Semi-Implicit Incompressible SPH

Category: Research

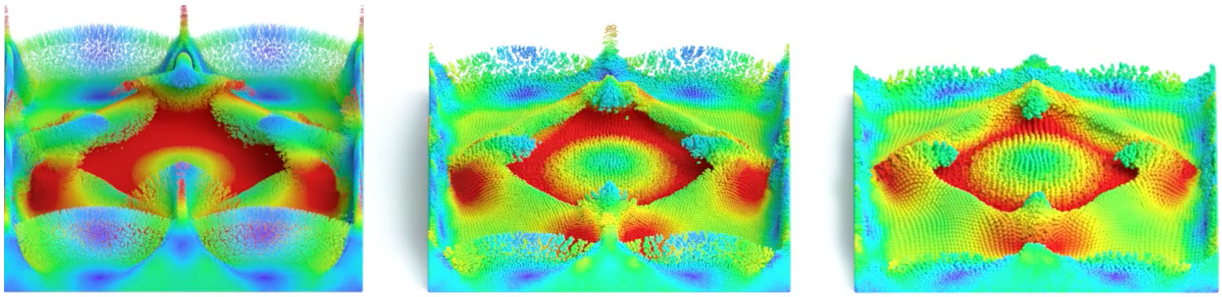


Figure 1: The simulation scenario encompasses various particle sizes, specifically 0.01m (left), 0.03m (middle), and 0.05m (right), with corresponding particle counts of 23 million (left), 0.83 million (middle), and 0.19 million (right) particles. For improved visualization, particle velocities are represented through color-coding in the rendering. Note our method can reliably generate stable and consistent simulation outcomes for varying particle sizes.

Abstract

When simulating incompressible 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. We introduce a semi-implicit successive substitution method for iteratively adjusting particle positions to ensure fluid incompressibility. Unlike the Gauss–Newton method commonly used in the PBD framework, the proposed semi-implicit approach eliminates the need to adjust numerical parameters for each scenario. It reliably generates consistent and stable simulation outcomes for fluid systems characterized by diverse numerical parameters, including variations in particle size, time step, etc. In contrast to the PBF method, our approach demonstrates superior adaptability to diverse particle-spacing values, offers a more straightforward implementation, and enhances efficiency. Furthermore, we have developed a real-time simulation system based on our approach, showing its convenience in generating real-time fluids for virtual reality scenarios.

Index Terms: Computing methodologies—Computer Graphics—Animation—Physical simulation

1 Introduction

In real-time simulation of particle-based fluids, position based approaches are commonly used to enforce fluid incompressibility. Its basic idea is to formulate fluid incompressibility as a set of non-linear positional constraints that solve constant density for particles [20]. During the time integration, the Gauss–Newton method is used to iteratively correct the predicted positions until all constraints are satisfied under 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 [21]. 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. Another issue associated with PBD arises from employing the Gauss–Newton method. Within PBD, each constraint is solved separately, therefore the subsystem that corresponds to each constraint is highly under-determined. To overcome this problem, PBD linearizes the constraint function and restricts the position update to be in the direction of the gradient of the constraint equation [21]. As a result, only one scalar Lagrange multiplier λ has to be found for each constraint equation. Unfortunately, the convergence of the Gauss–Newton method is not guaranteed if the stepsize is not properly controlled [24,26]. For instance, when developing an asset for simulating incompressible fluids using PBD, it becomes essential to proportionally adapt the stepsize in accordance with the particle size. Failing to do so may result in either inadequate preservation of fluid volume or potential simulation failures.

In response to the constraints of conventional PBD methods, we introduce a semi-implicit incompressible solver tailored for real-time fluid simulations. In lieu of enforcing fluid incompressibility via density constraints, we introduce a reformulation that models 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 [5]. The resolution of fluid incompressibility is thus transformed into a nonlinear optimization problem. In pursuit of its minimization, we introduce a semi-implicit successive substitution method, inspired by Lu et al. [19], which iteratively and concurrently adjusts particle positions to enforce fluid incompressibility. Unlike the Gauss–Newton method, our proposed semi-implicit approach eliminates the need for stepsize adjustments in various scenarios. As depicted in Figure 1, our method consistently produces simulation results for fluid dynamics across different particle sizes, even with a minimal number of iterations.

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.

Figure 2: Waterfall. This scenario consists of 1 million particles, and the particle size is $0.01m$.

2.1 Constant-density approach

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 [8] first introduced smoothed particle hydrodynamics into computer graphics for animating highly deformable bodies. Subsequently, several non-iterative EOS solvers were introduced, differing only slightly in their utilization of the equation of state (EOS). As an illustration, Müller et al. [22] employ a modified version of the method proposed by Desbrun and Gascuel [8], whereas Becker and Teschner [2] 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 [27] introduced the inaugural iterative EOS Solver, referred to as Predictive-Corrective ISPH (PCISPH), designed to enforce fluid incompressibility. Subsequently, He et al. [10] introduced an integral form derived from the pressure Poisson equation, aiming to enhance the convergence rate of the predictive-corrective scheme. Bodin et al. [4] 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 [20] 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. [21]. Weiler et al. [31] 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. [1, 13] proposed to combine a symmetric SPH pressure force and an SPH discretization of the continuity equation to improve the convergence rate. Takahashi et al. [28] proposed a hybrid SPH solver with a new interface handling method to address issues in traditional projection-based solvers.

2.2 Divergence-free approach

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 [23] 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. [11] 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. [33] 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 [3] 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 [14] 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. [18] 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. [7] 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.

