Bachelor's Thesis

Incompressible SPH with Advanced Boundary Handling for Dynamic and Non-Uniform Boundaries

Thierry Meiers

University of Freiburg
Faculty of Engineering
Computer Science

February 27, 2025

Writing Period

28. 11. 2024 – 28. 02. 2025

Examiner & Adviser

Prof. Dr.-Ing. Matthias Teschner

Acknowledgments

First and foremost, I want to express my gratitude to my thesis supervisor, Prof. Dr.-Ing. Matthias Teschner. His support, insightful guidance, and constant patience were integral to the success of this project. The numerous discussions and feedback sessions provided invaluable direction and insight, helping to overcome the many challenges faced along the way. This thesis would not have been possible without his mentorship and dedication.

A heartfelt thank you is extended to my family—my mother, father, and girlfriend—for their steadfast encouragement. During the most difficult moments, whether navigating complex debugging or dealing with technical problems, their support provided the strength and motivation to continue. Their belief in me was an ever-present source of comfort and inspiration.

The tools that enabled this work deserve recognition as well. GitHub proved essential for organising both the code and the thesis itself, ensuring that progress remained steady and manageable. The Monogame framework, in particular, provided a robust foundation to bring the simulation project to life, offering the flexibility needed to realise the vision for this work.

Lastly, appreciation is extended to all those who supported me during these years of study. Whether through encouraging words or simple companionship, their presence turned the most stressful times into opportunities for growth. Their unwavering support played a crucial role in maintaining focus and determination, ultimately making it possible to reach the finish line.

Contents

1 Introduction **1**

2 Fundamentals **3**

2.1 Navier-Stokes Equations 4

2.1.1 Momentum equation 4

2.1.2 Continuity Equation 5

2.2 Smoothed Particle Hydrodynamics 5

2.2.1 Concept 6

2.2.2 Kernel Function 7

2.2.3 Discretization 8

2.2.4 Base Fluid Solver 9

2.3 Pressure computation 11

2.3.1 Local Pressure Solver 11

2.3.2 Global Pressure Solver 12

3 Implicit Incompressible SPH **13**

3.1 Pressure Poisson Equation 13

3.1.1 Derivation 14

3.1.2 Discretization 15

3.2 Pressure Solver 15

3.3 Implementation 17

4 Boundary Handling **20**

4.1 Representation 20

4.2 Pressure at Boundaries 21

4.2.1 Pressure Extrapolation 22

4.2.2 Pressure Mirroring 22

4.2.3 Zero-Pressure Boundary 23

4.3 Diagonal element 23

5 Results **24**

5.1 Demonstration 25
5.1.1 Breaking Dam 25
5.1.2 Dynamic and non-uniform boundaries 27
5.2 IISPH behaviour 28
5.2.1 Solver Iterations 28
5.2.2 Compression Error 29
5.3 Boundary Handling 30
5.4 Performance of IISPH 32
5.5 WCSPH or IISPH 34
5.5.1 Performance 34
5.5.2 Stability and Usability 35

6 Conclusion **36**

7 Appendix **38**

Bibliography **40**

1 Introduction

Smoothed Particle Hydrodynamics (SPH) is an essential tool in many scientific and industrial fields, including computer graphics, engineering, and physical modelling. It enables detailed analysis of fluid flow and plays a key role in simulating real-world physical phenomena. Beyond fluid simulations, SPH is widely applied in various domains. These include elastic solids [DG96; Pee+18], rigid body dynamics [Gis+19], viscous fluids [Tak+15; Wei+18], ferrofluids [HHM19], snow simulation [Gis+20], and even astrophysical modelling [Spr10]. These diverse applications highlight SPH's diversity in capturing complex physical behaviours across different scientific disciplines.

SPH methods for fluid simulations can be divided into two main branches. One approach computes hydrostatic pressure locally by evaluating compression weighted with a user-defined stiffness value, referred to as Weakly Compressible Smoothed Particle Hydrodynamics (WCSPH). Since WCSPH does not fully enforce incompressibility, small density fluctuations occur, making the fluid slightly compressible.

In contrast, some SPH methods enforce incompressibility more strictly by solving a Pressure Poisson Equation (PPE), leading to what is generally referred to as Incompressible Smoothed Particle Hydrodynamics (ISPH). ISPH describes the concept of using a linear system of equations to determine a divergence-free velocity field, which results in more stable and physically accurate fluid behaviour, particularly in high-pressure scenarios such as splashing water. By minimising density fluctuations, ISPH-based methods ensure volume conservation, preventing unnatural compression artefacts.

This work primarily focuses on a specific ISPH implementation known as Implicit Incompressible Smoothed Particle Hydrodynamics (IISPH). IISPH introduces additional computational overhead compared to WCSPH, as it requires iteratively solving a linear system of equations. In addition, IISPH presents significant implementation challenges due to its complexity.

Furthermore, this work examines different boundary handling implementations. Boundary handling plays a crucial role in SPH simulations, as accurate boundary interactions ensure realistic fluid behaviour and stability. The goal of this research is to analyse the behaviour of IISPH, evaluate its stability and performance, and compare it to WCSPH under different conditions.

This work is structured into seven main chapters, each covering a key aspect of the theoretical concepts, implementation, and evaluation of the IISPH solver.

Chapter 2 presents the fundamental concepts necessary to understand the core principles behind SPH-based fluid simulations. The governing equations of fluid dynamics, particularly the Navier-Stokes equations, are introduced, followed by an overview of the SPH method, including kernel functions, discretization techniques, and the formulation of a base fluid solver. Furthermore, different pressure computation approaches are discussed, differentiating between local and global pressure solvers.

Chapter 3 focuses on the IISPH method. It details the derivation and discretization of the Pressure Poisson Equation (PPE) and describes the numerical pressure solver used in IISPH. Additionally, implementation aspects are discussed.

Chapter 4 investigates boundary handling methods and their impact on fluid simulations. Different approaches for computing boundary pressure are examined, including pressure extrapolation, pressure mirroring, and zero-pressure boundaries.

Chapter 5 presents and evaluates the simulation results. The IISPH solver is demonstrated in various scenarios, including a breaking dam simulation and cases involving dynamic, non-uniform boundaries. The solver's behaviour is analysed in terms of convergence, compression error, and overall stability. Furthermore, the effects of different boundary handling methods are compared. The trade-offs between IISPH and WCSPH are examined, focusing on performance, stability, and usability.

Finally, Chapter 6 summarizes the key findings of this work, highlighting the advancements in IISPH, its performance compared to WCSPH, and the impact of optimized boundary handling and spatial hashing on fluid simulations.

2 Fundamentals

This chapter provides an introduction to the fundamentals of Smoothed Particle Hydrodynamics (SPH). It begins with an overview of the Navier-Stokes equations, discussing their significance in fluid dynamics and their role in SPH. Next, the key principles and methodology of SPH are presented. Finally, various approaches to pressure computation are discussed, highlighting their respective advantages and disadvantages.

Symbol	Quantity	Unit
d	Spatial dimension	–
A	Auxiliary function	–
t	Time	s
ρ	Density	kg m^{-3}
p	Pressure	Pa
m	Mass	kg
V	Volume	m^3
μ	Dynamic viscosity	Pa·s
ν	Kinematic viscosity	$\mathrm{m}^2\ \mathrm{s}^{-1}$
h	Smoothing length	m
$\hbar$	Radius of kernel support	m
$\tilde{h}$	Particle size	m
α_d	Kernel normalization factor for dimension d	m^{-d}
$\mathbf{x}$	Position vector of material particle	m
$\mathbf{x}_{ij}$	Distance vector $\mathbf{x}_i - \mathbf{x}_j$	m
$\mathbf{v}$	Velocity vector of material particle	$\mathrm{m\ s}^{-1}$
$\mathbf{a}$	Acceleration vector of material particle	$\mathrm{m\ s}^{-2}$
$\mathbf{F}$	Force	N

Table 1: List of symbols, quantities, and units.

2.1 Navier-Stokes Equations

In this section, the Navier-Stokes equations are introduced. These equations form a set of partial differential equations that describe the motion of viscous fluid substances. They serve as the fundamental basis for simulating fluid behaviour in Smoothed Particle Hydrodynamics (SPH).

First, the Momentum equation is derived and explained in detail. Then, the Continuity equation is presented.

2.1.1 Momentum equation

The Momentum equation models the movement of the fluid, taking into account the effects of pressure, viscosity, and external forces on the flow behaviour. It can be easily derived from Newton's second law of motion. Since mass remain constant in SPH, Equation (1) depends solely on the velocity change of the fluid sample.

$$\mathbf{F} = m\mathbf{a} = \rho V \frac{D\mathbf{v}}{Dt} \tag{1}$$

The forces acting within and on the fluid are summarized in Equation (2). The term $-\nabla pV$ represents the pressure forces, while $\mu\nabla^2\mathbf{v}V$ describes the viscosity forces within the fluid. Forces, like gravity, are represented by $\rho\,\mathbf{a}^{\text{ext}}V$, acting on the substance from outside.

$$\mathbf{F} = \underbrace{-\nabla pV}_{\text{Pressure force}} + \underbrace{\mu\nabla^2\mathbf{v}V}_{\text{Viscosity force}} + \underbrace{\rho\,\mathbf{a}^{\text{ext}}V}_{\text{External force}} \tag{2}$$

By combining Equation (1) and Equation (2), we obtain the resulting Navier-Stokes momentum equation Equation (3) [Ihm+14b].

$$\begin{aligned}
\rho V\frac{D\mathbf{v}}{Dt} &= (-\nabla p + \mu\nabla^2\mathbf{v} + \rho\,\mathbf{a}^{\text{ext}})V \\
\rho\frac{D\mathbf{v}}{Dt} &= -\nabla p + \rho\nu\nabla^2\mathbf{v} + \rho\,\mathbf{a}^{\text{ext}} \\
\frac{D\mathbf{v}}{Dt} &= \underbrace{-\frac{1}{\rho}\nabla p}_{\text{Pressure acceleration}} + \underbrace{\nu\nabla^2\mathbf{v}}_{\text{Viscosity acceleration}} + \underbrace{\mathbf{a}^{\text{ext}}}_{\text{External acceleration}}
\end{aligned} \tag{3}$$

For clarity, Equation (3) calculates the velocity change $\frac{D\mathbf{v}_i}{Dt}$ by accumulating the pressure acceleration $-\frac{1}{\rho_i}\nabla p_i$, the viscosity acceleration $\nu\nabla^2\mathbf{v}_i$ and the external accelerations $\mathbf{a}_i^{\text{ext}}$ acting on the fluid. This velocity change results in the displacement of the fluid, simulating the movement of a fluid.

2.1.2 Continuity Equation

The continuity equation models the relationship between pressure changes and the divergence of velocity within a fluid. It is straightforward that the flow of the fluid, expressed as velocity, impacts the material's density. A velocity field converging towards a point increases the pressure, represented by a negative divergence of the velocity field. Conversely, a positive divergence decreases the pressure.

