Implicit Position-Based Fluids

ELIE DIAZ, University of Utah, USA
JERRY HSU, University of Utah, USA
EISEN MONTALVO-RUIZ, University of Utah, USA
CHRIS GILES, Roblox, USA
CEM YUKSEL, University of Utah, USA

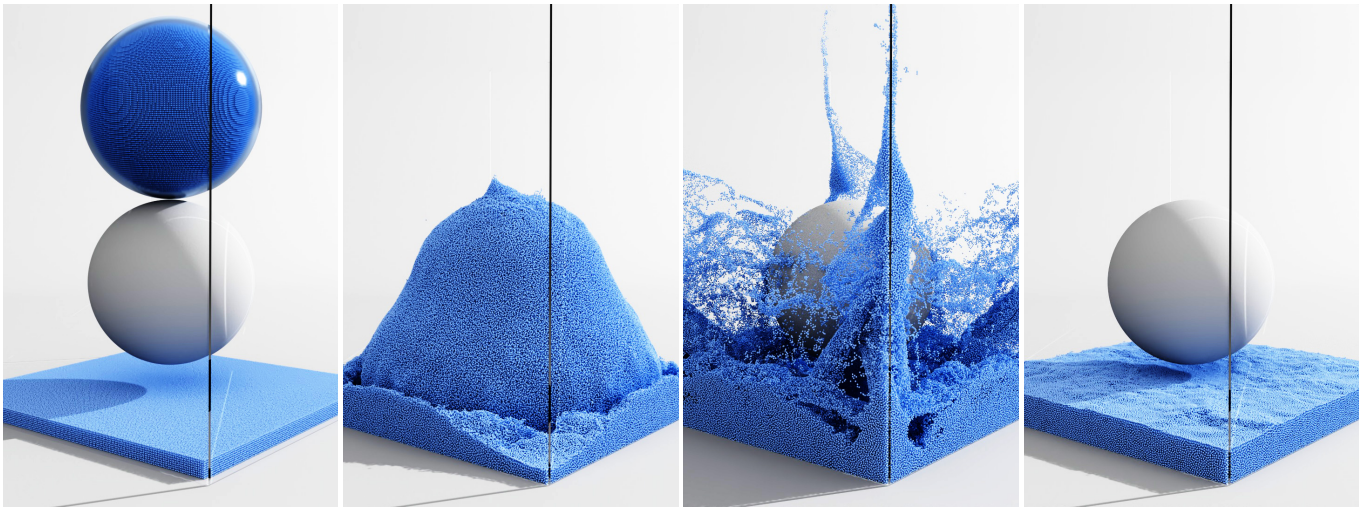


Fig. 1. *Simulation of one million fluid particles falling onto a sphere, using our method on an NVIDIA RTX 4090 GPU. The simulation ran at an average of 159 milliseconds per frame.*

The efficient simulation of incompressible fluids remains a difficult and open problem. Prior works often make various tradeoffs between incompressibility, stability, and cost. Yet, it is rare to obtain all three. In this paper, we introduce a novel incompressible Smoothed Particle Hydrodynamics (SPH) scheme which uses a second-order implicit descent scheme to optimize a variational energy specially formulated to approach incompressibility. We demonstrate that our method is superior in both incompressibility and stability with a minimal cost to computational budget. Furthermore, we demonstrate that our method is unconditionally stable even under extreme time steps, making it suitable for interactive applications.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Smooth Particle Hydrodynamics, Fluid Simulation, Incompressible Fluids, Vertex Block Descent

ACM Reference Format:
Elie Diaz, Jerry Hsu, Eisen Montalvo-Ruiz, Chris Giles, and Cem Yuksel. 2025. Implicit Position-Based Fluids. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3757377.3764005

Authors' Contact Information: Elie Diaz, University of Utah, Salt Lake City, UT, USA, elie.diaz@utah.edu; Jerry Hsu, University of Utah, Salt Lake City, UT, USA, jerry.hsu.research@gmail.com; Eisen Montalvo-Ruiz, University of Utah, Salt Lake City, UT, USA, eisen.montalvo@utah.edu; Chris Giles, Roblox, San Mateo, CA, USA, cgiles@roblox.com; Cem Yuksel, University of Utah, Salt Lake City, UT, USA, cem@cemyuksel.com.

SA Conference Papers '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/25/12
https://doi.org/10.1145/3757377.3764005

1 Introduction

Lagrangian particles based on smoothed particle hydrodynamics (SPH) [Müller et al. 2003] has been a popular technique for fluid simulations in computer graphics. Though it is often used for simulating incompressible fluids (such as water in liquid form), achieving incompressibility with SPH is a fundamental challenge. This is because, in the SPH formulation, incompressibility corresponds to hard constraints which maintain fluid density. As a result, explicit velocity integration [Müller et al. 2003] expectedly requires extremely small time steps for stability and even smaller timesteps to achieve incompressibility using highly nonlinear constraints [Becker and Teschner 2007]. Therefore, most recent work has adapted variants of semi-implicit velocity integrations that adaptively adjust the constraint stiffness per particle [Bender and Koschier 2015; Ihmsen et al. 2014]. Unfortunately, these approaches still struggle with stability issues or excessive compression with complex simulations, requiring relatively small time steps. Thus hindering one of the strongest advantage of SPH-based methods: their computational performance with parallel computing.

One notable exception has been the *position-based fluids* (PBF) method [Macklin and Müller 2013], which offers exceptional stability with larger time steps by converting SPH into a constraint formulation similar to position-based dynamics [Macklin et al. 2016;

Müller et al. 2007]. However, the first-order approximations used in PBF degrade the quality of the animations, substantially dampening the motion and deviating from physically plausible results.

In this paper, we introduce another position-based formulation for SPH, starting with the variational form of implicit Euler to offer a fully implicit integration for SPH. Therefore, we call our method *implicit position-based fluids* (IPBF), although our integration differs significantly from PBF. We use relaxed Jacobi iterations with a second-order descent formulation, similar to *vertex block descent* (VBD) [Chen et al. 2024], which avoids excessive damping while achieving unconditional stability and allowing larger time steps. In addition, we can model incompressibility using arbitrarily stiff density constraints, achieving improved incompressibility at larger time steps.

Stiff systems are known to inject energy, particularly when simulated with limited iteration counts that prevent convergence. IPBF is no exception. Though, even with infinite stiffness, the energy injection is relatively minor and does not lead to instabilities, it prevents the fluid from coming to a rest, which might be undesirable for some applications. Therefore, we also introduce a simple damping formulation that extracts kinetic energy by comparing to the solution of a less stiff constraint. This optional damping is designed to have minimal impact in animations until the motion subsides.

