EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

A Semi-Implicit SPH Method for Compressible and Incompressible Flows with Improved Convergence

Xiaowei He[†1], Shusen Liu[1], Yuzhong Guo[1], Jian Shi[2] and Ying Qiao[1]

[1]Institute of Software, Chinese Academy of Sciences, China
[2]Institute of Automation, Chinese Academy of Sciences, China

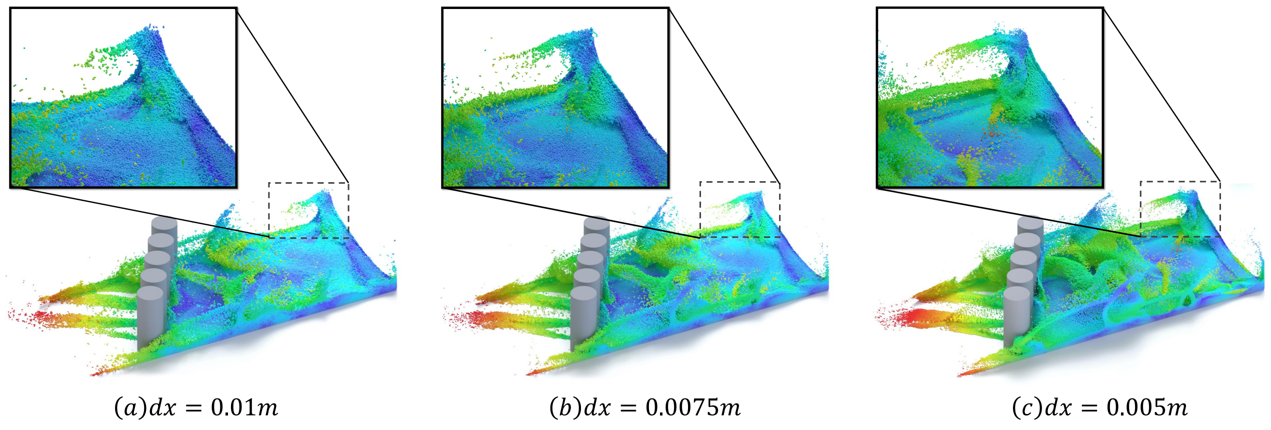


Figure 1: *Our method generates consistent behaviors in incompressible fluid simulations across varying particle sizes. The particle resolutions are set to $dx = 0.01m$, $0.0075m$, and $0.005m$ from left to right, respectively. Particle velocities are color-coded.*

Abstract

In simulating fluids using position-based dynamics, the accuracy and robustness depend on numerous numerical parameters, including the time step size, iteration count, and particle size, among others. This complexity can lead to unpredictable control of simulation behaviors. In this paper, we first reformulate the problem of enforcing fluid compressibility/incompressibility into an nonlinear optimization problem, and then introduce a semi-implicit successive substitution method (SISSM) to solve the nonlinear optimization problem by adjusting particle positions in parallel. In contrast to calculating an intermediate variable, such as pressure, to enforce fluid incompressibility within the position-based dynamics (PBD) framework, the proposed semi-implicit approach eliminates the necessity of such calculations. Instead, it directly employs successive substitution of particle positions to correct density errors. This method exhibits reduced dependency to numerical parameters, such as particle size and time step variations, and improves consistency and stability in simulating fluids that range from highly compressible to nearly incompressible. We validates the effectiveness of applying a variety of different techniques in accelerating the convergence rate.

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

1. Introduction

In real-time simulation of particle-based fluids, position based approaches are commonly used to enforce fluid incompressibility [KBST22]. The basic idea is to formulate fluid incompressibility as a set of non-linear positional constraints that solve constant density for particles. During the time integration procedure, an intermediate variable resembling the role of pressure [SP09; HLL*12; MM13; BK16] is calculated for each particle to iteratively correct the predicted positions until all constraints are satisfied un-

[†] corresponding: xiaowei@iscas.ac.cn

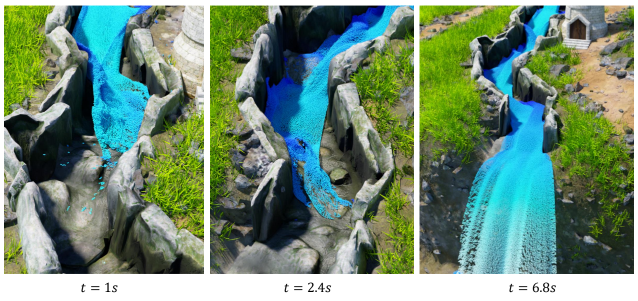


Figure 2: *Waterfall. This scenario consists of a maximum of 1.03 million particles, and the particle size is* $0.005m$*.*

der a user-defined threshold or the maximum iteration number is reached.

The position based dynamics is popular due to its high efficiency and robustness in handling large time steps. However, PBD also has several limitations. One of the most notorious problems is that the simulation behavior is dependent on both the iteration number and time step size [MMC16]. For example, as the iteration number increases, the positional constraints used to enforce constant density can become arbitrary stiff. This artifact makes it difficult to control the behaviors of simulated objects because the final behaviors are determined not only by their physical material parameters but also a bunch of numerical parameters. This coupling of parameters presents challenges not only for creating scenes in which soft bodies interact with nearly rigid bodies [MMC16], but also for multiphase flows where compressible fluids interact with incompressible fluids, as illustrated in Figure 11(top). While it is feasible to adjust the iteration count to approximate fluid compressibility, the nonlinear effects of iteration count on material stiffness can complicate the intuitive control of fluid compressibility.

To overcome the limitations of conventional PBD methods, we introduce a semi-implicit SPH method tailored for both compressible and incompressible flows. In lieu of enforcing fluid incompressibility via density constraints, we propose to reformulate fluid compressibility and incompressibility as a variational representation of particle positions. This representation amalgamates fluid momentum potential and bulk energy, drawing inspiration from the principles elucidated in projective dynamics [BML*14]. The resolution of fluid compressibility or incompressibility is thus transformed into a nonlinear optimization problem. To minimize this objective, we propose a semi-implicit SPH (SISPH) method inspired by Lu et al. [LHG*23]. In this approach, a fixed-point iteration strategy, based solely on the first-order derivative of the objective function, is used to iteratively and concurrently adjust particle positions to enforce the desired fluid compressibility. Unlike traditional PBD method, the proposed semi-implicit approach presents a unified solver to model fluids ranging from highly compressible to nearly incompressible, exhibits reduced dependency to parameter adjustments across different configurations of fluid simulations, regardless of particle sizes and time steps. As depicted in results, our method produces highly accurate simulation results for both compressible and incompressible fluids.

2. Related Works

The meshless Lagrangian particle methods employed for addressing fluid incompressibility are categorizable into two principal approaches: the constant-density approach and the divergence-free approach. The following section presents a discussion of these two approaches mainly within the computer graphics community.

