EUROGRAPHICS 2022
D. Meneveaux and G. Patanè
(Guest Editors)

Volume 41 (2022), Number 2
STAR – State of The Art Report

A Survey on SPH Methods in Computer Graphics

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

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

Abstract

Throughout the past decades, the graphics community has spent major resources on the research and development of physics simulators on the mission to computer-generate behaviors achieving outstanding visual effects or to make the virtual world indistinguishable from reality. The variety and impact of recent research based on Smoothed Particle Hydrodynamics (SPH) demonstrates the concept's importance as one of the most versatile tools for the simulation of fluids and solids. With this survey, we offer an overview of the developments and still-active research on physics simulation methodologies based on SPH that has not been addressed in previous SPH surveys. Following an introduction about typical SPH discretization techniques, we provide an overview over the most used incompressibility solvers and present novel insights regarding their relation and conditional equivalence. The survey further covers recent advances in implicit and particle-based boundary handling and sampling techniques. While SPH is best known in the context of fluid simulation we discuss modern concepts to augment the range of simulatable physical characteristics including turbulence, highly viscous matter, deformable solids, as well as rigid body contact handling. Besides the purely numerical approaches, simulation techniques aided by machine learning are on the rise. Thus, the survey discusses recent data-driven approaches and the impact of differentiable solvers on artist control. Finally, we provide context for discussion by outlining existing problems and opportunities to open up new research directions.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

An increasing number of emerging technologies, *e.g.*, computer games engines, special-effects creator tools for the film industry, virtual reality or even engineering applications, rely on realistic virtual worlds. The goal of bringing these virtual environments to life, has been a driving force for the research and development of physics-based simulators. Within this research topic, the concept of Smoothed Particle Hydrodynamics (SPH) has become a popular tool for the simulation of fluids and solids and the major research advances in the past decade highlight its relevance. Due to the Lagrangian nature of SPH-based approaches, there are many examples where they naturally handle complex scenarios where traditional Eulerian grid-based approaches struggle. 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 mechanical assemblies with optimized lubrication. Thus, we expect this research direction to be further pursued; not only by the computer graphics community but also to advance the state-of-the-art in engineering or physics applications.

Since the last survey on SPH methods of Ihmsen et al. [IOS*14], almost a decade has passed and a forest of new research has been published. As incompressibility solvers have been a core part of their survey, we would like to stress the fact that new solvers have been proposed and that new insights regarding their similarities and conditional equivalence have been gained as presented in our new survey. It should also be mentioned that their work was purely focused on low-viscous fluid simulation with particle-based boundary handling while a large body of work outside that core topic has emerged. Since then, a multitude of new approaches for the simulation of a wider variety of materials, *e.g.*, highly-viscous fluids or deformable solids, new coupling techniques between those or new concepts to preserve turbulence has been published and even a whole new research direction has been opened up with data-driven and machine-learning aided techniques. Nevertheless, Ihmsen et al.'s work covers a few topics peripheral to core modeling and simulation techniques which are not captured in this survey. Therefore, we would like to refer the reader to their work for an overview

over neighborhood search algorithms, surface reconstruction and rendering.

This survey is a general overview over recent research advancing the state-of-the-art of SPH simulators. To provide a comprehensive overview, we have structured this work as follows. In Section 2 we give an introduction to the fundamentals of the SPH formalism as a spatial discretization technique for field quantities and differential operators, describe a general form of the governing equations for fluid and solid simulation, discuss the core idea of operator splitting and typical time integration schemes. Section 3 gives an overview over techniques to enforce incompressibility and discusses their formal relation and conditional equivalence. Boundary handling approaches, including particle-based and implicit boundary representations as well as boundary pressure computation methods, are presented in Section 4. We further discuss approaches to simulate various materials specific to either solids or fluids as well as coupling between both, *i.e.*, viscosity solvers, deformable solids and respective stability improvements, rigid body coupling and handling multiple phases, in Section 5. Methods to improve turbulence and maintain vorticity are presented in Section 6. Section 7 discusses data-driven and machine-learning aided SPH simulation techniques as a newly emerged research field which received increased attention in recent years. Finally, we conclude this survey with a discussion by outlining remaining problems and opportunities to open up new research directions.

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 works of Price [Pri12] and Monaghan [Mon05] and includes important insights that have been gained over the years.

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 in relation to physics-based simulations targeting computer graphics applications. Thirdly, we show how 1^{st}- and 2^{nd}-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.

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*. To outline the basic idea, we first 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 $\int \delta(\mathbf{r})\, dv = 1$. The Dirac-$\delta$ 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

Due to the fact that $\delta(\mathbf{r})$ is a distribution and therefore can not be directly discretized, a continuous approximation to the Dirac-δ distribution is made 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. The Gaussian is, however, not an optimal choice due to its non-compact support domain and we 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

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

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 d denotes the spatial dimension and $\hbar$ 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. 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 to make equation systems arising in the discretization sparse such that only particles that are supported at a point of field evaluation need to be considered which significantly improves the performance of simulators. In order to find those supported neighboring particles, algorithms for so-called *neighborhood searches* are

employed. Please, see the report of Ihmsen et al. [IOS*14] for a detailed discussion on those.

For the sake of brevity, 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.

2.3. Discretization

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

$$(A*W)(\mathbf{x}_i) = \int A(\mathbf{x}')W(\mathbf{x}_i-\mathbf{x}',h)dv' \approx \sum_{j\in\mathcal{F}} A_j \frac{m_j}{\rho_j} W(\mathbf{x}_i-\mathbf{x}_j,h), \tag{4}$$

where m is the mass, ρ is the density, and $\mathcal{F}$ 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 report. 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.

In general the discretization is not guaranteed to be 0^{th}-order consistent. However, 0^{th}-order consistency can be easily restored by normalizing the SPH approximation with $\sum_j \frac{m_j}{\rho_j} W_{ij}$. Even 1^{st}-order consistency can be restored by the cost of a small matrix inversion (see [Pri12]). Besides those mitigations, 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-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. (4) results in

$$\rho_i = \sum_j m_j W_{ij} \tag{5}$$

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. (5) but the reconstructed density is typically underestimated at the free surface due to particle deficiency. This must be considered when implementing a pressure solver as discussed in Section 3.

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 document, we will assume that the smoothing length h is constant in space (and time). Based on the discrete SPH approximation in Eq. (4) the gradient of the underlying field can be approximated straightforwardly using

$$\nabla A_i \approx \sum_j A_j \frac{m_j}{\rho_j} \nabla_i W_{ij}. \tag{6}$$

In order to improve readability, we will drop the differentiation index for differential operators in the remainder of this report. 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.*, $\nabla_i W_{ij} \equiv \nabla W_{ij}$.

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

$$\nabla \mathbf{A}_i \approx \sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \otimes \nabla W_{ij} \tag{7}$$

$$\nabla\cdot\mathbf{A}_i \approx \sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \cdot \nabla W_{ij} \tag{8}$$

$$\nabla\times\mathbf{A}_i \approx -\sum_j \frac{m_j}{\rho_j} \mathbf{A}_j \times \nabla W_{ij}, \tag{9}$$

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 [Pri12]. For this reason many discrete differential operators have emerged over time.

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

Difference Formula As the direct derivative of the discretization is not guaranteed to be 0^{th}-order consistent, the idea of the *difference formula* is to subtract the first error term of the Taylor series recovering consistency, i.e.

$$\nabla A_i \approx \sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla W_{ij}. \tag{10}$$

The same formula can be straightforwardly applied to the higher-dimensional first-order differential operators presented in Eqs. (7) to (9). 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 as described in [Pri12], *i.e.*,

$$\nabla A_i \approx \mathbf{L}_i \left(\sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla W_{ij} \right)$$
$$\mathbf{L}_i = \left(\sum_j \frac{m_j}{\rho_j} \nabla W_{ij} \otimes (\mathbf{x}_j - \mathbf{x}_i) \right)^{-1} . \tag{11}$$

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

$$\nabla A_i \approx \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij}, \tag{12}$$

which is not designed to recover a certain order of consistency. However, the 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 the discretization of certain physical quantities, *e.g.*, forces or impulses.

2.5.1. Discretization of the Laplace Operator

Similar to the direct 1st-order derivatives (Eqs. (7)-(9)) the Laplace operator can be directly discretized, *i.e.*,

$$\nabla^2 A_i \approx \sum_j \frac{m_j}{\rho_j} A_j \nabla_i^2 W_{ij} . \tag{13}$$

As discussed by Brookshaw [Bro85], this leads again to a very poor estimate of the 2nd-order differential and, thus, they presented an improved discrete operator for the Laplacian:

$$\nabla^2 A_i \approx -\sum_j \frac{m_j}{\rho_j} A_{ij} \frac{2\|\nabla W_{ij}\|}{\|\mathbf{r}_{ij}\|} . \tag{14}$$

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

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

$$\nabla^2 \mathbf{A}_i = -\sum_j \frac{m_j}{\rho_j} \mathbf{A}_{ij} \frac{2\|\nabla W_{ij}\|}{\|\mathbf{r}_{ij}\|} \tag{15}$$

$$\nabla(\nabla \cdot \mathbf{A}_i) = \sum_j \frac{m_j}{\rho_j} \left[(d+2)(\mathbf{A}_{ij} \cdot \tilde{\mathbf{r}}_{ij}) \tilde{\mathbf{r}}_{ij} - \mathbf{A}_{ij} \right] \frac{\|\nabla W_{ij}\|}{\|\mathbf{r}_{ij}\|} , \tag{16}$$

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. (15) in the context of physics simulations is that forces derived using this operator, *e.g.*, viscosity forces, are not momentum conserving. Fortunately, in the case of divergence-free vector fields, *i.e.*, $\nabla \cdot \mathbf{A} = 0$, the Laplace operator can be discretized by adding together and transforming Eqs. (15) and (16) resulting in

$$\nabla^2 \mathbf{A}_i \approx 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{A}_{ij} \cdot \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^2 + 0.01h^2} \nabla W_{ij} , \tag{17}$$

where a regularization term $0.01h^2$ is introduced in the denominator. Thus, derived forces consist of terms that solely act along the "line of sight" between two interacting particles i and j such that momentum conservation is recovered (see [Pri12]). Therefore, we recommend to use Eq. (17) as discrete Laplace operator for divergence-free vector fields.

2.6. Governing Equations for Fluids and Solids

In this section, we will summarize the most important local conservation laws required for the numerical simulation of (in)compressible fluids and deformable solids.

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

$$\frac{D\rho}{Dt} = -\rho (\nabla \cdot \mathbf{v}) , \tag{18}$$

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

$$\frac{D\rho}{Dt} = 0 \quad \Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0 \tag{19}$$

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

Please note that 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. While the derivative expressed in *Eulerian coordinates* contains the convection term $\mathbf{v} \cdot \nabla_{\mathbf{x}}(\cdot)$, the term vanishes when using *Lagrangian coordinates*. For a detailed explanation we would like to refer the reader to the work of Bridson [Bri15]. In the remainder of this report, we will exclusively describe quantities using Lagrangian coordinates.

2.6.1. Conservation Law of Linear Momentum and the Navier-Stokes Equation

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.*,

$$\rho \frac{D^2 \mathbf{x}}{Dt^2} = \nabla \cdot \mathbf{T} + \mathbf{f}_{\text{ext}} , \tag{20}$$

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.

A typical constitutive relation for incompressible flow is

$$\mathbf{T} = -p\mathbb{1} + \mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T) , \tag{21}$$

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. (19) is fulfilled. Plugging Eq. (21) into Eq. (20) yields the incompressible Navier-Stokes equation

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}_{\text{ext}}\,. \tag{22}$$

For modeling deformable solids constitutive relations that typically model the stress tensor as a function of strain (and strain-rate for visco-elasticity) are required. Sec. 5.2 discusses the definition of such constitutive models in more detail.

2.7. Mixed Initial-Boundary Value Problem

The previously introduced linear momentum conservation law (Eq. (20)) in combination with a constitutive relation, *e.g.*, Eq. (21), 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 associated 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*.

2.8. Operator Splitting

The basic idea of operator splitting is to decompose the underlying PDE, *e.g.*, the Navier-Stokes equation, 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. For a more detailed discussion on operator splitting, we would like to refer the reader to works of Bridson [Bri15] and Ihmsen et al. [IOS*14].

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.*, employed in [IAAT12, SB12, ICS*14, BK17]. 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 5.1.

Naturally, we aim for the best performance of our simulator and, therefore, try to use a very large time step width. 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 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.*, $\Delta t \leq \lambda \frac{\tilde{h}}{\|\mathbf{v}^{\max}\|}$, where $\tilde{h}$, $\mathbf{v}^{\max}$, and λ denote the particle diameter, 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].

3. Incompressibility

Most fluids we encounter in the real world are nearly incompressible, *e.g.*, water. Besides this fact, incompressibility in fluids is facilitating the occurrence of interesting visual phenomena such as the formation of waves and ripples on a fluid surface and therefore contributes to the realism of fluid simulations. To model a material which withstands compression, a constitutive relation $\mathbf{T} = -p\mathbb{1}$ (*cf.*, Eq. (21)) is introduced which applies hydrostatic pressure p in order to withstand compression. The scalar pressure is typically determined using one of the following two approaches.

In the *first approach*, incompressibility is weakly enforced by penalizing volume deviations dependent on changes in the mass density, *i.e.*, $p = p(\rho)$. A typical choice is the linear relation $p = k(\rho - \rho^0)$, where k denotes a stiffness constant. Finally, the specific pressure acceleration field is defined by $\mathbf{a}^{\text{p}} = -\frac{1}{\rho}\nabla p$ (*cf.*, Eq. (22)).

In the *second approach* the incompressibility constraint in Eq. (19) is strongly enforced by solving an equation system to compute the pressure field that leads to accelerations satisfying the constraint. Most solvers of the second type follow a predictor-corrector scheme. They predict a velocity $\mathbf{v}^* = \mathbf{v} + \Delta t\mathbf{g} + \Delta t\nu\nabla^2\mathbf{v} = \mathbf{v} + \Delta t\mathbf{a}^{\text{nonp}}$ for each particle after applying only non-pressure accelerations $\mathbf{a}^{\text{nonp}}$. Then, pressure accelerations $\mathbf{a}^{\text{p}}$ are computed to adapt the predicted velocities using $\mathbf{v} = \mathbf{v}^* + \Delta t\mathbf{a}^{\text{p}}$ such that all particles are advected into a state that minimizes volume deviations

and therefore enforces incompressibility. As the pressure accelerations are functions of the pressure, *i.e.*, $\mathbf{a}^p = -\frac{1}{\rho}\nabla p$, this results in a linear equation system in the unknown pressure values to finally derive the incompressibility-enforcing pressure acceleration field $\mathbf{a}^p$.

The first of the two introduced concepts computes particle pressure locally and independently at each individual particle. Such variants can be referred to as *local pressure solvers*, *e.g.*, [HLL*12, BGPT18]. In contrast, variants of the second option conceptually solve a system in the pressure computation. Therefore, such implementations can be referred to as *global pressure solvers*, *e.g.*, [HLL*12, BGI*18]. In the following, we start with a discussion of how to quantify the compression of a particle and then we continue with local and global pressure solvers.

3.1. Measuring Compression

A compression measure is an integral part of all pressure solvers as it indicates the error in the incompressibility constraint. Typical choices to quantify compression are to measure the difference between either the particle's current volume and uncompressed rest volume or current density ρ and rest density ρ^0. There is no practical or conceptual difference between the two options. Nevertheless, the vast majority of particle fluid solvers employs the density formulation. A rare case, where the volume formulation is used instead is the work of Band et al. [BGI*18]. As an alternative to volumes or densities, compression can also be quantified using the time derivative of the density or volume. Starting from an uncompressed state with $\rho = \rho^0$, incompressibility can be preserved over time, if the time rate of change in the density vanishes, *i.e.*, $\frac{D\rho}{Dt} = 0$. Hence, the derivative can be directly used to quantify the rate of compression. Analogously, we can use the continuity equation (Eq. (18)) to instead express the rate of compression with $\nabla \cdot \mathbf{v} = 0$. To summarize, the compression (rate) can be quantified with either $\rho - \rho^0$, $\frac{D\rho}{Dt}$ or $\nabla \cdot \mathbf{v}$. If any of the terms is not zero, the incompressibility constraint is violated.

