Consistent SPH Rigid-Fluid Coupling

Jan Bender ![ORCID], Lukas Westhofen ![ORCID] and Stefan Rhys Jeske ![ORCID]

Visual Computing Institute, RWTH Aachen University
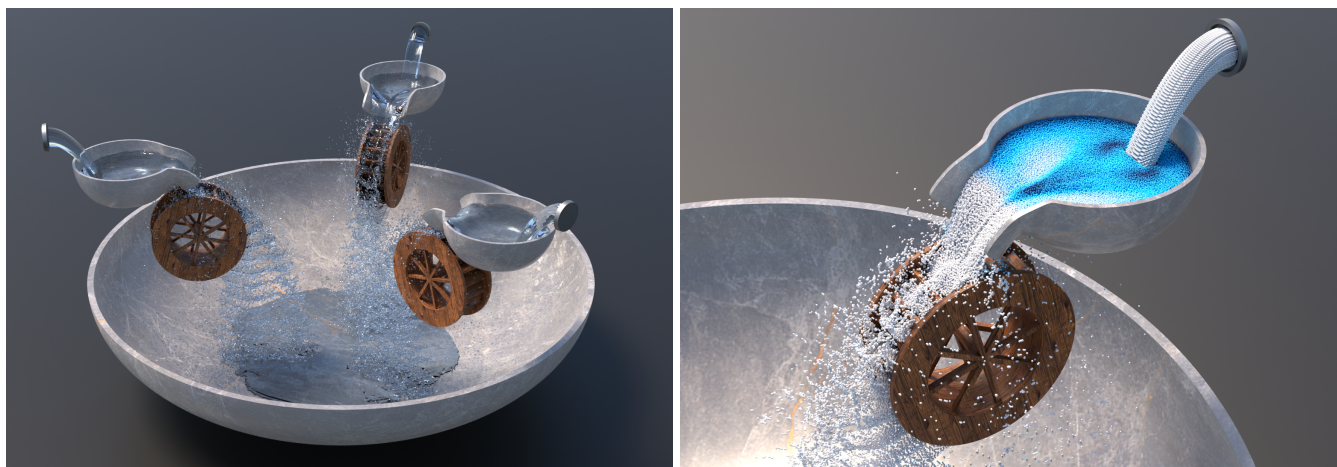
Figure 1: *Fluid particles interact with three water wheels. This experiment demonstrates a stable simulation of two million fluid particles interacting with dynamic rigid bodies using our SPH rigid-fluid coupling approach. The close-up on the right shows the fluid particles.*

Abstract
A common way to handle boundaries in SPH fluid simulations is to sample the surface of the boundary geometry using particles. These boundary particles are assigned the same properties as the fluid particles and are considered in the pressure force computation to avoid a penetration of the boundary. However, the pressure solver requires a pressure value for each particle. These are typically not computed for the boundary particles due to the computational overhead. Therefore, several strategies have been investigated in previous works to obtain boundary pressure values. A popular, simple technique is pressure mirroring, which mirrors the values from the fluid particles. This method is efficient, but may cause visual artifacts. More complex approaches like pressure extrapolation aim to avoid these artifacts at the cost of computation time.
We introduce a constraint-based derivation of Divergence-Free SPH (DFSPH) — a common state-of-the-art pressure solver. This derivation gives us new insights on how to integrate boundary particles in the pressure solve without the need of explicitly computing boundary pressure values. This yields a more elegant formulation of the pressure solver that avoids the aforementioned problems.

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

1. Introduction

Smoothed Particle Hydrodynamics (SPH) is an established mesh-less Lagrangian simulation approach. In computer graphics it has been investigated to simulate various materials like fluids, deformable solids, granular materials, or snow. The most important component of an SPH simulation framework for incompressible fluids is the pressure solver. It enforces the incompressibility of the fluid and implements the two-way coupling with rigid boundaries.

In recent years different approaches have been investigated to represent the boundary in SPH fluid simulations. The most popular methods are either based on an explicit particle sampling of the boundary surface or an implicit boundary representation. Both

concepts virtually extend field quantities of the fluid (e.g., density, pressure, etc.) into the boundary geometry. When a fluid particle gets close to the boundary, the virtual density contribution of the boundary domain then leads to a local compression. This compression is implicitly resolved by the pressure solver when enforcing incompressibility and a boundary penetration is avoided. In this way rigid-fluid coupling is implemented in state-of-the-art pressure solvers.

The concept of extending the fluid's field quantities to the boundary domain enables identical handling of fluid and boundary in the pressure solver. However, this means that state-of-the-art solvers also require quantities like pressure or pressure acceleration at the boundary which are costly to compute in each simulation step. Different approaches have been investigated to approximate these quantities at the boundary and therefore to avoid a computational overhead. However, simple methods like *pressure mirroring* [AIA*12, IOS*14, BKWK19] introduce small visual artifacts while more sophisticated methods like *pressure extrapolation* [AHA12, BGPT18] reduce these artifacts, but are more expensive to compute. Another issue is that it is not completely clear how to correctly implement this kind of boundary handling since previous works often use different simplifications of the formulation and therefore already many different variations of the same boundary handling approach exist. Typical simplifications are: setting the density of particles to the rest density [SP09], ignoring particle masses [MM13], or assuming that particle masses, densities or pressure values of neighboring particles are equal [SP09, AIA*12].