Our experiments show that under the same computation budget, our method provides numerical stability superior to velocity-based methods and improved incompressibility behavior against prior SPH-based methods, including PBF. Notably, we achieve these without dampening the fluid motion or producing unnatural animations.

2 Prior Work

SPH is a method invented originally for simulating interstellar flows [Gingold and Monaghan 1977; Lucy 1977] that discretizes fluids into particles that store mass, momentum, and other field quantities. SPH was introduced to computer graphics applications by Müller et al. [2003], and has since led to a large body of research into improving the estimation techniques and force calculations available for SPH-based solvers, as described in Koschier et al. [2022]. The pressure solve in particular has been the subject of many works, as maintaining incompressibility in fluids leads to realistic and dynamic simulations.

Standard SPH [Müller et al. 2003] introduced a state equation pressure solve for incompressible fluids, where each particle calculates a penalty force to compensate for density deviation from rest. Later improved upon with a non-linear formulation [Becker and Teschner 2007], these state equation solvers require a high stiffness constants to achieve realistic fluid flows, forcing the solvers to use low timesteps to maintain numerical stability.

Improvements to maintaining stability with higher timesteps were made by iterative solvers that use a predictive-correction scheme to calculate penalty forces based on density fluctuations due to non-pressure forces [He et al. 2012; Solenthaler and Pajarola 2009]. Instead of user-defined stiffnesses, some solvers, such as *Implicit Incompressible SPH* (IISPH) calculate per-particle stiffness parameters when solving for updated pressure forces [Ihmsen et al. 2014]. While improving on stability at higher timesteps, these methods fail to remain incompressible when presented with minimal iteration counts and large residual build-up.

Methods such as *Divergence-Free SPH* (DFSPH) use two solvers to address both the constant density constraint and the divergence-free velocity constraint that are present with incompressible fluids [Bender and Koschier 2015], can improve convergence, but also risks high energy injection when presented with existing error in the system. Recovering from such high-energy states proves impossible without enough iterations or low enough timesteps.

Position-based solvers, such as PBF [Macklin and Müller 2013] and *Semi-Implicit SPH* (SISPH) [He et al. 2025] are a promising approach for maintaining a simulation stable with large timesteps and low iteration counts, usually taking advantage of substepping or careful step size calculations to prevent overcorrection when solving for optimal positions that can minimize density fluctuations. Similar to our method, SISPH is derived starting from the variational form of implicit Euler, but uses a *semi-implicit successive substitution* method to minimize the energy of the system, whereas our solution uses a second-order descent formulation with local Newton iterations, similar to VBD [Chen et al. 2024].

Other improvements in SPH tackle other outstanding problems in the SPH approach to incompressible fluids, such as handling highly viscous fluid motion [Peer et al. 2015] and improving tensile instability in SPH fluid simulations [He et al. 2020; Liu et al. 2024]. Indeed, our work functions in parallel to these improvements, and they can be used in conjunction to our solver to further improve fluid motion.

3 Implicit Position-Based Fluids

We formulate our implicit integration for SPH using the variational form of implicit Euler [Gast and Schroeder 2015]. It allows us to simulate incompressibility using stiff constraints. We solve the resulting minimization problem using relaxed Jacobi iterations with position updates. Finally, we discuss approximations to the second-order terms to maintain stability and introduce a damping formulation to counteract the energy injection from solves with high stiffness.

In this section, we begin with deriving the implicit Euler integration with the variational form (Section 3.1), then we describe how we adapt the variational energy to account for arbitrarily large stiffness values (Section 3.2). We then demonstrate how we integrate the results our numerical computation and update the resulting positions (Section 3.4). Finally, to further prevent instability due to indeterminate matrices, we describe a Hessian approximation technique (Section 3.5) and an artificial damping formulation to counteract the injected energy from high stiffness solves. (Section 3.6).

3.1 Implicit Euler Integration for SPH

The variational form of implicit Euler aims to minimize the variational energy Ψ for each time step. In a position-based formulation, this can be written as

$$\mathbf{x} = \arg\min_{\mathbf{x}} \Psi(\mathbf{x}) \,, \tag{1}$$

such that the positions of all particles $\mathbf{x}$ is determined by minimizing the variational energy

$$\Psi(\mathbf{x}) = \frac{1}{2h^2}||\mathbf{x} - \mathbf{y}||_M^2 + E(\mathbf{x}) \,, \tag{2}$$

where the first term is the *momentum potential* that contains the time step size h and the mass-weighted norm $||\cdot||_M$ of the difference between positions of all particles $\mathbf{x}$ and their inertial positions

$$\mathbf{y} = \mathbf{x}^t + h\mathbf{v}^t + h^2\mathbf{a}^\star \tag{3}$$

calculated using the particle positions and velocities at the beginning of the time step $\mathbf{x}^t$ and $\mathbf{v}^t$, and a constant acceleration term $\mathbf{a}^\star$.

The last term $E(\mathbf{x})$ is the *total internal potential* and it includes the potential energy of all internal forces that are not included in $\mathbf{a}^\star$. It can be written as a sum of all local internal potential terms $E_i(\mathbf{x})$ calculated around each particle i, such that

$$E(\mathbf{x}) = \sum_i E_i(\mathbf{x}) \,. \tag{4}$$

The acceleration due to gravity and other external forces that are constant over the time step should be included in $\mathbf{a}^\star$. By making the assumption that the viscosity forces remain constant within the time step, we include them in $\mathbf{a}^\star$ such that all that remains for $E(\mathbf{x})$ is the potential energy due to pressure.

Following typical SPH formulations, we can model the pressure energy as a quadratic potential

$$E_i(\mathbf{x}) = \frac{1}{2}\, k \left(C_i(\mathbf{x})\right)^2 \,, \tag{5}$$

where k is the stiffness and $C_i(\mathbf{x})$ is the density constraint, which is often referred to as the *state equation* in SPH. There are different formulations of the state equation in prior work and we opt for the simple form of

$$C_i(\mathbf{x}) = \frac{\rho_i}{\rho} - 1 \tag{6}$$

where ρ_i is the fluid density around the particle i and ρ is the default density. This simple constraint formulation penalizes deviations from the rest density and works well in practice with our implicit solution. We found that it offers better stability than alternative forms that use (highly) nonlinear terms. Although, it should not be enforced blindly. Specifically, at the fluid surface, the density should naturally fade to 0. In order to avoid clumping due to negative pressures, a typical solution is to clamp the constraint to zero [Koschier et al. 2022] and we use this approach in all our tests.