Generally, all three terms can be used in combination with all pressure solvers. However, the terms have different effects on the average density error of the discretized fluid, which is generally not independent from the pressure solver. If the density deviation $\rho - \rho^0$ is used, then the density oscillates about the rest density. This oscillation is often visible at the free fluid surface when local solvers are used. When using global solvers, the oscillations are significantly reduced and typically imperceptible, but still occur. When $\frac{D\rho}{Dt}$ or the velocity divergence $\nabla \cdot \mathbf{v}$ is used, then the particle density drifts and typically grows over time. Even when a perfect divergence-free velocity field is maintained, particle advection due to numerical time integration still introduces a density deviation. Unfortunately, these deviations stay undetected, when using the velocity divergence as the sole compression measure. Hence, dependent on the compression measure, the goal is to either minimize density/volume oscillations or to minimize density/volume drift over time.

Dependent on the formulation of a fluid solver, it can be easier or more involved or more or less accurate to work with a certain compression measure. Grid fluid solvers, *e.g.*, often do not have an explicit notion of the density in a grid cell. It would also be involved to explicitly compute volume changes of a fluid portion in a static cell with constant volume. Therefore, grid approaches quite generally use the velocity divergence to quantify compression. In contrast, particle approaches are more flexible. In an SPH discretization the density deviation can be approximated by $\rho_i - \rho^0 = \sum_j m_j W_{ij} - \rho^0$ and the velocity divergence or the derivative of the density can be approximated by $\nabla \cdot \mathbf{v}_i = -\frac{1}{\rho_i}\frac{D\rho}{Dt} = -\frac{1}{\rho}\sum_j m_j(\mathbf{v}_i - \mathbf{v}_j)\nabla W_{ij}$. It is also interesting to note that grid approaches often state the incompressibility constraint as $\nabla \cdot \mathbf{v} = 0$, while particle approaches typically assume $\rho - \rho^0 = 0$.

3.2. Local Pressure Solvers

Positive pressure is generally proportional to the compression at a particle and the so-called pressure acceleration $\mathbf{a} = -\frac{1}{\rho}\nabla p$ accelerates particles into the direction of the negative pressure gradient, *i.e.*, from regions with higher to regions with lower pressure. Local pressure solvers typically use a state-equation to compute the pressure at a particle which is solely dependent on their immediately neighboring particles making the computation local, *e.g.*, [BT07]. Therefore, the computation for the pressure acceleration can be performed independently for all particles without the requirement to solve an equation system.

Local pressure solvers are commonly referred to as compressible or weakly compressible approaches, *e.g.*, [CR99, BT07], while global pressure solvers are generally referred to as incompressible, *e.g.*, . [CR99, ICS*14]. The differences between these approaches, however, are not as clear as the terms suggest. As discussed in, *e.g.*, [BT07], local pressure formulations result in dynamically changing densities that oscillate around the rest density. This property, however, also applies to a lesser extend to global pressure solvers that take the density deviation to quantify compression as discussed in, *e.g.*, [ICS*14]. The main benefit of global pressure formulations lies in the stable handling of challenging scenarios with imperceptible density deviations.

3.3. Global Pressure Solvers

This section covers four popular solver variants, namely IISPH, PCISPH, PBF, and DFSPH. We discuss the governing concepts and we further show the interesting fact that the implementations of the four solvers can be transformed into each other. We therefore start with a discussion of the pressure Poisson equation (PPE) and its discretization, followed by IISPH which solves a discretized PPE with relaxed Jacobi. Then, we show that the implementation of PCISPH is equivalent to IISPH. After that, we show the equivalence of PBF and PCISPH and finally, we show that each of the two solvers in DFSPH is also equivalent to PCISPH. So, IISPH, PCISPH, PBF and DFSPH are Jacobi PPE solvers. The section concludes with a discussion of implementation details of each solver and further aspects that might be responsible for the significant performance differences presented in the original publications.

Governing equation: As outlined in the introduction, global pressure solvers first determine a predicted velocity $\mathbf{v}^* = \mathbf{v} + \Delta t \mathbf{a}^{nonp}$, where all accelerations except the pressure acceleration have been applied. If a particle would be advected with this velocity, its predicted density would be $\rho^* = \rho - \Delta t \rho \nabla \cdot \mathbf{v}^*$. Now, the

goal of a global pressure solver is the computation of a pressure p that causes a pressure acceleration $\mathbf{a}^{\mathrm{p}} = -\frac{1}{\rho}\nabla p$ which corresponds to a velocity change $-\Delta t \frac{1}{\rho}\nabla p$ whose divergence $-\nabla\cdot(\Delta t \frac{1}{\rho}\nabla p)$ is equal to a density change per time $-\rho\nabla\cdot(\Delta t \frac{1}{\rho}\nabla p)$ that cancels the predicted density deviation per time $\frac{\rho^0-\rho^*}{\Delta t}$, *i.e.*, $\frac{\rho^0-\rho^*}{\Delta t} - \rho\nabla\cdot(\Delta t \frac{1}{\rho}\nabla p) = 0$. This is one form of a PPE, typically written as

$$\Delta t \nabla^2 p = \frac{\rho^0 - \rho^*}{\Delta t}. \tag{23}$$

The predicted density ρ^* can be estimated in various ways, *e.g.*, using predicted particle positions $\mathbf{x}^*$ or the divergence of the predicted velocity can be used:

$$\Delta t \nabla^2 p = \frac{\rho^0 - (\rho - \Delta t \rho \nabla\cdot\mathbf{v}^*)}{\Delta t}. \tag{24}$$

Alternatively, just the divergence of the predicted velocity can be used

$$\Delta t \nabla^2 p = \rho \nabla\cdot\mathbf{v}^*. \tag{25}$$

All three forms are used in particle simulations and they mainly differ in the behavior of the density error over time as already discussed for local pressure solvers (see *e.g.*, [CBG*18]). Eqs. (23) and (24) cause density oscillations, while Eq. (25) causes a density drift. Grid approaches are often limited to Eq. (25) as they often cannot explicitly compute the density ρ or the predicted density ρ^*.

Linear system: A PPE is considered for each particle. The terms on the right-hand side are either known, *e.g.*, ρ^0 and Δt, or they can be computed with SPH approximations, *e.g.*, ρ, ρ^*, $\mathbf{v}^*$ and $\nabla\cdot\mathbf{v}^*$. For a particle i, the respective SPH implementation is referred to as source term s_i and the set of all source terms is referred to as source vector $\mathbf{s}$. The pressure Laplacian on the left-hand side $\nabla^2 p$ is also expressed with SPH approximations and the set of all these discretizations can be written as a matrix-vector product $\mathbf{Ap}$, where $\mathbf{A}$ is referred to as the discretized Laplacian operator and $\mathbf{p}$ is the solution vector that consists of all unknown pressure values p_i. So, any continuous form of the PPE, *e.g.*, Eqs. (23), (24) or (25), considered at all particles can be approximately represented as a linear system $\mathbf{Ap} = \mathbf{s}$ with unknown pressure values $\mathbf{p}$. The system matrix $\mathbf{A}$ and the source vector $\mathbf{s}$ are constructed from SPH approximations of the respective terms. There do not only exist different forms of the PPE, but each form can also be discretized with different SPH formulations.

Relaxed Jacobi: Relaxed Jacobi is a popular alternative to solve PPEs. There exist various ways to denote a solver iteration with relaxed Jacobi, but

$$p_i^{l+1} = p_i^l + \alpha_i \frac{s_i - (\mathbf{Ap}^l)_i}{a_{ii}} \tag{26}$$

is the form that is closest to a typical implementation of a PPE solver. The solver iteration is indicated with l, α_i is a user-defined relaxation factor and a_{ii} denotes a diagonal element of $\mathbf{A}$.

Various pressure solvers clamp negative pressures. This indicates a close relation to linear complementary problems and it is certainly one reason, why more efficient solvers, *e.g.*, the conjugate gradient method, are rarely considered in the context of PPEs.

IISPH background: IISPH [ICS*14] is a global pressure solver that works with a transformed version of Eq. (24):

$$\Delta t^2 \nabla^2 p_i = \rho^0 - (\rho_i - \Delta t \rho_i \nabla\cdot\mathbf{v}_i^*). \tag{27}$$

The equations are discretized with SPH and the resulting system $\mathbf{Ap} = \mathbf{s}$ is solved with relaxed Jacobi. IISPH does not explicitly generate the matrix $\mathbf{A}$, as this matrix is not required in the Jacobi solver. Instead, the Jacobi update requires the product $\mathbf{Ap}^l$ and this vector is computed in each solver iteration l with two loops over particles i. A first loop computes pressure accelerations

$$(\mathbf{a}_i^{\mathrm{p}})^l = -\frac{1}{\rho_i}\nabla p_i^l = -\sum_j m_j \left(\frac{p_i^l}{\rho_i^2} + \frac{p_j^l}{\rho_j^2}\right)\nabla W_{ij} \tag{28}$$

while the second loop computes the density change within Δt that results from the divergence of the velocity change $\Delta t(\mathbf{a}_i^{\mathrm{p}})^l$:

$$(\Delta\rho_i^{\mathrm{II,p}})^l \stackrel{\mathrm{def}}{=} \Delta t \sum_j m_j (\Delta t(\mathbf{a}_i^{\mathrm{p}})^l - \Delta t(\mathbf{a}_j^{\mathrm{p}})^l)\nabla W_{ij}. \tag{29}$$

Here, superscript II,p denotes the IISPH form of the predicted density change due to the pressure accelerations $(\mathbf{a}_i^{\mathrm{p}})^l$. This density change is the discretized form of

$$-\Delta t \rho_i \nabla\cdot(\Delta t(\mathbf{a}_i^{\mathrm{p}})^l) = -\Delta t \rho_i \nabla\cdot(-\frac{\Delta t}{\rho_i}\nabla p_i^l) = \Delta t^2 \nabla^2 p_i^l = (\mathbf{Ap}^l)_i, \tag{30}$$

which is required in the Jacobi solver. While Eq. (29) is the SPH approximation of the left-hand side of the PPE in Eq. (27), the source term on the right-hand side of Eq. (27) is implemented with

$$\Delta\rho_i^{\mathrm{II,error}} \stackrel{\mathrm{def}}{=} \rho^0 - \rho_i - \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*)\nabla W_{ij}. \tag{31}$$

This term is the estimated density deviation at particle i at the next time step. Here superscript II,error denotes the IISPH form of the predicted density error.

IISPH implementation: The solver starts with current velocities $\mathbf{v}_i(t)$ and computes velocities $\mathbf{v}_i(t+\Delta t)$ at the next timestep. First, the current density ρ_i is computed and non-pressure accelerations are applied to compute the predicted velocity $\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \mathbf{a}^{\mathrm{nonp}}$. Then, the source term is computed with $s_i = \Delta\rho_i^{\mathrm{II,error}}$. In each solver iteration, IISPH computes $(\mathbf{a}_i^{\mathrm{p}})^l$ and $(\Delta\rho_i^{\mathrm{II,p}})^l = (\mathbf{Ap}^l)_i$ and updates the pressure with

$$p_i^{l+1} = p_i^l + k_i^{\mathrm{II}}(\Delta\rho_i^{\mathrm{II,error}} - (\Delta\rho_i^{\mathrm{II,p}})^l). \tag{32}$$

The coefficient k_i^{II} is equal to the IISPH standard relaxation factor $\alpha_i = \frac{1}{2}$ divided by the diagonal element a_{ii}, which gives

$$k_i^{\mathrm{II}} = \frac{-1}{2\Delta t^2 \frac{m_i^2}{(\rho^0)^2}\left(\sum_j(\nabla W_{ij}^{\mathrm{T}})\nabla W_{ij} + \sum_j(\sum_j \nabla W_{ij}^{\mathrm{T}})\nabla W_{ij}\right)} \tag{33}$$

for $m_i = m_j$. Finally, the predicted velocity is updated with the result from the last solver iteration: $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \Delta t(\mathbf{a}_i^{\mathrm{p}})^l$.

Equivalence of IISPH and PCISPH: PCISPH [SP09] and IISPH are derived in different ways, but the PCISPH implementation is equivalent to IISPH. To see this, we first note that the final velocity

update in IISPH can be rewritten as

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \Delta t(\mathbf{a}_i^{\mathrm{p}})^l = \mathbf{v}_i^* + \sum_l \Delta t\left((\mathbf{a}_i^{\mathrm{p}})^l - (\mathbf{a}_i^{\mathrm{p}})^{l-1}\right). \quad (34)$$

Instead of updating the predicted velocity once in IISPH, PCISPH updates the velocity $\mathbf{v}_i^*$ in each solver iteration l with

$$(\mathbf{v}_i^*)^{l+1} = (\mathbf{v}_i^*)^l + \Delta t\left((\mathbf{a}_i^{\mathrm{p}})^{l+1} - (\mathbf{a}_i^{\mathrm{p}})^l\right). \quad (35)$$

Then, the final velocity is equal to the predicted velocity from the last solver iteration: $\mathbf{v}_i(t+\Delta t) = (\mathbf{v}_i^*)^{l+1}$. The term

$$\begin{aligned}(\Delta\mathbf{a}_i^{\mathrm{p}})^{l+1} &= (\mathbf{a}_i^{\mathrm{p}})^{l+1} - (\mathbf{a}_i^{\mathrm{p}})^l \\ &= -\sum_j m_j\left(\frac{p_i^{l+1}-p_i^l}{\rho_i^2} + \frac{p_j^{l+1}-p_j^l}{\rho_j^2}\right)\nabla W_{ij} \end{aligned} \quad (36)$$

depends on the pressure differences between two iterations and PCISPH computes these pressure differences in each solver iteration. So, instead of the IISPH update

$$p_i^{l+1} = p_i^l + k_i^{\mathrm{II}}(\Delta\rho_i^{\mathrm{II,error}} - (\Delta\rho_i^{\mathrm{II,p}})^l), \quad (37)$$

PCISPH computes a pressure difference – referred to as $\tilde{p}$ in the original paper – in each iteration with

$$p_i^{l+1} - p_i^l = k_i^{\mathrm{PCI}}(\Delta\rho_i^{\mathrm{PCI,error}})^l. \quad (38)$$

Each PCISPH iteration l computes $p_i^{l+1} - p_i^l$ which is used to compute $(\mathbf{a}_i^{\mathrm{p}})^{l+1} - (\mathbf{a}_i^{\mathrm{p}})^l$ with Eq. (36) which is used to update $(\mathbf{v}_i^*)^{l+1}$ with Eq. (35). Now, if the pressure updates in Eqs. (37) and (38) are equivalent, *i.e.*, $k_i^{\mathrm{PCI}}(\Delta\rho_i^{\mathrm{PCI,error}})^l = k_i^{\mathrm{II}}(\Delta\rho_i^{\mathrm{II,error}} - (\Delta\rho_i^{\mathrm{II,p}})^l)$, then PCISPH and IISPH are equivalent. The coefficient k_i^{PCI} is given as

$$k_i^{\mathrm{PCI}} = \frac{(\rho^0)^2}{2\Delta t^2 m_i^2\left(\sum_j(\nabla W_{ij}^{\mathrm{T}})\nabla W_{ij} + \sum_j(\sum_j \nabla W_{ij}^{\mathrm{T}})\nabla W_{ij}\right)} \quad (39)$$

which means that $k_i^{\mathrm{PCI}} = -k_i^{\mathrm{II}}$ and the question is whether $(\Delta\rho_i^{\mathrm{PCI,error}})^l = (\Delta\rho_i^{\mathrm{II,p}})^l - \Delta\rho_i^{\mathrm{II,error}}$. PCISPH defines

$$(\Delta\rho_i^{\mathrm{PCI,error}})^l \overset{\mathrm{def}}{=} (\rho_i^*)^l - \rho^0, \quad (40)$$

where the density $(\rho_i^*)^l$ is computed in each solver iteration for predicted particle positions $(\mathbf{x}_i^*)^l = \mathbf{x}_i + \Delta t(\mathbf{v}_i^*)^l$. This predicted density can alternatively be expressed using the divergence of $(\mathbf{v}_i^*)^l$:

$$(\Delta\rho_i^{\mathrm{PCI,error}})^l = \rho_i + \Delta t\sum_j m_j\left((\mathbf{v}_i^*)^l - (\mathbf{v}_j^*)^l\right)\nabla W_{ij} - \rho^0. \quad (41)$$

Using Eq. (34) and Eq. (35), this can be written as

$$\begin{aligned}(\Delta\rho_i^{\mathrm{PCI,error}})^l = &-\rho^0 + \rho_i + \Delta t\sum_j m_j\left(\mathbf{v}_i^* - \mathbf{v}_j^*\right)\nabla W_{ij} \\ &+ \Delta t\sum_j m_j(\Delta t(\mathbf{a}_i^{\mathrm{p}})^l - \Delta t(\mathbf{a}_j^{\mathrm{p}})^l)\nabla W_{ij}. \end{aligned} \quad (42)$$

From the definitions in Eqs. (29) and (31), we see that $(\Delta\rho_i^{\mathrm{PCI,error}})^l = (\Delta\rho_i^{\mathrm{II,p}})^l - \Delta\rho_i^{\mathrm{II,error}}$ which means that the IISPH and PCISPH updates in Eq. (37) and Eq. (38) are equivalent.

