Divergence-Free Smoothed Particle Hydrodynamics

Jan Bender* Dan Koschier†
Graduate School CE
TU Darmstadt

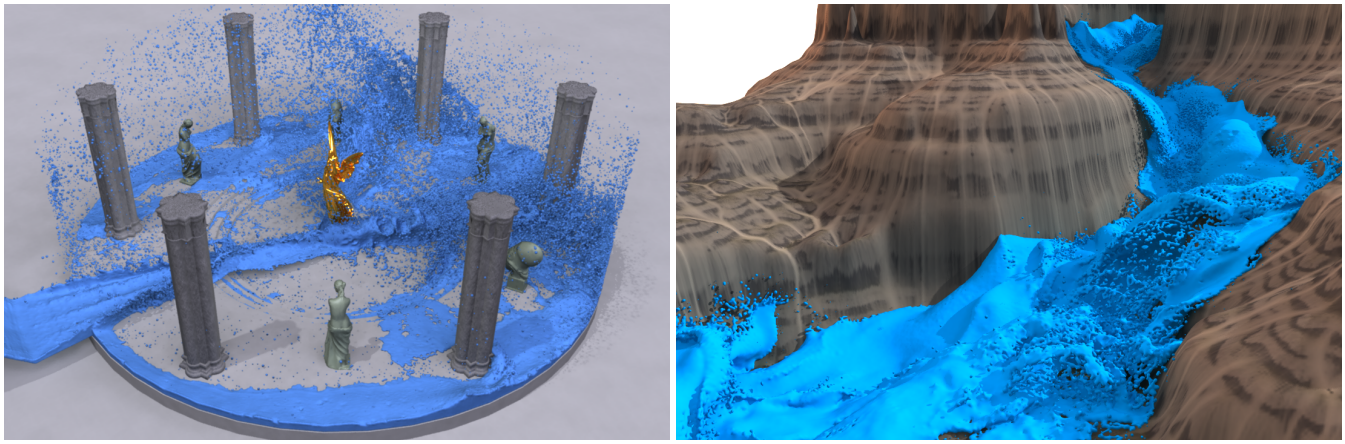

Figure 1: *Our new SPH method allows a stable simulation of incompressible fluids with high velocities while maintaining a divergence-free velocity field. This is shown on the left in a simulation with 2.4 million fluid particles and 6 million boundary particles. Moreover, our approach is significantly faster than current state-of-the-art SPH methods and is able to simulate complex scenes consisting of 5 million fluid particles and 40 million boundary particles in 5 seconds per time step with a maximum volume compression of 0.01 % (right).*

Abstract

In this paper we introduce an efficient and stable implicit SPH method for the physically-based simulation of incompressible fluids. In the area of computer graphics the most efficient SPH approaches focus solely on the correction of the density error to prevent volume compression. However, the continuity equation for incompressible flow also demands a divergence-free velocity field which is neglected by most methods. Although a few methods consider velocity divergence, they are either slow or have a perceivable density fluctuation.

Our novel method uses an efficient combination of two pressure solvers which enforce low volume compression (below $0.01\,\%$) and a divergence-free velocity field. This can be seen as enforcing incompressibility both on position level and velocity level. The first part is essential for realistic physical behavior while the divergence-free state increases the stability significantly and reduces the number of solver iterations. Moreover, it allows larger time steps which yields a considerable performance gain since particle neighborhoods have to be updated less frequently. Therefore, our *divergence-free* SPH (DFSPH) approach is significantly faster and more stable than current state-of-the-art SPH methods for incompressible fluids. We demonstrate this in simulations with millions of fast moving particles.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: fluid simulation, Smoothed Particle Hydrodynamics, divergence-free fluids, incompressibility, implicit integration

1 Introduction

In the last years Smoothed Particle Hydrodynamics (SPH) became a popular method in computer graphics to simulate complex water effects. SPH is a meshless Lagrangian method which computes the fluid quantities at a given point by only considering a finite set of neighboring particles. One of the most challenging problems in this field is to enforce incompressibility which is essential to simulate realistic physical behavior.

In general the SPH simulation of incompressible fluids is performed using the incompressible, isothermal Navier-Stokes equations in Lagrangian coordinates

$$\frac{D\rho}{Dt} = 0 \quad \Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0 \tag{1}$$

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

where $D(\cdot)/Dt$ denotes the material derivative and ρ, p, ν, $\mathbf{v}$ and $\mathbf{f}$ denote density, pressure, kinematic viscosity, velocity and body forces, respectively. Note that due to the incompressibility the partial derivative of the density with respect to the time is zero, i.e. $\partial\rho/\partial t = 0$, which implies the equivalence of both equations

*e-mail:bender@gsc.tu-darmstadt.de
†e-mail:koschier@gsc.tu-darmstadt.de

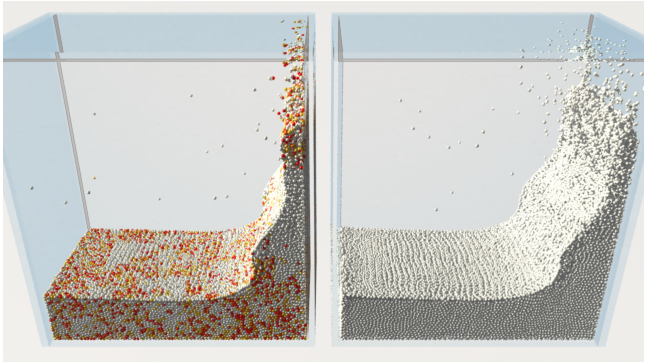

Figure 2: *Comparison between the velocity divergence of IISPH (left) and DFSPH (right) in a breaking dam simulation with 125k particles. The divergence errors are color coded, where white is the minimum and red is the maximum.*

in (1). According to these equations, an incompressible fluid must fulfill the *divergence-free condition* $\nabla \cdot \mathbf{v} = 0$, which means that the fluid must have a divergence-free velocity field. From the continuity equation $\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}$ and the divergence-free condition it follows that the density must stay constant over time $\frac{D\rho}{Dt} = 0$. Hence, in theory, a fluid with a divergence-free velocity field has a constant density and therefore is incompressible. However, in practice, enforcing the divergence-free condition in a simulation is not sufficient to guarantee incompressibility. Inevitable errors of numerical time integration lead to density deviations which sum up over the simulation. Since the divergence-free condition does not consider the resulting density error, volume compression due to numerical errors can not be avoided. To correct the numerical density error a second condition must be fulfilled $\rho - \rho_0 = 0$, which we call the *constant density condition*. In summary, the Navier-Stokes equations for incompressible fluids demand a divergence-free velocity field and additionally a stabilization in form of the constant density condition is required to counteract numerical errors.

