Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids

Dan Koschier[1], Jan Bender[2], Barbara Solenthaler[3], and Matthias Teschner[4]

[1]University College London, UK
[2]RWTH Aachen University, Germany
[3]ETH Zurich, Switzerland
[4]University of Freiburg, Germany

Disclaimer: This document is work-in-progress and will be maintained and updated over a longer period. Please find the current version of the document under `InteractiveComputerGraphics.github.io/SPH-Tutorial`.

Abstract

Graphics research on Smoothed Particle Hydrodynamics (SPH) has produced fantastic visual results that are unique across the board of research communities concerned with SPH simulations. Generally, the SPH formalism serves as a spatial discretization technique, commonly used for the numerical simulation of continuum mechanical problems such as the simulation of fluids, highly viscous materials, and deformable solids. Recent advances in the field have made it possible to efficiently simulate massive scenes with highly complex boundary geometries on a single PC [Com16b, Com16a]. Moreover, novel techniques allow to robustly handle interactions among various materials [Com18, Com17]. As of today, graphics-inspired pressure solvers, neighborhood search algorithms, boundary formulations, and other contributions often serve as core components in commercial software for animation purposes [Nex17] as well as in computer-aided engineering software [FIF16].

This tutorial covers various aspects of SPH simulations. Governing equations for mechanical phenomena and their SPH discretizations are discussed. Concepts and implementations of core components such as neighborhood search algorithms, pressure solvers, and boundary handling techniques are presented. Implementation hints for the realization of SPH solvers for fluids, elastic solids, and rigid bodies are given. The tutorial combines the introduction of theoretical concepts with the presentation of actual implementations.

Keywords: Physically-based animation, Smoothed Particle Hydrodynamics, fluids, elastic solids, rigid bodies

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

The SPH concept is increasingly popular in a large variety of application areas that range from entertainment technologies to engineering. On the one hand, this popularity is based on the fact that Lagrangian approaches in general – and SPH in particular – can naturally handle scenarios that would be rather involved for Eulerian approaches. A favorable example is a free-surface fluid with geometrically complex and dynamic solid boundaries. Such settings are especially relevant for special effects productions in industry. The scenario, however, has the same relevance in engineering, *e.g.*, for the analysis of vehicles in water passages, for the prediction of rain water evacuation on a vehicle with moving wipers or for the design of gear boxes with optimized lubrication.

A second important aspect for the impressive advances in SPH based techniques is the fact that various research communities contribute to different aspects of SPH simulations. *E.g.*, the simulation community has a strong focus on the accuracy of SPH discretizations or on specific properties of the discretizations. Kernel functions and the effect of the size of kernel support domain are investigated. Effects of the sampling quality onto SPH approximations are analyzed, leading to concepts such as kernel gradient correction, particle shift, ambient pressure or density diffusion, just to name a few. The computer science community – the graphics community in particular – focuses on efficient algorithms for neighborhood searches, efficient pressure solvers, and flexible boundary handling. Also, pre- and post-processing is a typical graphics topic, *e.g.*, boundary sampling and visualization. The graphics community also experiments with combinations of different discretization concepts. *E.g.*, some projects have started to investigate combinations of SPH and MLS discretizations which is less typical in

the simulation community, where we currently see a strong focus on SPH within Lagrangian approaches with exclusive SPH conferences and SPH initiatives.

Still, simulation and computer science are different communities, but there is a growing acceptance of advances across communities. Graphics papers use state-of-the-art kernel functions, ranging from cubic spline to Wendland kernel types. The kernel gradient correction is employed in a growing number of approaches. Vice versa, the simulation community adopts efficient data structures for neighborhood searches, concepts for non-uniformly boundary samplings, and efficient pressure solvers.

This tutorial aims at a practical introduction of the SPH concept and its application in the simulation of fluids, elastic solids, and rigid solids. It starts with a description of the SPH concept and its usage for the interpolation of field quantities and for the computation of spatial derivatives. Then, the governing equations for fluids and solids are stated and the SPH concepts for the simulation of fluids and solids are outlined. In the following, various aspects of SPH simulations are explained in more detail. One of these aspects is the neighborhood search that is required for all SPH computations, as the interpolation of a quantity or a spatial derivative is always computed as a sum over adjacent particles. Another important aspect is incompressibility which is not only relevant for fluids, but also, *e.g.*, in the case of elastic solids. Next, boundary handling concepts are explained, *e.g.*, the interaction for fluid particles at solid walls, at free surfaces, *i.e.*, at the interface between fluid and air, or the interaction of particles from different fluids, *i.e.*, multiphase fluids. Other topics are viscosity, surface tension, and vorticity. Further, the SPH simulation of elastic solids and SPH-based contact handling between rigid bodies is described. Moreover, the techniques for the usage of SPH discretizations in data driven fluid simulations are presented. Finally, SPlisHSPlasH, an open-source library for the physically-based SPH simulation of fluids and solids, is introduced. The most important quantities that will be used throughout this tutorial are summarized in Tab. 1.

Variable	Description	Unit
d	Spatial dimension	–
A	Auxiliary function	–
t	Time	s
ρ	Volumetric mass density	$\mathrm{kg\,m^{-3}}$
p	Pressure	Pa
m	Mass	kg
Ψ	Pseudo-mass	kg
μ	Dynamic viscosity	Pa s
ν	Kinematic viscosity	$\mathrm{m^2\,s^{-1}}$
h	Smoothing length	m
$\hbar$	Kernel support radius	m
$\tilde{h}$	Particle size	m
σ	Kernel normalization factor	$\mathrm{m^{-3}}$
$\mathbf{x}$	Position vector of material particle	m
$\mathbf{r}$	Distance vector between two material particles	m
$\mathbf{u}$	Displacement of a material particle	m
$\mathbf{v}$	Velocity vector of material particle	$\mathrm{m\,s^{-1}}$
$\mathbf{a}$	Acceleration vector of material particle	$\mathrm{m\,s^{-2}}$
ω	Angular velocity vector of material particle	$\mathrm{rad\,s^{-1}}$
$\mathbf{f}$	Body force	$\mathrm{N\,m^{-3}}$
$\mathbf{F}$	Force	N
τ	Body torque	$\mathrm{N\,m\,m^{-3}}$
$\boldsymbol{\tau}$	Torque	N m
Θ	Microinertia	$\mathrm{m^2\,s^{-1}}$
$\mathbf{T}$	Cauchy stress tensor	$\mathrm{N\,m^{-2}}$
$\mathbf{P}$	1st Piola-Kirchhoff stress tensor	$\mathrm{N\,m^{-2}}$
$\mathbf{J}$	Deformation gradient	–
ε	Strain tensor	–
$\mathbf{E}$	Strain rate tensor	$\mathrm{s^{-1}}$

Table 1: Table of notation.

2. Foundations

In this section, we introduce the fundamental concept of SPH for the phenomenological simulation of fluids and solids. The section is primarily based on the excellent work of Price [Pri12] and Monaghan [Mon05] but, moreover, includes important theoretical and practical insights that we have gained over the years working on SPH based techniques.

We first show how the SPH formalism discretizes spatial quantities using a set of particles equipped with a *kernel* function. Secondly, we discuss the approximation quality that can be expected and provide practical examples to illustrate the consequences for physics-based simulations targeting computer graphics applications. Thirdly, we show how 1st- and 2nd-order differential operators are discretized and present specialized variants of the discrete operators tailored to specific circumstances. Finally, we give a brief introduction of the conservation law of linear momentum and the concept of stress in order to derive the governing equations for fluids and elastic solids and present a simple approach to simulate weakly compressible fluids using the knowledge that we have gained up to this point.

2.1. SPH Discretization

The concept of SPH can be generally understood as a method for the discretization of *spatial field quantities* and *spatial differential operators*, *e.g.*, gradient, divergence, curl, *etc*. In order to understand the basic idea, we first have to introduce the Dirac-δ distribution and the corresponding Dirac-δ identity. δ is a generalized function defined as

$$\delta(\mathbf{r}) = \begin{cases} \infty & \text{if} \quad \mathbf{r} = \mathbf{0} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and satisfies $\int \delta(\mathbf{r})\, dv = 1$.

To provide a physical intuition of what this distribution describes, consider the following example. In physics the mass of a body is usually defined as the spatial integral in the volumetric mass density, *i.e.*, $m =: \int \rho(\mathbf{x})\, dv$. However, if an idealized point mass is considered, the concept of a density function loses its meaning as the point mass has no spatial extents. In this case the density can not be described as a function, anymore, but collapses to the Dirac-

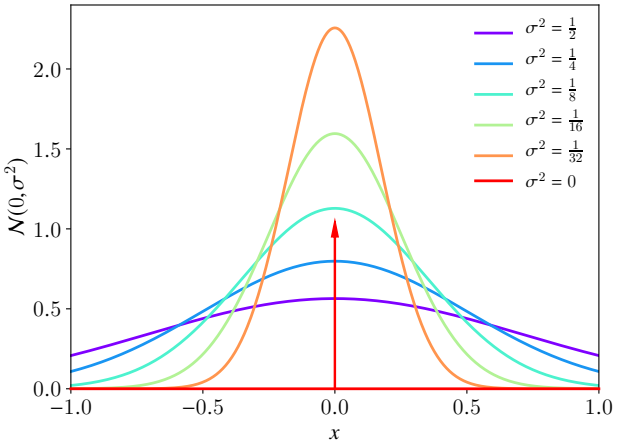


Figure 1: Gaussian bell function (normal distribution) $\mathcal{N}(0,\sigma^2)$ with varying variance σ^2. For $\sigma^2 \to 0$ the function approaches the Dirac-δ distribution. The arrow indicates a function value of ∞. The function family has non-compact support.

δ distribution scaled using the point mass. Another intuition of interpreting the Dirac-δ distribution is to understand it as the limit of the Gaussian normal distribution as the variance approaches zero (see Fig. 1).

Now that we have understood the Dirac-δ distribution, we can apply the Dirac-δ identity as the basis for the discretization. The identity states that the convolution of a continuous compactly supported function $A(\mathbf{x})$ with the Dirac-δ distribution is identical to A itself, *i.e.*,

$$A(\mathbf{x}) = (A * \delta)(\mathbf{x}) = \int A(\mathbf{x}')\delta(\mathbf{x}-\mathbf{x}')dv', \tag{2}$$

where dv' denotes the (volume) integration variable corresponding to $\mathbf{x}'$.

2.2. Continuous Approximation

We will later approximate the integral of Eq. (2) using a sum for numerical quadrature. Since $\delta(\mathbf{r})$ is, however, neither a function nor can be discretized, we first make a continuous approximation to the Dirac-δ distribution as a preparation to the discrete approximation of the integral. A natural choice to approximate δ is to use a normalized Gaussian since δ is equal to the normal distribution with zero variance. Consequently, convolving a field quantity A with a Gaussian effectively smoothes A. We will later see that the Gaussian is, however, not an optimal choice due to its non-compact support domain and will therefore consider more general smoothing functions $W : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}$ which we will refer to as *kernel functions* or *smoothing kernels*. Formally the continuous approximation to $A(\mathbf{x})$ with $W(\mathbf{r},h)$ is

$$\begin{aligned} A(\mathbf{x}) &\approx (A * W)(\mathbf{x}) \\ &= \int A(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)dv', \end{aligned} \tag{3}$$

where h denotes the kernel's smoothing length. The smoothing length controls the amount of smoothing and consequently how strongly the value of A at position $\mathbf{x}$ is influenced by the values in its close proximity. This means the smoothing effect increases with growing smoothing lengths. The following properties are furthermore desired:

$$\int_{\mathbb{R}^d} W(\mathbf{r}',h)dv' = 1 \qquad \text{(normalization condition)}$$

$$\lim_{h'\to 0} W(\mathbf{r},h') = \delta(\mathbf{r}) \qquad \text{(Dirac-}\delta\text{ condition)}$$

$$W(\mathbf{r},h) \geq 0 \qquad \text{(positivity condition)}$$

$$W(\mathbf{r},h) = W(-\mathbf{r},h) \qquad \text{(symmetry condition)}$$

$$W(\mathbf{r},h) = 0 \text{ for } \|\mathbf{r}\| \geq \hbar, \qquad \text{(compact support condition)}$$

$\forall\, \mathbf{r} \in \mathbb{R}^d, h \in \mathbb{R}^+$, where $\hbar$ denotes the support radius of the kernel function. Moreover, the kernel should be at least twice continuously differentiable to enable a consistent discretization of 2^{nd}-order partial differential equations (PDEs). It is essential to use a kernel that satisfies the first two conditions (normalization and Dirac-δ), in order to ensure that the approximation in Eq. (3) remains valid. The positivity condition is not strongly required (there are also kernels that do not have this property). However, in the context of physical simulations kernels that take negative values may lead to physically inconsistent estimates of field quantities, *e.g.*, negative mass density estimates, and should therefore be avoided. We will later see that the symmetry condition ensures 1^{st}-order consistency of the continuous approximation. Finally, ensuring that the kernel is compactly supported is a purely practical consideration that will come into play after discretizing the continuous integral and will be discussed later. To keep this tutorial practical, we refrain from discussing how to construct SPH kernels and would like to refer the reader to the review of Liu and Liu [LL10] for a discussion on kernel construction and an overview over a range of smoothing kernels suitable for SPH.

A typical choice for the smoothing kernel is the cubic spline kernel

$$W(\mathbf{r},h) = \sigma_d \begin{cases} 6(q^3-q^2)+1 & \text{for } 0 \leq q \leq \frac{1}{2} \\ 2(1-q)^3 & \text{for } \frac{1}{2} < q \leq 1 \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

with $q = \frac{1}{h}\|\mathbf{r}\|$. The kernel normalization factors for the respective dimensions $d = 1,2,3$ are $\sigma_1 = \frac{4}{3h}$, $\sigma_2 = \frac{40}{7\pi h^2}$, and $\sigma_3 = \frac{8}{\pi h^3}$. Please note that there exist different formulations of the cubic spline kernel throughout SPH literature that are differently parametrized with respect to h. This kernel fulfills all of the discussed kernel properties and has the particular advantage that its smoothing length is identical to the kernel support radius, *i.e.*, $h = \hbar$, which helps to avoid confusions in the implementation. For a graphical illustration please see Fig. 2. The plots demonstrate that the kernel is C^2-continuous. Therefore, derivatives of order > 1 are not really useful in practice due to the lack of smoothness. That is, however, not a major issue as there are more sophisticated approximations for 2^{nd}-order derivatives solely based on the kernel gradient. Otherwise, if

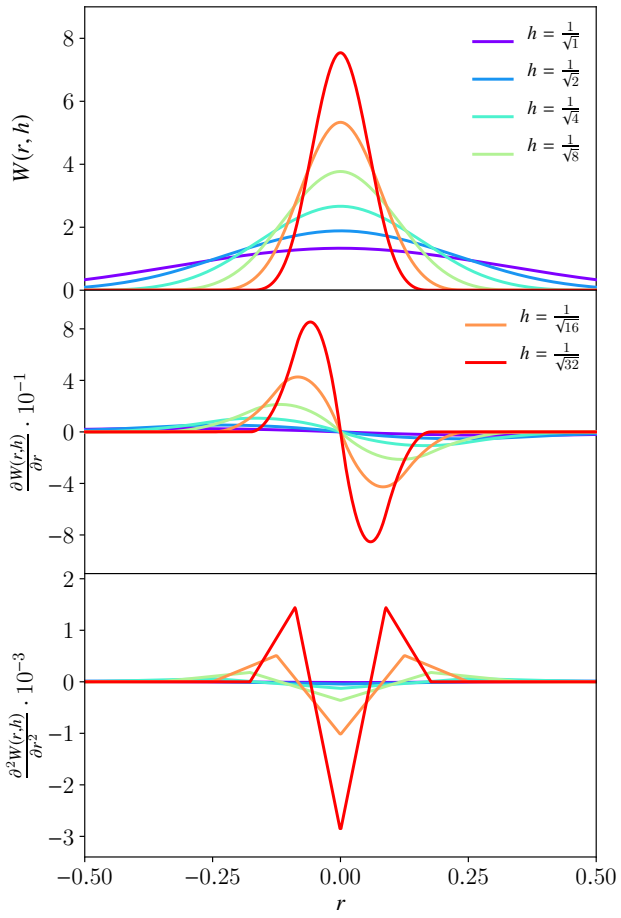


Figure 2: Graph of the cubic spline kernel (see Eq. (4)) and its derivatives.

desired, kernels of higher regularity can be found in the literature, *e.g.*, in the work of [LL10].

Let us consider the field $A : \mathbb{R}^d \to \mathbb{R}$. In order to investigate the accuracy of the continuous approximation, a Taylor series expansion of A in $\mathbf{x}'$ about $\mathbf{x}$ can be applied, *i.e.*,

$$(A*W)(\mathbf{x}) = \int \Big[A(\mathbf{x}) + \nabla A|_{\mathbf{x}} \cdot (\mathbf{x}' - \mathbf{x}) + \frac{1}{2}(\mathbf{x}' - \mathbf{x}) \cdot \nabla\nabla A|_{\mathbf{x}} (\mathbf{x}' - \mathbf{x}) + \mathcal{O}((\mathbf{x}' - \mathbf{x})^3) \Big] W(\mathbf{x} - \mathbf{x}', h) dv' \tag{5}$$

$$= A(\mathbf{x}) \int W(\mathbf{x} - \mathbf{x}') dv' + \nabla A|_{\mathbf{x}} \cdot \int (\mathbf{x} - \mathbf{x}') W(\mathbf{x} - \mathbf{x}') dv' + \mathcal{O}((\mathbf{x} - \mathbf{x}')^2). \tag{6}$$

It is trivial to see that the approximation of $(A*W)$ to A is 1^{st}-order accurate if the integral in the first term of Eq. (6) becomes 1, and if the integral in the second term vanishes. The first condition is automatically fulfilled if the kernel is normalized (*cf.*, normalization condition). The second condition is met if the kernel is symmetric (*cf.*, symmetry condition). Consequently, given a normalized, symmetric kernel we can expect that the approximation is (at least) able to exactly reproduce functions up to 1^{st}-order.

2.3. Discretization

The remaining step to realize the SPH discretization is to replace the analytic integral in Eq. (3) by a sum over discrete sampling points as follows:

$$(A*W)(\mathbf{x}_i) = \int \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')} W(\mathbf{x} - \mathbf{x}', h) \underbrace{\rho(\mathbf{x}') dv'}_{dm'} \tag{7}$$

$$\approx \sum_{j \in \mathcal{F}} A_j \frac{m_j}{\rho_j} W(\mathbf{x}_i - \mathbf{x}_j, h) =: \langle A(\mathbf{x}_i) \rangle, \tag{8}$$

where $\mathcal{F}$ is the set containing all point samples and where all field quantities indexed using a subscript denote the field evaluated at the respective position, *i.e.*, $A_j = A(\mathbf{x}_j)$. For improved readability, we will drop the second argument of the kernel function and use the abbreviation $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j, h)$ in the remainder of this tutorial. The physical interpretation of this is that we keep track of a number of points that "carry" field quantities. In this particular case, each point j has a certain location $\mathbf{x}_j$ and carries a mass sample m_j and a field sample A_j. It is not mandatory that the particle keeps track of its density ρ_j as this field can be reconstructed from its location and mass as explained later. Due to the analogy to physical particles the term *smoothed particle* has been coined in the pioneering work of Gingold and Monaghan [GM77]. Nevertheless, we would like to stress the fact that a set of SPH particles must not be misunderstood as discrete physical particles but simply as a spatial function discretization.

Analogously to the brief error analysis for the continuous approximation, a Taylor series expansion of $\langle A \rangle$ in $\mathbf{x}_j$ about $\mathbf{x}_i$ reveals the accuracy of the discretization

$$\langle A(\mathbf{x}_i) \rangle = A_i \sum_j \frac{m_j}{\rho_j} W_{ij} + \nabla A|_{\mathbf{x}_i} \cdot \sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) W_{ij} + \mathcal{O}((\mathbf{x}_j - \mathbf{x}_i)^2). \tag{9}$$

Due to the discretization the resulting approximation is only 1st-order accurate if

$$\sum_j \frac{m_j}{\rho_j} W_{ij} = 1 \quad \text{and} \quad \sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) W_{ij} = \mathbf{0}. \tag{10}$$

Even presuming that a normalized symmetric kernel is used, the conditions are highly dependent on the sampling pattern leading to the fact that not even a 0th-order consistent discretization can be guaranteed. In practice, however, the approximation is sufficiently accurate to approximate physical field functions to obtain realistic simulations. If desired, 0th-order consistency can be easily restored by normalizing the SPH approximation with $\sum_j \frac{m_j}{\rho_j} W_{ij}$ or even 1st-order consistency can be restored by the cost of a small matrix inversion (see [Pri12]).

To give the reader a notion of the quality of the discrete approximation of functions, we have discretized a linear and a quadratic polynomial as well as a trigonometric function using a fairly coarse SPH discretization equipped with the cubic spline kernel. The sampling pattern is illustrated in Fig. 3 while the the function and approximation graphs are depicted in Fig. 4. In this example we have

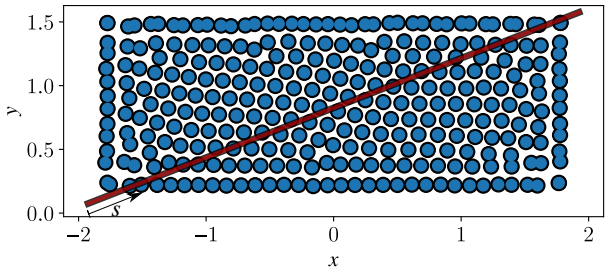


Figure 3: Point sampling of rectangular domain. Test functions are discretized using SPH. Function values are sampled along the red path parametrized by s.

used the cubic spline kernel with a smoothing length of $h = 0.3$m and particle masses $m_i = 18$kg. In order to find a suitable smoothing length given a dense (but not overlapping) sampling, we heuristically set the smoothing length to four times the particle radius, *i.e.*, $h = 2\tilde{h}$. We, also recommend this heuristic to estimate a good smoothing length in practice. In three-dimensional discretizations this leads to a number of approx. $30-40$ particles in a fully populated neighborhood.

Although no consistency can be strongly guaranteed in the absence of certain particle configurations that strongly fulfill the conditions in Eq. (10), the graphs demonstrate that even a coarse sampling results in a discretization with good accuracy away from the boundary of the particle set. The phenomenon of decreasing approximation quality in the close proximity of the domain boundary can be simply explained by the lack of sampling points outside the domain and is usually referred to as *boundary deficiency*. In the course of this tutorial practical solutions to this particular problem will be discussed. We would also like to assure the reader that even without further considerations to recover the consistency order, SPH based approaches are able to produce robust and highly-

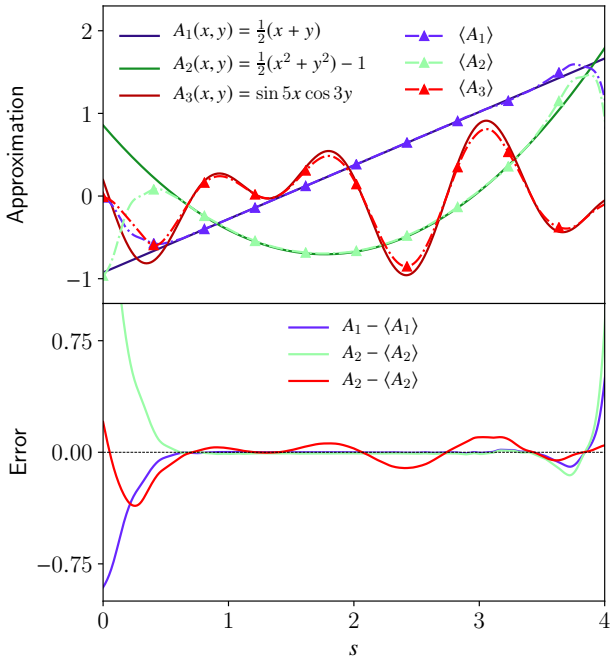


Figure 4: Comparison between analytic test functions and according SPH discretization using the sampling pattern illustrated in Fig. 3.

realistic results as demonstrated in countless publications that have been published within recent decades.

2.4. Mass Density Estimation

As previously mentioned, it is not required that the particles "carry" the mass density field as it can be reconstructed. Evaluating the density field at position $\mathbf{x}_i$ using the SPH discretization in Eq. (8) results in

$$\rho_i = \sum_j m_j W_{ij} \tag{11}$$

and is therefore solely dependent on the sample position and the mass field. Alternatively, the density can be tracked by discretizing the mass density field using the SPH sampling and by numerical integration of the continuity equation which describes the density evolution, *i.e.*, $\dot{\rho} = -\rho(\nabla \cdot \mathbf{v})$. However, as also discussed by Randles and Libersky [RL96], this approach is less robust and leads to accumulating errors in the density field due to the errors of the underlying numerical integration of the continuity equation.

Note that the density can be reconstructed at any position by Eq. (11) but the reconstructed density is typically underestimated at the free surface due to particle deficiency (*cf.*, Fig. 5). This must be considered when implementing a pressure solver as discussed in-depth in Section 4.

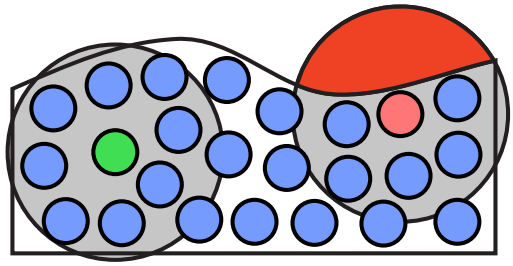


Figure 5: Particle deficiency problem. The green particle has a full neighborhood while the red particle at the free surface has no neighbors in the air. Therefore, its density is underestimated by Eq. (11).

2.5. Discretization of Differential Operators

Besides the discretization of field quantities, it is usually necessary to discretize spatial differential operators in order to numerically solve physical conservation laws. In the remainder of this tutorial, we will assume that the smoothing length h is constant in space (and time). Based on the discrete SPH approximation in Eq. (8) the gradient of the underlying field can be approximated straightforwardly using

$$\nabla A_i \approx \sum_j A_j \frac{m_j}{\rho_j} \nabla W_{ij}. \tag{12}$$

Given discrete representations of higher-dimensional functions, *e.g.*, $\mathbf{A} : \mathbb{R}^d \rightarrow \mathbb{R}^n$, even more complex first-order spatial differential operators can be directly discretized, *e.g.*,

$$\nabla \mathbf{A}_i \approx \sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \otimes \nabla W_{ij} \tag{13}$$

$$\nabla \cdot \mathbf{A}_i \approx \sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \cdot \nabla W_{ij} \tag{14}$$

$$\nabla \times \mathbf{A}_i \approx -\sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \times \nabla W_{ij}\,, \tag{15}$$

where $\mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^T$ denotes the dyadic product. Unfortunately, these "direct" derivatives lead to a poor approximation quality and unstable simulations. For this reason many discrete differential operators have emerged over time.

In this tutorial, we will cover the two most widely used formulations for first order derivatives, *i.e.*, the *difference formula* and the *symmetric formula*.

Difference Formula

Analyzing the error in the gradient based on Taylor series expansion (similar to the one carried out in Eq. (9)) reveals that the gradient estimate is only 0^{th}-order (1^{st}-order) accurate if the first (both) of the following constraints are fulfilled:

$$\langle \nabla 1 \rangle = \sum_j \frac{m_j}{\rho_j} \nabla_i W_{ij} = \mathbf{0} \quad \text{and} \quad \sum_j \frac{m_j}{\rho_j} (\mathbf{x}_j - \mathbf{x}_i) \otimes \nabla_i W_{ij} = \mathbb{1}\,. \tag{16}$$

In order to recover 0^{th}-order accuracy we can simply subtract the first error term of the Taylor series resulting in the improved approximation

$$\nabla A_i \approx \langle \nabla A_i \rangle - A_i \langle \nabla 1 \rangle = \sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla_i W_{ij}\,. \tag{17}$$

In the rest of this thesis we will refer to this gradient estimate as *difference formula*. The same formula can be straightforwardly applied to the higher-dimensional first-order differential operators presented in Eqs. (13) to (15). This gradient estimate finally results in a more accurate discretization but keep in mind that we still expect a linear error. However, linear accuracy is sometimes required and can be restored at the cost of solving a small linear equation system per evaluation, *i.e.*,

$$\begin{aligned} \nabla A_i &\approx \mathbf{L}_i \left(\sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla_i W_{ij} \right) \\ \mathbf{L}_i &= \left(\sum_j \frac{m_j}{\rho_j} \nabla_i W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \right)^{-1}. \end{aligned} \tag{18}$$

Symmetric Formula

Motivated from classical mechanics for hydrodynamical systems, a discrete formula for the pressure force/gradient, starting from the discrete Lagrangian and the density estimate, can be derived. This results in the following approximation

$$\begin{aligned} \nabla A_i &\approx \rho_i \left(\frac{A_i}{\rho_i^2} \langle \nabla \rho \rangle + \left\langle \nabla \left(\frac{A_i}{\rho_i} \right) \right\rangle \right) \\ &= \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla_i W_{ij}\,. \end{aligned} \tag{19}$$

Please note, that we did not include the lengthy derivation as this is out of the scope of this tutorial but kindly refer the reader to the report of Price [Pri12].

Since this formula does not satisfy the constraints in Eq. (16), it is clear that it is not able to exactly reproduce constant or linear gradient functions. However, the massive advantage of this is that discrete physical forces using this particular gradient estimate exactly conserve linear and angular momentum which is an essential criterion for robust simulations.

Deriving the criterion using Taylor series expansion of Eq. (19) reveals that the constant error of the symmetric gradient is governed by how much

$$\sum_j m_j \left(\frac{1}{\rho_i^2} + \frac{1}{\rho_j^2} \right) \nabla_i W_{ij} \approx \mathbf{0} \tag{20}$$

deviates from 0. As noted by Price [Pri12], the symmetric formulation "cares" about the particle ordering and the discrete physical forces will try to reorder the particle configuration until Eq. (20) is fulfilled. This is in contrast to forces formulated using the difference formula.