The constant-density approach focuses on maintaining a consistent particle density throughout the simulation. This is achieved by adjusting the particle spacing or using a kernel function that adapts to ensure a constant number of neighboring particles within a specified radius. Desbrun and Gascuel [DG96] first introduced smoothed particle hydrodynamics into computer graphics for animating highly deformable bodies. Subsequently, several non-iterative equation of state (EOS) solvers were introduced, differing only slightly in their utilization of the EOS. As an illustration, Müller et al. [MCG03] employ a modified version of the method proposed by Desbrun and Gascuel [DG96], whereas Becker and Teschner [BT07] opt for the utilization of Tait's equation to more effectively enforce fluid incompressibility. The major limitation with the non-iterative EOS solvers is that the time step is strictly restricted by the Courant-Friedrichs-Lewy(CFL) condition. In order to mitigate the constraints imposed by timestep size, Solenthaler and Pajarola [SP09] introduced the inaugural iterative EOS Solver, referred to as Predictive-Corrective ISPH (PCISPH), designed to enforce fluid incompressibility. Subsequently, He et al. [HLL*12] introduced an integral form derived from the pressure Poisson equation, aiming to enhance the convergence rate of the predictive-corrective scheme. Bodin et al. [BLS11] advocated the application of holonomic kinematic constraints on density as a means to model incompressible fluids and achieve enhanced stability in contrast to the conventional SPH method. Macklin and Müller [MM13] introduced a method to address particle incompressibility within the context of the Position Based Dynamics (PBD) framework. However, it's important to note that this approach inherits several numerical limitations from PBD, as discussed by Macklin et al. [MMC16]. Weiler et al. [WKB16] proposed an alternative approach for addressing particle incompressibility using the projective dynamics framework. However, their method inherits several key characteristics from Projective Dynamics, such as the inability of the global solver to handle nonlinear constraints, which results in fluids being slightly compressible. Finally, a constant density field can be enforced by solving a pressure Poisson equation. Ihmsen et al. [ICS*14; BGI*18] proposed to combine a symmetric SPH pressure force and an SPH discretization of the continuity equation to improve the convergence rate. Takahashi et al. [TDNL18] proposed a hybrid SPH solver with a new interface handling method to address issues in traditional projection-based solvers.

The divergence-free approach directly enforces the incompressibility condition by focusing on the divergence of the velocity field. This is accomplished by solving a pressure Poisson equation to ensure a zero-divergence condition. In contrast to constant-density approaches, implementing a divergence-free approach is typically more complex and necessitates meticulous treatment of boundary conditions and numerical stability. Because directly enforcing the divergence-free condition on particles is challenging, the early work by Raveendran [RWT11] introduced a hybrid method that combines a local SPH density solver with an Eulerian velocity divergence solver to enhance the performance of fluid simulations. Returning to the fully Lagrangian framework, in order to tackle the zero-energy mode problem, He et al. [HLW*12] introduced staggered particles alongside the original ones, and devised a new approximate projection method capable of enforcing divergence-free behavior for particles both within and near boundaries. Yang et al. [YHW*16] further improved the accuracy of the projection method by incorporating a semi-analytical scheme near the free-surface boundary. Divergence-free approaches can be computationally efficient as they directly enforce incompressibility without the need for redistributing particles. However, they may have stability challenges in some scenarios. Bender and Koschier [BK16] advocated to enforce incompressibility both on position level and velocity level, and proposed to combine two pressure solvers to enforce both low volume compression and a divergence-free velocity field, therefore achieving faster and more stable simulation of incompressible fluids. Kang and Sagong [KS14] also introduced a method for simulating an incompressible fluid that simultaneously adheres to the divergence-free and constant-density conditions. Alternatively, one may opt to concurrently address the divergence-free and constant-density conditions through a unified solution. Losasso et al. [LTKF08] employ a source term that incorporates both the divergence-free and constant-density conditions, facilitating the enforcement of incompressibility and the control of particle number density within a unified Poisson solving framework. Nevertheless, Cornelis et al. [CBG*19] contend that integrating both of these conditions into the source term introduces numerical artifacts. As a remedy, they suggest solving two PPEs despite incurring a twofold increase in computational costs.

Enforcing fluid incompressibility can be reinterpreted as a variational problem. Batty et al. [BBB07] reformulate the pressure projection step as a minimization of kinetic energy, thereby unifying the enforcement of fluid incompressibility and the resolution of solid-fluid coupling within a single framework. Larionov et al.[LBB17] further integrate viscosity and pressure to formulate the Stokes problem as an implicit variational problem, allowing grid-based fluid simulators to effectively reproduce complex viscous liquid motions such as coiling and buckling. In simulation of Lagrangian solid-fluid coupling, Xie et al. [XLYJ23] propose a variational formulation to approximate fluid incompressibility. Our method is inspired by this variational interpretation of fluid incompressibility; however, rather than formulating the variational approach in terms of particle velocities, we aim to solve the variational problem using a purely position-based dynamics (PBD) framework to enhance stability.

3. Fundamentals

For completeness, this section presents a brief description of all mathematical theories required to derive the semi-implicit SPH method.

3.1. Fixed-point iteration

Fixed-point iteration is a commonly employed technique in numerical analysis. Suppose f is a function defined on a real number x, the fixed-point iteration is conventionally formulated as follows:

$$x_{k+1} = f(x_k), \quad k = 0, 1, 2, ... \tag{1}$$

where k is the iteration number. As the iteration number k increases, the sequence of real numbers x_0, x_1, x_2, ... is anticipated to converge towards a fixed point denoted as x_{fix}, where $x_{\text{fix}} = f(x_{\text{fix}})$. Nevertheless, it is important to note that not all fixed-point iteration sequences converge to x_{fix}. The Banach fixed-point theorem provides a sufficient condition for the convergence of fixed-point iteration sequences [Lat14]. To ensure both existence and uniqueness, this theorem necessitates that the sequence of real numbers resides within an open neighborhood of the fixed point x_{fix} and that $\|f'(x_k)\| < 1$.

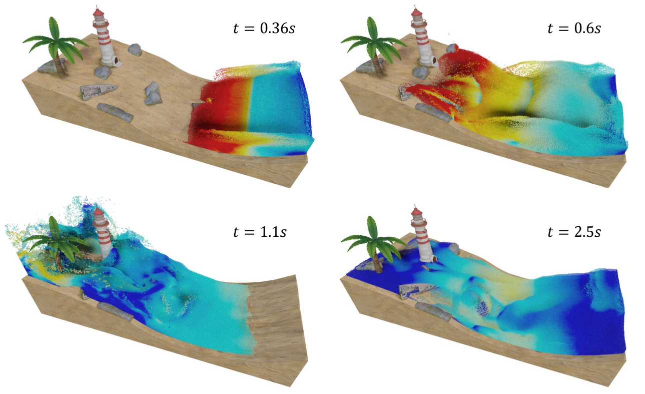


Figure 3: *Seaside. This example demonstrates a stable flood simulation of seawater over the seaside, where the seawater is modeled with 1.6 million particles and the seaside is represented by 2.5k boundary triangles.*

One of the most appealing aspects of employing fixed-point iteration is its inherent simplicity. It's noteworthy that this method entails solely the repeated application of the function to an initial estimate until the convergence condition is met. Furthermore, it obviates the need for employing mathematical tools like matrix inversion or gradient calculations. Consequently, fixed-point iteration emerges as an ideal candidate for GPU-based parallelization. Nonetheless, the Banach fixed-point theorem suggests that this method may fail to converge to a solution, particularly when the initial guess is distant from the fixed point. When dealing with a non-linear optimization problem, it necessitates a thorough analysis of both the problem and the initial estimate before opting for fixed-point iteration.

3.2. Semi-implicit successive substitution method

Lu et al. [LHG*23] initially introduced the semi-implicit successive substitution method for solving nonlinear problems with a general formula expressed as follows:

$$x = f(x, x')(x' - x) + c, \tag{2}$$

where both x and x' are regarded as real numbers here for the sake of simplicity, x' can be viewed as a point in the neighborhood of x, and c represents a constant. It is important to note that when $f(x, x')$

represents an arbitrarily chosen nonlinear function, it is possible that the sufficient condition mandated by the Banach fixed-point theorem may not be met. Hence, if we directly apply fixed-point iteration to solve Equation 2, the sequence of x_k is not assured to converge towards a fixed point. To tackle this challenge, Lu et al. [LHG*23] suggest employing a semi-implicit strategy for linearizing Equation 2 at (x_k, x'_k), as follows:

$$x_{k+1} = f^+(x_k, x'_k)(x'_k - x_{k+1}) + f^-(x_k, x'_k)(x'_k - x_k) + c, \quad (3)$$

where $f(x_k, x'_k) = f^+(x_k, x'_k) + f^-(x_k, x'_k)$. To elaborate further, $f^+(x_k, x'_k)$ denotes a fraction with a consistently positive value, whereas $f^-(x_k, x'_k)$ signifies a fraction with a consistently negative value. Following the semi-implicit linear approximation, we proceed with a Jacobi iterative step to refine the initial estimate as follows:

$$x_{k+1} = \frac{c + f^+(x_k, x'_k)x'_k + f^-(x_k, x'_k)(x'_k - x_k)}{1 + f^+(x_k, x'_k)}. \quad (4)$$

This procedure should be iterated until an adequately precise threshold is attained. It is worth noting that the entire procedure within SISSM bears a resemblance to fixed-point iteration, as neither method necessitates the use of intricate mathematical tools. However, SISSM's convergence is assured when we meticulously split $f(x, x')$ into its positive and negative components.