The density change $\frac{D\rho}{Dt}$ and the divergence of the velocity field $-\rho\nabla\cdot\mathbf{v}$ cancel each other out, leading to the equation:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{v} = 0, \tag{4}$$

which simplifies to the general form of the continuity equation:

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{v}.$$

Based on the continuity equation, constraints are defined for incompressible fluids, where the density does not change over time $\frac{D\rho}{Dt} = 0$, or the velocity field is divergence-free $\nabla\cdot\mathbf{v} = 0$. Later in this work, one of these constraints will be used to ensure incompressibility.

2.2 Smoothed Particle Hydrodynamics

This section provides an overview of Smoothed Particle Hydrodynamics (SPH). First, the concept of SPH is presented and explained in detail. Next, the Kernel Function is introduced and formulated. Then, the underlying equations utilised in SPH are discretized. Finally, a simple SPH algorithm is constructed and presented for simulating fluid behaviour.

Smoothed Particle Hydrodynamics (SPH) is a computational method for simulating the mechanics of fluids by approximating the Navier-Stokes Equations. It is a mesh-free Lagrangian method, meaning that the coordinates move with the fluid, allowing SPH to capture the behaviour of the fluid from the perspective of the particles, rather than from a fixed grid. The method discretizes spatial field quantities within the fluid by representing the fluid as a collection of particles, each carrying properties such as mass, velocity, and density [Ihm+14b; HY24].

2.2.1 Concept

The concept of SPH is based on breaking a fluid into a set of particles, where each particle represents a small portion of the fluid. This is achieved by the discretization of spatial field quantities and differential operators. To understand SPH discretization, the Dirac-δ identity must be introduced. It is used to simplify the representation of continuous fields by focusing on discrete points as explained in [Kos+20].

$$A_i = (A \cdot \delta)(\mathbf{x}_i) = \int A_j \delta(\mathbf{x}_{ij}) dv_j \tag{5}$$

By using Equation (5), partial properties such as density or pressure, which are typically spread over a region, are then represented as point values. Since the Dirac-δ function is not a standard function, it is replaced by a smoothing function $W : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}$ resulting in:

$$A_i \approx (A \cdot W)(\mathbf{x}_i) = \int A_j W(\mathbf{x}_{ij}, h) dv_j. \tag{6}$$

The smoothing function, referred to as the kernel function, is introduced in the next section. To realise SPH discretization, the analytic integral of Equation (6) is then replaced by a sum over sample points j, where neighbour particles j have a mass m_j, a density ρ_j, and a value A_j [Ihm+14b].

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{x}_{ij}, h). \tag{7}$$

In addition to the discretization of spatial quantities, spatial differential operators also need to be discretized to model physical processes such as gradients, divergence, and diffusion. For instance, the gradient of a scalar field, the divergence of a vector field, and

the Laplacian of a scalar field are formulated as follows [Mon92; Ihm+14b]:

$$\nabla A_i = \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right) \nabla W(\mathbf{x}_{ij}, h) \tag{8}$$

$$\nabla \cdot \mathbf{A}_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{A}_{ij} \cdot \nabla W(\mathbf{x}_{ij}, h) \tag{9}$$

$$\nabla^2 A_i = 2 \sum_j \frac{m_j}{\rho_j} \frac{A_{ij}\mathbf{x}_{ij}}{(\mathbf{x}_{ij})^2 + 0.01h^2} \cdot \nabla W(\mathbf{x}_{ij}, h) \tag{10}$$

2.2.2 Kernel Function

The kernel function is essential for discretizing spatial properties in SPH. It works by smoothing the quantities of surrounding particles within a specified radius h, referred to as the kernel's smoothing length. The smoothing length determines how strongly the unknown value A_i at a position $\mathbf{x}$ is influenced by the known values of neighbouring particles within its supported range $\hbar$.

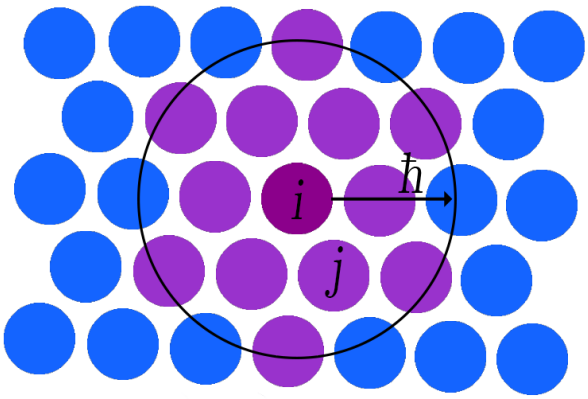


Figure 1: Illustration of the kernel support radius in SPH. The central particle i (dark purple) interacts with its neighbouring particles j (lighter purple) within a support radius $\hbar$, shown by the black circle. Particles outside this range (blue) do not contribute to the kernel function.

As described in [Kos+20], the following properties are desired for the kernel function:

$$\begin{aligned}
\int_{\mathbb{R}^d} W(\mathbf{x}_{ij}, h) dv_j &= 1 \\
\lim_{h' \to 0} W(\mathbf{x_{ij}}, h') &= \delta(r) \\
W(\mathbf{x}_{ij}, h) &\geq 0 \\
W(\mathbf{x}_{ij}, h) &= W(\mathbf{x}_{ji}, h) \\
W(\mathbf{x}_{ij}, h) &= 0 \text{ for } \|\mathbf{x}_{ij}\| \geq \hbar
\end{aligned}$$

This work employs the cubic spline kernel, a commonly used function that satisfies these properties, as defined in [Mei24; Tesna]:

$$W(\mathbf{x}_{ij}, h) = \alpha_d \begin{cases} (2-q)^3 + 4(1-q)^3 & 0 \leq q < 1 \\ (2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

where $q = \frac{\|\mathbf{x}_{ij}\|}{h}$, and the kernel normalization factor for the two-dimensional case ($d = 2$) is:

$$\alpha_2 = \frac{5}{14\pi h^2}.$$

The corresponding derivative of the cubic spline kernel is:

$$\nabla W(\mathbf{x}_{ij}, h) = \alpha_d \frac{\mathbf{x_{ij}}}{\|\mathbf{x_{ij}}\| h} \begin{cases} -3(2-q)^2 + 12(1-q)^2 & 0 \leq q < 1 \\ -3(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2 \end{cases}$$

2.2.3 Discretization

Section 2.2.1 introduces a method to compute unknown properties by using the known values of neighbouring particles. To recall, SPH approximates the Navier-Stokes equations by discretizing a fluid into particle samples that carry properties such as mass, velocity, and density.

The next step towards an SPH formulation is to discretize the terms of the Navier-Stokes

Momentum Equation (3). The pressure acceleration is calculated using Equation 8:

$$\mathbf{a}_i^{\mathrm{p}} = -\frac{1}{\rho_i}\nabla p_i \overset{8}{=} -\sum_j m_j\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W(\mathbf{x}_{ij}, h) \tag{11}$$

and the viscosity acceleration is calculated using Equation 10:

$$\mathbf{a}_i^{\mathrm{v}} = \nu\nabla^2\mathbf{v}_i \overset{10}{=} 2\nu\sum_j \frac{m_j}{\rho_j}\mathbf{v}_{ij}\frac{\mathbf{x}_{ij}\nabla W(\mathbf{x}_{ij}, h)}{(\mathbf{x}_{ij})^2 + 0.01h^2}. \tag{12}$$

Calculating Equation (11) and Equation (12) requires the computation of the pressure p_i, which is discussed in Section 2.3, and density ρ_i, which is calculated using Equation 7:

$$\rho_i \overset{7}{=} \sum_j m_j W(\mathbf{x}_{ij}.h). \tag{13}$$

2.2.4 Base Fluid Solver

At this point, enough information has been presented to construct and demonstrate a base SPH simulation loop. The underlying idea is to iteratively update the position of each particle by calculating the accelerations that influence its velocity. These are determined by summing the viscosity acceleration, pressure acceleration, and external acceleration for a given particle.

Algorithm 1 outlines the base of a simple SPH simulation step. The process consists of five main loops, iteratively updating values for all particles in the system:

1. In the first loop, neighbouring particles are identified within a radius $\hbar$, and the density ρ is computed using Equation (13). In this work, Spatial Hashing, introduced in Appendix 7, is implemented to optimize neighbour search. A kernel support radius $\hbar$ equal to twice the particle size $\tilde{h}$ ($\hbar = 2\tilde{h}$) is chosen as it provides a good trade-off between simulation accuracy and performance, as demonstrated in [BGT20].
2. Second, intermediate velocities $\mathbf{v}_i^*$ are calculated based on external accelerations $\mathbf{a}_i^{\mathrm{ext}}$ and viscosity accelerations $\mathbf{a}_i^{\mathrm{v}}$ using Equation (12). This step ensures that the effects of these accelerations on pressure are accounted for in the next loop.
3. In the third loop, the pressure p_i is determined for each particle. This step is critical because pressure directly affects the compressibility of the fluid during the

simulation. The impact of the pressure computation on the fluid is discussed in detail in Section 2.3.

4. Then follows the computation of Pressure-driven accelerations $\mathbf{a}_i^{\mathrm{p}}$ using Equation (11).

5. Finally, the velocities and positions of the particles are updated. Intermediate velocities and pressure-driven accelerations are combined to compute the final velocities, which are then used to update the particle positions.

forall *particle* i **do**
  find neighbours j ▷ see Appendix 7
  compute Density ρ_i ▷ with Equation (13)
forall *particle* i **do**
  compute $\mathbf{a}_i^{\mathrm{v}} = \nu\nabla^2\mathbf{v}_i$ ▷ with Equation (12)
  $\mathbf{v}_i^* \leftarrow \mathbf{v}_i(t) + \Delta t(\mathbf{a}_i^{\mathrm{v}} + \mathbf{a}_i^{\mathrm{ext}})$
compute Pressure p_i for every particle ▷ see section 2.3
forall *particle* i **do**
  compute $\mathbf{a}_i^{\mathrm{p}} = -\frac{1}{\rho_i}\nabla p_i$ ▷ with Equation (11)
forall *particle* i **do**
  $\mathbf{v}_i(t+\Delta t) \leftarrow \mathbf{v}_i^* + \Delta t\mathbf{a}_i^{\mathrm{p}}$
  $\mathbf{x}_i(t+\Delta t) \leftarrow \mathbf{x}_i(t) + \Delta t\mathbf{v}_i(t+\Delta t)$

Algorithm 1: Base of a simple SPH simulation step

Note that steps 2 and 5 use a variable Δt, referred to as the time step, to update the velocities and positions of the particles. The time step is critical for controlling the stability of the simulation by influencing the travel distance of particles per simulation step. The largest permissible time step is constrained by the following condition:

$$\Delta t \leq \lambda \frac{\tilde{h}}{\mathbf{v}^{\max}} \tag{14}$$

where $\tilde{h}$, $\mathbf{v}^{\max}$, and λ represent the particle size, the velocity of the fastest particle in the fluid, and a scaling factor, respectively. The right-hand side of the equation, known as the Courant-Friedrichs-Lewy (CFL) condition, indicates the velocity relative to the particle size. For $\lambda = 1$, a CFL of 1 signifies that a particle travels a distance equal to its size per simulation step. Typically, a scaling factor of $\lambda = 0.4$ is chosen to ensure particles do not traverse beyond their neighbouring particles during a single step.