Recently, in the area of computer graphics implicit pressure solvers became popular in SPH simulations to simulate incompressible fluids. These solvers determine pressure forces for the particle model in order to prevent volume compression. Currently, the most efficient SPH pressure solvers for incompressible fluids solely consider the constant density condition which only depends on the particle positions. However, the resulting velocity field is generally not divergence-free (see Figure 2, left) which is demanded by the continuity equation for incompressible flow. So far only few SPH approaches exist which consider velocity divergence but either they cannot enforce a low density error or they are comparatively slow.

In this paper we introduce a novel efficient and stable SPH approach which in contrast to most previous methods fulfills the divergence-free condition and the constant density condition. This is achieved by the combination of two pressure solvers which consider the divergence error and the density error, respectively. Our first solver enforces a divergence-free velocity field (see Figure 2, right) which has several advantages in SPH simulations. First of all, the simulation gets more stable, especially in simulations with fast moving particles (see Figure 1). The increased stability allows us to perform larger time steps which increases the overall performance since the computationally expensive neighborhood search is required less frequently. Moreover, the maximum time step size can be determined more accurately by the CFL condition which depends on the maximum particle velocity. Finally, enforcing a divergence-free velocity field significantly decreases the number of iterations required by the second solver which corrects the density error. There already exist different constant density solvers which could be combined with our divergence-free solver, e.g. Predictive-Corrective Incompressible SPH (PCISPH) [Solenthaler and Pajarola 2009] and Implicit Incompressible SPH (IISPH) [Ihmsen et al. 2014a]. However, in this work we introduce a new method that perfectly fits our divergence-free solver since it uses an analogue technique and can therefore reuse precomputed coefficients. The simulation with our new approach yields speedup factors of more than 20 in comparison to current state-of-the-art SPH methods.

2 Related Work

Within the field of computer graphics a variety of methods for the simulation of incompressible fluids were developed. In this section we give a brief overview over related approaches. For a general survey we refer to the work of Bridson [2008] while the recent state of the art report of Ihmsen et al. [2014b] specifically discusses SPH methods for fluid simulation.

Monaghan [1994] used state equations to weakly enforce incompressibility by penalizing density errors using a stiffness-weighted pressure term. They choose the stiffness coefficient dependent on the speed of sound to keep the compressibility considerable. Desbrun and Gascuel [1996] made use of an equation of state (EOS) based solver in order to animate highly deformable bodies. Later, Müller et al. [2003] introduced an EOS-based approach for fluid simulation to the computer graphics community while Adams et al. [2007] extended this method by spatial adaptivity. Becker and Teschner [2007] proposed an EOS method to restrict the maximum compression by predetermined, scenario-dependent stiffness coefficients. However, small density tolerances lead to high stiffness coefficients resulting in stiff differential equations and therefore to strict time-step restrictions.

To further enhance density preservation, pressure can be computed using an intermediate density determined after advecting the particles without pressure forces, known as the concept of splitting [Chorin 1968; Bridson 2008]. Solenthaler and Pajarola [2009] adopted this splitting concept and extended it by an iterative pressure solver in order to keep the maximum density error within a user-defined tolerance. Later, the method was extended by rigid-fluid coupling [Akinci et al. 2012] and a novel surface tension model [Akinci et al. 2013]. Macklin and Müller [2013] proposed Position Based Fluids (PBF) – a similar concept that iteratively refines particle positions to enforce incompressibility. Holonomic constraints on the density error motivated by rigid body mechanics were solved on a velocity level by Bodin et al. [2012]. However, they reported compression errors of up to 17 % due to numerical approximations, a regularization parameter and errors in time-integration. All of these EOS-based iterative methods with splitting enforce incompressibility only either on velocity or density level while our method considers both. Recently, Kang and Sagong [2014] proposed a method extending the work of Macklin and Müller [2013] by additionally satisfying the divergence-free condition. However, they cannot guarantee a divergence-free velocity field after each timestep as particle positions are updated subsequently to the velocity projection. Moreover, they report runtimes being similar or worse compared to the method of Macklin and Müller while our method yields large speed-up factors of up to one order of magnitude especially for large time steps (see Table 2).

In contrast to non-iterative as well as iterative EOS solvers, the Pressure Poisson Equation (PPE) can be solved in order to project intermediate velocities onto a divergence-free state. Especially, in grid-based, Eulerian approaches this is common practice [Bridson 2008]. Therefore, Cummins and Rudman [1999] solved the PPE

using a multigrid approach where the particles are treated as the finest level grid. Later, a semi-Lagrangian method was employed by Foster and Fedkiw [2001] where incompressibility is enforced on a simulation grid while they counteracted mass dissipation using a level set and freely moving inertialess particles. Further hybrid approaches using particles and a background grid were developed in the following years [Zhu and Bridson 2005; Raveendran et al. 2011; Ando et al. 2013]. An advanced method was proposed by Losasso et al. [2008] where a PPE is solved on a background grid not only to keep the field divergence-free but to maintain a predefined target density. Another interesting approach was introduced by Sin et al. [2009] as they solve the PPE on a Voronoi diagram constructed from the point-samples in each step. However, all of these approaches have to maintain and sometimes reconstruct additional grid data-structures resulting in high memory consumption, especially for large domains. Our method fully avoids background grids and therefore consumes a smaller amount of memory especially for large scenarios like in Figure 1.