4. A Semi-implicit SPH Method

Our approach is formulated by initially rephrasing the concept of fluid incompressibility in a variational manner. To ensure fluid incompressibility, the position-based method typically seeks to resolve a constraint for each individual particle, as described in [MM13]

$$C_i = \lambda_i - 1 = 0, \quad \lambda_i = \frac{\rho_i}{\rho_0}, \quad (5)$$

where ρ_0 is the rest density and ρ_i is calculated with a particle approximation as [KBST19]

$$\rho_i = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, H). \quad (6)$$

In the simulation of incompressible fluids with a fixed particle size, it is often convenient to treat both the particle mass m_j and the smoothing length H as constants. Nevertheless, because the choice of the kernel function $W(\cdot)$ typically involves a nonlinear dependence on $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ to prevent significant volume loss, enforcing the constraint in Equation 5 is, in essence, tantamount to solving a nonlinear optimization problem, such as seeking a local minimum of C_i^2.

From an optimization perspective, the iterative density solver introduced within the position-based dynamics framework [MM13] can be seen as a variant of the nonlinear Jacobi algorithm. Nevertheless, as highlighted in Macklin's work [MMC16], a significant drawback of position-based dynamics is that the stiffness of constraints varies with both the iteration count and time step size, posing challenges in achieving predictable physical behaviors. Inspired by the principles of projective dynamics [KUJH21; LHG*23; BML*14], we expand the constant-density constraint and

Algorithm 1: A Semi-Implicit SPH (SISPH) Method

```
while $t < t_{stop}$ do
    for all particles $i$ do
        $\mathbf{v}_i^* \leftarrow \mathbf{v}_i^n + h \cdot \mathbf{f}_i$;
        $\mathbf{x}_i^* \leftarrow \mathbf{x}_i^n + h \cdot \mathbf{v}_i^*$;
    end
    for all particles $i$ do
        find neighboring particles;
    end
    for all particles $i$ do
        $\mathbf{x}_i^0 = \mathbf{x}_i^*$;
    end
    set $k = 0$ ;
    while $k < N \parallel \varepsilon^k < \eta$ do
        for all particles $i$ do
            compute $\rho_i$ using Eq. 6 based on $\mathbf{x}^k$;
            clamp $\rho_i$ using $\rho_i = max(\rho_i, \rho_0)$;
            compute $\mathbf{x}_i^{k+1}$ using Eq. 14;
            compute $\alpha_i^k$ using Eq. 20;
            $\mathbf{x}_i^{k+1} \leftarrow \mathbf{x}_i^k + \alpha_i^k \left(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right)$;
            $k = k + 1$;
        end
    end
    for all particles $i$ do
        set $\mathbf{x}_i^{n+1} = \mathbf{x}_i^N$;
        update velocity $\mathbf{v}_i^{n+1} = (\mathbf{x}_i^{n+1} - \mathbf{x}_i^n)/h$;
    end
end
```

formulate the objective function in the following variational manner:

$$\psi = \frac{1}{2h^2} m \left\|\mathbf{x} - \mathbf{x}^*\right\|^2 + \mu \sum_i B(\lambda_i), \quad (7)$$

where the first term defines the momentum potential while the second term defines the bulk energy potential. Within above definition, $\mathbf{x}$ denotes the column vector containing all particle positions, $\mathbf{x}^*$ represents the intermediate position as shown in Algorithm 1, h signifies the time step, $\|\cdot\|$ stands for the l^2-norm, and $B(\lambda_i)$ represents the bulk energy associated with particle i whose exact formulation will be given in Section 4.3. The optimization problem now involves balancing the momentum potential and the bulk energy potential, with the weighting determined by the constant μ. It is worth noting that when the momentum potential is omitted, the energy optimization problem is analogous to the constant-density constraint, provided that the bulk energy function $B(\lambda_i)$ remains convex and continuously differentiable.

4.1. Semi-implicit successive substitution

In accordance with optimization theory, when $\mathbf{x}$ serves as a local minimizer of a convex and continuously differentiable objective function ψ, we can establish the following relationship for each particle by differentiating ψ with respect to $\mathbf{x}_i$

$$\mathbf{x}_i = \mathbf{x}_i^* + \mu \frac{h^2}{\rho_0} \dot{B}(\lambda_i) \sum_j \frac{\partial W_{ij}}{\partial r_{ij}} \frac{\mathbf{x}_j - \mathbf{x}_i}{r_{ij}}, \tag{8}$$

where $\dot{B}$ represents the first-order derivative of B with respect to λ_i, W_{ij} is short for $W(\mathbf{x}_i - \mathbf{x}_j, H)$. Please note that when we interpret $\mathbf{x}_j - \mathbf{x}_i$ in Equation 8 as $x' - x$ in Equation 2, the remaining part of the second term excluding $\mathbf{x}_i - \mathbf{x}_j$ as $f(x, x')$, and $\mathbf{x}_i^*$ as c, Equation 8 now conforms to the format of Equation 2. Nevertheless, when taking the derivative of ψ with respect to $\mathbf{x}_i$ to obtain Equation 8, it is important to notice we have temporally neglected contributions from all neighboring particle j to make the following discussion concise. Later, we will provide additional details on how to ensure the conservation of momentum during position updates.

Having established that Equation 8 now conforms to the format of Equation 2, our next task is to investigate how to separate the second term of the RHS into two parts. Given that μ, h, ρ_0, and r_{ij} are consistently positive, the sign of the coefficient preceding $\mathbf{x}_i - \mathbf{x}_j$ is solely influenced by $\dot{B}(\lambda_i)$ and $\frac{\partial W}{\partial r}$. In the simulation of position-based incompressible fluids, as two particles approach each other, we anticipate the emergence of stronger repulsion forces to prevent particle clustering. Specifically, the ideal kernel function W employed for ensuring incompressibility should possess a non-vanishing gradient near its center, and its first derivative should exhibit a monotonic increase while vanishing at the boundary of the support domain, as suggested in prior work [HLW*12; GK16]. Consequently, we employ Desbrun's spiky kernel [MCG03]

$$W = \frac{15}{\pi H^6} \begin{cases} (H-r)^3 & 0 \le r < H \\ 0 & H < r \end{cases}. \tag{9}$$

It can be verified that its first-order derivative is negative for all r. Now, our task is to examine the sign of each term in $\dot{B}(\lambda_i)$ exclusively. Given that ρ_i always has a value greater than 0, it follows that $\lambda_i > 0$ should hold indefinitely. Hence, a general approach to simplify the decomposition of $\dot{B}(\lambda_i)$ is to express it as a polynomial function of λ_i, such as

$$\dot{B}(\lambda_i) = \sum_{n=-\infty}^{\infty} b_n \lambda_i^n, \tag{10}$$

where b_n represents the coefficient of λ_i^n. Subsequently, we can readily perform the split as $\dot{B}(\lambda_i) = \dot{B}^-(\lambda_i) + \dot{B}^+(\lambda_i)$, where the positive component $\dot{B}^+(\lambda_i)$ and the negative component $\dot{B}^-(\lambda_i)$ are represented as

$$\begin{aligned} \dot{B}^+(\lambda_i) &= \sum_n b_n \lambda_i^n, \quad b_n > 0 \\ \dot{B}^-(\lambda_i) &= \sum_n b_n \lambda_i^n, \quad b_n < 0 \end{aligned} \tag{11}$$