2.3 Pressure computation

In SPH simulations, pressure is one of the primary forces influencing particle behaviour. It plays a crucial role in maintaining the fluid's volume by generating acceleration in the direction of the negative pressure gradient, which arises from variations in the density field. This acceleration induces changes in velocity, which, in turn, changes particle positions. These positional adjustments preserve the fluid's volume, effectively maintaining incompressibility [PT24].

The goal during the simulation is to keep the fluid in a minimally compressible state, making it essential to monitor compressibility. Pressure directly influences the volume of the fluid resulting in density changes. Therefore, the relative compression error ρ_i^{err} is computed for every fluid particle by:

$$\rho_i^{\text{err}} = max\left(\frac{\rho_i - \rho^0}{\rho^0}, 0\right) \tag{15}$$

where ρ_i is the pressure at particle i and ρ^0 the rest density of the fluid. The term $\frac{\rho_i - \rho^0}{\rho^0}$ is clamped to only consider the compressed error. The average of these is used to quantify the compressibility of the fluid.

The computation of pressure in SPH is typically divided into two approaches: local pressure solvers and global pressure solvers. Each method has its own advantages and disadvantages in terms of compressibility, maintainability, stability, and performance. This section introduces both methods and highlights their respective strengths and limitations.

2.3.1 Local Pressure Solver

Local pressure solvers are so called Compressible or Weak Compressible Smoothed Particle Hydrodynamics (CSPH or WCSPH) [BT07]. These methods commonly use an equation of state to calculate the pressure of a fluid particle. This computation depends on the neighbouring particles, making the pressure calculation inherently local.

An example for a state equation, to compute the discrete pressure of a particle i, is a variation of the ideal gas equation:

$$p_i = k \left(\frac{\rho_i}{\rho^0} - 1 \right),$$

as presented in [Ada+07; Kos+20]. Here, k is the stiffness constant, ρ_i denotes the density at particle i, and ρ^0 represents the rest density of the fluid. Since ρ_i is computed based on the contributions of neighbouring particles by Equation (13), this approach emphasizes the localized nature of the pressure computation.

Overall, WCSPH is powerful due to its simplicity. However, the stiffness and time step must be carefully adjusted for each scenario, which makes WCSPH less maintainable and less reliable for complex simulations. Specifically, tuning both values to minimize compressibility can be a difficult and time-consuming process.

2.3.2 Global Pressure Solver

Global pressure solvers fundamentally differ from local methods by solving a linear system of equations that enforces incompressibility across the entire fluid domain. This approach ensures that density conservation and non-divergent velocity fields are addressed globally rather than relying on localized corrections.

While solving a linear system at every simulation step is computationally intensive and inherently slower than local pressure methods, global solvers offer several advantages that make them indispensable for incompressible flow simulations. One key advantage is the absence of a stiffness parameter, which is commonly required in local solvers for stability. This eliminates the need for extensive parameter tuning, making global solvers more maintainable and robust.

Despite their higher computational cost, global solvers ensure system-wide incompressibility and prevent the accumulation of numerical errors. Their accuracy and adaptability make them well-suited for highly accurate applications where stability and precision are critical, justifying the trade-off in computational speed.

3 Implicit Incompressible SPH

Implicit Incompressible SPH (IISPH) is part of a group of algorithms known as Incompressible SPH (ISPH). Besides IISPH exists a large set of Incompressible SPH methods such as Divergence-Free SPH (DFSPH) [BK15] or Predictive–Corrective Incompressible SPH (PCISPH) [SP09]. As the name implies, ISPH algorithms utilise linear systems of equations to solve pressure globally to strictly enforce the incompressibility of fluids.

This chapter presents and analyses the core components of the IISPH method and focuses specifically on the IISPH implementation proposed in [Ihm+14a]. The discussion begins with the derivation and exploration of the Pressure Poisson Equation (PPE), which serves as the foundation for achieving incompressibility in fluid simulations.

Next, the chapter explains how the Relaxed Jacobi method is utilised to solve the PPE and compute pressure globally across the fluid. Finally, the IISPH simulation step is presented, accompanied by a detailed explanation of the implementation process.

3.1 Pressure Poisson Equation

The Pressure Poisson Equation (PPE) is a key component in ISPH simulations used to enforce incompressibility. The PPE solves for pressure, which generates pressure acceleration, leading to a velocity change and a position update. The goal is to adjust the position of a particle in a way to preserve the incompressibility of the fluid.

In this section, the derivation of the PPE is explained and the final form is presented, followed by its discretization for use in SPH.

3.1.1 Derivation

First, the density of the fluid is predicted after applying all non-pressure accelerations like viscosity or external accelerations. Non-pressure accelerations alter the velocity and position of fluid particles, leading to density fluctuations that deviate from the fluid's rest density. Thereby the predicted density ρ_i^* is determined using [Ihm+14a; Kos+20]:

$$\rho_i^* = \rho_i - \Delta t \rho_i \nabla \cdot \mathbf{v}_i^* \tag{16}$$

where ρ_i and $\mathbf{v}_i^*$ represent the density and intermediate velocity at particle i.

The goal of the PPE is to eliminate a compression error $\frac{\rho^0 - \rho_i^*}{\Delta t}$, which results from the predicted displacement of all particles due to non-pressure accelerations. To achieve this, a pressure acceleration $\mathbf{a}_i^{\text{pres}} = -\frac{1}{\rho_i}\nabla p_i$ must be determined. This pressure acceleration corresponds to a velocity change:

$$-\Delta t \frac{1}{\rho_i} \nabla p_i,$$

which induces a position change which results in a density change:

$$\rho_i \nabla \cdot \left(-\Delta t \frac{1}{\rho_i} \nabla p_i \right) = -\Delta t \nabla^2 p_i,$$

as explained in [Kos+20].

To ensure incompressibility, a pressure acceleration, to cancel out the predicted density, has to be determined:

$$\frac{\rho^0 - \rho_i^*}{\Delta t} - \Delta t \nabla^2 p_i = 0.$$

Finally, the Pressure Poisson Equation is formulated as:

$$\frac{\rho^0 - \rho_i^*}{\Delta t} = -\Delta t \nabla^2 p_i \tag{17}$$

For clarity, the left-hand side of Equation (17) represents the compression error resulting from non-pressure forces, while the right-hand side contains the negative pressure corrections needed to restore incompressibility.

3.1.2 Discretization

Before using the PPE, the translation of it into a particle-based framework, suitable for SPH, is needed. First, the predicted density ρ_i^* is discretized by applying Equation (9) to the divergence of the intermediate velocity $\nabla \cdot \mathbf{v}^*$:

$$\rho_i^* \overset{16}{=} \rho_i - \Delta t \rho_i \nabla \cdot \mathbf{v}_i^* \overset{9}{=} \rho_i - \Delta t \sum_j m_j \mathbf{v}_{ij}^* \cdot \nabla W(\mathbf{x}_{ij}, h).$$

The left-hand side of Equation (17) is then rewritten as:

$$\frac{\rho^0 - \rho_i - \Delta t \sum_j m_j \mathbf{v}_{ij}^* \cdot \nabla W(\mathbf{x}_{ij}, h)}{\Delta t}. \tag{18}$$

The right-hand side of Equation (17) is discretized by tracing it back to the formulation of pressure acceleration. Then the divergence of the pressure acceleration is discretized using Equation (9):

$$-\Delta t \nabla^2 p_i = -\rho_i \nabla \cdot (\Delta t \mathbf{a}_i^{\text{pres}}) \overset{9}{=} \Delta t \sum_j m_j \mathbf{a}_{ij}^{\text{p}} \cdot \nabla W(\mathbf{x}_{ij}, h).$$

3.2 Pressure Solver

The basic approach of the pressure solver is to apply the PPE to every particle in the fluid domain. This results in n equations with n unknown pressure values p, forming a linear system. By solving this system, the exact pressure values are determined to cancel out the compression errors induced by non-pressure forces.

This section introduces the method used to solve the pressure values form the linear system of equations. IISPH implements a numerical technique, referred to as Relaxed Jacobi method, to solve the linear system.

As introduced, a linear system $\mathbf{Ap} = \mathbf{s}$ is considered, where each particle i in the fluid solves an equation of the form $(\mathbf{Ap})_i = s_i$ to determine the unknown pressure p_i. This notation represents the PPE for each particle, where $(\mathbf{Ap})_i$ and s_i correspond to the

left-hand side and right-hand side of the PPE (Equation (17)), respectively. The source term is:

$$s_i = \frac{\rho^0 - \rho_i - \Delta t \sum_j m_j \mathbf{v}_{ij}^* \cdot \nabla W(\mathbf{x}_{ij}, h)}{\Delta t} \tag{19}$$

describes the compression error that needs correction while the Laplacian:

$$(\mathbf{Ap})_i = -\Delta t \nabla^2 p_i = \Delta t \sum_j m_j \mathbf{a}_{ij}^{\mathrm{p}} \cdot \nabla W(\mathbf{x}_{ij}, h), \tag{20}$$

contains the unknown pressure accelerations requires to correct the implied compression error. Note that $\mathbf{a}_i^{\mathrm{p}}$ is calculated using Equation (11). So the Laplacian also contains unknown pressure value of particle i at neighbours and at neighbours of neighbours.

Next, the linear system $\mathbf{Ap} = \mathbf{s}$ is solved numerically using the Relaxed Jacobi method. This works by iteratively updating the unknown value p_i in $\mathbf{p}$, progressively approaching the exact solution. As shown in [Kos+20], p_i is updated by:

$$p_i^{(l+1)} = \max\left(p_i^{(l)} + \frac{\omega}{a_{ii}}\left(s_i - (\mathbf{Ap}^{(l)})_i\right), 0\right). \tag{21}$$

where $p_i^{(l)}$, ω, a_{ii}, s_i, and $(\mathbf{Ap}^{(l)})_i$ represent the pressure at iteration l, the relaxation coefficient, the diagonal element of the matrix $\mathbf{A}$, the source term, and the Laplacian at iteration l for particle i, respectively. Negative pressure is clamped to zero as proposed in [Ihm+14a] to avoid attraction effects which can lead to artificial cohesion and instability. Doing this does not affect the pressure gradient because pressure forces depend on differences in pressure rather than absolute values [Kos+20].

Updating the pressure using Equation (21) requires the computation of the diagonal element a_{ii}. It represents the influence of particle i on itself in terms of its displacement or force and is calculated by:

$$a_{ii} = \sum_j m_j \left(\underbrace{-\Delta t \sum_j \frac{m_j}{\rho_j^2} \nabla W(\mathbf{x}_{ij}, h)}_{\mathbf{d}_{ii}} - \underbrace{\Delta t \frac{m_j}{\rho_j^2} \nabla W(\mathbf{x}_{ij}, h)}_{\mathbf{d}_{ij}} \right) \cdot \nabla W(\mathbf{x}_{ij}, h) \tag{22}$$

as illustrated in [Ihm+14a]. Here $\mathbf{d}_{ii}$ and $\mathbf{d}_{ij}$ refers to the influence on the displacement where particle i is affected by itself and on the part where particle i is influenced by its neighbours j.