In our paper we present a mathematically consistent derivation of an implicit pressure solver with boundary handling which solves these problems. First, we introduce a constraint-based formulation of a state-of-the-art pressure solver, namely DFSPH, and show that this yields exactly the same equations as the original formulation. Then we extend the derivation by defining a density constraint for each fluid particle which also considers the virtual density contribution of the boundary. However, in contrast to previous works we clearly distinguish between dynamic fluid particles and static boundaries and do not handle them in the same way. This means we only define constraint functions for fluid particles, and static boundaries only add a constant contribution to these functions. In this way the constraint-based derivation automatically considers the difference between fluid particles and boundaries. In the derivation we avoid any simplification so that the solver can also be used in more complex simulations, e.g., with different particle sizes [FFWL*22]. This yields a pressure solver that does not require an explicit computation of boundary pressure values and accelerations. Therefore, it avoids the visual artifacts introduced by pressure mirroring, the computational overhead of pressure extrapolation, and an additional implementation of such methods. However, we believe that the main contribution of this paper is to provide a consistent and mathematically sound derivation of a pressure solver with boundary handling. Finally, we demonstrate the benefits of our solver in complex simulations (see Figure 1).

2. Related Work

The Smoothed Particle Hydrodynamics formulation was originally introduced by Gingold and Monaghan [GM77] in the field of as-

trophysics. SPH has also become an important simulation method in the graphics community and has been used to simulate different kinds of materials, e.g., fluids [ICS*14, BK17, WKB16, CBG*19], sand [LD09, AO11], deformable solids [PGBT18, KBFF*21], highly viscous materials [PT16, WKBB18, WJB23], snow [GHB*20], and ferrofluids [HHM19]. For an overview of SPH methods in computer graphics we refer the reader to recent surveys [IOS*14, KBST19, KBST22].

Pressure solvers Many SPH simulators integrate the boundary handling of the fluid into the pressure solver. Current state-of-the-art solvers determine the pressure-driven acceleration either explicitly with an equation of state or implicitly by solving a linear system given by the continuity equation and intermediate state values from the non-pressure forces. The former includes Weakly-Compressible SPH (WCSPH) [BT07], which employs the Tait equation to calculate the pressure from the current particle arrangement. In contrast to WCSPH, implicit pressure solvers enforce incompressibility via the continuity equation. Prominent examples include PCISPH [SP09], PBF [MM13], IISPH [ICS*14] and Projective Fluids [WKB16]. In particular, we will further investigate Divergence-Free SPH [BK17, CBG*19] in this work, which solves two linear systems for enforcing both the constant density and divergence-free condition of the continuity equation for incompressible fluids. An extensive overview and further derivations can be found in the survey of Koschier et al. [KBST22].

Boundary handling One approach to couple SPH fluids with the boundary is to use separate solvers for both and to explicitly compute forces at the interface between fluid and boundary. This allows for various representations of the boundary domain. Triangle mesh representations are used by Bodin et al. [BLS12] and Fujisawa and Miura [FM15] for rigid-fluid coupling, and by Huber et al. [HEW15] to strongly couple fluids and cloth. Harada et al. [HKK07] employ an implicit surface representation using signed distance fields (SDF). For particle-based boundaries, Monaghan [Mon94] as well as Becker and Teschner [BT07] compute penalty forces to simulate rigid-fluid coupling. Moreover, Becker et al. [BTT09] introduce interaction forces computed by a predictor-corrector scheme to update both fluid and boundary particles.

An alternative approach that has become popular is to integrate the boundary handling in the pressure solver instead of computing interface forces. Akinci et al. [AIA*12] and Ihmsen et al. [IAGT10] use a particle-based representation of the boundary and consider these additional boundary particles in the pressure solver. In contrast, Koschier et al. [KB17] and Bender et al. [BKWK19] use implicit boundary representations and precompute the contribution of the boundary for the density and pressure computations. However, to consider the boundary in the pressure solver, previous works require density and pressure values for the boundary. While the density value is typically set to the rest density of the fluid, there exist different methods to determine the pressure value like pressure mirroring [AIA*12], pressure extrapolation [AHA12, BGPT18], or pressure boundaries [BGI*18]. In our work we show that the computation of these boundary pressure values can introduce viusal artifacts or computational overhead. Therefore, we introduce a method which does not require the explicit computation of a boundary pressure value.

3. Method

In the following we first introduce the governing equations for the simulation of incompressible fluids and the spatial discretization. Then in Section 3.2 we discuss a common derivation of an implicit divergence-free SPH pressure solver for incompressible fluids. Since our work focuses on boundary handling, the next step is to extend the pressure solver to consider boundaries (see Section 3.3). For reasons of clarity and comprehensibility we first only consider static boundaries.

One of the most important approaches to handle boundaries in the pressure solve is to represent the boundary by additional sampling points. The motivation of the approach is to simply add additional boundary particles to the SPH formulation and handle them in the same way as the fluid particles. This means that each boundary particle also requires a pressure value in the implicit pressure solve. However, it is not clear how to choose the pressure values for the additional particles and there exist different methods to compute boundary pressures in literature. We discuss some of these methods at the end of Section 3.3.

In Section 3.4 we introduce our approach which is based on an alternative derivation of the implicit pressure solver using constraints. When considering boundaries, this derivation leads to a slightly modified pressure solver which does not require additional pressure values at the boundary sampling points. In this way we can avoid the above mentioned problems. Finally, we discuss the handling of dynamic boundaries in Section 3.5.