To summarize, the difference formula does indeed lead to a more accurate gradient estimate than the symmetric formula. In the context of physical forces the higher accuracy comes at the cost of a loss in momentum conservation and can therefore lead to unstable simulations. For the stated reasons, we recommend to use gradient

estimates of the symmetric type when quantities are discretized that directly affect particle trajectories, *e.g.*, physical forces, impulses, and to use the difference formula when 1st-order differentials are estimated for indirect use, *e.g.*, the velocity divergence during pressure solves.

2.5.1. Discretization of Laplace Operator

Similar to the direct 1st-order derivatives (Eqs. (13)-(15)) the Laplace operator can be directly discretized, *i.e.*,

$$\nabla^2 A_i \approx \sum_j \frac{m_j}{\rho_j} A_j \nabla_i^2 W_{ij}\,. \tag{21}$$

This, however, leads again to a very poor estimate of the 2nd-order differential. A improved discrete operator for the Laplacian was presented by Brookshaw [Bro85]:

$$\nabla^2 A_i \approx -\sum_j \frac{m_j}{\rho_j} A_{ij} \frac{2\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|}\,. \tag{22}$$

The main idea leading to this particular formulation is to effectively use solely a 1st-order derivative of the kernel function and to realize the second derivative using a finite-difference-like operation, *i.e.*, dividing by the particle distance.

2nd-order derivatives of vectorial field quantities are realized analogously resulting in

$$\nabla^2 \mathbf{A}_i = -\sum_j \frac{m_j}{\rho_j} \mathbf{A}_{ij} \frac{2\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|} \tag{23}$$

$$\nabla(\nabla\cdot\mathbf{A}_i) = \sum_j \frac{m_j}{\rho_j} \left[(d+2)(\mathbf{A}_{ij}\cdot\tilde{\mathbf{r}}_{ij})\tilde{\mathbf{r}}_{ij} - \mathbf{A}_{ij}\right] \frac{\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|}\,, \tag{24}$$

where d and $\tilde{\mathbf{r}} = \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|}$ denote the spatial dimension and the normalized distance vector between particles i and j, respectively. A problem of the discrete Laplace operator defined in Eq. (23) in the context of physics simulations is that forces derived using this operator, *e.g.*, viscosity forces, are not momentum conserving. Fortunately, we get the following expression by adding together Eqs. (23) and (24):

$$\sum_j \frac{m_j}{\rho_j} (\mathbf{A}_{ij}\cdot\tilde{\mathbf{r}}_{ij})\tilde{\mathbf{r}}_{ij} \frac{\|\nabla_i W_{ij}\|}{\|\mathbf{r}_{ij}\|} \approx \frac{\nabla(\nabla\cdot\mathbf{A}_i)}{d+2} - \frac{\nabla^2\mathbf{A}_i}{2(d+2)}\,. \tag{25}$$

This identity has the important consequence that in the case of a divergence-free vector field, *i.e.*, $\nabla\cdot\mathbf{A}=0$, the Laplace operator can be discretized using

$$\nabla^2 \mathbf{A}_i \approx 2(d+2)\sum_j \frac{m_j}{\rho_j} \frac{\mathbf{A}_{ij}\cdot\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^2} \nabla_i W_{ij}\,, \tag{26}$$

resulting in forces composed of terms that solely act along the "line of sight" between two interacting particles i and j. This particular choice has the advantage that derived physical forces recover momentum conservation [Pri12]. Therefore, we recommend to use Eq. (26) as discrete Laplace operator for divergence-free vector fields. In order to improve readability, we will drop the differentiation index for differential operators in the remainder of this tutorial. We will use the convention that the spatial operators always differentiate with respect to the variable according to the first index such that *e.g.*, $\nabla W_{ij} \equiv \nabla_i W_{ij}$.

2.6. Governing Equations for Fluids and Solids

In order to simulate the dynamic behavior of fluids and solids, a mathematical model that describes physical phenomena and motion of the matter is required. In computer graphics related research, we are generally interested in the appearance of objects and fluids in motion on humanly perceivable scales which is dominantly governed by the matter's macroscopic behavior. An important class of mathematical models that describe the macroscopic mechanical behavior of fluids and solids is based on continuum theory. Unfortunately, we can not cover an introduction to continuum mechanics as this is out of the scope of this tutorial. For a thorough introduction we would like to refer the reader to the works of Abeyaratne [Abe12] and Lai et al. [LKR09]. Nevertheless, we would like to informally describe the basic idea of continuum theoretical models in the following.

Physics teaches us that all matter is formed out of discrete particles such as atoms, molecules, *etc*. Therefore, we know that the distribution of mass within matter is not continuous but can rather be interpreted as a system of discrete mass points. Nonetheless, the vast majority of macroscopic mechanical phenomena can be accurately described when the corresponding matter is idealized as a continuum, *i.e.*, a region of continuously distributed mass. This idealization then implies that a portion of matter can always be divided into smaller portions independent of the size of the regions. This in turn confirms the theoretical existence of a *material particle*, *i.e.*, a portion of matter contained in an infinitesimal volume. Continuum theory then aims to model macroscopic physical phenomena and neglects effects that can be observed on microscales. In the following, we will summarize the most important local conservation laws required for the numerical simulation of (in)compressible fluids and solids.

Continuity Equation

The continuity equation describes the evolution of an object's mass density ρ over time, *i.e.*,

$$\frac{D\rho}{Dt} = -\rho\,(\nabla\cdot\mathbf{v})\,, \tag{27}$$

where $\frac{D(\cdot)}{Dt}$ denotes the *material derivative*. This relation is especially important when incompressible materials are modelled. In this particular case the constraint

$$\frac{D\rho}{Dt} = 0 \quad\Leftrightarrow\quad \nabla\cdot\mathbf{v} = 0 \tag{28}$$

has to be fulfilled at every material point and at all times within the described matter.

A note on the material derivative: The material derivative describes the time rate of change of a field quantity at a material point. It is important to understand that the explicit form of the material derivative is dependent on the type of coordinates that are used to the describe the system. *Eulerian coordinates* describe a field quantity at spatially fixed points in space, observing the motion of the continuum as time passes. This type of coordinates is usually employed for mesh-based simulation techniques for fluids. In contrast, *Lagrangian coordinates* "track" the individual material particles as they move through space and time. Lagrangian co-

ordinates are commonly employed for the particle based simulation of fluids, such as SPH, or the mesh-based simulation of elastic solids. Given the same field quantity once described in Eulerian coordinates $A^E(t,\mathbf{x})$ and Lagrangian coordinates $A^L(t)$ the material derivative has the following explicit forms

$$\frac{DA^E}{Dt} = \frac{\partial A^E}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{x}} A^E \quad \text{and} \quad \frac{DA^L}{Dt} = \frac{\partial A^L}{\partial t}. \tag{29}$$

The second term of the material derivative for Eulerian coordinates is referred to as *convection term* or *self-advection* term. As opposed to some people's beliefs, the convection term is non-existent when a quantity is described in Lagrangian coordinates. In the remainder of this tutorial, we will exclusively describe quantities using Lagrangian coordinates.

2.6.1. Conservation Law of Linear Momentum

The conservation law of linear momentum can be interpreted as a generalization of Newton's second law of motion for continua and is also often called the *equation of motion*. It states that the rate of change of momentum of a material particle is equal to the sum of all internal and external volume forces acting on the particle, *i.e.*,

$$\rho\frac{D^2\mathbf{x}}{Dt^2} = \nabla\cdot\mathbf{T} + \mathbf{f}_{\text{ext}}, \tag{30}$$

where $\mathbf{T}$ denotes the stress tensor and $\mathbf{f}_{\text{ext}}$ body forces – we understand a body force as a force per unit volume. This equation is independent of the material of the underlying matter as the material's behavior is "encoded" in the stress tensor and described using so-called constitutive laws.

Navier-Stokes Equation A typical constitutive relation for incompressible flow is

$$\mathbf{T} = -p\mathbb{1} + \mu(\nabla\mathbf{v} + \nabla\mathbf{v}^T), \tag{31}$$

where p and μ denote the pressure and dynamic viscosity of the fluid. If the incompressibility is intended to be strongly enforced, the pressure p can be interpreted as a Lagrange multiplier that has to be chosen such that the Eq. (28) is fulfilled. If strong enforcement of incompressibility is not required, the constitutive relation can be instead completed by a so-called *state equation* that relates geometric compression (changes in mass density) with the pressure, *i.e.*, $p = p(\rho)$ (see Section 4.4). A simple example for a state equation is a variation of the ideal gas equation that linearly penalizes deviations from a rest density ρ^0 scaled by a positive stiffness factor k resulting in $p(\rho) = k\left(\frac{\rho}{\rho^0} - 1\right)$.

By plugging Eq. (31) into Eq. (30) we arrive at the incompressible Navier-Stokes equation

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{v} + \mathbf{f}_{\text{ext}}. \tag{32}$$

Elasticity The stress tensor of elastic solids is solely dependent on the geometric deformation of an object, *e.g.*, $\mathbf{T} = \mathbf{T}(\mathbf{J})$, where $\mathbf{J}$ denotes the deformation gradient which will be later introduced in Section 10. Obviously, the constitutive model can be augmented accordingly, if viscoelastic, plastic, thermoelastic, or other deformation inducing phenomena have to be modeled.

2.7. Mixed Initial-Boundary Value Problem

The previously introduced linear momentum conservation law (Eq. (30)) in combination with a constitutive relation, *e.g.*, Eq. (31), is a PDE in time and space that describes the motion of any object composed of the material modeled by the constitutive law. In order to model a specific problem and to ensure a unique solution, initial conditions, *i.e.*, the initial shape and velocity of the object at every point, and boundary conditions constraining the position and/or velocity field have to be specified. As there is, in general, no known analytic solution to the mixed initial-boundary value problem in arbitrary scenarios, numerical solving is inevitable and requires discretization of the assiociated differential operators. In the previous sections, we have seen several discrete differential operators based on the SPH formalism that can be employed to discretize the spatial differential operators. After spatial discretization, we are left with a system of ordinary differential equations (this methodology is often called *method of lines*) that is typically discretized using standard time integration schemes such as the implicit or explicit Euler method, Runge-Kutta schemes, *etc*. In the remainder of this tutorial, we will see several variations of these discretizations tailored to specific problems in physics based simulation.

2.8. Operator Splitting

Before we will discuss a simple example of a complete simulation loop, the concept of operator splitting is introduced. Its importance is emphasized by the fact that the vast majority of today's SPH based simulators follow the concept. The basic idea is to decompose the underlying PDE, *e.g.*, the Navier-Stokes equation in the case of fluids, into several sequential subproblems and to employ individual techniques for solving each subproblem. This simplifies the complexity of the overall problem and sometimes also decouples field variables such as velocity and pressure in the numerical solver. It moreover allows us to use stable implicit updates for stiff subproblems while cheap explicit updates for the remaining terms can be used. A potential operator split for the incompressible Navier-Stokes equation (Eq. (32)) for low-viscous fluids with strong enforcement of the incompressibility constraint (Eq. (28)) might look as described in the following. Given the current geometry of the continuum $\mathbf{x}(t)$ and its velocity field $\mathbf{v}(t)$ at time t, we split the problem into a sequence of subproblems in order to obtain $\mathbf{x}(t+\Delta t)$ and $\mathbf{v}(t+\Delta t)$:

1. Update $\mathbf{v}$ by solving $\frac{D\mathbf{v}}{Dt} = \nu\nabla^2\mathbf{v} + \frac{1}{\rho}\mathbf{f}_{\text{ext}}$,
2. determine ∇p by enforcing $\frac{D\rho}{Dt} = 0$,
3. update $\mathbf{v}$ by solving $\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p$ and
4. update $\mathbf{x}$ by solving $\frac{D\mathbf{x}}{Dt} = \mathbf{v}$,

where $\nu = \frac{\mu}{\rho}$ denotes the kinematic viscosity. In this way, the "weaker" forces could be handled using explicit time integration while we can solve for the pressure gradients using a more sophisticated implicit solver in order to keep the simulation robust for large time steps. It should further be noticed that the individual steps are not performed in parallel but the updated fields (in this case $\mathbf{v}$ and ∇p) are fed forward into the next substep resulting in a somewhat implicit handling which has demonstrated to improve stability in practice as also discussed by Bridson [Bri15] for grid-based fluid simulation.

2.9. Time Integration

As previously described, an SPH discretization of the underlying PDE leaves us with a system of ordinary differential equations (ODEs) in time following the method of lines. This, of course, requires us to discretize the ODE in time. Due the operator splitting approach, as introduced in the previous section, each individual subproblem has to be numerically integrated in time. Theoretically, a different time integration scheme can be employed for each individual step. In, practice most methods mainly rely on simple and efficient explicit time integration schemes. The, by far, most frequently used scheme is the semi-implicit Euler scheme, as *e.g.*, employted in [BK17, IAAT12, SB12, ICS*14]). The integration scheme is often also referred to as symplectic Euler or Euler-Cromer scheme. Sometimes it is useful to solve some of the individual substeps using implicit time integration schemes to ensure stability in the case of "stiff" forces. A typical example where this strategy is employed is in the case of simulating highly viscous fluids. Here, the viscosity force is often integrated implicitly using the implicit Euler scheme as discussed in Section 6.

Naturally, we aim for the best performance of our simulator and, therefore, try to use a very large time step width Δt[†]. However, we also understand that choosing an overly large time step width results in decreased accuracy of the numerical approximation and may lead to a less stable simulation which might ultimately result in a breakdown of the simulation. In the context of computer graphics research, we care most about carrying out a robust and stable simulation in a resource-efficient manner while the numerical accuracy is often of subordinate importance. This does not mean that we do not care about accuracy at all, as the realism of the resulting animations often improves with better accuracy of the numerical approximation. We are simply putting a higher priority on maintaining a robust and stable simulation under extreme conditions and in highly complex scenarios than on achieving the highest possible accuracy.

In order to find a "good" time step width Δt that is as large as possible to achieve high performance but sufficiently small to maintain stability, the vast majority of approaches adaptively estimate the time step using a heuristic based on the Courant-Friedrichs-Lewy (CFL) condition. The CFL condition is a necessary condition for the convergence of numerical solvers for differential equations and, as a result, provides an upper bound for the time step width, *i.e.*,

$$\Delta t \leq \lambda \frac{\tilde{h}}{\|\mathbf{v}^{\max}\|}, \tag{33}$$

where $\tilde{h}$, $\mathbf{v}^{\max}$, and λ denote the particle size, the velocity at which the fastest particle travels and a user-defined scaling parameter, respectively. The intuition behind this condition is that all particles are only allowed to move less than the particle diameter per time step for $\lambda = 1$. As this is only a necessary but no generally sufficient condition, the scaling parameter is heuristically chosen to keep the simulation stable, *i.e.*, $\lambda \approx 0.4$ [Mon92]. This can not strongly guarantee stability but experience from practice has shown that the condition typically leads to stable simulations [SP09, ICS*14, BK17]. Although obvious from Eq. (33), we would like to stress the fact that the maximally allowed time step decreases with higher velocities and spatial resolution. We would further like to point out that it is in practice useful to specify global bounds, *i.e.*, a lower and upper bound, for the time step as we want to produce a certain number of frames per second and want to avoid that the simulation comes to halt if a single particle moves with very high velocity.

[†] We will later see that larger time step widths not always result in better performance. This is especially true when iterative pressure solvers are employed (see Sec. 4).

2.10. Example: Simple Fluid Simulator

Based on the knowledge that we have acquired up to this point, we are now able to implement a simple state-equation based simulator for weakly compressible fluids with operator splitting using SPH and symplectic Euler integration.

for all *particle i* **do**
  Reconstruct density ρ_i at $\mathbf{x}_i$ with Eq. (11)
for all *particle i* **do**
  Compute $\mathbf{F}_i^{\text{viscosity}} = m_i \nu \nabla^2 \mathbf{v}_i$, *e.g.*, using Eq. (23)
  $\mathbf{v}_i^* = \mathbf{v}_i + \frac{\Delta t}{m_i}(\mathbf{F}_i^{\text{viscosity}} + \mathbf{F}_i^{\text{ext}})$
for all *particle i* **do**
  Compute $\mathbf{F}_i^{\text{pressure}} = -\frac{1}{\rho}\nabla p$ using state eq. and Eq. (19)
for all *particle i* **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \frac{\Delta t}{m_i}\mathbf{F}_i^{\text{pressure}}$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i + \Delta t \mathbf{v}_i(t+\Delta t)$

Algorithm 1: Simulation loop for SPH simulation of weakly compressible fluids.

The few lines in Algorithm 1 are already enough to implement a simple fluid solver. However, the algorithm does, unfortunately, not handle boundary conditions. A practical workaround to model boundaries in the discrete model is to sample the boundary geometry with static (non-moving) fluid particles. The pressure forces will then "push away" particles that attempt to penetrate the boundary. A more consistent handling of boundary conditions will be discussed in Section 5.

3. Neighborhood Search

A major insight that we can gain from Algorithm 1 is that evaluating the individual force terms is rather inefficient. It requires to compute the previously defined discrete differential operators which in turn require to compute a sum over all particles resulting in a runtime complexity of $\mathcal{O}(n^2)$, where n is the number of particles. If we, however, use a smoothing kernel that fulfills the *compact support condition*, most terms of the sums vanish since the kernel function and its derivatives for particles that are further away from i than the kernel support radius $\hbar$ vanish. Assuming that we have a list of *neighbors* for each particle i that lie within a radius of $\hbar$ around i, the algorithmic complexity reduces to $\mathcal{O}(mn)$, where m is the maximum number of neighboring particles. In practice, m is usually bounded by a constant such that we can expect linear runtime complexity, *i.e.*, $\mathcal{O}(n)$.

The problem of finding the neighbor list is commonly referred to

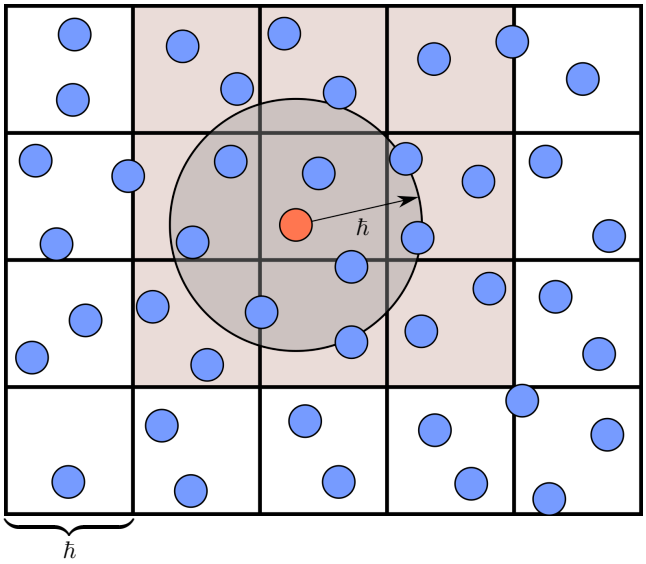


Figure 6: Uniform grid approach to efficiently find neighboring particles. Particles that are closer to the orange particle than the kernel support radius $\hbar$ must lie within the one ring of the cell containing the orange particle if the grid size is $\geq \hbar$.

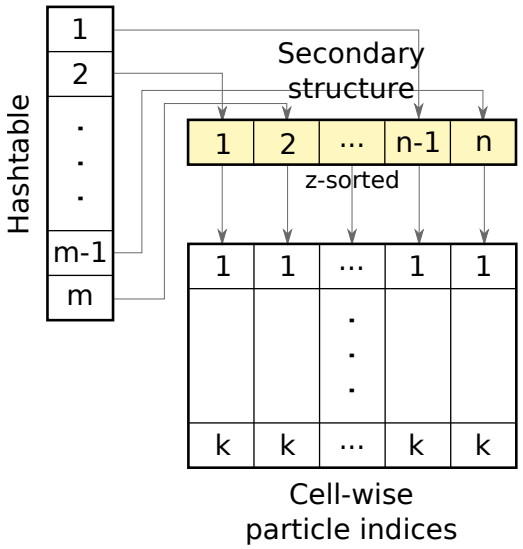


Figure 7: Hash table of size m pointing to secondary data structure (of n non-empty cells) to realize compact hashing. Each array to store the particle index lists in the secondary structure preallocates k entries to minimize the number of memory allocations in the course of the simulation.

as the *fixed-radius near neighbor problem* and is widely addressed in the computational geometry literature. The naïve approach, *i.e.*, brute-force, has a computational complexity of $\mathcal{O}(n^2)$ and is therefore not optimal. In this section, we will present an algorithm to approach the problem in a computationally more efficient way, *i.e.*, compact hashing [IABT11]. The basic idea of the approach is to place a uniform grid over the domain spanned by the particles with a grid cell size equal to the kernel support radius $\hbar$. Assuming that a particle is located in the grid cell represented by the tuple $\mathbf{c} = (i, j, k)$, where i, j, and k denote row, column, and depth column of the cell in the grid. Then, it is obvious that we only have to query for potentially neighboring particles in the cell c itself and its one-ring, *i.e.*, $(i-1, j-1, k-1)$, $(i-1, j-1, k)$, $(i-1, j-1, k+1)$, $\ldots$, $(i+1, j+1, k+1)$. The strategy then results in an algorithm with a computational complexity of $\mathcal{O}(n)$ for construction and $\mathcal{O}(1)$ to find the neighbors of a single particle implying $\mathcal{O}(n)$ to find the set of all the neighbors of all of the particles. Obviously, the grid-based approach can easily be generalized to higher dimensions. Please see Fig. 6 for a graphical illustration.

3.1. Compact Hashing

As discussed before, the uniform grid based approach results in a good computational runtime complexity. However, there is still potential for optimizations in terms of memory consumption, cache efficiency, and parallel processing. In this regard, the concept of *compact hashing* was proposed by Ihmsen et al. [IABT11] and will be explained in the following. An open source C++ implementation of a variant of this approach can be found online [Kos19].

A particular disadvantage of spatial grids is that memory for all cells in the grid has to be allocated although only a small number of cells might be occupied by particles. Due to the curse of dimensionality the memory requirements increase quickly with increasing domain size. It would be more memory efficient to only store the populated cells and, hence, employ a sparse representation of the grid. Therefore, Ihmsen et al. suggest to store the grid cells in a hash map by hashing the index tuple $\mathbf{c} = (i, j, k)$ to a scalar index following [THM*03]:

$$\text{hash}(\mathbf{c}) = [(p_1\, i) \text{ XOR } (p_2\, j) \text{ XOR } (p_3\, k)] \bmod m, \tag{34}$$

where $p_1 = 73856093$, $p_2 = 19349663$, and $p_3 = 83492791$ are large prime numbers and where m is the hash table size. Please note, that it generally cannot be avoided that several spatial cells are mapped to the same hash value (hash collision). The effect of overpopulated entries in the hash table might lead to a slow-down of the neighborhood query. However, as suggested by Teschner et al. [THM*03] the number of hash collisions can be reduced by increasing the hash table size, *i.e.*, trading memory for speed. As noted by Ihmsen et al. the hash table is usually sparsely filled when used in conjuction with SPH discretizations. Therefore, we would like to avoid to unnecessarily preallocate a large amount of memory. Moreover, the cache-hit rate of this approach can not expected to be optimal as the cells that are spatially close are not necessarily close in memory.

In order to reduce the frequency of allocations, Ihmsen et al. suggest to only store a handle per hash table entry that points to a secondary data structure – a contiguous array of the populated cells (see Fig. 7). Each item of the secondary structure stores a list of the particle indices contained in the respective cell. In this way memory for a used cell is only allocated if it contains particles and the memory can be (optionally) deallocated if the cell gets empty. Each storage for the index arrays in the secondary data structure can be further preallocated with the maximally expected number

of particles in a cell. To summarize, the memory consumption now scales linear with the number of particles and not with the volume of the simulation domain.

As it lies in the nature of spatial hash tables to scatter data according to spatially close cells, the indirection to the secondary data structure allows us to optimize for spatial locality in memory. To realize this, Ihmsen et al. suggest to sort the non-empty cells according to a space-filling Z-curve. The cache hit-rate can further be optimized by analogously sorting the per-particle data in the same way. However, performing the actual sort ($\mathcal{O}(n \log n)$) causes computational overhead and since the particles are constantly moving throughout space during the simulation, it is advised to update the Z-sort in fixed intervals, *e.g.*, after every 1000^{th} time step. This is justified as the order is expected to be roughly maintained over a small number of time steps due to temporal coherence.

Finally, several operations such as the hash table construction, updates and neighborhood queries can be (partially) parallelized to further optimize performance. For further details on the approach, we would like to refer the reader to the according original paper [IABT11].

4. Pressure Solvers

Incompressibility is an essential aspect in realistic fluid simulations. The fluid volume should not noticeably oscillate or generally grow or shrink over time. Fluid solvers preserve the fluid volume by computing a pressure acceleration $-\frac{1}{\rho}\nabla p$ where the pressure p is proportional to the volume deviation. Then, the term $-\frac{1}{\rho}\nabla p$ accelerates particles from high pressure, *i.e.*, regions with large volume deviations, to low pressure, *i.e.*, regions with small volume deviations. If there would be no volume deviation everywhere in the fluid, the pressure would be zero and the pressure gradient and the pressure acceleration would also be zero.

Solver implementations typically distinguish pressure acceleration $\mathbf{a}^{\text{p}} = -\frac{1}{\rho}\nabla p$ and all other non-pressure accelerations $\mathbf{a}^{\text{nonp}}$ which improves the intuition of the incompressibility concept. First, a predicted velocity is computed with, *e.g.*, $\mathbf{v}^* = \mathbf{v}(t) + \Delta t \mathbf{a}^{\text{nonp}}(t)$. Then, pressure is computed from the volume deviation after advecting the fluid with $\mathbf{v}^*$. Finally, the respective pressure acceleration would be applied as, *e.g.*, $\mathbf{v}(t+\Delta t) = \mathbf{v}^* + \Delta t \mathbf{a}^{\text{p}}(t)$ to minimize the volume deviation. This final velocity update is often referred to as pressure projection which is related to the fact that the velocity change $\Delta t \mathbf{a}^{\text{p}}(t)$ should be minimal. *I.e.*, the pressure acceleration should change the velocity field as little as possible.

Conceptually, pressure is proportional to the volume deviation. However, there exist various alternatives to actually compute the pressure. First, the volume deviation can be explicitly computed from the density or the velocity divergence can be used to compute a differential update of the volume deviation. Second, pressure can be computed locally with a state equation or it can be computed globally by solving a Pressure Poisson Equation (PPE). The first aspect determines whether the fluid volume oscillates or continuously changes, while the second aspect influences the solver performance.

4.1. Explicit Volume Deviation

The volume deviation is typically deduced from the density deviation. Although SPH solvers can easily handle both formulations, it is probably due to historical reasons that the density formulation is preferred over the volume formulation. The SPH density at a particle i is computed with $\rho_i = \sum_j m_j W_{ij}$ and the deviation to the rest density ρ^0 is considered for the pressure computation. Note that the density deviation is often clamped, *e.g.*, $\max(\rho_i - \rho^0, 0)$ or $\max\left(\frac{\rho_i}{\rho^0} - 1, 0\right)$, as a simple solution to the particle deficiency problem at the free surface (see Fig. 5).

4.2. Differential Volume Deviation

The continuity equation relates the time derivative of the density to the velocity divergence: $\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}$. This fact can be used to predict a particle density from its previous density and, *e.g.*, the predicted velocity: $\rho_i^* = \rho_i(t) - \Delta t \rho_i(t) \nabla \cdot \mathbf{v}_i^*$. Here, ρ_i^* is a prediction of the particle density after advecting the particles with $\mathbf{v}_i^*$ for time Δt. If it is assumed that the current density equals the rest density, the predicted density is computed as $\rho_i^* = \rho^0 - \Delta t \rho^0 \nabla \cdot \mathbf{v}_i^*$ which means that $\rho_i^* - \rho^0 = -\Delta t \rho^0 \nabla \cdot \mathbf{v}_i^*$ can be used as a measure for the density deviation. It can be seen that minimizing the density deviation is related to minimizing the velocity divergence. The term $-\Delta t \rho^0 \nabla \cdot \mathbf{v}_i^*$ is a density change at a particle if the particles are advected with $\mathbf{v}_i^*$ for time Δt.

4.3. Discussion – Explicit vs. Differential Volume Deviation

Both forms imply challenges. If pressure accelerations are derived from the explicit form of the volume deviation, the fluid volume oscillates due to an over-correction of the pressure acceleration. These oscillations have to minimized. At least, they should not be perceivable. Using the differential form to compute the volume deviation results in a drift of the fluid volume, typically a volume loss. The differential form assumes that the current density is correct. It minimizes density changes between simulation steps, but potentially existing density deviations are not detected or corrected. Here, the challenge is to minimize the volume drift. Although volume drift often occurs in Eulerian pressure solvers and volume oscillations often occur in Lagrangian solvers, both issues are not related to the Eulerian or Lagrangian perspective. If an SPH solver was using the differential form to compute the density deviation, it would suffer from volume drift. If a Eulerian solver, *e.g.*, FLIP, was using the explicit form for the density computation, it would suffer from oscillations.

4.4. State Equation SPH (SESPH)

State equations are used to compute pressure from density deviations. The density deviation can be computed explicitly or from a differential form. The deviation can be represented as a quotient or a difference of actual and rest density. One or more stiffness constants are involved. Some examples are: $p_i = k\left(\frac{\rho_i}{\rho^0} - 1\right)$, $p_i = k(\rho_i - \rho^0)$ or $p_i = k_1\left(\left(\frac{\rho_i}{\rho^0}\right)^{k_2} - 1\right)$. As $\rho_i < \rho^0$ is not considered to solve the particle deficiency problem at the free surface, the

computed pressure is always non-negative. SPH fluid simulations that use a state equation to compute pressure are often referred to as compressible or weakly compressible. In contrast, fluid implementations that solve a PPE to compute pressure are known as incompressible. These terms basically indicate that it is more challenging to minimize compression with a state equation than with a PPE.