PBF derivation: PBF [MM13] is derived in an alternative way compared to PCISPH and IISPH, but its implementation is equivalent to PCISPH and IISPH. Before showing the equivalence of PBF with PCISPH, we discuss the derivation of a PBF variant. While [MM13] assumes particle masses of one in some computations and omits the respective terms, we keep the mass terms to simplify the equivalence discussion.

PBF considers a vector of n constraints $\mathbf{C}(\mathbf{x})$ that depend on n particle positions $\mathbf{x}$ and it computes n displacements $\Delta\mathbf{x}$ such that $\mathbf{C}(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{0}$. The displacements are modeled as $\Delta\mathbf{x} = \mathbf{J}(\mathbf{x})^{\mathrm{T}}\lambda$ with $\mathbf{J}$ being the Jacobian and λ being a vector of scalar coefficients. Now, PBF uses the Taylor approximation $\mathbf{C}(\mathbf{x}+\Delta\mathbf{x}) \approx \mathbf{C}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{J}^{\mathrm{T}}(\mathbf{x})\lambda = \mathbf{0}$, computes $\lambda = -(\mathbf{J}(\mathbf{x})\mathbf{J}^{\mathrm{T}}(\mathbf{x}))^{-1}\mathbf{C}(\mathbf{x})$ and the respective displacements $\Delta\mathbf{x} = \mathbf{J}^{\mathrm{T}}(\mathbf{x})\lambda$. PBF ignores all off-diagonal elements of $(\mathbf{J}(\mathbf{x})\mathbf{J}^{\mathrm{T}}(\mathbf{x}))^{-1}$ and assumes exactly one constraint per particle which results in $\lambda_i = -\frac{1}{\nabla C_i \nabla C_i^{\mathrm{T}}}C_i$. The respective particle displacements are computed with $\Delta\mathbf{x}_i = \lambda_i\nabla_i C_i$. The term ∇C_i denotes row i from $\mathbf{J}$, ∇C_i^{T} denotes column i from $\mathbf{J}$ and $\nabla_i C_i$ is a column vector of the partial derivatives of C_i with respect to the components of position $\mathbf{x}_i$. In terms of the constraint C_i, PBF considers the density deviation $C_i = \frac{\rho_i^*}{\rho^0} - 1$, and SPH is used for the density: $\rho_i^* = m_i\sum_j W_{ij}$. PBF also uses SPH to approximate $\nabla_i C_i \approx \sum_j C_j V_j \nabla W_{ij} = \frac{m_i}{\rho^0}\sum_j \nabla W_{ij}$ and $\nabla_j C_i \approx -\frac{m_i}{\rho^0}\nabla W_{ij}$. The vector $\nabla_i C_i$ is computed with a standard SPH formulation, while the vector $\nabla_j C_i$ is only roughly approximated. It ignores all neighbors of j except particle i. The coefficient λ_i is then

$$\lambda_i = -\frac{1}{\frac{m_i^2}{(\rho^0)^2}\left(\sum_j(\nabla W_{ij}^{\mathrm{T}})\nabla W_{ij} + \sum_j(\sum_j \nabla W_{ij}^{\mathrm{T}})\nabla W_{ij}\right)}\left(\frac{\rho_i^*}{\rho^0} - 1\right) \quad (43)$$

and the respective symmetrized displacement is

$$\Delta\mathbf{x}_i = \lambda_i\nabla_i C_i = 2\lambda_i\frac{m_i}{2\rho^0}\sum_j \nabla W_{ij} \approx \frac{m_i}{2\rho^0}\sum_j(\lambda_i+\lambda_j)\nabla W_{ij}. \quad (44)$$

PBF iteratively refines the result. Each PBF solver iteration l computes $(\lambda)_i^l$ and $(\Delta\mathbf{x}_i)^l$ and updates predicted densities $(\rho_i^*)^l$ and predicted positions and kernel gradients $(\nabla W_{ij})^l$.

Equivalence of PCISPH and PBF: PBF performs the same computations as PCISPH, but reformulates everything in terms of intermediate positions $(\mathbf{x}_i^*)^l$ instead of intermediate velocities $(\mathbf{v}_i^*)^l$. Therefore, the relation $(\mathbf{x}_i^*)^l = \mathbf{x}_i + \Delta t(\mathbf{v}_i^*)^l$ is used and Eq. (35) from PCISPH is rewritten as

$$(\mathbf{x}_i^*)^{l+1} = (\mathbf{x}_i^*)^l + \Delta t^2\left((\mathbf{a}_i^{\mathrm{p}})^{l+1} - (\mathbf{a}_i^{\mathrm{p}})^l\right). \quad (45)$$

PBF computes the distance $(\Delta\mathbf{x}_i)^l = \Delta t^2\left((\mathbf{a}_i^{\mathrm{p}})^{l+1} - (\mathbf{a}_i^{\mathrm{p}})^l\right)$ in each iteration and updates the positions instead of the velocities. In order to compute the predicted distance, Eq. (36) from PCISPH is multiplied with Δt^2:

$$\Delta t^2(\Delta\mathbf{a}_i^{\mathrm{p}})^l = (\Delta\mathbf{x}_i)^l = -\Delta t^2\sum_j m_j\left(\frac{p_i^{l+1}-p_i^l}{\rho_i^2} + \frac{p_j^{l+1}-p_j^l}{\rho_j^2}\right)\nabla W_{ij}. \quad (46)$$

Now, we use Eq. (38) from PCISPH and assume that all particles have equal mass and rest density. Then, we can rewrite the PBF displacement from Eq. (46) as

$$(\Delta\mathbf{x}_i)^l = \frac{m_i}{2\rho^0}\sum_j\left((\lambda_i)^l + (\lambda_j)^l\right)\nabla W_{ij} \quad (47)$$

$$\text{with} \qquad (\lambda_i)^l = \frac{-2\Delta t^2 k_i^{\text{PCI}} (\Delta\rho_i^{\text{PCI,error}})^l}{\rho^0} \tag{48}$$

which is equivalent to the PBF formulations in Eqs. (43) and (44). PBF computes $(\lambda_i)^l$ in each iteration and updates $(\Delta\mathbf{x}_i)^l$ accordingly which is equivalent to PCISPH. While PCISPH computes accelerations $(\Delta\mathbf{a}_i^{\text{p}})^l$ with Eq. (36), PBF computes the respective distances $\Delta t^2(\Delta\mathbf{a}_i^{\text{p}})^l$ with Eq. (46). PCISPH updates velocities and positions, while PBF directly updates the positions.

DFSPH concept: DFSPH [BK15] solves two PPEs. It first solves $-\Delta t^2\nabla^2 p^{\text{DI}} = \frac{\rho^* - \rho^0}{\Delta t}$, applies the respective pressure acceleration with $\mathbf{v}^{**} = \mathbf{v}^* + \Delta t\mathbf{a}^{\text{DI}}$ and advects the particles with $\mathbf{x}^{\text{DI}} = \mathbf{x} + \Delta t\mathbf{v}^{**}$. Superscript DI stands for density invariance. It indicates that the respective quantities are related to a PPE with density invariance as source term. As the first PPE solve only addresses the density deviation, a second step solves $-\Delta t\nabla^2 p^{\text{VD}} = -\rho\nabla\cdot\mathbf{v}^{**}$ to account for a potentially remaining divergence in the velocity field $\mathbf{v}^{**}$. The respective pressure acceleration $\mathbf{a}^{\text{VD}}$ is applied with $\mathbf{v}^{\text{VD}} = \mathbf{v}^{**} + \Delta t\mathbf{a}^{\text{VD}}$ and $\mathbf{x}^{\text{DI}}$ and $\mathbf{v}^{\text{VD}}$ are the final positions and velocities. Superscript VD stands for velocity divergence, indicating the formulation of the source term in the respective PPE. It is remarkable that DFSPH computes two pressure values per particle. DFSPH is one option to combine two PPEs with different source terms. Alternative combinations are, *e.g.*, discussed in [CBG*18].

Equivalence of PCISPH and DFSPH: In both PPE solvers, DFSPH updates predicted velocities and pressure differences in the same way as PCISPH. PCISPH updates $(\mathbf{v}_i^*)^{l+1} = (\mathbf{v}_i^*)^l + \Delta t\left((\mathbf{a}_i^{\text{p}})^{l+1} - (\mathbf{a}_i^{\text{p}})^l\right)$ with

$$(\mathbf{a}_i^{\text{p}})^{l+1} - (\mathbf{a}_i^{\text{p}})^l = -\sum_j m_j\left(\frac{p_i^{l+1} - p_i^l}{\rho_i^2} + \frac{p_j^{l+1} - p_j^l}{\rho_j^2}\right)\nabla W_{ij} \tag{49}$$

and $p_i^{l+1} - p_i^l = k_i^{\text{PCI}}(\Delta\rho_i^{\text{PCI,error}})^l$. These equations can be written as

$$(\mathbf{v}_i^*)^{l+1} = (\mathbf{v}_i^*)^l - \Delta t\sum_j m_j\left(\frac{k_i^{\text{PCI}}(\Delta\rho_i^{\text{PCI,error}})^l}{\rho_i^2} + \frac{k_j^{\text{PCI}}(\Delta\rho_j^{\text{PCI,error}})^l}{\rho_j^2}\right)\nabla W_{ij}.$$

The original DFSPH notation for this update reads

$$(\mathbf{v}_i^*)^{l+1} = (\mathbf{v}_i^*)^l - \Delta t\sum_j m_j\left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j}\right)\nabla W_{ij} \tag{50}$$

and we have to check whether $\kappa_i = \frac{k_i^{\text{PCI}}(\Delta\rho_i^{\text{PCI,error}})^l}{\rho_i}$. Using $\frac{(\rho^0)^2}{\rho_i} \approx \rho_i$, we get

$$\frac{k_i^{\text{PCI}}(\Delta\rho_i^{\text{PCI,error}})^l}{\rho_i} = \frac{1}{2}\frac{1}{\Delta t}\frac{((\rho_i^*)^l - \rho^0)}{\Delta t}\frac{\rho_i}{m_i^2\left(\sum_j\nabla W_{ij}^{\text{T}}\nabla W_{ij} + \sum_j(\sum_j\nabla W_{ij}^{\text{T}})\nabla W_{ij}\right)}$$

which is equal to the definition of κ_i for the constant density solver in [BK17] up to the scalar factor $\frac{1}{2}$. The second PPE solver in DFSPH works exactly as the first one, but replaces $\frac{((\rho_i^*)^l - \rho^0)}{\Delta t}$ with $-\rho_i\nabla\cdot(\mathbf{v}_i^*)^l$ in the previous equation. So, the solver implementations of PCISPH and DFSPH are equivalent.

Algorithm 1 IISPH implementation

```
for all particle i do
    v_i^* = v_i(t) + Δt a^nonp
    compute Δρ_i^{II,error} using Eq. (31)
    init residuum r_i^0 = −Δρ_i^{II,error}
    p_i^0 = 0
    l = 0
while |r_avg^l| > tolerance do
    for all particle i do
        compute pressure accelerations (a_i^p)^l using Eq. (28)
    for all particle i do
        compute density change (Δρ_i^{II,p})^l using Eq. (29)
        update residuum r_i^{l+1} = (Δρ_i^{II,p})^l − Δρ_i^{II,error}
        update pressure p_i^{l+1} using Eq. (32)
        clamp negative pressure p_i^{l+1} = max(p_i^{l+1}, 0)
        l = l + 1
for all particle i do
    v_i(t + Δt) = v_i^* + Δt (a_i^p)^l
```

Discussion: This section has shown that the implementations of PCISPH, IISPH, PBF and DFSPH can be converted into each other with simple transformations. Algorithm 1 shows the implementation of IISPH as an example. Nevertheless, the solvers differ in terms of implementation details. *E.g.*, PCISPH uses the same coefficient k_i^{PCI} for all particles in the solver update, which is in contrast to all other discussed solvers. The computation is efficient, but negatively affects the convergence and also the stability. IISPH and DFSPH use individual computations of this term, but work with the same kernel gradients in all iterations. This is in contrast to PBF, where the kernel gradients are updated per iteration. This probably improves the convergence, but is also expensive to compute. DFSPH solves two systems. This sounds expensive, but various terms can be reused in both solvers. Also, the second solver has a positive effect on the convergence of the first solver, which means that the overall iteration number of both solvers is typically smaller compared to the other solvers. Last, but not least, all original solver implementations have used different boundary handlings. The recently very active research in SPH boundary handling indicates that the combination of pressure solver and boundary handling significantly influences the stability and the solver performance.

4. Boundary Handling

Most SPH approaches handle boundary conditions by virtually extending field quantities (*e.g.*, density or pressure) into the domain of the boundary. In SPH the fluid domain is represented by particles (see Figure 1). To extend a field a suitable representation of the boundary is required and field quantities must be determined for the boundary domain. In recent years, several approaches have been introduced to represent boundary geometries and to enforce boundary conditions, *e.g.*, [AIA*12, FM15, KB17, BGPT18, BGI*18, GPB*19, BKWK19, BKWK20]. In this section we first in-

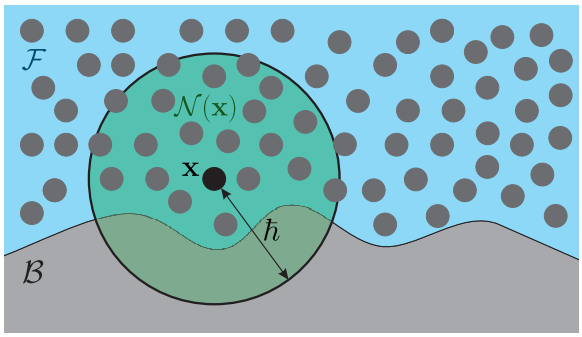

Figure 1: In an SPH simulation the fluid domain $\mathcal{F}$ (blue) is discretized by particles. Each particle $\mathbf{x}$ has a support domain $\mathcal{N}(\mathbf{x})$ (green) which is defined by the support radius $\hbar$ of the compact kernel function. The boundary domain $\mathcal{B}$ is shown in gray.

troduce approaches to represent a boundary and then discuss how to compute pressure values at the boundary for the pressure solver.

4.1. Boundary Representation

A common way to resolve boundary penetrations is to extend the density field into the boundary. If a fluid particle comes close to the boundary, the resulting density contribution leads to local compression that violates the incompressibility constraint (Eq. (19)) and the corresponding pressure force prevents a penetration.

By substituting the density function $\rho(\mathbf{x})$ in the continuous approximation in Eq. (3), we get an integral to approximate the density at position $\mathbf{x}$:

$$\rho(\mathbf{x}) \approx \int_{\mathcal{N}(\mathbf{x})} \rho(\mathbf{x}')\, W(\|\mathbf{x}-\mathbf{x}'\|,h)\, d\mathbf{x}'. \tag{51}$$

We can split this integral in a fluid and boundary part (see Figure 1)

$$\begin{aligned}\rho(\mathbf{x}) \approx & \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{F}} \rho(\mathbf{x}')\, W(\|\mathbf{x}-\mathbf{x}'\|,h)\, d\mathbf{x}' + \\ & \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{B}} \rho(\mathbf{x}')\, W(\|\mathbf{x}-\mathbf{x}'\|,h)\, d\mathbf{x}' \\ = & \rho_{\mathcal{F}}(\mathbf{x}) + \rho_{\mathcal{B}}(\mathbf{x}),\end{aligned} \tag{52}$$

where $\mathcal{F}$ and $\mathcal{B}$ denote the fluid and boundary domain, respectively. Now it is possible to solve the fluid and the boundary integral with different methods. In this section we focus on the boundary integral and assume that the fluid integral is computed by using the SPH discretization introduced in Eq. (5). To solve the boundary integral a discretization is required. In the following we will discuss explicit particle-based discretizations and implicit ones.

Note that there exist also mesh-based approaches for boundary handling. For example, Bodin et al. [BLS12] avoid a penetration of fluid particles and a triangle mesh by adding unilateral constraints to the simulation. Huber et al. [HEW15] use a continuous collision detection in combination with correction impulses to couple fluids and cloth models. However, such approaches are typically significantly slower than particle-based or implicit methods, especially for complex boundary geometries [FM15]. Therefore, these methods are not discussed in detail in this report.