3.2 Incompressibility

For nearly incompressible fluids like water, small variations in density correspond to strong pressure forces. This requires using a large value for the stiffness term k. While large k values lead to instabilities with explicit integration, they can also cause numerical issues with implicit integration, particularly when using a low-precision floating-point representation.

To avoid this, we normalize the variational energy by dividing it by k, resulting in

$$\bar{\Psi}(\mathbf{x}) = \frac{\alpha}{2h^2}||\mathbf{x} - \mathbf{y}||_M^2 + \frac{1}{2}\sum_i \left(C_i(\mathbf{x})\right)^2 \,, \tag{7}$$

where $\alpha = 1/k$ is the *compliance* parameter. This does not alter the results, since $\mathbf{x}$ that minimizes $\bar{\Psi}(\mathbf{x})$ also minimizes $\Psi(\mathbf{x})$.

This normalization allows us to use arbitrarily large stiffness, including $k = \infty$ with $\alpha = 0$, which effectively removes the first term in the minimization. Yet, care must be taken when using $\alpha = 0$, as the minimization includes no term that would prevent arbitrary energy injection to satisfy the density constraint. This means that when $\alpha = 0$, we cannot guarantee convergence to a physically meaningful solution even under infinitely small time steps and infinitely many iterations.

In practice, however, when using relatively large time steps with limited iteration counts, the numerical solution with a sufficiently small α would be close (or be even identical) to $\alpha = 0$. Therefore, we omit the momentum term using $\alpha = 0$ in most of our results, unless otherwise specified, but we begin our iterations by initializing $\mathbf{x}$ to $\mathbf{y}$, to ensure that the existing fluid momentum has some impact. In fact, this is recommended for any small value of α.

3.3 Numerical Integration

To solve the minimization problem in Equation 1, we follow a similar approach to *vertex block descent* (VBD) [Chen et al. 2024] by assuming that all other particles are fixed. Then, for each particle, the global variation energy can be reduced to a local variational energy per particle

$$\Psi_i(\mathbf{x}) = \alpha\frac{m_i}{2h^2}||\mathbf{x}_i - \mathbf{y}_i||^2 + \frac{1}{2}\sum_{j\in\mathcal{P}_i}\left(C_i(\mathbf{x})\right)^2 \,, \tag{8}$$

where m_i, $\mathbf{x}_i$, and $\mathbf{y}_i$ are the mass, position, and inertial position of the particle i, respectively, and $\mathcal{P}_i$ is the set of all particles around particle i within the SPH kernel radius. Thus, $\mathcal{P}_i$ includes all particles that consider particle i for evaluating their densities.

As in VBD, minimizing the local variational energy $\Psi_i(\mathbf{x})$ by moving only $\mathbf{x}_i$ guarantees a reduction in the global variational energy $\Psi(\mathbf{x})$. This forms an efficient numerical integration method via Gauss-Seidel iterations.

However, Gauss-Seidel can be prohibitively expensive for SPH under typical conditions as $\mathcal{P}_i$ can include hundreds of thousands of particles, severely limiting the potential for parallel computation. Moreover, after moving each particle, the densities of all of its neighbors must be updated, which also incurs a substantial computational cost and potential data hazards when the same neighboring particle are updated. This would mean either a further reduction in parallelization or using atomic operations that hinder performance.

Therefore, we opt for a relaxed Jacobi scheme, which is common with SPH-based methods [Ihmsen et al. 2014; Solenthaler and Pajarola 2009]. After computing a position update $\Delta\mathbf{x}_i$ for all particles, we apply half of it using $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta\mathbf{x}_i/2$ in parallel.

3.4 Position Updates

Similar to VBD, we adjust particle positions using Newton's Method, which involves solving the second-order linear system

$$\mathbf{H}_i\Delta\mathbf{x}_i = \mathbf{f}_i \tag{9}$$

where $\mathbf{f}_i$ is the force calculated from the negative spatial derivative of the variational form

$$\mathbf{f}_i = -\frac{\partial \bar{\Psi}_i(\mathbf{x})}{\partial \mathbf{x}_i} = -\alpha \frac{m_i}{h^2}(\mathbf{x}_i - \mathbf{y}_i) - \frac{1}{2}\sum_{j \in \mathcal{P}_i} \frac{\partial E_j(\mathbf{x})}{\partial \mathbf{x}_i} \tag{10}$$

and $\mathbf{H}_i$ is the Hessian of the variational form

$$\mathbf{H}_i = \frac{\partial^2 \bar{\Psi}_i(\mathbf{x})}{\partial \mathbf{x}_i^2} = \alpha \frac{m_i}{h^2}\mathbf{I} + \frac{1}{2}\sum_{j \in \mathcal{P}_i} \frac{\partial^2 E_j(\mathbf{x})}{\partial \mathbf{x}_i^2} \ . \tag{11}$$

SPH uses smoothing kernels W to calculate the particle densities ρ_i from the positions and masses of the neighboring particles

$$\rho_i = \sum_{j \in \mathcal{P}_i} m_j W(\mathbf{x}_i - \mathbf{x}_j) \ . \tag{12}$$

Thus, the derivatives of the density constraints can be calculated using the derivatives of the kernel function, such that

$$\frac{\partial C_i}{\partial \mathbf{x}_i} = \frac{1}{\rho}\sum_j m_j \frac{\partial W_{ij}}{\partial \mathbf{x}_i} \quad \text{and} \quad \frac{\partial^2 C_i}{\partial \mathbf{x}_i^2} = \frac{1}{\rho}\sum_j m_j \frac{\partial^2 W_{ij}}{\partial \mathbf{x}_i^2} \ , \tag{13}$$

where $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j)$. We use a cubic spline kernel in our implementation [Koschier et al. 2019], but our formulation works with any SPH kernel formulation.

The position change is calculated with a 3×3 matrix inverse

$$\Delta \mathbf{x}_i = \mathbf{H}_i^{-1}\, \mathbf{f}_i \ . \tag{14}$$

3.5 Approximate Hessians

SPH-based Hessians of pressure energies $E_j(\mathbf{x})$ take the form

$$\frac{\partial^2 E_j(\mathbf{x})}{\partial \mathbf{x}_i^2} = \left(\frac{\partial C_j(\mathbf{x})}{\partial \mathbf{x}_i}\right)^{\mathrm{T}} \frac{\partial C_j(\mathbf{x})}{\partial \mathbf{x}_i} + \mathbf{C}_j(\mathbf{x}) \frac{\partial^2 C_j(\mathbf{x})}{\partial \mathbf{x}_i^2} \ . \tag{15}$$

Here, the first term is always symmetric positive-definite, but the second term is not guaranteed to be positive-definite and invertible, especially when using typical pressure clamping techniques to address the particle deficiency problem.