3.3 Implementation

In Section 2, a basic SPH simulation loop was illustrated. This section presents the implementation of the pressure solver and its integration into the resulting IISPH simulation step. First, the realisation of the Relaxed Jacobi pressure solver is detailed. Next, the pressure solver is incorporated into the basic SPH simulation loop. The IISPH simulation loop is demonstrated, and implementation details are clarified.

As described in Section 3.2, the pressure is iteratively updated to solve the linear system using Equation (21). This requires the computation of the terms $p_i^{(0)}$, a_{ii}, s_i, and $(\mathbf{Ap}^{(l)})_i$. The implementation consists of two main steps and is illustrated in Algorithm 2:

1. Since the initial pressure $p_i^{(0)}$, the diagonal element a_{ii}, and the source term s_i do not depend on the current iteration, they are calculated once at the start. This is done in a loop over all particles in the fluid domain. The source term and the diagonal element are computed using Equation (19) and Equation (22). The initial pressure is initialized using Equation (21) with $(\mathbf{Ap}^{(0)})_i = 0$ and $p_i^{(0)} = 0$, resulting in a state equation.
2. Secondly, a while loop is performed to iteratively update the pressure and solve the linear system. This loop is composed of two nested for loops:
   - In the first loop the pressure acceleration $\mathbf{a}_i^{\mathrm{p}}$ is computed using Equation (11).
   - In the second loop, the Laplacian $(\mathbf{Ap})_i$ is calculated using Equation (20). Finally, the pressure is updated using Equation (21).

   Typically, the relaxation coefficient is set to $\omega = 0.5$ to ensure stable convergence.

The last step is to define the break condition to stop the pressure update process. A predicted compression error $\rho_i^{\texttt{est\_err}}$ is computed by:

$$\rho_i^{\texttt{est\_err}} \leftarrow max(100 \cdot (\frac{\mathbf{Ap}_i - s_i}{\rho_0}), 0). \tag{23}$$

to keep track of the solver process. Since the goal of the pressure solver is to minimize fluid compressibility, an average compression error of every fluid particle is computed and a convergence criterion is chosen. Typically, an average predicted compression error of $\rho_{\texttt{avg}}^{\texttt{est\_err}} = 0.1\%$ is used.

While solving pressure in complex scenarios might lead to large iteration counts, the solver is limited by a maximal iteration count of 100. In addition IISPH requires at least 3 iterations [Cor+19].

Finally, Algorithm 2 illustrates a simulation step of IISPH. The implementation consists of four steps:

1. First, neighbouring particles are identified using Spatial Hashing, which is introduced in Appendix 7, and the density ρ is computed using Equation (13).
2. Next, intermediate velocities $\mathbf{v}_i^*$ are calculated based on external accelerations $\mathbf{a}_i^{\text{ext}}$ and viscosity accelerations $\mathbf{a}_i^{\text{v}}$ using Equation (12).
3. To ensure incompressibility, relaxed jacobi pressure solver is used to compute the pressures of the particles, minimising fluid compressibility.
4. Finally, the velocities and positions of the particles are updated.

In this implementation, the pressure acceleration is not calculated separately as it is shown in the base SPH simulation step (see Algorithm 1). Instead, it is already computed within the pressure solver, so there is no need to repeat this calculation.

Function `PressureSolver`**:**

- **foreach** *particle* i **do**
  - compute s_i ▷ with Equation (19)
  - compute a_{ii} ▷ with Equation (22)
  - initialize $p_i^{(0)} \leftarrow max(\frac{\omega}{a_{ii}} s_i, 0)$ ▷ with Equation (21)
- $l \leftarrow 0$ ▷ Initialize iteration counter
- **while** $(\rho_{avg}^{est\_err} > 0.1\% \vee l \leq 2) \wedge l \leq 100$ **do**
  - **foreach** *particle* i **do**
    - compute $\mathbf{a}_i^{\mathrm{p}} = -\frac{1}{\rho_i} \nabla p_i$ ▷ with Equation (11)
  - **foreach** *particle* i **do**
    - compute $(\mathbf{Ap})_i$ ▷ with Equation (20)
    - update Pressure $p_i^{(l+1)}$ ▷ with Equation (21)
    - $\rho_i^{\mathtt{est\_err}} \leftarrow max(100 \cdot (\frac{\mathbf{Ap}_i - s_i}{\rho_0}), 0)$
  - $l += 1$

Function `SimulationStep`**:**

- **forall** *particle* i **do**
  - find neighbours j ▷ see Appendix 7
  - compute Density ρ_i ▷ with Equation (13)
- **forall** *particle* i **do**
  - compute $\mathbf{a}_i^{\mathrm{v}} = \nu \nabla^2 \mathbf{v}_i$ ▷ with Equation (12)
  - $\mathbf{v}_i^* \leftarrow \mathbf{v}_i(t) + \Delta t(\mathbf{a}_i^{\mathrm{v}} + \mathbf{a}_i^{\mathtt{ext}})$
- `PressureSolver` () ▷ Solve pressure using Relaxed Jacobi
- **forall** *particle* i **do**
  - $\mathbf{v}_i(t + \Delta t) \leftarrow \mathbf{v}_i^* + \Delta t \mathbf{a}_i^{\mathrm{p}}$
  - $\mathbf{x}_i(t + \Delta t) \leftarrow \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$

Algorithm 2: IISPH Simulation Step with Relaxed Jacobi Pressure Solver

4 Boundary Handling

This chapter explores the representation and treatment of boundaries in SPH simulations, emphasizing their critical role in IISPH. Boundaries define the spatial limits and constraints of the fluid domain. Typically, the boundary geometry is discretized into particles, referred to as particle-based boundaries, representing physical objects such as walls, containers, or obstacles. Proper boundary treatment is essential in IISPH, as achieving incompressibility and stability depends on accurately resolving particle-boundary interactions.

First, this chapter discusses the representation of boundaries used in this work. Then, three boundary-handling methods are presented and analysed.

4.1 Representation

This work considers only particle-based boundaries. However, multiple approaches exist for representing such boundaries, each affecting how flexibly objects made of boundary particles can be modelled and manipulated.

A flexible method for representing particle-based boundaries is the use of non-uniform boundary samples, as described in [Aki+12] and further discussed in [Kos+20]. This approach assigns each boundary particle an individual volume and mass, allowing boundaries to be placed in a non-uniform manner, without requiring equal spacing between particles. This enables accurate and flexible modeling of complex geometries.

The volume of a boundary particle, V_b, is calculated using the kernel function W, which evaluates the influence of neighbouring boundary particles:

$$V_b = \frac{\gamma_1}{\sum_{b_b} W(\mathbf{x}_{bb_b}, h)},$$

where γ_1 is a scaling factor compensating for sparse sampling.

Based on this volume, the mass m_b of a boundary particle is computed to maintain consistency with the fluid's rest density ρ^0:

$$m_b = \rho^0 V_b.$$

Thus, the density equation Equation (13) is modified for fluid particles near the boundary:

$$\rho_i = \sum_{j_f} m_{j_f} W(\mathbf{x}_{ij_f}, h) + \gamma_2 \sum_{j_b} m_{j_b} W(\mathbf{x}_{ij_b}, h),$$

where j_f and j_b represent the fluid and boundary neighbours of particle i, respectively.

This representation ensures that boundary particles interact realistically with fluid particles, even when spaced non-uniformly. Boundary particles can be placed at different distances from each other, allowing for flexible and accurate modeling of complex geometries. The non-uniform sampling approach is particularly advantageous in scenarios with irregular boundary structures.

Additionally, this method supports dynamically changing boundaries, such as displacing and overlapping boundaries, by updating the volume and mass of boundary particles at each simulation step. This dynamic capability makes it well-suited for simulations requiring real-time interaction with deformable or mobile boundaries.

4.2 Pressure at Boundaries

This section introduces the computation of pressure at boundary particles. While the pressure of fluid particles is computed using local or global pressure solvers, boundary pressure is handled separately. For a fluid particle f, the calculation of pressure acceleration $\mathbf{a}_f^{\mathrm{p}}$ using Equation (11) is adapted to:

$$\mathbf{a}_f^{\mathrm{p}} = -\sum_{f_f} m_j \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W(\mathbf{x}_{ff_f}, h) - \gamma_3 \sum_{f_b} m_j \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_b}}{\rho_{f_b}^2} \right) \nabla W(\mathbf{x}_{ff_b}, h). \quad (24)$$

where f_f and f_b represent the fluid and boundary neighbours of fluid particle f, respectively.

The challenge lies in determining the unknown boundary pressure p_{f_b}. Three methods are presented to address this issue, with their impacts on performance and implementation discussed below.

4.2.1 Pressure Extrapolation

SPH pressure extrapolation is based on Pascal's law for hydrostatic pressure. As described in [Ban+18], the pressure of a specific boundary particle b is estimated using a fluid neighbour b_f:

$$p_b = p_{b_f} + \rho_{b_f} \mathbf{g} \cdot \mathbf{x}_{bb_f},$$

where $\mathbf{g}$ represents gravitational acceleration. To account for all fluid neighbours, their contributions are summed and weighted by the kernel function W, resulting in [AHA12; Ban+18]:

$$p_b = \frac{\sum_{b_f} p_{b_f} W(\mathbf{x}_{bb_f}, h) + g \cdot \sum_{b_f} \rho_{b_f} \mathbf{x}_{bb_f} W(\mathbf{x}_{bb_f}, h)}{\sum_{b_f} W(\mathbf{x}_{bb_f}, h)} \tag{25}$$

Pressure extrapolation requires computing the pressure for each boundary particle based on its fluid neighbours. This process scales linearly with the number of boundary particles, which introduces additional computational costs in solving pressure.

4.2.2 Pressure Mirroring

Pressure mirroring, as proposed in [Kos+20], provides a simpler approach by assigning pressure values to boundary particles based on the pressure of adjacent fluid particles.

This approach is based on the idea that, at the limit of very small particle sizes, the mirrored pressures closely resemble the values obtained through pressure extrapolation. By avoiding the need for explicit boundary pressure calculations, it reduces computational costs while maintaining accuracy at high particle resolutions.

Instead of directly calculating boundary pressure, pressure mirroring modifies Equation (24) by reflecting the pressure of neighbouring fluid particles. The resulting equation is:

$$\mathbf{a}_f^{\mathrm{p}} = -\sum_{f_f} m_j \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W(\mathbf{x}_{ff_f}, h) - \gamma_3 \sum_{f_b} m_j 2 \frac{p_f}{\rho_f^2} \nabla W(\mathbf{x}_{ff_b}, h),$$

where the first term sums over fluid neighbours, and the second term accounts for contributions from boundary particles by mirroring the pressure of adjacent fluid particles.

This method avoids explicit pressure computation for boundary particles, deriving their pressure by reflecting nearby fluid pressures, simplifying implementation while maintaining realistic boundary-fluid interactions.

4.2.3 Zero-Pressure Boundary

The Zero-Pressure Boundary method, proposed in [Aki+12], is the simplest boundary-handling approach among those considered. This method omits explicit pressure computation at boundary particles. Instead, boundary particles influence fluid particles indirectly by contributing to their density and pressure forces. As described in [Ihm+14a], this approach ensures computational simplicity and stability while maintaining the necessary interaction between the fluid and boundaries.