Particle-based boundary representation A popular approach to solve the boundary integral in Eq. (52) is to use an additional particle discretization for the boundary, *e.g.*, [IAGT10, AIA*12, BGPT18, BGI*18, GPB*19]. The *boundary particles* serve as additional sampling points and typically have the same radius as the fluid particles. In this way the boundary integral can be solved analogously to the fluid integral

$$\rho_i = \rho_{\mathcal{F}}(\mathbf{x}_i) + \rho_{\mathcal{B}}(\mathbf{x}_i) \approx \sum_j m_j W_{ij} + \sum_k \tilde{m}_k W_{ik}, \tag{53}$$

where j and k are the fluid and boundary neighbors of particle i, respectively. The mass $\tilde{m}_k$ is used to extend the fluid density field into the boundary and does not represent the material properties of the boundary object. This means that a boundary particle has the same rest density ρ^0 as a fluid particle. Therefore, $\tilde{m}_k$ is often called a pseudo mass. The boundary particles are also considered when computing pressure values and pressure forces for fluid particles close to the boundary.

There exist different strategies to sample the boundary. We can use multiple layers of boundary particles or just a single layer. Moreover, there exist uniform or non-uniform sampling strategies. Using multiple layers typically requires more memory and computation time. Moreover, it can be difficult to find a good representation for arbitrary geometries when using a uniform sampling. Therefore, one of the most popular strategies is to use a single layer non-uniform sampling. Such a sampling was proposed by Akinci et al. [AIA*12]. This approach determines the pseudo mass of a boundary particle as

$$V_k = \frac{1}{\sum_l W_{kl}}, \qquad \tilde{m}_k = V_k \rho^0, \tag{54}$$

where l denotes the boundary neighbors of particle k and V_k is the volume represented by particle k. Note that the volume V_k gets larger for sparsely sampled regions and smaller for densely sampled regions. In this way the non-uniform sampling is considered in the computation.

An advantage of this particle-based sampling is that it is simple to generate the boundary particles. Moreover, it is straightforward to integrate these particles in the SPH computation. However, even simple geometries require a large number of boundary particles which must be considered in the neighborhood search and in the computation of field quantities (*e.g.*, density). Furthermore, when sampling a smooth geometry by particles, this leads to a somehow bumpy representation of the surfaces which can reduce the accuracy of the force computation and cause artificial friction [KB17, BKWK20]. Note that this problem can be solved by an extended pressure force computation at the boundary [BGPT18] or by using an implicit boundary representation as discussed below.

Implicit boundary representation In contrast to explicit surface representations, implicit approaches typically represent the boundary by a function which returns the signed distance to the boundary, the density contribution of the boundary or the boundary volume in the support domain [HKK07a, HKK07b, BLS12, KB17, BKWK20]. Such methods have the advantage that the particle size is decoupled from the boundary representation. This is especially important in adaptive simulations [WAK20]. Moreover, this enables the usage of

flexible data structures, like adaptive octrees with higher-order approximations [KDBB17], to represent the boundary in a memory-efficient and accurate way. Another advantage is that the problems of a bumpy surface representation (see above) are avoided.

Harada et al. [HKK07a, HKK07b] represent the boundary using a signed distance field (SDF). In their approach the density contribution of a planar surface is sampled depending on the distance to a prototype fluid particle. However, the approach does not provide correct density values for curved geometries. To solve this problem Koschier and Bender [KB17] precompute the boundary integral in Eq. (52) for the nodes of a regular spatial grid, which they call *density map*. The integral is determined using adaptive Gauss-Kronrod quadrature. However, the density function in Eq. (52) is a discontinuous, piecewise constant function which returns zero outside and ρ_0 inside the boundary. Integrating such a function using a fixed pattern quadrature scheme leads to staircase artifacts [BKWK20]. Therefore, the authors propose a modification of the integral

$$\rho_{\mathcal{B}}(\mathbf{x}) = \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{B}} \gamma(\Phi(\mathbf{x}'))\, W\big(\|\mathbf{x}-\mathbf{x}'\|, h\big) d\mathbf{x}' \tag{55}$$

using a linear increasing density function γ instead of the discontinuous, piecewise constant function, where

$$\gamma(x) = \begin{cases} \rho_0\left(1-\frac{x}{r}\right) & \text{if } x < r \\ 0 & \text{otherwise.} \end{cases} \tag{56}$$

In Eq. (55) the new function γ depends on the signed distance which is defined as

$$\Phi(\mathbf{x}) = \begin{cases} -\inf_{\tilde{\mathbf{x}}\in\partial\mathcal{B}} \|\mathbf{x}-\tilde{\mathbf{x}}\| & \text{if } \mathbf{x}\in\mathcal{B} \\ \inf_{\tilde{\mathbf{x}}\in\partial\mathcal{B}} \|\mathbf{x}-\tilde{\mathbf{x}}\| & \text{otherwise.} \end{cases} \tag{57}$$

Finally, the boundary density contribution for a fluid particle can be efficiently queried during runtime by interpolating the precomputed values of the grid cell which contains the particle. Higher-order polynomials are used for the interpolation to improve the accuracy. Moreover, the memory consumption is reduced by storing only the density map in a narrow band around the boundary surface.

Bender et al. [BKWK20] improve this concept by precomputing the volume $V_{\mathcal{B}}$ of the boundary part $\mathcal{N}(\mathbf{x})\cap\mathcal{B}$ instead of the density. The boundary volume is determined by solving an integral

$$V_{\mathcal{B}}(\mathbf{x}) = \int_{\mathcal{N}(\mathbf{x})\cap\mathcal{B}} \gamma^*(\Phi(\mathbf{x}'))d\mathbf{x}' \tag{58}$$

using Gauss-Legendre quadrature. This approach uses a cubic function to avoid staircase artifacts:

$$\gamma^* = \begin{cases} \frac{C(x)}{C(0)} & \text{if } 0 < x < r \\ 1 & \text{if } x \le 0 \\ 0 & \text{otherwise,} \end{cases} \tag{59}$$

where C is the cubic spline kernel [Mon05]. In comparison to the linear function γ, the cubic function γ^* has a smooth transition at distance $x = r$. Similar to the density maps approach, Bender et al. store the values $V_{\mathcal{B}}(\mathbf{x})$ in a narrow band grid and approximate the required values at runtime by interpolation with cubic shape functions. Moreover, they define a sampling point $\mathbf{x}^*$ at the closest position to $\mathbf{x}$ on the boundary to represent the boundary volume $V_{\mathcal{B}}$. In this way the boundary density value can be approximated as $\rho_{\mathcal{B}} \approx V_{\mathcal{B}}(\mathbf{x})\rho^0 W(\mathbf{x}-\mathbf{x}^*, h)$.

In the pressure solver typically the density gradient is required. When using a density map, the gradient is determined by differentiating the shape functions of the map. However, since the smoothness at cell interfaces is not guaranteed [KDBB17], a high-resolution map is required to avoid artifacts. In contrast the implicit boundary volume computation uses a classical SPH formulation to compute the gradient and therefore solves this problem.

Implicit boundary approaches avoid the problems of a bumpy surface representation and are fast at runtime. However, when using high-resolution grids, the precomputation can take a significant amount of time. Moreover, the implementation and the integration in existing SPH codes is more complex than for particle-based methods.

4.2. Boundary Pressure Computation

SPH pressure solvers typically use the symmetric formula in Eq. (12) to compute the pressure acceleration (cf. Eq. (28)). At the boundary the sum is split in a fluid part and a boundary part:

$$\mathbf{a}_i^{\mathrm{p}} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} - \sum_k \tilde{m}_k \left(\frac{p_i}{\rho_i^2} + \frac{p_k}{(\rho^0)^2}\right)\nabla W_{ik}\,. \tag{60}$$

As discussed above different boundary representations can be used to compute the term $\tilde{m}_k \nabla W_{ik}$. Moreover, the density of the boundary is typically set to the rest density of the fluid ρ^0. However, the pressure value at the boundary p_k is still unknown. In the following we introduce two methods to determine this value.

Pressure mirroring A simple way to determine the boundary pressure is to mirror the pressure value from the corresponding fluid particle in Eq. (60), *i.e.*, $p_k = p_i$. In this way the pressure at the boundary particle is similar to the pressure in its neighborhood. This approach is quite popular due to its simplicity and efficiency. However, it leads to inconsistent pressure values at the boundary since a boundary particle can represent different pressure values for different adjacent fluid particles. Another problem is that all boundary neighbors of a fluid particle have the same pressure value in the computation. Both problems can lead to visual artifacts [BGI*18, BGPT18].

Pressure extrapolation An alternative approach that solves the problem of inconsistent boundary pressure values is pressure extrapolation. The core idea is to extrapolate the fluid pressure to the boundary. Adami et al. [AHA12] use an SPH formulation to compute the extrapolated pressure for a boundary particle k as

$$p_k = \frac{\sum_l p_l W_{kl} + \mathbf{g}\cdot\sum_l \rho_l(\mathbf{x}_k - \mathbf{x}_l)W_{kl}}{\sum_l W_{kl}}, \tag{61}$$

where l denote the fluid neighbors of boundary particle k and $\mathbf{g}$ is the gravity vector. The approach requires an additional loop over all boundary particles, their fluid neighbors must be determined and the pressure at the boundary must be stored. The advantage of this computation is that it provides consistent pressure values. However, the method suffers from the particle deficiency problem at the boundary which reduces the accuracy.

Band et al. [BGPT18] solve this problem by performing the extrapolation using MLS instead of SPH. In a first step they translate the positions of the boundary particle $\bar{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{c}_k$ and its fluid neighbors $\bar{\mathbf{x}}_l = \mathbf{x}_l - \mathbf{c}_k$, where $\mathbf{c}_k$ is the center of the neighboring fluid particles $\mathbf{c}_k = \frac{\sum_l \mathbf{x}_l V_l W_{kl}}{\sum_l V_l W_{kl}}$. Then they define a linear pressure function for each boundary particle $p_k(\mathbf{x}) = \alpha_k + \beta_k x + \gamma_k y + \delta_k z$ and determine the unknown coefficients by computing the best fit of $p_k(\mathbf{x})$ in the least-squares sense to the known pressure values p_l at the fluid neighbors of k. This yields the following coefficients

$$\alpha_k = \frac{\sum_l p_l V_l W_{kl}}{\sum_l V_l W_{kl}} \tag{62}$$

$$(\beta_k, \gamma_k, \delta_k)^T = \left(\sum_l \bar{\mathbf{x}}_l \bar{\mathbf{x}}_l^T V_l W_{kl} \right)^{-1} \cdot \sum_l \bar{\mathbf{x}}_l p_l V_l W_{kl}. \tag{63}$$

Finally, the boundary pressure in Eq. (60) is determined by $p_k = p_k(\bar{\mathbf{x}}_k)$.

5. Materials

In recent years SPH methods have been introduced to simulate different types of materials like highly viscous fluids, deformable solids or rigid bodies. Since all these methods are based on SPH, the materials can be easily combined in a unified solver. In the following we introduce SPH methods for different material types and discuss how to compute the pressure forces at their interfaces.

5.1. Viscous Fluids

Viscosity is an important phenomenon which is responsible for various visually appealing effects like coiling and buckling as well as for the general energy dissipation. Therefore, in this section different methods for the simulation of low viscous fluids like water and highly viscous fluids like mud, honey, and dough (see Fig. 2) are introduced.

5.1.1. Viscous Force

The viscous force in the Navier-Stokes equations (see Eq. (22)) for incompressible fluids

$$\mathbf{f}_{\text{visco}} = \mu \nabla^2 \mathbf{v} \tag{64}$$

is derived by substituting the stress tensor of a Newtonian fluid

$$\mathbf{T} = -p\mathbb{1} + 2\mu\mathbf{E} \tag{65}$$

in the conservation law of linear momentum (see Eq. (20)) and considering the incompressibility constraint $\nabla \cdot \mathbf{v} = 0$. The material parameter μ is also known as the dynamic viscosity and $\mathbf{E}$ defines the strain rate tensor

$$\mathbf{E} = \frac{1}{2}(\nabla \mathbf{v} + (\nabla \mathbf{v})^T). \tag{66}$$

Recent viscosity solvers either compute the Laplacian of the velocity field using an SPH formulation or use a strain rate based formulation. The first approach considers the incompressibility constraint by definition. However, when using a strain rate based method, a divergence-free velocity field has to be enforced to avoid undesired bulk viscosity [Lau11, PICT15].

5.1.2. Explicit Viscosity

The viscous force for a particle i can be approximated by using the standard SPH discretization of the Laplacian of the velocity field

$$\nabla^2 \mathbf{v}_i = \sum_j \frac{m_j}{\rho_j} \mathbf{v}_j \nabla^2 W_{ij}. \tag{67}$$

However, this formulation is sensitive to particle disorder [Mon05] and problems could occur since the second derivative of the kernel changes sign inside the kernel domain and can even be discontinuous (*e.g.*, cubic spline kernel) [Pri12].

One way to avoid these issues is to compute the second derivative by taking two first SPH derivatives [FMH*94, WBF*96, TDF*15]. However, this approach introduces additional smoothing, requires more memory and increases the computational costs. Another way is to combine an SPH derivative with finite differences [Bro85]. This approach became very popular and was applied for scalar [Mon92, CM99, IOS*14] and vector quantities [ER03, JSD04, Mon05, Pri12, WKBB18]. Following this idea the Laplacian of the velocity can be approximated as [Mon05]:

$$\nabla^2 \mathbf{v}_i = 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij}, \tag{68}$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, d is the number of spatial dimensions and the term $0.01h^2$ is introduced in the denominator to avoid singularities. This formulation has several advantages [Mon92]: it conserves linear and angular momentum, is Galilean invariant and vanishes for rigid body rotation.

For low viscous fluids with a small dynamic viscosity coefficient μ the resulting force (see Eq. (64)) can be directly used in the explicit time integration step. Simulation methods for highly viscous materials are discussed in the following section.

5.1.3. Implicit Viscosity

In recent years the simulation of highly viscous fluids has become popular in computer graphics. Since such materials have a large viscosity coefficient, implicit methods are required for a stable simulation. In the following we introduce different strain rate based formulations [TDF*15, PICT15, PT16, BK17] and an approach using directly the Laplacian of the velocity field [WKBB18].

Implicit strain rate based solver Takahashi et al. [TDF*15] take two first SPH derivatives to first compute the strain rate tensor (Eq. (66)) and then the divergence of the corresponding stress tensor (Eq. (65)). This yields the following viscosity force:

$$\mathbf{f}_{\text{visco}} = \nabla \cdot (2\mu\mathbf{E}) = 2\mu \sum_j m_j \left(\frac{\mathbf{E}_i}{\rho_i^2} + \frac{\mathbf{E}_j}{\rho_j^2} \right) \nabla W_{ij}. \tag{69}$$

An implicit Euler integration is applied to compute new particle velocities by solving

$$\mathbf{v}(t + \Delta t) = \mathbf{v}^* + \frac{\Delta t}{\rho} \mathbf{f}_{\text{visco}}(t + \Delta t), \tag{70}$$

where $\mathbf{v}^*$ is a predicted velocity which considers all non-pressure forces except viscosity. The implicit integration scheme enables a stable simulation of highly viscous fluids while the viscosity force is independent of the spatial and temporal resolution. However,

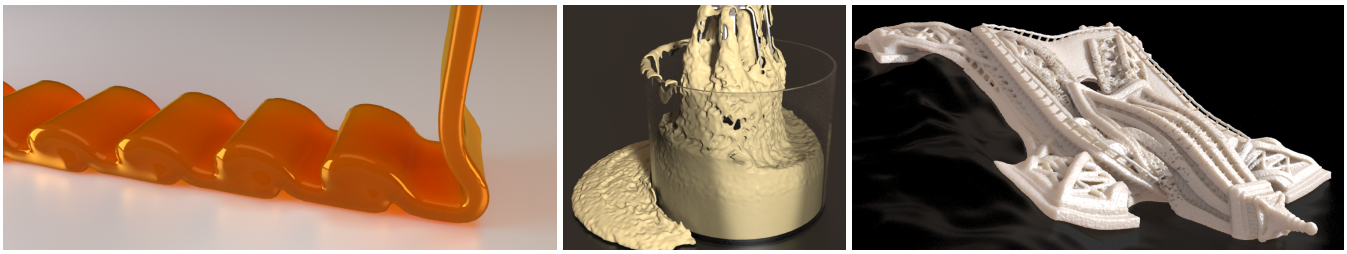

Figure 2: Simulation of highly viscous behavior. Left: buckling effects [WKBB18], center: viscous dough interacting with fast moving solid [PICT15], right: melting effects (the model is courtesy of Pranav Panchal) [PT16].

the solution of the linear system (70) is computationally expensive since the formulation of Takahashi et al. considers the second-ring neighbors of all particles which leads to a large number of non-zero elements in the system matrix.