3.1. Foundations

In this work we focus on the simulation of incompressible fluids. Therefore, we solve the incompressible Navier-Stokes equations in Lagrangian coordinates

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}^{\text{ext}} \tag{1}$$

$$\frac{D\rho}{Dt} = 0 \iff \nabla \cdot \mathbf{v} = 0, \tag{2}$$

where $\rho, \mathbf{v}, t, p, \mu$ and $\mathbf{f}^{\text{ext}}$ denote density, velocity, time, pressure, dynamic viscosity and external body forces, respectively. Eq. (2) defines the incompressibility condition of the fluid which is derived from the continuity equation.

A common way to simplify the solution of Eq. (1) is to apply the concept of operator splitting [IOS*14, KBST22]. The core idea of this concept is to decompose the Navier-Stokes equation into subproblems, e.g., to solve for the pressure and viscosity forces sequentially. In this way we can employ an optimized solver for each subproblem. Since our work focuses on the development of a pressure solver with boundary handling, in the following we will combine all non-pressure forces $\mathbf{f}^{\text{np}} = \mu \nabla^2 \mathbf{v} + \mathbf{f}^{\text{ext}}$ to improve readability.

In order to solve the Navier-Stokes equations we use the Smoothed Particle Hydrodynamics (SPH) formulation. This is a Lagrangian approach which discretizes the spatial domain using particles that carry the field quantities. Using SPH a quantity A_i at position $\mathbf{x}_i$ is approximated by the quantities A_j at neighboring particle positions $\mathbf{x}_j$ as [Mon92]

$$A_i \approx \sum_j \frac{m_j}{\rho_j} A_j W_{ij}, \tag{3}$$

where m_j is the mass of particle j and $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j, h)$ is a Gaussian-like kernel function with compact support. h is the smoothing length of the kernel. In our work we use the cubic spline kernel.

3.2. Divergence-Free SPH Pressure Solver

In SPH simulations boundary handling is often performed by the pressure solver. Therefore, in the following we will derive the divergence-free SPH (DFSPH) pressure solver introduced by Bender and Koschier [BK17]. Note that other popular pressure solvers like PCISPH [SP09] or IISPH [ICS*14] can be derived in a similar way as shown in [KBST22]. In this subsection we will derive the pressure solver without boundary handling and show in the next subsection how to extend it to consider the boundary.

DFSPH solves two linear systems to enforce a constant density and a divergence-free velocity field, respectively. In the next paragraphs we derive the first system in detail and discuss how to solve it. At the end we show that the second system can be derived and solved analogously.

Since the pressure solver should only focus on the computation of pressure accelerations, we perform operator splitting (see Section 3.1) and first compute all non-pressure accelerations. These accelerations are used to determine the predicted velocities for all particles as

$$\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \mathbf{a}_i^{\text{np}}(t) \tag{4}$$

where $\mathbf{a}_i^{\text{np}}$ is the sum of all non-pressure accelerations (e.g. gravity, viscosity) acting on particle i.

In order to enforce a constant density we have to solve the pressure Poisson equation (PPE)

$$\Delta t \nabla^2 p = \frac{\rho_0 - \rho^*}{\Delta t}, \tag{5}$$

where ρ_0 is the rest density of the simulated material. The right hand side of the system is determined by the difference between the rest density and the predicted density

$$\rho_i^* = \rho_i + \Delta t \frac{D\rho_i}{Dt} = \rho_i + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \cdot \nabla W_{ij}, \tag{6}$$

where the density is computed by the SPH formulation in Eq. (3) as

$$\rho_i = \sum_j m_j W_{ij}. \tag{7}$$

By using the predicted density the non-pressure accelerations are considered in the pressure solve.

In the SPH formulation the PPE defines a linear system which can be written as

$$\mathbf{A}\mathbf{p} = \mathbf{s}, \tag{8}$$

where $\mathbf{p}$ denote the unknown pressure values and $\mathbf{s}$ the source

---

Algorithm 1 Constant density solver (DFSPH)

1: compute predicted velocities $\mathbf{v}_i^*$ (Eq. (4))
2: compute predicted densities ρ_i^* (Eq. (6))
3: determine source terms $\mathbf{s}_i = \frac{\rho_0 - \rho_i^*}{\Delta t}$ (Eq. (5))
4: compute diagonal elements $\mathbf{A}_{ii}$ (Eq. (12))
5: **while** residuum > tolerance **do**
6:     compute pressure accelerations $\mathbf{a}_i^p$ (Eq. (9))
7:     compute $\Delta t \nabla^2 p_i$ (Eq. (10))
8:     determine $p_i^{(l+1)}$ (Eq. (11))
9: update velocities $\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* + \Delta t \mathbf{a}_i^p(t)$

---

term vector with the elements $\mathbf{s}_i = \frac{\rho_0 - \rho_i^*}{\Delta t}$. To compute the left hand side of this system, we have to determine the Laplacian of p (see Eq. (5)). Therefore, we first compute the pressure accelerations [Mon92]

$$\mathbf{a}_i^p = -\frac{1}{\rho_i} \nabla p_i = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \qquad (9)$$

and then the divergence of ∇p_i to obtain the Laplacian of p_i as

$$\nabla^2 p_i = \nabla \cdot \nabla p_i = \sum_j m_j \left(\mathbf{a}_i^p - \mathbf{a}_j^p \right) \cdot \nabla W_{ij}. \qquad (10)$$