We avoid this problem by approximating the Hessian, replacing the second term with a diagonal approximation based on the norm of its columns [Andrews et al. 2017]. This treatment guarantees that the approximated Hessians are always invertible.

3.6 Artificial Damping

Our implicit integration and position-based updates allow stable SPH simulations even with large time steps and limited iteration counts that prevent convergence. The remaining constraint error from the previous time step does not lead to numerical explosion, but it can inject energy, depending on the stiffness of the density constraint. Although implicit Euler is known to introduce numerical damping, the injected energy from simulating incompressible fluids with highly stiff constraints and limited iteration counts can overwhelm this effect and prevent the simulation from ever reaching a steady solution without motion. This can be undesirable for some applications. Therefore, we include an artificial damping formulation that extracts kinetic energy from the fluid by comparing its solution to a solution that uses lower stiffness.

Our damping formulation only modifies the final velocity of the particle by removing kinetic energy, and it is designed to have a minimal impact. Damping is only applied when the animation is coming close to a stop, attempting to remove extra kinetic energy introduced due to numerical errors.

We begin by computing an alternative position $\mathbf{x}_i^*$ for each particle using a larger compliance parameter α^* (we use $\alpha^* = 1/1000$ in our tests). In practice, we use a single iteration to calculate $\mathbf{x}_i^*$ during the final iteration of the solver, starting with the positions $\mathbf{x}$ at the beginning of the last iteration. In this way, the computation of $\mathbf{x}_i^*$ involves minimal overheads.

At the end of the last iteration, we apply damping only if the two positions $\mathbf{x}_i$ and $\mathbf{x}_i^*$ are closer than a user-specified multiple β of the kernel radius r (we use $\beta = 60$ in our tests). This implies that a large amount of momentum is injected through high stiffness. In that case, we calculate two velocities for each particle using their final and alternative positions, such that

$$\mathbf{v}_i = \frac{\mathbf{x}_i - \mathbf{x}_i^t}{h} \qquad \text{and} \qquad \mathbf{v}_i^* = \frac{\mathbf{x}_i^* - \mathbf{x}_i^t}{h} \ . \tag{16}$$

Then, we compare the kinetic energy of the particle K_i to the alternative kinetic energy K_i^* with this alternative velocity. If $K_i \leq K_i^*$, we do not apply any damping. Otherwise, we bring the kinetic energy of the particle closer to the alternative. Thus, the final velocity is computed using

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i \begin{cases} 1, & \text{if } \|\mathbf{x}_i - \mathbf{x}_i^*\| \geq \beta r \\ 1, & \text{if } \|\mathbf{v}_i^*\| \geq \|\mathbf{v}_i\| \\ \sqrt{1 - d\,\frac{\|\mathbf{v}_i\|^2 - \|\mathbf{v}_i^*\|^2}{\|\mathbf{v}_i\|^2}}, & \text{otherwise,} \end{cases} \tag{17}$$

where the damping weight d is computed using

$$d = 1 - \frac{\|\mathbf{x}_i^* - \mathbf{x}_i\|}{\beta r} \ . \tag{18}$$

Algorithm 1 *IPBF Algorithm*

Input: $\mathbf{x}^t$: position at the beginning of the time step.
$\mathbf{v}^t$: velocity at the beginning of the time step.
$\mathbf{a}^\star$: non-pressure forces at the start of the time step.
Output: $\mathbf{x}^{t+1}, \mathbf{v}^{t+1}$, position and velocity at the end of the time step for every particle

$\mathbf{y} \leftarrow \mathbf{x}^t + h\mathbf{v}^t + h^2\mathbf{a}^\star$
$\mathbf{x}^{\text{guess}} \leftarrow \mathbf{y}$
$l = 0$
while $l < MAX_{\text{iterations}}$ **do**
  Update all density and density gradients based on $\mathbf{x}^{\text{guess}}$
  for all particles i **do**
    Calculate $\mathbf{f}_i$ using Equation 10
    Calculate $\mathbf{H}_i$ using Equation 11
    $\Delta \mathbf{x}_i \leftarrow \mathbf{H}_i^{-1}\mathbf{f}_i$
    $\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i^{\text{guess}} + \Delta \mathbf{x}_i/2$
  end for
  $\mathbf{x}^{\text{guess}} \leftarrow \mathbf{x}^{\text{new}}$
  $l = l + 1$
end while
$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^{\text{guess}}$
Set $\mathbf{v}_i^{t+1}$ using Equation 17

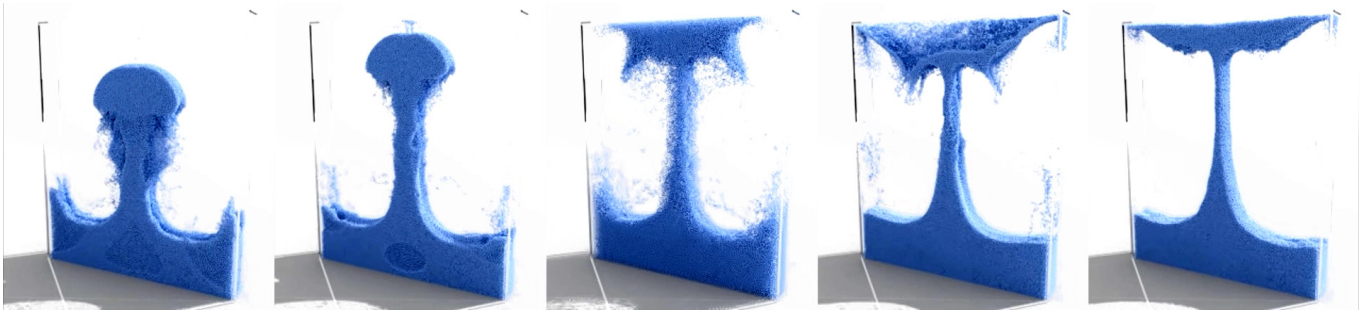


Fig. 2. *Double Dam Break example using 4 iterations per time step, run with PBF, SISPH, IISPH, DFSPH (2+2 iterations), and our method. All methods in this figure run with a 60 ms compute budget, achieved using $h = 1/240$ seconds.*