It might look confusing that arbitrary pressure values can be computed for a given density $\rho_i > \rho^0$ dependent on the state equation and the stiffness constant(s). Here, it is interesting to note that the parameters do not govern the pressure, but the compressibility of the SPH fluid. This can be seen in a simple example with a fluid at rest under gravity. In this case, the pressure acceleration at all particles cancels gravity, *i.e.*, $\mathbf{g} - \frac{1}{\rho_i}\nabla p_i = \mathbf{0}$. Discretizing the pressure gradient with SPH yields $\mathbf{g} = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij}$. Using, *e.g.*, $p_i = k(\rho_i - \rho^0)$, yields $\mathbf{g} = k\sum_j m_j \left(\frac{\rho_i - \rho^0}{\rho_i^2} + \frac{\rho_i - \rho^0}{\rho_j^2}\right)\nabla W_{ij}$. It can be seen that a variation of the stiffness constant k is related to a variation in the density deviation $\rho_i - \rho^0$. This relation, however, is not simply $k(\rho_i - \rho^0) = \text{const}$ since the erroneous particle sampling for $\rho_i \neq \rho^0$ influences the SPH discretization of the pressure gradient. But generally, the stiffness constant in the state equation governs the density deviation. Larger values result in smaller deviations and require smaller time steps. Smaller values lead to larger density deviations, *i.e.*, less realistic simulations. Also, the boundary handling fails if the tolerated density deviation is too large.

4.5. Pressure Poisson Equation (PPE)

The general idea of the pressure computation is to end up with pressure accelerations that cause velocity changes that in turn cause displacements such that all particles are uncompressed, *i.e.*, have their rest density or rest volume. PPE solvers or projection schemes solve a linear system to compute the respective pressure field.

4.5.1. Derivation

We consider the predicted velocity after all non-pressure accelerations: $\mathbf{v}^* = \mathbf{v}(t) + \Delta t \mathbf{a}^{\text{nonp}}(t)$. If the particles would be advected with this velocity, we can use the continuity equation to estimate a predicted density $\rho^* = \rho(t) - \Delta t \rho(t) \nabla \cdot \mathbf{v}^*$. Now, the goal of the pressure computation is a pressure acceleration $-\frac{1}{\rho(t)}\nabla p(t)$ that corresponds to a velocity change $-\Delta t \frac{1}{\rho(t)}\nabla p(t)$ whose divergence $-\nabla \cdot \left(\Delta t \frac{1}{\rho(t)}\nabla p(t)\right)$ corresponds to a density change per time $-\rho(t)\nabla \cdot \left(\Delta t \frac{1}{\rho(t)}\nabla p(t)\right)$ that cancels the predicted density deviation per time $\frac{\rho^0 - \rho^*}{\Delta t}$, *i.e.*, $\frac{\rho^0 - \rho^*}{\Delta t} - \rho(t)\nabla \cdot \left(\Delta t \frac{1}{\rho(t)}\nabla p(t)\right) = 0$. This is one form of a PPE, typically written as

$$\Delta t \nabla^2 p(t) = \frac{\rho^0 - \rho^*}{\Delta t}. \tag{35}$$

Note that $\nabla \cdot \nabla p(t) = \nabla^2 p(t)$. In this equation, the pressure p is unknown. We have one equation per particle, resulting in a system with n equations and n unknown pressure values for n particles. Various similar PPE forms can be derived more formally, *e.g.*, starting with the continuity equation at time $t + \Delta t$: $-\rho(t+\Delta t)\nabla \cdot \mathbf{v}(t+\Delta t) = \frac{D\rho(t+\Delta t)}{Dt}$. The time derivative of the density is approximated with $\frac{D\rho(t+\Delta t)}{Dt} = \frac{\rho(t+\Delta t) - \rho(t)}{\Delta t}$. The velocity is written as $\mathbf{v}(t+\Delta t) = \mathbf{v}^* - \Delta t \frac{1}{\rho(t+\Delta t)}\nabla p(t+\Delta t)$ using an implicit update with the pressure acceleration at time $t + \Delta t$. Imposing the constraint $\rho(t+\Delta t) = \rho^0$, we get $-\rho^0 \nabla \cdot \mathbf{v}^* + \nabla \cdot (\Delta t \nabla p(t+\Delta t)) = \frac{\rho^0 - \rho(t)}{\Delta t}$ and finally

$$\Delta t \nabla^2 p(t+\Delta t) = \frac{\rho^0 - (\rho(t) - \Delta t \rho^0 \nabla \cdot \mathbf{v}^*)}{\Delta t}. \tag{36}$$

If compared carefully, there are very minor differences between Eqs. (35) and (36) in the computation of the predicted density and the pressure acceleration. As $\rho(t) \approx \rho^0$, however, these differences are negligible. The biggest difference would actually be the SPH discretizations of the Laplacian. Eq. (35) works with the particle neighborhood at time t, while Eq. (36) requires the neighborhood at time $t + \Delta t$. As this would require an additional neighbor search per simulation step, it is generally ignored.

Eqs. (35) and (36) make use of the density invariance as source term in the PPE. Motivated by the continuity equation, the divergence of the predicted velocity could be used alternatively. To derive the respective form, we start with $\Delta t \frac{1}{\rho(t)}\nabla p(t) = \mathbf{v}^* - \mathbf{v}(t+\Delta t)$. Taking the divergence and imposing the constraint that the velocity field at $t + \Delta t$ should be divergence free, *i.e.*, $\nabla \cdot \mathbf{v}(t+\Delta t) = 0$, we get $\nabla \cdot \left(\Delta t \frac{1}{\rho(t)}\nabla p(t)\right) = \nabla \cdot \mathbf{v}^*$ and

$$\Delta t \nabla^2 p(t) = \rho(t) \nabla \cdot \mathbf{v}^*. \tag{37}$$

We search for pressure p such that the pressure acceleration $-\frac{1}{\rho(t)}\nabla p(t)$ corresponds to a velocity change $-\Delta t \frac{1}{\rho(t)}\nabla p(t)$ whose divergence $-\nabla \cdot \left(\Delta t \frac{1}{\rho(t)}\nabla p(t)\right)$ cancels the divergence of the predicted velocity, *i.e.*, $-\nabla \cdot \left(\Delta t \frac{1}{\rho(t)}\nabla p(t)\right) + \nabla \cdot \mathbf{v}^* = 0$.

SPH fluid solvers can easily employ any of the PPE forms. If the density invariance is taken as source term, density oscillations occur and have to be minimized. If the predicted velocity divergence is taken as source term, the fluid volume tends to drift. Volume or density oscillations are not a general SPH issue, but related to the pressure computation. In the same way, volume drift is not, *e.g.*, a FLIP issue, but only due to the typically used velocity divergence as source term in the PPE.

4.6. Discretization with Implicit Incompressible SPH (IISPH)

There exist various alternative discretizations with benefits and drawbacks. Here, we discuss one option referred to as IISPH which has been proposed in [ICS*14]. We consider a slightly rewritten form of the PPE in Eq. (35) for particle i:

$$\Delta t^2 \nabla^2 p_i = \rho^0 - \rho_i^*. \tag{38}$$

If a quantity is considered at time t, *e.g.*, pressure, the time index is omitted. Using SPH, the source term is computed as

$$\rho^0 - \rho_i^* = \rho^0 - \rho_i - \Delta t \sum_j m_j \left(\mathbf{v}_i^* - \mathbf{v}_j^*\right) \cdot \nabla W_{ij} \tag{39}$$

with $\mathbf{v}_i^* = \mathbf{v}_i + \Delta t \mathbf{a}_i^{\text{nonp}}$. The computation of the Laplacian is realized as the computation of the divergence of the velocity change

due to the pressure acceleration, *i.e.*,

$$\begin{aligned}\Delta t^2 \nabla^2 p_i &= -\Delta t \rho_i \nabla \cdot \left(\Delta t \mathbf{a}_i^{\mathrm{p}}\right) \\ &= \Delta t^2 \sum_j m_j \left(\mathbf{a}_i^{\mathrm{p}} - \mathbf{a}_j^{\mathrm{p}}\right) \cdot \nabla W_{ij} \end{aligned} \tag{40}$$

with pressure acceleration

$$\mathbf{a}_i^{\mathrm{p}} = -\frac{1}{\rho_i} \nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}. \tag{41}$$

Using Eqs. (39) and (40), we can compute the left-hand and right-hand side of Eq. (38) at each particle using SPH sums over adjacent particles. The IISPH discretization of Eq. (38) is

$$\Delta t^2 \sum_j m_j \left(\mathbf{a}_i^{\mathrm{p}} - \mathbf{a}_j^{\mathrm{p}}\right) \cdot \nabla W_{ij} = \rho^0 - \rho_i^*. \tag{42}$$

Introducing the velocity change due to pressure acceleration $\mathbf{v}_i^{\mathrm{p}} = \Delta t \mathbf{a}_i^{\mathrm{p}}$, Eq. (42) can be written as

$$\Delta t \sum_j m_j \left(\mathbf{v}_i^{\mathrm{p}} - \mathbf{v}_j^{\mathrm{p}}\right) \cdot \nabla W_{ij} = \Delta t \rho_i \, \nabla \cdot \mathbf{v}_i^{\mathrm{p}} = \rho^0 - \rho_i^*. \tag{43}$$

It can be seen that we search for pressure values such that the pressure acceleration causes a velocity change $\mathbf{v}_i^{\mathrm{p}}$ such that the divergence of $\mathbf{v}_i^{\mathrm{p}}$ causes a density change that corrects from the predicted density ρ_i^* to the rest density ρ^0, *i.e.*, $\rho_i^* + \Delta t \rho_i \, \nabla \cdot \mathbf{v}_i^{\mathrm{p}} = \rho^0$.

4.6.1. IISPH Solver

System Eq. (42) is considered at all particles. We have n equations with n unknown pressure values p_i. Each equation does not only contain an unknown pressure value p_i, but also unknown pressure values at neighbors of i and - due to Eq. (41) - also unknown pressures at neighbors of neighbors of i. The set of Eq. (42) at all particles forms a system $\mathbf{Ap} = \mathbf{s}$ and Eq. (42) at particle i can be written as $(\mathbf{Ap})_i = s_i$:

$$(\mathbf{Ap})_i = \Delta t^2 \sum_j m_j \left(\mathbf{a}_i^{\mathrm{p}} - \mathbf{a}_j^{\mathrm{p}}\right) \cdot \nabla W_{ij} \tag{44}$$

$$s_i = \rho^0 - \rho_i^*. \tag{45}$$

$\mathbf{A}$ is not a diagonal matrix, but at least sparse. In row i, matrix elements a_{ii} and also elements a_{ij} are non-zero with j being a neighbor or a neighbor of a neighbor of i.

Extracting the matrix elements a_{ij} would not be impossible, but rather tedious and error-prone. Fortunately, element extraction is not required to solve the system as solver implementations typically just require the computation of $(\mathbf{Ap})_i$. The term $(\mathbf{Ap})_i$ can be computed by first evaluating $\mathbf{a}_i^{\mathrm{p}}$ with Eq. (41), followed by the computation of $\Delta t^2 \nabla^2 p_i$ using Eq. (40). An explicit notion of the elements of $\mathbf{A}$ is typically not required as, *e.g.*, for Conjugate Gradients. Some solvers, *e.g.*, Jacobi, require the diagonal elements a_{ii}.

Relaxed Jacobi scheme In the following, we discuss the implementation of a Jacobi variant to solve $\mathbf{Ap} = \mathbf{s}$. The method starts with an initialization of the pressure vector, *e.g.*, $\mathbf{p}^{(0)} = \mathbf{0}$. Then, the weighted / damped / relaxed Jacobi scheme iteratively updates all pressure values using

$$p_i^{(l+1)} = (1-\omega) p_i^{(l)} + \frac{\omega}{a_{ii}} \left(s_i - \sum_{j \neq i} a_{ij} p_j^{(l)}\right) \tag{46}$$

with l indicating the iteration and ω being a relaxation coefficient. The relaxation coefficient is typically set to $\omega = 0.5$ in IISPH implementations. Smaller values reduce the convergence, larger values are unstable. The update in Eq. (46) seems to indicate that the diagonal elements a_{ii}, but also elements a_{ij} for neighboring particles are required in an implementation. Interestingly, the update in Eq. (46) can be rewritten as

$$\begin{aligned} p_i^{(l+1)} &= (1-\omega) p_i^{(l)} + \frac{\omega}{a_{ii}} \left(s_i - (\mathbf{Ap}^{(l)})_i + a_{ii} p_i^{(l)}\right) \\ &= p_i^{(l)} + \frac{\omega}{a_{ii}} \left(s_i - (\mathbf{Ap}^{(l)})_i\right). \end{aligned} \tag{47}$$

This update requires the computation of the term $(\mathbf{Ap}^{(l)})_i$ which can be computed from Eqs. (40) and (41) without notion of the elements a_{ij}. As IISPH only considers non-negative pressure, the actual update is

$$p_i^{(l+1)} = \max\left(p_i^{(l)} + \frac{\omega}{a_{ii}} \left(s_i - (\mathbf{Ap}^{(l)})_i\right), 0\right). \tag{48}$$

Pressure clamping The clamping has been proposed in [ICS*14] where simulation artifacts are discussed in case of negative pressure values. Negative pressure values are sometimes briefly discussed, but rarely carefully analyzed. Theoretically, arbitrary constant offsets to all pressure values do not change the pressure gradient. Shifting a pressure range from $[0, p_{\max}]$ to, *e.g.*, $[-100, p_{\max} - 100]$ or $[100, p_{\max} + 100]$ by adding constant offsets to all values does not change the pressure gradient. SPH discretizations, however, behave differently for negative and positive pressure values. Reasons have not been investigated yet, but one could speculate that negative and positive pressures result in different pressure gradients in case of incomplete neighborhoods where missing contributions are implicitly assumed to have zero pressure. In IISPH simulations, minimum pressure is generally zero being consistent with the implicit assumption of zero pressure for missing samples at free surfaces.

Diagonal element The update in Eq. (48) requires the diagonal element a_{ii}. The element can be calculated by accumulating all coefficients of p_i after inserting Eq. (41) into Eq. (40). Here, it is important to keep in mind that i is one of the neighbors of each neighbor of i. Finally, the diagonal element is

$$\begin{aligned} a_{ii} = &-\Delta t^2 \sum_j m_j \left(\sum_j \frac{m_j}{\rho_j^2} \nabla W_{ij}\right) \cdot \nabla W_{ij} \\ &-\Delta t^2 \sum_j m_j \left(\frac{m_i}{\rho_i^2} \nabla W_{ij}\right) \cdot \nabla W_{ij}. \end{aligned} \tag{49}$$

Stop criterion There is no agreement on when to stop the Jacobi iterations in Eq. (48). The iterations could be stopped after a fixed number. Alternatively, a predicted density deviation is often considered. This is motivated by the fact that $(\mathbf{Ap}^{(l)})_i$ is a predicted density change at particle i due to the pressure field at iteration l. *I.e.*, $\rho_i^{\mathrm{err},*} = ((\mathbf{Ap}^{(l)})_i - s_i)/\rho^0 = ((\mathbf{Ap}^{(l)})_i + \rho_i^* - \rho^0)/\rho^0$ is a

predicted relative density error at particle i, if pressure accelerations according to the pressure field $\mathbf{p}^{(l)}$ would be applied. Typically, the average of all $\rho_i^{\text{err},*}$ is taken as a stop criterion, *e.g.*, $\rho_i^{\text{avg\_err},*} = \frac{1}{n}\sum_i |\rho_i^{\text{err},*}|$. In [ICS*14], it is proposed to stop if, *e.g.*, $\rho_i^{\text{avg\_err},*} < 0.1\%$, *i.e.*, the oscillation of the overall fluid volume is below 0.1%. In addition to the predicted average density error, the maximum of the predicted density errors could also be taken as a stop criterion.

Implementation Alg. 2 shows the implementation of IISPH. Source term s_i and diagonal element a_{ii} are computed once. In iteration l, the pressure Laplacian $(\mathbf{Ap}^{(l)})_i = \Delta t^2 \nabla^2 p_i$ is computed in two steps. First, the pressure acceleration $(\mathbf{a}_i^{\text{p}})^{(l)}$ is computed and stored at the particles. Then, the divergence of the velocity change due to the pressure acceleration, *i.e.*, $(\mathbf{Ap}^{(l)})_i$, is computed. Neighborhood search, the computation of the predicted velocity $\mathbf{v}_i^*$ and the advection of particles are omitted in Alg. 2.

for all *particle i* **do**
  compute diagonal element a_{ii} with Eq. (49)
  compute source term s_i with Eq. (39)
  initialize pressure $p_i^{(0)} = 0$
$l = 0$
repeat
  for all *particle i* **do**
    compute pressure acceleration $(\mathbf{a}_i^{\text{p}})^{(l)}$ with Eq. (41)
  for all *particle i* **do**
    compute Laplacian $(\mathbf{Ap}^{(l)})_i$ with Eq. (40)
    update pressure $p_i^{(l+1)}$ with Eq. (48)
  $l = l + 1$
until $\rho_i^{\text{avg\_err},*} < 0.1\%$

Algorithm 2: Pressure computation with the IISPH PPE solver.

4.7. Predictive–Corrective Incompressible SPH (PCISPH)

In the following a pressure solver based on a predictor-corrector approach is introduced [SP09].

Motivation The density $\rho_i(t+\Delta t)$ can be estimated with

$$\begin{aligned}\rho_i(t+\Delta t) = &\sum_j m_j W_{ij} + \Delta t \sum_j m_j (\mathbf{v}_i - \mathbf{v}_j)\cdot\nabla W_{ij} \\ &+ \Delta t \sum_j m_j (\Delta t \mathbf{a}_i^{\text{nonp}} - \Delta t \mathbf{a}_j^{\text{nonp}})\cdot\nabla W_{ij} \\ &+ \Delta t \sum_j m_j (\Delta t \mathbf{a}_i^{\text{p}} - \Delta t \mathbf{a}_j^{\text{p}})\cdot\nabla W_{ij}. \end{aligned} \tag{50}$$

Again, the time index is omitted for quantities at time t. Using the predicted density

$$\begin{aligned}\rho_i^* = &\sum_j m_j W_{ij} + \Delta t \sum_j m_j (\mathbf{v}_i - \mathbf{v}_j)\cdot\nabla W_{ij} \\ &+ \Delta t \sum_j m_j (\Delta t \mathbf{a}_i^{\text{nonp}} - \Delta t \mathbf{a}_j^{\text{nonp}})\cdot\nabla W_{ij} \end{aligned} \tag{51}$$

and the constraint $\rho_i(t+\Delta t) = \rho^0$, we can write

$$\rho^0 = \rho_i^* + \Delta t \sum_j m_j (\Delta t \mathbf{a}_i^{\text{p}} - \Delta t \mathbf{a}_j^{\text{p}})\cdot\nabla W_{ij}. \tag{52}$$

When using the symmetric SPH formulation for the pressure acceleration, the terms $\mathbf{a}_i^{\text{p}}$ and $\mathbf{a}_j^{\text{p}}$ are computed from unknown pressure values at particle i, at neighbors of i and at neighbors of neighbors of i. Now, PCISPH introduces approximations and simplifications to end up with only one unknown pressure value p_i in Eq. (52) [SP09]. Then, each pressure value p_i can be computed from one equation. Solving a linear system is avoided.

Simplifications The pressure acceleration is discretized with the symmetric formulation. It is assumed that the pressure p_j at neighboring particles equals the pressure p_i at particle i. Further, $m_j = m_i$ for all neighbors and $\rho_i = \rho_j \approx \rho^0$. Then, the pressure acceleration can be written as

$$\mathbf{a}_i^{\text{p}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} \tag{53}$$

$$\approx -m_i \frac{2p_i}{(\rho^0)^2}\sum_j \nabla W_{ij}. \tag{54}$$

Using this approximation, Eq. (52) can be written as

$$\rho^0 = \rho_i^* + 2\Delta t^2 \frac{m_i^2}{(\rho^0)^2}\sum_j \left(-p_i \sum_j \nabla W_{ij} + p_j \sum_k \nabla W_{jk}\right)\cdot\nabla W_{ij}. \tag{55}$$

Using $p_i \approx p_j$ and approximating $\sum_k \nabla W_{jk} \approx \nabla W_{ji}$, the equation can further be simplified to

$$\begin{aligned}\rho^0 &= \rho_i^* + p_i \frac{2\Delta t^2 m_i^2}{(\rho^0)^2}\sum_j \left(-\sum_j \nabla W_{ij} + \nabla W_{ji}\right)\cdot\nabla W_{ij} \\ &= \rho_i^* - p_i \underbrace{\frac{2\Delta t^2 m_i^2}{(\rho^0)^2}\left(\sum_j \nabla W_{ij}\cdot\sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij}\cdot\nabla W_{ij})\right)}_{-\frac{1}{k^{\text{PCI}}}}. \end{aligned} \tag{56}$$

In PCISPH, the coefficient k^{PCI} is considered for a template particle with perfect sampling, *i.e.*, $\sum_j \nabla W_{ij}\cdot\sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij}\cdot\nabla W_{ij}) = \text{const}$. Finally, we have the following equation per particle:

$$p_i = k^{\text{PCI}}(\rho^0 - \rho_i^*) \tag{57}$$

with

$$k^{\text{PCI}} = -\frac{0.5(\rho^0)^2}{\Delta t^2 m_i^2}\cdot\frac{1}{\sum_j \nabla W_{ij}\cdot\sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij}\cdot\nabla W_{ij})} \tag{58}$$

for a template particle i with perfect sampling. This is a state equation, where the stiffness constant is not user-defined, but motivated by the fact that Eq. (50) should be satisfied, *i.e.*, the pressure should induce pressure accelerations such that the particles have their rest density at time $t+\Delta t$.

Iterative refinement The locally optimized state equation is one important property of the PCISPH concept. A second significant characteristics is the iterative refinement of the pressure field. While this sounds expensive, it is motivated by large time steps compared to simple state-equation solvers. PCISPH computes a first estimate of the pressure $p_i^{(1)}$ with Eq. (57). This predicted pressure field is used to compute $(\mathbf{a}_i^{\mathrm{p}})^{(1)}$ with Eq. (53). Then, the pressure field is iteratively refined by

$$p_i^{(l+1)} = p_i^{(l)} + k^{\mathrm{PCI}}(\rho^0 - \rho_i^* - \Delta t \sum_j m_j(\Delta t(\mathbf{a}_i^{\mathrm{p}})^{(l)} - \Delta t(\mathbf{a}_j^{\mathrm{p}})^{(l)}) \cdot \nabla W_{ij}) \tag{59}$$

in iteration l. The term

$$(\rho_i^{\mathrm{p}})^{(l)} = \Delta t \sum_j m_j(\Delta t(\mathbf{a}_i^{\mathrm{p}})^{(l)} - \Delta t(\mathbf{a}_j^{\mathrm{p}})^{(l)}) \cdot \nabla W_{ij} \tag{60}$$

is one option to predict the density change due to the pressure accelerations. If this density change does not cancel the predicted density deviation $\rho^0 - \rho_i^*$, the predicted pressure is corrected. Similarly to IISPH, the process is stopped if $(\rho^0 - \rho_i^* + (\rho_i^{\mathrm{p}})^{(l)})/\rho^0$ is sufficiently small, *e.g.*, smaller than 0.1%. The original PCISPH depiction proposes to compute ρ_i^{p} from particle displacements due to the pressure accelerations, *i.e.*,

$$(\rho_i^{\mathrm{p}})^{(l)} = m_i \left((\Delta \mathbf{x}_i)^{(l)} \cdot \sum_j \nabla W_{ij} - \sum_j \nabla W_{ij} \cdot (\Delta \mathbf{x}_j)^{(l)} \right). \tag{61}$$

If $(\Delta \mathbf{x}_i)^{(l)} = \Delta t^2 (\mathbf{a}_i^{\mathrm{p}})^{(l)}$ and $m_i = m_j$, Eqs. (60) and (61) are identical. We prefer Eq. (60) as it will also be used in the discussion of the relation between PCISPH and IISPH.

Implementation Alg. 3 shows the implementation of PCISPH. The stiffness constant k^{PCI} is computed once at the beginning of the simulation. The same coefficient is used for all particles. The pressure field is predicted with the state equation in Eq. (57). The effect of the respective pressure acceleration onto the density is estimated. Remaining deviations from the rest density are used to compute pressure corrections with Eq. (59). Neighbor search and the advection of the particles are omitted in Alg. 3.

4.8. Relations between SESPH, IISPH and PCISPH

The PCISPH pressure solver updates the pressure with

$$p_i^{(l+1)} = p_i^{(l)} + k^{\mathrm{PCI}}(\rho^0 - \rho_i^* - \Delta t \sum_j m_j(\Delta t(\mathbf{a}_i^{\mathrm{p}})^{(l)} - \Delta t(\mathbf{a}_j^{\mathrm{p}})^{(l)}) \cdot \nabla W_{ij}) \tag{62}$$

which simplifies to the state equation

$$p_i^{(1)} = k^{\mathrm{PCI}}(\rho^0 - \rho_i^*) \tag{63}$$

for the first update if the pressure is initialized with $p_i^{(0)} = 0$. The same applies to IISPH. The solver updates pressure with

$$p_i^{(l+1)} = p_i^{(l)} + \frac{\omega}{a_{ii}} \left(s_i - (\mathbf{A}\mathbf{p}^{(l)})_i \right) \tag{64}$$

compute stiffness constant k^{PCI} with Eq. (58)
for all *particle i* **do**
  compute predicted density ρ_i^* with Eq. (51)
  initialize pressure $p_i^{(1)}$ with Eq. (57)
$l = 1$
repeat
  for all *particle i* **do**
    compute pressure acceleration $(\mathbf{a}_i^{\mathrm{p}})^{(l)}$ with Eq. (53)
  for all *particle i* **do**
    compute density change $(\rho_i^{\mathrm{p}})^{(l)}$ with Eq. (60)
    update pressure $p_i^{(l+1)}$ with Eq. (59)
  $l = l + 1$
until $\rho_i^{\mathrm{avg\_err},*} < 0.1\%$

Algorithm 3: Pressure computation with the PCISPH solver.

which simplifies to the state equation

$$p_i^{(1)} = \frac{\omega}{a_{ii}} s_i = \frac{\omega}{a_{ii}} (\rho^0 - \rho_i^*) \tag{65}$$

for the first update if the pressure is initialized with $p_i^{(0)} = 0$. *I.e.*, if IISPH or PCISPH stop after one pressure update, they are state-equation solvers.

Another remarkable aspect are the stiffness constants in PCISPH (Eqs. (62) and (63)) and IISPH (Eqs. (64) and (65)). The PCISPH stiffness constant is

$$k^{\mathrm{PCI}} = -\frac{0.5(\rho^0)^2}{\Delta t^2 m_i^2} \cdot \frac{1}{\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij} \cdot \nabla W_{ij})}. \tag{66}$$

The IISPH constant is $\frac{\omega}{a_{ii}}$ with $\omega = 0.5$ and

$$a_{ii} = -\Delta t^2 \sum_j m_j \left(\sum_j \frac{m_j}{\rho_j^2} \nabla W_{ij} \right) \cdot \nabla W_{ij} - \Delta t^2 \sum_j m_j \left(\frac{m_i}{\rho_i^2} \nabla W_{ij} \right) \cdot \nabla W_{ij}. \tag{67}$$

Applying the same assumptions as for PCISPH, *i.e.*, $\rho_i = \rho^0$ and $m_i = m_j$, the diagonal element simplifies to

$$a_{ii} = -\frac{\Delta t^2 m_i^2}{(\rho^0)^2} \left(\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}) \right). \tag{68}$$

Thus,

$$k^{\mathrm{PCI}} = \frac{\omega}{a_{ii}}. \tag{69}$$

Further,

$$\Delta t \sum_j m_j(\Delta t(\mathbf{a}_i^{\mathrm{p}})^{(l)} - \Delta t(\mathbf{a}_j^{\mathrm{p}})^{(l)}) \cdot \nabla W_{ij}) = (\mathbf{A}\mathbf{p}^{(l)})_i \tag{70}$$

which means that the pressure update with Eq. (62) in the PCISPH solver is equal to the pressure update with Eq. (64) in the IISPH solver. There might be insignificant differences in the computations

of ρ_i^* and $(\mathbf{Ap})_i$ between PCISPH and IISPH, but both solvers are essentially equal.

Ihmsen et al. [ICS*14] report significant performance differences between PCISPH and IISPH. These differences are possibly due to the fact that PCISPH computes one global stiffness constant k^{PCI} for a template particle, while IISPH computes a_{ii} for each particle. In particular, $\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} + \sum_j (\nabla W_{ij} \cdot \nabla W_{ij})$ is assumed to be constant in PCISPH, while it is computed per particle in IISPH. The respective difference affects the performance. If $k^{\text{PCI}} < \frac{\omega}{a_{ii}}$, the convergence of PCISPH is worse than with IISPH. If $k^{\text{PCI}} > \frac{\omega}{a_{ii}}$, PCISPH could be unstable. Such potential instabilities are difficult to deal with and might result in the requirement of significantly smaller time steps compared to IISPH. There are also other smaller differences between PCISPH and IISPH. *E.g.*, PCISPH computes ρ_i^* from advected samples without updated neighborhood, while IISPH uses the velocity divergence to estimate ρ_i^*. Such differences, however, are probably less relevant for stability or convergence differences.

