0,1)$ is some constant parameter. To be able to apply this solver, one requires knowledge about the diagonal elements a_{ii} and the sum over all the other

elements $\sum_{j \neq i} a_{ij} p_j^l$. Consequently, it is necessary to extract these values from (4.1). As a result, one gets

$$\Delta\rho_i = p_i \underbrace{\sum_j m_j \left(\vec{d}_{ii} - \vec{d}_{ji}\right) \nabla W_{ij}}_{a_{ii}} + \sum_j \left(\sum_{j'} \vec{d}_{ij'} p_{j'} - \vec{d}_{jj} p_j - \sum_{k \neq i} \vec{d}_{jk} p_k \right) \nabla W_{ij},$$

which results in the following weighted Jacobi iteration

$$p_i^{l+1} = (1-\omega)\, p_i^l + \omega \frac{1}{a_{ii}} \left(\Delta\rho_i - \sum_j \left(\sum_{j'} \vec{d}_{ij'} p_{j'}^l - \vec{d}_{jj} p_j^l - \sum_{k \neq i} \vec{d}_{jk} p_k^l \right) \nabla W_{ij} \right). \quad (4.2)$$

The final algorithm, as proposed in [15], is shown in Algorithm 6

Algorithm 6 IISPH

for all particle i **do**
  Compute density $\rho_i = \sum_j m_j W ij$
  Compute non-pressure forces F_i^{adv}
  Compute predicted velocity $\vec{v}_i^{\text{adv}} = \vec{v}_i + \Delta t \frac{F_i^{\text{adv}}}{m_i}$
  Compute $\vec{d_{ii}} = \Delta t^2 \sum_j -\frac{m_j}{\rho_i^2} \nabla W_{ij}$
end for
for all particle i **do**
  Compute predicted density $\rho_i^{\text{adv}} = \rho_i + \Delta t \sum_j m_j \vec{v}_{ij}^{\text{adv}} \nabla W_{ij}$
  Initialize pressure $p_i^0 = 0$
  Compute $a_{ii} = \sum_j m_j \left(\vec{d}_{ii} - \vec{d}_{ji} \right) \nabla W_{ij}$
  Compute $\vec{d_{ii}} = \Delta t^2 \sum_j -\frac{m_j}{\rho_i^2} \nabla W_{ij}$
end for
while Density fluctuations too big **do**
  for all particle i **do**
    Compute $\sum_j \vec{d}_{ij} p_j = -\Delta t^2 \sum_j \frac{m_j}{\rho_j^2} p_j^l \nabla W_{ij}$
  end for
  for all particle i **do**
    Compute p_i^{l+1} as in (4.2)
  end for
  l = l+1
end while
for all particle i **do**
  $\vec{v}_i(t + \Delta t) = \vec{v}_i^{\text{adv}} + \frac{\Delta t}{m_i} \vec{F}_i^p$
  $\vec{x}_i(t + \Delta t) = \vec{x}_i + \Delta t \vec{v}_i(t + \Delta t)$
end for

5 Numerical Tests

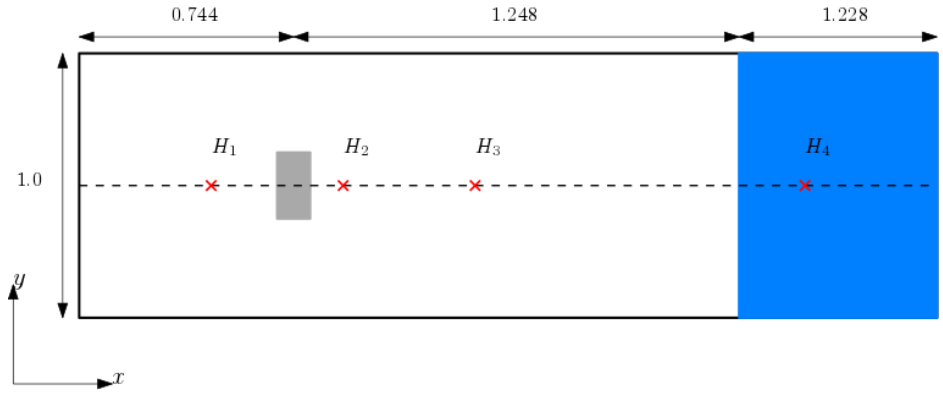
In this section, some numerical experiments are discussed to present the improvements described in Section 3 and the capabilities of SPH in general.

All tests were performed on a NVidia GeForce GTX 1080 graphics card.

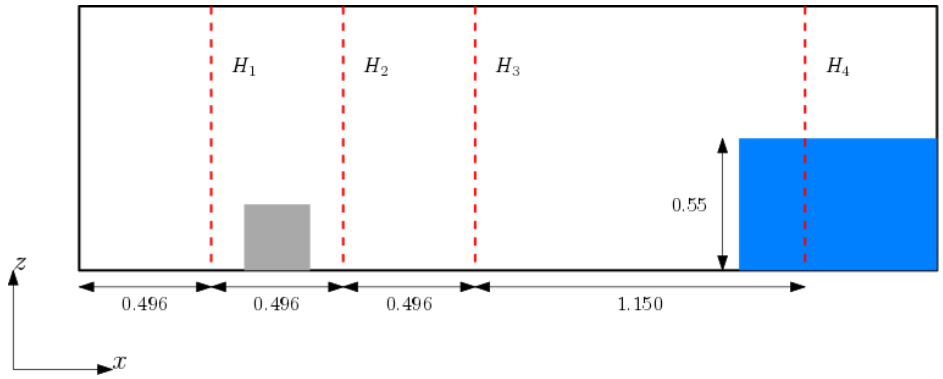
5.1 Dam Break

The first test case is a dam break scenario with an obstacle placed in the way of the fluid taken from [35]. It is a well known and often used scenario in the SPH community. The domain is sized $3.22m \times 1m \times 1m$. At the far end in x-direction, there is a cuboid shaped reservoir of liquid held by a gate. At $t = 0$, the gate is opened and the fluid can freely flow, hitting the obstacle, which is placed in the first quarter of the domain on its bottom. There are several measurement points installed in the setup. H_1 to H_4 are points were the height was measured and P_1 to P_8 mark the position of pressure sensors installed on the cuboid shaped obstacle. The fluid is assumed to be water. Figure 5.1 shows the initial setup for this test. This test is especially well-suited to show the performance of SPH for violent free-surface flows and impacts. The test case was used to investigate the difference between WCSPH, the standard PCISPH as in [7] and the improved PCISPH presented in Section 3. In addition, the method was compared to experimental results from [35]. Figure 5.2 shows the pressure field of the aforementioned three different numerical methods tested, where blue marks low pressure and red high pressure. One can see that the pressure field of the standard PCISPH approach is considerably different to the WCSPH and the modified PCISPH while the other two are quite similar, qualitatively speaking. The standard PCISPH method has considerably more artifacts, especially along the edges of the domain, and is in general not as smooth as one would expect, like in Figure 5.2a.

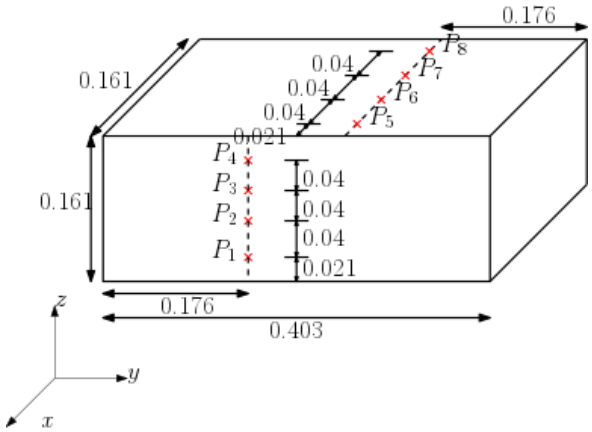
Figure 5.3 compares the height of the fluid at the measurement points $H2$ and $H4$ of the numerical result for $\Delta r = 0.02m$ to the experimental data from [35]. One can see that they provide a good match, although the waves are slightly delayed. Figure 5.4 shows a comparison of the pressure values at the measurement points $P1$ and $P3$, again for a discretization size of $\Delta r = 0.02m$. The data show that PCISPH has problems with shocks, which is understandable from the derivation in Section 3, since the method is based on fixing density errors within a single time step which can lead to high pressure values. To get a better understanding, Figure 5.5 shows a close-up of the shock at the same measurement points compared to the WCSPH method. The



(a) Top View



(b) Side View



(c) Obstacle

Figure 5.1: Initial state for the Dam Break scenario.