Since now we can compute the left and right hand side of the linear system in Eq. (5), the next step is to solve the system. SPH methods often use a parallelized Jacobi solver and clamp negative pressures in each iteration to avoid artifacts due to the particle deficiency problem at the free surface [KBST22]. In our work we use a relaxed Jacobi solver which determines the pressure values in each iteration as

$$p_i^{(l+1)} = p_i^{(l)} + \frac{\omega}{\mathbf{A}_{ii}} \left(\mathbf{s}_i - \sum_j \mathbf{A}_{ij} p_j^{(l)} \right), \qquad (11)$$

where ω is a user-defined relaxation factor and $\sum_j \mathbf{A}_{ij} p_j^{(l)} = \Delta t \nabla^2 p_i$ is the left hand side of our linear system in iteration l. In our experiments we set $\omega = 0.5$ as recommended by Ihmsen et al. [ICS*14]. The diagonal element is determined by accumulating all coefficients of p_i after substituting the pressure acceleration (Eq. (9)) in Eq. (10) (cf. [BGPT18])

$$\mathbf{A}_{ii} = -\frac{\Delta t}{\rho_i^2} \left(\| \sum_j m_j \nabla W_{ij} \|^2 + \sum_j m_i m_j \| \nabla W_{ij} \|^2 \right). \qquad (12)$$

The implementation of the derived constant density solver is shown in Algorithm 1. Note that the described solver is equivalent to the IISPH solver [ICS*14]. To implement the DFSPH pressure solver an additional linear system has to be solved in order to enforce a divergence-free velocity field. This system is defined by the following PPE

$$\Delta t \nabla^2 p = \rho \nabla \cdot \mathbf{v} \qquad (13)$$

which can be solved analogously to Eq. (5).

3.3. Classical Boundary Handling

In this subsection we introduce a common particle-based boundary handling which is often used in SPH simulations. For reasons of clarity and comprehensibility here we only consider static boundaries and discuss the extension to dynamic boundaries later in Section 3.5.

The key idea of particle-based boundary handling is to discretize the boundary with additional particles [Mon94, IAGT10, AIA*12, BKWK20]. These particles represent the boundary in the fluid simulation and are handled exactly in the same way as the fluid particles. The motivation for this is that the SPH sum (Eq. (3)) is an approximation of a volume integral in the spherical domain of the particle neighborhood. The boundary particles are used as additional sampling points to compute the integral in the boundary part. For a static boundary the additional particles can be seen as fluid particles which can not move. By considering the additional particles in the density computation, the density and therefore the fluid pressure increases as a fluid particle comes closer to the boundary. As a consequence, the pressure solver will push fluid particles out of the boundary since the boundary particles can not move.

Considering boundary particles in the density computation yields

$$\rho_i = \sum_j m_j W_{ij} + \sum_k \tilde{m}_k W_{ik}, \qquad (14)$$

where j and k denote the fluid and boundary neighbors of particle i, respectively. To compute the mass of the boundary particles $\tilde{m}_k$ we assume that they have the same rest density ρ_0 as the fluid since we treat these particles as static fluid particles. The mass is then determined by the volume the boundary particle represents times the rest density $\tilde{m}_k = \frac{\rho_0}{\sum_l W_{kl}}$, where l are the indices of the boundary neighbors of particle k [AIA*12].

The additional particles must also be considered in the other equations of the pressure solver (see Section 3.2). For a static boundary this concerns Eqs. (9), (10) and (12). The computation of the diagonal matrix elements in Eq. (12) can be easily extended by the additional particles since only their positions are required:

$$\mathbf{A}_{ii} = -\frac{\Delta t}{\rho_i^2} \left(\| \sum_j m_j \nabla W_{ij} + \sum_k \tilde{m}_k \nabla W_{ik} \|^2 \right.$$
$$\left. + \sum_j m_i m_j \| \nabla W_{ij} \|^2 + \sum_k m_i \tilde{m}_k \| \nabla W_{ik} \|^2 \right). \qquad (15)$$

However, the extended pressure acceleration (cf. Eq. (9)) requires a pressure value p_k for each boundary particle k [BKWK20]

$$\mathbf{a}_i^p = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} - \sum_k \tilde{m}_k \left(\frac{p_i}{\rho_i^2} + \frac{p_k}{\rho_0^2} \right) \nabla W_{ik}. \qquad (16)$$

Finally, the extended equation to compute the Laplacian of the pressure (cf. Eq. (10)) contains an additional term with the pressure acceleration $\mathbf{a}_k^p$ at the boundary particle k

$$\nabla^2 p_i = \sum_j m_j \left(\mathbf{a}_i^p - \mathbf{a}_j^p \right) \cdot \nabla W_{ij} + \sum_k \tilde{m}_k \left(\mathbf{a}_i^p - \mathbf{a}_k^p \right) \cdot \nabla W_{ik}. \qquad (17)$$

In previous works different strategies have been investigated to

determine this unknown boundary pressure p_k and pressure acceleration $\mathbf{a}_k^p$. In the following we discuss the most important ones:

Pressure mirroring [AIA*12, BKWK20] is a popular approach that is simple to implement. When computing the pressure acceleration for particle i we simply "mirror" its pressure value on each neighboring boundary particle by setting $p_k = p_i$. However, this approach leads to inconsistent pressure values since one boundary particle has different pressure values for different fluid particles which can cause visual artifacts [BGI*18, BGPT18]. The pressure accelerations at the boundary are typically neglected, i.e. $\mathbf{a}_k^p = 0$, since their computation is more involved [BGI*18].

Pressure boundaries [BGI*18] improve the robustness of the boundary handling by extending the PPE in Eq. (5) and computing the unknown boundary pressure values p_k in the same way as the fluid pressures. The method also explicitly computes pressure accelerations $\mathbf{a}_k^p$ for the boundary particles. However, this requires the solution of a larger linear system and therefore requires more memory and computation time [BGPT18].