4.9. PPE Variants

In addition to PCISPH, there exist various other pressure solvers that are closely related to a PPE solver, *e.g.*, Local Poisson SPH [HLL*12], Constraint Fluids (CF) [BLS12] and Position-based Fluids (PBF) [MM13]. It is beyond the scope of these course notes to provide a detailed analysis of the aforementioned solvers, but the close relation can be derived from three aspects. First, CF and PBF compute constraint values at particles: $C_i = \frac{\rho_i}{\rho^0} - 1$. The magnitudes of these constraints are equal to pressure values computed with a state equation $p_i = k\left(\frac{\rho_i}{\rho^0} - 1\right)$ with stiffness constant $k = 1$. Second, CF and PBF compute forces or position changes that are proportional to the negative of the constraint gradient. As constraint and pressure are closely related, constraint gradient and pressure gradient are related as well. Third, CF and PBF iteratively update their solution like a Jacobi solver.

In addition to these variants there exist publications that analyze the discretizations of the pressure Laplacian and the source term in the PPE. The general idea is always the same, *i.e.*, to compute pressure such that the resulting pressure accelerations minimize density deviations or the divergence of the velocity field. Nevertheless, the computed velocity field depends on the employed discretizations. *E.g.*, Fürstenau et al. [FAW17] compare three discretizations of the pressure Laplacian. They consider the following form of the PPE with velocity divergence as source term:

$$\nabla \cdot \left(\frac{1}{\rho_i}\nabla p_i\right) = \frac{\nabla \cdot \mathbf{v}_i^*}{\Delta t}. \tag{71}$$

Three variants to compute the pressure Laplacian are analyzed. The first variant is

$$\nabla \cdot \left(\frac{1}{\rho_i}\nabla p_i\right) = \frac{m_i}{\rho_i}\sum_j \frac{\rho_i + \rho_j}{\rho_i \rho_j}\frac{(p_j - p_i)(\mathbf{x}_i - \mathbf{x}_j)}{(\mathbf{x}_i - \mathbf{x}_j)^2 + \varepsilon} \cdot \nabla W_{ij}, \tag{72}$$

referred to as finite difference scheme, where ε is a small constant which is added to avoid singularities. The second variant is

$$\nabla \cdot \left(\frac{1}{\rho_i}\nabla p_i\right) = \frac{m_i}{\rho_i}\sum_j \left(\frac{\nabla p_j}{\rho_j} - \frac{\nabla p_i}{\rho_i}\right) \cdot \nabla W_{ij}, \tag{73}$$

referred to as double summation scheme. This approximation is used, *e.g.*, in IISPH [ICS*14]. It computes the divergence of the pressure accelerations. The third variant is

$$\nabla \cdot \left(\frac{1}{\rho_i}\nabla p_i\right) = \frac{m_i}{\rho_i}\sum_j \frac{\rho_i + \rho_j}{2\rho_i \rho_j}(p_j - p_i)\nabla^2 W_{ij}, \tag{74}$$

referred to as second derivative scheme. All these options can be used to compute the left-hand side of the PPE that contains the pressure Laplacian. If the PPE is solved with a Jacobi method, the required diagonal element a_{ii} varies accordingly.

The three discretizations result in different solutions for the velocity field. It is difficult to come up with general conclusions, as all discretizations have benefits and drawbacks. *E.g.*, the double-summation approach used in IISPH seems to have an improved solver convergence compared to the finite-difference scheme and the second-derivative scheme. On the other hand, the computed velocity field suffers from high-frequency noise. This noise, however, is rather low and it depends on the application whether this is an issue or not. In typical free-surface scenario, this noise is not an issue and the double-summation discretization is just faster than the other two options as less solver iterations are required for a specified tolerated density deviation.

Another degree-of-freedom is the form of the source term. As already shown and discussed in Section 4.5.1, the source term can either represent the divergence of the predicted velocity or the deviation of the predicted density to the rest density. The predicted velocity is the velocity after applying all non-pressure accelerations. The predicted density is the estimated density after advecting all samples with the predicted velocity. The equivalence of both formulations follows from the continuity equation. The PPE with density invariance as source term is

$$\Delta t^2 \nabla^2 p_i = \rho^0 - \rho_i^* \tag{75}$$

with source term

$$s_i = \rho^0 - \rho_i^*. \tag{76}$$

The PPE with velocity divergence as source term is

$$\Delta t^2 \nabla^2 p_i = \Delta t \rho_i \nabla \cdot \mathbf{v}_i^* \tag{77}$$

with source term

$$s_i = \Delta t \rho_i \nabla \cdot \mathbf{v}_i^*. \tag{78}$$

The properties of both variants have been analyzed in [CBG*18]. As already discussed, the density invariance results in an oscillating density deviation over time, while the velocity divergence causes a continuous drift of the density, typically a growing density over time. Another interesting aspect that is discussed in [CBG*18] is the artificial viscosity. Using the density invariance as source term causes more artificial viscosity than using the velocity divergence. Further, the velocity-divergence source term causes less high-frequency or short-range noise in the velocity field.

Considering all properties has an interesting conclusion. What about solving two PPEs, one with the density invariance as source term and one with the velocity divergence as source term? We would get two velocity changes due to pressure accelerations. According to the properties of the source terms, it can make sense to

advect the particles with the velocity from the PPE with density invariance as source term. This avoids the density drift. The velocity from the PPE with velocity divergence, however, could be used for the final velocities at the particles as these velocities have less artificial viscosity and less noise. This has been actually done by Bender and Koschier [BK15] who proposed to solve two PPEs with different source terms and to use one solution to compute the advected particle positions and one solution as the final velocity field. Solving two PPEs is obviously more expensive than a simple IISPH solver. However, each PPE solve typically requires very few iterations, *i.e.*, less than ten, and both PPEs share the same matrix which can be exploited to get an efficient solver as shown in the next section.

4.10. Divergence-Free SPH (DFSPH)

DFSPH [BK17] is a variant of the idea to solve two PPEs with different source terms. The original DFSPH solver does not compute pressure, but some stiffness parameter κ_i per particle i. According to the work of Band et al. [BGPT18], however, this stiffness parameter is closely related to pressure with $p_i = \kappa_i \rho_i$. In the following we introduce DFSPH using a pressure formulation and replace the stiffness parameter in order to get a formulation which is closer to the ones of PCISPH and IISPH. In this way it is easier to see the similarities and the differences.

DFSPH conceptually solves two PPEs, one with density invariance as source term, the other one with velocity divergence as source term. Instead of using one solution for the position update and one solution as the final velocity field, DFSPH combines both solutions to compute the final velocity field which is used to advect the particles. This combination is motivated by the fact that a first PPE solve with density invariance computes particle positions of an incompressible fluid state, but not necessarily a divergence-free velocity field. That's why, a second PPE solve with velocity divergence computes a divergence-free velocity field.

Divergence-Free Solver If we solve the PPE with velocity divergence as source term (see Eq. (77)), we can derive the following equation for the corresponding pressure value p_i^v of a particle i:

$$p_i^v = \frac{1}{\Delta t}\frac{D\rho_i}{Dt} \cdot \underbrace{\frac{\rho_i^2}{\|\sum_j m_j \nabla W_{ij}\|^2 + \sum_j \|m_j \nabla W_{ij}\|^2}}_{k_i^{\text{DFSPH}}}, \tag{79}$$

where the time derivative of the density is determined as

$$\frac{D\rho_i}{Dt} = \sum_j m_j (\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla W_{ij}. \tag{80}$$

Note that this formulation is similar to the pressure computation of PCISPH (see Eqs. (57) and (58)). But in contrast to PCISPH the source term is the velocity divergence and not the density deviation. Moreover, DFSPH does not use a global factor k^{PCI} that is determined for a template particle but computes the actual factor k_i^{DFSPH} for each particle i in each time step.

Algorithm 4 shows the divergence-free solver. In each iteration first the divergence is updated for all particles. Then the pressure values are determined and the predicted velocity is updated accordingly.

while $\left(\left(\frac{D\rho}{Dt}\right)^{\text{avg}} > \eta^{\text{div}}\right) \vee (\text{iter} < 1)$ **do**
  for all particles i **do**
    $\frac{D\rho_i}{Dt} = -\rho_i \nabla \cdot \mathbf{v}_i^*$
  for all particles i **do**
    $p_i^v = \frac{1}{\Delta t}\frac{D\rho_i}{Dt} k_i^{\text{DFSPH}}, \quad p_j^v = \frac{1}{\Delta t}\frac{D\rho_j}{Dt} k_j^{\text{DFSPH}}$
    $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{p_i^v}{\rho_i^2} + \frac{p_j^v}{\rho_j^2}\right) \nabla W_{ij}$

Algorithm 4: Divergence-free solver

Constant Density Solver The constant density solver uses the PPE with density deviation as source term (see Eq. (75)). For this PPE we get a pressure of

$$p_i = \frac{1}{\Delta t^2}(\rho_i^* - \rho_0) k_i^{\text{DFSPH}}, \tag{81}$$

where the predicted density is determined by

$$\rho_i^* = \rho_i + \Delta t \frac{D\rho_i}{Dt} = \rho_i + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \cdot \nabla W_{ij}. \tag{82}$$

Note that the factor k_i^{DFSPH} is used for both, the divergence-free and the constant density solver. Therefore, it has to be computed only once per simulation step. Algorithm 5 demonstrates an implementation of the constant density solver.

while $(\rho^{\text{avg}} - \rho_0 > \eta) \vee (\text{iter} < 2)$ **do**
  for all particles i **do**
    compute ρ_i^*
  for all particles i **do**
    $p_i = \frac{\rho_i^* - \rho_0}{\Delta t^2} k_i^{\text{DFSPH}}, \quad p_j = \frac{\rho_j^* - \rho_0}{\Delta t^2} k_j^{\text{DFSPH}}$
    $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W_{ij}$

Algorithm 5: Constant density solver

DFSPH Simulation Step Algorithm 6 shows a simulation step with DFSPH and how both solvers are integrated in the time step. Note that the neighborhoods, the particle densities and the factor k_i^{DFSPH} are computed once at the beginning of the simulation for the initial state and then updated once per time step. The algorithm first computes predicted velocities by integrating all non-pressure accelerations. Then the density deviation is corrected using the constant density solver which yields new particle positions. Hence, the neighborhoods, the density values and the factors must be updated. After correcting the density deviation the velocity field is typically not divergence-free. This is corrected in the last step by the divergence-free solver which gives us the final velocities. Note that the order of the steps is a bit different than the order of other solvers but in this way it is guaranteed that the density deviations and the divergence error are both corrected at the end of a time step. Moreover, in this way we have to update the factor k_i^{DFSPH} only once per time step but are able to use it twice: for the constant density solver and for the divergence-free solver.

DFSPH solves two PPEs which is more expensive than solving

for all particles i **do**
    compute non-pressure accelerations $\mathbf{a}_i^{\text{nonp}}$
adapt time step size Δt according to CFL condition
for all particles i **do**
    predict velocity $\mathbf{v}_i^* = \mathbf{v}_i + \Delta t \mathbf{a}_i^{\text{nonp}}$
correct density error using algorithm 5
for all particles i **do**
    update position $\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i + \Delta t \mathbf{v}_i^*$
update neighborhoods
for all particles i **do**
    update density ρ_i
    update factor k_i^{DFSPH} using Eq. (79)
correct divergence error using algorithm 4
for all particles i **do**
    update velocity $\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^*$

Algorithm 6: Simulation step with DFSPH

just one like PCISPH or IISPH. However, the second solve is not that expensive since the costly computation of the factor k_i^{DFSPH} has to be performed only once per step. Experiments have shown that solving both PPEs leads to a better stability which enables larger time steps and therefore a faster simulation [BK17]. The performance can be further improved by using a warm start. More details about this can be found in [BK17].

4.11. The Best Pressure Solver

Iterative PPE solvers are more expensive to compute than EOS solvers. Their utility, however, is motivated by the fact that PPE solvers work with significantly larger time steps compared to EOS solvers. Solenthaler and Pajarola [SP09] show an improved overall performance of PCISPH compared to the EOS solver in [BT07]. Ihmsen et al. [ICS*14] show an improved performance of IISPH compared to PCISPH and Bender and Koschier [BK17] show a performance gain of DFSPH compared to IISPH. So, DFSPH has the best overall performance of all discussed PPE variants.

Although PPE solvers work with big time steps, they do not reach their best overall performance for the largest possible time step as discussed in [ICS*14] and [IOS*14]. Although the number of neighborhood searches decreases for larger time steps, the solver iterations increase for larger time steps.

The reported performance gains of PPE solvers compared to EOS solvers have been estimated for so-called complex scenarios. The term complex refers basically to the height of a simulated fluid body under gravity. The higher the simulated fluid column, the more complex the scenario, the bigger the performance gain of a PPE solver. If a scenario is simple, *e.g.*, one layer of fluid particles on a planar boundary, EOS solvers are faster. The overall number of particles does not necessarily influence the solver performance. An EOS solver is more efficient than a PPE solver for one billion fluid particles in one layer on a plane, while a PPE solver is more efficient than an EOS solver for one hundred particles on top of each other in one column under gravity.

Independent from whether there is a performance gain of a PPE solver, they are more simple to handle than EOS solvers. In an EOS solver, the stiffness constant has to be found to realize a desired density deviation and the time step has to be found to get a stable simulation. In a PPE solver, the desired density deviation is explicitly specified. The time step is also easier to estimate as it is typically rather larger, corresponding to CFL numbers close to one.

5. Boundary Handling

In order to complete the discretization of a mixed initial-boundary value problem (see Section 2) the boundary of the simulation domain has to be discretized and the corresponding boundary conditions must be enforced. In recent years, a wide variety of approaches to represent boundary geometries and to enforce boundary conditions has been presented. The approaches can be roughly categorized into particle-based approaches, *e.g.*, [AIA*12,IAGT10, BT07, BGPT18, BGI*18, GPB*19], and implicit approaches, *e.g.*, [KB17, HKK07a, HKK07b, BLS12].

The particle based strategy is probably the most popular representation type. The main idea is to sample the boundary geometry using an additional set of so-called *boundary particles* equipped with a (sometimes specialized) kernel function. The advantages here are that the representation is consistent with the discretization of the fluid or solid. Modeling, either explicit/implicit boundary forces for weak satisfaction of boundary conditions or algorithms to strongly satisfy the constraints is probably more straightforward than using implicit or mesh-based techniques. Most methods, however, have the constraint that the particle size used to sample the boundary has to be the same as the particle size of the continuum discretization. The disadvantages are that even the representation of simple geometries, such as a plane, requires a large number of boundary particles that have to be accounted for during the neighborhood search and in the evaluation of field quantities, *e.g.*, Eq. (11). Moreover, determining "good" samplings is generally non-trivial. Too sparse samplings might not sufficiently cover the surface of the boundary leading to the issue of SPH particles penetrating the boundary. Too dense samplings lead to an increased computational effort and to higher memory requirements. Also a somewhat "bumpy" sampling might lead to a bias in the particle trajectories as the discretized surfaces' smoothness suffers from sampling noise leading to unwanted perturbations in the simulation (*cf.*, [KB17]) if no additional considerations are made, such as proposed by Band et al. [BGPT18].

Implicit boundaries use an implicit function – typically a signed distance field (SDF) – to represent the boundary geometry. Advantages of this type of approaches are that the boundary representation is decoupled from the particle size. As a result, more flexible data structures, *e.g.*, adaptive octrees with higher-order approximations [KDBB17], can be used to memory-efficiently and accurately represent the boundary geometry. This circumstance also avoids the problem of noisy boundary samplings, the resulting bias, and unwanted perturbations in the particle trajectories. Typical disadvantages are, that implicit representations do not directly integrate with particle based continuum discretizations. In order to couple both discretization types, special considerations have to be made and the corresponding implementation is rather involved.

In the remainder of this section, we will discuss approaches to

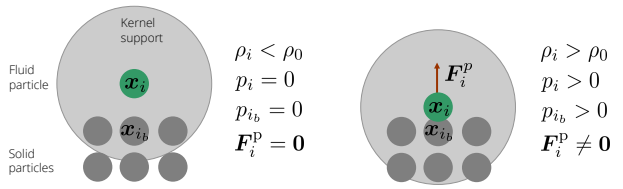


Figure 8: Idea of the particle-based boundary handling. The boundary is represented with particles $\mathbf{x}_{i_b}$. These particles are considered in the computation of density ρ_i , pressure p_i, and pressure force $\mathbf{F}_i^p$ of nearby fluid particles $\mathbf{x}_i$. If a fluid particle moves closer to the boundary, its density increases. If a fluid particle is too close, its density is larger than the rest density, *i.e.*, $\rho_i > \rho^0$, which causes pressure $p_i > 0$ which in turn causes a pressure force $\mathbf{F}_i^p \neq \mathbf{0}$ that accelerates the fluid particle away from boundary particles. Please note that i_b denote boundary neighbors of a fluid particle i. Accordingly, i_f refer to indices of fluid neighbors of a fluid particle i.

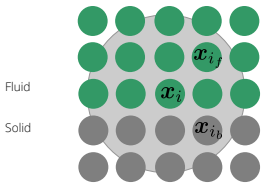


Figure 9: Sample-based boundary representation with several layers. The usage of several boundary layers avoids incomplete neighborhoods for fluid particles $\mathbf{x}_i$ close to the boundary.

handle non-penetration of rigid boundaries using particle sampling approaches. We gradually develop a formulation starting with a simple dense, uniform multilayer sampling of the boundary and show how the method can be simplified to a uniform single layer sampling and consequently even to a robust and consistent formulation using non-uniformly sampled boundaries. We moreover discuss how the fluid-boundary coupling can be improved using pressure mirroring or pressure extrapolation and how these techniques can be incorporated into the previously discussed pressure solvers. For implicit or mesh-based boundary handling techniques we would like to refer the reader to the corresponding literature, *e.g.*, [KB17, HKK07a, HKK07b, BLS12, MFK*15, FLR*13, FM15]

5.1. Particle-based Boundary Handling

In this concept, the boundary is represented with particles and these boundary particles are incorporated into the computation of density ρ_i, pressure p_i, and $\mathbf{F}_i^p$ at nearby fluid particles $\mathbf{x}_i$. This is illustrated in Fig. 8. There are two main aspects to discuss. First: The boundary can be sampled in different ways. *E.g.*, , the boundary can be sampled with particles of uniform size corresponding to the size of a fluid particle. Or the boundary can be sampled with particles of non-uniform size. Also, the boundary can be sampled with several layers of boundary particles as indicated in Fig. 8 or the boundary can be sampled with just one layer of particles. The second aspect is the actual computation of density ρ_i , pressure p_i, and pressure force $\mathbf{F}_i^p$ of nearby fluid particles $\mathbf{x}_i$. Such computations require information from boundary samples, *e.g.*, pressure. Such pressure at boundary samples can be estimated in different ways. Here, typical examples are pressure mirroring, *i.e.*, it is assumed that the pressure at the boundary particles equals the pressure at adjacent fluid particles. Alternatively, pressure can be extrapolated from the fluid into the boundary. While the pressure extrapolation is theoretically the correct choice, this concept is challenging to realize due to the fact that the computation of the pressure gradient at a fluid particle close to the boundary is error-prone. In the following, we discuss different combinations of the aforementioned variants.

5.1.1. Different Types of Boundary Samplings

Several layers with boundary samples of uniform size: If a fluid particle $\mathbf{x}_i$ is close to a boundary, it generally has fluid neighbors $\mathbf{x}_{i_f}$ and boundary neighbors $\mathbf{x}_{i_b}$ as illustrated in Fig. 9. All these neighbors contribute to the density computation, *i.e.*,

$$\rho_i = \sum_{i_f} m_{i_f} W_{ii_f} + \sum_{i_b} m_{i_b} W_{ii_b} \,. \tag{83}$$

All samples $\mathbf{x}_i, \mathbf{x}_{i_f}, \mathbf{x}_{i_b}$ have the same size and we take the boundary samples as static fluid samples resulting in the same rest density ρ^0 for all fluid and boundary particles. This also means that all masses are equal: $m_i = m_{i_f} = m_{i_b}$. Thus, the density computation could also be written as $\rho_i = m_i(\sum_{i_f} W_{ii_f} + \sum_{i_b} W_{ii_b})$. If the boundary samples belong to a rigid body, it is not perfectly intuitive to represent its boundary with particles that have mass and rest density of a fluid particle. This issue results from the fact that SPH formulations often prefer to weight the contribution of a particle i with $\frac{m_i}{\rho_i}$ instead of using its volume $V_i = \frac{m_i}{\rho_i}$. Now, in the boundary handling, one actually works with the volume of boundary particles. Nevertheless, this volume is often represented with some rest density - typically the fluid rest density - and the respective mass.

If the density ρ_i in Eq. (83) is larger than the rest density ρ^0, a pressure p_i is computed at fluid particles. We have seen that p_i can be computed from a state equation, *e.g.*, $p_i = k\left(\frac{\rho_i}{\rho^0} - 1\right)$ or from solving a PPE.

Now, the pressure force at the fluid particle can be computed as

$$\begin{aligned}\mathbf{F}_i^p = &- m_i m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2}\right) \nabla W_{ii_f} \\ &- m_i m_i \sum_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_b}}{\rho_{i_b}^2}\right) \nabla W_{ii_b} \,.\end{aligned} \tag{84}$$

It can be seen in Eq. (84) that we basically compute a pressure force component with respect to fluid neighbors and a pressure force component with respect to boundary neighbors. Using several layers of boundary particles guarantees that the neighborhood of a fluid particle is completely sampled, even if this particle is very close to the boundary. Fully filled neighborhoods keep the errors due to missing samples in Eqs. (83) and (84) small.

The computation in Eq. (84) requires positions, densities and pressures of adjacent fluid and boundary particles. While these quantities are known for fluid particles, density and pressure at

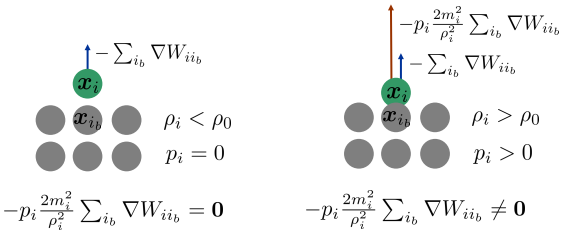


Figure 10: Working with equal pressure p_i at a fluid particle i and at its adjacent boundary samples results in a repulsion force in case of $p_i > 0$.

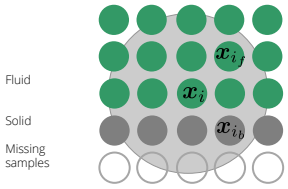


Figure 11: One layer of uniform boundary samples is easier to generate than several layers. The boundary layer is used to naturally detect the proximity of a fluid sample to a boundary. Due to the size of the kernel support, however, more than one layer of samples might be required in SPH computations. The contributions of these missing samples can be analytically estimated.

boundary samples are still unknown. As the boundary samples have a fixed volume and their mass is set with respect to the rest density of the fluid, it is appropriate to set the density of a boundary particle to the rest density of the fluid, *i.e.*, $\rho_{i_b} = \rho^0$. Regarding the pressure p_{i_b}, we have already briefly mentioned pressure mirroring and pressure extrapolation. Here, the simplest idea is to mirror the pressure from a fluid particle to an adjacent boundary particle, *i.e.*, $p_{i_b} = p_i$. These assumptions result in an adapted form of Eq. (84) for the pressure force where all required quantities at fluid and boundary neighbors are known:

$$\begin{aligned}\mathbf{F}_i^p = &- m_i m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} \\ &- m_i m_i \sum_{i_b} \left(\frac{p_i}{\rho_i^2} + \frac{p_i}{(\rho^0)^2} \right) \nabla W_{ii_b}. \end{aligned} \tag{85}$$

Working with equal pressure at a fluid sample and at its neighboring boundary samples theoretically corresponds to a pressure gradient of zero which in turn results in a pressure acceleration of zero. In practice, however, the SPH derivative approximation always results in the desired gradient with the respective repulsion force. If the neighborhood of a fluid particle is completely filled, as shown in Fig. 9, the true gradient is slightly underestimated by SPH. This leads to a small, practically not relevant amount of penetration of the fluid into the boundary. In the other case, where a single fluid particle without fluid neighbors is close to the boundary as depicted in Fig. 10, the contributions from missing fluid neighbors are implicitly assumed to be zero. Having a fluid neighbor with zero pressure or not having this fluid neighbor has the same effect on the SPH approximation.

For a single fluid particle at a boundary, Eq. (85) simplifies to $\mathbf{F}_i^p = -p_i \frac{2m_i^2}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b}$ for $\rho_i = \rho^0$. The term $-\sum_{i_b} \nabla W_{ii_b}$ can be interpreted as the surface normal of the boundary close to position $\mathbf{x}_i$. The pressure force at particle $\mathbf{x}_i$ corresponds to this vector scaled with the fluid particle pressure p_i. So, if the density of the fluid particle is larger than its rest density, we get a positive pressure and a repulsion force from the boundary into normal direction.

In summary, computing the fluid density ρ_i with Eq. (83), the pressure p_i with a state equation or a PPE, and the pressure force with Eq. (85) realizes a boundary handling with pressure mirroring in case of a uniformly sampled boundary with several layers. The boundary handling works for fluid particles with a complete or incomplete neighborhood.

One layer of uniform boundary samples: It can be difficult to generate multiple layers of uniform boundary samples for arbitrarily shaped geometries. Also, if the relative position of a fluid sample to the boundary can be determined, it is not necessarily required to explicitly represent the boundary particles. Instead, their contributions can analytically be estimated. Fig. 11 shows a setting with one layer of uniform boundary samples. These samples are used to estimate that a fluid particle is close to a boundary. For the computation of SPH approximations, however, more than one layer of samples might be required as indicated in Fig. 11.

For a given position $\mathbf{x}_i$ of a fluid particle close to a one-layer boundary, the density can be written as

$$\rho_i = m_i \sum_{i_f} W_{ii_f} + m_i \sum_{i_b} W_{ii_b} + m_i \sum_{i_m} W_{ii_m}. \tag{86}$$

The index i_m refers to missing samples in the neighborhood of particle i. While it is a natural way to encode the contributions of missing samples as an offset, these contributions are typically encoded as a correcting factor of the contributions from boundary samples instead, *i.e.*,

$$\rho_i = m_i \sum_{i_f} W_{ii_f} + \gamma_1 m_i \sum_{i_b} W_{ii_b}. \tag{87}$$

Correcting coefficients for contributions from missing boundary samples have been proposed in [AIA*12]. They are commonly used. Benefits and drawbacks to correcting offsets as in Eq. (86) are unclear and have not been analyzed yet.

The correcting coefficient γ_1 in Eq. (87) depends on various aspects, *e.g.*, kernel function, kernel support, dimensionality. In particular, however, it depends on the position of a fluid particle relative to the boundary. In practice, the correcting coefficient is determined for a template particle in a perfect sampling pattern as shown in Fig. 11. If the neighborhood of a fluid particle at the boundary is fully filled, the kernel sum over all neighbors gives one over the particle volume according to the general kernel properties, *i.e.*, $\sum_{i_f} W_{ii_f} + \sum_{i_b} W_{ii_b} = \frac{1}{V_i}$. If there are samples missing, this equation does not hold and we introduce the correcting coefficient γ_1 to ob-

tain the desired result:

$$\sum_{i_f} W_{ii_f} + \gamma_1 \sum_{i_b} W_{ii_b} = \frac{1}{V_i}. \tag{88}$$

Solving this equation gives the desired value for the correcting coefficient:

$$\gamma_1 = \frac{\frac{1}{V_i} - \sum_{i_f} W_{ii_f}}{\sum_{i_b} W_{ii_b}}. \tag{89}$$

The corrected density computation is the basis for the pressure computation which is typically not affected by missing boundary samples. A state equation just works with the density of the fluid particle itself. A PPE typically uses the densities of adjacent fluid particles. So, boundary samples are not required in the pressure computation.

In the subsequent computation of the pressure forces, however, the missing samples have to be accounted for. Similar to the density computation, a correcting coefficient γ_2 is introduced:

$$\mathbf{F}_i^p = -m_i m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - \gamma_2 p_i \frac{2m_i^2}{\rho_i^2} \sum_{i_b} \nabla W_{ii_b}. \tag{90}$$

This coefficient is derived from a property of the kernel gradient. If the neighborhood of a particle is perfectly sampled, the sum of the kernel gradient over the neighbors is zero: $\sum_{i_f} \nabla W_{ii_f} + \sum_{i_b} \nabla W_{ii_b} = \mathbf{0}$. In case of missing boundary samples, the sum is not zero and a correcting coefficient for the boundary contributions is introduced to meet the constraint:

$$\sum_{i_f} \nabla W_{ii_f} + \gamma_2 \sum_{i_b} \nabla W_{ii_b} = \mathbf{0}. \tag{91}$$

Solving this equation results in

$$\gamma_2 = \frac{\sum_{i_f} \nabla W_{ii_f} \cdot \sum_{i_b} \nabla W_{ii_b}}{\sum_{i_b} \nabla W_{ii_b} \cdot \sum_{i_b} \nabla W_{ii_b}}. \tag{92}$$

Similar to the coefficient γ_1, γ_2 also depends on dimenionality, kernel function and support. It also depends on the position $\mathbf{x}_i$ of fluid particle i relative to the boundary. In practice, however, the coefficient is determined for a template setting with perfect sampling as depicted in Fig. 11.

In summary: One-layer boundary representations are more simple to generate than multi-layer boundaries. The contributions of missing samples in the computation of the density and the pressure force at a fluid particle close to the boundary can be approximated with correcting coefficients.

One layer of non-uniform boundary samples: The next step to an even more flexible boundary representation is to work with samples of arbitrary size as proposed in [AIA*12] and illustrated in Fig. 12. This sampling is motivated by the fact that its generation is really simple. Particles can be arbitrarily placed on a boundary geometry, as long as each boundary particle is equal or smaller than a fluid particle. Even more than one boundary particle at the same position can be handled. This extreme case is certainly not optimal in terms of performance, but the boundary handling works.