Velocity gradient decomposition An alternative implicit formulation for the simulation of highly viscous fluids was presented by Peer et al. [PICT15,PT16]. The core idea of this approach is to first decompose the velocity gradient as

$$\nabla \mathbf{v} = \underbrace{\frac{1}{2}(\nabla \mathbf{v} - (\nabla \mathbf{v})^T)}_{\mathbf{R}} + \underbrace{\frac{1}{3}(\nabla \cdot \mathbf{v})\mathbb{1}}_{\mathbf{V}} + \underbrace{(\mathbf{E} - \frac{1}{3}(\nabla \cdot \mathbf{v})\mathbb{1})}_{\mathbf{S}}, \quad (71)$$

where $\mathbf{R}$ is the spin rate tensor, $\mathbf{V}$ the expansion rate tensor and $\mathbf{S}$ the traceless shear rate tensor. Then the shear rate tensor can be reduced by a user-defined factor $0 \leq \xi \leq 1$ without influencing the other components in order to obtain a target velocity gradient $\nabla \mathbf{v}^{\text{target}} = \mathbf{R} + \mathbf{V} + \xi \mathbf{S}$. The final particle velocities are reconstructed from the target gradient by a first-order Taylor approximation

$$\mathbf{v}_i(t+\Delta t) = \frac{1}{\rho_i} \sum_j m_j \left(\mathbf{v}_j(t+\Delta t) + \frac{\nabla \mathbf{v}_i^{\text{target}} + \nabla \mathbf{v}_j^{\text{target}}}{2} \mathbf{x}_{ij} \right) W_{ij}. \quad (72)$$

The resulting linear system for the velocities is solved for the x-, y- and z-component separately using the conjugate gradient method.

This approach was later extended by vorticity diffusion to improve the rotational motion [PT16]. The diffusion process is described by $\frac{D\omega}{Dt} = \nu \nabla^2 \omega$, where the vorticity ω can be extracted from the spin rate tensor $\mathbf{R}$. To consider this diffusion in the simulation the system $\nabla^2 \omega_i^{\text{target}} = \xi \nabla^2 \omega$. is solved to obtain ω_i^{target}. This target vorticity is used to construct a target spin rate tensor $\mathbf{R}^{\text{target}}$ which yields a new target velocity gradient $\nabla \mathbf{v}^{\text{target}} = \mathbf{R}_i^{\text{target}} + \mathbf{V} + \xi \mathbf{S}$ from which the final velocities can be reconstructed using Eq. (72).

While being efficient and stable, these methods also have some drawbacks. The linearization in Eq. (72) introduces a significant damping. Therefore, the approach is not recommended to simulate low viscous flow. Moreover, the reconstruction of the velocity field can be problematic when using SPH [BGFAO17]. Finally, the parameter ξ is not physically meaningful and depends on the temporal and spatial resolution.

Strain rate constraint formulation Bender and Koschier [BK17] define a velocity constraint $\mathbf{C}_i(\mathbf{v}) = \mathbf{E}_i - \gamma \mathbf{E}_i$ for each particle i to reduce the strain rate $\mathbf{E}_i$ by a user-defined coefficient $0 \leq \gamma \leq 1$. Due to the symmetry of the strain rate tensor the constraint can be defined as six-dimensional function containing the upper triangular part of the tensor. To enforce the constraint the following linear system for the Lagrange multiplier λ is solved by Jacobi iterations:

$$\left(\frac{1}{\rho_i} \frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_i} \left(\frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_i} \right)^T + \sum_j \frac{1}{\rho_i} \frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_j} \left(\frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_j} \right)^T \right) \lambda_i = \mathbf{E}_i - \gamma \mathbf{E}_i. \quad (73)$$

The final velocities are computed as

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^* + \frac{1}{m_i} \left(\frac{m_i}{\rho_i} \left(\frac{\partial \mathbf{E}_i}{\partial \mathbf{v}_i} \right)^T \lambda_i + \sum_j \frac{m_j}{\rho_j} \left(\frac{\partial \mathbf{E}_j}{\partial \mathbf{v}_i} \right)^T \lambda_j \right). \quad (74)$$

While the approach is computationally more expensive than the method of Peer et al., it can also simulate low viscous fluids. Apart from that it has the same drawback that the viscosity parameter is not physically meaningful and depends on the temporal and spatial resolution.

Implicit Laplacian based solver In contrast to the previous approaches, Weiler et al. [WKBB18] introduce an implicit viscosity solver based on the Laplacian of the velocity field instead of using the strain rate. To compute the viscosity acceleration they substitute Eq. (68) in Eq. (64) and make the resulting term symmetric:

$$\mathbf{a}_{\text{visco}} = \frac{\mu}{\rho_i} \nabla^2 \mathbf{v}_i = 2(d+2) \sum_j \frac{\mu \overline{m}_{ij}}{\rho_i \rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij}, \quad (75)$$

where $\overline{m}_{ij} = 0.5(m_i + m_j)$ is the average mass. This yields a symmetric linear system for the new velocities when applying the implicit Euler integration scheme

$$\mathbf{v}(t+\Delta t) = \mathbf{v}^* + \Delta t \mathbf{a}_{\text{visco}}(t+\Delta t). \quad (76)$$

This system can be solved efficiently by a matrix-free preconditioned conjugate gradient method.

Comparison All implicit viscosity solvers discussed above except the implicit Laplacian based solver use a strain rate based formulation where the tensor $\mathbf{E}$ (Eq. (66)) is determined by the SPH discretization of the velocity gradient (see Section 2.5):

$$\nabla \mathbf{v}_i = \frac{1}{\rho_i} \sum_j m_j (\mathbf{v}_j - \mathbf{v}_i) \nabla W_{ij}^T. \quad (77)$$

However, this discretization suffers from particle deficiency at the free surface of a fluid. Weiler et al. [WKBB18] showed that for a ro-

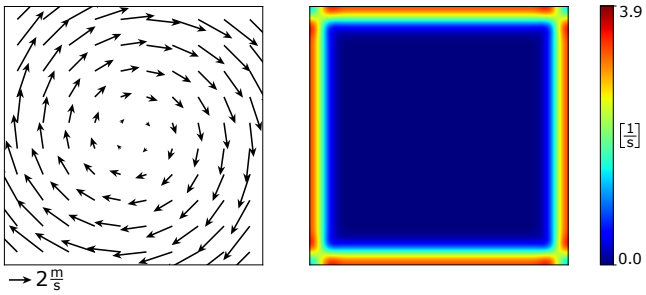

Figure 3: The velocity field of a rotational motion (left) should lead to a zero strain rate tensor. However, the SPH discretization in Eq. (77) causes errors at the surface due to particle deficiency as the Frobenius norm of the tensors shows (right) [WKBB18].

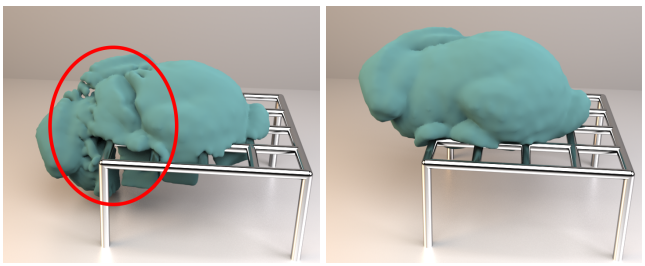

Figure 4: Left: The particle deficiency at the free surface leads to a wrong approximation of the strain rate tensor and therefore to artifacts. Right: Using the formulation of Weiler et al. [WKBB18] based on the Laplacian of the velocity field solves this problem.

tational motion, Eq. (77) leads to a significant error at the boundary (see Fig. 3). The strain rate based viscosity solvers counteract this erroneous strain rate at the boundary by computing forces which however cause visual artifacts (see Fig. 4, left) and a loss of angular momentum.

Weiler et al. use a formulation based on the Laplacian of the velocity field since the approximation of the Laplacian in Eq. (75) vanishes for rigid body rotations and conserves linear and angular momentum [Mon92]. In this way they avoid the issues of strain rate based methods (see Fig. 4, right).

While the methods of Takahashi et al. [TDF*15] and Weiler et al. [WKBB18] use a physically meaningful viscosity coefficient, the coefficients used in the other formulations are not physically motivated and depend on the temporal and spatial resolution.

In a performance comparison the solver of Peer et al. [PICT15] was the fastest method followed by the approach of Weiler et al. The other methods were significantly slower since they either use a Jacobi solver [BK17] or have to consider the second-ring neighbors [TDF*15].

5.2. Deformable Solids

In computer graphics most simulation methods for deformable solids are mesh-based, like the finite element method (FEM) [KBT17, KKB18] and Position-Based Dynamics (PBD) [BKCW14, BMM14, BMM17]. The advantage of using a meshless approach like SPH is that a unified representation for fluids and solids simplifies the coupling between the different materials and allows a simulation of state transitions like solidification or melting. A meshless representation of a solid can be obtained by sampling its geometry. Different sampling techniques are discussed in [KBFF*21].

5.2.1. Continuum Formulation

In the following we use a Lagrangian formulation where the equations of motion and the constitutive laws are described by positions $\mathbf{X}$ in the undeformed rest pose (reference configuration). The deformed configuration at time t is defined by the deformation mapping $\mathbf{x} = \mathbf{x}(\mathbf{X}, t)$ [Sif12]. Differentiating the deformation mapping w.r.t. the reference positions $\mathbf{X}$ yields the deformation gradient $\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}$ which can be used to define a constitutive model.

In the following we introduce different corotated SPH approaches which are either based on the linear or on the corotated linear constitutive model with the elastic energy densities [Sif12]

$$\Psi^{\text{linear}}(\mathbf{J}) = \mu \varepsilon : \varepsilon + \frac{\lambda}{2} \text{tr}^2(\varepsilon) \tag{78}$$

$$\Psi^{\text{corotated}}(\mathbf{J}) = \mu \|\mathbf{J} - \mathbf{R}\|_F^2 + \frac{\lambda}{2} \text{tr}(\mathbf{R}^T \mathbf{J} - \mathbb{1})^2, \tag{79}$$

where $\varepsilon = \frac{1}{2}(\mathbf{J} + \mathbf{J}^T) - \mathbb{1}$ is the linear infinitesimal strain tensor, μ and λ are the Lamé coefficients, $\mathbf{R}$ is the rotational part of $\mathbf{J}$, and $\| \cdot \|_F$ denotes the Frobenius norm. The total elastic energy is determined by an integral over the rest-pose domain Ω of the body

$$E = \int_\Omega \Psi(\mathbf{J}) dX. \tag{80}$$

Finally, the elastic body forces can be computed as the divergence of the first Piola–Kirchhoff stress tensor $\mathbf{f} = \nabla \cdot \mathbf{P}$, where $\mathbf{P}(\mathbf{J}) = \frac{\partial \Psi}{\partial \mathbf{J}}$.

5.2.2. SPH Discretization

The deformation of a deformable solid is computed with respect to its reference configuration and its topology does not change during the simulation. Therefore, the particle neighborhood $\mathcal{N}^0$ in reference configuration is used for the SPH discretization of the deformation gradient

$$\mathbf{J}_i(\mathbf{x}, \mathbf{X}) = \sum_{j \in \mathcal{N}_i^0} V_j^0 \mathbf{x}_{ji} \left(\mathbf{L}_i \nabla W \left(\mathbf{X}_{ij} \right) \right)^T, \tag{81}$$

where $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$, $\mathbf{X}_{ij} = \mathbf{X}_i - \mathbf{X}_j$, V^0 is the rest volume and $\mathbf{L}_i$ is a correction matrix to restore 1st-order consistency (cf. Section 2.3) that is defined as [BL99]

$$\mathbf{L}_i = \left(\sum_{j \in \mathcal{N}_i^0} V_j^0 \nabla W \left(\mathbf{X}_{ij} \right) \mathbf{X}_{ji}^T \right)^{-1}. \tag{82}$$

This matrix is required to correctly capture the rotational motion and can be precomputed at the beginning of the simulation. Note that the Moore–Penrose inverse can be used if the matrix is singular, *e.g.*, due to a collinear or coplanar particle configuration.

Corotated Linear Elasticity In computer graphics often the linear infinitesimal strain tensor is used to avoid the solution of a non-linear system in an implicit time integration step. However, this tensor is not invariant under rotations. A well-known solution is to use a corotational approach (*e.g.*, [KKB18]), where the rotation is extracted from the deformation gradient in order to compute the strain measure in an unrotated frame. An explicit corotational SPH approach was first introduced by Becker et al. [BIT09] to get a unified representation for fluids and solids. Since this approach is only conditionally stable, also implicit methods were investigated [PGBT17, KBFF*21].

The first step of a corotational SPH method is to extract the rotation $\mathbf{R}_i$ for each particle i from the kernel gradient $\mathbf{J}_i$ (see Eq. (81)), *e.g.*, by applying the efficient and stable method of Müller et al. [MBCM16]. Peer et al. [PGBT17] propose to rotate the reference configuration so that the displacement vector $\mathbf{u} = \mathbf{x} - \mathbf{RX}$ contains no rotation. The resulting corotated deformation gradient is given by

$$\mathbf{J}_i^*(\mathbf{x},\mathbf{X}) = \mathbb{1} + \sum_{j\in\mathcal{N}_i^0} V_j^0 \left(\mathbf{x}_{ji} - \mathbf{R}_i\mathbf{X}_{ji}\right)\left(\mathbf{R}_i\mathbf{L}_i\nabla W\left(\mathbf{X}_{ij}\right)\right)^T. \quad (83)$$

Peer et al. use the linear constitutive model (see Eq. (78)) to determine the corotated stress tensor $\mathbf{P}^* = \mathbf{P}(\mathbf{J}^*)$ and finally the elastic force [Gan15]

$$\mathbf{F}_i = V_i^0 \sum_{j\in\mathcal{N}_i^0} V_j^0 \left(\mathbf{P}_i^*\mathbf{R}_i\mathbf{L}_i\nabla W\left(\mathbf{X}_{ij}\right) - \mathbf{P}_j^*\mathbf{R}_j\mathbf{L}_j\nabla W\left(\mathbf{X}_{ij}\right)\right). \quad (84)$$

In contrast, Kugelstadt et al. [KBFF*21] directly substitute the deformation gradient $\mathbf{J}$ from Eq. (81) in the corotated linear constitutive model (see Eq. (79)) to obtain the stress tensor $\mathbf{P}^{\text{corotated}}(\mathbf{J})$. Finally, the elastic forces can be determined using Eq. (84).

5.2.3. Implicit Time Integration

Explicit time integration schemes are only conditionally stable and require small time steps when simulating stiff deformable solids. Therefore, Peer et al. [PGBT17] propose to use an implicit Euler integration scheme

$$\mathbf{v}^{t+\Delta t} = \mathbf{v}^t + \frac{\Delta t}{m}\mathbf{F}\left(\mathbf{J}^{t+\Delta t}\right), \quad (85)$$

where $\mathbf{J}^{t+\Delta t}$ is computed using $\mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t\mathbf{v}^{t+\Delta t}$. Since the elastic forces in Eq. (84) depend linearly on the particle positions, this yields a linear system for the new velocities. Peer et al. use a matrix-free conjugate gradient method to efficiently solve this system.

Kugelstadt et al. [KKB18, KBFF*21] split the corotated linear constitutive model (see Eq. (79)) in a stretching term $\mu\|\mathbf{J}-\mathbf{R}\|_F^2$ and a volume term $\frac{\lambda}{2}\text{tr}(\mathbf{R}^T\mathbf{J}-\mathbb{1})^2$ and keep the rotation constant during the step. The resulting stretching force linearly depends on $\mathbf{x}$ while the resulting volume force also depends on the rotation matrix $\mathbf{R}$. Hence, the linear system for the stretching term, which has to be solved in the implicit integration step, has a constant matrix. Therefore, its Cholesky factorization can be precomputed at the beginning of the simulation and the system can be solved very efficiently in each step. The volume conservation term is solved using a matrix-free conjugate gradient solver. Finally, this splitting approach is more than an order of magnitude faster than the method of Peer et al. However, it can only be applied if there are no topology changes during the simulation.