Pressure extrapolation [AHA12, BGPT18] determines the unknown pressure values p_k by extrapolating the fluid pressures onto the boundary. The approach typically sets the pressure accelerations to zero, similar to pressure mirroring.

Finally, note that implicit boundary representations [KB17, BKWK19] also require pressure values on the boundary and therefore have the same problem as the discussed particle-based method.

3.4. Our Approach

In the following we introduce an alternative way to derive the implicit pressure solver of Section 3.2. For the derivation we use a constraint-based formulation and show that this leads to exactly the same solver. However, the constraint-based formulation gives us some insights on how to include the additional boundary particles in a consistent way. This is discussed in detail in the second part of this section.

3.4.1. Constraint-Based Pressure Solver

In order to enforce a constant density in the fluid we define a nonlinear density constraint for each fluid particle i as

$$C_i = \rho_0 - \rho_i = \rho_0 - \sum_j m_j W_{ij}, \qquad (18)$$

where ρ_0 is the rest density and ρ_i is determined using the SPH formulation in Eq. (7).

Similar to the impulse-based dynamic simulation approach [Mir96, BS06, BET14] we compute a Lagrange multiplier λ_i for each constraint by solving

$$\frac{1}{\rho_i} \left(\left\| \frac{\partial C_i}{\partial \mathbf{x}_i} \right\|^2 + \sum_j \left\| \frac{\partial C_i}{\partial \mathbf{x}_j} \right\|^2 \right) \lambda_i = -\frac{C_i^{**}}{\Delta t^2}. \qquad (19)$$

This equation is solved iteratively by a predictor-corrector method where in each iteration the constraint value C_i^{**} at the end of the time step is predicted by considering pressure and non-pressure accelerations to compute a solution of the non-linear problem.

Note that this formulation is also similar to the position-based fluids (PBF) approach of Macklin and Müller [MM13]. However, in our derivation we compute the constraint gradients using the current positions which corresponds to state-of-the-art solvers like IISPH [ICS*14] or DFSPH [BK17]. In contrast PBF updates the gradients in each iteration which has been shown to be less efficient [BK17]. Moreover, we solve on acceleration-level instead of using a position-based approach.

We need the constraint gradients to determine the left hand side of Eq. (19) which are determined as

$$\frac{\partial C_i}{\partial \mathbf{x}_i} = -\sum_j m_j \nabla_i W_{ij} \qquad (20)$$

$$\frac{\partial C_i}{\partial \mathbf{x}_j} = m_j \nabla_i W_{ij}. \qquad (21)$$

If we substitute these gradients in Eq. (19), we can rewrite the system using the value $\mathbf{A}_{ii}$ of Eq. (12)

$$\rho_i \mathbf{A}_{ii} \lambda_i = \frac{C_i^{**}}{\Delta t}. \qquad (22)$$

Note that due to our definition of the linear system in Eq. (19) the Lagrange multiplier represents the quotient of the pressure and the density, i.e. $\lambda_i = p_i / \rho_i$. In this way we can already see the similarity to the linear system of the original derivation (Eq. (8)). However, on the left hand side only the diagonal matrix elements are considered and on the right hand side we also consider pressure accelerations. Now we show that this is equivalent to Eq. (8).

To determine the predicted constraint value C_i^{**} we first have to compute the pressure forces. For a constraint C_i the force on a particle j is determined by

$$\mathbf{f}_{j \leftarrow i}^p = \frac{m_i}{\rho_i} \frac{\partial C_i}{\partial \mathbf{x}_j} \lambda_i. \qquad (23)$$

The volume of the particle is required since due to our formulation, the product of the constraint gradient and the Lagrange multiplier gives us a force per unit volume. If we sum up all pressure forces for particle i we get

$$\mathbf{f}_i^p = \frac{m_i}{\rho_i} \frac{\partial C_i}{\partial \mathbf{x}_i} \lambda_i + \sum_j \frac{m_j}{\rho_j} \frac{\partial C_j}{\partial \mathbf{x}_i} \lambda_j = -m_i \sum_j m_j \left(\frac{\lambda_i}{\rho_i} + \frac{\lambda_j}{\rho_j} \right) \nabla W_{ij}. \qquad (24)$$

The corresponding pressure acceleration $\mathbf{a}_i^p = \mathbf{f}_i^p / m_i$ is equivalent to the acceleration in the original derivation (cf. Eq. (9)).

In the next step we use the pressure and non-pressure accelerations to determine the predicted particle velocities

$$\mathbf{v}_i^{**} = \mathbf{v}_i(t) + \Delta t \left(\mathbf{a}_i^{np}(t) + \mathbf{a}_i^p(t) \right). \qquad (25)$$

Analogous to Eq. (6) we now determine the right hand side of the linear system (22) as

$$\frac{C_i^{**}}{\Delta t} = \frac{1}{\Delta t} \left(\rho_0 - \rho_i - \Delta t \sum_j m_j (\mathbf{v}_i^{**} - \mathbf{v}_j^{**}) \cdot \nabla W_{ij} \right)$$
$$= \frac{\rho_0 - \rho_i^*}{\Delta t} - \Delta t \sum_j m_j \left(\mathbf{a}_i^p - \mathbf{a}_j^p \right) \cdot \nabla W_{ij}. \qquad (26)$$

Solving the linear system using the relaxed Jacobi method yields the following update rule for the Lagrange multiplier

$$\lambda_i^{(l+1)} = \lambda_i^{(l)} + \frac{\omega}{\rho_i \mathbf{A}_{ii}} \frac{C_i^{**}}{\Delta t} \tag{27}$$

$$= \lambda_i^{(l)} + \frac{\omega}{\rho_i \mathbf{A}_{ii}} \left(\mathbf{s}_i - \sum_j \mathbf{A}_{ij} p_j^{(l)} \right) \tag{28}$$

which is equivalent to the iteration update of the original formulation in Eq. (11).