The basic idea of the boundary handling with non-uniform boundary samples is the consideration of the actual contribution,

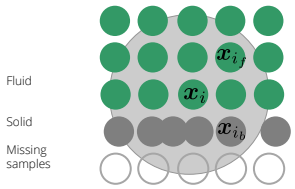


Figure 12: The boundary geometry is sampled with one layer of particles of non-uniform size. The sampling should be sufficiently dense, *i.e.*, each boundary particle should be equal or smaller than a fluid particle. Otherwise, leakage could occur.

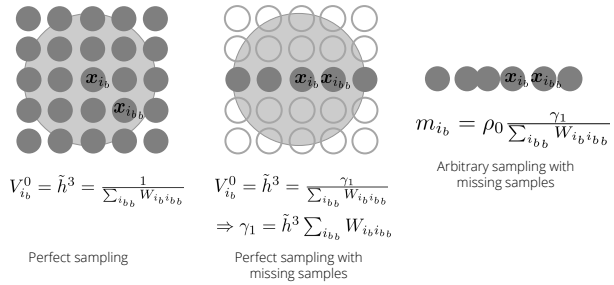


Figure 13: Derivation of the mass m_{i_b} of a boundary sample of arbitrary size. The artificial mass is computed from the volume of the sample and the rest density of the fluid. The left image shows the volume for a filled neighborhood and uniform sample size. The volume is $\tilde{h}^3$. It can also be computed as one over the sum of the kernel values from neighboring particles. In the middle image, we have removed all samples except the samples in a plane which represents the boundary. We still know that the size of $\mathbf{x}_{i_b}$ should be $\tilde{h}^3$, but the kernel sum would not provide the correct result. That's why the correcting coefficient γ_1 is introduced to get the expected result $\tilde{h}^3$. *I.e.*, γ_1 accounts for missing neighbors in the volume computation. The volume computation $V_{i_b}^0 = \frac{\gamma_1}{\sum_{i_{b_b}} W_{i_b i_{b_b}}}$ also works for arbitrary sample sizes within the plane as shown in the image on the right. The artificial mass of a boundary sample is finally computed from its volume and the rest density of a fluid particle.

i.e., the actual volume of each boundary sample. The density of a fluid particle near the boundary is computed with

$$\rho_i = m_i \sum_{i_f} W_{ii_f} + \sum_{i_b} m_{i_b} W_{ii_b}, \tag{93}$$

where m_{i_b} represents the contribution of boundary sample i_b. If a boundary sample is bigger, its contribution is bigger and this contribution is encoded in the mass m_{i_b}. The contribution, *i.e.*, the artificial mass of a boundary sample is deduced from its volume and the rest density of the fluid as explained in Fig. 13. So, the mass m_{i_b} of a boundary sample is computed as

$$m_{i_b} = \rho^0 \frac{\gamma_1}{\sum_{i_{b_b}} W_{i_b i_{b_b}}}. \tag{94}$$

The correcting coefficient γ_1 actually accounts for missing contributions in two cases. It cancels missing contributions in the computation of the mass m_{i_b} of a boundary sample. In parallel, it accounts for missing contributions in the density computation of a nearby

fluid particle using Eq. 93. The pressure force is now computed with

$$\mathbf{F}_i^p = -m_i m_i \sum_{i_f} \left(\frac{p_i}{\rho_i^2} + \frac{p_{i_f}}{\rho_{i_f}^2} \right) \nabla W_{ii_f} - \gamma_2 p_i \frac{2m_i}{\rho_i^2} \sum_{i_b} m_{i_b} \nabla W_{ii_b}, \tag{95}$$

which is very similar to the pressure force for uniform samples. The only difference is the consideration of the individual masses m_{i_b} of the boundary particles instead of the standard mass m_i for samples of uniform size. The correcting factor γ_2 is

$$\gamma_2 = \frac{\sum_{i_f} \nabla W_{ii_f} \cdot \sum_{i_b} \nabla W_{ii_b}}{\sum_{i_b} \nabla W_{ii_b} \cdot \sum_{i_b} \nabla W_{ii_b}}. \tag{96}$$

The same factor has already been used, motivated, and derived in the case of a one-layer boundary with uniform samples.

6. Viscosity

Modeling and simulating viscosity is often vital for physics simulations as the phenomenon is responsible for various visually appealing effects, such as buckling and coiling but also general energy dissipation. In recent years, various methods for the realistic simulation of low viscous fluids like water as well as highly viscous fluids like honey, mud, or dough were proposed. Fig. 14 shows different examples of highly viscous materials.

In this section we first introduce the term for the viscous force in the Navier-Stokes equations. Then we discuss important methods for the simulation of low viscous flow and highly viscous materials. We present explicit approaches which are typically used for low viscous fluids and we discuss implicit methods for highly viscous materials.

6.1. Viscous Force

In the Navier-Stokes equations for incompressible fluids, the viscosity term is determined by a material parameter μ and the Laplacian of the velocity field $\nabla^2 \mathbf{v}$ (see Eq. (32)). Before we discuss different approaches to compute this viscosity term, we first want to show how this term is derived.

The stress tensor of a Newtonian fluid is defined as

$$\mathbf{T} = -p\mathbb{1} + 2\mu\mathbf{E}, \tag{97}$$

where $\mathbf{E}$ is the strain rate tensor which is determined as

$$\mathbf{E} = \frac{1}{2}(\nabla\mathbf{v} + (\nabla\mathbf{v})^T). \tag{98}$$

When substituting the stress tensor in the conservation law of linear momentum (see Eq. (30)) and considering the incompressibility constraint $\nabla \cdot \mathbf{v} = 0$ (see Eq. (28)), we end up with the Navier-Stokes equations

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \underbrace{\nabla \cdot \nabla \mathbf{v}}_{\nabla^2 \mathbf{v}} + \mu \underbrace{\nabla \cdot (\nabla \mathbf{v})^T}_{\nabla(\nabla \cdot \mathbf{v}) = \mathbf{0}} + \mathbf{f}_{\text{ext}}, \tag{99}$$

where the right term of the strain rate tensor vanishes and the final viscosity force is determined as

$$\mathbf{f}_{\text{visco}} = \mu \nabla^2 \mathbf{v}. \tag{100}$$

Recent viscosity solvers either use a strain rate based formulation to compute viscous forces or they directly determine the Laplacian of the velocity field. While the Laplacian formulation ensures that the right term of the strain rate tensor vanishes by definition, approaches based on the strain rate have to enforce a divergence-free velocity field, otherwise $\nabla \cdot (\nabla \mathbf{v})^T \neq \mathbf{0}$ which leads to undesired bulk viscosity [Lau11, PICT15]. In the following we introduce both approaches and discuss the advantages and disadvantages.

6.2. Explicit Viscosity

In the Navier-Stokes equations for incompressible fluids the viscous force is defined by the Laplacian of the velocity field (see Eq. (100)). The standard SPH discretization of this Laplacian is determined as

$$\nabla^2 \mathbf{v}_i = \sum_j \frac{m_j}{\rho_j} \mathbf{v}_j \nabla^2 W_{ij}. \tag{101}$$

However, this formulation has two major disadvantages. First, it is sensitive to particle disorder [Mon05, Pri12]. Second, typically Gaussian-like kernel functions are used in SPH and their second derivatives changes the sign inside the support radius (see Fig. 2).

Different approaches were proposed to avoid this problem. First, instead of computing the second derivative directly, it can also be determined taking two first SPH derivatives [FMH*94, WBF*96, TDF*15]. However, this method increases the computation time and memory consumption and introduces additional smoothing. Another approach, which was introduced by Brookshaw [Bro85], is to determine one derivative using SPH and the second one using finite differences. This method is very popular and has been used for scalar quantities [Mon92, CM99, IOS*14] and for vector quantities [ER03, JSD04, Mon05, Pri12, WKBB18]. In the following we will discuss this approach in more detail.

In order to approximate the Laplacian of the velocity field we combine an SPH derivative with a finite difference derivative which yields the following equation [Mon05]:

$$\nabla^2 \mathbf{v}_i = 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij}, \tag{102}$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ and d is the number of spatial dimensions. Note that a term $0.01h^2$ is introduced in the denominator to avoid singularities. The approximation of the Laplacian in Eq. (102) has some nice features. First, it is Galilean invariant. Moreover, it vanishes for rigid body rotation. This is an important property since there is no friction if all particles rotate uniformly. Finally, the introduced formulation conserves linear and angular momentum [Mon92].

Instead of computing the Laplacian of the velocity field in order to simulate viscosity, in some works XSPH is used as artificial viscosity model (*e.g.*, [SB12]). XSPH determines the smoothed velocity $\hat{\mathbf{v}}_i$ of a particle i as

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \alpha \sum_j \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) W_{ij}, \tag{103}$$

where $0 \leq \alpha < 1$ is a user-defined parameter. The core idea of smoothing the velocity field in this way is to reduce the particle disorder by reducing the velocity difference between a particle and its

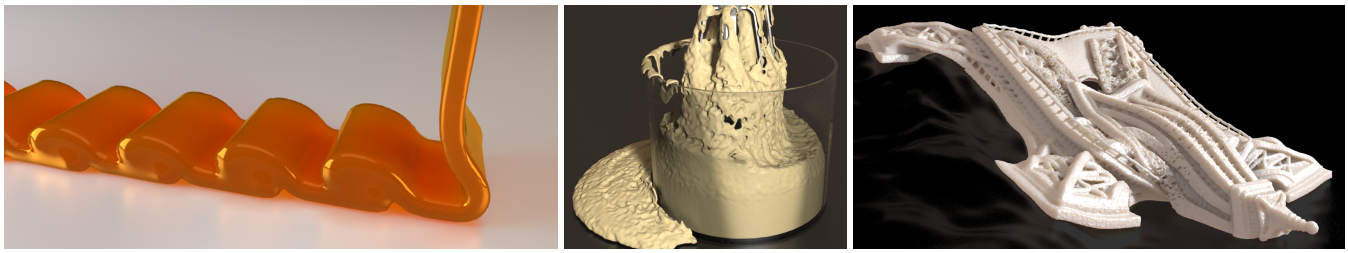


Figure 14: Different examples for the simulation of viscous behavior. Left: The simulation of highly viscous fluids enables realistic buckling effects [WKBB18]. Center: A viscous dough interacts with a fast moving solid [PICT15]. Right: A melting Eiffel tower is simulated (the model is courtesy of Pranav Panchal) [PT16].

neighborhood. The advantage of this formulation is that no kernel derivative is required. The disadvantage is that α is not physically meaningful.

6.3. Implicit Viscosity

Simulating the behavior of highly viscous materials implies that the viscosity coefficient is large. However, in this case explicit viscosity solvers tend to get unstable. Therefore, it is recommended to use an implicit method in order to simulate highly viscous fluids. In this subsection we introduce some of the most important implicit viscosity solvers [TDF*15, PICT15, PT16, WKBB18]. We first present the concepts of these solvers in chronological order and then compare the different approaches.

Takahashi et al. [TDF*15] As discussed in the previous subsection, one way to compute the second derivative of the velocity field is to take two first SPH derivatives. This approach is used by Takahashi et al. to formulate an implicit integration scheme for the viscosity term in the Navier-Stokes equations. The implicit integration enables a stable simulation of highly viscous fluids. In each simulation step Takahashi et al. first determine the strain rate $\mathbf{E}_i$ for each particle i using Eq. (98). Following the Navier-Stokes equations (see Eq. (99)) the authors then compute the divergence of the strain rate as

$$\nabla \cdot \left(\nabla \mathbf{v}_i + (\nabla \mathbf{v}_i)^T \right) = \sum_j m_j \left(\frac{2\mathbf{E}_i}{\rho_i^2} + \frac{2\mathbf{E}_j}{\rho_j^2} \right) \nabla W_{ij}. \tag{104}$$

Using this formulation the implicit integration scheme can be derived as

$$\mathbf{v}(t+\Delta t) = \mathbf{v}^* + \frac{\Delta t}{\rho} \mu \nabla \cdot \left(\nabla \mathbf{v}(t+\Delta t) + (\nabla \mathbf{v}(t+\Delta t))^T \right), \tag{105}$$

where $\mathbf{v}^*$ is the predicted velocity which is determined by integrating all non-pressure forces except viscosity. Takahashi et al. substitute Eq. (104) in Eq. (105) and solve the resulting formula to get the new velocities of the particles. The advantage of this implicit scheme is that highly viscous fluids can be simulated in a stable way while the viscosity is independent of the temporal and spatial resolution. However, in this formulation all second-ring neighbors of a particle have to be considered in order to compute one first-order SPH derivative after the other. This leads to many non-zero elements in the system matrix which decreases the performance significantly.

Peer et al. [PICT15, PT16] Instead of using a classical implicit time integration scheme, Peer et al. propose to decompose and modify the velocity gradient $\nabla \mathbf{v}$. The goal of the authors is to modify only the shear rate in order to simulate a viscous behavior. Hence, they exploit the fact that the velocity gradient can be decomposed as

$$\nabla \mathbf{v} = \mathbf{R} + \mathbf{V} + \mathbf{S}, \tag{106}$$

where $\mathbf{R} = \frac{1}{2}(\nabla \mathbf{v} - (\nabla \mathbf{v})^T)$ is the spin rate tensor, $\mathbf{V} = \frac{1}{3}(\nabla \cdot \mathbf{v})\mathbb{1}$ the expansion rate tensor and $\mathbf{S} = \mathbf{E} - \mathbf{V}$ the traceless shear rate tensor. This decomposition enables to modify the traceless shear rate tensor without influencing the other components of the velocity gradient. Therefore, the authors define a target velocity gradient

$$\nabla \mathbf{v}^{\text{target}} = \mathbf{R} + \mathbf{V} + \xi \mathbf{S} \tag{107}$$

which reduces the shear rate by a user-defined factor $0 \leq \xi \leq 1$. This modified velocity gradient can be used to determine new particle velocities by a Taylor approximation of first order

$$\mathbf{v}_i(t+\Delta t) = \frac{1}{\rho_i} \sum_j m_j \left(\mathbf{v}_j(t+\Delta t) + \frac{\nabla \mathbf{v}_i^{\text{target}} + \nabla \mathbf{v}_j^{\text{target}}}{2} \mathbf{x}_{ij} \right) W_{ij}. \tag{108}$$

This yields a linear system $\mathbf{A}\mathbf{v}(t+\Delta t) = \mathbf{b}$, where the matrix entries and the right hand side vector are defined as

$$\mathbf{A}_{ij} = -m_j W_{ij}, \tag{109}$$

$$\mathbf{A}_{ii} = \rho_i - m_i W_{ii}, \tag{110}$$

$$\mathbf{b}_i = \sum_j m_j \frac{\nabla \mathbf{v}_i^{\text{target}} + \nabla \mathbf{v}_j^{\text{target}}}{2} \mathbf{x}_{ij} W_{ij}. \tag{111}$$

This system can be decomposed to get three smaller linear systems for the x-, y- and z-component of the velocity. Finally, the authors propose to solve the three systems using a conjugate gradient method.

Later, Peer and Teschner [PT16] extended this method by simulating vorticity diffusion in order to improve the rotational motion. The diffusion process in a viscous fluid is described by $\frac{D\omega}{Dt} = \nu \nabla^2 \omega$. The authors determine $\omega = (\omega_x, \omega_y, \omega_z)^T$ from the spin rate tensor as

$$\mathbf{R} = \frac{1}{2} \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{112}$$

Analogous to Eq. (107) the vorticity is reduced by solving the system

$$\nabla^2 \omega_i^{\text{target}} = \xi \nabla^2 \omega. \tag{113}$$

The resulting vector ω_i^{target} is used to determine a target spin rate tensor $\mathbf{R}_i^{\text{target}}$ which is substituted in Eq. (107) before reconstructing the velocity field using Eq. (108).

The proposed methods are very efficient and enable a stable simulation of highly viscous materials. However, these methods have also some disadvantages. The reconstruction of the velocity field using SPH is problematic as discussed in [BGFAO17] and introduces a significant damping. When simulating highly viscous fluids, this damping effect is not that crucial but this approach is not recommended for the simulation of low viscous flow. Another disadvantage of the methods is that the viscosity parameter ξ is not physically meaningful and depends on the temporal and spatial resolution.

Bender and Koschier [BK17] The authors of this work also reduce the strain rate by introducing a user-defined coefficient which is similar to the core idea of Peer et al. However, instead of modifying the velocity gradient and reconstructing the velocity field, Bender and Koschier define a velocity constraint function $\mathbf{C}_i(\mathbf{v}) = \mathbf{E}_i - \gamma \mathbf{E}_i$ with the user-defined coefficient $0 \leq \gamma \leq 1$. The constraint is defined as six-dimensional vector function where the vector contains the elements of the upper triangular part of the symmetric strain rate tensor. Finally, the constraint is enforced by first solving the linear system

$$\left(\frac{1}{\rho_i} \frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_i} \left(\frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_i} \right)^T + \sum_j \frac{1}{\rho_i} \frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_j} \left(\frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_j} \right)^T \right) \mu_i = \mathbf{E}_i - \gamma \mathbf{E}_i \tag{114}$$

for the Lagrange multiplier μ by Jacobi iterations. The final velocities are then determined as

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \frac{1}{m_i} \left(\frac{m_i}{\rho_i} \left(\frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_i} \right)^T \mu_i + \sum_j \frac{m_j}{\rho_j} \left(\frac{\partial \mathbf{E}_j}{\partial \mathbf{v}_i} \right)^T \mu_j \right). \tag{115}$$

Details about the computation of $\partial \mathbf{E} / \partial \mathbf{v}$ can be found in [BK17].

The advantage of solving a constraint function instead of using the velocity field reconstruction approach of Peer et al. is that also low viscous fluids can be simulated. The disadvantages of the method are that solving six-dimensional constraints using Jacobi iterations is computationally expensive and the introduced viscosity coefficient depends on the temporal and spatial resolution.

Weiler et al. [WKBB18] All implicit viscosity methods introduced so far, use a formulation based on the strain rate tensor $\mathbf{E}$. The strain rate is determined by Eq. (98), where the velocity gradient $\nabla \mathbf{v}$ is computed using the following SPH discretization (see Section 2.5):

$$\nabla \mathbf{v}_i = \frac{1}{\rho_i} \sum_j m_j (\mathbf{v}_j - \mathbf{v}_i) \nabla W_{ij}^T. \tag{116}$$

Weiler et al. found out that this SPH discretization is negatively affected by the particle deficiency problem at the free surface of a fluid. For a rotational velocity field (see Fig. 15, left) the strain rate should be $\mathbf{E} = \mathbf{0}$ since a rotation is a rigid body motion which does not deform the body. However, when using the SPH discretization in Eq. (116), the strain rate is not zero at the free surface (see Fig. 15, right). In this experiment we can observe a significant error at the boundary. The main problem is that the viscosity solver tries to counteract this erroneous strain rate which leads to ghost forces. These forces causes severe visual artifacts and a loss of angular momentum which is discussed later in more detail.

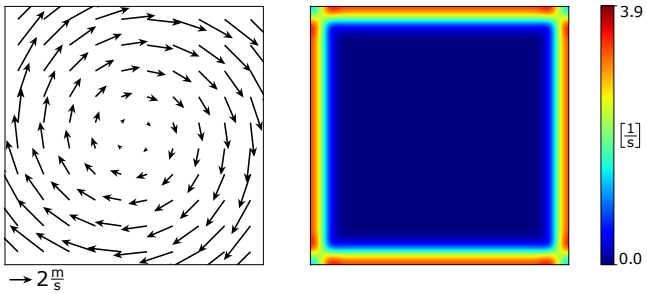


Figure 15: When computing the strain rate tensor using Eq. (116), errors occur at the free surface due to particle deficiency [WKBB18]. Left: Velocity field of a rotational motion. Right: The corresponding strain rate should be zero for all particles. However, the plot of the Frobenius norm of the tensors shows an error at the free surface.

To solve this problem, Weiler et al. developed an implicit viscosity solver which directly determines the Laplacian of the velocity field instead of using the strain rate. Their approach is based on the implicit integration scheme

$$\mathbf{v}(t+\Delta t) = \mathbf{v}^* + \frac{\Delta t}{\rho} \mu \nabla^2 \mathbf{v}(t+\Delta t). \tag{117}$$

This is similar to the one of Takahashi et al. [TDF*15] but uses the Laplacian of the velocity field instead of the divergence of the strain rate. To compute the Laplacian, the approximation in Eq. (102) is used. Since this approximation vanishes for rigid body rotations and conserves linear and angular momentum [Mon92], the proposed approach solves the problems of the methods above.

Eq. (117) is a linear system which has to be solved to get the unknown new velocities $\mathbf{v}(t+\Delta t)$. Using the SPH discretization of the Laplacian in Eq. (102), we can rewrite this linear system as

$$(\mathbb{1} - \Delta t \mathbf{A}) \mathbf{v}(t+\Delta t) = \mathbf{v}^*, \tag{118}$$

where the matrix $\mathbf{A}$ contains a 3×3 block $\mathbf{A}_{ij}$ for each pair of neighboring particles i and j:

$$\mathbf{A}_{ij} = -2(d+2) \frac{\mu \overline{m}_{ij}}{\rho_i \rho_j} \frac{\nabla W_{ij} \mathbf{x}_{ij}^T}{\|\mathbf{x}_{ij}\|^2 + 0.01 h^2}, \qquad \mathbf{A}_{ii} = -\sum_j \mathbf{A}_{ij}. \tag{119}$$

Note that the average mass $\overline{m}_{ij} = 0.5(m_i + m_j)$ is used in order to obtain a symmetric system. The resulting system can be solved efficiently by a matrix-free conjugate gradient method. The convergence can be improved by a block Jacobi preconditioner where the preconditioner matrix is block diagonal with the 3×3 blocks $\mathbb{1} - \Delta t \mathbf{A}_{ii}$. Moreover, starting the conjugate gradient solver with an

initial guess of $\mathbf{v}^* + \Delta\mathbf{v}$ using the velocity difference of the last step $\Delta\mathbf{v} = \mathbf{v}(t) - \mathbf{v}^*(t - \Delta t)$ further improves the performance.

Weiler et al. propose to extend this viscosity formulation also for the boundary in order to simulate materials that stick to solid objects. In SPH often a particle-based surface representation of the boundary is used. For such a surface representation the diagonal matrix blocks in Eq. (119) and the right hand side of the system in Eq. (118) have to be adapted as

$$\mathbf{A}_{ii} = -\sum_j \mathbf{A}_{ij} + 2(d+2)\sum_k \frac{\mu_b m_k}{\rho_i^2} \frac{\nabla W_{ik}\mathbf{x}_{ik}^T}{\|\mathbf{x}_{ik}\|^2 + 0.01h^2}, \tag{120}$$

$$\mathbf{b}_i = \mathbf{v}_i^* - 2(d+2)\Delta t \sum_k \frac{\mu_b m_k}{\rho_i^2} \frac{\mathbf{v}_k \cdot \mathbf{x}_{ik}}{\|\mathbf{x}_{ik}\|^2 + 0.01h^2} \nabla W_{ik}, \tag{121}$$

where m_k is the mass of the boundary particle k (see Section 5). Note that a reaction force has to be applied to the boundary particles to get a consistent two-way coupling [AIA*12]. Using the proposed extension sticky and separating boundaries can be simulated.

Comparison In this part we want to compare all implicit viscosity solvers introduced above. All approaches except the one of Weiler et al. [WKBB18] are based on a strain rate formulation. As discussed above and shown in Fig. 15, this leads to errors at the free surface due to particle deficiency. In practice this can lead to artifacts at the surface (see Figs. 16a, 16b and 16c). Moreover, the strain rate error at the free surface causes a significant loss of angular momentum which leads to a damped rotational motion (see Figs. 17a, 17b and 17c).

Weiler et al. analyzed these problems and proposed a new method which computes the Laplacian of the velocity field instead of using the divergence of the strain rate. Moreover, they use an SPH approximation of the Laplacian that vanishes for rigid body rotation and conserves linear and angular momentum. In this way the problems at the free surface can be solved (see Figs. 16d and 17d).

While the methods of Peer et al. [PICT15, PT16] and Bender and Koschier [BK17] use a viscosity parameter that depends on the temporal and spatial resolution, Takahashi et al. [TDF*15] and Weiler et al. [WKBB18] solve this problem by using a consistent implicit time integration.

Finally, when comparing the performance, the approach of Peer et al. [PICT15] is the fastest method. This is due to the fact that they can decompose their linear system in three smaller ones while this cannot be done for the approach of Weiler et al. Bender and Koschier use a Jacobi solver which converges slower and Takahashi et al. have to consider the second-ring neighbors which results in large computational overhead.

6.4. Conclusion

For the simulation of low viscous fluids an explicit viscosity formulation should be used since explicit methods are computationally less expensive. We recommend to compute the viscous force in Eq. (100) by approximating the Laplacian of the velocity field using Eq. (102). An alternative, which is computationally less expensive but also less accurate, is to use XSPH as artificial viscosity.

An implicit viscosity solver is recommended for the simulation of highly viscous fluids due to stability reasons. As discussed above, the implicit strain rate based formulations suffer from an error in the SPH approximation that causes visual artifacts and leads to a loss of angular momentum. The method of Weiler et al. avoids the problem and therefore generates more realistic results. Finally, we think that it would be an interesting open problem for future research to find a better SPH approximation of the strain rate without problems at the free surface. Such an approximation would solve the problems of the strain rate based formulations.

Note that all viscosity methods that were discussed in this section are implemented in our open-source framework SPlisHSPlasH [Ben19b].

7. Surface Tension

Surface tension is an important physical phenomenon which is a ubiquitous effect in daily life. For example, surface tension forces keep liquid molecules together when pouring water into a glass. The surface forces are the result of intermolecular attractive forces at microscopic scales. The molecules attract each other inside of a fluid while the molecules at the surface are pulled inwards. Therefore, surface tension minimizes the surface area which causes droplets of water to form a sphere when external forces are excluded. We typically speak of cohesion if molecules of the same type attract each other while adhesion describes the attractive forces between molecules of different types. Cohesion and adhesion are important effects when simulating surface tension.

In recent years, various methods were proposed to simulate surface tension effects in SPH-based fluid simulations (*e.g.*, [BT07, AAT13, HWZ*14]). Fig. 18 shows SPH simulation examples of surface tension. Typically we can differentiate between surface tension approaches that are inspired by a microscopic point of view and approaches that compute the forces on a macroscopic level. In the following, we will present one microscopic method and one macroscopic method in order to introduce the core idea of both approaches.

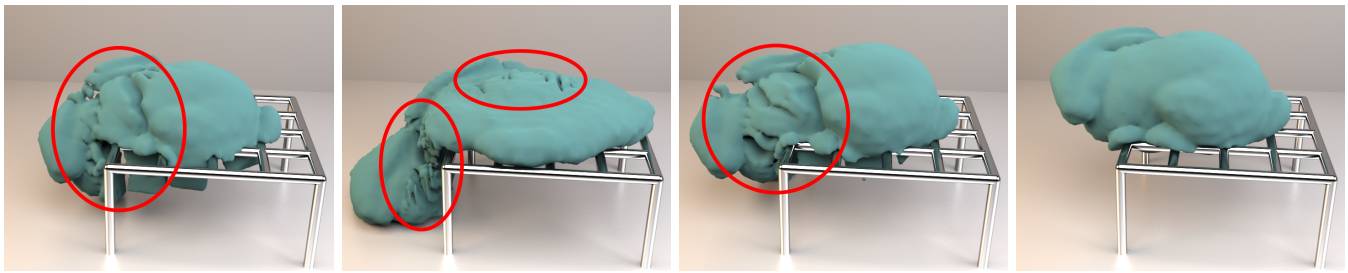
7.1. Microscopic Approach

Surface tension is the result of attracting forces between molecules. Methods that are based on a microscopic point of view aim to simulate the intermolecular cohesive forces. However, since the smallest element in an SPH simulation is a particle, these forces are determined at the particle-level.

Becker and Teschner [BT07] propose to compute the particle acceleration due to cohesion by

$$\frac{D\mathbf{v}_i}{Dt} = -\frac{\alpha}{m_i}\sum_j m_j \left(\mathbf{x}_i - \mathbf{x}_j\right) W_{ij}, \tag{122}$$

where α is a coefficient to control the surface tension of the fluid. This equation interpolates the position differences in the neighborhood of a particle i and computes an acceleration to attract the neighboring particles. Note that adhesion can also be simulated by Eq. (122) when using a sum over all neighboring boundary particles.



(a) Takahashi et al. [TDF*15] (b) Peer & Teschner [PT16] (c) Bender & Koschier [BK17] (d) Weiler et al. [WKBB18]

Figure 16: The SPH approximation of the strain rate tensor is erroneous at the free surface due to particle deficiency which leads to artifacts (a,b,c). The formulation of Weiler et al. avoids this strain rate computation by using the Laplacian of the velocity field which solves this problem (d) [WKBB18].

(a) Takahashi et al. [TDF*15] (b) Peer & Teschner [PT16] (c) Bender & Koschier [BK17] (d) Weiler et al. [WKBB18]

Figure 17: When simulating the rope coiling effect with the introduced implicit viscosity solvers, we can observe a loss of angular momentum caused by errors in the SPH strain rate approximation (a,b,c). Using a formulation based on the Laplacian of the velocity field instead, solves this problem and enables a uniform coiling (d) [WKBB18].

7.2. Macroscopic Approach

Cohesion Akinci et al. [AAT13] present a method based on a macroscopic point of view. Instead of only considering cohesive forces, also forces to minimize the surface area are determined. Their method computes the cohesive force of a particle as

$$\mathbf{F}_{i\leftarrow j}^{\text{cohesion}} = -\alpha m_i m_j \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} W^{\text{cohesion}}\big(\|\mathbf{x}_i - \mathbf{x}_j\|\big), \tag{123}$$

where i and j are neighboring particles and W^{cohesion} is a special cohesion kernel which is defined by

$$W^{\text{cohesion}}(r) = \frac{32}{\pi h^9} \begin{cases} (h-r)^3 r^3 & 2r > h \wedge r \leq h \\ 2(h-r)^3 r^3 - \frac{h^6}{64} & r > 0 \wedge 2r \leq h \\ 0 & \text{otherwise.} \end{cases} \tag{124}$$

In contrast to the model of Becker and Teschner [BT07] the cohesive forces can become positive and negative. In this way repulsion forces for close particles are generated which prevents undesired particle clustering at the free surface.