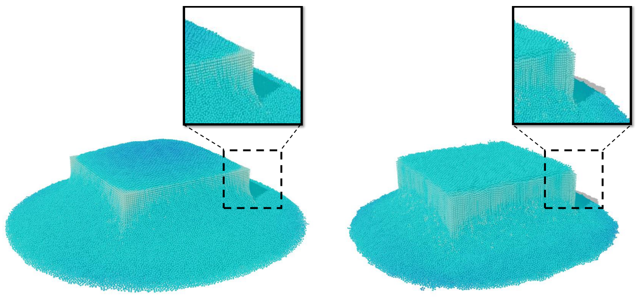
The question at hand is whether it is possible to express an arbitrary function of $\dot{B}(\lambda_i)$ as a polynomial. Indeed, the answer is affirmative, as we can consistently reframe an infinitely differentiable function as an infinite series of terms involving its derivatives evaluated at λ_i, following the principles of the Taylor series. Following the partitioning of $\dot{B}(\lambda_i)$, a semi-implicit step for updating positions is expressed as

$$\mathbf{x}_i^{k+1} = \frac{\mathbf{x}_i^* + \sum_j A_{ij}^{k-}(\mathbf{x}_j^k - \mathbf{x}_i^k) + \sum_j A_{ij}^{k+}(\mathbf{x}_j^k)}{1 + \sum_j A_{ij}^{k+}}, \tag{12}$$

where A_{ij}^{k+} and A_{ij}^{k-} are evaluated at each iteration explicitly as

$$\begin{aligned} A_{ij}^{k+} &= \mu \frac{h^2}{\rho_0} \dot{B}^-(\lambda_i^k) \frac{\partial W(r_{ij}^k, H)}{r_{ij}^k \partial r_{ij}^k}, \\ A_{ij}^{k-} &= \mu \frac{h^2}{\rho_0} \dot{B}^+(\lambda_i^k) \frac{\partial W(r_{ij}^k, H)}{r_{ij}^k \partial r_{ij}^k}. \end{aligned} \tag{13}$$

It's important to note that A_{ij}^{k+} is positive, while A_{ij}^{k-} is negative for all values of r_{ij} and λ_i. Furthermore, Equation 12 suggests that each particle's position can be independently updated. This characteristic renders the algorithm highly parallelizable and well-suited for modern GPU architectures.



(a) Conservation of momentum (b) Nonconservation of momentum

Figure 4: *A comparison illustrating momentum conservation using (a) the conservative scheme described in Equation 14 and the non-conservative scheme described in Equation 12. Particle velocities are color-coded. It is noteworthy that the fluid exhibits oscillations when momentum is not conserved.*

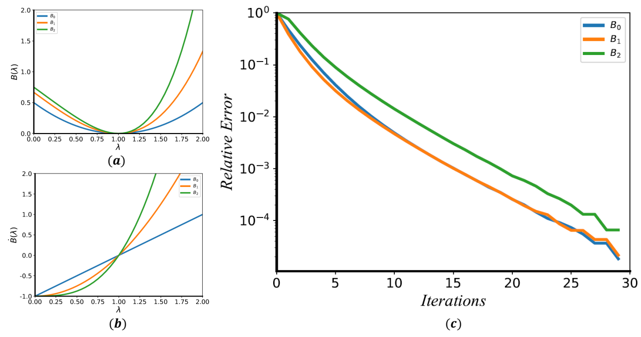
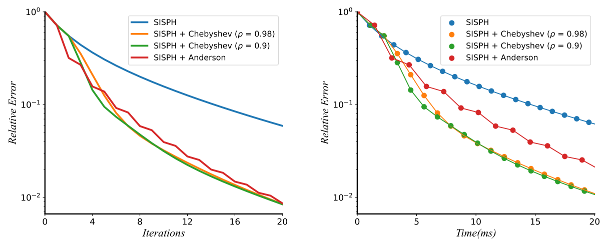


Figure 5: *Three representative bulk energy functions are employed to assess convergence: (a) the original bulk energy functions, (b) their first-order derivatives, and (c) the convergence rates for all three bulk energy functions.*



(a) (b)

Figure 6: *(a) The convergence of the semi-implicit successive substitution method is demonstrated to be effective with both Anderson acceleration and Chebyshev acceleration, achieving comparable convergence rates in terms of iteration numbers. (b) However, Chebyshev acceleration, being more suitable for GPU implementation, achieves superior overall performance.*

4.2. Momentum conservation

In standard SPH theory, when a particle j is located within the support domain of particle i, both particles i and j should be mutually stored in each other's neighbor list. However, this approach can significantly increase the size of neighbor lists for particles in highly compressible regions. Therefore, a more efficient solution is to store only a constant number of neighbors (e.g., around 30 to 40) that are closest to the central particles and scale the kernel function in Equation 9 with a constant at the beginning of simulation to ensure that the rest density remains at ρ_0. Consequently, when particle j is included in the neighbor list of particle i, there is no guarantee that particle i will reciprocally be present in the neighbor list of particle j, resulting in a breakdown of momentum conservation. To tackle this issue, we employ Newton's Third Law to enforce mutual interactions. Specifically, when a positional displacement of $\mu\frac{h^2}{\rho_0}\dot{B}(\lambda_i)\frac{\partial W_{ij}}{\partial r_{ij}}\frac{\mathbf{x}_i-\mathbf{x}_j}{r_{ij}}$ is applied to particle i by its neighboring particle j, particle i will, in turn, impose a reaction positional displacement of $-\mu\frac{h^2}{\rho_0}\dot{B}(\lambda_i)\frac{\partial W_{ij}}{\partial r_{ij}}\frac{\mathbf{x}_i-\mathbf{x}_j}{r_{ij}}$ onto particle j. Consequently, the semi-implicit step in Equation 12 can be reformulated to conserve momentum during position update as follows:

$$\mathbf{x}_i^{k+1} = \frac{\mathbf{x}_i^* + \sum_j \left(A_{ij}^{k-} + A_{ji}^{k-}\right)\left(\mathbf{x}_j^k - \mathbf{x}_i^k\right) + \sum_j \left(A_{ij}^{k+} + A_{ji}^{k+}\right)\mathbf{x}_j^k}{1 + \sum_j \left(A_{ij}^{k+} + A_{ji}^{k+}\right)} \tag{14}$$

Figure 4 illustrates a comparison that highlights the enhancement in fluid momentum conservation achieved through the use of Equation 14. When employing Equation 12 for updating particle positions, the fluid undergoes continuous oscillations, as depicted in Figure 4(b). Upon substituting the position updating scheme with Equation 14, the fluid gradually stabilizes, as evident in Figure 4(a).