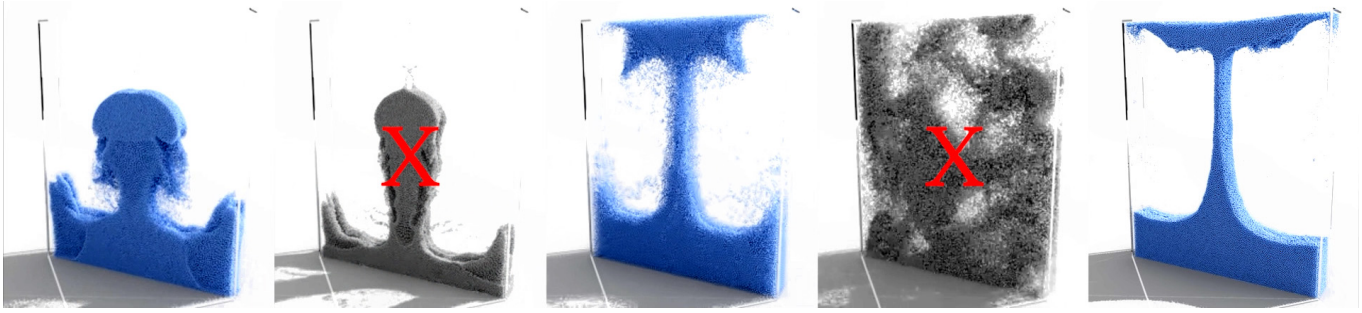


Fig. 3. *Double Dam Break example using 2 iterations per time step, except for DFSPH, which uses an additional divergence iteration (1+2 iterations). In this case, IISPH and DFSPH fail to maintain numerical stability.*

4 Results

We have implemented our method (see Algorithm 1) using CUDA on an NVIDIA RTX 4090. Unless otherwise specified, we use $\alpha = 0$ and our damping. Particle size is set to a diameter of 0.5 with a kernel size of 1. We compare our IPBF method against PBF [Macklin and Müller 2013], IISPH [Ihmsen et al. 2014], SISPH [He et al. 2025], and DFSPH [Bender and Koschier 2015]. All methods have a similar computation cost per iteration, but the computation cost can increase when a method fails to maintain the fluid density and particles begin to have more neighbors. As such, we opt to simply use the same iteration count when comparing different methods. One exception is DFSPH, which requires at least 1 divergence iteration and 2 density iterations (denoted as 1+2 when reporting iteration counts).

Figure 2 and Figure 3 show results from our Double Dam Break example, where two columns of fluid collide in the middle and form a thin jet. All methods produce comparable results when they are stable with 4 iterations (2+2 for DFSPH), as shown in Figure 2. In this case, DFSPH exhibits slight numerical instability but excellent density preservation, SISPH has slight extra energy injection due to density fluctuations, IISPH dissipates energy due to extra compression, and PBF dissipates the most energy, as can be seen from the height of the jet in comparison to the others.

Measuring the density error in Figure 2 at the start of every frame further highlights the results from stable animation in all methods, as plotted in Figure 4. As expected from the animations in our supplemental video, the graph shows DFSPH having excellent density preservation when stable, even surpassing our method, while SISPH and IISPH have higher density error and fluctuations. PBF has the largest density error and fluctuations, as shown by larger shifts in density error even as the animation is coming to rest.

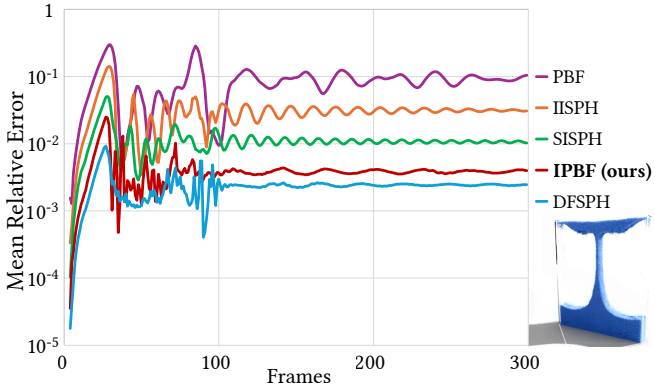


Fig. 4. *The mean relative error of density with different methods for the Double Dam Break examples in Figure 2.*

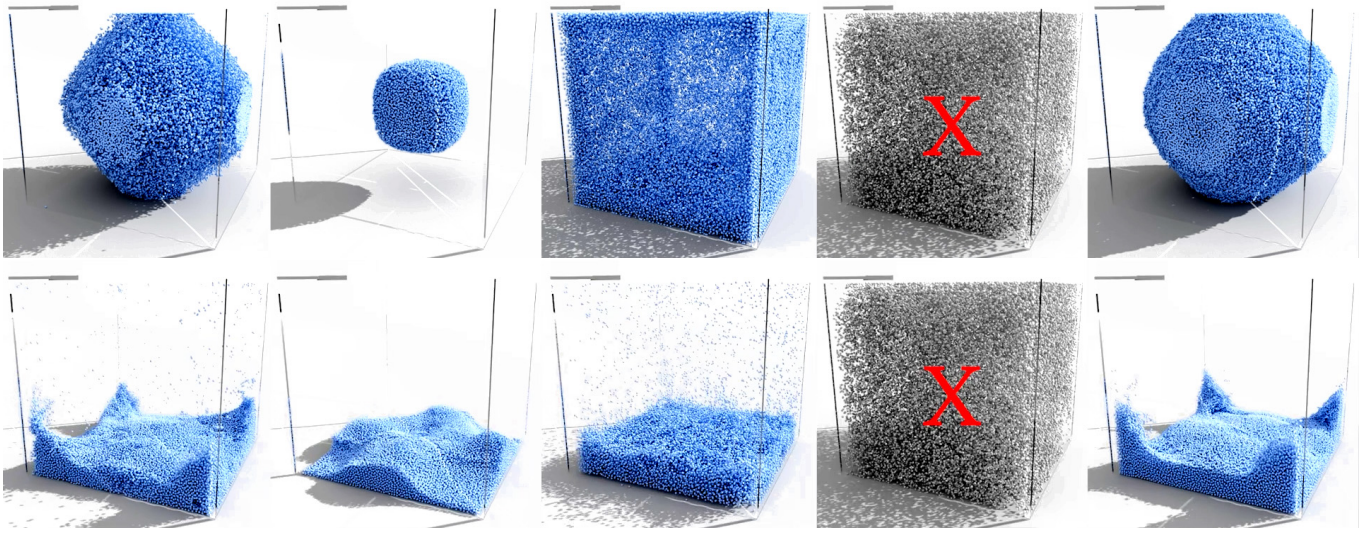


Fig. 5. *A compression stability test that begins with a compressed ball of fluid at the center with 7 times the rest density. Our IPBF method recovers to a stable state while DFSPH explodes, IISPH does not recover from high density error, and both PBF and SISPH remain with unstable particles. All examples were run with $h = 1/120$ seconds and with 3 iterations (3+3 for DFSPH). The top row shows an earlier frame of the animations.*