Some approaches directly solve the PPE on the meshless discretization. An incompressible SPH method for multiphase flows was proposed by Hu et al. [2007], where the time integration step to obtain the velocities is subdivided into two halfsteps. During the first halfstep density fluctuations are eliminated using a position-altering iterative gradient descent solver. During the second halfstep errors in velocity divergence are resolved in a similar fashion. However, they applied their method exclusively to non-complex two-dimensional scenarios of up to 14,400 particles. He et al. [2012] extended the work of Solenthaler and Parajola [2009] by a local Poisson solver enforcing a divergence-free velocity field and constant density. Unfortunately, they have to determine particle neighbors in each solver iteration for their Poisson solve, while the integration domain is not necessarily equal or a subset of the local kernel function support. This leads to a considerable computational effort since the search radius increases. Moreover, they report only a small speedup factor of approximately 1.5 in comparison to the underlying method of Solenthaler and Parajola while we experienced speedup factors of more than 20 with DFSPH. In a recent work of Ihmsen et al. [2014a] a PPE is used in order to iteratively decrease the density deviation to 0.1% or even less. However, they do not consider eliminating velocity divergence which leads to a large number of solver iterations compared to our method and can even cause artifacts in special cases (see Section 4).

3 Fluid Simulation

In our simulation we use the incompressible, isothermal Navier-Stokes equations (see Section 1). However, we skip the second, viscous term on the right hand side of Equation (2) and use the XSPH variant proposed by Schechter and Bridson [2012] to simulate viscosity. We spatially discretize Equation (2) using the SPH method as described in the following while the incompressibility condition represented by Equation (1) is fulfilled by enforcing the divergence-free condition and the constant density condition as described in Sections 3.2 and 3.3, respectively.

Using the SPH concept a quantity at position $\mathbf{x}_i$ is approximated by the values at a set of neighboring particles $\mathbf{x}_j$ [Monaghan 1992]. One of the most important quantities in the SPH formulation is the density which can be approximated using this concept as:

$$\rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j W_{ij} = \sum_j m_j W_{ij},$$

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 the support radius h. After computing the density, the pressure field of a fluid is typically determined by the following equation of state (EOS):

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

where κ and γ are stiffness parameters and ρ_0 is the rest density. In our work we consider the special case

$$p_i = \kappa(\rho_i - \rho_0), \tag{3}$$

where $\gamma = 1$ which is a common choice in computer graphics [Desbrun and Gascuel 1996; Müller et al. 2003; Ihmsen et al. 2014b].

In the standard SPH approach [Monaghan 1992] the pressure field computed by the EOS is directly used to determine forces in order to encounter volume compression. However, weakly compressible fluids require a high stiffness coefficient κ which restricts the time step size and therefore the overall performance considerably. Hence, implicit pressure solvers, e.g. [Solenthaler and Pajarola 2009; Ihmsen et al. 2014a], were investigated to simulate incompressible fluids. These solvers typically compute the pressure field by solving a linear system which allows larger time steps and as a consequence a significant performance gain.

In this paper we present a novel implicit SPH method for incompressible fluids. Our approach uses two solvers to fulfill the divergence-free condition and the constant density condition, respectively. The goal of the *divergence-free solver* (see Section 3.2) is to obtain a divergence-free velocity field. However, as discussed in Section 1, this is not sufficient to guarantee incompressibility in practice since density deviations due to numerical errors cannot be corrected. Therefore, we employ a *constant density solver* (see Section 3.3) as a stabilization which eliminates the density errors. The key idea of the solvers is to determine an individual stiffness coefficient κ_i (see Equation (3)) for each neighborhood to solve the respective condition locally. In order to fulfill the conditions globally the solvers process the neighborhoods in a parallel Jacobi fashion which is a common choice in SPH solvers [Solenthaler and Pajarola 2009; Macklin and Müller 2013; Ihmsen et al. 2014a]. Adjusting the stiffness coefficients individually is equivalent to an implicit computation of the pressure field. However, this allows us a simple and consistent formulation of the constant density and the divergence-free forces.

3.1 Simulation Step

At the end of a simulation step both the constant density condition and the divergence-free condition must be fulfilled. The divergence-free solver computes pressure forces which are integrated once to obtain the velocity changes that satisfy the corresponding condition. The pressure forces determined by the constant density solver must be integrated twice to get the required position changes. Therefore, as a side-effect, it also modifies the velocities. For this reason, first, we execute the constant density solver and modify the velocities and positions, and then the divergence-free solver which corrects the resulting velocities to obtain a divergence-free state. Since both steps are executed in a loop, performing the density stabilization before computing a divergence-free velocity field does not impose any restrictions.

Algorithm 1 outlines the simulation with our novel method in detail. First, we determine the particle neighborhoods N_i using compact hashing [Ihmsen et al. 2011]. Then for each particle we compute the density ρ_i and the factor α_i (see Section 3.2). α_i is a common factor of both solvers which is required to correct the density error and the divergence error, respectively. Since this factor solely depends on the current positions, it is precomputed before executing the solvers which reduces the computational effort of both solvers significantly.

Algorithm 1 Simulation

```
1:  function performSimulation
2:      for all particles i do                          // init neighborhoods
3:          find neighborhoods N_i(0)
4:      for all particles i do                          // init ρ_i and α_i
5:          compute densities ρ_i(0)
6:          compute factors α_i(0)
7:      while (t < t_max) do                            // start simulation loop
8:          for all particles i do
9:              compute non-pressure forces F_i^adv(t)
10:         adapt time step size Δt according to CFL condition
11:         for all particles i do                      // predict velocities v_i^*
12:             v_i^* = v_i + Δt F_i^adv / m_i
13:         correctDensityError(α, v^*)                 // fulfill ρ^* − ρ_0 = 0
14:         for all particles i do                      // update positions
15:             x_i(t + Δt) = x_i(t) + Δt v_i^*
16:         for all particles i do                      // update neighborhoods
17:             find neighborhoods N_i(t + Δt)
18:         for all particles i do                      // update ρ_i and α_i
19:             compute densities ρ_i(t + Δt)
20:             compute factors α_i(t + Δt)
21:         correctDivergenceError(α, v^*) // fulfill Dρ/Dt = 0
22:         for all particles i do                      // update velocities
23:             v_i(t + Δt) = v_i^*
```

The first step in the simulation loop is to determine all non-pressure forces $\mathbf{F}^{\text{adv}}$ such as gravity, surface tension and viscosity and to adapt the time step size by the Courant-Friedrich-Levy (CFL) condition $\Delta t \leq 0.4\frac{d}{\|\mathbf{v}_{\max}\|}$ [Monaghan 1992], where d is the particle diameter and $\mathbf{v}_{\max}$ is the maximum particle velocity. In line 12 the non-pressure forces are used to compute predicted velocities $\mathbf{v}_i^*$. The constant density solver uses this prediction and the precomputed factors α_i to determine the pressure forces for each neighborhood in order to correct the density error $\rho_i^* - \rho_0$ (see Section 3.3). Then the positions are integrated forward in time. Therefore, the neighborhoods N_i, the densities ρ_i and the factors α_i have to be updated. In line 21 the divergence-free solver computes pressure forces to fulfill $\frac{D\rho_i}{Dt} = 0$ in order to make the velocity field divergence-free (see Section 3.2). Finally, the resulting velocity changes are used to update the particle velocities.