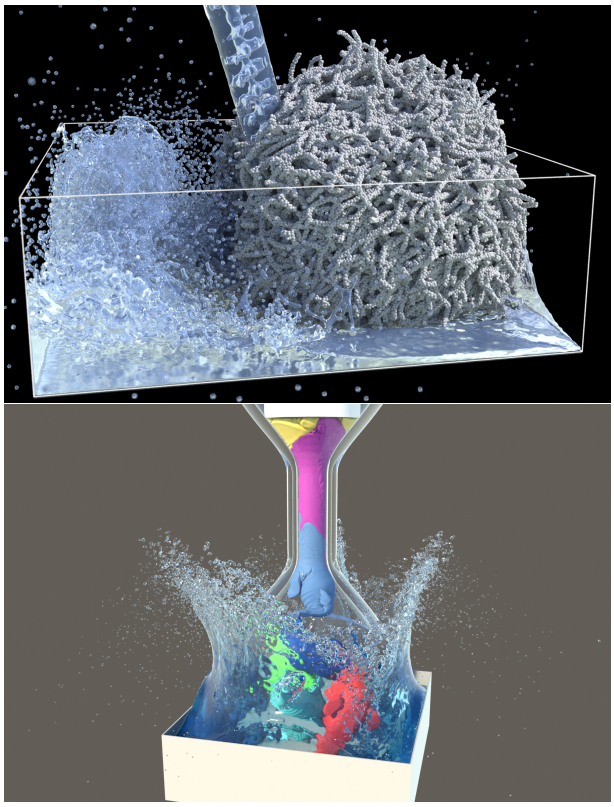

Figure 5: SPH simulation of deformable. Top: A deformable hairball interacts with water [PGBT17]. Bottom: Walrus models are pushed through a tight funnel [KBFF*21].

Simulations with both implicit methods are shown in Fig. 5.

5.2.4. Zero-Energy Mode Suppression

In the introduced SPH formulation for deformable solids only one deformation gradient $\mathbf{J}_i$ is computed to describe the deformation of the neighborhood of particle i. However, $\mathbf{J}_i$ can only represent a linear deformation and cannot capture non-linear deformations, *e.g.*, due to strong collision forces. Therefore, the non-linear part of the deformation does not contribute to the elastic energy and prevents that particles return to their rest-pose (hence the name zero-energy modes). Note that zero-energy modes are similar to hour glass modes in finite element methods [Gan15].

We can determine the deformed position $\mathbf{x}(\mathbf{X}_j)$ for a particle j, which is a neighbor of i, by a Taylor series

$$\mathbf{x}(\mathbf{X}_j) = \mathbf{x}(\mathbf{X}_i) + \mathbf{J}_i(\mathbf{X}_j - \mathbf{X}_i) + \mathbf{e}_{ij}^i, \quad (86)$$

where $\mathbf{e}_{ij}^i$ represents all higher-order terms. One approach to handle the zero-energy modes is to attempt to suppress the problematic local higher-order terms $\mathbf{e}_{ij}^i = \mathbf{J}_i\mathbf{X}_{ij} - \mathbf{x}_{ij}$.

Ganzenmüller [Gan15] adds an explicit penalty force to the sys-

tem in order to suppress the zero-energy modes:

$$\mathbf{F}_i^{\text{HG}} = -\frac{1}{2}\alpha k \sum_{j\in\mathcal{N}_i^0} V_i^0 V_j^0 \frac{W(\mathbf{X}_{ij})}{\|\mathbf{X}_{ij}\|^2}\left(\frac{\mathbf{e}_{ji}^i\cdot\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|}+\frac{\mathbf{e}_{ij}^j\cdot\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}\right)\frac{\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|}, \tag{87}$$

where the coefficient α controls the magnitude of the zero-energy mode suppression and k is the Young's modulus. However, the force minimizes $\mathbf{e}_{ij}^i$ only in the direction of the vector $\mathbf{x}_{ij}$. In contrast to that, Kugelstadt et al. [KBFF*21] propose an implicit zero-energy mode control by minimizing the quadratic energy function

$$E^{ZE} = \frac{\alpha'}{2}\sum_i \mu V_i^0 \sum_{j\in\mathcal{N}_i^0} V_j^0 \frac{\|\mathbf{e}_{ij}^i\|^2}{\|\mathbf{X}_{ij}\|^2} W_{ij}, \tag{88}$$

where α' is the zero-energy mode coefficient and μ the first Lamé coefficient. Since the energy is quadratic in $\mathbf{x}$, the corresponding stiffness matrix is constant which enables an efficient implicit Euler integration step by using a precomputed Cholesky factorization. Finally, the zero-energy mode suppression improves the stability of the simulation.

5.3. Rigid Solids

The simulation of rigid bodies with contact and friction is an active research topic in computer graphics [BET14]. Recently, it has been shown that rigid body contact handling can also be realized using an SPH approach [GPB*19]. This enables a unified SPH simulation of fluids, rigid bodies, deformable solids and highly viscous materials (see Fig. 7).

5.3.1. Rigid Body Solver

The SPH rigid body solver computes contact forces for each pair of colliding bodies using an approach which is similar to the concept of particle-based boundary handling (see Section 4). A particle sampling for each body surface is determined to obtain a unified particle representation of all objects in the simulation. Each sampling point defines a rigid body particle r. The artificial rest volume for r can be computed as [GPB*19]

$$V_r^0 = \frac{0.7}{\sum_{k\in\mathcal{R}} W_{rk}}, \tag{89}$$

where $\mathcal{R}$ denotes the set of all particles of rigid body R. This representation enables to detect rigid body contacts using the neighborhood search and to resolve contacts by an artificial pressure force.

To determine this force an artificial rest density $\rho_r^0 = 1$ is introduced for each rigid particle r, which defines the particle mass $m_r = V_r^0\rho_r^0$. Since we are only interested in a density deviation, the magnitude of ρ_r^0 can be chosen arbitrarily. To compute the deviation the density is determined as $\rho_r = \sum_k m_k W_{rk}$, where the sum considers the particles k of all rigid bodies within the support radius. A deviation of $\rho_r - \rho_r^0 > 0$ corresponds to an interpenetration which has to be resolved by a contact force $\mathbf{F}^{\text{rr}}$ that enforces $\rho_r = \rho_r^0$.

A linear system for the contact force is derived by first applying a backward difference time discretization to the continuity equation and considering the constant density constraint $\rho_r^{t+\Delta t} = \rho_r^0$:

$$\frac{\rho_r^0 - \rho_r}{\Delta t} = -\rho_r \nabla\cdot\mathbf{v}_r^{t+\Delta t}, \tag{90}$$

where $\mathbf{v}_r^{t+\Delta t} = \mathbf{v}_R^{t+\Delta t} + \omega_R^{t+\Delta t}\times\mathbf{r}_r^{t+\Delta t}$ is the rigid particle velocity at time $t+\Delta t$, $\mathbf{v}_R$ and ω_R are the linear and angular velocity of the rigid body R, respectively, and $\mathbf{r}_r$ is the vector from the center of mass of the rigid body to the position of the particle r. The velocities $\mathbf{v}_R^{t+\Delta t}$ and $\omega_R^{t+\Delta t}$ can be approximated by an Euler integration step considering all known external forces $\mathbf{F}_R$ and torques $\boldsymbol{\tau}_R$ as well as the unknown rigid-rigid contact forces $\mathbf{F}_r^{\text{rr}} = -\frac{m_r}{\rho_r}\nabla p_r$. Substituting the resulting velocities in Eq. (90) and introducing the approximation $\mathbf{r}_r^{t+\Delta t} = \mathbf{r}_r^t$ yields a linear system for the unknown contact pressure p_k

$$\rho_r\nabla\cdot\left(\Delta t\sum_{k\in\mathcal{R}}\frac{m_k}{\rho_k}\mathbf{K}_{rk}\nabla p_k\right) = \frac{\rho_r^0-\rho_r}{\Delta t} + \rho_r\nabla\cdot\mathbf{v}_r^s. \tag{91}$$

The matrix $\mathbf{K}_{rk}$, which is well-known in the area of rigid body solvers [Mir96, BET14], is defined as

$$\mathbf{K}_{rk} = \frac{1}{m_R}\mathbb{1} - \tilde{\mathbf{r}}_r\mathbf{I}_R^{-1}\tilde{\mathbf{r}}_k, \tag{92}$$

where m_R and $\mathbf{I}_R$ are mass and inertia tensor of rigid body R, respectively, and $\tilde{\mathbf{r}}_r$ is the cross product matrix of $\mathbf{r}_r$. The vector

$$\mathbf{v}_r^s = \mathbf{v}_R + \frac{\Delta t}{m_R}\mathbf{F}_R + \left(\omega_R + \Delta t\mathbf{I}_R^{-1}\left(\boldsymbol{\tau}_R + (\mathbf{I}_R\omega_R)\times\omega_R\right)\right)\times\mathbf{r}_r. \tag{93}$$

determines the new velocity of a particle r after a time step which considers all forces and torques except the unknown contact forces. After solving the linear system for the unknown pressure values, we can compute the contact forces as $\mathbf{F}_r^{\text{rr}} = -\frac{m_r}{\rho_r}\nabla p_r$.

To solve the linear system a relaxed Jacobi solver is applied which updates the pressure in iteration $l+1$ as

$$p_r^{l+1} = p_r^l + \frac{\beta_r^{\text{RJ}}}{b_r}\left(s_r - \rho_r\nabla\cdot\left(\Delta t\sum_{k\in\mathcal{R}} V_k\mathbf{K}_{rk}\nabla p_k^l\right)\right), \tag{94}$$

where b_r is the diagonal element of the linear system and β_r^{RJ} is the relaxation coefficient which is set to $\beta_r^{\text{RJ}} = \frac{0.5}{\text{num\_contacts}}$. The divergence on the right-hand side of the system (Eq. (91)) is computed using an SPH formulation

$$\nabla\cdot\mathbf{v}_r^s = \frac{1}{\rho_r}\sum_{k\in\mathcal{R}} V_k\rho_k\left(\mathbf{v}_k^s - \mathbf{v}_r^s\right)\cdot\nabla W_{rk}. \tag{95}$$

The linear system contains one equation for each rigid particle. However, the equations for all particles which have no contact can be removed to improve the performance.

The introduced rigid body solver is able to resolve thousands of simultaneous contacts accurately (see Fig. 6). More details about the derivation of the system and an extension to simulate friction effects can be found in [GPB*19].

5.4. Further Materials

In the previous sections we discussed SPH methods for highly viscous fluids, deformable solids and rigid bodies. However, the simulation with the SPH approach is an active research topic and also other material types have been investigated in recent years. It is out of scope of this paper to discuss all of them in detail but we want to at least mention some important ones. Alduán and Otaduy [AO11]

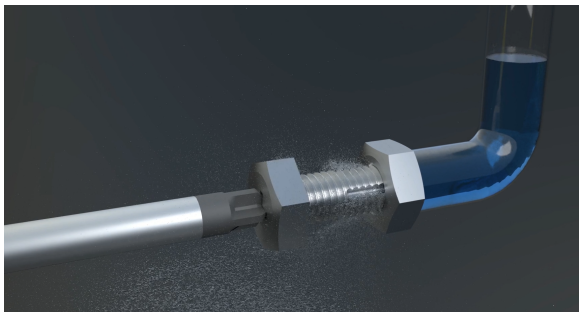

Figure 6: Simulation of a bolt which is loosened by a screwdriver while interacting with water particles.

introduced an SPH method for the simulation of granular materials which was later improved by Ihmsen et al. [IWT13]. The simulation of viscoelastic materials has been investigated by Takahashi et al. [TDFN14] and Barreiro et al. [BGFAO17]. Gissler et al. [GHB*20] introduced an SPH formulation for elastoplastic material behavior in order to simulate snow. Yan et al. [YJL*16] present a multi-phase method for fluids and solids. Finally, even for complex materials like ferrofluids SPH approaches were developed [HHM19].

5.5. Handling Material Interfaces

In the previous sections SPH formulations for different materials were introduced. Since all of them use the same discretization, it is easy to couple the models in a unified simulation. To determine contact forces at the interface between two materials, a common pressure solver is used. However, to consider the different material rest densities, a special handling at the interfaces is required. The standard SPH formulation computes a quantity at a position by interpolating the values of all adjacent particles. However, in case of different rest densities, the desired density discontinuity across the interface is smoothed due to this interpolation which leads to wrong density values at interface particles. This affects the pressure and force computation which manifests as spurious interface tension [Hoo98, AMS*07]. Large density ratios between the materials intensify the problem and significantly degrade the simulation stability regardless of the time step size.

In order to solve this problem an adapted SPH formulation to model density discontinuities across material interfaces was introduced based on the so-called number density $\delta_i = \sum_j W_{ij}$ [HA06, SP08]. This value is used to define an adapted particle density as

$$\tilde{\rho}_i = m_i \delta_i = m_i \sum_j W_{ij}. \tag{96}$$

The core idea of this adapted density computation is that each particle treats its neighbors as if they would have the same rest density as the particle itself. In this way only the geometric contribution W_{ij} of a neighboring particle j is considered but not its mass properties. The adapted density can then be used to obtain according pressure values and pressure forces.

Solenthaler et al. [SP08] replace the original SPH density ρ in the Tait equation by the adapted value $\tilde{\rho}$ to compute pressure values. This adapted equation of state has the following form

$$\tilde{p}_i = \frac{\kappa \rho^0}{\gamma} \left(\left(\frac{\tilde{\rho}_i}{\rho^0} \right)^\gamma - 1 \right), \tag{97}$$

where κ and γ are stiffness parameters. Substituting the adapted density and pressure in the symmetric SPH gradient formula (12) yields the adapted pressure force

$$\mathbf{F}_i^p = m_i \mathbf{a}_i^{\mathrm{p}} = -\frac{\nabla \tilde{p}_i}{\delta_i} = -\sum_j \left(\frac{\tilde{p}_i}{{\delta_i}^2} + \frac{\tilde{p}_j}{{\delta_j}^2} \right) \nabla W_{ij}, \tag{98}$$

where $\mathbf{a}_i^{\mathrm{p}} = -\frac{\nabla \tilde{p}_i}{m_i \delta_i}$ is the adapted pressure acceleration. Note that similar equations can be found in [TM05, HA06].

In subsequent works it was shown that this approach can also be used in combination with an implicit pressure solver [AIA*12, GPB*19]. Moreover, non-pressure forces can also be adapted using the number density. For example, this was demonstrated for elastic models [PGBT17, KBFF*21], highly viscous material models [PICT15, BKWK20] and solid-fluid coupling [AIA*12, GPB*19]. It has also been successfully applied in multi-scale SPH methods [SG11, HS13]. Finally, this enables a simple two-way coupling of fluids, highly viscous materials, deformable solids, and rigid bodies (see Fig. 7).

When applied to a single material, the derived equations are identical to the standard SPH formulation. However, for multiple materials the adapted forces eliminate spurious tension effects and increases stability. Note that these problems can also be avoided by advecting the density over time using the continuity equation (see Section 2.4) instead of performing the SPH density computation (Equation (5)) in each step. In this way the density computation does not suffer from smoothing artifacts across fluid interfaces. However, in case of density advection numerical integration errors sum up over time and cause a drift from true mass conservation when not preforming small time steps or applying higher-order time integration schemes [SP08, SB12].

6. Vorticity

The generation of realistic turbulent flows is a challenging research topic in SPH fluid simulations. Turbulent details quickly get lost due to a coarse sampling of the velocity field [IOS*14, CIPT14] or due to numerical diffusion [dGWH*15]. Therefore, in recent years several approaches have been investigated to facilitate the formation of vortices and to counteract numerical diffusion [MM13, BKKW18, WLB*20, LWB*21]. In the following we introduce two different approaches in detail.