The definition of a constraint to enforce a divergence-free velocity field and the derivation of a corresponding solver can be done analogously.

In summary, this demonstrates that our constraint-based derivation leads to exactly the same result as the original derivation. However, we find that our formulation is more intuitive when adding boundary handling to the system. In the next part of this section we will see the benefit of this as it helps us to avoid the computation of unnecessary pressure values at the boundary.

3.4.2. Boundary Handling

Now we extend the density constraint of Eq. (18) to consider the boundary. We still assume that the boundary is static and will discuss dynamic boundaries in the next subsection. The constraint function is extended analogously to the classical boundary handling approach (cf. Eq. (14))

$$C_i = \rho_0 - \rho_i = \rho_0 - \sum_j m_j W_{ij} - \sum_k \tilde{m}_k W_{ik}. \tag{29}$$

However, in contrast to the classical approach we do not handle the boundary particles in the same way as dynamic fluid particles. Since we assume a static boundary, the constraint function only depends on the positions of the fluid particle i and its fluid neighbors j. It does not depend on $\mathbf{x}_k$ since these values are constant. Moreover, we only define constraints for fluid particles since static boundary particles can not move.

Therefore, the gradient of C_i with respect to the position of a boundary particle $\mathbf{x}_k$ is zero. The remaining gradients of the extended constraint function are determined as

$$\frac{\partial C_i}{\partial \mathbf{x}_i} = -\sum_j m_j \nabla W_{ij} - \sum_k \tilde{m}_k \nabla W_{ik} \tag{30}$$

$$\frac{\partial C_i}{\partial \mathbf{x}_j} = m_j \nabla W_{ij}. \tag{31}$$

Substituting these gradients in Eq. (19) yields a modified definition of $\mathbf{A}_{ii}$ which is required to solve Eq. (22):

$$\mathbf{A}_{ii} = -\frac{\Delta t}{\rho_i^2} \left(\| \sum_j m_j \nabla W_{ij} + \tilde{m}_k \nabla W_{ik} \|^2 + \sum_j \| m_j \nabla W_{ij} \|^2 \right). \tag{32}$$

In the next step we compute the extended pressure accelerations as (cf. Eq. (24))

$$\mathbf{a}_i^{\mathrm{p}} = -\sum_j m_j \left(\frac{\lambda_i}{\rho_i} + \frac{\lambda_j}{\rho_j} \right) \nabla W_{ij} - \sum_k \tilde{m}_k \frac{\lambda_i}{\rho_i} \nabla W_{ik}. \tag{33}$$

Note that in contrast to previous approaches (see Section 3.3) this equation does not contain any boundary pressure value p_k (or in our case λ_k) since density constraints are only defined for dynamic fluid particles, but not for static boundary particles.

Finally, we extend the right hand side of the PPE (see Eq. (26)) in order to consider the boundary:

$$\frac{C_i^{**}}{\Delta t} = \frac{\rho_0 - \rho_i^*}{\Delta t} - \Delta t \sum_j m_j \left(\mathbf{a}_i^{\mathrm{p}} - \mathbf{a}_j^{\mathrm{p}} \right) \cdot \nabla W_{ij} - \Delta t \sum_k \tilde{m}_k \mathbf{a}_i^{\mathrm{p}} \cdot \nabla W_{ik}. \tag{34}$$

This extension of our constraint-based derivation yields a solver which can handle static boundaries without the need of explicitly computing boundary pressure values.

3.5. Dynamics Boundaries

So far we assumed that we only have static boundaries in the simulation. Now we discuss how two-way coupling with dynamic boundaries can be simulated.

We follow the approach of Akinci et al. [AIA*12] and compute the pressure force that acts from a fluid particle i on a boundary particle k as

$$\mathbf{f}_{k \leftarrow i}^{\mathrm{p}} = m_i \tilde{m}_k \left(\frac{p_i}{\rho_i^2} \right) \nabla W_{ik}. \tag{35}$$

The resulting force is applied to the rigid body at the position of the boundary particle k to update the motion of the body. This approach is quite common and has been used in several research works to handle dynamic boundaries with weak (e.g., [ICS*14, KB17]) and strong two-way coupling (e.g. [GPB*19]). Interestingly in this approach the pressure force does not depend on the boundary pressure value p_k. Therefore, this force perfectly fits to our formulation and can also directly be derived from Eq. (33).

4. Results

For the experiments in this section we implemented our approach in the open-source framework SPlisHSPlasH [B*23] using an OpenMP parallelization. We used the surface reconstruction of Böttcher et al. [BLJB23] for visualization. Timings were measured on an Intel Core i9-9900K processor with 8 physical cores at 3.60 GHz.

Comparison In the first experiment we compare our approach with pressure mirroring [KBST22] and the MLS pressure extrapolation method of Band et al. [BGPT18]. Since the latter method is based on a particle-based boundary representation, we use such a representation [AIA*12] in this experiment.

For the comparison we drop a cube of fluid particles in a sphere and wait until the fluid comes to rest. Figure 2 shows the fluid particles on the bottom of the sphere at the end of the simulation for all three methods. The color-coding of the particles represents the pressure in the fluid.