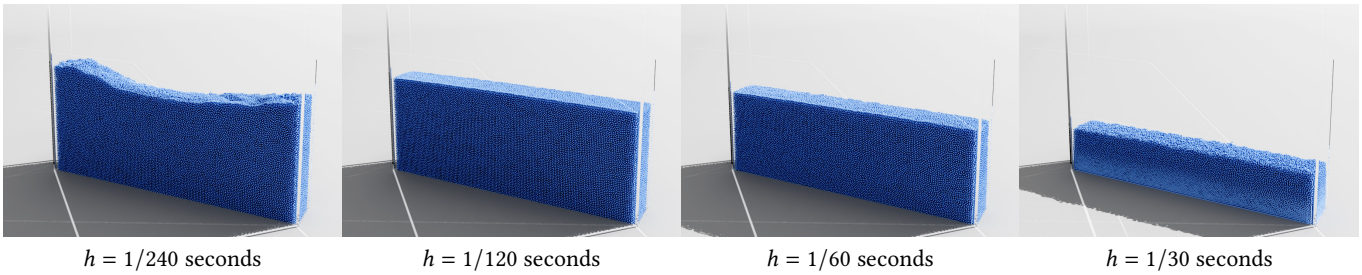


Fig. 6. *Our method produces stable results for a double dam break even when increasing the time step size up to $h = 1/30$ seconds and limiting iteration counts to 2 iterations per time step. The error due to a large time step size and limited iterations leads to excessive compression.*

Figure 3 shows the results with 2 iterations per time step (1+2 for DFSPH). In this case, DFSPH and IISPH fail due to numerical explosion, but all position-based methods produce a similar result as before.

Stability. Figure 5 shows a stress test for stability. In this example, we initialize a sphere of fluid with an initial state of 7× the rest density and simulate under the same time step size and equal iteration count, except for DFSPH, which is given +3 iterations for its divergence solve. Still, unlike other methods, DFSPH is unable to recover from the large deviation in density, and experiences numerical explosion soon after the simulation begins. While IISPH maintains stability, it does so at the cost of significant volume loss, as it can not recover to the rest density. Similarly to ours, PBF and SISPH are able to recover without significant volume loss, but fail to organize a coherent flow with many individual fluid particles flying around. In contrast, our method recovers from this significant density deviation without volume loss or incoherent flow. This is notable as volume loss is often a trade-off with stability. While a stiffer system can result in less compression, it is typically also less numerically stable. Our method is able to achieve both. In fact, our method exhibits unconditional stability in all our tests.

We demonstrate this behavior of our method in Figure 6, where we gradually increase the time step size h while only using 2 iterations per time step, using the same Double Dam Break example in Figure 2. Our method shows little change, raising the h from $1/240$ to $1/60$ seconds. Even at a time step size of $1/30$ seconds, our method remains stable at the cost of noticeable compression. This test shows our method operating with a substantial residual, accumulated over many frames, and unable to correct the density error with only 2 iterations for a large time step size, but still remaining stable. In general, our method trades incompressibility for stability when necessary and thus remains stable even at extreme time step sizes.

Incompressibility. When such a tradeoff is not needed, our method also naturally exhibits superior incompressibility. We demonstrate this in Figure 7 with a large static column of water simulated in 2D.

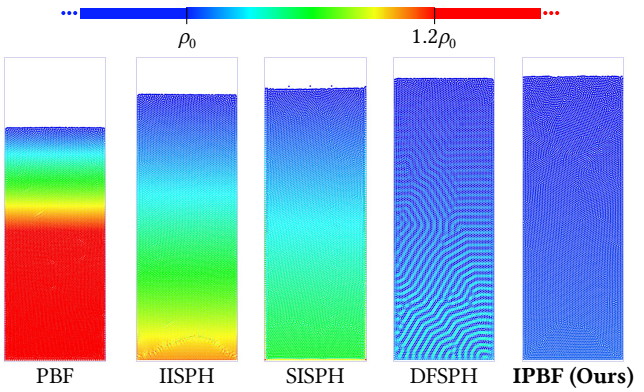


Fig. 7. *By dropping a column of water in 2D, we measured how much compression each method exhibited with 8 iterations (4+4 for DFSPH) and with $\alpha = 10^{-6}$ ($\mu = 10^6$ for SISPH). At a converged state, IPBF visually shows the least amount of compression.*

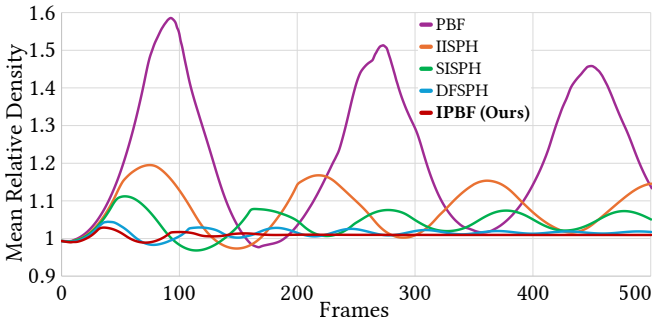


Fig. 8. *This graph shows the average density of particles in the 2D compression test over the course of the animation shown in Figure 7 and in our supplemental video. IPBF shows very rapid convergence and minimal deviation from the rest density (of 1) as the animation plays.*

Under equal computation time, our method has a similar compression as DFSPH, while PBF, SISPH, and IISPH compress substantially. In Figure 8, we show the average density over the first 500 frames for all methods. Notice that our IPBF method quickly reaches a steady state, while other methods exhibit more oscillations, in addition to higher compression.

The same behavior can be observed in 3D. In Figure 11, we drop a block of water onto a puddle. This creates a large amount of compression error upon contact. Figure 12 shows the state of the simulation immediately after this contact, comparing different solvers. Here, in an attempt to enforce incompressibility, DFSPH is unable to maintain stability and explodes upon contact. PBF and IISPH maintained stability, but exhibited excessive compression, which also leads to a substantial increase in computation cost, because, with high compression, each particle interacts with many more particles within a kernel radius around it. In this example, SISPH also remains stable, but shows some larger energy injection due to a large error residual.