Note that N_i, ρ_i and α_i are determined only once per simulation step. However, we do not compute these values in the beginning of the simulation loop as in previous works since our two solvers are executed at different points of time. Instead, these values have to be initialized before the first simulation step in lines 2-6. They are then updated in each time step in lines 16-20.

3.2 Divergence-Free Solver

Our divergence-free solver enforces the condition $\frac{D\rho}{Dt} = 0$ which means that the density does not change over time. This is equivalent to the divergence-free condition (see Equation (1)) and therefore leads to a divergence-free velocity field.

If the condition $\frac{D\rho_i}{Dt} = 0$ is not fulfilled for a particle i, the solver computes a set of pressure forces for the particle and its neighborhood which correct the divergence error. The pressure force of particle i is determined by

$$\mathbf{F}_i^{\text{p}} = -\frac{m_i}{\rho_i}\nabla p_i, \tag{4}$$

where the pressure gradient is computed by differentiating Equation (3) w.r.t. $\mathbf{x}_i$ using the SPH formulation [Ihmsen et al. 2014b]:

$$\nabla p_i = \kappa_i^{\text{v}}\nabla\rho_i = \kappa_i^{\text{v}}\sum_j m_j\nabla W_{ij},$$

where κ_i^{v} is the stiffness parameter that we want to determine. Furthermore, we consider the pressure forces $\mathbf{F}^{\text{p}}_{j\leftarrow i}$ that act from particle i on the neighboring particles j to obtain a set of symmetric pressure forces which fulfill the condition $\mathbf{F}_i^{\text{p}} + \sum_j \mathbf{F}^{\text{p}}_{j\leftarrow i} = \mathbf{0}$. This means that all inner forces sum up to zero which is required to conserve momentum. The forces $\mathbf{F}^{\text{p}}_{j\leftarrow i}$ are computed analogously to Equation (4) except that we differentiate the pressure with respect to the neighboring position $\mathbf{x}_j$:

$$\mathbf{F}^{\text{p}}_{j\leftarrow i} = -\frac{m_i}{\rho_i}\frac{\partial p_i}{\partial \mathbf{x}_j} = \frac{m_i}{\rho_i}\kappa_i^{\text{v}} m_j\nabla W_{ij}. \tag{5}$$

The solver must determine pressure forces that change the velocities of the particles so that the condition $\frac{D\rho_i}{Dt} = 0$ is fulfilled. The current density change rate in particle i is computed by employing the SPH formulation of the divergence [Ihmsen et al. 2014b]:

$$\frac{D\rho_i}{Dt} = \sum_j m_j(\mathbf{v}_i - \mathbf{v}_j)\nabla W_{ij}. \tag{6}$$

This value should be zero after applying the symmetric pressure forces determined for particle i. The pressure forces cause the velocity changes $\Delta v_i = \Delta t\mathbf{F}_i^{\text{p}}/m_i$ and $\Delta v_j = \Delta t\mathbf{F}^{\text{p}}_{j\leftarrow i}/m_i$. Inserting these terms in Equation (6) yields:

$$\frac{D\rho_i}{Dt} = \Delta t\sum_j m_j\left(\frac{\mathbf{F}_i^{\text{p}}}{m_i} - \frac{\mathbf{F}^{\text{p}}_{j\leftarrow i}}{m_i}\right)\nabla W_{ij}. \tag{7}$$

Using Equations (4) and (5) in Equation (7) gives us an equation for the stiffness parameter κ_i^{v}:

$$\begin{aligned}
\frac{D\rho_i}{Dt} &= \Delta t\sum_j m_j\left(\frac{\mathbf{F}_i^{\text{p}}}{m_i} - \frac{\mathbf{F}^{\text{p}}_{j\leftarrow i}}{m_i}\right)\nabla W_{ij}\\
\frac{D\rho_i}{Dt} &= -\frac{\Delta t}{\rho_i}\sum_j m_j\left(\kappa_i^{\text{v}}\sum_j m_j\nabla W_{ij} + \kappa_i^{\text{v}} m_j\nabla W_{ij}\right)\nabla W_{ij}\\
\frac{D\rho_i}{Dt} &= -\kappa_i^{\text{v}}\frac{\Delta t}{\rho_i}\left(\left|\sum_j m_j\nabla W_{ij}\right|^2 + \sum_j |m_j\nabla W_{ij}|^2\right).
\end{aligned}$$

Solving for κ_i^{v} yields:

$$\kappa_i^{\text{v}} = \frac{1}{\Delta t}\frac{D\rho_i}{Dt}\cdot\underbrace{\frac{-\rho_i}{\left|\sum_j m_j\nabla W_{ij}\right|^2 + \sum_j |m_j\nabla W_{ij}|^2}}_{\alpha_i}, \tag{8}$$

where α_i is a factor that only depends on the current positions. The pressure forces computed with the stiffness parameter κ_i^{v} exactly fulfill the condition $\frac{D\rho_i}{Dt} = 0$ which means that the velocity field in the neighborhood of particle i is divergence-free. However, since the stiffness parameters of neighboring particles depend on each other, they are determined iteratively. Note that the denominator of α_i can cause instabilities in the special case that particle i has a low number of neighbors. To solve this problem we simply clamp the denominator if it gets too small. In our simulations we used a threshold of 10^{-6} which did not cause any visual artifacts.