Additionally, Akinci et al. compute a force in order to minimize the surface area. This additional force counteracts the surface curvature which requires the computation of the surface normals. The normals can be determined by a so-called color field. The idea is to set the color of a particle to 1 while it is 0 everywhere else. Then the gradient of the smoothed color field

$$\mathbf{n}_i = h \sum_j \frac{m_j}{\rho_j} \nabla W_{ij} \tag{125}$$

yields a surface normal pointing into the fluid. Note that the factor h is used to make the normal scale independent. The magnitude of the resulting vector is close to zero in the interior of the fluid and proportional to the curvature at the free surface. Hence, a symmetric force that counteracts the curvature can be defined as

$$\mathbf{F}_{i\leftarrow j}^{\text{curvature}} = -\alpha m_i (\mathbf{n}_i - \mathbf{n}_j). \tag{126}$$

This force is used to minimize the surface area.

Finally, both forces are combined as

$$\mathbf{F}_{i\leftarrow j}^{\text{surface tension}} = K_{ij}(\mathbf{F}_i^{\text{cohesion}} + \mathbf{F}_i^{\text{curvature}}), \tag{127}$$

where $K_{ij} = \frac{2\rho_0}{\rho_i + \rho_j}$ is a symmetric factor that amplifies the surface tension forces at the free surface. At the surface ρ_i and ρ_j are underestimated due to particle deficiency and $K_{ij} > 1$ while for a particle with a full neighborhood $K_{ij} \approx 1$.

Adhesion To simulate the attractive forces between fluid particles and the boundary, an adhesion force is introduced as

$$\mathbf{F}_{i\leftarrow k}^{\text{adhesion}} = -\beta m_i m_k W_{ik}^{\text{adhesion}} \frac{\mathbf{x}_i - \mathbf{x}_k}{\|\mathbf{x}_i - \mathbf{x}_k\|}, \tag{128}$$

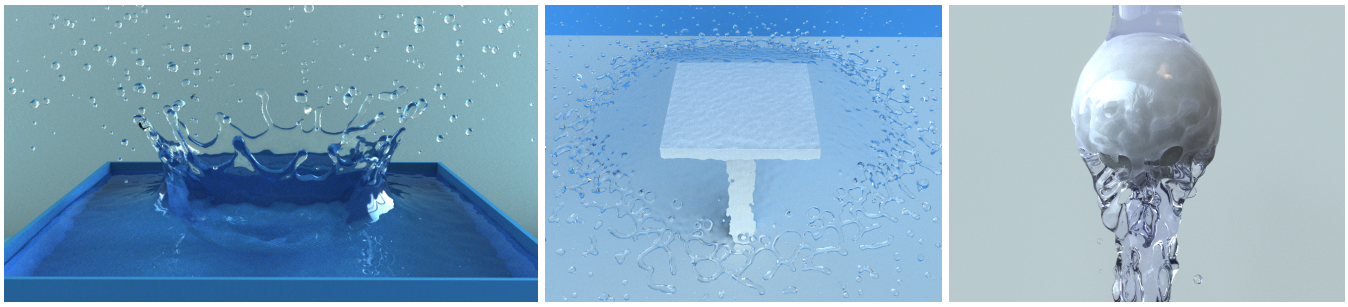


Figure 18: Different surface tension effects are realized by approximating intermolecular attractive forces in the fluid [AAT13].

where β is the adhesion coefficient and k denotes a neighboring boundary particle. The computation of the mass m_k of a boundary particle is described in detail in Section 5. Akinci et al. propose to use another specialized kernel function for the computation of the adhesion forces which is defined as

$$W^{\text{adhesion}}(r) = \frac{0.007}{h^{3.25}} \begin{cases} \sqrt[4]{-\frac{4r^2}{h} + 6r - 2h} & 2r > h \wedge r \leq h \\ 0 & \text{otherwise.} \end{cases} \tag{129}$$

Note that only fluid particles with a distance between $h/2$ and h are attracted by the boundary.

8. Vorticity

One of the most visually appealing phenomena in dynamic fluids is the evolution of chaotic structures due to turbulences. Turbulent motions are largely caused by the interaction of many unsteady vortices on various scales. A vortex is, moreover, defined as a local spinning motion in the fluid – mathematically spoken the vorticity is a vector field

$$\omega = \nabla \times \mathbf{v}. \tag{130}$$

In SPH fluid simulations turbulent details quickly get lost due to numerical diffusion [dGWH*15] or due to coarse sampling of the velocity field [IOS*14, CIPT14] which negatively influences the visual liveliness of the flow. In order to facilitate the formation of vortices in the simulation and to counteract numerical diffusion, a range of approaches has been proposed in the past. Most of these approaches originate from research concerning Eulerian, grid-based discretizations and can roughly be categorized into vorticity confinement techniques, Lagrangian vortex methods, fluid up-sampling, and more recently micropolar models. In this section we will discuss a simple vorticity confinement approach following Macklin and Müller [MM13] and an SPH discretization of a micropolar model that facilitates the formation of vortices as proposed by Bender et al. [BKKW18].

8.1. Vorticity Confinement

As already discussed, SPH discretizations tend to overly dissipate energy in turbulent flow. Therefore, Macklin and Müller [MM13] employ a method based on vorticity confinement in order to counteract the dissipation by amplifying existing vortices. The technique consists of three steps:

1. The vorticity ω_i for each particle i is computed using a discrete curl operator, *e.g.*,
$$\omega_i = \nabla \times \mathbf{v}_i = -\sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \times \nabla_i W_{ij}. \tag{131}$$
2. A corrective force is computed and applied that amplifies the already existing vortical motion, *i.e.*,
$$\mathbf{F}_i^{\text{vorticity}} = \varepsilon^{\text{vorticity}} \left(\frac{\eta}{\|\eta\|} \times \omega_i \right) \tag{132}$$
$$\eta = \sum_j \frac{m_j}{\rho_j} \|\omega_j\| \nabla_i W_{ij}, \tag{133}$$
where $\varepsilon^{\text{vorticity}}$ denotes a small constant used to steer the amount of amplification.
3. The velocity field is smoothed using XSPH (see Eq. (103)) in order to enforce a coherent particle motion.

Algorithmically, this correction is applied just before the advection of the SPH particle positions.

While this approach is very simple and effectively amplifies existing vortices it has some drawbacks. It is hard to choose the control parameter $\varepsilon^{\text{vorticity}}$ such that overamplification is avoided. Moreover, the method can, in the best case, only conserve existing vortices but does not facilitate the formation of new ones.

8.2. Micropolar Model

The most prominent mathematical model describing the dynamics of Newtonian fluids is the Navier-Stokes model. As described in Section 2 the model can be derived from the conservation law of linear momentum (see Eq. (30)) and by presuming that the mechanical stress is composed of isotropic pressure and a diffusing viscous term. An important assumption of the model is that the infinitesimally small particles which compose a fluid continuum are not subject to rotational motion. This also implies that the law of angular momentum conservation is identically fulfilled if and only if the stress tensor is symmetric.

In this section we introduce the concept of micropolar fluids and present a material model that generalizes the Navier-Stokes equations for the simulation of incompressible, inviscid turbulent flow as proposed by Bender at al. [BKKW18]. Following the definition of Łukaszewicz [Łuk99], a micropolar fluid follows constitutive laws modeled using a generally non-symmetric stress tensor.

Moreover, the definition includes that the fluid consists of rigid, spherical (and therefore rotationally invariant) particles. Based on the non-symmetric stress measures, the micropolar model additionally models rotating motions of the infinitesimal spherical particles using an angular velocity field. Due to the additional rotational degrees of freedom, the generation of vortices is facilitated and a wider range of potential dynamic effects are captured by the model. Please note, that for this section we will neglect any dissipation terms, such as viscosity, as the main goal is to generate undamped, highly turbulent flows. For the complete model, we would like to kindly refer the reader to the original paper [BKKW18].

Balance Law for Angular Momentum Conservation

Similar to the conservation law of linear momentum (see Eq. (30)) a balance law for angular momentum can be derived, *i.e.*,

$$\rho\Theta\frac{D\omega}{Dt} = \mathbf{T}_\times + \tau, \tag{134}$$

with $[\mathbf{T}_\times]_i = \sum_j \sum_k \varepsilon_{ijk} T_{jk}$, where ε_{ijk} and τ denote the Levi-Civita tensor and external body torque. Further, Θ represents a scalar, isotropic microinertia coefficient. A physical interpretation for this quantity is that each infinitesimal fluid particle has a certain inertial resistance against rotational accelerations. Bender et al. suggest to choose $\Theta = 2\mathrm{m}^2\,\mathrm{s}^{-1}$ based on experimentation. We would further like to stress the fact that Θ is not at all related to the spatial extents of an SPH particle as it is defined in the continuous setting.

Constitutive Model

It is essential to understand that the classical model actually also respects angular momentum conservation (see Eq. (134)). It is in this context, however, rarely explicitly mentioned as the balance law is usually identically fulfilled based on two assumptions. Firstly, the classical approach does not model external torques, *i.e.*, $\tau \equiv \mathbf{0}$. Secondly, stress tensor $\mathbf{T}$ is usually chosen as a symmetric tensor and such that $\mathbf{T}_\times \equiv 0$ and, hence, $\frac{D\omega}{Dt} \equiv 0$. For this reason, the balance law of angular momentum is not particularly useful for symmetric stress measures.

In order to account for the microstructured particles and to utilize the balance law of angular momentum, Bender et al. propose to use the following constitutive relation:

$$\mathbf{T} = -p\mathbb{1} - \mu_t \nabla\mathbf{v} + \mu_t \omega^\times, \tag{135}$$

with $\left[\omega^\times\right] = \sum_i \varepsilon_{jik}\omega_i$, where μ_t denotes the "transfer coefficient". We will later discuss a physical interpretation of the term in the final PDE that is controlled using μ_t. In order to ensure consistency with the second law of thermodynamics $\mu_t \geq 0$ must be satisfied. Please note that this constitutive model allows for a non-symmetric stress resulting in the fact that we have to explicitly account for the angular balance law in our simulation.

Equations of Motion

As the conservation laws and constitutive equation are now established, we can finally derive the augmented equations of motion that build the basis for the numerical simulation. By plugging the constitutive relation (135) into the conservation laws (30) and (134) and by applying the incompressibility condition (28), we arrive at the following representation:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu_t \nabla\times\omega + \frac{\mathbf{f}_{\text{ext}}}{\rho} \tag{136}$$

$$\Theta\frac{D\omega}{Dt} = \nu_t(\nabla\times\mathbf{v} - 2\omega) + \frac{\tau}{\rho}, \tag{137}$$

where ν_t denotes the kinematic transfer coefficient. Looking at Eq. (136), we quickly notice that it is identical to the inviscid Navier-Stokes equation (Euler equation) but augmented by the term $\nu_t\nabla\times\omega$. A complementary term also governed by the transfer coefficient resides in Eq. (137), *i.e.*, $\nu_t(\nabla\times\mathbf{v} - 2\omega)$. The terms effectively convert angular accelerations into linear accelerations and vice versa. Physically, they can be interpreted as dissipation-free friction or as a dissipation-free viscosity coupling linear and rotational motion.

In order to realize the model in the implementation, it is required to discretize Eqs. (136) and (137). This means that additional to a discrete representation of $\mathbf{x}$ and $\mathbf{v}$ it is necessary to discretize the vorticity ω. Please note, that due to the assumption in the micropolar model, that the microstructure of the material particles is spherical we do not have to discretize and track the rotational field (only the angular velocity field) which makes the implementation less complicated and delivers better performance. The transfer forces and torques can then, following the splitting approach, simply be applied in line with the non-pressure forces and integrated explicitly as they are considerably less stiff than the pressure forces. It is further advised to "filter" the resulting velocity field using XSPH (see Eq. (103)) in order to ensure coherent particle motion. Algorithm 7 shows an exemplary pseudocode of the resulting method.

for all *particle i* **do**
  Find neighbors
for all *particle i* **do**
  Compute density ρ_i
for all *particle i* **do**
  Compute non-pressure forces $\mathbf{F}_i^{\text{non-pressure}}$
  Compute transfer forces $\mathbf{F}_i^{\text{transfer}} = m_i\nu_t\nabla\times\omega_i$
  Compute transfer torque $\boldsymbol{\tau}_i^{\text{transfer}} = m_i\nu_t(\nabla\times\mathbf{v}_i - 2\omega_i)$
Compute time step size Δt according to CFL
for all *particle i* **do**
  $\mathbf{v}_i^* = \mathbf{v}_i + \frac{\Delta t}{m_i}(\mathbf{F}_i^{\text{non-pressure}} + \mathbf{F}_i^{\text{transfer}} + \mathbf{F}_i^{\text{ext}})$
for all *particle i* **do**
  Enforce incompressibility using pressure solver
  Update $\mathbf{v}_i^*$
for all *particle i* **do**
  $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^*$
  $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i + \Delta t\,\mathbf{v}_i(t+\Delta t)$
  $\omega_i(t+\Delta t) = \omega_i(t) + \frac{\Delta t}{m_i\Theta_i}(\boldsymbol{\tau}_i^{\text{transfer}} + \boldsymbol{\tau}_i^{\text{ext}})$

Algorithm 7: Simulation loop for SPH simulation turbulent micropolar fluids.

Bender et al. [BKKW18] demonstrated the effect of the transfer coefficient using an intuitive example (see Fig. 19). In the experiment a fluid flowing in a narrow channel was simulated. Moreover,

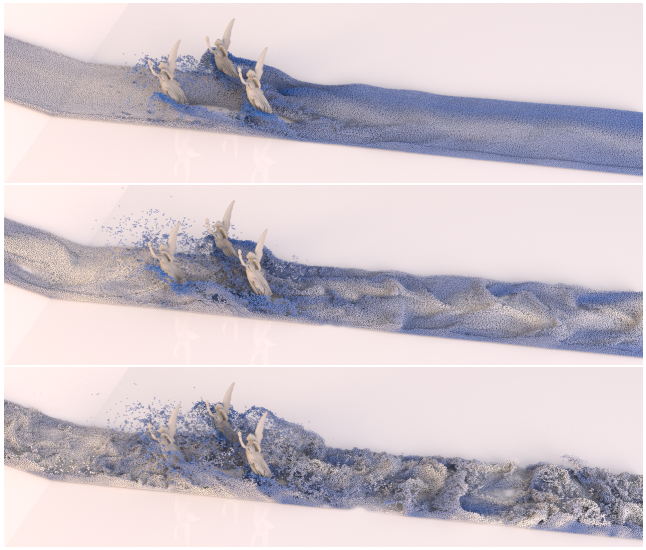


Figure 19: Simulation of 4.7M turbulent fluid particles with three obstacles and increasing transfer coefficient ν_t. Top-down: $\nu_t = 0.2\text{m}^2\,\text{s}^{-1}$, $\nu_t = 0.3\text{m}^2\,\text{s}^{-1}$, $\nu_t = 0.4\text{m}^2\,\text{s}^{-1}$.

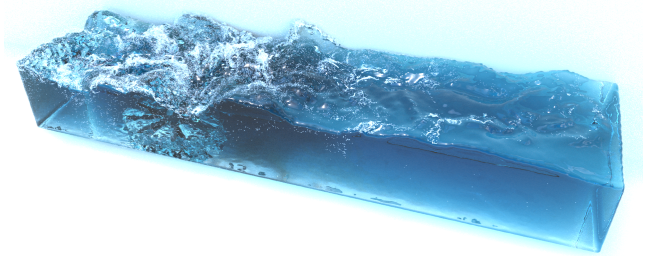


Figure 20: 1M fluid particles interact with a fast rotating propeller resulting in highly turbulent flow.

three obstacles were placed in the channel to provoke turbulences while the transfer coefficient was continuously increased. In the top image we can see that that the flow is only moderately turbulent for a transfer coefficient $\nu_t = 0.2\text{m}^2\,\text{s}^{-1}$. For larger values the vorticity significantly increases (middle) and even tends to get unrealistic for values greater than $0.4\text{m}^2\,\text{s}^{-1}$ (bottom). Furthermore, they showcase the visual realism that can be achieved in turbulent scenarios (see Fig. 20).

Discussion

Two methods to improve the behavior of the simulation in the presence of turbulences have been explained. In this paragraph, we would like to discuss the similarities and differences between vorticity confinement and the micropolar model.

Both methods build on the concept of obtaining/maintaining a vorticity field (angular velocity field) ω following Eq. (130). However, the main idea of vorticity confinement is to merely identify and amplify existing vortices. Moreover, the vorticity will always be derived from the linear field. In contrast, the micropolar approach builds on the concept of angular momentum conservation and on modeling a constitutive model for turbulences. In this more sophisticated setting, the velocity field and the vorticity are discretized independently and strongly coupled via the transfer terms in Eqs. (136) and (137). In this way there is a complex interaction between both physical quantities that not only conserves existing vortices better but also facilitates the formation of new vortices. This effect can be exemplified using the lid-driven cavity experiment carried out by Bender et al. [BKKW18] (see Fig. 21). In this experiment the "lid" (top-side) of a two-dimensional domain filled with water is accelerated with constant velocity. Given suitable model parameters, the velocity field is expected to stabilize in a big central vortex and three minor vortices rotating in the opposite direction. This result shows that vorticity confinement successfully amplifies the vortical motion but is not able to form the additional corner vortices. In contrast, the micropolar approach yields the expected result.

9. Multiphase Fluids

The simulation of multiple immiscible and miscible fluids greatly enhance visual effects in graphics. In contrast to Eulerian approaches, the particle representation of SPH offers the advantage that fluid interfaces are sharply defined. In this section, we first present how the standard SPH equations can be adapted to model density discontinuity across fluid interfaces, and introduce the resulting adapted force equations. We then discuss models for capturing complex mixing phenomena.

9.1. Fluid Interfaces

A simple approach to simulate multiple fluids with SPH is to assign different labels to particles of different phases, and assigning them with corresponding physical attributes such as masses and rest densities [MSKG05]. Typically, each particle's rest volume remains constant to ensure a uniform particle sampling, thus $\frac{m_a}{\rho_a^0} = \frac{m_b}{\rho_b^0}$ for two fluid types a and b. The momentum equation can be solved with the single flow SPH formulation presented in the previous sections, while simply using the physical attributes stored on the particles. However, for high density ratios between phases, this can lead to instability problems that are not time step related. The desired density discontinuity across the interface is smoothed due to the nature of SPH of summing up contributions from particle neighbors. As a consequence, pressure and force fields are affected, which manifests as spurious interface tension [Hoo98, AMS*07] between the phases. Larger density ratios between the fluids (>10x) intensify the problems and severely degrade simulation stability regardless of the time step size.

To capture the density discontinuity across the interface with SPH, the number density $\delta_i = \sum_j W_{ij}$ was introduced and the standard SPH equations were adapted accordingly [TM05, HA06, SP08]. The density of a particle i is then computed as

$$\tilde{\rho}_i = m_i \delta_i. \tag{138}$$

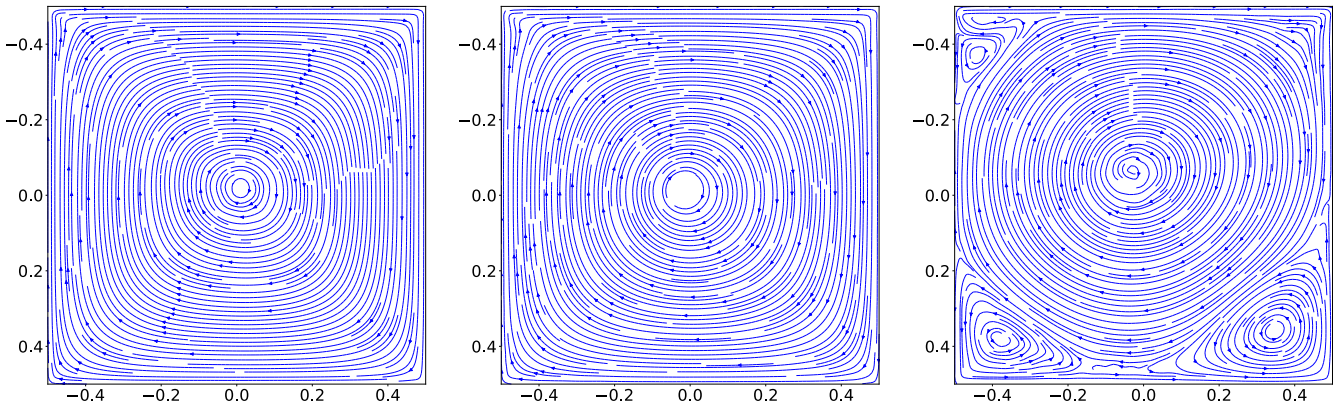


Figure 21: Velocity fields of the lid-driven cavity benchmark. Standard SPH (left) and vorticity confinement (middle) are only able to produce one large central vortex. In contrast, the micropolar approach yields (right) the expected result, *i.e.*, one central vortex and three smaller vortices in the corners which are rotating in the opposite direction.

Like this, the density of particle i is not influenced by the mass of its neighbors j, while still receiving the geometric contribution W_{ij} from j. The state equation of Sections 4.4 can then be changed such that the pressure is computed with the adapted density as

$$\tilde{p}_i = k_1 \left(\left(\frac{\tilde{\rho}_i}{\rho^0} \right)^{k_2} - 1 \right). \tag{139}$$

Solenthaler et al. [SP08] derived adapted forces by substituting $\tilde{\rho}_i$ and $\tilde{p}_i$ into the Navier-Stokes equations and applying the SPH formalism. The resulting pressure force term is then given as $\mathbf{F}^p = -\frac{\nabla \tilde{p}}{\delta}$. By employing the quotient rule we then get $\frac{\nabla \tilde{p}}{\delta} = \nabla(\frac{\tilde{p}}{\delta}) + \frac{\tilde{p}}{\delta^2}\nabla\delta$. After applying the SPH rule and replacing V by $\frac{1}{\tilde{\rho}}$, the pressure force equation can be written as

$$\mathbf{F}_i^p = -\sum_j \left(\frac{\tilde{p}_j}{{\delta_j}^2} + \frac{\tilde{p}_i}{{\delta_i}^2} \right) \nabla W_{ij}. \tag{140}$$

Similar derivations can be found in [TM05, HA06]. The viscosity force (and other force terms) can be derived analogously and is given as

$$\mathbf{F}_i^v = \frac{1}{\delta_i} \sum_j \frac{\mu_i + \mu_j}{2} \frac{1}{\delta_j} (\mathbf{v}_j - \mathbf{v}_i) \nabla^2 W_{ij}. \tag{141}$$

Note that the above equations are identical to the standard SPH equations when applied to a single phase flow. For multiple fluids, however, the adapted method eliminates any spurious tension effects and notably increases stability. The method has been extended with an incompressibility condition and solid-fluid coupling [AIA*12, GPB*19], an example is shown in Fig. 22. Moreover, the resolution at the interface has been increased using the two-scale (or multi-scale) particle simulation method in [SG11, HS13].

The above described problems can be circumvented by replacing the summation density by the continuity equation that evolves the density over time (Section 2.4) and hence does not suffer from smoothing artifacts across fluid interfaces. However, this typically requires higher-order time stepping schemes and careful considerations of time step sizes to avoid accumulation of integration errors over time and thus drift from true mass conservation [SP08, SB12]. The density summation equation was also used in combination with the Shepard kernel to accurately preserve the discontinuity at the interface [GAC*09]. The method considers the volume distribution and the rate of change of the volume estimated by the continuity equation.

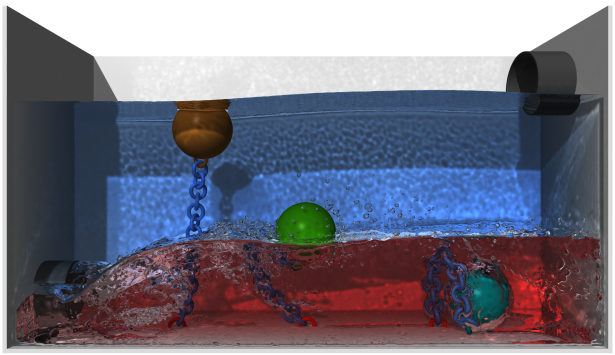


Figure 22: Three solid buoys interacting with two IISPH fluids with different densitites [GPB*19].

9.2. Complex Mixing Phenomena

Fluid mixing can be simulated by solving the diffusion equation $\frac{\partial C}{\partial t} = \alpha \nabla^2 C$, which evolves the concentration C over time. With SPH, this equation can be written as [MSKG05]

$$\frac{\partial C_i}{\partial t} = \alpha \sum_j m_j \frac{C_j - C_i}{\rho_j} \nabla^2 W_{ij}, \tag{142}$$

where α defines the diffusion strength. Another SPH formulation for computing the diffusion has been presented in [LLP11].

More complex mixing effects can be simulated by taking the

flow motion and force distributions into account as demonstrated in the SPH-based mixture model of Ren et al. [RLY*14]. The continuity equation of the mixture model is defined as

$$\frac{\mathrm{D}\rho_m}{\mathrm{D}t} = \frac{\partial \rho_m}{\partial t} + \nabla \cdot (\rho_m \mathbf{v}_m) = 0, \tag{143}$$

where ρ_m is the rest density of the mixture and $\mathbf{v}_m$ is the mixture velocity, averaged over all phases. ρ_m and $\mathbf{v}_m$ are computed using the volume fraction α_k of a phase k with rest density ρ_k, *i.e.*, $\rho_m = \sum_k \alpha_k \rho_k$ and $\mathbf{v}_m = \frac{1}{\rho_m \sum_k \alpha_k \rho_k \mathbf{v}_k}$. The momentum equation for the mixture is given as

$$\frac{\mathrm{D}(\rho_m, \mathbf{v}_m)}{\mathrm{D}t} = -\nabla p + \nabla \cdot (\tau_m + \tau_{Dm}) + \rho_m \mathbf{g}, \tag{144}$$

where τ_m and τ_{Dm} are the mixture's viscous stress and diffusion tensors, respectively.

In each simulation step, the drift velocity $\mathbf{v}_{mk} = \mathbf{v}_k - \mathbf{v}_m$ is computed, which represents the relative velocity of phase k to the mixture. The equation can be rewritten using individual terms for slip velocity due to body forces, pressure effects that cause fluid phases to move from high to low pressure regions, and a Brownian diffusion term that represents phase drifting from high to low concentration regions. The drift velocity is then used to calculate the diffusion tensor τ_{Dm} and change in volume fraction $\mathrm{D}\alpha_k/\mathrm{D}t$. The SPH equations for the mixture model described above can be found in the work of Ren et al. [RLY*14]. They demonstrate complex mixing effects including chemical reactions. The model uses WCSPH, since a divergence-free velocity field cannot be directly integrated since neither the mixture nor phase velocities are zero, even if the material is incompressible.

Yan et al. [YJL*16] extended the mixture model to handle the interaction between fluid and solid phases, and demonstrated various effects including dissolution of solids, flows in porous media, and interaction with elastic materials. Another extension has been presented by Yang et al. [YCR*15] where an energy-based model was used. The approach integrates the Cahn-Hilliard equation that describes phase separation, expanding the capability of a multi-fluid solver and enabling incompressible flows.

10. Deformable Solids

The simulation of deformable solids is an active research topic in computer graphics. The most popular simulation approaches in this area, like the finite element method (FEM) [KBT17, KKB18] and Position-Based Dynamics (PBD) [BKCW14, BMM14, BMM17], are mesh-based. However, also meshless methods were investigated like the moving least squares (MLS) method [AW09]. In this section we show that SPH is also an interesting meshless method to simulate deformable solids. An advantage of an SPH-based simulation of deformables is that this enables a simple coupling between fluids and solids in a unified framework.

10.1. Linear Elasticity

In this subsection we first introduce a continuum mechanical formulation for linear elasticity. In the next subsection we then show how to discretize the resulting equations using SPH.

The deformation of a solid is defined by the function

$$\Phi(\mathbf{X}) = \mathbf{X} + \mathbf{u} = \mathbf{x} \tag{145}$$

which maps a point $\mathbf{X}$ in the reference configuration to its current position $\mathbf{x}$ in the deformed configuration, where $\mathbf{u} = \mathbf{x} - \mathbf{X}$ is the displacement vector. Differentiating this function with respect to the reference position $\mathbf{X}$ gives us the deformation gradient

$$\mathbf{J} = \frac{\partial \Phi(\mathbf{X})}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \mathbb{1} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}. \tag{146}$$

This quantity can be used to measure the strain of a deformed body. In computer graphics often a linear strain measure is used to avoid the solution of a non-linear system of equations. For the same reason we introduce the linear infinitesimal strain tensor

$$\varepsilon(\mathbf{J}) = \frac{1}{2}(\mathbf{J} + \mathbf{J}^T) - \mathbb{1}. \tag{147}$$

The next step is to define a constitutive model for linear elasticity. We follow the work of Sifakis [Sif12] and define it in terms of the strain energy density:

$$\Psi(\mathbf{J}) = \mu \varepsilon : \varepsilon + \frac{\lambda}{2} \mathrm{tr}^2(\varepsilon), \tag{148}$$

where μ and λ are the Lamé coefficients [Sif12]. The first Piola–Kirchhoff stress tensor is determined by differentiating the strain energy density with respect to the deformation gradient

$$\mathbf{P}(\mathbf{J}) = \frac{\partial \Psi}{\partial \mathbf{J}}. \tag{149}$$

For our linear elasticity model this yields

$$\mathbf{P}(\mathbf{J}) = 2\mu\varepsilon + \lambda \mathrm{tr}(\varepsilon)\mathbb{1}. \tag{150}$$

Note that the Lamé coefficients can be computed from Young's modulus k and Poisson's ratio ν by

$$\mu = \frac{k}{2(1+\nu)}, \qquad \lambda = \frac{k\nu}{(1+\nu)(1-2\nu)}. \tag{151}$$

This is often more intuitive since Young's modulus is a measure of stretch resistance while Poisson ratio is a measure of incompressibility. Finally, the elastic body forces are determined as the divergence of the stress tensor

$$\mathbf{f} = \nabla \cdot \mathbf{P}. \tag{152}$$

In the following subsection we will discuss how this continuous elastic material model can be discretized using the SPH formulation.