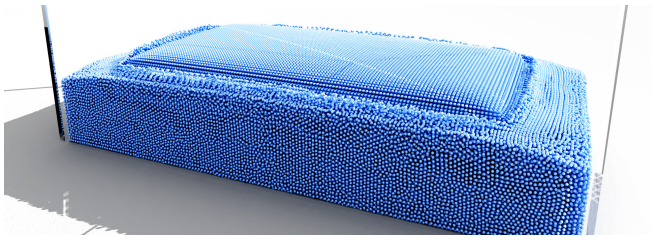


Fig. 9. *The frame used to run a 3D convergence test. It is the moment a large block of water falls on a puddle of water below.*

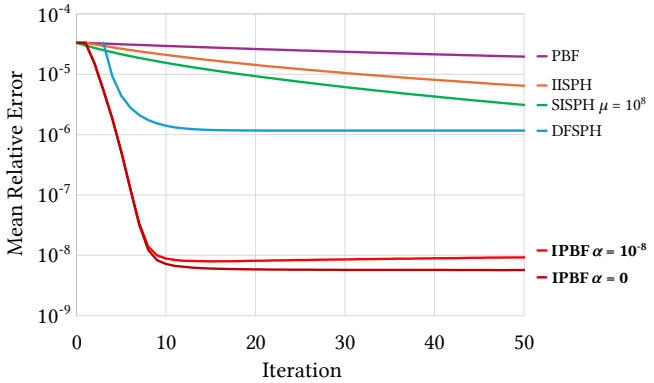


Fig. 10. *Our method converges to a lower density error with fewer iterations when starting at the same initial state, shown in Figure 9.*

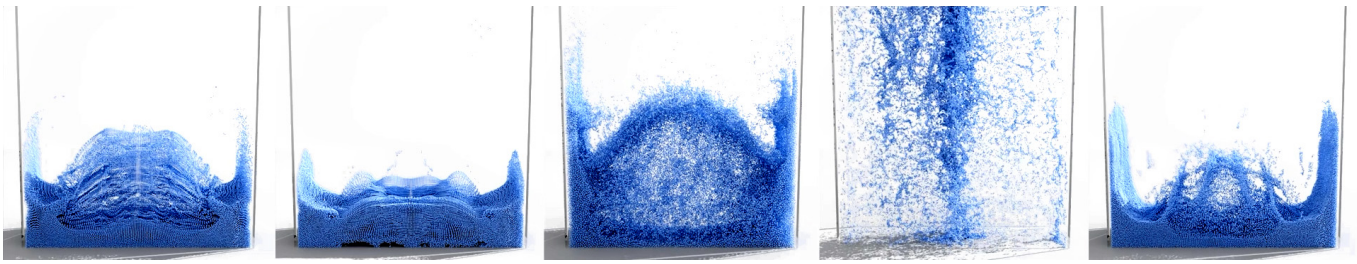
We present a convergence test in Figure 10, showing the mean relative error of fluid density for one time step with increasing iteration count. All methods in this test begin with the exact simulation state shown in Figure 9, which corresponds to a frame of the animation in Figure 11 when the block of water collides with the puddle. Notice in Figure 10 that our IPBF method quickly converges to a low mean relative error with less than 10 iterations in this test. Using a non-zero α parameter, a slightly larger density error is observed to account for the momentum term in the variational energy. DFSPH includes 2 additional iterations for its velocity divergence solve, which lags its convergence; otherwise, it also converges quickly, but to a larger error. The other methods have much slower convergence. Although not shown in this graph, SISPH continues to reduce the error with further iterations below the error level of DFSPH, but retains a higher error than our IPBF method even after 1000 iterations.

Speed. Since our method has a similar computational cost per iteration, yet allows larger time step sizes to produce stable simulations, our method can be significantly faster than prior work.

Table 1 shows the performance results with settings that produce a similar density error for all methods. This is achieved by tuning the iteration count and time step size for each method. Notice that for both examples presented in this paper, our IPBF method has significantly lower computation time per frame, as compared to the other methods.

Figure 1 shows a large simulation example where we pop a balloon filled with particles over a sphere and a puddle below it, amounting

Fig. 11. *The Block Flop example with 250K particles simulated using our IPBF method. A block of fluid drops onto a puddle. The resulting simulation results in a period of high density error, but with 2 iterations and a time step size of $h = 1/240$, we maintain stability with incompressibility, running at an average of 60 ms per frame.*



PBF IISPH SISPH DFSPH **IPBF (ours)**

Fig. 12. *A frame of the Block Flop animation from Figure 11 soon after the block hits the puddle, simulated using different methods. All methods have a similar per-frame computation time of 60 ms, using $h = 1/240$ seconds and all methods with 2 iterations, except for DFSPH, which had 2 additional divergence iterations that incur extra computation cost. Nonetheless, DFSPH is entirely unstable in this example. SISPH injects energy due to density error.*

Table 1. *Performance comparison with similar average density error, achieved using different iteration counts per step and different step sizes per frame for different methods. The total computation times per frame are measured on an NVIDIA RTX 4090 GPU.*

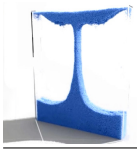
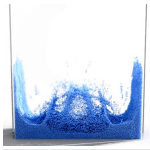
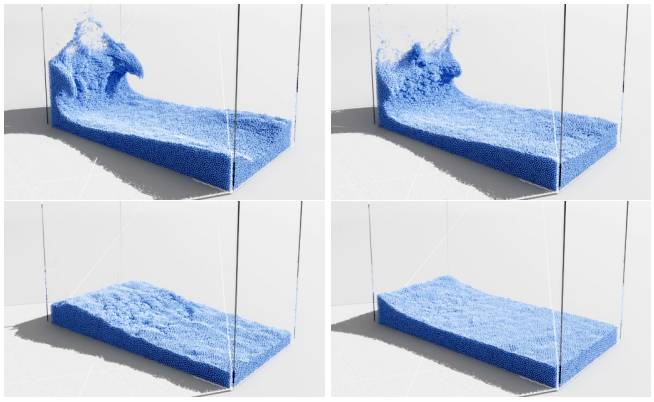
Figure 2	Solver	Error	Iter.	Step Size	Time
	**IPBF (Ours)**	9.2e-4	2	1 / 480 s	70 ms
	DFSPH	1.2e-3	4+4	1 / 240 s	111 ms
	SISPH	2.5e-3	2	1 / 480 s	117 ms
	IISPH	5.5e-3	5	1 / 480 s	250 ms
	PBF	5.6e-3	4	1 / 960 s	250 ms

Figure 12	Solver	Error	Iter.	Step Size	Time
	**IPBF (Ours)**	4.6e-4	1	1 / 480 s	50 ms
	DFSPH	5.8e-4	4+4	1 / 240 s	110 ms
	SISPH	1.3e-3	2	1 / 480 s	100 ms
	IISPH	2.6e-3	5	1 / 480 s	250 ms
	PBF	5.5e-3	2	1 / 960 s	142 ms



No damping Damped

Fig. 13. *We compare a small dam break with a solve using $\alpha = 0$ and the same solve using our artificial damping formulation. Without the damping, the fluid surface is noisier and does not come to rest, whereas it settles to rest after some time with our damping.*

to 900K particles in the simulation. On an NVIDIA RTX 4090, we are able to maintain stability in this example using 159 milliseconds per frame.