Algorithm 2 Divergence-free solver

1: **function** CORRECTDIVERGENCEERROR(α, $\mathbf{v}^*$)
2: **while** $\left(\left(\frac{D\rho}{Dt}\right)_{\text{avg}} > \eta^{\text{v}}\right) \vee (\text{iter} < 1)$ **do**
3: **for all** particles i **do** // compute $\frac{D\rho}{Dt}$
4: $\frac{D\rho_i}{Dt} = -\rho_i \nabla \cdot \mathbf{v}_i^*$
5: **for all** particles i **do** // adapt velocities
6: $\kappa_i^{\text{v}} = \frac{1}{\Delta t}\frac{D\rho_i}{Dt}\alpha_i, \quad \kappa_j^{\text{v}} = \frac{1}{\Delta t}\frac{D\rho_j}{Dt}\alpha_j$
7: $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{\kappa_i^{\text{v}}}{\rho_i} + \frac{\kappa_j^{\text{v}}}{\rho_j}\right) \nabla W_{ij}$

Finally, we determine the total force $\mathbf{F}^{\text{p}}_{i,\text{total}}$ for particle i including the forces from neighboring particles j as

$$\mathbf{F}^{\text{p}}_{i,\text{total}} = \mathbf{F}^{\text{p}}_i + \sum_j \mathbf{F}^{\text{p}}_{i\leftarrow j} = -m_i \sum_j m_j \left(\frac{\kappa_i^{\text{v}}}{\rho_i} + \frac{\kappa_j^{\text{v}}}{\rho_j}\right) \nabla W_{ij},$$

where $\mathbf{F}^{\text{p}}_{i\leftarrow j}$ is computed analogously to Equation (5). Note that this pressure force is equivalent to the symmetric pressure force introduced by Monaghan [1992].

Our solver determines pressure forces in parallel using Jacobi iteration in order to make the complete velocity field divergence-free. Since the factors α_i only depend on the current positions, they can be precomputed before the iterative process and do not have to be updated in each iteration step. This yields computationally cheap iteration steps since α_i is the most complex term used in our solver. Note that since $\mathbf{F}^{\text{p}}_{j\leftarrow i} = 0$ if particle j is not dynamic, the equation for κ_i^{v} must be adapted accordingly for static boundary particles.

Algorithm 2 outlines our divergence-free solver. The solver performs at least one iteration and finishes when the average density change rate is smaller than a user-defined threshold η^{v}. The convergence can be significantly improved by performing a "warm start" of the solver. This means that we sum up the stiffness values κ_i^{v} for each particle. Then in the next time step we first evaluate line 7 for each particle using the resulting values before starting the divergence-free solver.

3.3 Constant Density Solver

While the solver described in the last section makes the velocity field divergence-free, our constant density solver minimizes the density error which is determined by the deviation $\rho - \rho_0$ of the actual density to the rest density. There already exist different pressure solvers that minimize the density deviation and that could be combined with our divergence-free solver, e.g. PCISPH [Solenthaler and Pajarola 2009] or IISPH [Ihmsen et al. 2014a]. However, in this work we introduce a new solver which works analogous to our divergence-free solver and therefore has the advantage that it can reuse the factor α. This reduces the computational effort significantly since the presented iterative method is particularly fast if the factor α is already known. Our new constant density solver employs a predictor-corrector scheme in order to obtain particle positions after the time integration that correct the density error. The key idea of this scheme is similar to the one of PCISPH. However, in contrast to PCISPH we do not use a precomputed prototype configuration with a filled neighborhood to solve the system.

We perform an Euler integration step for the density using Equation (6) in order to compute a prediction of the density error $\rho_i^* - \rho_0$:

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

Algorithm 3 Constant density solver

1: **function** CORRECTDENSITYERROR(α, $\mathbf{v}^*$)
2: **while** $(\rho_{\text{avg}} - \rho_0 > \eta) \vee (\text{iter} < 2)$ **do**
3: **for all** particles i **do** // predict density
4: compute ρ_i^*
5: **for all** particles i **do** // adapt velocities
6: $\kappa_i = \frac{\rho_i^* - \rho_0}{\Delta t^2}\alpha_i, \quad \kappa_j = \frac{\rho_j^* - \rho_0}{\Delta t^2}\alpha_j$
7: $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j}\right) \nabla W_{ij}$

Analogous to Equation (7), we determine pressure forces that correct this density error by solving:

$$\rho_i^* - \rho_0 = \Delta t^2 \sum_j m_j \left(\frac{\mathbf{F}^{\text{p}}_i}{m_i} - \frac{\mathbf{F}^{\text{p}}_{j\leftarrow i}}{m_i}\right) \nabla W_{ij}. \tag{9}$$

This yields the following stiffness parameter:

$$\kappa_i = \frac{1}{\Delta t^2}(\rho_i^* - \rho_0)\alpha_i.$$

Our implicit pressure solver is outlined in Algorithm 3. Note that analogous to the divergence-free solver we perform a warm start to improve the convergence of the solver.

3.4 Kernel

The kernel function used in SPH simulations is an approximation of the Gaussian. In previous works several kernel functions were introduced such as the poly6 kernel, the spiky kernel and the cubic spline kernel. In some works even different kernels are used to compute W_{ij} and its gradient ∇W_{ij}, e.g. in [Müller et al. 2003]. However, in our predictor-corrector scheme it is important to use the same kernel for both since otherwise the prediction and correction steps do not fit together. In our work we use the cubic spline kernel [Monaghan 1992].

In SPH simulations typically a kernel with compact support is used which vanishes at a finite distance also known as the support radius h. In general a kernel function can be written as $W_h(q(\mathbf{x}))$ with $q = \frac{\|\mathbf{x}\|}{h}$. This means that the kernel is only non-zero for $0 \leq q < 1$. Hence, the first spatial derivative of such a kernel $\nabla W_h(q(\mathbf{x}))$ has the same compact support.

Especially the evaluation of the kernel gradient is one of the most time consuming tasks in a simulation step since gradients have to be determined for the whole neighborhood of each particle in each iteration step of both solvers. Moreover, they are required for different non-pressure forces and to compute the factors α_i. However, storing the gradients of all neighborhoods requires much memory and is not recommended for large scenes. To speed up the simulation we propose a faster computation of the kernel and its gradient by using precomputed lookup tables. The idea of using lookup tables for a fast function evaluation is not new. However, to the best of our knowledge it has not been employed yet in SPH simulations.