4.3. Guarantee of convergence

It is essential to emphasize that the selection of the bulk energy function is not unique. In this section, we will examine three representative cases to illustrate the impact of the energy function on the convergence rate (see Figure 5(a) and 5(b) for the original functions and their first derivatives). The simplest choice is to employ a quadratic function, i.e., $B_0(\lambda_i) = \frac{1}{2}(\lambda_i - 1)^2$. We can derive the split of its first-order derivative as follows:

$$\dot{B}_0^+(\lambda_i) = \lambda_i, \qquad \dot{B}_0^-(\lambda_i) = -1. \tag{15}$$

The second choice involves an energy function, $B_1(\lambda_i) = \frac{\lambda_i^3-1}{3} - \lambda_i + 1$, inspired by [XHC*18]. The expression for the split of its first-order derivative is as follows:

$$\dot{B}_1^+(\lambda_i) = \lambda_i^2, \qquad \dot{B}_1^-(\lambda_i) = -1. \tag{16}$$

Similarly, the third choice uses a higher-order energy function of $B_2(\lambda_i) = \frac{\lambda_i^4-1}{4} - \lambda_i + 1$. The expression for the split of its first-order derivative is as follows:

$$\dot{B}_2^+(\lambda_i) = \lambda_i^3, \qquad \dot{B}_2^-(\lambda_i) = -1. \tag{17}$$

It should be noted that we have made slight adjustments to the coefficients of the energy functions to ensure that the values of $\dot{B}^+$ and $\dot{B}^-$ are equal for all three cases when $\lambda = 1$. Additionally, to prevent artificial particle clumping, it is common practice to clamp negative pressures to zero [ICS*14; MMCK14; BK16]. According to the definition of widely used equations of state (EOS), such as Tait's equation, clamping negative pressures to zero is equivalent to clamping particle densities below ρ_0 to ρ_0. Since our method does not require the calculation of an EOS-related variable, such as pressure, we propose using $\rho_i = max(\rho_i, \rho_0)$, and set $\rho_0 = 1000kg/m^3$ for all scenarios to ensure simplicity. Consequently, λ is guaranteed to be no smaller than 1.

For the purpose of comparing convergence rate, we utilize the relative error defined as

$$\varepsilon^k = \frac{\psi\left(\mathbf{x}^k\right) - \psi\left(\mathbf{x}^N\right)}{\psi\left(\mathbf{x}^0\right) - \psi\left(\mathbf{x}^N\right)}, \tag{18}$$

where $\mathbf{x}^0$ is the initial particle position, $\mathbf{x}^k$ is the k-th iterate, $\mathbf{x}^N$ is the final iterate. Here N should be chosen large enough so that the value of $\mathbf{x}^N$ is close to the exact solution of the energy minimization problem. In our present experimental studies, we have determined that a maximum iteration number of $N = 50$ is sufficient. The decrease of relative error for three cases is shown in Figure 5(c), where the step length search method proposed by Lu et al. [LHG*23] is used for all three cases to avoid the overshooting problem. To ensure completeness, we provide additional details on how to calculate the step length. Given $\mathbf{x}_i^k$ and $\mathbf{x}_i^{k+1}$ in Equation 14, the goal is to identify a new position between $\mathbf{x}_i^k$ and $\mathbf{x}_i^{k+1}$ that is able to minimize the total energy of ψ_i

$$\mathbf{x}_i \leftarrow \mathbf{x}_i^k + \alpha_i^k\left(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right) \tag{19}$$

where $\alpha_i \in [0,1]$. By expressing ψ_i into a first-order Taylor polynomial around $\mathbf{x}_i^k$, specifically $\psi_i = \psi_i^k + \alpha_i^k \frac{\partial\psi_i^k}{\partial\mathbf{x}_i^k}\left(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right)$, we define a relaxed upper limit estimation for α_i^k as follows

$$\alpha_i^k = \min\left[-\frac{\psi_i^k}{\frac{\partial\psi_i^k}{\partial\mathbf{x}_i^k}\left(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right)}, 1\right], \tag{20}$$

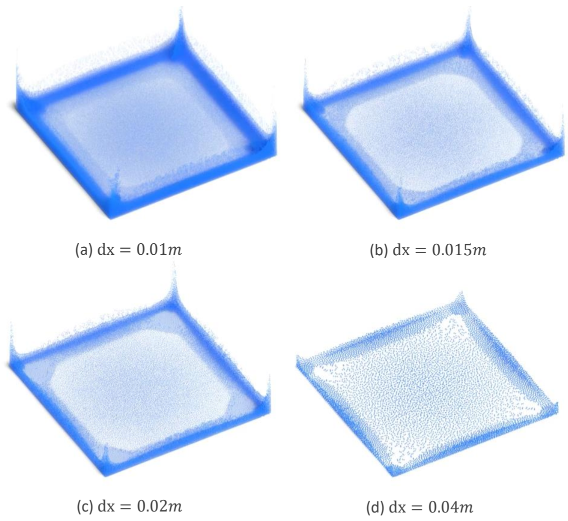


Figure 7: *Simulation of a dam-break scenario with a sampling distance ranging from* $0.01m$, $0.015m$, $0.02m$ *to* $0.04m$. *The time steps are all set to* $0.001s$, *the smoothing lengths are defined as 1.2 times the particle size dx, and the gravity is* $9.8m/s^2$.

where two relationships including $\psi_i \geq 0$ and $\frac{\partial \psi_i^k}{\partial \mathbf{x}_i^k}\left(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right) \leq 0$ are applied. It it noteworthy the value of $\frac{\partial \psi_i^k}{\partial \mathbf{x}_i^k}$ can be easily derived from Equation 14, the additional cost associated with calculating the step length is negligible.

During our experimental investigations, we have observed that as the order of the energy function steadily increases, the convergence rate ultimately decreases. This is because a higher-order energy function typically requires a smaller time step length. The only exception here is that the convergence speed of using B_1 seems to be slightly faster than that of using B_0. This indicates we can design a polynomial function with an order between 2 and 4 to achieve the fastest convergence speed. However, an advantage of utilizing B_0 lies in its obviation of the need for a step length search method. Therefore, we will consistently employ B_0 as the bulk energy function in the subsequent context if not specified.

Acceleration. As a specialized form of fixed-point iteration methods, the semi-implicit successive substitution method shares many features with the Jacobi method. For instance, the convergence rate is typically linear, which is often too slow for solving nonlinear optimization problems. Anderson acceleration is a commonly used technique to speed up the convergence of the fixed-point sequence [And65; PDZ*18]. Therefore, we first attempt to accelerate the convergence rate by applying Anderson acceleration and updating the sequence accordingly

$$\mathbf{x}^{k+1} = \sum_{i=0}^{n} a_i^k \mathbf{f}^{k-n+i}, \tag{21}$$

where $a^k = \arg\min_a \left\| \sum_{i=0}^n a_i (\mathbf{f} - \mathbf{x})^{k-n+i} \right\|_2$ represents the solution to the least square problem with $\sum_{i=0}^n a_i = 1$. By setting $n = 1$, SISSM has been found to converge faster. However, solving a^k requires dot product and reduction operations at each iteration, making Anderson acceleration unsuitable for GPU implementation. An alternative solution is to apply the Chebyshev acceleration [Wan15] and update the sequence as

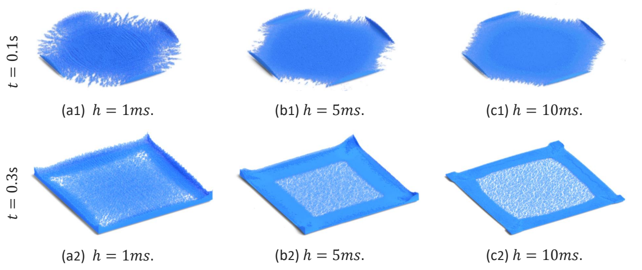


Figure 8: *Simulation of 70,000 particles with time steps set to* $1ms$, $5ms$, *and* $10ms$ *from left to right. The particle sizes are consistently set at* $0.01m$. *The smoothing lengths are uniform at* $1.2dx$, *equivalent to* $0.012m$. *The strengths of the artificial viscosity solver (XSPH) [SB12] are identically set at* 0.3.