Δr $[m]$	N	FPS		Δt $[s]$		T_{sim} $[h]$		Speed-Up
		PCISPH	WCSPH	PCISPH	WCSPH	PCISPH	WCSPH	
0.01833	229885	78.0	120.1	7.47×10^{-4}	7.66×10^{-5}	0.028	0.181	6.46
0.01222	634657	28.5	46.1	4.28×10^{-4}	5.67×10^{-5}	0.137	0.639	4.66
0.00916	1346169	12.4	21.1	2.99×10^{-4}	3.80×10^{-5}	0.450	2.082	4.63
0.00733	2447640	5.5	11.3	2.38×10^{-4}	2.53×10^{-5}	1.281	5.848	4.57

Table 5.1: Performance comparison of PCISPH and WCSPH for the dam-break test case.

graph shows that WCSPH is clearly much better at handling the shock than PCISPH.

A performance comparison can be seen in Table 5.1, where Δr denotes the particle size, N the number of particles in the simulation, FPS the average computed time steps per second, Δt the average time step, and T_{sim} the time necessary to finish the simulation. One can see that PCISPH is significantly faster despite the longer computation time for each time step. The data for the table were provided by my colleague Dr. Chong Peng.

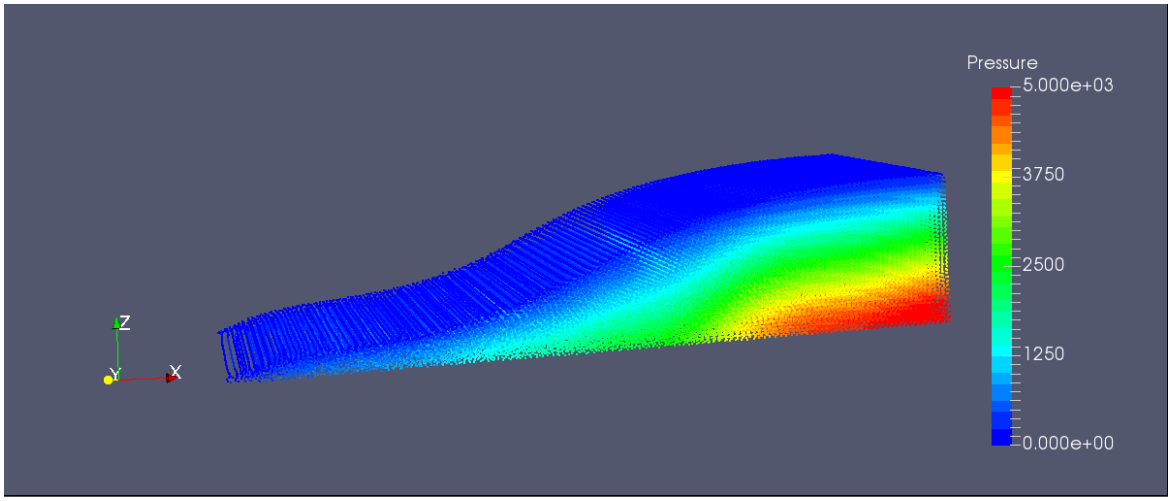
5.2 Lid-Driven Cavity

The second test is a lid-driven cavity. The domain consists of a box sized $1m \times 1m \times 1m$ filled with liquid. The lid of the box is moved with $1\frac{m}{s}$ in the positive x direction. Figure 5.6 depicts the setup. The simulation was performed for Reynold's number $Re = 1000$. The analysis of the results was performed for the steady state solution. A qualitative comparison of the computed velocity field after reaching the steady state can be seen in Figure 5.7. The characteristics of this flow are the big central vortex and the two smaller vortices in the right and left lower corners. One can see that they are clearly visible in both WCSPH and PCISPH.

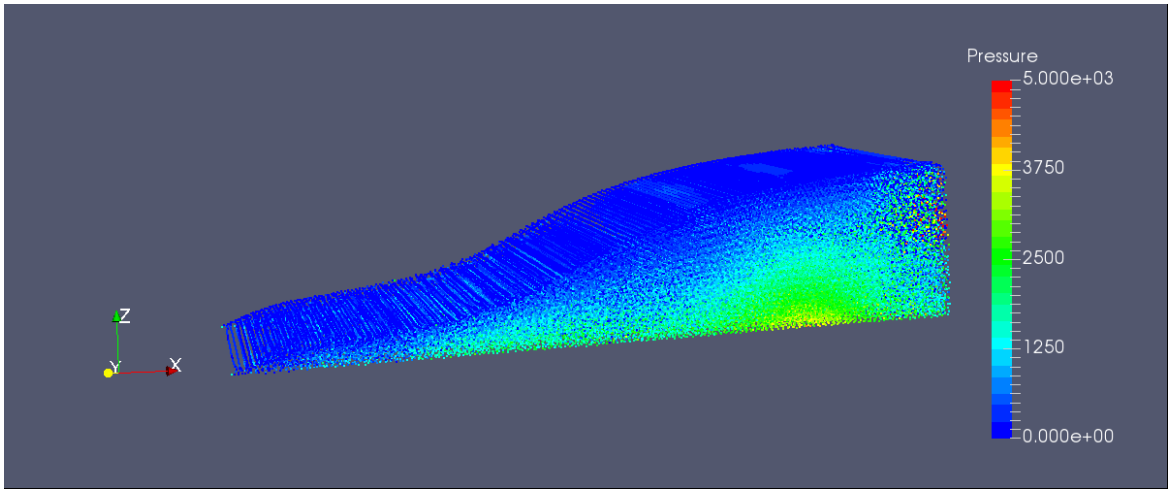
Additionally, the velocities in x direction along the vertical line at $x = y = 0.5$ and the velocity in z direction along the line $y = z = 0.5$ were compared to results from [48] in Figure 5.8. Those results are based on the averaged velocities of 10 seconds in the simulation after reaching the steady-state.

Table 5.2 compares the performance between PCISPH and WCSPH, where Δr denotes the particle size, N the number of particles in the simulation, FPS the average computed time steps per second, Δt the average time step, and T_{sim} the time necessary to finish the simulation. The performance gain is significant. Again, the data for the table are a courtesy of my colleague Dr. Chong Peng.

(a) WCSPH



(b) Standard PCISPH



(c) Modified PCISPH

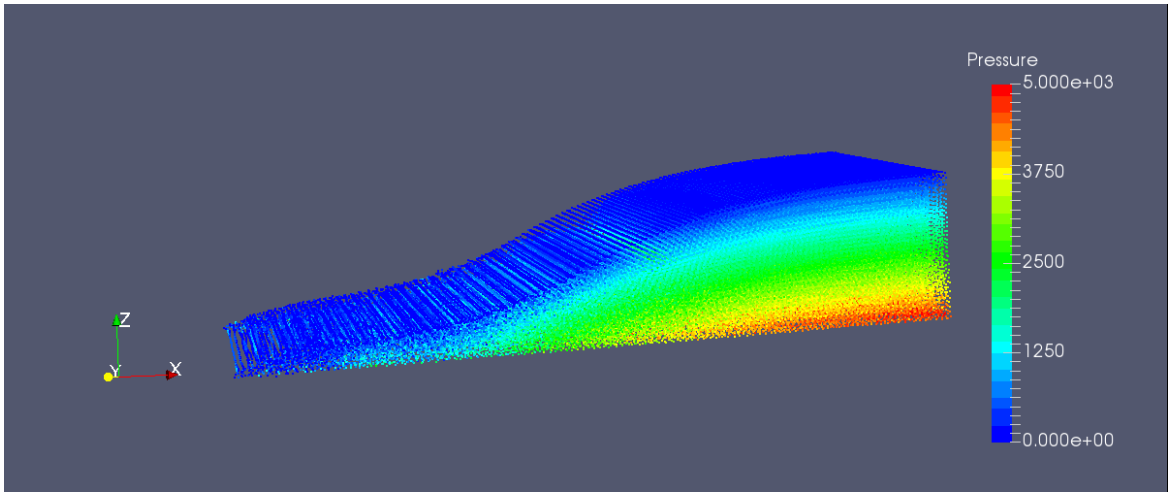


Figure 5.2: Results of the Dam Break at $t = 0.4s$.