10.2. SPH Discretization

The deformation of an elastic body is determined with respect to its reference configuration (see Eq. (145)). Typically the initial shape of a body is used as reference configuration in an SPH simulation. Since the topology of an elastic body does not change during the simulation, we store the neighborhood of each particle i in the reference configuration. In the following this reference neighborhood is denoted by $\mathcal{N}_i^0$.

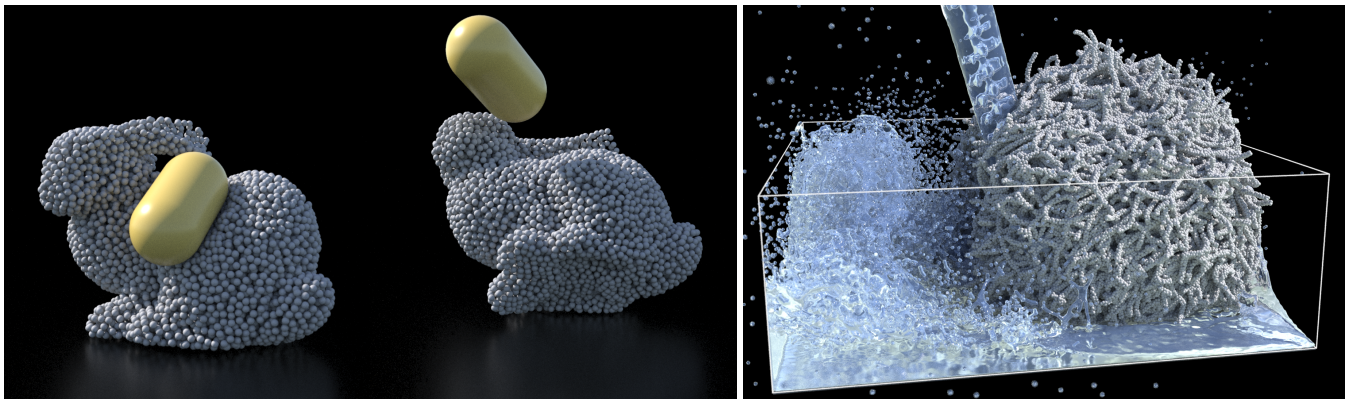


Figure 23: Deformable solids simulated using an SPH formulation [PGBT17]. Left: An elastic bunny model is hit by a rigid capsule. Right: A deformable hairball interacts with water.

Deformation Gradient A straightforward SPH discretization of the deformation gradient is given by

$$\mathbf{J}_i(\mathbf{x},\mathbf{X}) = \sum_{j \in \mathcal{N}_i^0} V_j^0 \mathbf{x}_{ji} \nabla W\left(\mathbf{X}_{ij}\right)^T, \tag{153}$$

where $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$ and $\mathbf{X}_{ij} = \mathbf{X}_i - \mathbf{X}_j$. Since this SPH approximation is determined in the reference configuration, the rest volume V_j^0 has to be used in the sum. However, this formulation fails to capture rotational motion since it is not first-order consistent (see Section 2.3).

Kernel Gradient Correction Bonet and Lok [BL99] have shown that the gradient of the kernel has to fulfill the following condition to ensure that the computation is first-order consistent and therefore correctly captures rotational motion:

$$\sum_{j \in \mathcal{N}_i^0} V_j^0 \mathbf{X}_{ji} \nabla W\left(\mathbf{X}_{ij}\right)^T = \mathbb{1}. \tag{154}$$

Now we can formulate a corrected kernel gradient which satisfies the condition by construction:

$$\tilde{\nabla} W_i\left(\mathbf{X}_{ij}\right) = \mathbf{L}_i \nabla W\left(\mathbf{X}_{ij}\right), \tag{155}$$

where $\mathbf{L}_i$ is a correction matrix that is defined as

$$\mathbf{L}_i = \left(\sum_{j \in \mathcal{N}_i^0} V_j^0 \nabla W\left(\mathbf{X}_{ij}\right) \mathbf{X}_{ij}^T \right)^{-1}. \tag{156}$$

Note that this correction matrix only depends on the rest volume and the particle positions in the reference configuration. Therefore, this matrix is precomputed at the beginning of the simulation. If the matrix in Eq. (156) is singular and cannot be inverted, *e.g.*, due to a collinear or coplanar particle configuration, the Moore–Penrose inverse is used instead.

Corotated Approach Using the corrected kernel gradient we get a first-order consistent SPH formulation for the deformation gradient:

$$\mathbf{J}_i(\mathbf{x},\mathbf{X}) = \sum_{j \in \mathcal{N}_i^0} V_j^0 \mathbf{x}_{ji} \tilde{\nabla} W\left(\mathbf{X}_{ij}\right)^T. \tag{157}$$

The deformation gradient is used to compute the linear infinitesimal strain tensor by Eq. (147). Note that we use a linear strain measure since in an implicit formulation it is more efficient to solve a linear system than a non-linear one. However, the linear strain tensor is not invariant under rotations. In computer graphics a common solution for this problem is to use a corotational approach [BIT09, KKB18]. The core idea of this approach is to extract the rotation and to compute the strain measure in an unrotated frame. In the following we will show how the rotation can be extracted and introduce the computation of a corotated deformation gradient.

The deformation gradient computed with the corrected kernel gradient (see Eq. (157)) is able to capture the rotation correctly. Hence, the per-particle rotation $\mathbf{R}_i$ can be directly extracted from $\mathbf{J}_i$, *e.g.*, by using the efficient and stable method of Müller et al. [MBCM16].

The extracted rotation matrix $\mathbf{R}$ is used to rotate the reference configuration so that the resulting displacement vector $\mathbf{u} = \mathbf{x} - \mathbf{R}\mathbf{X}$ contains no rotation. Since we rotate the reference configuration, we also have to rotate the corrected kernel gradient as it depends on the reference positions. This yields the rotated corrected kernel gradient

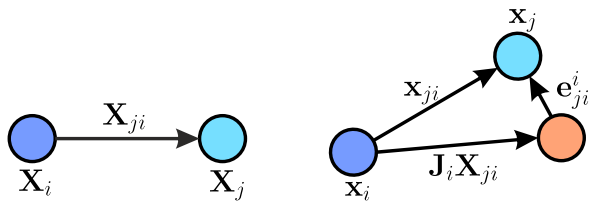
$$\overset{*}{\nabla} W_i\left(\mathbf{X}_{ij}\right) = \mathbf{R}_i \mathbf{L}_i \nabla W\left(\mathbf{X}_{ij}\right). \tag{158}$$

Putting all together gives us the corotated deformation gradient

$$\mathbf{J}_i^*(\mathbf{x},\mathbf{X}) = \mathbb{1} + \sum_{j \in \mathcal{N}_i^0} V_j^0 \left(\mathbf{x}_{ji} - \mathbf{R}_i \mathbf{X}_{ji}\right) \overset{*}{\nabla} W_i\left(\mathbf{X}_{ij}\right)^T. \tag{159}$$

Now we compute the strain tensor $\varepsilon(\mathbf{J}^*)$ using Eq. (147) and the stress tensor $\mathbf{P}(\mathbf{J}^*)$ using Eq. (150). Finally, the force is determined as the divergence of the stress tensor. In our SPH formulation this yields [Gan15]:

$$\mathbf{F}_i\left(\mathbf{J}^*\right) = \sum_{j \in \mathcal{N}_i^0} V_i^0 V_j^0 \left(\mathbf{P}_i\left(\mathbf{J}_i^*\right) \overset{*}{\nabla} W_i\left(\mathbf{X}_{ij}\right) - \mathbf{P}_j\left(\mathbf{J}_j^*\right) \overset{*}{\nabla} W_j\left(\mathbf{X}_{ij}\right) \right). \tag{160}$$



(a) reference configuration (b) deformed configuration

Figure 24: Due to numerical errors the transformed reference vector $\mathbf{J}_i\mathbf{X}_{ji}$ and the current deformed vector $\mathbf{x}_{ji}$ are typically not equal in an SPH simulation. The error is defined by the difference $\mathbf{e}^i_{ji} = \mathbf{J}_i\mathbf{X}_{ji} - \mathbf{x}_{ji}$.

If we simply add these particle forces to our system, we get an explicit approach for the simulation of deformable bodies. However, this approach is only conditionally stable and requires small time steps when simulating stiff solids. To improve the stability we will introduce an implicit approach in the next subsection.

10.3. Implicit Approach

The implicit method described in the following is based on the work of Peer et al. [PGBT17]. Since the elastic forces depend linearly on the particle positions, an implicit formulation of the time step is straightforward

$$\mathbf{v}^{t+\Delta t} = \mathbf{v}^t + \frac{\Delta t}{m}\mathbf{F}\left(\mathbf{J}^{t+\Delta t}\right), \tag{161}$$

where $\mathbf{J}^{t+\Delta t} = \mathbf{J}^*\left(\mathbf{x}^{t+\Delta t}, \mathbf{X}\right)$ is the deformation gradient at the end of the time step. This means that we use the new particle positions $\mathbf{x}^{t+\Delta t}$ to determine the elastic forces at the end of the time step.

In this formulation we have unknown positions $\mathbf{x}^{t+\Delta t}$ and unknown velocities $\mathbf{v}^{t+\Delta t}$. In the next step we substitute the positions by $\mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t\mathbf{v}^{t+\Delta t}$ to get a linear system for the new velocities. Moreover, we split the computation of the force into

$$\mathbf{F}\left(\mathbf{J}^{t+\Delta t}\right) = \mathbf{F}\left(\mathbf{J}^t\right) + \mathbf{F}\left(\mathbf{J}^{\Delta t}\right), \tag{162}$$

where $\mathbf{J}^t = \mathbf{J}^*\left(\mathbf{x}^t, \mathbf{X}\right)$ is the deformation gradient at the beginning of the time step and $\mathbf{J}^{\Delta t} = \mathbf{J}^*\left(\Delta t\mathbf{v}^{t+\Delta t}, \mathbf{0}\right)$ is the deformation gradient that corresponds to the position change in one time step due to the velocities $\mathbf{v}^{t+\Delta t}$. Now we can bring all terms that depend on the unknown new velocities to the left hand side and the rest to the right hand side:

$$\mathbf{v}^{t+\Delta t} - \frac{\Delta t}{m}\mathbf{F}\left(\mathbf{J}^*\left(\Delta t\mathbf{v}^{t+\Delta t}, \mathbf{0}\right)\right) = \mathbf{v}^t + \frac{\Delta t}{m}\mathbf{F}\left(\mathbf{J}^*\left(\mathbf{x}^t, \mathbf{X}\right)\right). \tag{163}$$

This yields a linear system for the velocities $\mathbf{v}^{t+\Delta t}$ which can efficiently be solved using a matrix-free conjugate gradient method. More details about the matrix-free solver can be found in the work of Peer et al. [PGBT17].

10.4. Zero-Energy Mode Suppression

In SPH simulations zero-energy modes can occur which are similar to hour glass modes in finite element methods [Gan15]. The displacement field in the neighborhood of a particle i has to be defined exactly by its corresponding deformation gradient $\mathbf{J}_i$. This means that if we transform the vector $\mathbf{X}_{ji}$ from particle i to one of its neighbors j in reference space using the deformation gradient $\mathbf{J}_i\mathbf{X}_{ji}$, this should give us the actual vector $\mathbf{x}_{ji}$ in the deformed configuration. Hence, the vector

$$\mathbf{e}^i_{ji} = \mathbf{J}_i\mathbf{X}_{ji} - \mathbf{x}_{ji} \tag{164}$$

should be zero. However, this is typically not the case due to numerical errors (see Fig. 24).

Ganzenmüller [Gan15] proposes to compute a penalty force to minimize the error vector $\mathbf{e}_{ji}$:

$$\mathbf{F}_i^{\text{HG}} = -\frac{1}{2}\alpha k \sum_{j\in\mathcal{N}_i^0} V_i^0 V_j^0 \frac{W(\mathbf{X}_{ij})}{\|\mathbf{X}_{ij}\|^2}\left(\frac{\mathbf{e}^i_{ji}\cdot\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|} + \frac{\mathbf{e}^j_{ij}\cdot\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}\right)\frac{\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|}, \tag{165}$$

where the coefficient α controls the amplitude of the zero-energy mode suppression and k is the Young's modulus. In this way the system gets more stable and hourglass modes are suppressed.

11. Rigid Solids

Recently, it has been demonstrated that SPH can also be used to realize a rigid body simulation with contact handling [GPB*19]. In the following we will introduce the SPH formulation for rigid bodies and show that this enables a strong two-way coupling of fluids and rigid bodies (see Fig. 25).

The core idea of the SPH-based contact handling for rigid bodies is similar to the concept of particle-based boundary handling (see Section 5). Therefore, in the beginning of the simulation the surface of each rigid body is sampled by particles. But instead of only computing pressure forces between fluid and boundary particles, we also determine artificial pressure forces between the particles on the surfaces of the rigid bodies in order to resolve contacts. Due to the unified particle representation of all bodies in the simulation, the neighborhood search can be used to detect the collision of rigid body particles. Hence, no additional collision detection method for rigid bodies is required.

Such an SPH-based rigid body solver can easily be combined with SPH discretizations of other materials. This enables a simple two-way coupling of fluids, rigid bodies, deformable solids and highly viscous materials (see Fig. 26).

11.1. Rigid Body Solver

In the following we will discuss how to compute rigid-rigid contact forces $\mathbf{F}^{\text{rr}}$. We want to compute these forces similar to the fluid-rigid interface forces which were discussed in Section 5. Therefore, we first introduce an artificial rest density $\rho_r^0 = 1$ for each rigid particle r. Note that the magnitude of the rest density can be chosen arbitrarily since we are only interested in a density deviation. If there is a contact, we get a density deviation of $\rho_r - \rho_r^0 > 0$ for

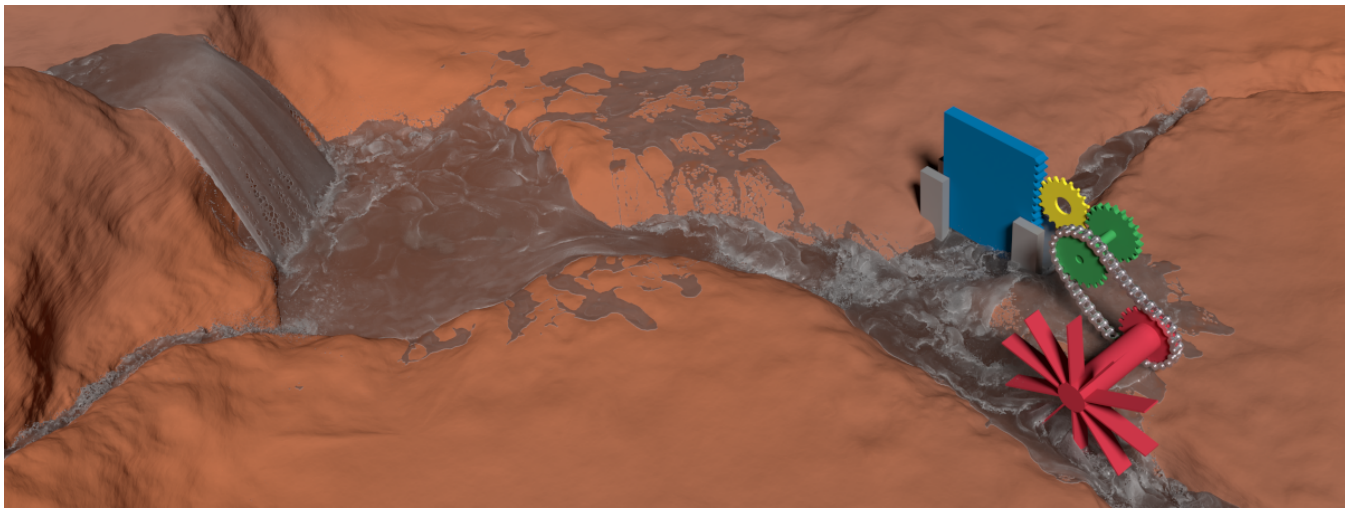


Figure 25: The SPH-based rigid body solver enables a strong two-way coupling between fluids and rigid bodies. In this scenario 43.8M fluid particles interact with 50M static and 2.3M dynamic rigid body particles. Up to 90k simultaneous rigid-rigid contacts were resolved.

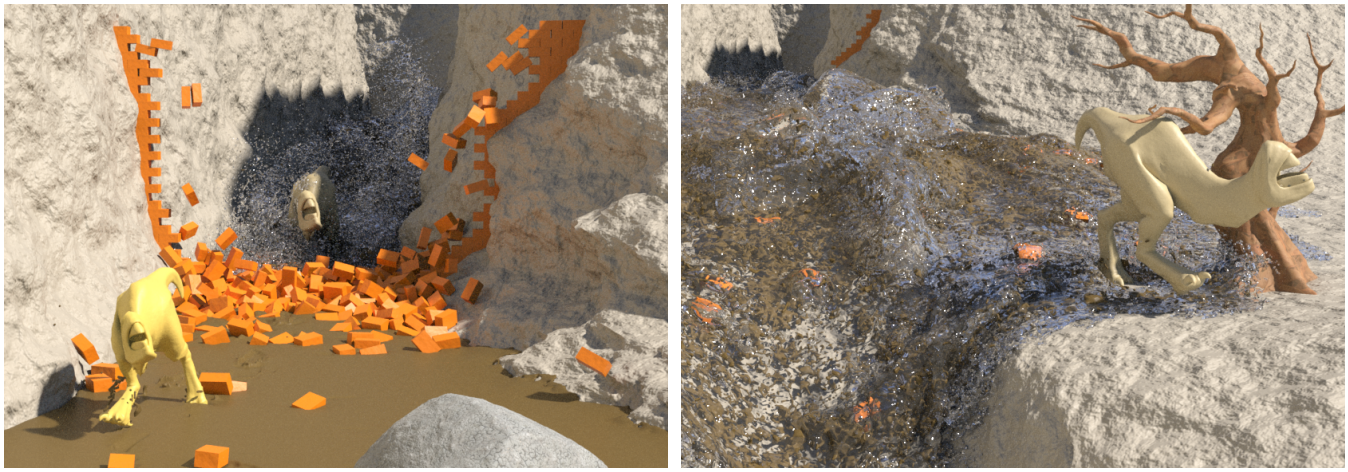


Figure 26: Two creatures run in highly viscous mud, break through a wall of rigid bodies and collide with a deformable tree. This complex simulation shows the power of a unified SPH solver.

the corresponding particle r. In this case our goal is to determine contact forces $\mathbf{F}^{\text{rr}}$ such that $\rho_r = \rho_r^0$.

Now we will derive an implicit method to compute the unknown contact forces. We start with the continuity equation

$$\frac{D\rho_r}{Dt} = -\rho_r \nabla \cdot \mathbf{v}_r, \tag{166}$$

where ρ_r and $\mathbf{v}_r$ are the density and the velocity of a rigid body particle r, respectively. Then we use a backward difference time discretization and introduce a constant density constraint $\rho_r^{t+\Delta t} = \rho_r^0$ to get

$$\frac{\rho_r^0 - \rho_r}{\Delta t} = -\rho_r \nabla \cdot \mathbf{v}_r^{t+\Delta t}, \tag{167}$$

where $\mathbf{v}_r^{t+\Delta t}$ is the velocity of the rigid body particle r at time $t+\Delta t$. This velocity vector can be written as

$$\mathbf{v}_r^{t+\Delta t} = \mathbf{v}_R^{t+\Delta t} + \omega_R^{t+\Delta t} \times \mathbf{r}_r^{t+\Delta t}, \tag{168}$$

where $\mathbf{v}_R^{t+\Delta t}$ and $\omega_R^{t+\Delta t}$ are the linear and angular velocity of the rigid body R at time $t+\Delta t$, respectively, and $\mathbf{r}_r^{t+\Delta t}$ is the vector from the center of mass of the rigid body to the position of the particle r. The new velocities are determined by an Euler integration step

$$\mathbf{v}_R^{t+\Delta t} = \mathbf{v}_R + \Delta t \frac{1}{m_R} \left(\mathbf{F}_R + \sum_{k \in \mathcal{R}} \mathbf{F}_k^{\text{rr}} \right) \tag{169}$$

$$\omega_R^{t+\Delta t} = \omega_R + \Delta t \mathbf{I}_R^{-1} \left(\boldsymbol{\tau}_R + (\mathbf{I}_R \omega_R) \times \omega_R + \sum_{k \in \mathcal{R}} \mathbf{r}_k \times \mathbf{F}_k^{\text{rr}} \right), \tag{170}$$

where $\mathbf{I}_R$ is the inertia tensor of the rigid body. The vectors $\mathbf{F}_R$

and $\boldsymbol{\tau}_R$ contain all forces and torques acting on the body except the unknown rigid-rigid contact forces $\mathbf{F}_k^{rr}$. $\mathcal{R}$ denotes the set of all particles of rigid body R. Note that all quantities on the right hand side are at time t. For improved readability we omitted the time parameter for all quantities at the current time t.

In the next step we substitute Eqs. (168)-(170) in Eq. (167) to get a linear system for the unknown rigid-rigid contact forces

$$\begin{aligned}\frac{\rho_r^0 - \rho_r}{\Delta t} =& -\rho_r \nabla \cdot \left(\mathbf{v}_R + \frac{\Delta t}{m_R} \mathbf{F}_R \right) - \rho_r \nabla \cdot \left(\frac{\Delta t}{m_R} \sum_{k \in \mathcal{R}} \mathbf{F}_k^{rr} \right) \\ & - \rho_r \nabla \cdot \left(\left(\omega_R + \Delta t \mathbf{I}_R^{-1} \left(\boldsymbol{\tau}_R + (\mathbf{I}_R \omega_R) \times \omega_R \right) \right) \times \mathbf{r}_r^{t+\Delta t} \right) \\ & - \rho_r \nabla \cdot \left(\Delta t \left(\mathbf{I}_R^{-1} \sum_{k \in \mathcal{R}} \mathbf{r}_k \times \mathbf{F}_k^{rr} \right) \times \mathbf{r}_r^{t+\Delta t} \right). \end{aligned} \tag{171}$$

To simplify this system we use the approximation $\mathbf{r}_r^{t+\Delta t} = \mathbf{r}_r$. Moreover, we introduce the velocity vector

$$\mathbf{v}_r^s = \mathbf{v}_R + \frac{\Delta t}{m_R} \mathbf{F}_R + \left(\omega_R + \Delta t \mathbf{I}_R^{-1} \left(\boldsymbol{\tau}_R + (\mathbf{I}_R \omega_R) \times \omega_R \right) \right) \times \mathbf{r}_r. \tag{172}$$

This vector determines the new velocity of a particle r after a time step which considers all forces and torques except the unknown contact forces. In this way we can write the right-hand side of our linear system in a compact form:

$$s_r = \frac{\rho_r^0 - \rho_r}{\Delta t} + \rho_r \nabla \cdot \mathbf{v}_r^s. \tag{173}$$

The left-hand side contains all terms of Eq. (171) that depend on the rigid-rigid contact forces $\mathbf{F}_k^{rr}$. The resulting linear system has the form

$$-\rho_r \nabla \cdot \left(\frac{\Delta t}{m_R} \sum_{k \in \mathcal{R}} \mathbf{F}_k^{rr} + \left(\Delta t \mathbf{I}_R^{-1} \sum_{k \in \mathcal{R}} \mathbf{r}_k \times \mathbf{F}_k^{rr} \right) \times \mathbf{r}_k \right) = s_r. \tag{174}$$

The left-hand side can further be simplified by introducing the matrix

$$\mathbf{K}_{rk} = \frac{1}{m_R} \mathbb{1} - \tilde{\mathbf{r}}_r \mathbf{I}_R^{-1} \tilde{\mathbf{r}}_k, \tag{175}$$

where $\tilde{\mathbf{r}}_r$ is the cross product matrix of $\mathbf{r}_r$ to get

$$-\rho_r \nabla \cdot \left(\Delta t \sum_{k \in \mathcal{R}} \mathbf{K}_{rk} \mathbf{F}_k^{rr} \right) = s_r. \tag{176}$$

Note that the matrix $\mathbf{K}_{rk}$ is well-known in the area of rigid body solvers [Mir96, BET14].

Our goal is to resolve the contacts by a pressure force. Therefore, we define

$$\mathbf{F}_k^{rr} = -V_k \nabla p_k, \tag{177}$$

where V_k is an artificial volume of particle k and p_k is an unknown pressure which is used to resolve the collision. This yields the final linear system

$$\rho_r \nabla \cdot \left(\Delta t \sum_{k \in \mathcal{R}} V_k \mathbf{K}_{rk} \nabla p_k \right) = \frac{\rho_r^0 - \rho_r}{\Delta t} + \rho_r \nabla \cdot \mathbf{v}_r^s. \tag{178}$$

Solving the linear system gives us the unknown pressure values for all rigid particles. Note that the linear system contains one equation for each rigid particle. However, if a particle r has no contact to a particle of another rigid body, we can remove the corresponding equation from the system and set $p_r = 0$ as no contact must be resolved in this case.

11.2. Implementation

In the following we describe how the quantities in the derived linear system are computed.

The artificial rest volume of a rigid particle r is determined as

$$V_r^0 = \frac{0.7}{\sum_{k \in \mathcal{R}} W_{rk}}. \tag{179}$$

A detailed discussion about this computation can be found in the work of Gissler et al. [GPB*19]. Together with the artificial rest density $\rho_r^0 = 1$ the actual density of a rigid particle is computed as $\rho_r = \sum_k V_r^0 \rho_r^0 W_{rk}$, where the sum considers the particles k of all rigid bodies within the support radius of the kernel. Due to the sum over the particles of neighboring rigid bodies, we get a density deviation of $\rho_r - \rho_r^0 > 0$ in case of a contact. In this case we compute the actual volume of a rigid particle r as $V_r = \frac{\rho_r^0 V_r^0}{\rho_r}$.

The divergence on the right-hand side of the linear system is determined as

$$\nabla \cdot \mathbf{v}_r^s = \frac{1}{\rho_r} \sum_{k \in \mathcal{R}} V_k \rho_k \left(\mathbf{v}_k^s - \mathbf{v}_r^s \right) \cdot \nabla W_{rk}. \tag{180}$$

Finally, we solve the linear system using a relaxed Jacobi solver and update the pressure in iteration $l+1$ as

$$p_r^{l+1} = p_r^l + \frac{\beta_r^{RJ}}{b_r} \left(s_r - \rho_r \nabla \cdot \left(\Delta t \sum_{k \in \mathcal{R}} V_k \mathbf{K}_{rk} \nabla p_k^l \right) \right), \tag{181}$$

where b_r is the diagonal element of the linear system and β_r^{RJ} is the relaxation coefficient which is set to $\beta_r^{RJ} = \frac{0.5}{\text{num\_contacts}}$.

11.3. Conclusion

The introduced SPH-based rigid body solver enables a strong coupling between fluids and rigid bodies. As shown in Fig. 27 the solver is able to accurately handle complex scenarios with thousands of simultaneous contacts. It can be easily extended to simulate friction effects [GPB*19]. Finally, together with the SPH-based simulation of deformable solids (see Section 10), it can be combined to a unified SPH solver which supports the coupling of fluids, rigid bodies, deformable solids and highly viscous materials (see Fig. 26).

12. Data Driven Fluid Simulation

Using machine learning for fluid simulations is a largely unexplored research area, but first results are promising and indicate the potential of such data-driven approaches. In the Lagrangian context, the seminal work of Ladický et al. [LJS*15] employed Regression forests to infer particle accelerations (and velocities) using handcrafted, SPH-based features. We discuss this work in more detail below. Um et al. [UHT18] presented a method to augment

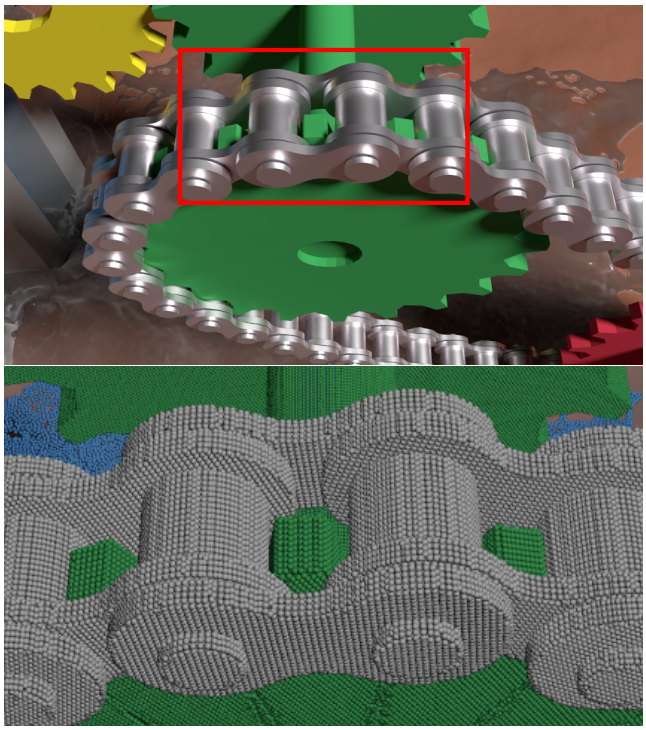


Figure 27: Closeup of the water gate scene in Fig. 25. The single chain elements are connected only by rigid-rigid contacts.