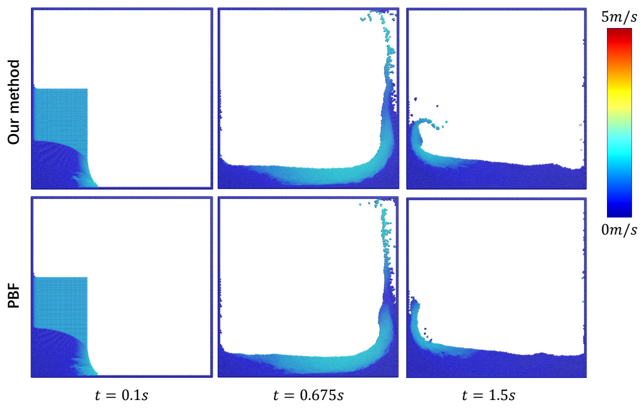


Figure 9: *A side by side comparison of our method with PBF in simulating incompressible fluids, showing consist velocity fields throughout the entire time history.*

$$\mathbf{x}^{k+1} = \omega_{k+1}(\mathbf{f}^k - \mathbf{x}^{k-1}) + \mathbf{x}^{k-1}, \tag{22}$$

where ω_{k+1} represents the relaxation parameter defined as

$$\omega_1 = 1, \quad \omega_2 = \frac{2}{2-\rho^2}, \quad \textit{and} \quad \omega_{k+1} = \frac{4}{4-\rho^2 \omega_k} \quad \textit{for } k \geq 2. \tag{23}$$

Figure 6 shows as the spectral radius ρ is set to 0.9, Chebyshev acceleration achieves a convergence rate comparable to Anderson acceleration. Furthermore, since Chebyshev acceleration does not require dot product or reduction operations, it introduces minimal computational overhead to each SISSM iteration. Consequently, Chebyshev acceleration demonstrates higher performance in terms of total computational cost, as shown in Figure 6(b).

5. Results and Discussion

We configure various scenarios to showcase features of our method. All experiments were performed on a laptop computer equipped with an Intel i9-12900H processor running at 2.90 GHz, an NVIDIA GeForce RTX 3080Ti laptop GPU, and 64GB of RAM. Time-consuming tasks, such as neighbor-list searching and numerical optimization, are parallelized on the GPU using CUDA. To address complex solid-fluid interactions, we have integrated the semi-analytical boundary handling technique [CLH*20], enabling the coupling of fluid particles with boundary triangle meshes. We use the XSPH method [SB12] to model artificial viscosity. The pseudocode of our method is available in Algorithm 1, and the time-cost statistics for the scenarios can be found in Table 5.4.

5.1. Performance comparison

To demonstrate the adaptability of our method to varying particle sizes, we simulate a dam-break scenario using different particle sizes, as depicted in Figure 7. As the particle sampling distance is increased from $0.01m$ to $0.04m$, our method maintains stable and consistent simulation results for all configurations. To better illustrate the versatility of our method across diverse particle sizes, Figure 1 presents a comparison encompassing three configurations of particle sizes. In this setup, five fluid cubes and cylindrical obstacles are introduced to enhance fluid details. A side-by-side comparison demonstrates that our method consistently models fluid behavior across various spacings. In addition to particle size, the timestep size is another crucial numerical parameter that may influence simulation outcomes [ICS*14; BK16; SP09]. Figure 8 shows a comparison of our method using different timestep sizes. Our method consistently produces simulation results that remain relatively stable across varying timestep sizes.

5.2. Compressible vs. Incompressible Flows

Figure 10 illustrates an example of fluid simulation that ranges from highly compressible to nearly incompressible. For simplicity, the quadratic function $B_0(\lambda_i) = \frac{1}{2}(\lambda_i - 1)^2$ is selected to model the bulk energy. Notably, when the constant coefficient μ is set to a small value of 10^{-4}, the momentum potential significantly contributes to the total energy. Consequently, high fluid compressibility is evident during the simulation. However, as the value of μ increases, the contribution from the momentum potential diminishes. When

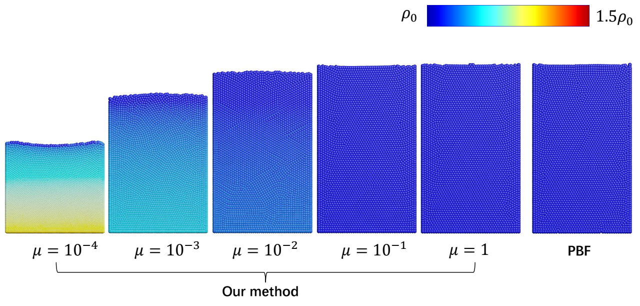


Figure 10: *Simulation of fluids ranging from highly compressible to nearly incompressible. Within our method, the bulk energy function is set to $B_0(\lambda_i) = \frac{1}{2}(\lambda_i - 1)^2$, the value of μ is changing from 10^{-4} to 1. Density field is color coded.*

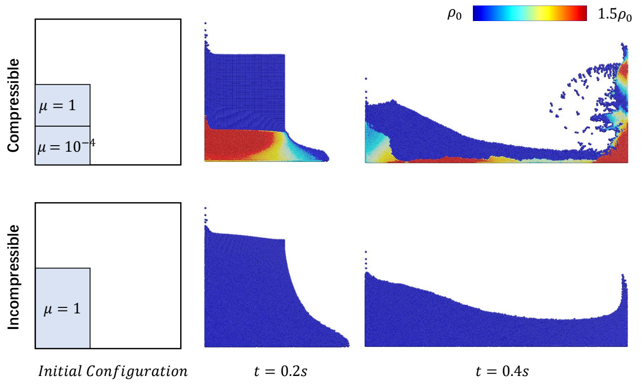


Figure 11: *A side by side comparison of our method in simulating compressible and incompressible fluids. Top: fluid simulation with half initialized as incompressible ($\mu = 1$) while the other half initialized as highly compressible ($\mu = 10^{-4}$); Bottom: the whole fluid is initialized as incompressible. Density is color coded for both simulations.*

μ reaches a sufficiently large value, this contribution can be effectively neglected. Specifically, when μ is increased to 1, the fluid behaviors simulated with our method converge to those of incompressible fluid as modeled by PBF [MM13]. Further increasing μ yields no additional differences in the simulation results.

To demonstrate the consistency of our method in generating incompressible fluids, Figure 9 shows a side-by-side comparison of our method with $\mu = 1$ and PBF [MM13]. The velocity fields for both methods are nearly consistent throughout the entire time history. Figure 11 provides another example that poses challenges for standard PBD methods. In this case, we couple a compressible fluid with an incompressible fluid by setting μ to 10^{-4} and 1, respectively. During the simulation, it is evident that the high compressibility of the compressible fluid and the incompressibility of the incompressible fluid are simultaneously maintained and well preserved.

5.3. Comparison among different SPH methods

Figure 12 demonstrates a comparison of our method with three widely used approaches—PBF [MM13], IISPH [ICS*14], and DFSPH [BK16]—under varying time step sizes. For fairness, our implementation of DFSPH includes only the constant density solver, excluding the divergence-free solver. Furthermore, to eliminate the influence of differing residual measurements, the number of iterations for all methods is fixed at 10 throughout the simulation. The comparison reveals that all methods produce consistent simulation results when the time step is set to a small value of $h = 1ms$. However, as the time step size increases, DFSPH generates more dynamic simulation results, whereas the results from other methods remain relatively consistent. Regarding stability, PBF, DFSPH, and our method can handle time steps as large as 32 ms, while IISPH becomes unstable at larger time steps. In terms of computational cost per iteration, DFSPH and our method demonstrate comparable performance, averaging approximately 0.4 ms per iteration. PBF