Figure 5.3: Dam-Break height comparison for $\Delta r = 0.02m$.

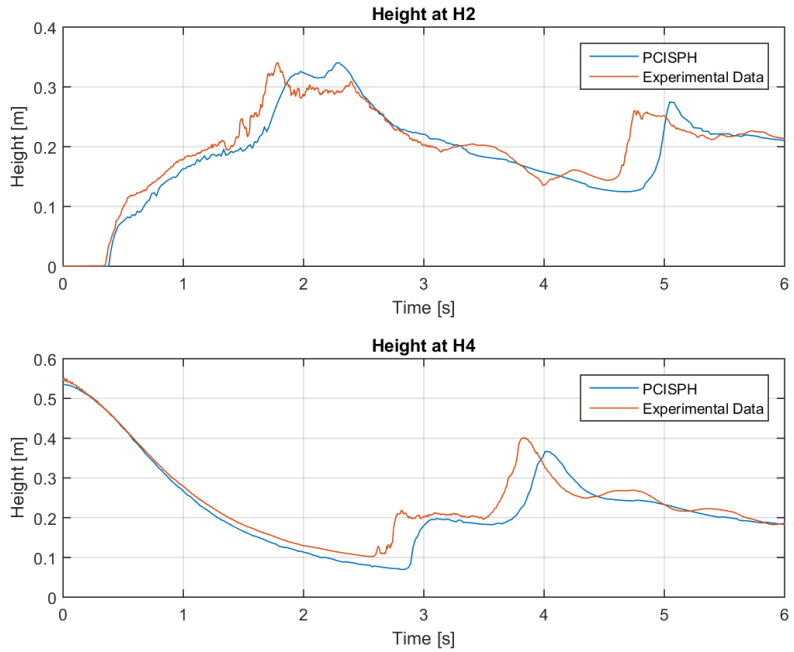


Figure 5.4: Dam-Break pressure comparison for $\Delta r = 0.02m$.

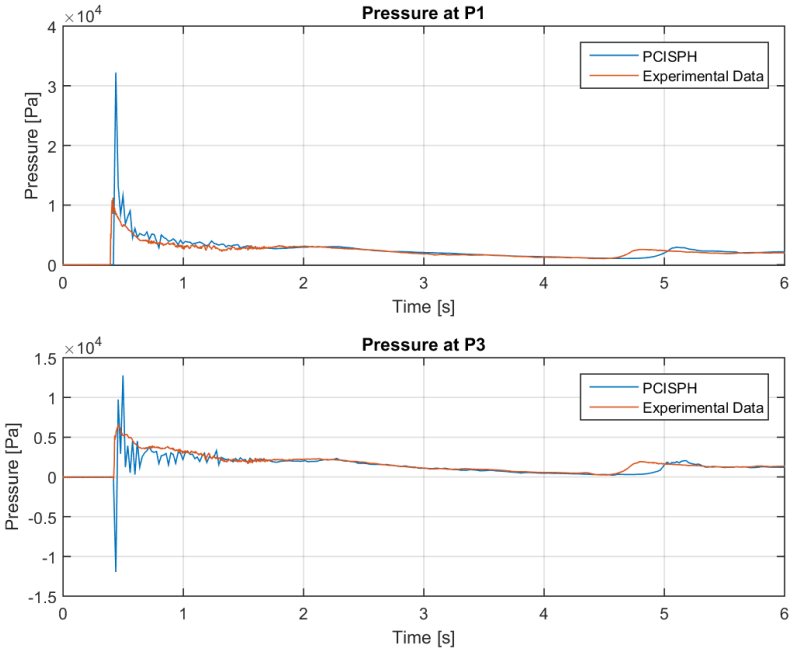
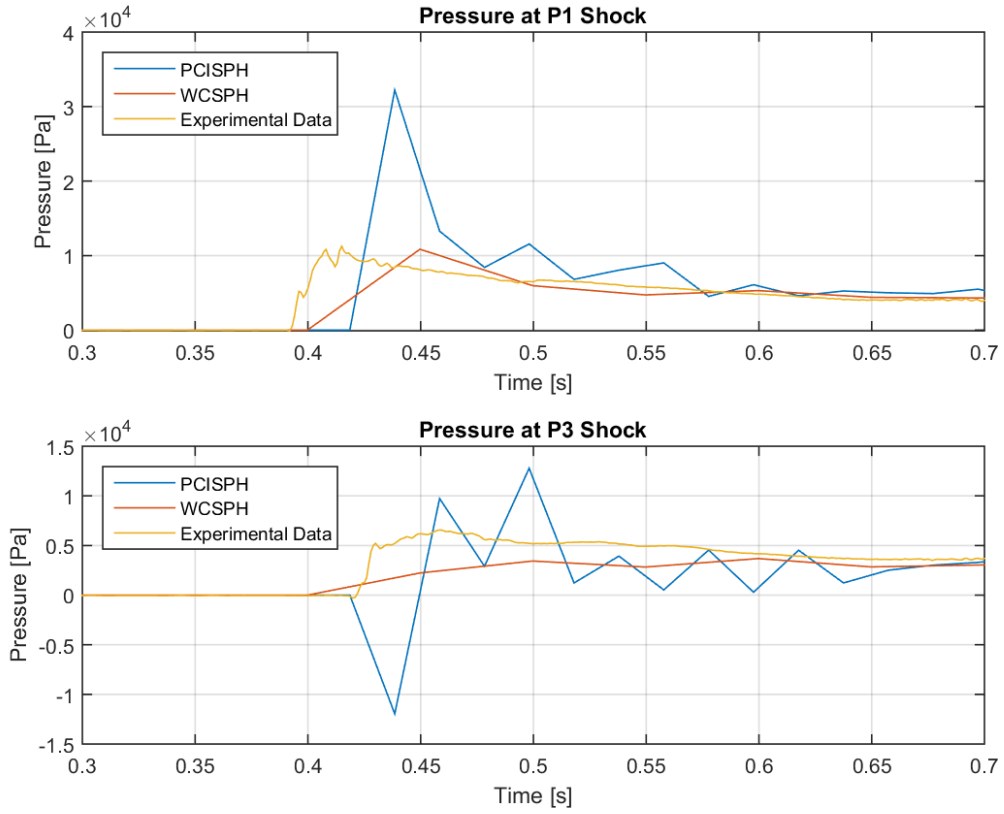


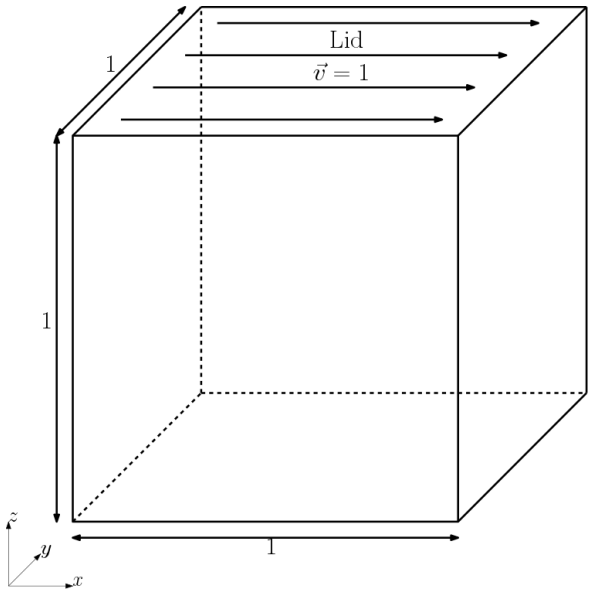
Figure 5.5: Dam-Break pressure comparison, close-up.