As expected pressure mirroring (see Figure 2a) causes small visual artifacts which can be seen in the noisy pressure and particle

(a) *Pressure mirroring* **(b)** *Pressure extrapolation* **(c)** *Our method*
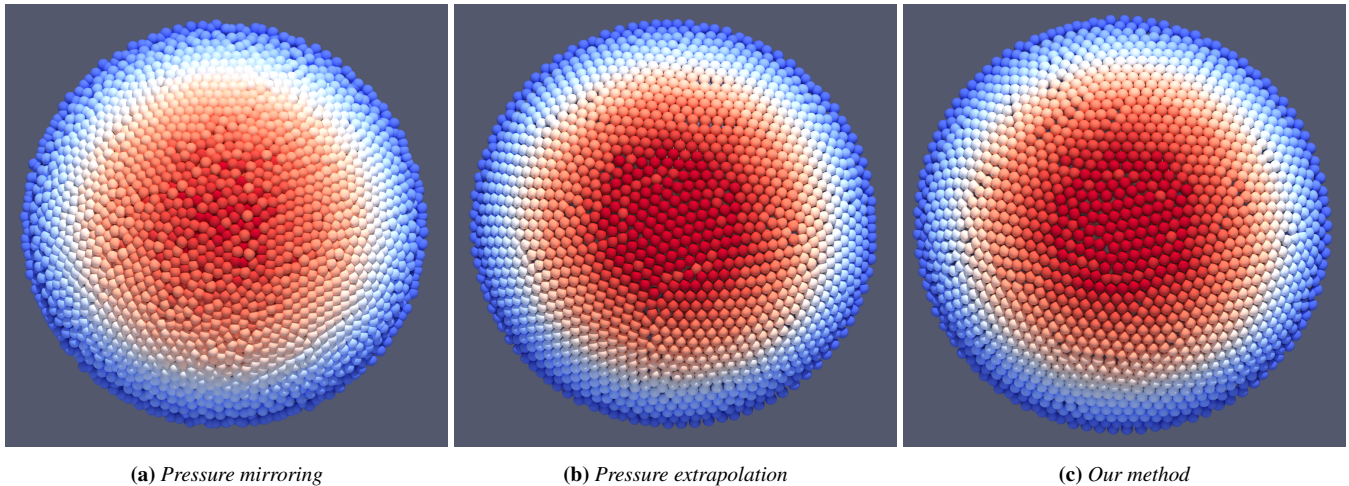
Figure 2: *Fluid particles at the bottom of a sphere with color-coded pressure values. (a) Pressure mirroring causes artifacts which leads to a noisy pressure and particle distribution. (b) Pressure extrapolation solves this problem at additional computational cost. (c) Our method also solves the problem without any computational overhead.*

distribution. Pressure extrapolation and our method solve this problem (see Figures 2b and 2c) and show a typical SPH particle distribution and a smooth pressure field. However, in contrast to our approach, the MLS pressure extrapolation method has to determine and invert an MLS matrix for each particle in each simulation step. In the experiment the extrapolation pressure solver required about 6.5 % more computation time than our method.

In large-scale simulations the small visual artifacts of pressure mirroring and also the computational overhead of pressure extrapolation might not be substantial. However, the implementation of our approach is not more complex than pressure mirroring but it avoids the artifacts. And the implementation is definitely simpler than MLS pressure extrapolation which requires the computation and inversion of a potentially singular MLS matrix. So our approach provides a clear benefit at no extra cost.

Complex simulations To demonstrate that our approach can handle complex scenarios with static and dynamic boundaries, we perform two simulations with large particle numbers. In contrast to the previous experiment we use an implicit boundary representation [BKWK20] to show that our approach is not restricted to particle-based boundaries. We compute vorticity [BKKW19] and drag forces [GBP*17] to get more realistic motion.

The first simulation (see Figure 3) shows that our solver can handle the rigid-fluid coupling with complex static boundary geometries. In the experiment three Armadillo models are sampled by 1.4 million fluid particles and interact with three static dragon models.

In the second simulation 2 million fluid particles are emitted and interact with dynamic water wheels (see Figure 1). The wheels are simulated as rigid bodies using a position-based method [DCB14]. This experiment demonstrates that our method can simulate a stable interaction with dynamic boundaries.
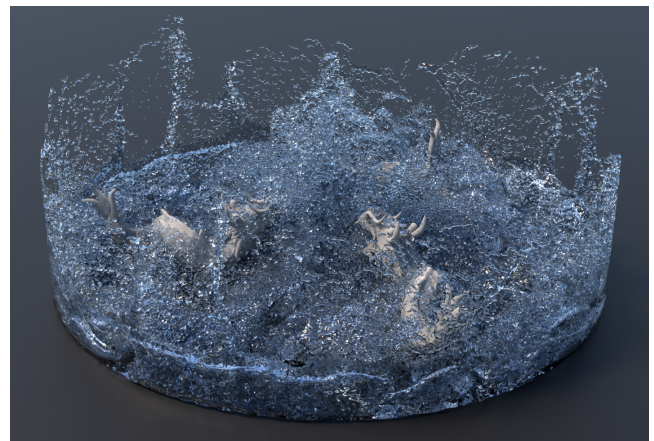
Figure 3: *Six fluid armadillos interact with three static dragon models using our boundary handling approach*

5. Conclusion

SPH approximates a volume integral over a spherical domain using a particle discretization. However, if a boundary lies in this domain, there are not enough sampling points. Therefore, a common way to handle rigid-fluid coupling in SPH simulations is to sample the boundary geometry by additional particles. To solve the integral these particles are handled in the same way as the dynamic fluid particles. This requires a virtual extension of the fluid's field quantities into the boundary. However, methods to compute these quantities for boundary particles either introduce a computational overhead or lead to visual artifacts. In this work we introduced a constraint-based derivation of DFSPH which does not require an explicit computation of the quantities and therefore avoids these problems. Finally, note that our boundary handling can be derived for other pressure solvers like PCISPH, IISPH or PBF analogously.