Since the kernel W_h is a scalar function with compact support, a lookup table is generated easily by a regular sampling. However, the gradient is handled differently. Instead of sampling the vector function ∇W_h in all three dimensions, we introduce a scalar function $g(q)$ to reduce the computational effort and the memory requirements:

$$\nabla W_h(q(\mathbf{x})) = \mathbf{x} \cdot g(q) \quad \text{with} \quad g(q) = \frac{\partial W_h}{\partial q} \cdot \frac{1}{h\|x\|}.$$

Δt [ms]	IISPH		PBF		PCISPH	
	solver	total	solver	total	solver	total
4.0	6.9	6.2	13.4	12.0	23.9	21.2
2.0	5.3	4.5	10.6	8.8	21.4	17.4
1.0	2.3	2.1	3.7	3.1	7.7	6.3
0.5	1.1	1.1	1.2	1.1	2.4	2.1
0.25	1.1	1.1	0.9	1.0	1.4	1.3

Table 2: *This table shows the speedup factors of DFSPH in comparison to IISPH, PBF and PCISPH based on the measured values in Table 1.*

The function $g(q)$ can also be sampled regularly to obtain a corresponding lookup table. Finally, a gradient is determined by a single lookup and a multiplication with $\mathbf{x}$.

The usage of lookup tables is a simple but efficient trick which can also be employed to speed up other SPH methods. In Section 4 we discuss details about the sampling distance, the approximation error and the performance gain.

4 Results

All timings in this section were measured on two Intel Xeon E5-2697 processors with 2.7 GHz, 12 cores per processor and 64GB RAM. We parallelized our fluid simulation using OpenMP. In our simulations we performed the neighborhood search using the parallel method of Ihmsen et al. [2011] and the boundary handling using the rigid-fluid coupling of Akinci et al. [2012]. To simulate the viscosity of the fluid we employed the XSPH variant of Schechter and Bridson [2012]. We successfully tested our method in combination with the surface tension models of Becker and Teschner [2007], Akinci et al. [2013] and He et al. [2014]. However, for the results in this paper we solely used the surface tension model of Akinci et al. [2013]. Particle deficiency at free surfaces is a well-known problem in SPH simulations which causes particle clustering. In the simulations we solved this issue by clamping negative pressures to zero which is a common solution, see e.g. [Ihmsen et al. 2014a]. We enforced an average density error of less than $0.01\,\%$ and a density error due to the density change rate of less than $0.1\,\%$ in all simulations. Unless otherwise stated, we used adaptive time-stepping in the simulations according to the CFL condition (see Section 3.1).

Performance We performed a breaking dam simulation with 125k particles in order to compare the performance of our novel method with IISPH, PBF and PCISPH. The particle radius was 0.02 m and we used different fixed time step sizes. Table 1 summarizes the performance measurements for a simulation over one second. In Table 2 the speedup factors are shown. Note that in [Ihmsen et al. 2014a] a similar scenario was used for a performance comparison between PCISPH and IISPH and comparable results were measured.

By fulfilling the divergence-free condition and the constant density condition at the same time, the density error is kept small during the simulation which decreases the required number of solver iterations significantly. Using a warm start in iterative solvers is another way to reduce the number of iterations. In our simulation we initialize the DFSPH solvers with the sum of the stiffness values of the last time step. This reduced the number of iterations by a factor of approximately 3 in the dam break scenario. While DFSPH performs best for a full warm start, IISPH has its best performance when multiplying the solution of the last step with a factor of 0.5 [Ihmsen et al. 2014a]. Due to the divergence-free velocity field in our simulation and the performed warm start of the solver, we measured

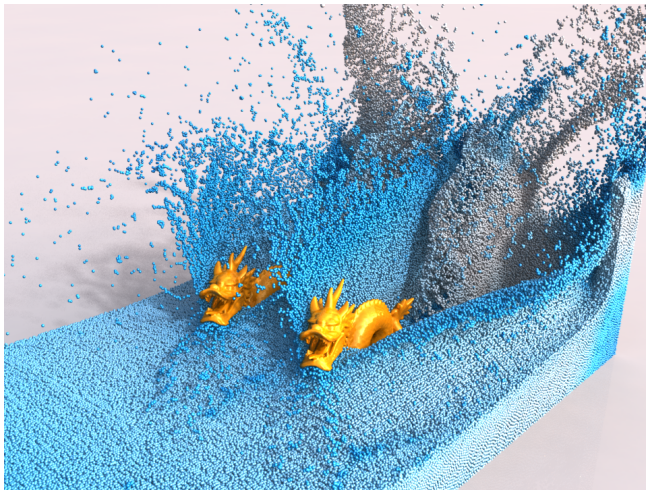

Figure 3: *Breaking dam model (2.3 million fluid particles) with two dragon models and a moving wall. The velocity field is color coded: blue is the minimum and white is the maximum.*

	neigh. search	α	$\mathbf{F}^{\text{adv}}$	const. density	div.-free	total
dragons	0.4	0.1	0.3	0.7	0.6	2.1
canyon	1.2	0.2	0.7	1.9	1.3	5.3

Table 3: *Average computation times (in seconds) per simulation step of the neighborhood search, the computation of α, the computation of all non-pressure forces $\mathbf{F}^{adv}$, the constant density solver and the divergence-free solver for the dam break simulation (see Figure 3) and the canyon simulation (see Figure 1, right).*

speedup factors of 6.9 in comparison to IISPH up to 23.9 in comparison to PCISPH for a time step size of 4 ms. The DFSPH solvers required only 4.5 and 2.8 iterations to correct the density error and the divergence error, respectively, while the second best method IISPH already required 50.5 iterations. In our experiment we measured a smaller speedup for smaller time step sizes since for DFSPH often the minimum number of iterations was used. However, for scenarios with more particles, where the number of iterations lies clearly above the minimum value, the speedup is also larger for small step sizes. In our experiments we noticed that our method performs best when using a time step size so that the number of iterations ranges between 2 and 20 iterations.

As Table 1 shows, our method has its best overall performance for larger time steps than IISPH, PBF and PCISPH. This has another advantage: When using larger time steps, the computationally expensive neighborhood search has to be performed less frequently.

In the breaking dam simulation we measured a performance gain of approximately 30 % by using the kernel optimization introduced in Section 3.4. For the sampling of the kernel function and its gradient we used 1000 sample points and measured an maximum local error of less than 10^{-11}. Since the implementation is very simple and the error is negligible, this is a nice extension for our method.