Δr $[m]$	N	FPS		Δt $[s]$		T_{sim} $[h]$		Speed-Up
		PCISPH	WCSPH	PCISPH	WCSPH	PCISPH	WCSPH	
0.04	29791	250.2	421.2	4.01×10^{-3}	3.61×10^{-4}	0.028	0.183	6.53
0.02	175626	52.7	103.7	2.38×10^{-3}	1.81×10^{-4}	0.221	1.497	6.77
0.0133	531441	14.4	30.5	1.48×10^{-3}	1.21×10^{-4}	1.303	7.567	5.81
0.01	1191016	6.2	12.9	1.03×10^{-3}	9.14×10^{-5}	4.326	23.515	5.44

Table 5.2: Performance comparison of PCISPH and WCSPH for the lid-driven cavity test case.

Figure 5.6: Lid-driven cavity test, setup.



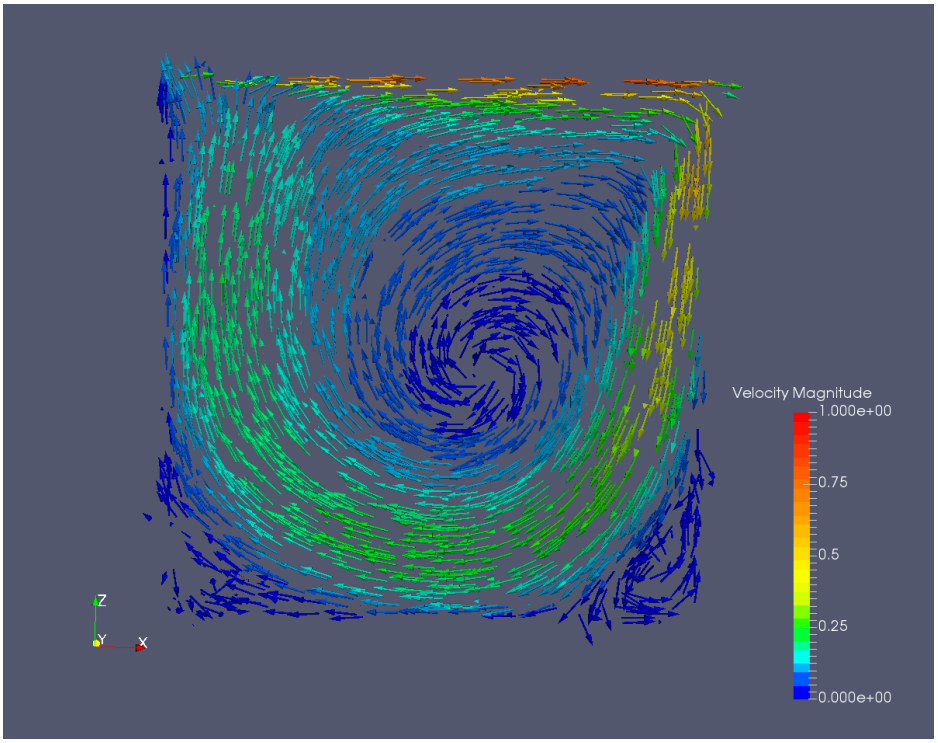
5.3 Hydrostatic Equilibrium

This test is designed to test the stability and accuracy of the newly developed multi-phase PCISPH approach. It consists of a simple unit box where the lower half is filled with a higher density fluid with density ρ_0 and the upper half with a lower density fluid with density ρ_1. Figure 5.9 depicts this setup. The only external force applied is gravity. One expects the fluids to not move or mix. The resulting pressure was compared to the analytical solution which can be computed from

$$p = \rho g h + p_{atm},$$

where p denotes the pressure, ρ the density, g the gravity, h the height of the fluid on top of the point where p is measured, and p_{atm} the atmospheric pressure acting on top of the fluid. The atmospheric pressure was set to 0 for this test. Four tests were performed with density ratios $\rho_1/\rho_0 = 1$, which equals a single-phase simulation, $\rho_1/\rho_0 = 0.8$, $\rho_1/\rho_0 = 0.4$, and $\rho_1/\rho_0 = 0.2$. One can see from the results shown in Figure 5.10 that the numerical results are nearly identical to the analytical solution except for values close to the boundary, where strong deviations can be observed. This is due to the employed Adami Boundary Condition. Important to note is that the simulation was stable for arbitrary long simulation times. This was not the case for the standard PCISPH. Due to this, it was not possible to perform the computation with the standard PCISPH at all.

(a) WCSPH



(b) PCISPH

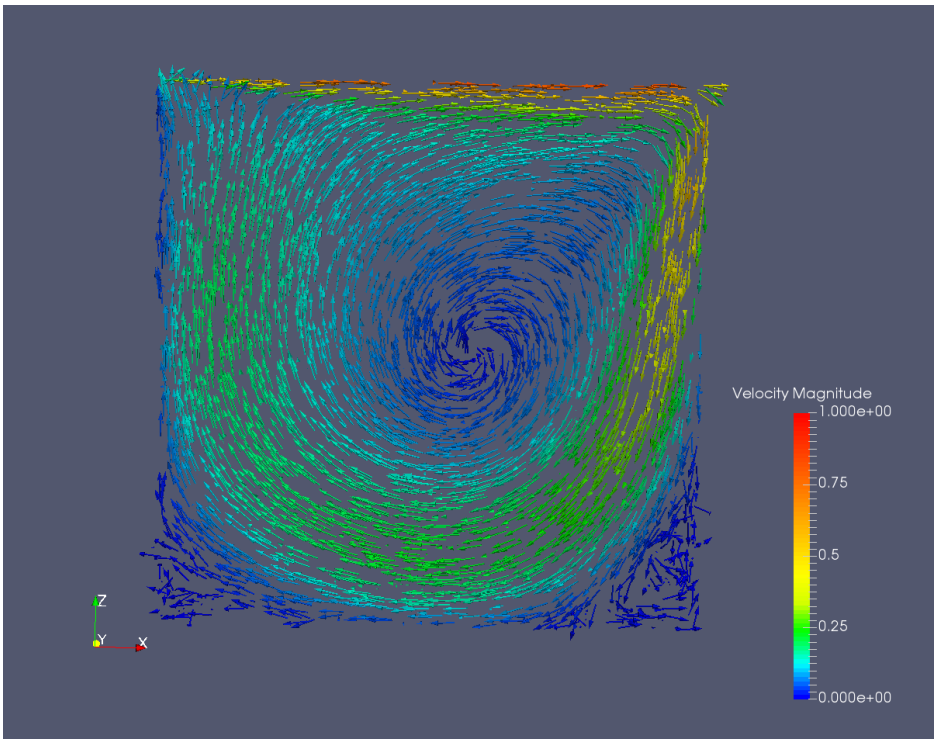
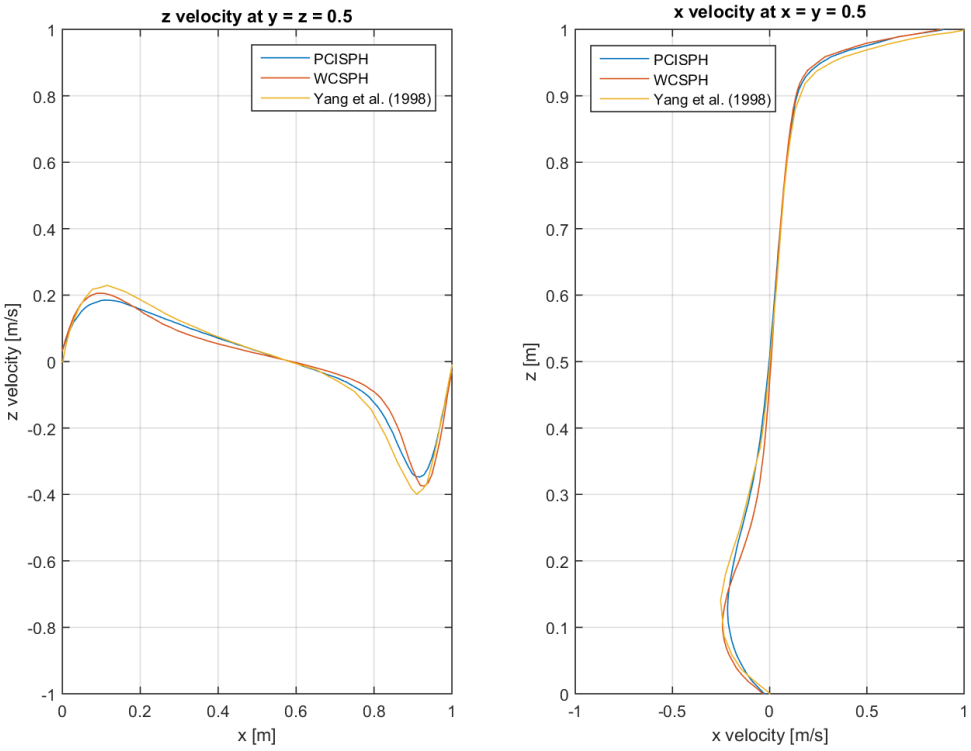