Vorticity Confinement Turbulent motions are largely caused by the interaction of unsteady vortices on various scales. Macklin and Müller [MM13] counteract the dissipation in SPH fluid simulation by amplifying existing vortices using a vorticity confinement approach. The vorticity, which describes the local spinning motion, is defined by the vector field $\omega = \nabla \times \mathbf{v}$. The SPH discretization of this vector field determines the vorticity ω_i for each particle i as

$$\omega_i = \nabla \times \mathbf{v}_i = -\sum_j \frac{m_j}{\rho_j} \mathbf{v}_{ij} \times \nabla W_{ij}. \tag{99}$$

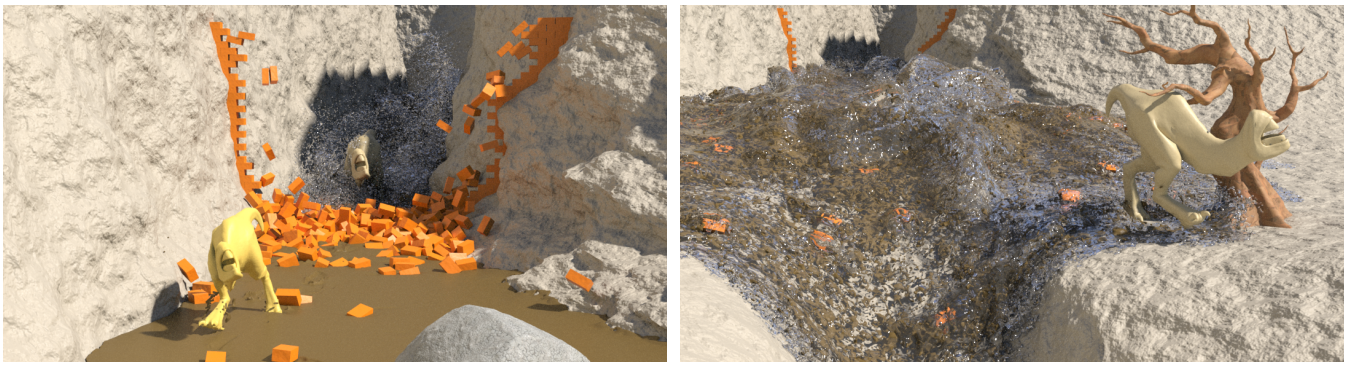

Figure 7: A unified SPH solver enables complex simulations with different materials. In this example two creatures run in highly viscous mud, break through a wall of rigid bodies and collide with a deformable tree while water is flowing down the canyon.

To amplify existing vortices a force is applied

$$\mathbf{F}_i^{\text{vorticity}} = \varepsilon^{\text{v}} \left(\frac{\eta}{\|\eta\|} \times \omega_i \right), \quad \eta = \sum_j \frac{m_j}{\rho_j} \|\omega_j\| \nabla W_{ij}, \tag{100}$$

where ε^{v} is a user-defined parameter which controls the amplification. Finally, the velocity field is smoothed using XSPH [Mon92] to get a coherent particle motion.

Vorticity Confinement is simple to implement and effectively amplifies existing vortices. However, the parameter ε^{v} must be chosen carefully to avoid an overamplification and therefore an energy gain. Moreover, the method only amplifies existing vortices but does not facilitate the formation of new ones.

Micropolar Model In general a micropolar fluid has a non-symmetric stress tensor and the model assumes that the infinitesimally small particles which compose a fluid continuum are rotationally invariant [Łuk99]. In comparison to a Newtonian fluid, a micropolar fluid additionally models a rotational motion using an angular velocity field due to the non-symmetric stress measure. The additional rotational degrees of freedom facilitate the generation of vortices and a wider range of potential dynamic effects are captured by the model.

Bender et al. [BKKW18] introduce a micropolar model that generalizes the Navier-Stokes model to simulate incompressible, inviscid turbulent flow. They define a non-symmetric stress tensor

$$\mathbf{T} = -p\mathbb{1} - \mu_t \nabla \mathbf{v} + \mu_t \omega^{\times}, \tag{101}$$

where $\omega^{\times} = \sum_i \varepsilon_{jik} \omega_i$, ε_{ijk} is the Levi-Civita tensor, and μ_t denotes the *transfer coefficient*. Note that in this section we neglect any dissipation term, such as viscosity, and focus on the generation of undamped, highly turbulent flows.

Substituting the stress tensor in the conservation laws for linear and angular momentum in combination with the incompressibility condition (19) yields the equations of motion:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu_t \nabla \times \omega + \frac{\mathbf{f}_{\text{ext}}}{\rho} \tag{102}$$

$$\Theta \frac{D\omega}{Dt} = \nu_t (\nabla \times \mathbf{v} - 2\omega) + \frac{\tau_{\text{ext}}}{\rho}, \tag{103}$$

where $\nu_t \geq 0$ is the kinematic transfer coefficient, τ_{ext} the external body torque, and Θ a scalar, isotropic microinertia coefficient. Bender et al. suggest to set $\Theta = 2\text{m}^2\,\text{s}^{-1}$ based on experimentation. Eq. (102) is identical to the inviscid Navier-Stokes equation but augmented by the term $\nu_t \nabla \times \omega$. This term and the complementary term $\nu_t(\nabla \times \mathbf{v} - 2\omega)$ in Eq. (103) effectively convert angular accelerations into linear accelerations and vice versa. These terms can be physically interpreted as a model for dissipation-free friction between the infinitesimal material particles in the fluid which couples the linear and rotational motion.

To compute the transfer forces $\mathbf{F}_i^{\text{transfer}} = m_i \nu_t \nabla \times \omega_i$ and torques $\boldsymbol{\tau}_i^{\text{transfer}} = m_i \nu_t (\nabla \times \mathbf{v}_i - 2\omega_i)$ an SPH discretization is applied using the difference curl formulation [Mon92]:

$$\nabla \times \mathbf{A}_i \approx \frac{1}{\rho_i} \sum_j m_j (\mathbf{A}_i - \mathbf{A}_j) \times \nabla W_{ij}. \tag{104}$$

These forces are then handled as non-pressure forces and integrated explicitly as they are considerably less stiff than the pressure forces. Moreover, the vorticity ω has to be stored in each particle and is also integrated explicitly:

$$\omega_i(t + \Delta t) = \omega_i(t) + \frac{\Delta t}{m_i \Theta_i} (\boldsymbol{\tau}_i^{\text{transfer}} + \boldsymbol{\tau}_i^{\text{ext}}). \tag{105}$$

Finally, it is recommended to smooth the resulting linear and angular velocity fields, *e.g.*, using XSPH [Mon92], to ensure coherent particle motion.

Figure 8 demonstrates the effect of an increasing transfer coefficient in a simulation with three obstacles that cause turbulences. A lid-driven cavity experiment (see Fig. 9) shows the difference between vorticity confinement and the micropolar model. In the experiment the "lid" (top-side) of a two-dimensional cavity, which is filled with water, is accelerated with a constant velocity. The velocity field is expected to stabilize in a big central vortex and three minor vortices rotating in the opposite direction. While vorticity confinement derives the vorticity from the linear velocity field and is only able to amplify the existing central vortex, the micropolar approach discretizes the linear and angular velocity field independently and couples both fields via the transfer terms. This conserves existing vortices, facilitates the formation of new vortices, and yields the expected result in the lid-driven cavity experiment.

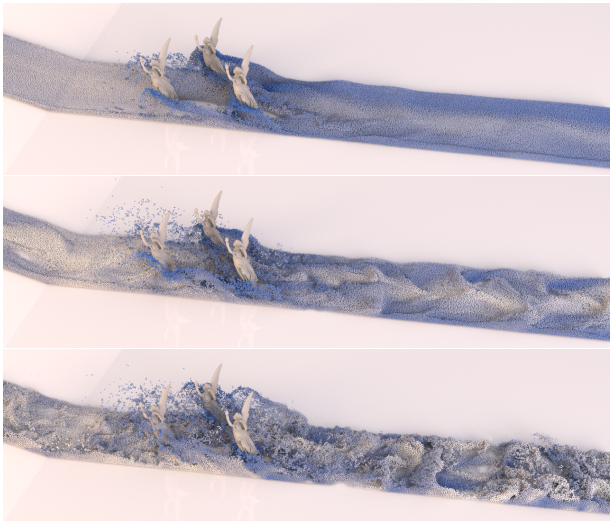

Figure 8: Simulation of 4.7M fluid particles with three obstacles using the micropolar model with increasing transfer coefficient ν_t. Top-down: $\nu_t = 0.2\text{m}^2\,\text{s}^{-1}$, $\nu_t = 0.3\text{m}^2\,\text{s}^{-1}$, $\nu_t = 0.4\text{m}^2\,\text{s}^{-1}$.

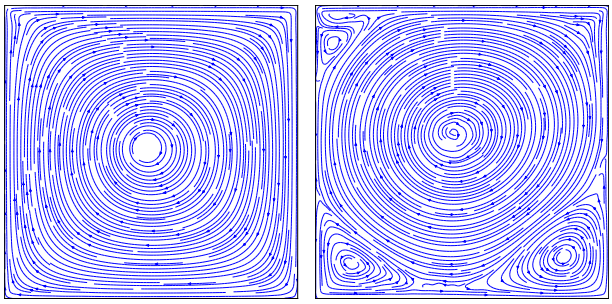

Figure 9: Velocity fields of the lid-driven cavity benchmark. Vorticity confinement (left) is only able to generate one large central vortex. In contrast, the micropolar approach (right) produces the expected result, *i.e.*, one central vortex and three smaller vortices in the corners which are rotating in the opposite direction.

7. Data Driven Fluid Simulation

Combining classical numerical simulation techniques with machine learning methods – and in particular deep neural networks – is a very active and rapidly growing research field. Different approaches can be followed to embed domain knowledge in the form of physical models into the learning, either separately or in tandem. Karniadakis et al. [KKL*21] distinguishes between observational, learning and inductive biases that can steer the learning process towards identifying physically consistent solutions. Observational biases can be introduced directly through data (real or simulated), which corresponds to the classic machine learning approach. A large volume of data is typically necessary to reinforce the biases and to generate predictions that respect physical principles. Learning biases are introduced over the loss functions (soft constraints), favoring convergence towards solutions that adhere to the underlying physics. Inductive biases require the neural network architecture to encode specific physical properties such that they are implicitly satisfied (hard constraints). Thuerey et al. [THM*21] as well as Wang and Yu [WY21] additionally list the class of hybrid approaches, where a traditional physics solver is coupled with an output from a deep neural network. This requires the physics solver to be fully differentiable and represents the tightest coupling between simulation and learning. The scientific literature on physics-based deep learning is rapidly growing across different disciplines, and we refer the interested reader to the excellent survey article of Karniadikis et al. [KKL*21] and the digital book of Thuerey et al. [THM*21]. In the following, we focus our survey on particle simulations only.

7.1. Particle State Prediction with Regression Forests

The seminal work of Ladický et al. [LJS*15] introduced a surrogate model for SPH by employing Regression forests to infer the system's state, eliminating the expensive computation of numerical approximations of the PDEs. For each particle a feature vector $\Phi_{\mathbf{x}_i}$ is evaluated and serves as the input to the regressor. The features are designed such that they represent the individual forces and constraints of the Navier-Stokes equations and model pressure, incompressibility, viscosity and surface tension. Features are computed as flat-kernel sums of rectangular regions surrounding a particle, and different box sizes capture both close and distant neighbors. The regression problem is formulated as $\mathbf{a}_i(t) := Reg(\Phi_{\mathbf{x}_i^*})$, where $Reg(.)$ is the learned regression function and $\mathbf{x}_i^*$ is the particle position after adding external forces, advection and collision handling. Using this intermediate particle state has two advantages: First, it enables the regressor to predict compression and hence to counteract those with a corrective acceleration or velocity (conceptually mimicking PCISPH [SP09] or PBF [MM13]). Second, arbitrary external forces (such as surface tension, friction, drag) can be added at test time without requiring model retraining. This regression fluid approach enables the simulation of over a million particles in real time.

7.2. Lagrangian Fluids with Convolutional Neural Networks

The work of Tumanov et al. [TKC21] follows the idea of Ladický et al. [LJS*15] to regress corrective values to speed up particle-based fluid simulations. Instead of using a decision tree based model that requires hand-crafted features, a neural network regressor is used instead. The regression is formulated such that it substitutes the expensive iterative Jacobi-style method to enforce incompressibility in PBF [MM13]. The intermediate particle states and solid obstacles are first rasterized onto a Cartesian grid to enable the use of 3D convolutional networks. The deep learning network then computes a latent representation of this data on the grid, and either a fully connected network or sub-pixel convolutions are used to obtain the final velocity correction for each particle. With this approach, large particle numbers running at high frame rates have been achieved. Although being slightly slower than a decision tree driven solver [LJS*15], the method clearly outperforms previous network-based approaches (*e.g.*, [UPTK20, SGGP*20]) in terms of particle count and computational speed.

Ummenhofer et al. [UPTK20] use continuous convolutions that are repeatedly applied to particles to create a latent space at each location. After expanding the size of the latent space over the course

of a few layers, it is contracted again to produce the desired result, *e.g.*, an acceleration or velocity correction. The convolution is evaluated on a set of particle positions $\mathbf{x}_i$ in a radial neighborhood as $(f*g)(\mathbf{x}) = \sum_i f(\mathbf{x}_i) g(\Lambda(\mathbf{x}_i - \mathbf{x}))$. Λ represents a mapping from the unit ball to the unit cube and plays a central role, as it allows the use of a simple grid to represent the unknowns in the convolutional kernel. A radial weighting function was additionally added to ensure a smooth kernel falloff and hence that the learned influence smoothly drops to zero for each of the individual convolutions. The fluid simulation network is trained in a supervised fashion based on particle trajectories produced by a physics simulation. The loss function combines two subsequent time steps, and parameters account for neighbor deficiency and small motions. The method achieves a high level of accuracy compared to other data-driven approaches. It was reported to outperform a graph-based network approach [LWT*19] while having a reduced implementation complexity.

7.3. Graph Network-based Simulations

Graph networks and their variants (*e.g.*, interaction networks) can learn the dynamics of various material types. A Hierarchical Relation Network based on hierarchical graph convolution was used in [MZW*18] to simulate elastic bodies represented by particles. A Dynamic Particle Interaction network was employed in [LWT*19] for solid objects and fluids, reporting superior performance compared to the aforementioned work. Sanchez et al. [SGGP*20] introduced a Graph Network-based Simulator, which leverages that particle-based simulation can be viewed as message-passing on a graph. The nodes correspond to particles, and the edges represent pairwise relations between particles, over which interactions are computed. Applied to SPH, messages passed between nodes correspond, for example, to the pressure computation using the smoothed density. Since this neural architecture models physical states and interactions, it imposes a strong inductive bias. An encoder $\mathcal{X} \rightarrow \mathcal{G}$ is used to embed the particle-based state representation as a latent graph; a processor $\mathcal{G} \rightarrow \mathcal{G}$ then computes interactions among nodes to generate a sequence of updated learned latent graphs; and a decoder $\mathcal{G} \rightarrow \mathcal{Y}$ extracts the dynamics information, such as accelerations, from the nodes of the final latent graph. The presented examples demonstrate that the Graph Network Simulation architecture trained on data generated with SPH, PBD and MPM can successfully learn the dynamics of different materials up to a resolution of ~20K particles in 3D. In general, Graph Network Simulations come at a high level of generality with respect to material types, while other neural methods are typically tailored towards fluid dynamics. The architecture was extended in [LF20] by introducing two types of graph neural networks, a node-focused graph network to predict advection and projection, and an edge-focused graph network to model an elastic collision.

7.4. Differentiable Solvers and Neural Networks

Differentiable physics solvers have proven to be very powerful in various control task problems. These optimization tasks require the computation and backpropagation of analytical gradients of a simulation step. By implementing a physics solver as a neural network prediction this functionality is intrinsically given. This approach was applied to PBF fluids by Schenk and Fox [SF18] to optimize fluid parameters from data and to control liquids. Neural network controllers can also be seamlessly combined with a traditional physics solver, requiring the solver to be fully differentiable. While auto-differentiation frameworks (such as Theano, TensorFlow, PyTorch) can be used, the work of Hu et al. [HAL*20] is particularly noteworthy as their programming language is tailored for high-performance differentiable physics simulations, and was used, among others, in conjunction with MPM simulations. Artistic manipulation of fluids, and in particular neural style transfer from images to smoke, was achieved in Kim et al. [KAGS20] by coupling a differentiable fluid solver with a pre-trained image classification network. The optimization computes a control velocity, such that the fluid density is transported into the desired target configuration. The Lagrangian representation of the fluid enables temporally smooth stylization results, but requires particle-to-grid operations in order to couple it with the pre-trained network. Results show that arbitrary structures given by 2D input images can be transferred coherently in space and time onto 3D smoke simulations with a Lagrangian formulation.

8. Conclusion

In this survey, we have summarized and discussed developments in the field of SPH simulations in the graphics community. We showed that with the recent improvements, the methods have ultimately emerged as a significant technology to enrich virtual worlds with physical behavior and their wide adoption has demonstrated its impact on a vast number of application domains. Many long-standing challenges have been overcome such that a wide variety of materials are simulatable while the improvements in robustness and efficiency allow us to simulate millions of particles on a single desktop computer as of today. Nevertheless, there are still many open problems to be solved and challenges to be overcome which we will outline in the following.