In order to measure the performance of DFSPH in simulations with large numbers of fluid particles, we first simulated a breaking dam scenario with two dragon models and a moving wall (see Figure 3). The model consists of 2.3 million fluid particles and 0.7 million boundary particles. Additionally, we simulated a real breaking dam with 5 million particles flowing through a canyon sampled by 40 million boundary particles (see Figure 1, right). The average com-

Δt [ms]	DFSPH			IISPH			PBF			PCISPH		
	iter. (cd/df)	solver [s]	total [s]	iter.	solver [s]	total [s]	iter.	solver [s]	total [s]	iter.	solver [s]	total [s]
4.0	4.5/2.8	45.2	**51.3**	50.5	312.1	318.1	105.7	607.1	613.5	160.0	1079.1	1085.7
2.0	2.1/1.3	47.8	59.4	21.4	256.4	267.9	42.7	508.4	520.7	73.9	1021.2	1033.5
1.0	2.0/1.0	85.0	107.5	7.3	197.9	**220.7**	13.2	314.8	338.0	23.9	656.9	680.0
0.5	2.0/1.0	164.1	210.8	2.3	182.3	225.6	3.4	194.1	**240.5**	6.7	394.9	**440.9**
0.25	2.0/1.0	288.3	372.0	2.0	322.5	402.2	2.0	263.3	354.0	3.0	409.3	498.0

Table 1: *Comparison of DFSPH with IISPH, PBF and PCISPH using different fixed time step sizes for a breaking dam scenario with 125k particles (see Figure 2). The table shows the average number of required iterations, the total computation time of the solvers and the total time including the neighborhood search in a simulation over one second. The lowest total computation times are marked bold. Note that for DFSPH we used the sum of the times needed by the divergence-free solver and the constant density solver since the iteration steps of both solvers need almost the same time. For DFSPH the column with the iteration count contains the values for the constant density solver (cd) and the divergence-free solver (df).*

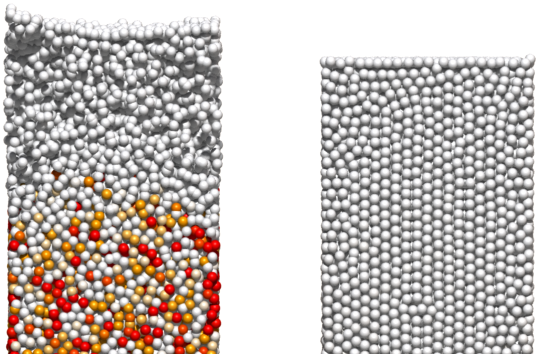

Figure 4: *Top of a resting fluid pillar with 80k particles. The large divergence errors in the IISPH simulation lead to jumping artifacts (left). DFSPH maintains a divergence-free velocity field and therefore allows a stable simulation without artifacts (right). The divergence errors are color coded: white is the minimum and red is the maximum.*

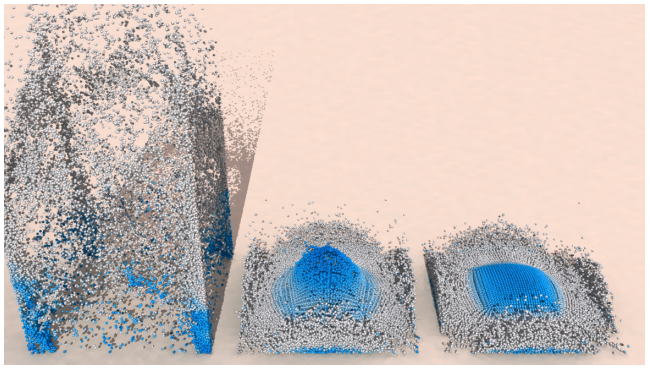

Figure 5: *Stability comparison in a simulation where a cube with 27k particles is falling on the ground with a high velocity. PCISPH (left) gets instable due to the impact and several fluid particles pass through the boundary, DFSPH without divergence-free solver (middle) shows artifacts and DFSPH with divergence-free solver stays stable. The same color coding as in Figure 3 is used.*

putation times for the main steps in Algorithm 1 are shown in Table 3 for both simulations.

Memory Requirements An advantage of our method especially when simulating large scale scenarios with millions of particles is that the memory requirements are low. Per particle we only have to store the scalar value α_i which is used by both solvers. When performing a warm start, we have to store one additional scalar for each solver. For comparison, IISPH requires seven scalar values for the solver and one for the warm start. Hence, our method requires significantly less memory than IISPH.

Stability In the following we show that current state-of-the-art pressure solvers which do not enforce a divergence-free velocity field explicitly cannot fulfill the condition $\frac{D\rho}{Dt} = 0$ as demanded by the continuity equation. This can lead to instabilities in simulations, especially when particles have high velocities.

Figure 2 compares the velocity divergence error of IISPH and DFSPH in a breaking dam simulation with 125k particles. This figure shows that our approach maintains a divergence-free velocity field in contrast to methods which do not correct the velocity divergence. The maximum local divergence error of IISPH was 108.3 while the one of DFSPH was only 1.9. In another divergence comparison we simulated a resting fluid pillar with 80k fluid particles (see Figure 4). Since the CFL condition returns arbitrary large values for the resting particles, we restricted the time step size to a maximum of 5 ms. Large divergence errors of up to 72.9 in the IISPH simulation in combination with large time steps led to visual artifacts: fluid particles sometimes jump up due to the errors (see Figure 4, left). The maximum local divergence error of DFSPH was 2.5 which allowed a stable simulation without artifacts.

We performed a stability test with a cube of 27k fast moving particles falling on the ground to show how this influences the stability (see Figure 5). The time step size was chosen according to the CFL condition. In this test we compared PCISPH and DFSPH. The test with DFSPH was performed twice. In the first simulation we deactivated our divergence-free solver and activated it in the second one in order to show that the simulation is more stable with a divergence-free velocity field. In the simulation PCISPH got instable due to the impact, DFSPH without the divergence-free solver showed artifacts and DFSPH using both solvers stayed stable. The instability in the PCISPH simulation even led to several fluid particles which passed through the boundary. For a stable simulation of this scenario with PCISPH the time step size had to be decreased considerably which reduced the overall performance significantly.