Figure 5.7: Slice of lid-driven cavity steady-state solution in 3D.

Figure 5.8: Computed velocity profiles compared to data from [48].

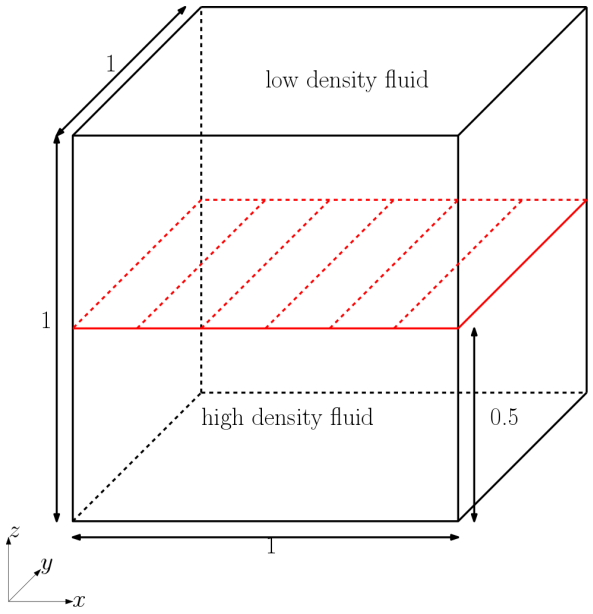


5.4 Rayleigh-Taylor Instability

The so-called Rayleigh-Taylor instability occurs on the interface of two fluids with different densities, when the lighter fluid pushes the heavier fluid [39]. It is therefore a well-suited phenomenon to test the newly developed multiphase flow approach for PCISPH from Section 3.

Two tests were performed. The first one is a 2D scenario. The domain is $1m \times 2m$, where the lower half is filled with a lighter fluid with density ρ_0 and the upper half is filled with a heavier fluid of density ρ_1. The density ratio ρ_0/ρ_1 is set to 0.55. A sine shaped instability, $1 - 0.15\sin(2\pi x)$, is introduced at the interface. Gravity $\vec{g}$ is set to $\vec{g} = (0, -1)$ and the kinematic viscosity ν of both fluids is set to $\nu = 1/420$. This results in a Reynolds number of $Re = 420$. Figure 5.11 shows a sketch of the test case. Figure 5.12 shows the results. The white lines, where applied, indicate the shapes of the same test case computed with WCSPH in [38]. The qualitative comparison shows that the values on the boundaries are different. This is due to the different employed boundary conditions and because the used Adami Boundary Condition did not enforce

Figure 5.9: Hydrostatic test, setup.



a true no-slip condition. On the other hand, one can see that the newly developed PCISPH method allows finer details like the fine curls at the tips.

A second test case for a 3D Rayleigh-Taylor instability was performed. It is intended as a proof of concept and to provide validation data to other developers, since data for such 3D cases are non-existent. Analogously to the 2D case, the domain is cuboid shaped $1m \times 1m \times 2m$, where the lower half is filled with a fluid of density ρ_0 and the upper half with a denser fluid of density ρ_1. The density ratio ρ_1/ρ_0 is again set to $\rho_1/\rho_0 = 0.55$ and the gravity $\vec{g}$ to $\vec{g} = (0, 0, -1)$. The interface is not flat, parallel to the x-y-plane, but sine shaped to introduce an initial instability. The Reynolds number is again set to $Re = 420$. Figure 5.13 is a sketch of this setup. The evolution of the 3D interface is shown in Figure 5.14. The higher density fluid is set transparent to be able to see the interface. Slices of the domain parallel to the x-z-plane at $y = 0.25$ and $y = 0.5$ are shown in Figure 5.15 for future reference.

Figure 5.10: Hydrostatic test, results.

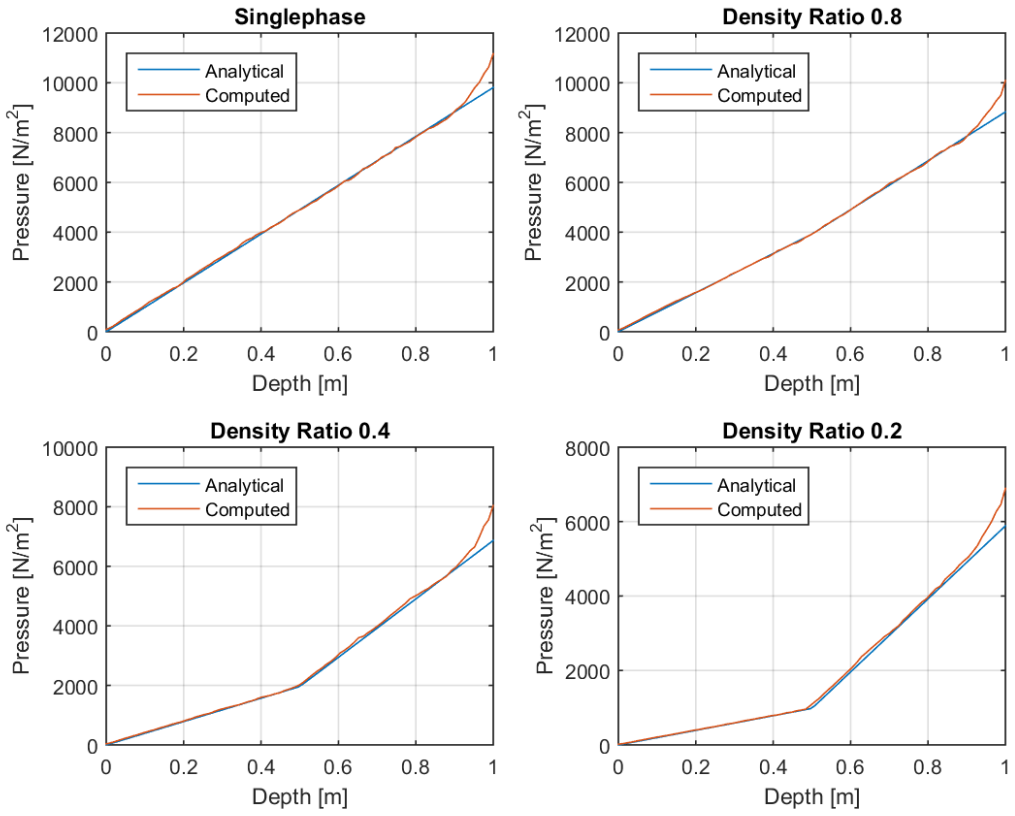
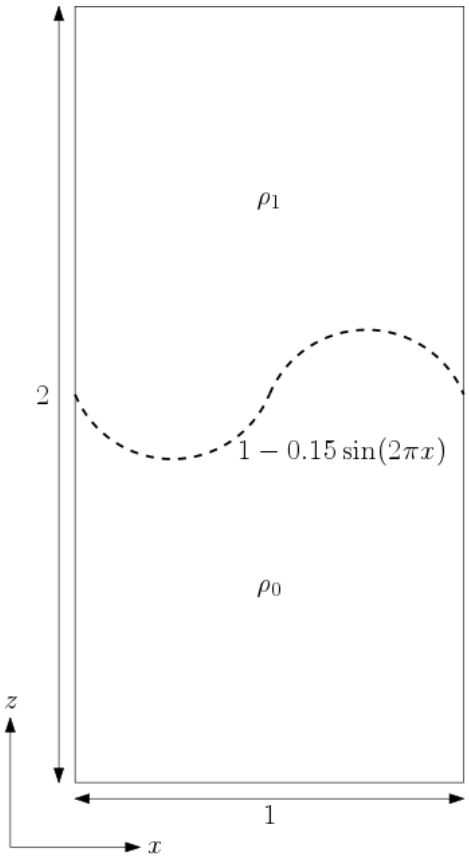


Figure 5.11: Rayleigh-Taylor 2D test, setup.