4.3 Diagonal element

Including the boundary into the IISPH pressure solver requires adapting Equation (22). Therefore $\mathbf{d}_{ii}$ is now considered for both, fluid f_f,and boundary f_b neighbour particles. The influence $\mathbf{d}_{ij}$ remains the same, only considering fluid neighbours f_f. Equation (22) is then rewritten as [Ihm+14a]:

$$\begin{aligned} a_{ii} = \sum_{j \in f_f \cup f_b} m_i \left(\underbrace{-\Delta t \sum_j \frac{m_j}{\rho_j^2} \nabla W(\mathbf{x}_{ij}, h)}_{\mathbf{d}_{ii}} \right) \cdot \nabla W(\mathbf{x}_{ij}, h) \\ + \sum_{j \in f_f} m_i \left(\underbrace{-\Delta t \frac{m_j}{\rho_j^2} \nabla W(\mathbf{x}_{ij}, h)}_{\mathbf{d}_{ij}} \right) \cdot \nabla W(\mathbf{x}_{ij}, h) \end{aligned} \tag{26}$$

5 Results

This chapter presents and illustrates the results of the IISPH solver and its implementation. The IISPH solver is integrated into an SPH .NET Framework called FlowLab (see [Mei25]). An optimized neighbour search method, referred to as Spatial Hashing (see 7), is used to accelerate the overall performance of the simulations. Both solver updates and neighbour queries are parallelized using SIMD and built-in parallelized loops, improving performance and allowing for higher particle counts, enabling more detailed and accurate simulations.

First, the accuracy and performance of the IISPH solver are demonstrated through a series of benchmark simulations. The process includes standard tests, such as the dam-breaking scenario and simulations involving moving and non-uniform boundaries, to demonstrate correct solver behaviour.

Second, the analysis focuses on solver behaviour by examining the impact of different parameters. The relationship between the number of solver iterations and factors like time step size and fluid height is investigated. Additionally, the effect of compression error on simulation stability is explored.

Next, boundary handling methods are compared, and optimal parameter settings for different approaches are determined.

Then, performance evaluation is conducted by measuring the computational time required for simulations under various conditions. The efficiency of the solver is analysed by determining the time step that maintains stability while optimizing computation time and accuracy.

Finally, IISPH is compared with WCSPH, highlighting differences in performance, stability, and usability to determine their respective advantages and limitations.

5.1 Demonstration

In this section, a series of benchmark simulations are presented to validate the operability and accuracy of the implemented IISPH solver with dynamic and non-uniform boundary handling. These tests serve to verify that the solver not only reproduces expected free-surface phenomena but also robustly handles complex boundary motions and geometries. The section begins with the classic breaking dam test and then moves on to scenarios featuring dynamic and non-uniform boundaries.

5.1.1 Breaking Dam

Figure 2 illustrates the breaking dam scenario using the IISPH method. It contains eight images, showing the progress of the simulation over time. The particle count is 130k, the time step is $\Delta t = 0.1s$ and the maximal compression error is set to 0.1%. The velocity is color-coded, with higher CFL values appearing in red, indicating high-velocity regions. The total computation time was five hours.

Overall, the images show how the fluid block in Figure 2a collapses under gravity. The mass forms a wave, rapidly accelerating toward the right side of the boundary box. The impact with the wall creates a secondary wave and splashes, propagating to the left side of the box (2c). This pattern repeats until the fluid stabilizes (2f). In Figure 2d and Figure 2e, turbulence are noticeable, caused by the impact of waves on the boundaries.

The simulation data are illustrated in Figure 3. The left plot shows the iteration count over time. The initial peak corresponds to the high-pressure gradients when the fluid first starts moving. The iteration count then stabilizes around the mean of 19.65 iterations, confirming that the solver efficiently resolves the pressure field.

The right plot presents the compression error over time, which stabilizes at 0.1%. This demonstrates that the IISPH solver effectively maintains incompressibility with minimal deviation. The initial oscillations correspond to the high-velocity impact phase, as the system adjusts to the sudden fluid motion.

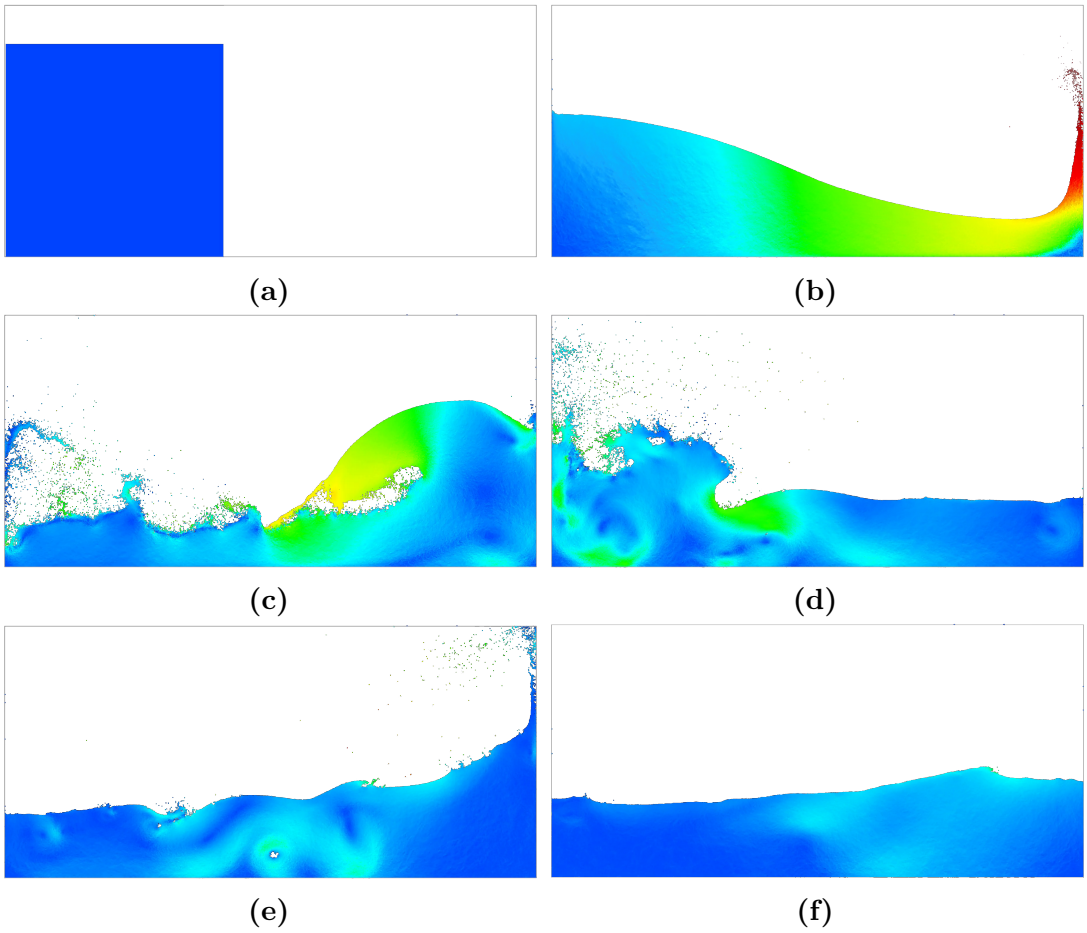


Figure 2: IISPH breaking dam simulation with 130k particles, a time step of $\Delta t = 0.1s$ and maximal compression of 0.1%. Velocity is color-coded, with higher CFL values appearing in red. Total computation time: 5h

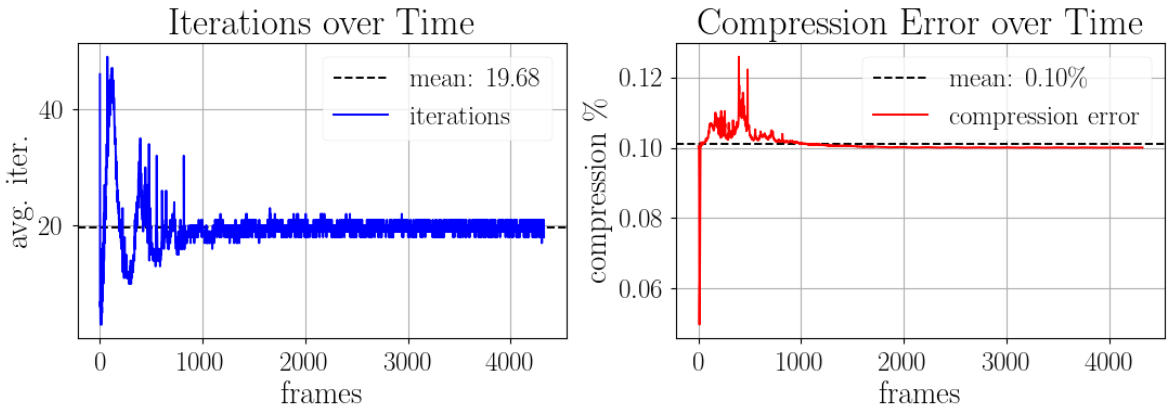


Figure 3: IISPH breaking dam simulation with 130k particles, a time step of $\Delta t = 0.1s$ and maximal compression of 0.1%. The left plot shows the average iteration count over time. The right plot presents the compression over time.

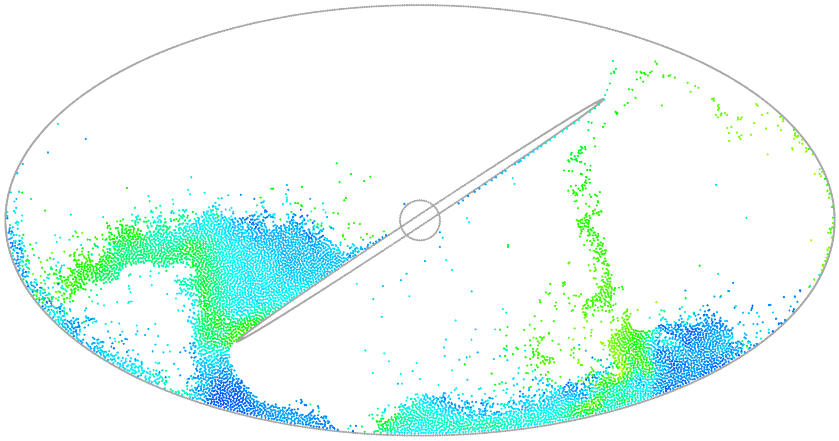


Figure 4: Simulation of 10k particles with IISPH in an oval container containing a rotor rotating clockwise. Velocity is color-coded, with higher CFL values appearing in red.

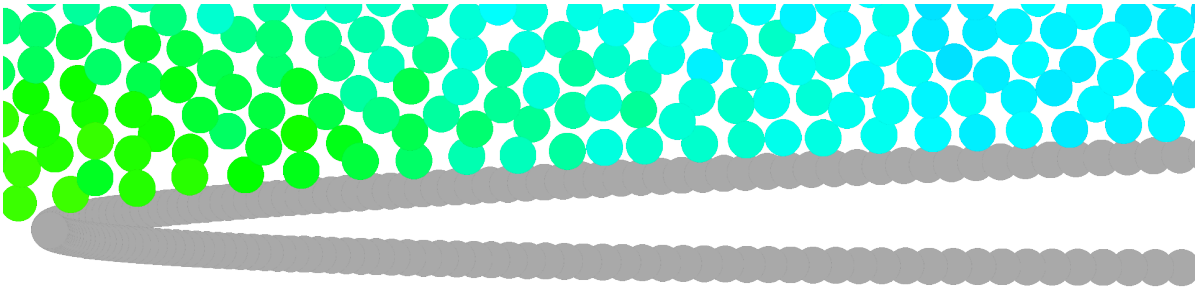


Figure 5: Zoom into the rotor spike region, showing detailed interactions of particles near the rotating structure. The image highlights the non-uniform representation of the particle boundary.