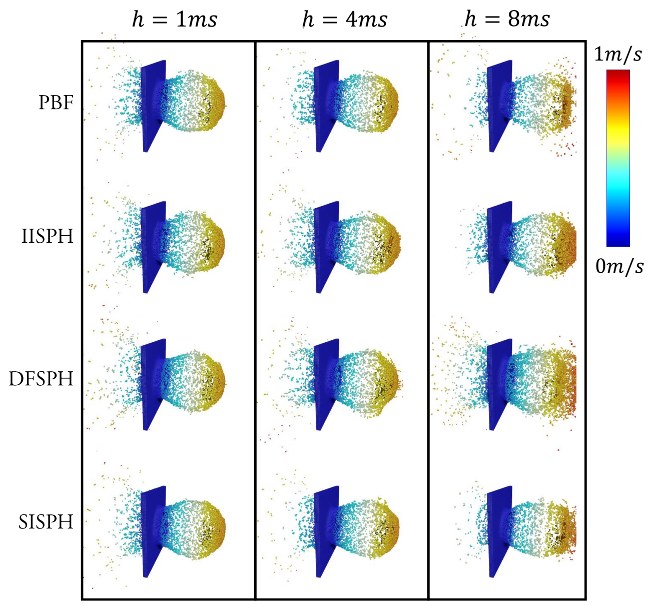


Figure 12: *This example shows the snap shot of a simulation at time $t = 0.832s$ to demonstrate the performance comparison among PBF [MM13], IISPH [ICS*14], DFSPH [BK16] and our method (SISPH) with different time step sizes. The particle size is set to $dx = 0.005m$, the smoothing length is $2dx$, and the constant μ is 1.*

incurs a slightly higher computational cost of 0.6 ms per iteration, whereas IISPH is the most computationally expensive solver, requiring about 2.4 ms per iteration. For more details and comparison regarding to the particle size and the smoothing length, we direct readers to the supplementary video and the accompanying source code available in PeriDyno. [†].

5.4. More examples

Our method provides more consistent animations to various timestep and particle sizes, therefore can readily integrate with other techniques to simulate increasingly complex scenarios. As an illustration, Figure 3 presents a flood simulation seawater over the seaside, which consists of 1.6 million particles and 2.5k triangles. Figure 13 simulated a fountain consisting of a maximum of 2.3 million fluid particles and 1.5 million boundary triangles. We utilize Continuous Collision Detection (CCD) [WTTM15] for both examples to detect collision between fluid particles and boundary triangles and then correct the particles' positions to prevent fluid particles from penetrating the solid. In a different scenario, as illustrated in Figure 2, we simulate a waterfall by employing a particle emitter. The sampling distance in this simulation can be adjusted to accommodate the specific requirements of the application, and there is no need for concern regarding numerical issues that may arise during the adjustment of particle size.

[†] https://github.com/peridyno/peridyno

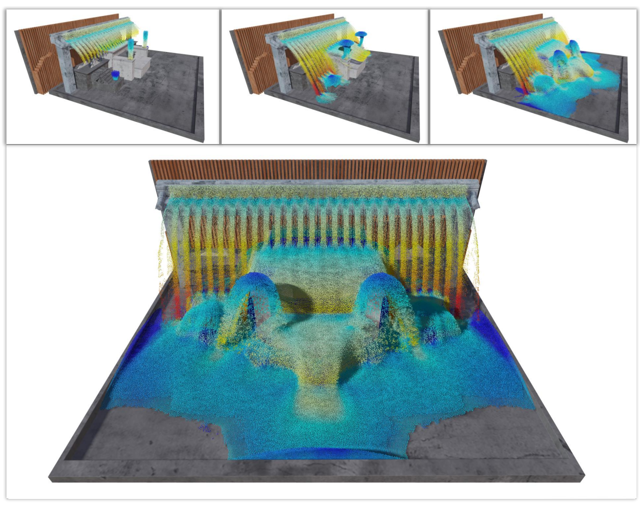


Figure 13: *Fountain. This scenario consists of a maximum of 2.5 million fluid particles and 1.5 million boundary triangle meshes, the particle sizes are set to $0.008m$.*

6. Conclusion and Limitations

By formulating the problem of solving fluid compressibility and incompressibility as a nonlinear optimization problem, we propose a semi-implicit successive substitution method that is GPU-friendly for solving fluid dynamics for SPH. Compared to the traditional PBD method, our approach alleviates the artificial dependence of numerical parameters, such as the iteration number, on material stiffness and enables the simulation of physically accurate fluids that range from highly compressible to nearly incompressible. Our method is easy to implement and compatible with GPU acceleration. It can reliably generate stable and consistent simulation outcomes for fluid systems characterized by numerical parameters including variations in particle size, time step, etc. The proposed method is expected to make particle-based fluid simulation more widely applicable for real-time applications.

The primary limitation of this study is that we tested only a subset of all bulk energy functions, leaving the optimal convergence rate undetermined. Although convergence rates can be improved

Figure	N_p	dx	$t_{neighbor}$	t_{solver}	t_{total}
Figure 1 Teaser(Left).	283k	0.01m	7.59ms	2.85ms	15.3ms
Figure 1 Teaser(Middle).	657k	0.075m	18.2ms	6.8ms	36.6ms
Figure 1 Teaser(Right).	2.18M	0.005m	61.3ms	21.9	125ms
Figure 2. Waterfall	0∼1.03M	0.005m	—	—	0∼105ms
Figure 3. Seaside	1.60M	0.005m	27.5ms	19.8ms	59.7ms
Figure 13. Fountain	0∼2.5M	0.008m	—	—	0∼262ms

Table 1: *In these scenarios, the iteration number of SISSM is set to 5, the time step is set to $1ms$, and the smoothing length is set to 1.2 times of the particle size, i.e., $1.2\mathrm{dx}$; N_p represents the number of particles; dx represents the particle size (particle spacing); $t_{neighbor}$ is the computational cost to find neighbors; t_{solver} is the computational cost for each SISSM iteration; t_{total} represents the average computational cost per time step.*

using acceleration techniques from traditional solvers, overall performance is influenced by various factors, such as the spectral radius and the number of sequences from previous iterations used to compute the new iteration. Moreover, our current implementation of the semi-implicit solver does not include special treatments for particles near the boundary. Therefore, an additional boundary handling step is necessary to prevent particle penetration into the solid. In future work, we aim to extend our method to address both one-way and two-way fluid-solid coupling. Finally, fluid compressibility is not modeled after real materials. It would be intriguing to extend our method to simulate real compressible materials, such as snow simulation [GHB*20].

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments. This research was supported by the National Science and Technology Major Project (No. 2022ZD0119404) and the National Natural Science Foundation of China (No.62302490), the Basic Research Project of ISCAS (No. ISCAS-JCMS-202403).

References

[And65] ANDERSON, DONALD G. "Iterative procedures for nonlinear integral equations". *Journal of the ACM (JACM)* 12.4 (1965), 547–560 7.

[BBB07] BATTY, CHRISTOPHER, BERTAILS, FLORENCE, and BRIDSON, ROBERT. "A fast variational framework for accurate solid-fluid coupling". *ACM Trans. Graph.* 26.3 (July 2007), 100–es. ISSN: 0730-0301 3.

[BGI*18] BAND, STEFAN, GISSLER, CHRISTOPH, IHMSEN, MARKUS, et al. "Pressure boundaries for implicit incompressible SPH". *ACM Trans. Graph.* 37.2 (2018), 1–11 2.

[BK16] BENDER, JAN and KOSCHIER, DAN. "Divergence-free SPH for incompressible and viscous fluids". *IEEE Transactions on Visualization and Computer Graphics* 23.3 (2016), 1193–1206 1, 3, 6, 8, 9.

[BLS11] BODIN, KENNETH, LACOURSIERE, CLAUDE, and SERVIN, MARTIN. "Constraint fluids". *IEEE Transactions on Visualization and Computer Graphics* 18.3 (2011), 516–526 2.

[BML*14] BOUAZIZ, SOFIEN, MARTIN, SEBASTIAN, LIU, TIANTIAN, et al. "Projective dynamics: fusing constraint projections for fast simulation". *ACM Trans. Graph.* 33.4 (2014), 1–11 2, 4.

[BT07] BECKER, MARKUS and TESCHNER, MATTHIAS. "Weakly compressible SPH for free surface flows". *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2007, 209–217 2.