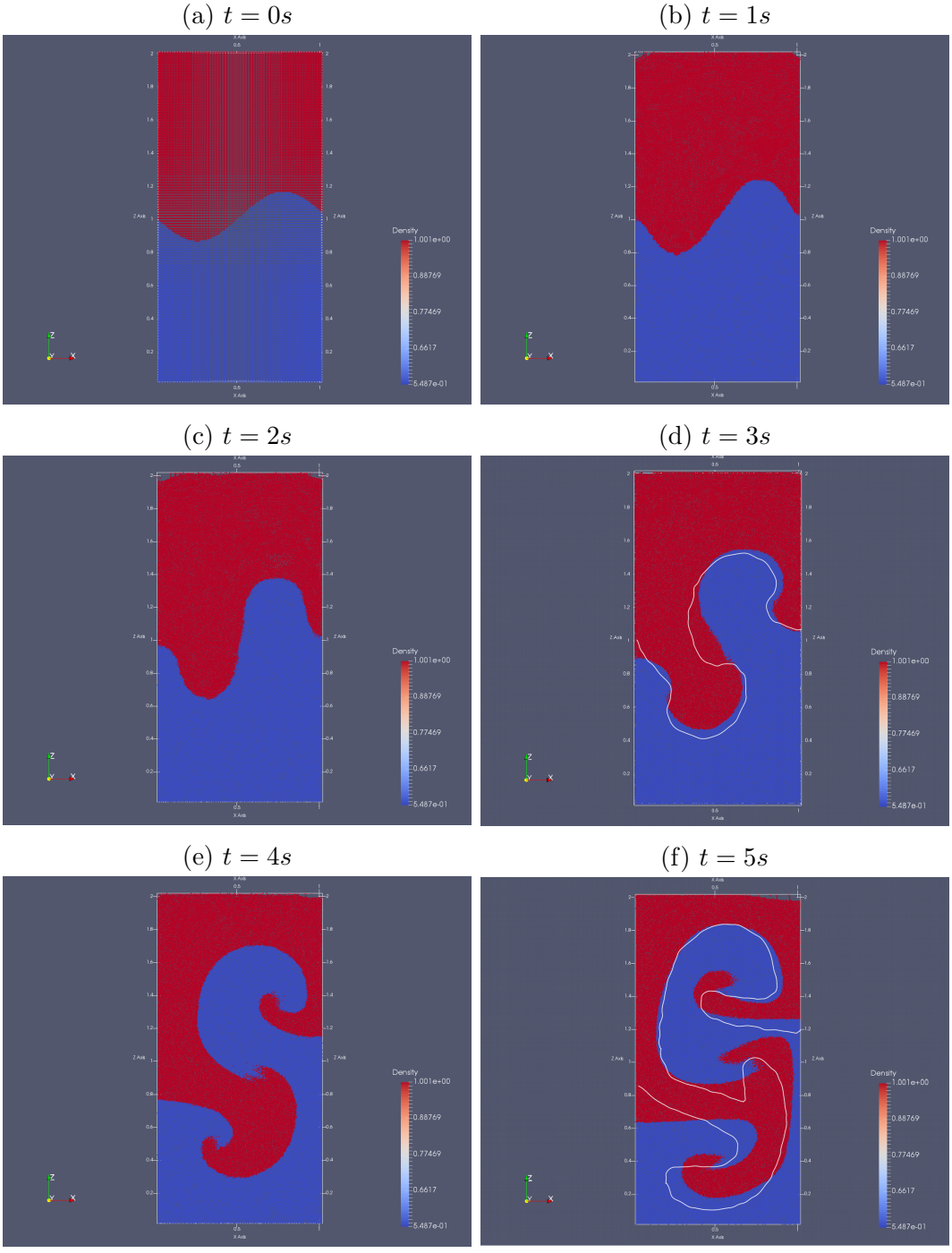
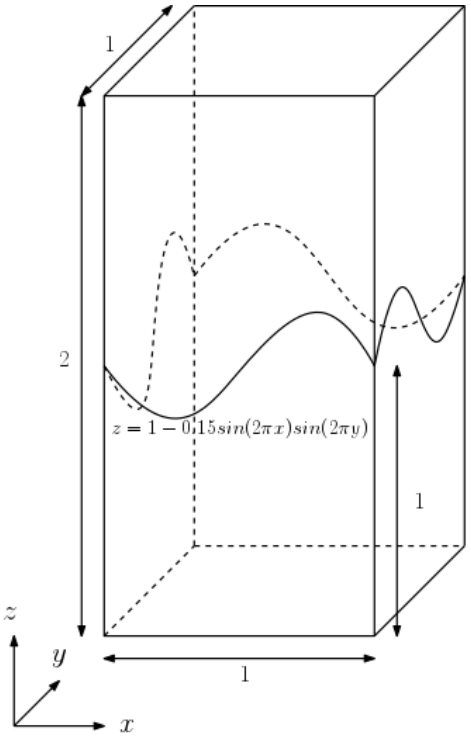


Figure 5.12: Result of the 2D Rayleigh-Taylor instability computed with the modified PCISPH.

Figure 5.13: Rayleigh-Taylor test, setup.