Approximation Quality and its Implications A long standing problem in the domain of SPH simulation is the degradation of approximation quality in spatial regions where the particle count is low. This phenomenon is often referred to as the *particle deficiency problem*. Even though numerical accuracy is rarely the first priority in the context of graphics applications, low-quality approximations often have an immense effect on the visual quality of the simulation outcome. Those particle deficiencies are naïvely unavoidable near free surfaces. Numerically, this effect leads to an underestimation of physical quantities such as mass density, forces, vorticity, *etc*. One of the main consequences of this underestimation is that negative pressures are usually clamped to avoid particle clumping near the free-surface (*cf.*, Section 3) and, hence, projecting iterative solvers, *e.g.*, the projected Jacobi or Gauss-Seidel iterations, have to be employed while more efficient solvers such as the Conjugate Gradient Method can not be used. While some solutions to this problem exist, they are often either computationally intensive, *e.g.*, [SB12], or their lack of accuracy makes them insufficient, *e.g.*, using Shepherd filters. Practical yet sufficiently accurate solutions have still to be found.

Unified Solver and Ultimate Coupling While there have been works focusing on the coupling between SPH solvers simulating dedicated materials, *e.g.*, [GBP*17], many solvers are highly

specialized to model certain phenomena, *e.g.*, [HHM19], and it is therefore non-trivial to incorporate them into existing implementations. The design of a common framework that accommodates the specialized requirements of existing solvers which also scales well with particle resolution is still a matter of research.

Artist Control As discussed in this survey the community has developed methods to simulate a wide variety of different materials. However, their behavior is in most cases solely governed by physical laws while many graphics applications require artists to control and edit the behavior to achieve the desired visual effects. Although there is high-demand from industry, work on artist control still seems to be niche and so far only a few SPH-specific papers have been published, *e.g.*, [ZYWL15, LCY*19, SPDM20].

Data-driven Solvers Massive advances have been made in the field of machine-learning but, so far, the body of work on SPH solvers is relatively small. Given the existing work discussed in this survey, there is good potential to enhance existing solvers especially with respect to artist control and stylization but potentially also to contribute to any other subdomain such as material modeling, neighborhood search, vorticity improvement *etc*.

We hope that this survey helps to explain the basics as well as the most recent advances of the SPH-related research to a larger audience in research and industry and thus facilitate future works to build on the foundation which has been layed so far.

References

[AHA12] ADAMI S., HU X., ADAMS N. A.: A generalized wall boundary condition for Smoothed Particle Hydrodynamics. *Journal of Computational Physics 231*, 21 (2012), 7057–7075. 11

[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. 9, 10, 17

[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. 17

[AO11] ALDUÁN I., OTADUY M. A.: SPH granular flow with friction and cohesion. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM Press. 16

[Ben22] BENDER J.: SPlisHSPlasH Library. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH, 2022. 23

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 16

[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. 13, 17

[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. 6, 9, 10, 11

[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. 6, 9, 10, 11, 12

[BIT09] BECKER M., IHMSEN M., TESCHNER M.: Corotated SPH for deformable solids. In *Proceedings of Eurographics Conference on Natural Phenomena* (2009), pp. 27–34. 15

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

[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. 5, 9, 12, 13, 14

[BKCW14] BENDER J., KOSCHIER D., CHARRIER P., WEBER D.: Position-Based Simulation of Continuous Materials. *Computers & Graphics 44*, 1 (2014), 1–10. 14

[BKKW18] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.: Turbulent micropolar SPH fluids with foam. *IEEE Transactions on Visualization and Computer Graphics* (2018). 17, 18

[BKWK19] BENDER J., KUGELSTADT T., WEILER M., KOSCHIER D.: Volume maps: An implicit boundary representation for SPH. In *Proceedings of ACM SIGGRAPH Conference on Motion, Interaction and Games* (2019), MIG '19, ACM. 9

[BKWK20] BENDER J., KUGELSTADT T., WEILER M., KOSCHIER D.: Implicit frictional boundary handling for SPH. *IEEE Transactions on Visualization and Computer Graphics 26*, 10 (2020), 2982–2993. 9, 10, 11, 17

[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. 14

[BLS12] BODIN K., LACOURSIÈRE C., SERVIN M.: Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18* (2012), 516–526. 10

[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. 14

[BMM17] BENDER J., MÜLLER M., MACKLIN M.: A Survey on Position Based Dynamics, 2017. In *EUROGRAPHICS 2017 Tutorials* (2017), Eurographics Association. 14

[Bri15] BRIDSON R.: *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis, 2015. 4, 5

[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. 4, 12

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 1–8. 6

[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). 7, 9

[CIPT14] CORNELIS J., IHMSEN M., PEER A., TESCHNER M.: IISPH-FLIP for incompressible fluids. *Computer Graphics Forum 33*, 2 (may 2014), 255–262. 17

[CM99] CLEARY P. W., MONAGHAN J. J.: Conduction Modelling Using Smoothed Particle Hydrodynamics. *Journal of Computational Physics 148*, 1 (1999), 227 – 264. 12

[CR99] CUMMINS S. J., RUDMAN M.: An SPH Projection Method. *Journal of Computational Physics 152* (1999), 584–607. 6

[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. 17

[ER03] ESPANOL P., REVENGA M.: Smoothed dissipative particle dynamics. *Physical Review E 67*, 2 (2003), 026705. 12

[FIF18] FIFTY2 TECHNOLOGY: PreonLab. www.fifty2.eu, 2018. 24

[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. 9, 10

[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. 12

[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. 15

[GBP*17] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Approximate air-fluid interactions for SPH. In *Virtual Reality Interactions and Physical Simulations* (Apr. 2017), pp. 1–10. 20

[GHB*20] GISSLER C., HENNE A., BAND S., PEER A., TESCHNER M.: An implicit compressible SPH solver for snow simulation. *ACM Transactions on Graphics 39*, 4 (Aug. 2020), 1–16. 17

[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. 3

[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. 9, 10, 16, 17

[HA06] HU X., ADAMS N.: A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics 213*, 2 (2006), 844–861. 17

[HAL*20] HU Y., ANDERSON L., LI T.-M., SUN Q., CARR N., RAGAN-KELLEY J., DURAND F.: DiffTaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations (ICLR)* (2020). 20

[HEW15] HUBER M., EBERHARDT B., WEISKOPF D.: Boundary Handling at Cloth-Fluid Contact. *Computer Graphics Forum 34*, 1 (2015), 14–25. 10

[HHM19] HUANG L., HÄDRICH T., MICHELS D. L.: On the accurate large-scale simulation of ferrofluids. *ACM Transactions on Graphics 38*, 4 (July 2019), 93:1–93:15. 17, 21

[HKK07a] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed Particle Hydrodynamics in complex shapes. In *Spring Conference on Computer Graphics* (2007), pp. 191–197. 10, 11

[HKK07b] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed Particle Hydrodynamics on GPUs. In *Computer Graphics International* (2007), pp. 63–70. 10, 11

[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. 6

[Hoo98] HOOVER W.: Isomorphism linking smooth particles and embedded atoms. *Physica A: Statistical Mechanics and its Applications 260*, 3 (1998), 244–254. 17

[HS13] HORVATH C. J., SOLENTHALER B.: Mass preserving multi-scale SPH. Pixar Technical Memo 13-04, Pixar Animation Studios, 2013. 17

[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. 5

[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. 10

[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. 5, 6, 7

[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. 1, 3, 5, 12, 17

[IWT13] IHMSEN M., WAHL A., TESCHNER M.: A Lagrangian framework for simulating granular material with high detail. *Computers & Graphics 37*, 7 (nov 2013), 800–808. 17

[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. 12

[KAGS20] KIM B., AZEVEDO V. C., GROSS M., SOLENTHALER B.: Lagrangian neural style transfer for fluids. *ACM Transactions on Graphics 39*, 4 (2020). 20

[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. 9, 10, 11

[KBFF*21] KUGELSTADT T., BENDER J., FERNÁNDEZ-FERNÁNDEZ J. A., JESKE S. R., LÖSCHNER F., LONGVA A.: Fast corotated elastic SPH solids with implicit zero-energy mode control. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 3 (2021). 14, 15, 16, 17

[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. 14

[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. 11

[KKB18] KUGELSTADT T., KOSCHIER D., BENDER J.: Fast corotated FEM using operator splitting. *Computer Graphics Forum 37*, 8 (2018). 14, 15

[KKL*21] KARNIADAKIS G., KEVREKIDIS Y., LU L., PERDIKARIS P., WANG S., YANG L.: Physics-informed machine learning. *Nat Rev Phys 3* (2021), 422–440. 19

[Lau11] LAUTRUP B.: *Physics of Continuous Matter*. Taylor & Francis, 2011. 12

[LCY*19] LU J.-M., CHEN X.-S., YAN X., LI C.-F., LIN M., HU S.-M.: A rigging-skinning scheme to control fluid simulation. *Computer Graphics Forum 38*, 7 (oct 2019), 501–512. 21

[LF20] LI Z., FARIMANI A. B.: Accelerating Lagrangian fluid simulation with graph neural networks. In *ICLR 2021 SimDL Workshop* (2020). 20

[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. 19

[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

[Łuk99] ŁUKASZEWICZ G.: *Micropolar Fluids*. Modeling and Simulation in Science, Engineering and Technology. Birkhäuser Boston, 1999. 18

[LWB*21] LIU S., WANG X., BAN X., XU Y., ZHOU J., KOSINKA J., TELEA A. C.: Turbulent details simulation for SPH fluids via vorticity refinement. *Computer Graphics Forum 40*, 1 (2021), 54–67. 17

[LWT*19] LI Y., WU J., TEDRAKE R., TENENBAUM J. B., TORRALBA A.: Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations (ICLR)* (2019). 20

[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. 15

[Mir96] MIRTICH B. V.: *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California at Berkeley, 1996. 16

[MM13] MACKLIN M., MÜLLER M.: Position Based Fluids. *ACM Transactions on Graphics 32*, 4 (2013), 1–5. 8, 17, 19

[Mon92] MONAGHAN J.: Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 5, 12, 14, 18

[Mon05] MONAGHAN J. J.: Smoothed Particle Hydrodynamics. *Reports on Progress in Physics 68*, 8 (2005), 1703–1759. 2, 11, 12

[MZW*18] MROWCA D., ZHUANG C., WANG E., HABER N., FEI-FEI L., TENENBAUM J. B., YAMINS D. L.: Flexible neural representation for physics prediction. In *Neural Information Processing Systems (NIPS)* (2018). 20

[PGBT17] PEER A., GISSLER C., BAND S., TESCHNER M.: An implicit SPH formulation for incompressible linearly elastic solids. *Computer Graphics Forum* (2017). 15, 17

[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. 12, 13, 14, 17

[Pri12] PRICE D. J.: Smoothed Particle Hydrodynamics and Magnetohydrodynamics. *Journal of Computational Physics 231*, 3 (Feb. 2012), 759–794. 2, 3, 4, 12

[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. 12, 13

[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. 3

[SB12] SCHECHTER H., BRIDSON R.: Ghost SPH for animating water. *ACM Transactions on Graphics 31*, 4 (2012), 61:1–61:8. 5, 17, 20

[SF18] SCHENK C., FOX D.: SPNets: Differentiable fluid dynamics for deep neural networks. In *Proceedings of the Second Conference on Robot Learning (CoRL)* (2018). 20

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *TOG 30*, 4 (2011), 72:1–72:8. 17

[SGGP*20] SANCHEZ-GONZALEZ A., GODWIN J., PFAFF T., YING R., LESKOVEC J., BATTAGLIA P. W.: Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICLM))* (2020). 19, 20

[Sif12] SIFAKIS E.: *SIGGRAPH 2012 Course Notes FEM Simulation of 3D Deformable Solids Part 1*. Tech. rep., University of Wisconsin-Madison, 2012. 14

[SP08] SOLENTHALER B., PAJAROLA R.: Density Contrast SPH Interfaces. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 211–218. 17

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (2009), 40:1–40:6. 5, 7, 19

[SPDM20] SCHOENTGEN A., POULIN P., DARLES E., MESEURE P.: Particle-based liquid control using animation templates. *Computer Graphics Forum 39*, 8 (nov 2020), 79–88. 21

[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. 12, 14

[TDFN14] TAKAHASHI T., DOBASHI Y., FUJISHIRO I., NISHITA T.: Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections. *The Visual Computer* (2014). 17

[THM*21] THUEREY N., HOLL P., MUELLER M., SCHNELL P., TROST F., UM K.: Physics-based deep learning. https://physicsbaseddeeplearning.org, 2021. 19

[TKC21] TUMANOV E., KOROBCHENKO D., CHENTANEZ N.: Data-driven particle-based liquid simulation with deep learning utilizing sub-pixel convolution. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 1 (2021). 19

[TM05] TARTAKOVSKY A., MEAKIN P.: Modeling of surface tension and contact angles with Smoothed Particle Hydrodynamics. *Physical Review E 72*, 2 (2005), 026301. 17

[UPTK20] UMMENHOFER B., PRANTL L., THUEREY N., KOLTUN V.: Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations (ICLR)* (2020). 19

[WAK20] WINCHENBACH R., AKHUNOV R., KOLB A.: Semi-analytic boundary handling below particle resolution for Smoothed Particle Hydrodynamics. *ACM Transactions on Graphics 39*, 6 (nov 2020), 1–17. 10

[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. 12

[WKBB18] WEILER M., KOSCHIER D., BRAND M., BENDER J.: A physically consistent implicit viscosity solver for SPH fluids. *Computer Graphics Forum 37*, 2 (2018). 12, 13, 14

[WLB*20] WANG X., LIU S., BAN X., XU Y., ZHOU J., KOSINKA J.: Robust turbulence simulation for particle-based fluids using the Rankine vortex model. *The Visual Computer 36* (2020), 2285–2298. 17

[WY21] WANG R., YU R.: Physics-guided deep learning for dynamical systems: A survey. arXiv, 2021. 19

[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. 17

[ZYWL15] ZHANG S., YANG X., WU Z., LIU H.: Position-based fluid control. In *Symposium on Interactive 3D Graphics and Games* (feb 2015), ACM. 21

Appendix A: Biographies

Dan Koschier has been a research associate in the Smart Geometry Processing group at University College London until 2019 and has since moved on to work in the technology industry. He received his PhD in Computer Science from RWTH Aachen University in 2018. His research interests include physically-based simulation of deformable solids, cutting, fracture, fluids, machine learning aided physical simulations as well as character animation.

Jan Bender is professor of computer science and leader of the Computer Animation Group at RWTH Aachen University. He received his diploma, PhD and habilitation in computer science from the University of Karlsruhe. His research interests include interactive simulation methods, multibody systems, deformable solids, fluid simulation, collision handling, cutting, fracture, GPGPU and real-time visualization. He serves on program committees of major graphics conferences, has been program chair of VRIPHYS, VMV and ACM SIGGRAPH / Eurographics SCA and associate editor for IEEE Computer Graphics and Applications. Finally, he is the main developer of SPlisHSPlasH [Ben22], an open-source library for the physically-based SPH simulation of fluids and solids.

Barbara Solenthaler is a senior research scientist at the Computer Science department at ETH Zurich, where she leads the research on simulation and animation. Prior to joining ETH, she received her Ph.D. in Computer Science from the University of Zurich. In her research she develops algorithms and techniques for data-driven physics simulations, where a particular challenge is to synergistically combine machine learning with 3D modeling and the laws of

dynamics. Her research includes efficient reconstruction of simulations, image based modeling, and artist-controllable simulations, aiming at transforming and simplifying workflows in visual effects and medical fields. Barbara is currently also affiliated with the Institute for Advanced Study at the Technical University of Munich, where she holds a Hans Fischer Fellowship awarded by the Siemens AG on digital twin technologies. Barbara serves on various technical program and organization committees of major graphics conferences, is appointed as an Associate Editor of Computer Graphics Forum, and is a co-founder of Apagom AG that provides a real-time fluid engine using machine learning.

Matthias Teschner is professor of Computer Science and head of the Computer Graphics group at the University of Freiburg. He received the PhD degree in Electrical Engineering from the University of Erlangen-Nuremberg in 2000. From 2001 to 2004, he was research associate at Stanford University and at the ETH Zurich. His research interests comprise physically-based simulation and rendering. Recent research focuses on the development of cutting-edge technology for Lagrangian simulations which are applied in engineering, entertainment technology, art, computational medicine and robotics. In 2015, Matthias Teschner has been a co-founder of FIFTY2 Technology, where the SPH-based Preon solver [FIF18] is being developed. He serves on program committees of major graphics conferences. He serves or has served as an associate editor for Computers & Graphics (2009 -), Computer Graphics Forum (2011 - 2014) and IEEE Transactions on Visualization and Computer Graphics (2018 - 2022). He was conference co-chair of ACM SIGGRAPH/Eurographics SCA 2016 and program co-chair of VMV 2019. He received the highly prestigious Günter Enderle Award at Eurographics 2014 for his research on fluid simulation.