[CBG*19] CORNELIS, JENS, BENDER, JAN, GISSLER, CHRISTOPH, et al. "An optimized source term formulation for incompressible SPH". *The Visual Computer* 35.4 (2019), 579–590 3.

[CLH*20] CHANG, YUE, LIU, SHUSEN, HE, XIAOWEI, et al. "Semi-analytical Solid Boundary Conditions for Free Surface Flows". *Computer Graphics Forum*. Vol. 39. 7. Wiley Online Library. 2020, 131–141 8.

[DG96] DESBRUN, MATHIEU and GASCUEL, MARIE-PAULE. "Smoothed Particles: A new paradigm for animating highly deformable bodies". *Computer Animation and Simulation'96: Proceedings of the Eurographics Workshop in Poitiers, France, August 31–September 1, 1996*. Springer. 1996, 61–76 2.

[GHB*20] GISSLER, CHRISTOPH, HENNE, ANDREAS, BAND, STEFAN, et al. "An implicit compressible SPH solver for snow simulation". *ACM Trans. Graph.* 39.4 (Aug. 2020). ISSN: 0730-0301 10.

[GK16] GOTOH, HITOSHI and KHAYYER, ABBAS. "Current achievements and future perspectives for projection-based particle methods with applications in ocean engineering". *Journal of Ocean Engineering and Marine Energy* 2 (2016), 251–278 5.

[HLL*12] HE, XIAOWEI, LIU, NING, LI, SHENG, et al. "Local Poisson SPH For Viscous Incompressible Fluids". *Computer Graphics Forum* 31.6 (Sept. 2012), 1948–1958. ISSN: 0167-7055 1, 2.

[HLW*12] HE, XIAOWEI, LIU, NING, WANG, GUOPING, et al. "Staggered meshless solid-fluid coupling". *ACM Trans. Graph.* 31.6 (2012), 1–12 3, 5.

[ICS*14] IHMSEN, MARKUS, CORNELIS, JENS, SOLENTHALER, BARBARA, et al. "Implicit Incompressible SPH". *IEEE Transactions on Visualization and Computer Graphics* 20.3 (2014), 426–435 2, 6, 8, 9.

[KBST19] KOSCHIER, DAN, BENDER, JAN, SOLENTHALER, BARBARA, and TESCHNER, MATTHIAS. "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids". *Eurographics 2019-Tutorials* (2019) 4.

[KBST22] KOSCHIER, DAN, BENDER, JAN, SOLENTHALER, BARBARA, and TESCHNER, MATTHIAS. "A Survey on SPH Methods in Computer Graphics". *Computer Graphics Forum* 41.2 (2022), 737–760 1.

[KS14] KANG, NAHYUP and SAGONG, DONGHOON. "Incompressible SPH using the divergence-free condition". *Computer Graphics Forum* 33.7 (2014), 219–228 3.

[KUJH21] KEE, MIN HYUNG, UM, KIWON, JEONG, WOOSEOK, and HAN, JUNGHYUN. "Constrained projective dynamics: real-time simulation of deformable objects with energy-momentum conservation". *ACM Trans. Graph.* 40.4 (2021), 1–12 4.

[Lat14] LATIF, ABDUL. "Banach Contraction Principle and Its Generalizations". *Topics in Fixed Point Theory*. Ed. by ALMEZEL, SALEH, ANSARI, QAMRUL HASAN, and KHAMSI, MOHAMED AMINE. Cham: Springer International Publishing, 2014, 33–64 3.

[LBB17] LARIONOV, EGOR, BATTY, CHRISTOPHER, and BRIDSON, ROBERT. "Variational stokes: a unified pressure-viscosity solver for accurate viscous liquids". *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301 3.

[LHG*23] LU, ZIXUAN, HE, XIAOWEI, GUO, YUZHONG, et al. "Projective Peridynamic Modeling of Hyperelastic Membranes With Contact". *IEEE Transactions on Visualization and Computer Graphics* (2023) 2–4, 6.

[LTKF08] LOSASSO, FRANK, TALTON, JERRY, KWATRA, NIPUN, and FEDKIW, RONALD. "Two-Way Coupled SPH and Particle Level Set Fluid Simulation". *IEEE Transactions on Visualization and Computer Graphics* 14.4 (2008), 797–804 3.

[MCG03] MÜLLER, MATTHIAS, CHARYPAR, DAVID, and GROSS, MARKUS. "Particle-based fluid simulation for interactive applications". *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Citeseer. 2003, 154–159 2, 5.

[MM13] MACKLIN, MILES and MÜLLER, MATTHIAS. "Position based fluids". *ACM Trans. Graph.* 32.4 (2013), 1–12 1, 2, 4, 8, 9.

[MMC16] MACKLIN, MILES, MÜLLER, MATTHIAS, and CHENTANEZ, NUTTAPONG. "XPBD: position-based simulation of compliant constrained dynamics". *Proceedings of the 9th International Conference on Motion in Games*. 2016, 49–54 2, 4.

[MMCK14] MACKLIN, MILES, MÜLLER, MATTHIAS, CHENTANEZ, NUTTAPONG, and KIM, TAE-YONG. "Unified particle physics for real-time applications". *ACM Trans. Graph.* 33.4 (July 2014). ISSN: 0730-0301 6.

[PDZ*18] PENG, YUE, DENG, BAILIN, ZHANG, JUYONG, et al. "Anderson acceleration for geometry optimization and physics simulation". *ACM Trans. Graph.* 37.4 (2018), 1–14 7.

[RWT11] RAVEENDRAN, KARTHIK, WOJTAN, CHRIS, and TURK, GREG. "Hybrid smoothed particle hydrodynamics". *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*. 2011, 33–42 2.

[SB12] SCHECHTER, HAGIT and BRIDSON, ROBERT. "Ghost SPH for animating water". *ACM Trans. Graph.* 31.4 (2012), 1–8 7, 8.

[SP09] SOLENTHALER, B and PAJAROLA, R. "Predictive-corrective incompressible SPH". Vol. 28. 3. ACM New York, NY, USA, 2009, 1–6 1, 2, 8.

[TDNL18] TAKAHASHI, TETSUYA, DOBASHI, YOSHINORI, NISHITA, TOMOYUKI, and LIN, MING C. "An efficient hybrid incompressible SPH solver with interface handling for boundary conditions". *Computer Graphics Forum*. Vol. 37. 1. Wiley Online Library. 2018, 313–324 2.

[Wan15] WANG, HUAMIN. "A chebyshev semi-iterative approach for accelerating projective and position-based dynamics". *ACM Trans. Graph.* 34.6 (Nov. 2015). ISSN: 0730-0301 7.

[WKB16] WEILER, MARCEL, KOSCHIER, DAN, and BENDER, JAN. "Projective fluids". *Proceedings of the 9th International Conference on Motion in Games*. 2016, 79–84 2.

[WTTM15] WANG, ZHENDONG, TANG, MIN, TONG, RUOFENG, and MANOCHA, DINESH. "TightCCD: Efficient and Robust Continuous Collision Detection using Tight Error Bounds". *Computer Graphics Forum* 34.7 (2015), 289–298 9.

[XHC*18] XU, LIYOU, HE, XIAOWEI, CHEN, WEI, et al. "Reformulating Hyperelastic Materials with Peridynamic Modeling". *Computer Graphics Forum* 37.7 (2018), 121–130 6.

[XLYJ23] XIE, TIANYI, LI, MINCHEN, YANG, YIN, and JIANG, CHENFANFU. "A Contact Proxy Splitting Method for Lagrangian Solid-Fluid Coupling". *ACM Trans. Graph.* 42.4 (July 2023). ISSN: 0730-0301 3.

[YHW*16] YANG, SHENG, HE, XIAOWEI, WANG, HUAMIN, et al. "Enriching SPH simulation by approximate capillary waves". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '16. Zurich, Switzerland: Eurographics Association, 2016, 29–36. ISBN: 9783905674613 3.