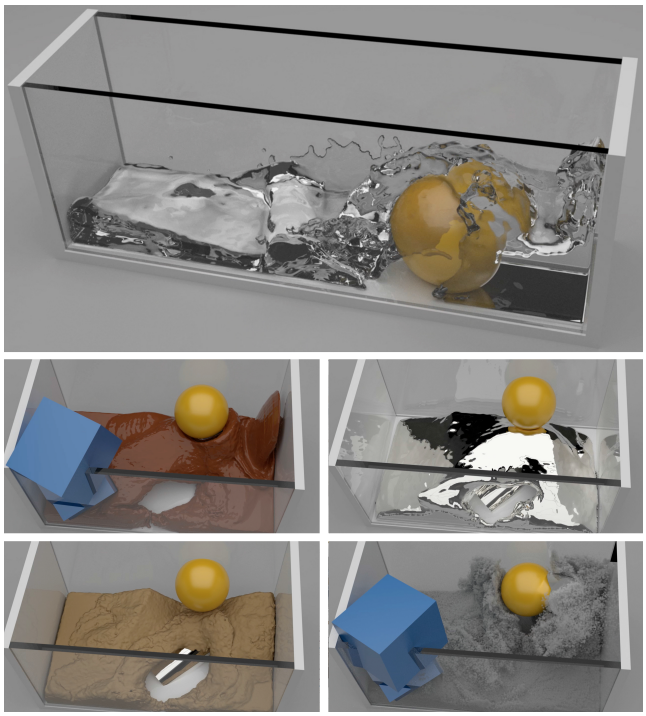


Figure 28: The regression forest approach for SPH [LJS*15] enables real-time simulations of over a million particles. At runtime, additional external forces can be added to mimic different material properties without retraining the model.

simulations with learned splashes from a high-resolution FLIP simulation, but included also an example where SPH training data was used. Somewhat related to SPH, Schenk et al. [SF18] proposed a differentiable PBF solver [MM13] for deep neural networks. They have presented convolution layers for summing up contributions from neighbors and for fluid-object interaction, which potentially can be adapted to SPH fluids as well. Other work mainly focused on Eulerian simulations, for example to substitute the pressure projection step with a CNN [TSSP16], to synthesize flow simulations from a set of reduced parameters [KAT*19], to compute smoke super-resolution with GAN networks [XFCT18], or to predict pressure field changes for multiple subsequent time-steps with LSTM [WBT18].

12.1. Regression Fluid

In the following, we give an overview of the regression forest approach for SPH presented by Ladický et al. [LJS*15]. The work aimed at enabling real-time applications of millions of particles for games and virtual reality applications. The main idea is to formulate an SPH solver as a regression problem, where the acceleration (or velocity) of each particle at time $t+\Delta t$ is efficiently estimated given the state at time t.

As input to the regressor, a feature vector $\Phi_{\mathbf{x}_i}$ is evaluated for each particle. The features are designed such that they represent the individual forces and constraints of the Navier-Stokes equations: the used features model pressure, incompressibility, viscosity and surface tension. In order to evaluate features without using an explicit neighbor search step, context-based integral features are computed that are defined as flat-kernel sums of rectangular regions surrounding a particle. The different box sizes allow to capture the behavior of both close and distant particles, and the features can be evaluated in constant time and are robust to small deviations. More details on the computation of context-based integral features can be found in [LJS*15]. Three different ways were considered for the learning strategy:

1. Learning naïve prediction: The first approach directly learns particle accelerations $\mathbf{a}$, given the evaluated features at state t. The regression problem is formulated as

$$\mathbf{a}_i(t) := Reg(\Phi_{\mathbf{x}_i}), \tag{182}$$

where $Reg(.)$ is the learned regression function. Velocities and positions are then integrated with

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \mathbf{a}_i(t)\Delta t \tag{183}$$

$$\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \frac{\mathbf{v}_i(t)+\mathbf{v}_i(t+\Delta t)}{2}\Delta t. \tag{184}$$

This formulation mimics standard SPH and hence does not consider incompressibility.

2. Learning prediction with hindsight: The second strategy first computes external forces, advects particles, and applies collision handling. Then, this intermediate state with particle positions $\mathbf{x}_i^*$ is used to compute integral features. The regression learns a corrective

acceleration and is defined as

$$\mathbf{a}_i(t) := Reg(\Phi_{\mathbf{x}_i^*}), \tag{185}$$

followed by advection as in Eqs. (183) and (184). Unlike the naïve approach, the regressor is able to predict compressions and hence to counteract those with a corrective acceleration. Conceptually, this approach mimics PCISPH [SP09].

3. Learning correction: The third approach starts similarly as the second one, but instead of learning accelerations, corrective velocities are computed. The regression problem is defined as

$$\Delta\mathbf{v}_i^{corr} := Reg(\Phi_{\mathbf{x}_i^*}), \tag{186}$$

and positions and velocities are updated with

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^*(t) + \Delta\mathbf{v}_i^{corr} \tag{187}$$

$$\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i^*(t) + \frac{\Delta\mathbf{v}_i^{corr}}{2}\Delta t. \tag{188}$$

This approach counteracts compressions as well, and conceptually mimics PBF [MM13]. Unlike PBF, the regressor takes into account information from a larger neighborhood, and hence does not require several iterations to converge.

For training the regression forest, 165 scenes consisting of 1-6 million particles and moving obstacles (sphere, box, cylinder) were randomly generated and computed for 6 seconds. The training time was 4 days on 12 CPUs, and the size of the resulting model was about 40 MB. With the regression fluid approach it is possible to simulate 1 to 1.5 million particles in real-time, and hence this approach represents an attractive alternative to traditional solvers for games and virtual reality applications (Fig. 28).

With the naïve prediction, strong compression artifacts are visible. The system cannot self-correct in the next frames since the model has never seen such distorted states during the training. Both prediction with hindsight and learning corrections can handle incompressibility well, however the third approach seems to lead to smaller errors compared to the ground truth data. Additionally, with the second and third approaches, external forces can be added without retraining the model. This allows adding surface tension, friction, or drag effects at runtime to mimic different material properties as illustrated in Fig. 28. The disadvantage of the regression fluids approach - and in fact of all machine learning based strategies - is that learning methods are not capable to extrapolate the model far outside the training data (*e.g.*, when domain size or fluid resolution change).

13. SPlisHSPlasH

In this section we want to introduce SPlisHSPlasH [Ben19b] which is an SPH-based open-source library for the physically-based simulation of fluids (see Fig. 29). The SPlisHSPlasH framework contains a reference implementation of many of the methods introduced in this tutorial and several simulations shown in the figures were performed using this library. Therefore, we think that our open-source framework perfectly supplements these course notes.

In the current version SPlisHSPlasH implements six of the most popular explicit and implicit pressure solvers [BT07, SP09, MM13, ICS*14, BK15, WKB16] which enable the simulation of incompressible fluids with several million particles. Moreover, the explicit [SB12, Mon92] and implicit viscosity methods [TDF*15, PICT15, PT16, BK17, WKBB18] introduced in Section 6 are implemented. Hence, the library supports the simulation of low viscous flow and highly viscous materials. Surface tension effects (see Section 7) can also be simulated using SPlisHSPlasH. The framework currently implements microscopic and macroscopic approaches [BT07, AAT13, HWZ*14]. To simulate turbulent fluids, SPlisHSPlasH implements vorticity confinement [MM13] and the micropolar model [BKKW17] which were discussed in Section 8.

Aside from forces which act within the same phase, SPlisHSPlasH also supports multiphase simulations [SP08] and provides functionality to couple different materials. The interaction between air phase and fluid phase is realized using drag forces [MMCK14, GBP*17]. The framework implements the approach of Akinci et al. [AIA*12] to simulate the coupling between rigid bodies and fluids. The required surface sampling of the bodies is performed automatically using a Poisson disk sampling. For the simulation of dynamic rigid bodies SPlisHSPlasH uses the open-source PositionBasedDynamics library [Ben19a]. This library simulates the rigid bodies using a position-based approach [DCB14]. Collisions between the bodies are efficiently detected using signed distance fields [KDBB17] while the contacts are resolved using a projected Gauss-Seidel method [BET14]. Finally, SPlisHSPlasH implements different methods for the simulation of deformable solids [BIT09, PGBT17] using an SPH formulation (see Section 10). Since an SPH formulation is used, the two-way coupling between solids and fluids is simply handled by the implemented multiphase method.

SPlisHSPlasH uses a neighborhood search based on the compact hashing approach of Ihmsen et al. [IABT11]. This approach is discussed in more detail in Section 3. The neighborhood search is implemented in our open-source library CompactNSearch [Kos19].

The SPlisHSPlasH framework has many more features like emitters, adaptive time-stepping or the support of different kernel functions. Moreover, the library has some useful tools like volume sampling of closed geometries or the export of particle data for Maya or Houdini. New simulation scenarios can be created easily using a JSON-based scene file format. Finally, due to a modular concept it is simple to extend the library and to integrate own SPH methods. Therefore, we think that SPlisHSPlasH is a good starting point for all beginners in the area of SPH-based simulations.

14. Conclusion

This tutorial introduced state-of-the-art SPH techniques for the physics based simulation of fluids and solids in graphics and presented practical guidelines for implementations. Various concepts that are particularly relevant for graphics applications were discussed. We showed that with the recent improvements SPH models have matured and ultimately emerged as a competitive alternative to Eulerian fluid simulations or hybrid approaches. Particular challenges of SPH concerning neighborhood search algorithms, pressure solvers, or versatile fluid-solid interaction techniques have been overcome. With the improved robustness and efficiency, millions of particles can today be simulated on a single desktop com-

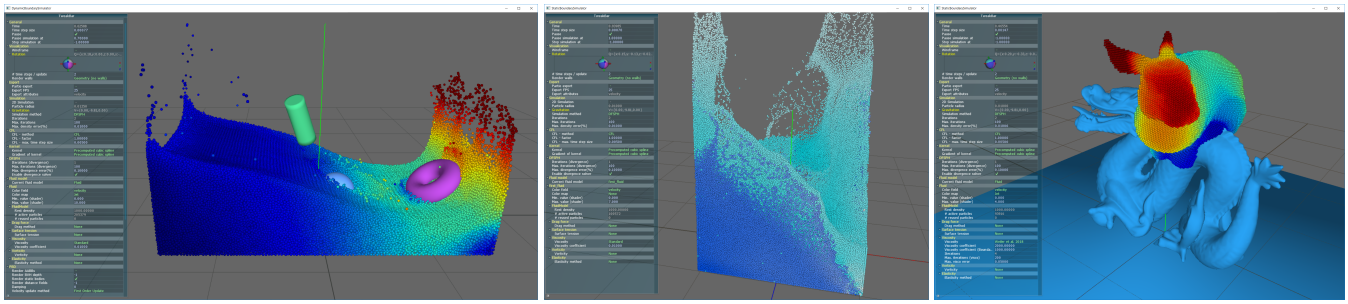


Figure 29: Screenshots of the SPlisHSPlasH fluid simulation framework. Left: Two-way coupling of dynamic rigid bodies and a color-coded fluid. Center: Two-phase double dam break simulation. Right: Highly viscous bunny collides with a static dragon model.

puter. Accordingly, the Lagrangian SPH method reaches an unprecedented level of visual quality, where fine-scale surface effects and flow details are reliably captured.

The SPH community – in graphics as well as in other research disciplines – is very active and the field advances quickly. Each community contributes to different aspects of SPH simulations, and the research often finds applications across disciplines. For graphics applications, it was especially important to efficiently enforce incompressibility on unstructured particles and hence to eliminate the severe time step restrictions of standard SPH techniques. We have presented a practical introduction to various SPH concepts that enforce volume conservation and/or divergence-free velocity fields. A current difficulty is that these approaches render time stepping more challenging, since the largest possible time step does not necessarily result in the best overall performance. Future work is certainly necessary to establish a CFL condition for these methods, as well as to overcome the current main performance bottleneck which is still the time step restriction especially when using millions of particles.

Using large particle numbers is one of the key components for high visual quality and production level results. Such high-resolution simulations pose new challenges and existing concepts might need to be revisited. Especially speed, flexibility and controllability are core aspects, for which solutions are still largely missing. This problem, however, affects not only the SPH field but the entire fluid community in graphics likewise, and has triggered research on pre- and post-processing methods or data-driven approaches.

Our tutorial introduced the SPH-based open-source library SPlisHSPlasH that contains reference implementations of many concepts that we discussed. This implementation is an excellent starting point for students, researchers and practitioners, and may serve as a valuable tool for future research.

References

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics 32*, 6 (2013), 1–8. 25, 26, 27, 37

[Abe12] ABEYARATNE R.: *Continuum Mechanics: Volume II of Lecture Notes on The Mechanics of Elastic Solids*. techreport, MIT Department of Mechanical Engineering, 2012. 7

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics 31*, 4 (July 2012), 1–8. 18, 20, 21, 25, 30, 37

[AMS*07] AGERTZ O., MOORE B., STADEL J., POTTER D., MINIATI F., READ J., MAYER L., GAWRYSZCZAK A., KRAVTSOV A., MONAGHAN J., NORDLUND A., PEARCE F., QUILIS V., RUDD D., SPRINGEL V., STONE J., TASKER E., TEYSSIER R., WADSLEY J., WALDER R.: Fundamental differences between SPH and grid methods. *Mon. Not. R. Astron. Soc. 380*, 3 (2007), 963–978. 29

[AW09] ADAMS B., WICKE M.: Meshless Approximation Methods and Applications in Physics Based Modeling and Animation. In *Proceedings of the Eurographics conference* (2009), EG '09, Eurographics Association, pp. 213–239. 31

[Ben19a] BENDER J.: PositionBasedDynamics Library. https://github.com/InteractiveComputerGraphics/PositionBasedDynamics, 2019. 37

[Ben19b] BENDER J.: SPlisHSPlasH Library. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH, 2019. 25, 37

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 35, 37

[BGFAO17] BARREIRO H., GARCÍA-FERNÁNDEZ I., ALDUÁN I., OTADUY M. A.: Conformation constraints for efficient viscoelastic fluid simulation. *ACM Transactions on Graphics 36*, 6 (2017), 221.1–221.11. 24

[BGI*18] BAND S., GISSLER C., IHMSEN M., CORNELIS J., PEER A., TESCHNER M.: Pressure boundaries for implicit incompressible sph. *ACM Transactions on Graphics 37*, 2 (Feb. 2018), 14:1–14:11. 18

[BGPT18] BAND S., GISSLER C., PEER A., TESCHNER M.: MLS pressure boundaries for divergence-free and viscous SPH fluids. *Computers & Graphics 76* (nov 2018), 37–46. 17, 18

[BIT09] BECKER M., IHMSEN M., TESCHNER M.: Corotated SPH for deformable solids. In *Proceedings of Eurographics Conference on Natural Phenomena* (2009), pp. 27–34. 32, 37

[BK15] BENDER J., KOSCHIER D.: Divergence-Free Smoothed Particle Hydrodynamics. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), pp. 1–9. 17, 37

[BK17] BENDER J., KOSCHIER D.: Divergence-Free SPH for Incompressible and Viscous Fluids. *IEEE Transactions on Visualization and Computer Graphics 23*, 3 (2017), 1193–1206. 9, 17, 18, 24, 25, 26, 37

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-Based Simulation of Continuous Materials. *Computers & Graphics 44*, 1 (2014), 1–10. 31

[BKKW17] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.:

A micropolar material model for turbulent sph fluids. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2017), pp. 1–8. 37

[BKKW18] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.: Turbulent micropolar sph fluids with foam. *IEEE Transactions on Visualization and Computer Graphics* (2018). 27, 28, 29

[BL99] BONET J., LOK T.-S.: Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering 180*, 1 (1999), 97 – 115. 32

[BLS12] BODIN K., LACOURSIÈRE C., SERVIN M.: Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18* (2012), 516–526. 16, 18, 19

[BMM14] BENDER J., MÜLLER M., MACKLIN M.: A Survey on Position-Based Simulation Methods in Computer Graphics. *Computer Graphics Forum 33*, 6 (2014), 228–251. 31

[BMM17] BENDER J., MÜLLER M., MACKLIN M.: A survey on position based dynamics, 2017. In *EUROGRAPHICS 2017 Tutorials* (2017), Eurographics Association. 31

[Bri15] BRIDSON R.: *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis, 2015. 8

[Bro85] BROOKSHAW L.: A method of calculating radiative heat diffusion in particle simulations. *Publications of the Astronomical Society of Australia 6*, 2 (1985), 207–210. 7, 22

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 1–8. 18, 25, 26, 37

[CBG*18] CORNELIS J., BENDER J., GISSLER C., IHMSEN M., TESCHNER M.: An optimized source term formulation for incompressible SPH. *The Visual Computer* (Feb. 2018). 16

[CIPT14] CORNELIS J., IHMSEN M., PEER A., TESCHNER M.: IISPH-FLIP for incompressible fluids. *Computer Graphics Forum 33*, 2 (may 2014), 255–262. 27

[CM99] CLEARY P. W., MONAGHAN J. J.: Conduction modelling using smoothed particle hydrodynamics. *Journal of Computational Physics 148*, 1 (1999), 227 – 264. 22

[Com16a] COMPUTER ANIMATION, RWTH AACHEN UNIVERSITY: Divergence-free sph for incompressible and viscous fluids. www.youtube.com/watch?v=tl4mx0TtaAc, 2016. 1

[Com16b] COMPUTER GRAPHICS, UNIVERSITY OF FREIBURG: Terrain 2 - up to 500 million particles with PreonLab (FIFTY2). www.youtube.com/watch?v=4y-VBLzA9Mw, 2016. 1

[Com17] COMPUTER GRAPHICS, UNIVERSITY OF FREIBURG: Ship under attack. www.youtube.com/watch?v=_O6fqLOCTew, 2017. 1

[Com18] COMPUTER GRAPHICS, UNIVERSITY OF FREIBURG: An implicit SPH formulation for incompressible linearly elastic solids. www.youtube.com/watch?v=qd3gKVX89qo, 2018. 1

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds 27*, 2 (Sept. 2014), 103–112. 37

[dGWH*15] DE GOES F., WALLEZ C., HUANG J., PAVLOV D., DESBRUN M.: Power Particles: An incompressible fluid solver based on power diagrams. *ACM Transactions on Graphics 34*, 4 (2015), 50:1–50:11. 27

[ER03] ESPANOL P., REVENGA M.: Smoothed dissipative particle dynamics. *Physical Review E 67*, 2 (2003), 026705. 22

[FAW17] FÜRSTENAU J.-P., AVCI B., WRIGGERS P.: A comparative numerical study of pressure-Poisson-equation discretization strategies for SPH. In *12th International SPHERIC Workshop* (2017). 16

[FIF16] FIFTY2 TECHNOLOGY: PreonLab Promotion. www.youtube.com/watch?v=giS2r5JPgy0, 2016. 1

[FLR*13] FERRAND M., LAURENCE D. R., ROGERS B. D., VIOLEAU D., KASSIOTIS C.: Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *International Journal for Numerical Methods in Fluids 71*, 4 (Feb. 2013), 446–472. 19

[FM15] FUJISAWA M., MIURA K. T.: An Efficient Boundary Handling with a Modified Density Calculation for SPH. *Computer Graphics Forum 34*, 7 (2015), 155–162. 19

[FMH*94] FLEBBE O., MUENZEL S., HEROLD H., RIFFERT H., RUDER H.: Smoothed Particle Hydrodynamics: Physical viscosity and the simulation of accretion disks. *The Astrophysical Journal 431* (Aug. 1994), 754–760. 22

[GAC*09] GRENIER N., ANTUONO M., COLAGROSSI A., TOUZÉ D. L., ALESSANDRINI B.: An Hamiltonian interface SPH formulation for multi-fluid and free surface flows. *Journal of Computational Physics 228*, 22 (2009), 8380 – 8393. 30

[Gan15] GANZENMÜLLER G. C.: An hourglass control algorithm for lagrangian smooth particle hydrodynamics. *Computer Methods in Applied Mechanics and Engineering 286* (apr 2015), 87–106. 32, 33

[GBP*17] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Generalized drag force for particle-based simulations. *Computers & Graphics 69* (dec 2017), 1–11. 37

[GM77] GINGOLD R. A., MONAGHAN J.: Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars. *Monthly Notices of the Royal Astronomical Society*, 181 (1977), 375–389. 4

[GPB*19] GISSLER C., PEER A., BAND S., BENDER J., TESCHNER M.: Interlinked sph pressure solvers for strong fluid-rigid coupling. *ACM Transactions on Graphics 38*, 1 (Jan. 2019), 5:1–5:13. 18, 30, 33, 35

[HA06] HU X., ADAMS N.: A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics 213*, 2 (2006), 844–861. 29, 30

[HKK07a] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics in complex shapes. In *Spring Conference on Computer Graphics* (2007), pp. 191–197. 18, 19

[HKK07b] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed Particle Hydrodynamics on GPUs. In *Computer Graphics International* (2007), pp. 63–70. 18, 19

[HLL*12] HE X., LIU N., LI S., WANG H., WANG G.: Local Poisson SPH for Viscous Incompressible Fluids. *Computer Graphics Forum 31* (2012), 1948–1958. 16

[Hoo98] HOOVER W.: Isomorphism linking smooth particles and embedded atoms. *Physica A: Statistical Mechanics and its Applications 260*, 3 (1998), 244–254. 29

[HS13] HORVATH C. J., SOLENTHALER B.: Mass preserving multi-scale SPH. Pixar Technical Memo 13-04, Pixar Animation Studios, 2013. 30

[HWZ*14] HE X., WANG H., ZHANG F., WANG H., WANG G., ZHOU K.: Robust Simulation of Sparsely Sampled Thin Features in SPH-Based Free Surface Flows. *ACM Transactions on Graphics 34*, 1 (2014), 7:1–7:9. 25, 37

[IAAT12] IHMSEN M., AKINCI N., AKINCI G., TESCHNER M.: Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer 28*, 6-8 (2012), 669–677. 9

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A Parallel SPH Implementation on Multi-Core CPUs. *Computer Graphics Forum 30*, 1 (Mar. 2011), 99–112. 10, 11, 37

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for PCISPH. In *Virtual Reality Interactions and Physical Simulations* (2010), pp. 79–88. 18

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. 9, 12, 13, 14, 16, 18, 37

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH Fluids in Computer Graphics. *Eurographics (State of the Art Reports)* (2014), 21–42. 18, 22, 27

[JSD04] JUBELGAS M., SPRINGEL V., DOLAG K.: Thermal conduction in cosmological SPH simulations. *Monthly Notices of the Royal Astronomical Society 351*, 2 (2004), 423–435. 22

[KAT*19] KIM B., AZEVEDO V., THUEREY N., GROSS M., SOLENTHALER B.: Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum* (2019). 36

[KB17] KOSCHIER D., BENDER J.: Density maps for improved sph boundary handling. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (July 2017), pp. 1–10. 18, 19

[KBT17] KOSCHIER D., BENDER J., THUEREY N.: Robust eXtended Finite Elements for Complex Cutting of Deformables. *ACM Transactions on Graphics 36*, 4 (2017), 55:1–55:13. 31

[KDBB17] KOSCHIER D., DEUL C., BRAND M., BENDER J.: An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Transactions on Visualization and Computer Graphics 23*, 10 (2017), 2208–2221. 18, 37

[KKB18] KUGELSTADT T., KOSCHIER D., BENDER J.: Fast corotated FEM using operator splitting. *Computer Graphics Forum 37*, 8 (2018). 31, 32

[Kos19] KOSCHIER D.: CompactNSearch Library. https://github.com/InteractiveComputerGraphics/CompactNSearch, 2019. 10, 37

[Lau11] LAUTRUP B.: *Physics of Continuous Matter*. Taylor & Francis, 2011. 22

[LJS*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics 34*, 6 (Oct. 2015), 199:1–199:9. 35, 36

[LKR09] LAI M., KREMPL E., RUBEN D.: *Introduction to Continuum Mechanics*. Butterworth-Heinemann, 2009. 7

[LL10] LIU M., LIU G.: Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Archives of Computational Methods in Engineering 17*, 1 (2010), 25–76. 3, 4

[LLP11] LIU S., LIU Q., PENG Q.: Realistic simulation of mixing fluids. *The Visual Computer 27*, 3 (2011), 241–248. 30

[Łuk99] ŁUKASZEWICZ G.: *Micropolar Fluids*. Modeling and Simulation in Science, Engineering and Technology. Birkhäuser Boston, 1999. 27

[MBCM16] MÜLLER M., BENDER J., CHENTANEZ N., MACKLIN M.: A robust method to extract the rotational part of deformations. In *Proceedings of ACM SIGGRAPH Conference on Motion in Games* (2016), MIG '16, ACM. 32

[MFK*15] MAYRHOFER A., FERRAND M., KASSIOTIS C., VIOLEAU D., MOREL F.-X.: Unified semi-analytical wall boundary conditions in SPH: analytical extension to 3-D. *Numerical Algorithms 68*, 1 (Jan. 2015), 15–34. 19

[Mir96] MIRTICH B. V.: *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California at Berkeley, 1996. 35

[MM13] MACKLIN M., MÜLLER M.: Position Based Fluids. *ACM Transactions on Graphics 32*, 4 (2013), 1–5. 16, 27, 36, 37

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified Particle Physics for Real-Time Applications. *ACM Transactions on Graphics 33*, 4 (2014), 1–12. 37

[Mon92] MONAGHAN J.: Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 9, 22, 24, 37

[Mon05] MONAGHAN J. J.: Smoothed Particle Hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703–1759. 2, 22

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), p. 237. 29, 30

[Nex17] NEXTLIMIT: RealFlow Showreel 2017. www.youtube.com/watch?v=nnv-95w1d5A, 2017. 1

[PGBT17] PEER A., GISSLER C., BAND S., TESCHNER M.: An implicit sph formulation for incompressible linearly elastic solids. *Computer Graphics Forum* (2017), n/a–n/a. 32, 33, 37

[PICT15] PEER A., IHMSEN M., CORNELIS J., TESCHNER M.: An Implicit Viscosity Formulation for SPH Fluids. *ACM Transactions on Graphics 34*, 4 (2015), 1–10. 22, 23, 25, 37

[Pri12] PRICE D. J.: Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics 231*, 3 (Feb. 2012), 759–794. 2, 5, 6, 7, 22

[PT16] PEER A., TESCHNER M.: Prescribed velocity gradients for highly viscous SPH fluids with vorticity diffusion. *IEEE Transactions on Visualization and Computer Graphics* (2016), 1–9. 23, 25, 26, 37

[RL96] RANDLES P., LIBERSKY L.: Smoothed particle hydrodynamics: Some recent improvements and applications. *Computer Methods in Applied Mechanics and Engineering 139*, 1 (1996), 375 – 408. 5

[RLY*14] REN B., LI C., YAN X., LIN M. C., BONET J., HU S.-M.: Multiple-Fluid SPH Simulation Using a Mixture Model. *ACM Transactions on Graphics 33*, 5 (2014), 1–11. 31

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics 31*, 4 (2012), 61:1–61:8. 9, 22, 30, 37

[SF18] SCHENCK C., FOX D.: Spnets: Differentiable fluid dynamics for deep neural networks. In *CoRL* (2018), vol. 87 of *Proceedings of Machine Learning Research*, PMLR, pp. 317–335. 36

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *TOG 30*, 4 (2011), 72:1–72:8. 30

[Sif12] SIFAKIS E.: *SIGGRAPH 2012 Course Notes FEM Simulation of 3D Deformable Solids Part 1*. Tech. rep., University of Wisconsin-Madison, 2012. 31

[SP08] SOLENTHALER B., PAJAROLA R.: Density Contrast SPH Interfaces. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 211–218. 29, 30, 37

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (2009), 40:1–40:6. 9, 14, 18, 37

[TDF*15] TAKAHASHI T., DOBASHI Y., FUJISHIRO I., NISHITA T., LIN M.: Implicit Formulation for SPH-based Viscous Fluids. *Computer Graphics Forum 34*, 2 (2015), 493–502. 22, 23, 24, 25, 26, 37

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANTES D., GROSS M. H.: Optimized spatial hashing for collision detection of deformable objects. In *Vmv* (2003), vol. 3, pp. 47–54. 10

[TM05] TARTAKOVSKY A., MEAKIN P.: Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E 72*, 2 (2005), 026301. 29, 30

[TSSP16] TOMPSON J., SCHLACHTER K., SPRECHMANN P., PERLIN K.: Accelerating Eulerian Fluid Simulation With Convolutional Networks, jul 2016. 36

[UHT18] UM K., HU X., THUEREY N.: Liquid splash modeling with neural networks. *CGF 37*, 8 (2018), 171–182. 35

[WBF*96] WATKINS S. J., BHATTAL A. S., FRANCIS N., TURNER J. A., WHITWORTH A. P.: A new prescription for viscosity in smoothed particle hydrodynamics. *Astron. Astrophys. Suppl. Ser. 119*, 1 (1996), 177–187. 22

[WBT18] WIEWEL S., BECHER M., THUEREY N.: Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow, feb 2018. 36

[WKB16] WEILER M., KOSCHIER D., BENDER J.: Projective fluids. In *ACM Motion in Games* (2016), pp. 1–6. 37

[WKBB18] WEILER M., KOSCHIER D., BRAND M., BENDER J.: A physically consistent implicit viscosity solver for sph fluids. *Computer Graphics Forum 37*, 2 (2018). 22, 23, 24, 25, 26, 37

[XFCT18] XIE Y., FRANZ E., CHU M., THUEREY N.: tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *TOG 37*, 4 (2018), 95. 36

[YCR*15] YANG T., CHANG J., REN B., LIN M. C., ZHANG J. J., HU S.-M.: Fast multiple-fluid simulation using helmholtz free energy. *ACM Transactions on Graphics 34*, 6 (Oct. 2015), 201:1–201:11. 31

[YJL*16] YAN X., JIANG Y.-T., LI C.-F., MARTIN R. R., HU S.-M.: Multiphase sph simulation for interactive fluids and solids. *ACM Transactions on Graphics 35*, 4 (July 2016), 79:1–79:11. 31