5.1.2 Dynamic and non-uniform boundaries

Figure 4 presents a simple simulation with 10k particles in an oval container, where the central rotor is scripted to move in a circular motion. The rotor induces fluid motion, generating a complex velocity field within the container.

A key aspect of this demonstration is the use of non-uniform dynamic boundaries to construct the rotor shape. Figure 5 provides a zoomed-in view of the rotor's spike region, where boundary particles are strategically placed in a non-uniform manner to accurately represent the structure. This flexibility allows for precise modeling of intricate geometries while maintaining stable and efficient simulations.

5.2 IISPH behaviour

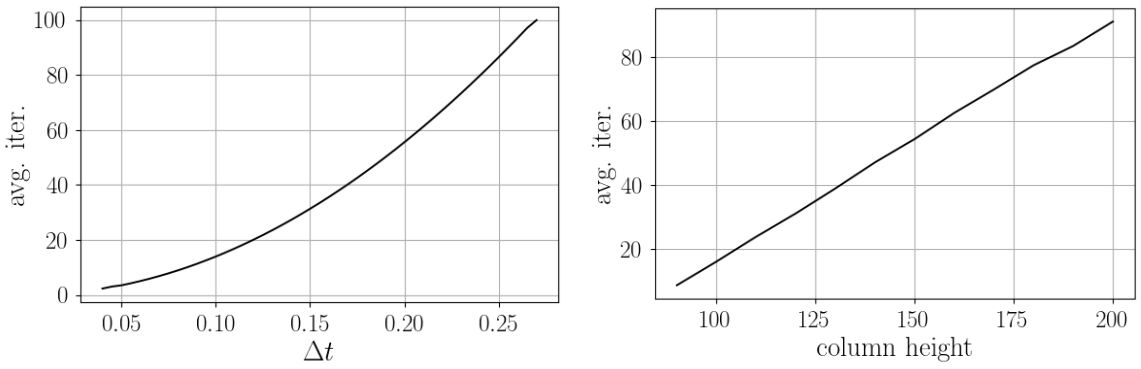
In this section, the behaviour of the IISPH solver is analysed. The results help in obtaining an impression of how parameters such as time step and the maximal compression error influence the behaviour and stability of the simulation. In addition, the influence of the simulation complexity on the solver is analysed and demonstrated.

5.2.1 Solver Iterations

A key aspect of the ISPH implementation is the iterative solution of the PPE. This section investigates the dependence of the required iteration count on both the time step size and the fluid height.

First, the influence of the time step is examined by simulating a fluid column with 5k particles. The objective is to observe changes in the average number of iterations required for different time steps Δt. To achieve this, the solver iterations are tracked while continuously adjusting the time step. Figure 6a presents the results of this experiment, showing that the average number of iterations increases as the time step becomes larger.

Second, the same setup is used, but this time with a fixed time step of $\Delta t = 0.2s$ while varying the fluid height. This is achieved by adding particles to the fluid, gradually increasing the height of the fluid column. As Figure 6b illustrates, an increase in fluid height also leads to a higher average number of iterations per simulation loop.



(a) 5k particles with different time steps Δt. **(b)** Different column height with $\Delta t = 0.2s$.

Figure 6: Average solver iterations of IISPH for a fluid column and a maximum volume compression error of 0.1%.

In both experiments, the average iterations rises due to the fact that the time step Δt and the fluid height lead to greater deviations from the target density. Larger time steps result in larger movement steps for the particles, leading to increased density changes that must be corrected by the PPE. Similarly, a deeper fluid column increases the density error in the fluid due to hydrostatic pressure. This effect is expected and physically accurate. Consequently, the solver requires more iterations to adjust the pressure field.

5.2.2 Compression Error

This section briefly analyses the behaviour of the compression error. Compression error is an important aspect of SPH, as realistic simulations require low compression errors to prevent noticeable volume changes. In IISPH, compression error is part of the break condition for the Jacobi solver. However, for this experiment, the solver will loop for a fixed number of iterations while tracking the compression error across different solver iteration counts.

Figure 7 illustrates the results of this experiment. The plot shows a decreasing compression error for higher solver iteration numbers. This behaviour is expected due to the nature of the Jacobi solver. As described in Section 3.2, the solver converges toward the optimal solution with more iterations. Approaching the optimal solution results in reduced volume collapse and a smaller compression error.

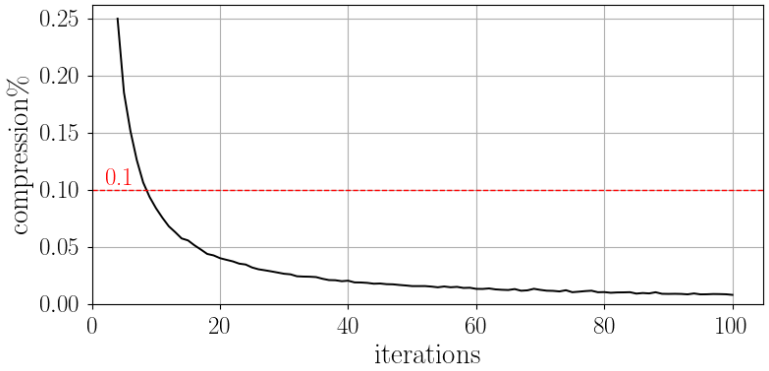


Figure 7: Compression Error of IISPH with different solver iterations for a fluid column and $\Delta t = 0.06s$

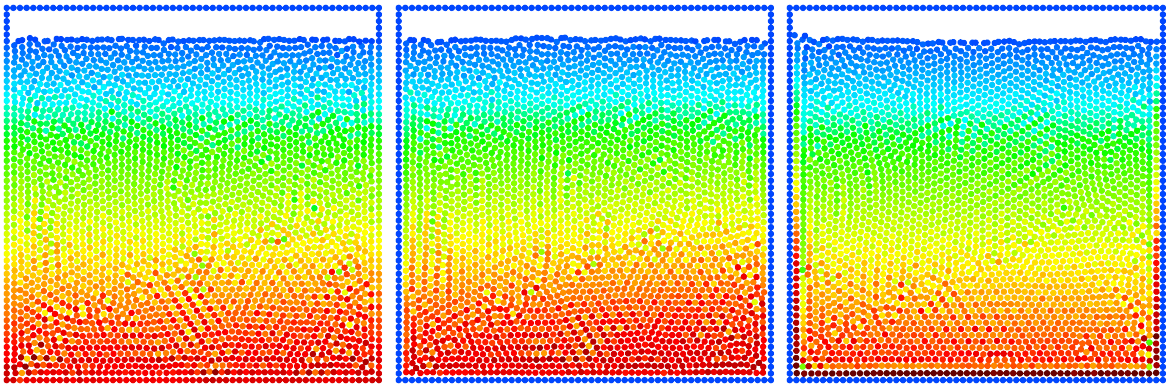
5.3 Boundary Handling

This section addresses boundary handling and its impact on the IISPH global pressure solver. The behaviour of the γ parameters, introduced in Chapter 4, are briefly analysed, and the optimal settings for this implementation are identified. Proper boundary handling is crucial for maintaining realistic pressure distribution and stability in the simulation.

As Figure 8 illustrates, boundary methods influence the internal pressure gradient of the fluid, yet they all consistently show an increase in pressure with fluid depth. Pressure extrapolation, as the name suggests, shows a seamless transition from fluid particles to boundary particles, demonstrating the realism and natural behaviour of the method. Pressure mirroring produces an almost identical internal pressure field to pressure extrapolation but results in zero pressure at the boundaries due to the pressure reflection embedded in Equation (24). Zero pressure leads to a weaker internal pressure field, with higher pressure values at fluid particles adjacent to boundary particles. These particles seems to support the entire fluid, forming a barrier and reducing fluid penetration at the boundary.

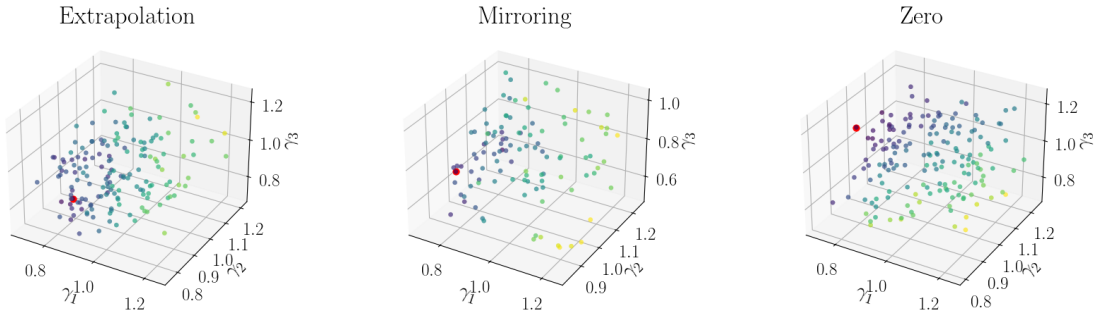
Due to different pressure gradients for each boundary handling method, the average number of iterations needed by the pressure solver varies. Furthermore, the γ parameters influence the stability and pressure gradient of the fluid, affecting solver convergence. Finding optimal settings that minimize the average iteration count helps improve the performance of the implemented solver and ensures numerical stability. To achieve this, arbitrary samples of $\gamma_1, \gamma_2, \gamma_3$ were chosen and tested. Figure 9 displays the data for each boundary handling method with its respective optimal γ values. Note that these values do not represent a global optimum but rather a local minimum within the tested parameter space.

With the given optimal settings, the solver iterations of each boundary handling method are compared and illustrated in Figure 10. The boxplot illustrates the iteration range, with its mean represented as a black dot. The results indicate that boundary handling has a measurable effect on solver efficiency. Overall, the differences are minimal, but Pressure Extrapolation shows slightly better performance, requiring fewer iterations on average.



(a) Pressure extrapolation. **(b)** Pressure mirroring. **(c)** Zero-pressure boundary.

Figure 8: Pressure gradient visualization for different boundary handling methods. The fluid pressure distribution is color-coded from blue (zero pressure) to red (maximum pressure), illustrating differences in pressure gradients across the methods.



(a) $\gamma_1 : 0.75$, $\gamma_2 : 0.95$, $\gamma_3 : 0.71$.

(b) $\gamma_1 : 0.81$, $\gamma_2 : 0.88$, $\gamma_3 : 1.20$.

(c) $\gamma_1 : 0.72$, $\gamma_2 : 0.98$, $\gamma_3 : 0.66$.

Figure 9: Optimal γ parameters from arbitrary samples. The average solver iterations are color-coded from yellow to purple, with optimal samples marked in red.