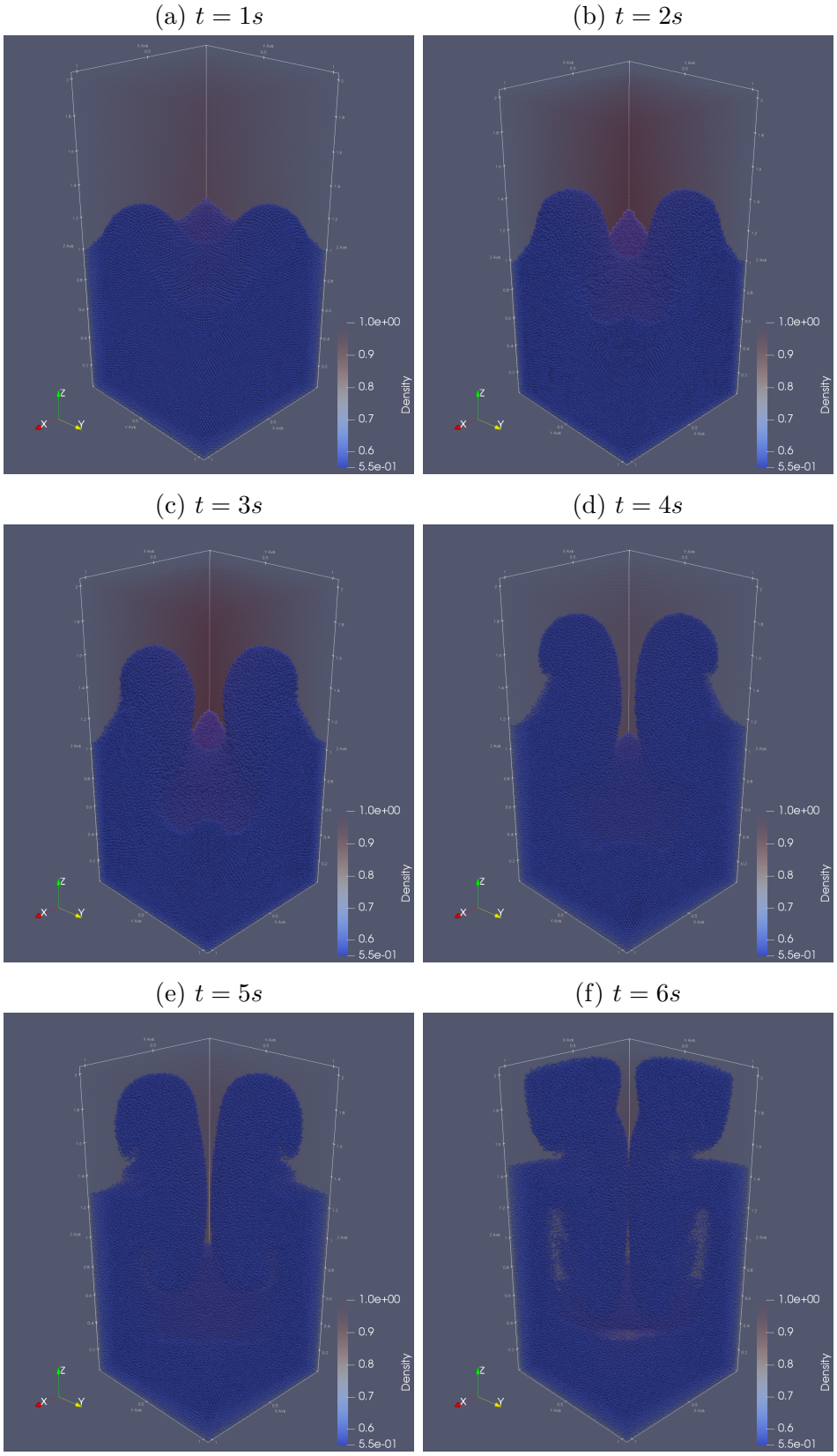
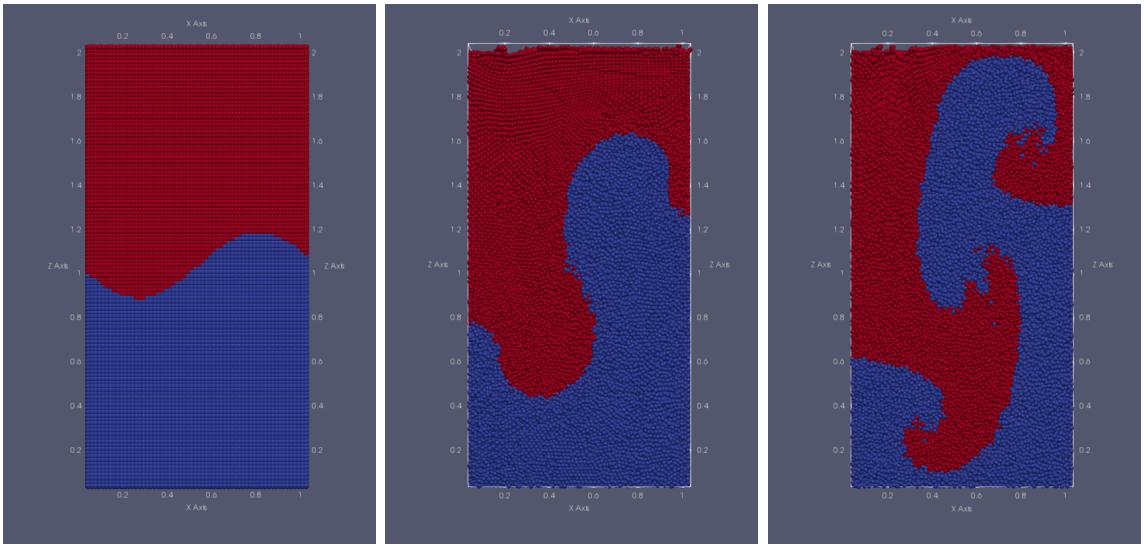


Figure 5.14: Result of the Rayleigh-Taylor instability computed with the modified PCISPH.

(a) $t = 0s$, $y = 0.25$ (b) $t = 3s$, $y = 0.25$ (c) $t = 5s$, $y = 0.25$



(d) $t = 0s$, $y = 0.5$ (e) $t = 3s$, $y = 0.5$ (f) $t = 5s$, $y = 0.5$

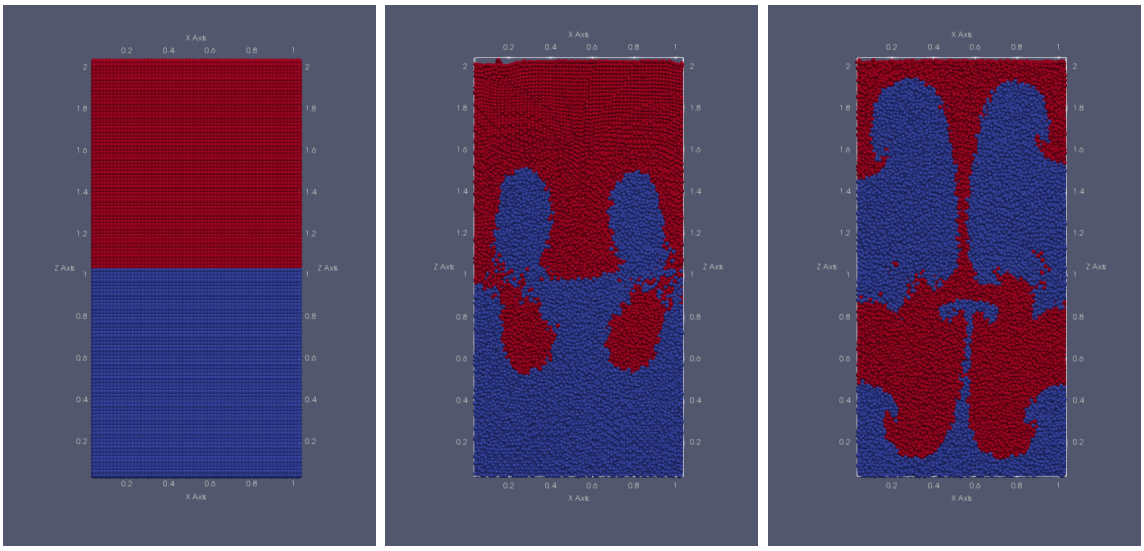


Figure 5.15: Result of the 3D Rayleigh-Taylor instability at $y = 0.25$ and $y = 0.5$ computed with the modified PCISPH.

6 Conclusion and Future Work

SPH is a promising CFD method which will definitely gain even more popularity in the next years despite the lacking mathematical background. Currently, the main disadvantage of SPH is the longer computation times compared to established methods. In this work, another step towards reducing the computation time was presented, using a modified PCISPH method. It was shown that high quality and physical correct results can be obtained while significantly increasing the performance.

But there are still some topics that require further attention. This includes a more detailed validation of PCISPH, testing different boundary methods (for example as presented in Section 2.5), improving PCISPH for high density ratio multiphase flows, improving the shock handling of PCISPH, testing different time stepping schemes for the outer and inner iterations, and a convergence analysis of the inner iteration of PCISPH. In addition, a direct comparison of other SPH variants, like IISPH [15] and DFSPH [17] with the presented PCISPH and WCSPH would be interesting. Especially, using different boundary methods. In general, the derivation of a proper error estimate for SPH is an interesting but presumably very difficult task, since nobody was able to do that in the last 40 years except for special cases [43]. Another interesting topic would be to investigate the similarities of SPH and mesh based methods like FVM or FEM. This might lead to a better understanding of the underlying math.

To conclude, SPH was repeatedly shown to be a serious alternative to mesh-based methods but there is still a lot of work until it will gain broad acceptance for industrial applications.