Damping. Our artificial damping formulation is useful in removing excess energy injected from residual error in the stiff system. In Figure 13, we compare our damping with an undamped solve, seeing improved energy dissipation without sacrificing the dynamic energy of splashes in the fluid. As can be seen in our supplemental video, our damping formulation has minimal effect on the overall fluid motion, and it only dampens the motion as the particles come close to rest.

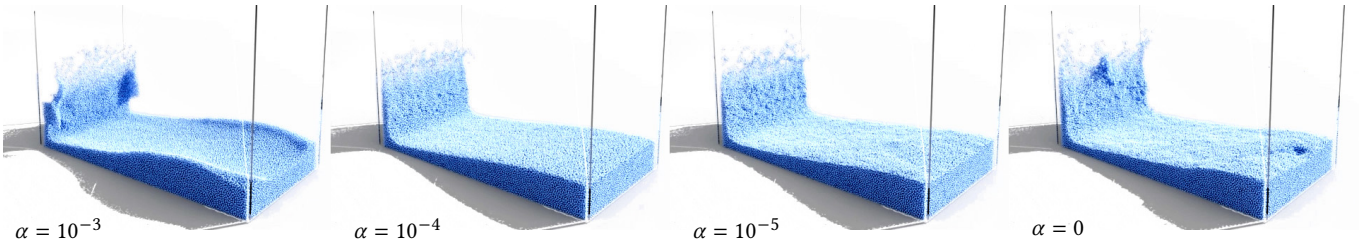
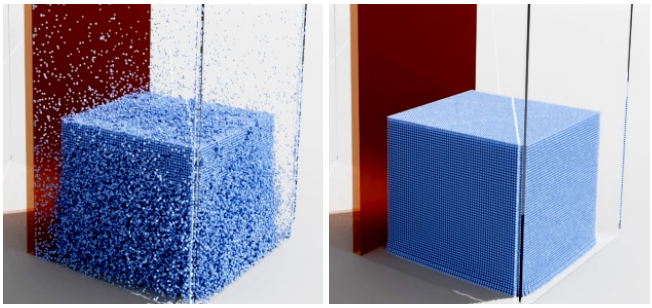


Fig. 14. *Simulation of a dam break using our IPBF method with different compliance parameters α for simulating different fluid compressibility.*



No Hessian approximation With Hessian approximation

Fig. 15. *Simulation of a dam break that fails to maintain numerical stability without our Hessian approximation.*

Compliance. Our IPBF method can be used with different compliance parameters α for simulating fluids with different compressibility behavior. This is demonstrated in Figure 14.

Hessian Approximation. Our Hessian approximation is critical for achieving numerical stability. As can be seen in Figure 15, without our Hessian approximations, simulations often quickly run into instability, even with relatively simple cases.

5 Conclusion

We have introduced an implicit position-based fluid method for solving SPH fluids with superior stability at large time steps while still maintaining low density error. We show that our method is able to outperform prior work while showing comparable quality of simulation and remaining stable even under extreme time steps.

Limitations & Future Work. Although we maintain stability, our method may exhibit artifacts due to energy injection from the high stiffnesses present in incompressible fluids. This is largely due to residual errors from zero α values, preventing our method from providing a convergence guarantee. Future work can address this by presenting alternative damping formulations that do not require user-specified tuning for simulations.

References

Sheldon Andrews, Marek Teichmann, and Paul G. Kry. 2017. Geometric Stiffness for Real-time Constrained Multibody Dynamics. *Computer Graphics Forum* 36, 2 (2017), 235–246. doi:10.1111/cgf.13122 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13122

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 209–217.

Jan Bender and Dan Koschier. 2015. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '15)*. Association for Computing Machinery, New York, NY, USA, 147–155. doi:10.1145/2786784.2786796

Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. 2024. Vertex Block Descent. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2024)* 43, 4, Article 116 (0766 2024), 16 pages. doi:10.1145/3658179

T. F. Gast and C. Schroeder. 2015. Optimization integrator for large time steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 31–40.

R. A. Gingold and J. J. Monaghan. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 181, 3 (12 1977), 375–389. doi:10.1093/mnras/181.3.375 arXiv:https://academic.oup.com/mnras/article-pdf/181/3/375/3104055/mnras181-0375.pdf

Xiaowei He, Ning Liu, Sheng Li, Hongan Wang, and Guoping Wang. 2012. Local Poisson SPH For Viscous Incompressible Fluids. *Computer Graphics Forum* 31, 6 (2012), 1948–1958. doi:10.1111/j.1467-8659.2012.03074.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03074.x

Xiaowei He, Shusen Liu, Yuzhong Guo, Jian Shi, and Ying Qiao. 2025. A Semi-Implicit SPH Method for Compressible and Incompressible Flows with Improved Convergence. *Computer Graphics Forum* (2025). doi:10.1111/cgf.70043

Xiaowei He, Huamin Wang, Guoping Wang, Hongan Wang, and Enhua Wu. 2020. A Variational Staggered Particle Framework for Incompressible Free-Surface Flows. arXiv:2001.09421 [cs.GR] https://arxiv.org/abs/2001.09421

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2014. Implicit Incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (March 2014), 426–435. doi:10.1109/TVCG.2013.105

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2019. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. doi:10.2312/egt.20191035

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2022. A Survey on SPH Methods in Computer Graphics. *Computer Graphics Forum* 41, 2 (2022), 737–760. doi:10.1111/cgf.14508 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14508

Shusen Liu, Xiaowei He, Yuzhong Guo, Yue Chang, and Wencheng Wang. 2024. A Dual-Particle Approach for Incompressible SPH Fluids. *ACM Trans. Graph.* 43, 3, Article 28 (April 2024), 18 pages. doi:10.1145/3649888

L. B. Lucy. 1977. A numerical approach to the testing of the fission hypothesis. *aj* 82 (Dec. 1977), 1013–1024. doi:10.1086/112164

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Trans. Graph.* 32, 4, Article 104 (July 2013), 12 pages. doi:10.1145/2461912.2461984

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. ACM, New York, NY, USA, 49–54. doi:10.1145/2994258.2994272

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 154–159.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J. Vis. Comun. Image Represent.* 18, 2 (April 2007), 109–118. doi:10.1016/j.jvcir.2007.01.005

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Trans. Graph.* 34, 4, Article 114 (July 2015), 10 pages. doi:10.1145/2766925

B. Solenthaler and R. Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 40, 6 pages. doi:10.1145/1576246.1531346