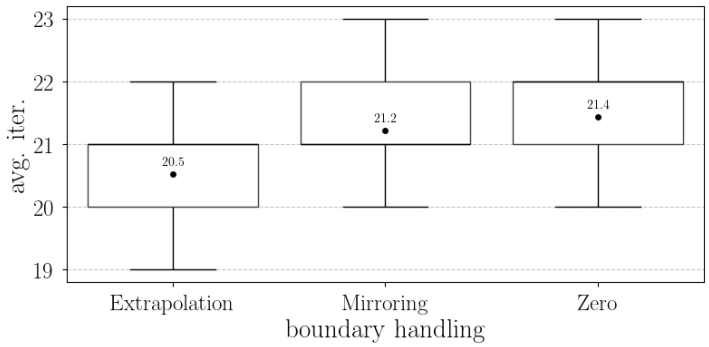


Figure 10: Solver iteration distributions for pressure-based boundary treatments (Extrapolation: 20.5, Mirroring: 21.2, Zero-pressure: 21.4) using optimized γ parameters. Extrapolation achieves the best convergence performance.

5.4 Performance of IISPH

This section first briefly demonstrates the benefits of the parallel processing implementation. Then the computation time of the IISPH implementation is illustrated and analysed.

To demonstrate the benefits of parallel processing, the computation time of the IISPH solver is compared between sequential and parallel processing. The simulation consists of a fluid column with zero to 12,000 particles, and the computation time is measured for both single-threaded and multi-threaded execution. As shown in Figure 11, the sequential implementation exhibits quadratic growth, while parallel processing significantly reduces computation time. This improvement in performance is crucial for handling large particle counts and achieving scalable simulations.

The quadratic growth in computation time for the sequential implementation is primarily due to the increasing fluid hight, leading to a higher number of iteration.

The analysis of the computation time is done by simulating a fluid column for a fixed number of frames and measuring the time required to complete the simulation. In this context, a frame represents a full time step. For example, if the time step is set to $\Delta t = 0.1$, it takes 10 simulation loops to complete one frame.

Table 2 shows the total computation time for 100 frames of a fluid column containing 5000 particles. The simulation is repeated for increasing time steps Δt, while tracking the average number of solver iterations, total computation time, and key component times such as neighbour query and pressure computation. Intuitively, one might expect that increasing Δt would reduce the overall computation time, leading to better performance. However, this is not entirely true.

While the total computation time does decrease with larger time steps, it reaches a minimum at $\Delta t = 0.065s$, where the solver achieves its best performance with an overall computation time of $17.93s$. This behaviour can be explained as follows: larger time steps require more solver iterations per simulation loop (see Figure 6a), increasing pressure computation time. At the same time, larger time steps reduce the number of simulation loops per frame, lowering the total time spent on neighbour queries. Since neighbour queries take almost the same amount of time per simulation loop, reducing the number of loops per frame directly lowers the overall neighbour query time. The combination of

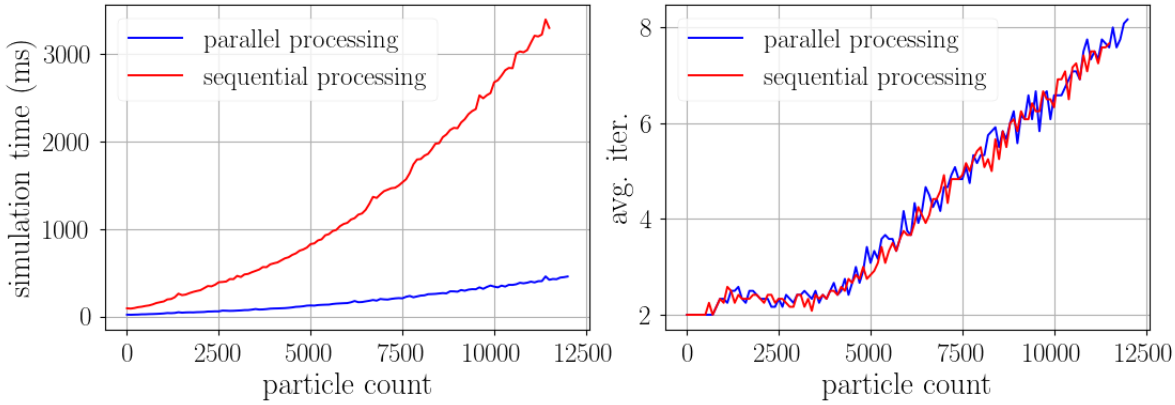


Figure 11: Comparison of IISPH performance between sequential and parallel processing. The left plot shows simulation time, where sequential processing exhibits quadratic growth, while parallel processing reduces computation time. The right plot shows the average solver iterations, increasing similarly for both approaches.

Performance of IISPH

		total computation times [s]		
Δt	avg. iter.	pressure	neighbour query	overall
0.05	3.5	9.71	4.62	18.96
0.055	4.23	9.74	4.19	18.06
0.065	5.9	10.89	3.57	**17.93**
0.07	6.85	11.64	3.32	18.23
0.075	7.86	12.75	3.1	18.94

Table 2: Performance analysis of 100 frames of IISPH for a fluid column with 5000 particles using different time steps. The table reports the average solver iterations, pressure computation time, neighbour search query time, and overall computation time. The lowest overall computation time is highlighted in bold.

increasing pressure computation time and decreasing neighbour query time leads to a "sweet spot", where the solver reaches peak efficiency.

The optimal performance range of IISPH is strongly influenced by the neighbour query computation time. Since neighbour search is directly tied to the number of simulation loops per frame, a more expensive neighbour search requires fewer loops to balance the trade-off. This shifts the optimal time step towards larger time steps, as fewer loops are needed to reach the point of minimum total computation time.

5.5 WCSPH or IISPH

This section compares the IISPH global pressure solver with the WCSPH local pressure solver. Both methods are evaluated in terms of performance, stability, and usability. Their respective benefits and drawbacks are discussed.

5.5.1 Performance

As mentioned in Section 2.3 and shown in Table 2 and Table 3, iteratively solving the PPE is more time-consuming than computing a state equation, with an optimal pressure computation time of 10.89s compared to 0.27s. However, IISPH is advantageous because it can operate with larger time steps while maintaining a nearly constant fluid volume. Figure 12 illustrates why WCSPH requires smaller time steps. As stiffness k increases, the compression error decreases. However, larger k values require smaller time steps to prevent numerical instability [BT07; Mei24].

Although IISPH can handle large time steps, Section 5.4 has shown that its optimal performance is not necessarily achieved at the largest possible time step. Thus, the choice between WCSPH and IISPH is not straightforward. As explained in [Kos+20], the decision to use a local pressure solver or IISPH depends on the simulation scenario. While WCSPH is less efficient for complex scenarios, it is preferable for simpler cases where IISPH would introduce unnecessary computational overhead. The complexity of a scenario depends more on the height or depth of the fluid under gravity rather than just the number of particles.

Performance of WCSPH

		total computation times [s]		
Δt	avg. error %	pressure	neighbour query	overall
0.01	0.1	0.8	23.58	59.21
0.02	0.1	0.41	11.88	30.11
0.03	0.1	0.27	7.93	**20.09**

Table 3: Performance analysis of 100 frames of WCSPH with different time steps and a stiffness $k = 55000$ for a fluid column containing 5000 particles. The lowest overall computation time is marked in bold.

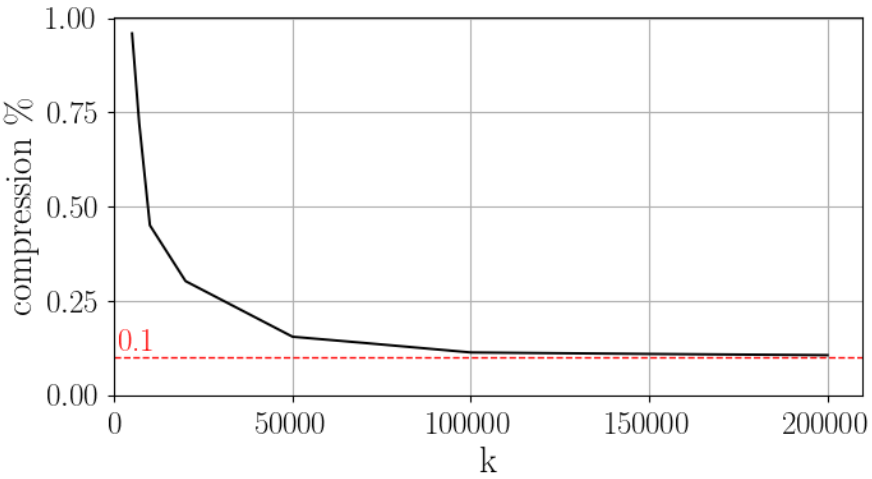


Figure 12: Compression Error of WCSPH with different stiffness k for a fluid column and $\Delta t = 0.001s$

For the tested simulation, IISPH demonstrates better overall performance with a total runtime of 17.93s, compared to 20.09s for WCSPH. This further reinforces that IISPH is more efficient for deeper fluid simulations.

5.5.2 Stability and Usability

As illustrated in [Mei24], the stability of a WCSPH simulation depends on the parameter settings. In contrast to IISPH, WCSPH primarily relies on fine-tuning the stiffness and time step for each simulation. This makes WCSPH both time-consuming and difficult to configure. Moreover, WCSPH introduces uncertainty regarding simulation stability, especially for complex scenarios where determining suitable parameters becomes even more challenging.

In IISPH, the compression error is explicitly controlled. Additionally, IISPH supports a wide range of time steps without sacrificing stability as shown in Figure 2. The time step is easier to estimate and is generally larger, making IISPH more predictable and stable across different simulation setups.

6 Conclusion

The primary focus of this work has been a specific Incompressible SPH implementation known as Implicit Incompressible SPH. IISPH introduces significant implementation challenges compared to Weakly Compressible SPH, as it requires solving a linear system of equations to determine a global pressure field. Furthermore, different boundary handling methods have been examined, playing a crucial role in SPH simulations, as accurate boundary interactions ensure realistic fluid behaviour and stability. Additionally, the performance of the IISPH solver has been optimized by parallelizing computations and implementing spatial hashing for a faster neighbour search. The main goal has been to analyse and understand the behaviour of IISPH, evaluate its stability and performance, and compare it to WCSPH.

The demonstration simulations have shown that IISPH works as expected, maintaining a nearly constant compression error and average solver iterations, with few fluctuations under complex conditions. Furthermore, experiments have highlighted that higher fluid density and larger time steps increase the initial density error, influencing the average solver iterations. The behaviour of the numerical Jacobi solver has been analysed, showing that increasing iterations reduces the actual fluid density error.

Next, different boundary handling implementations have been analysed and compared. First, it has been shown that different γ values have either a positive or negative impact on the average solver iterations. Then, optimal γ values were determined for each boundary handling method and compared, with Pressure Extrapolation showing slightly better performance compared to Pressure Mirroring and Zero-Pressure methods.