List of Figures

1.1 Plot of different kernel functions for $h = 1$. 17

2.1 Wall sampled with dummy particles. 29
2.2 2D sketch of neighbor search grid. 33
2.3 Comparison between high-end CPU and GPU. 33

5.1 Initial state for the Dam Break scenario. 48
5.2 Results of the Dam Break at $t = 0.4s$. 50
5.3 Dam-Break height comparison for $\Delta r = 0.02m$. 51
5.4 Dam-Break pressure comparison for $\Delta r = 0.02m$. 51
5.5 Dam-Break pressure comparison, close-up. 52
5.6 Lid-driven cavity test, setup. 53
5.7 Slice of lid-driven cavity steady-state solution in 3D. 54
5.8 Computed velocity profiles compared to data from [48]. 55
5.9 Hydrostatic test, setup. 56
5.10 Hydrostatic test, results. 57
5.11 Rayleigh-Taylor 2D test, setup. 58
5.12 Result of the 2D Rayleigh-Taylor instability computed with the modified PCISPH. 59
5.13 Rayleigh-Taylor test, setup. 60
5.14 Result of the Rayleigh-Taylor instability computed with the modified PCISPH. 61
5.15 Result of the 3D Rayleigh-Taylor instability at $y = 0.25$ and $y = 0.5$ computed with the modified PCISPH. 62

Bibliography

[1] Cuda. https://developer.nvidia.com/cuda-zone. Accessed: 2018-03-27.

[2] Geizhals. www.geizhals.at. Accessed: 2018-03-27.

[3] Intel xeon e5-2698 v4. https://ark.intel.com/de/products/91753/Intel-Xeon-Processor-E5-2698-v4-50M-Cache-2_20-GHz. Accessed: 2018-03-27.

[4] Nvidia geforce gtx 1080. https://www.nvidia.at/graphics-cards/geforce/pascal/gtx-1080/. Accessed: 2018-03-27.

[5] H. W. Alt. *Lineare Funktionalanalysis*. Springer, Berlin Heidelberg, 1985.

[6] Maurizio Landrini Andrea Colagrossi. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 192, 2003.

[7] R. Parajola B. Solenthaler. Predictive-corrective incompressible sph. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2009*, 28, 2009.

[8] R. Pramanik D. Deb. Failure process of brittle rock using smoothed particle hydrodynamics. *Jounral of Engineering Mechanics*, 139:1551–1565, 2013.

[9] A. Colagrossi D. Molteni. A simple procedure to improve the pressure evaluation in hydrodynamic context using sph. *Computer Physics Communications*, 180:861–872, 2009.

[10] R. Issa D. Violeau. Numerical modelling of complex turbulent free-surface flows with the sph method: an overview. *International Journal for numerical methods in fluids*, 53:277–304, 2006.

[11] M. Gross F. Thaler, B. Solenthaler. A parallel architecture for iisph fluids. *Workshop on Virtual Reality interaction and Physical Simulation*, 2014.

[12] G. L. Cassiday G. R. Fowles. *Analytical Mechanics*. Thomson Brooks/Cole, Belmont, USA, seventh edition, 2005.

[13] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[14] T. Reviol H. Oertel jr., M Böhle. *Strömungsmechanik für Ingenieure und Naturwissenschaftler*. Springer Vieweg, Wiesbaden, 1999.

[15] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics*, 20:426–435, 2014.

[16] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH Fluids in Computer Graphics. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[17] D. Kaschier J. Bender. Divergence-free smoothed particle hydrodynamics. *SCA '15 Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 147–155, 2015.

[18] J. B. Kajtar J. J. Monaghan. Sph particle boundary forces for arbitrary boundaries. *Computer Physics Communications*, 180:1811–1820, 2009.

[19] T. C. Carney J. K. Chen, J. E. Beraun. A corrective smoothed particled method for boundary value problems in heat conduction. *International Journal for Numerical Methods in Engineering*, 46:231–252, 1999.

[20] D. Valdez-Balderas B. D. Rogers M. Gomez-Gesteira J. M. Dominguez, A. J. C. Crespo. New multi-gpu implementation for smoothed particle hydrodynamics on heterogeneous clusters. *Computer Physics Communications*, 184:1848–1860, 2013.

[21] J. J. Monaghan J. P. Morris. A switch to reduce sph viscosity. *Journal of Computational Physics*, 136:41–50, 1997.

[22] Y. Zhu J. P. Morris, P. J. Fox. Modeling low reynolds number incompressible flows using sph. *Journal of Computational Physics*, 136:214–226, 1997.

[23] K. Königsberger. *Analysis 2*. Springer, Berlin Heidelberg New York, fifth edition, 2004.

[24] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, 1977.

[25] G. R. Liu M. B. Liu. Smoothed particle hydrodynamics (sph): an overview and recent developments. *Archives of Computational Methods in Engineering*, 17:25–76, 2010.

[26] M. Teschner B. Heidelberger M. Gross M. Müller, S. Schirm. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds*, 15:159–171, 2004.

[27] G. Sorbino S. Cuomo V. Drempetic M. Pastor, B. Haddad. A depth-integrated, coupled sph model for flow-like landslides and related phenomena. *International Journal for Numerical and Analytical Methods in Geomechanics*, 33:143–172, 209.

[28] J. J. Monaghan. Extrapolating b splines for interpolation. *Journal of Computational Physics*, 60:253–262, 1985.

[29] J. J. Monaghan. Smoothed particle hydrodynamics. *Journal of Computational Physics*, 30:543–574, 1992.

[30] J. J. Monaghan. Simulating free surface flows with sph. *Journal of Computational Physics*, 110:399–406, 1994.

[31] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703–1759, 2005.

[32] J. J. Monaghan. Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics*, 44:323–346, 2012.

[33] G. Akinci B. Solenthaler M. Teschner N. Akinci, M. Ihmsen. Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2012*, 31, 2012.

[34] L. D. Libersky P. W. Randles. Smoothed particle hydrodynamics: Some recent improvements and applications. *Computational Methods in Applied Mechanics and Engineering*, 139:375–408, 1996.

[35] D. Violeau R. Issa. 3d schematic dam break and evolution of the free surface. `http://spheric-sph.org/tests/test-2`, 2006. Accessed: 2017-07-09.

[36] N. A. Adams S. Adami, X. Y. Hu. A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics*, 231:7057–7075, 2013.

[37] P. Koumoutsakos S. E. Hieber. An immersed boundary method for smoothed particle hydrodynamics of self-propelled swimmers. *Journal of Computational Physics*, 227:8636–8654, 2008.

[38] M. Rudman S. J. Cummins. An sph projection method. *Journal of Computational Physics*, 152:584–607, 1999.

[39] D. H. Sharp. An overview of rayleigh-taylor instability. *Physica D: Nonlinear Phenomena*, 12:3–18, 1984.

[40] K. Szewc. Development of Smoothed Particle Hydrodynamics Approach for Modelling of Multiphase Flows with Interfaces. Dissertation, The Institute of Fluid Flow Machinery, Polish Academy of Sciences, Gdansk, Poland, 2013.

[41] R. L. Rivest C. Stein T. H. Cormen, C. E. Leiserson. *Introduction to Algorithms*. The MIT Press, Cambridge, London, third edition, 2009.

[42] Y. Kawaguchi T. Harada, S. Koshizuka. Smoothed particle hydrodynamics on gpus. *Computer Graphics International*, 40:63–70, 2007.

[43] J. P. Vila. On particle weighted methods and sph. *Mathematical Models and Methods in Applied Sciences*, 9:161–209, 1999.

[44] D. Violeau. *Fluid Mechanics and the SPH Method. Theory and Applications.* Oxford University Press, Oxford, 2012.

[45] A. Reusken W. Dahmen. *Numerik für Ingenieure und Naturwissenschaftler*. Springer, Berlin Heidelberg, second edition, 2006.

[46] W. T. Vetterling B. P. Flannery W. H. Press, S. A. Teukolsky. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, third edition, 2007.

[47] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995.

[48] J. Y. Yang, S. C. Yang, Y. N. Chen, and C. A. Hsu. Implicit weighted eno schemes for the three-dimensional incompressible navier-stokes equations. *Journal of Computational Physics*, 146:464–487, 1998.

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitely indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date	Signature