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. 2, 5

[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. 2, 4, 5, 6

[AO11] ALDUÁN I., OTADUY M. A.: SPH granular flow with friction and cohesion. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM Press. 2

[B*23] BENDER J., ET AL.: SPlisHSPlasH Library. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH, 2023. 6

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 5

[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. 2, 5

[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. 2, 4, 5, 6

[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. 2, 3, 5

[BKKW19] BENDER J., KOSCHIER D., KUGELSTADT T., WEILER M.: Turbulent micropolar SPH fluids with foam. *IEEE Transactions on Visualization and Computer Graphics 25*, 6 (June 2019), 2284–2295. 7

[BKWK19] BENDER J., KUGELSTADT T., WEILER M., KOSCHIER D.: Volume maps: An implicit boundary representation for SPH. In *Motion, Interaction and Games* (2019), ACM, p. 26. 2, 5

[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. 4, 5, 7

[BLJB23] BÖTTCHER T., LÖSCHNER F., JESKE S. R., BENDER J.: Weighted laplacian smoothing for surface reconstruction of particle-based fluids. In *Vision, Modeling, and Visualization* (2023), The Eurographics Association. 6

[BLS12] BODIN K., LACOURSIÈRE C., SERVIN M.: Constraint fluids. *IEEE Trans. on Visualization and Computer Graphics 18* (2012). 2

[BS06] BENDER J., SCHMITT A.: Fast dynamic simulation of multibody systems using impulses. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Madrid (Spain), Nov. 2006), pp. 81–90. 5

[BT07] BECKER M., TESCHNER M.: Weakly compressible SPH for free surface flows. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), Eurographics Association, p. 209–217. 2

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503. 2

[CBG*19] CORNELIS J., BENDER J., GISSLER C., IHMSEN M., TESCHNER M.: An optimized source term formulation for incompressible SPH. *The Visual Computer 35*, 4 (Apr 2019), 579–590. 2

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid body dynamics. *Computer Animation and Virtual Worlds 27*, 2 (2014), 103–112. 7

[FFWL*22] FERNÁNDEZ-FERNÁNDEZ J. A., WESTHOFEN L., LÖSCHNER F., JESKE S. R., LONGVA A., BENDER J.: Fast Octree Neighborhood Search for SPH Simulations. *ACM Transactions on Graphics 41*, 6 (2022), 13. 2

[FM15] FUJISAWA M., MIURA K.: An Efficient Boundary Handling with a Modified Density Calculation for SPH. *Computer Graphics Forum 34*, 7 (2015), 155–162. 2

[GBP*17] GISSLER C., BAND S., PEER A., IHMSEN M., TESCHNER M.: Generalized drag force for particle-based simulations. *Computers & Graphics 69* (dec 2017), 1–11. 7

[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. 2

[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. 2

[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 (2019). 6

[HEW15] HUBER M., EBERHARDT B., WEISKOPF D.: Boundary Handling at Cloth-Fluid Contact. *Comp. Graphics Forum 34*, 1 (2015). 2

[HHM19] HUANG L., HÄDRICH T., MICHELS D. L.: On the accurate large-scale simulation of ferrofluids. *ACM Transactions on Graphics 38*, 4 (2019). 2

[HKK07] HARADA T., KOSHIZUKA S., KAWAGUCHI Y.: Smoothed particle hydrodynamics in complex shapes. In *Spring Conf. on Computer Graph.* (2007), pp. 191–197. 2

[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. 2, 4

[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). 2, 3, 4, 5, 6

[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. 2, 3

[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. 2, 5, 6

[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. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 4*, 3 (2021). 2

[KBST19] KOSCHIER D., BENDER J., SOLENTHALER B., TESCHNER M.: Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. In *EUROGRAPHICS 2019 Tutorials* (2019), Eurographics Association. 2

[KBST22] KOSCHIER D., BENDER J., SOLENTHALER B., TESCHNER M.: A Survey on SPH Methods in Computer Graphics. *Computer Graphics Forum 41*, 2 (2022). 2, 3, 4, 6

[LD09] LENAERTS T., DUTRÉ P.: Mixing fluids and granular materials. *Computer Graphics Forum 28* (2009), 213–218. 2

[Mir96] MIRTICH B. V.: *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California at Berkeley, 1996. 5

[MM13] MACKLIN M., MÜLLER M.: Position Based Fluids. *ACM Transactions on Graphics 32*, 4 (2013), 1–5. 2, 5

[Mon92] MONAGHAN J.: Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. 3, 4

[Mon94] MONAGHAN J.: Simulating Free Surface Flows with SPH. *Journal of Computational Physics 110*, 2 (1994), 399–406. 2, 4

[PGBT18] PEER A., GISSLER C., BAND S., TESCHNER M.: An implicit SPH formulation for incompressible linearly elastic solids. *Computer Graphics Forum 37*, 6 (2018), 135–148. 2

[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. 2

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3 (2009). 2, 3

[WJB23] WESTHOFEN L., JESKE S. R., BENDER J.: A comparison of linear consistent correction methods for first-order SPH derivatives. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* (2023). 2

[WKB16] WEILER M., KOSCHIER D., BENDER J.: Projective fluids. In *ACM Motion in Games* (2016), pp. 1–6. 2

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