Lastly, the performance of IISPH has been investigated and compared with WCSPH. The experiments have shown that the largest possible time step does not necessarily yield the best performance. The data indicate that as the time step increases, the overall computation time for pressure increases while the neighbour query time decreases, forming a point where the solver achieves optimal performance. This highlights how neighbour

query computation time influences the optimal IISPH time step. The performance of WCSPH depends solely on the time step but is limited due to rapidly increasing instability for higher stiffness values and time steps.

IISPH outperforms WCSPH in complex, deep-fluid simulations by enabling larger time steps (Δt) while maintaining stability through explicit compression error control, achieving faster total runtimes (e.g., 17.93s vs. 20.09s). WCSPH, though computationally cheaper per iteration, requires intensive parameter tuning (stiffness k, smaller Δt) and struggles with stability in demanding scenarios. IISPH's robustness and minimal configuration make it ideal for dynamic boundaries and turbulence, whereas WCSPH remains practical for simpler, shallow-fluid cases. The choice hinges on simulation complexity: IISPH excels in depth-driven accuracy, while WCSPH suits basic setups prioritizing computational simplicity.

In summary, this work presents a computational framework for simulating incompressible fluids using Implicit Incompressible SPH (IISPH), addressing key challenges in accuracy, stability, and scalability. By integrating spatial hashing to optimize neighbour searches and introducing adaptable boundary-handling strategies (pressure extrapolation, mirroring, and zero-pressure), it enables efficient simulations of large-scale particle systems while maintaining physical realism. The IISPH algorithm employs a global pressure solver based on the Pressure Poisson Equation and the Relaxed Jacobi method to enforce incompressibility through iterative density corrections. This integration of spatial partitioning, boundary handling, and global pressure solving enhances SPH-based fluid modeling, making it a robust tool for both research and practical applications.

7 Appendix

Finding neighbouring particles is a performance-critical task that directly impacts the speed of an SPH fluid solver [BGT20]. A naïve approach, referred to as brute-force, checks the distance of each particle to every other particle, resulting in a quadratic time complexity $\mathcal{O}(n^2)$ in computation time as particle counts increase. Therefore, efficient methods for finding adjacent particles are essential to accelerate neighbour searches and achieve suitable simulation times for large particle counts.

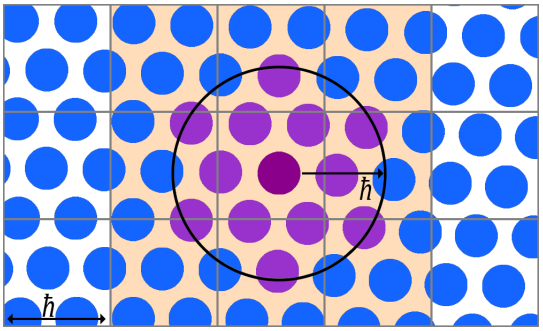


Figure 13: Illustration of Spatial Hashing: the computational domain is divided into uniform grid cells that store particles. Neighbour searches are limited to adjacent cells, significantly reducing computational complexity.

In this work, Spatial Hashing, as employed in [Tan+18; Ihm+11], is used for this task. The method works by dividing the spatial domain into discrete cells, each containing the particles within a specific region. These cells are stored in a hashmap, enabling fast access by hashing the positions being considered. Instead of searching the entire particle domain, the neighbour search is simplified by examining only the cells within the search radius.

To validate the efficiency of Spatial Hashing for neighbour search in SPH simulations, an experiment was conducted comparing it to a naïve quadratic search. The naïve approach checks all particle distances, resulting in $\mathcal{O}(n^2)$ complexity, whereas Spatial Hashing partitions space into grid cells, reducing the number of necessary distance checks.

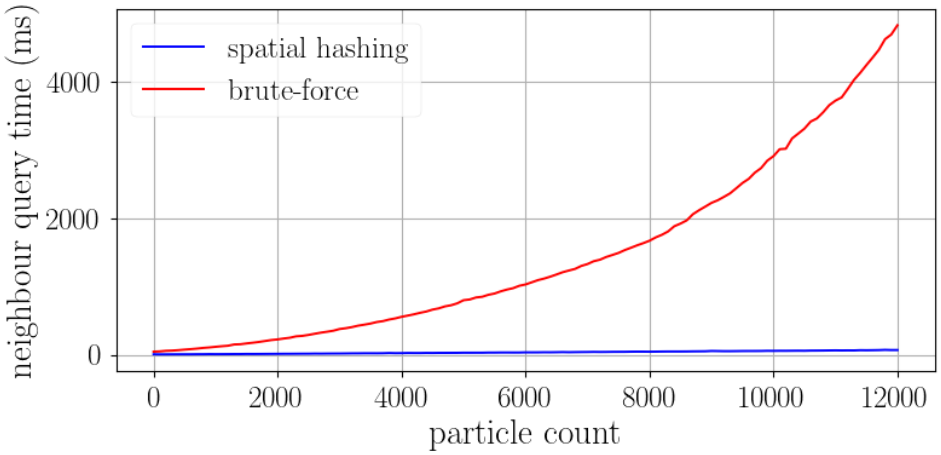


Figure 14: Comparison of neighbour search performance between the naïve approach and Spatial Hashing. The naïve approach exhibits quadratic growth, while Spatial Hashing remains nearly constant.

The experiment measured neighbour search times for varying particle counts, ranging from a few hundred to over 12,000. As shown in Figure 14, the naïve approach exhibits quadratic growth, making it infeasible for large-scale simulations. In contrast, Spatial Hashing maintains nearly constant query times, demonstrating its scalability.

Overall, Spatial Hashing significantly improves performance for simulations with large particle counts, as shown in [Mei24]. This efficiency enables the handling of larger particle systems, leading to more accurate and detailed simulations.

Bibliography

[Ada+07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. "Adaptively sampled particle fluids". In: *ACM Trans. Graph.* 26 (2007), 48–es.

[AHA12] Stefan Adami, Xiangyu Y. Hu, and Nikolaus A. Adams. "A generalized wall boundary condition for smoothed particle hydrodynamics". In: *Journal of Computational Physics* 231 (2012), pp. 7057–7075.

[Aki+12] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. "Versatile rigid-fluid coupling for incompressible SPH". In: *ACM Trans. Graph.* 31.4 (2012).

[Ban+18] Stefan Band, Christoph Gissler, Andreas Peer, and Matthias Teschner. "MLS pressure boundaries for divergence-free and viscous SPH fluids". In: *Computers & Graphics* 76 (2018), pp. 37–46.

[BGT20] Stefan Band, Christoph Gissler, and Matthias Teschner. "Compressed Neighbour Lists for SPH". In: *Computer Graphics Forum* 39.1 (2020), pp. 531–542.

[BK15] Jan Bender and Dan Koschier. "Divergence-free smoothed particle hydrodynamics". In: *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '15. Los Angeles, California: Association for Computing Machinery, 2015, pp. 147–155. ISBN: 9781450334969.

[BT07] Markus Becker and Matthias Teschner. "Weakly compressible SPH for free surface flows". In: *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '07. San Diego, California: Eurographics Association, 2007, pp. 209–217. ISBN: 9781595936240.

[Cor+19] Jens Cornelis, Jan Bender, Christoph Gissler, Dan Koschier, and Barbara Solenthaler. "An optimized source term formulation for incompressible SPH". In: *The Visual Computer* 35 (2019), pp. 579–590.

[DG96] Mathieu Desbrun and Marie-Paule Gascuel. “Smoothed Particles: A new paradigm for animating highly deformable bodies”. In: *Computer Animation and Simulation ’96*. Ed. by Ronan Boulic and Gerard Hégron. Vienna: Springer Vienna, 1996, pp. 61–76. ISBN: 978-3-7091-7486-9.

[Gis+19] Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. “Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling”. In: *ACM Trans. Graph.* 38 (2019).

[Gis+20] Christoph Gissler, Andreas Henne, Stefan Band, Andreas Peer, and Matthias Teschner. “An implicit compressible SPH solver for snow simulation”. In: *ACM Trans. Graph.* 39 (2020).

[HHM19] Libo Huang, Torsten Hädrich, and Dominik L. Michels. “On the accurate large-scale simulation of ferrofluids”. In: *ACM Trans. Graph.* 38 (2019).

[HY24] Haofeng Huang and Li Yi. *Journey into SPH Simulation: A Comprehensive Framework and Showcase*. Tech. rep. Tsinghua University, 2024. URL: `https://arxiv.org/abs/2403.11156`.

[Ihm+11] Markus Ihmsen, Nadir Akinci, Markus Becker, and Matthias Teschner. “A Parallel SPH Implementation on Multi-Core CPUs”. In: *Computer Graphics Forum* 30.1 (2011).

[Ihm+14a] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. “Implicit Incompressible SPH”. In: *IEEE Transactions on Visualization and Computer Graphics* 20 (2014), pp. 426–435.

[Ihm+14b] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. “SPH Fluids in Computer Graphics”. In: *Eurographics*. 2014.

[Kos+20] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. “SPH Techniques for the Physics Based Simulation of Fluids and Solids”. In: *arXiv preprint* arXiv:2009.06944 (2020). Eurographics Tutorial.

[Mei24] Thierry Meiers. *Einfacher SPH Flüssigkeitssimulatormit beschleunigter Nachbarschaftssuche*. Tech. rep. Albert Ludwig University of Freiburg, 2024. URL: `https://github.com/Meith0717/FlowLab/blob/main/projectReport.pdf`.

[Mei25] Thierry Meiers. *FlowLab: A Fluid Simulation Framework*. URL: `https://github.com/Meith0717/FlowLab`.

[Mon92] J. J. Monaghan. “Smoothed Particle Hydrodynamics”. In: *Annual Review of Astronomy and Astrophysics* 30 (1992), pp. 543–574.

[Pee+18] Andreas Peer, Christoph Gissler, Stefan Band, and Matthias Teschner. “An Implicit SPH Formulation for Incompressible Linearly Elastic Solids”. In: *Computer Graphics Forum* 37 (2018), pp. 135–148.

[PT24] Timo Probst and Matthias Teschner. “Unified Pressure, Surface Tension and Friction for SPH Fluids”. In: *ACM Trans. Graph.* 44 (2024).

[SP09] B. Solenthaler and R. Pajarola. “Predictive-corrective incompressible SPH”. In: *ACM Trans. Graph.* 28 (2009).

[Spr10] Volker Springel. “Smoothed Particle Hydrodynamics in Astrophysics”. In: *Annual Review of Astronomy and Astrophysics* 48.Volume 48, 2010 (2010), pp. 391–430.

[Tak+15] Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, Tomoyuki Nishita, and Ming C. Lin. “Implicit Formulation for SPH-based Viscous Fluids”. In: *Computer Graphics Forum* 34 (2015), pp. 493–502.

[Tan+18] Min Tang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. “PSCC: Parallel Self-Collision Culling with Spatial Hashing on GPUs”. In: *Proc. ACM Comput. Graph. Interact. Tech.* 1.1 (2018), pp. 1–18.

[Tesna] Matthias Teschner. *Simulation in Computer Graphics Particle Fluids 1.* Lecture, Albert Ludwig University of Freiburg.

[Wei+18] Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. “A Physically Consistent Implicit Viscosity Solver for SPH Fluids”. In: *Computer Graphics Forum* 37 (2018), pp. 145–155.

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Freiburg, February 27, 2025

Place, Date

Signature