In order to demonstrate the stability of our approach in a simulation with dynamic boundaries, we integrated the Bullet physics library [Coumans 2014] in our simulator. In another stability experiment we dropped several rigid bodies with different velocities in a breaking dam scenario with 330k particles. The result is shown in Figure 6 and the accompanying video. Finally, Figure 1 (left)

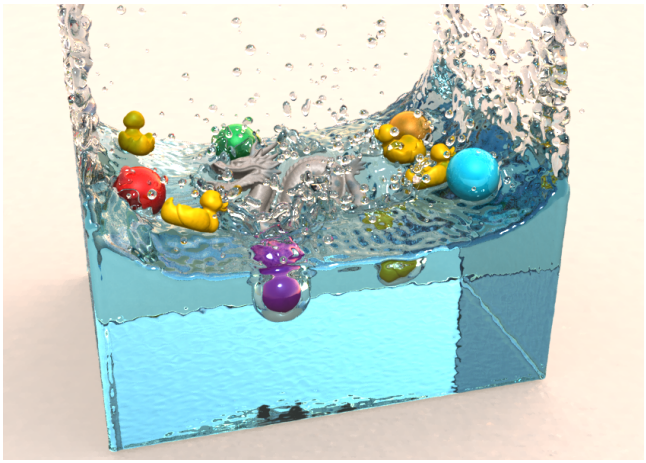

Figure 6: *Two-way coupling of 330k particles with dynamic rigid bodies.*

demonstrates that DFSPH even allows a stable simulation with 2.4 million fast moving particles.

5 Conclusion and Future Work

In this paper we presented a novel implicit SPH simulation method for incompressible fluids that prevents volume compression and enforces a divergence-free velocity field. The divergence-free configuration leads to a significantly faster convergence of the implicit pressure solver resulting in a substantial speedup compared to state-of-the-art methods. Moreover, we demonstrated that our method is able to handle scenarios with millions of fast moving particles robustly and produces a convincing physical behavior.

Our method also has some limitations. In SPH simulations the density near a free surface is underestimated which causes unnatural particle clustering artifacts. In our implementation this problem is solved by clamping negative pressures to zero. However, a better solution would be to introduce ghost particles as suggested by Schechter and Bridson [2012] in order to prevent particle deficiencies which improves the physical behavior of the fluid. Moreover, without pressure clamping more sophisticated solving algorithms like the conjugate gradient method could be employed and would enhance the converge rate even more. This is a goal for our future research. In this context we also plan to investigate if DFSPH can improve the stability of multi-phase simulations with high density contrasts.

Acknowledgements

The work of the authors is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at TU Darmstadt. We would like to thank Markus Ihmsen and Matthias Teschner for their help.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3, 48.

AKINCI, N., IHMSEN, M., AKINCI, G., SOLENTHALER, B., AND TESCHNER, M. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics 31*, 4, 62:1–62:8.

AKINCI, N., AKINCI, G., AND TESCHNER, M. 2013. Versatile surface tension and adhesion for sph fluids. *ACM Transactions on Graphics 32*, 6, 182:1–182:8.

ANDO, R., THÜREY, N., AND WOJTAN, C. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics 32*, 103:1–103:10.

BECKER, M., AND TESCHNER, M. 2007. Weakly compressible SPH for free surface flows. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 1–8.

BODIN, K., LACOURSIÈRE, C., AND SERVIN, M. 2012. Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics 18*, 516–526.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A K Peters / CRC Press.

CHORIN, A. J. 1968. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation 22*, 745–762.

COUMANS, E., 2014. The bullet physics library. `http://www.bulletphysics.org`.

CUMMINS, S. J., AND RUDMAN, M. 1999. An SPH Projection Method. *Journal of Computational Physics 152*, 584–607.

DESBRUN, M., AND GASCUEL, M.-P. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation*, 61–76.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. *ACM Transactions on Graphics 28*, 12–17.

HE, X., LIU, N., LI, S., WANG, H., AND WANG, G. 2012. Local poisson SPH for viscous incompressible fluids. *Computer Graphics Forum 31*, 1948–1958.

HE, X., WANG, H., ZHANG, F., WANG, H., WANG, G., AND ZHOU, K. 2014. Robust simulation of sparsely sampled thin features in sph-based free surface flows. *ACM Transactions on Graphics 34*, 1 (Dec.), 7:1–7:9.

HU, X., AND ADAMS, N. 2007. An incompressible multi-phase SPH method. *Journal of Computational Physics 227*, 264–278.

IHMSEN, M., AKINCI, N., BECKER, M., AND TESCHNER, M. 2011. A parallel sph implementation on multi-core cpus. *Computer Graphics Forum 30*, 1, 99–112.

IHMSEN, M., CORNELIS, J., SOLENTHALER, B., HORVATH, C., AND TESCHNER, M. 2014. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 426–435.

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., AND TESCHNER, M. 2014. SPH Fluids in Computer Graphics. *Eurographics (State of the Art Reports)*, 21–42.

KANG, N., AND SAGONG, D. 2014. Incompressible SPH using the Divergence-Free Condition. *Computer Graphics Forum 33*, 7, 219–228.

LOSASSO, F., TALTON, J. O., KWATRA, N., AND FEDKIW, R. 2008. Two-way coupled SPH and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 797–804.

MACKLIN, M., AND MÜLLER, M. 2013. Position Based Fluids. *ACM Transactions on Graphics 32*, 4, 1–5.

MONAGHAN, J. J. 1992. Smoothed particle hydrodynamics. *Ann. Rev. of Astron. and Astrophys. 30*, 543–574.

MONAGHAN, J. 1994. Simulating Free Surface Flows with SPH. *Journal of Computational Physics 110*, 399–406.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 154–159.

RAVEENDRAN, K., WOJTAN, C., AND TURK, G. 2011. Hybrid smoothed particle hydrodynamics. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 33–42.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost SPH for animating water. *ACM Transactions on Graphics 31*, 4, 61:1–61:8.

SIN, F., BARGTEIL, A. W., AND HODGINS, J. K. 2009. A point-based method for animating incompressible flow. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 247.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible SPH. *ACM Transactions on Graphics 28*, 3, 40:1–40:6.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (July), 965–972.