3 Overview

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

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 [17]. 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 $\|f'(x_k)\| < 1$.

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

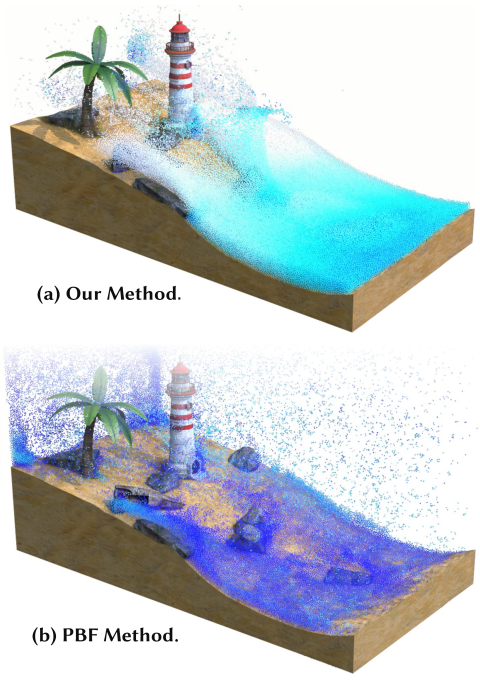


Figure 3: Seaside. The scenario consists of 600k particles and 2.5k boundary triangles. When the particle size is set to $0.02m$ (b) the PBF method exhibits artifacts when numerical parameters are not well adjusted, (a) whereas our method remains stable without the need to adjust numerical parameters.

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. [19] 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 we assume both x and x^* are real numbers here for the sake of simplicity, and c represents a constant. It's 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. [19] suggest employing a semi-implicit strategy for linearizing Equation 2 at (x_k, x_k^*), as follows:

$$x_{k+1} = f^+(x_k,x_k^*)(x_k^*-x_{k+1})+f^-(x_k,x_k^*)(x_k^*-x_k), \tag{3}$$

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:

$$x_{k+1} = \frac{f^+(x_k,x_k^*)x_k^* + f^-(x_k,x_k^*)(x_k^*-x_k)}{1+f^+(x_k,x_k^*)}. \tag{4}$$

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. Conversely, SISSM's convergence is assured when we meticulously split $f(x,x^*)$ into its positive and negative components.

4 Semi-implicit Incompressibility Solver

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 [20]

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

where ρ_0 is the rest density and ρ_0 is calculated with a particle approximation as [16]

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

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 $r_{ij} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|$ to prevent significant volume loss, enforcing the constraint in Equation 6 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 [20] can be seen as a variant of the Gauss-Newton algorithm. Nevertheless, as highlighted in Macklin's work [21], a significant drawback of position-based dynamics is that the stiffness of constraints varies with both the iteration count and step size, posing challenges in achieving predictable physical behaviors. Inspired by the principles of projective dynamics [5, 15, 19], we expand the constant-density constraint and formulate the objective function in the following variational manner:

$$\psi = \frac{1}{2h^2} m \|\mathbf{x}-\mathbf{s}\|_F^2 + \mu \sum_i B(\lambda_i), \tag{7}$$

where $\mathbf{x}$ denotes the column vector containing all particle positions, $\mathbf{s}$ represents the intermediate position, h signifies the time step, $\|\cdot\|_F^2$ stands for the Frobenius norm, and $B(\lambda_i)$ represents the bulk energy associated with particle i. 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

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

where $\dot{B}$ represents the first-order derivative of B with respect to λ, W_{ij} is short for $W(\mathbf{x}_i - \mathbf{x}_j, H)$. Please note that when we take the derivative of ψ with respect to $\mathbf{x}_i$ to obtain Equation 8, we have temporally neglected contributions from all neighboring particle j to make the following discussion concise. Later, we

ALGORITHM 1: Semi-Implicit Incompressible SPH

while $t < t_{stop}$ **do**
  for *all particles* i **do**
    $\mathbf{v}_i^* \leftarrow \mathbf{v}_i^n + h\cdot\mathbf{f}$;
    $\mathbf{x}_i^* \leftarrow \mathbf{x}_i^n + h\cdot\mathbf{v}_i^*$;
  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 $k < N$ **do**
    for *all particles* i **do**
      compute ρ_i using Eq. 6 based on $\mathbf{x}^k$;
      compute $\mathbf{x}_i^{k+1}$ using Eq. 14;
      $k = k+1$;
    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

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 job 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 [9, 11]. Consequently, we employ Debrun's spiky kernel [22]

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

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

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

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

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

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

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

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

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

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.

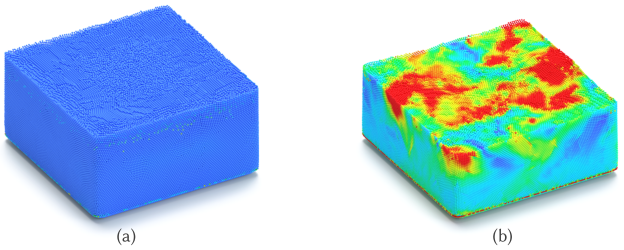


Figure 4: A comparison illustrating momentum conservation using (a) the conservative scheme described in Equation 14 and the non-conservative scheme described in Equation 12. Particle velocities are color-coded. It is noteworthy that the fluid exhibits oscillations when momentum is not conserved.

4.2 Momentum conservation

In our present implementation, we consistently maintain a maximum of 40 neighboring particles for each individual fluid particle. 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:

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

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).

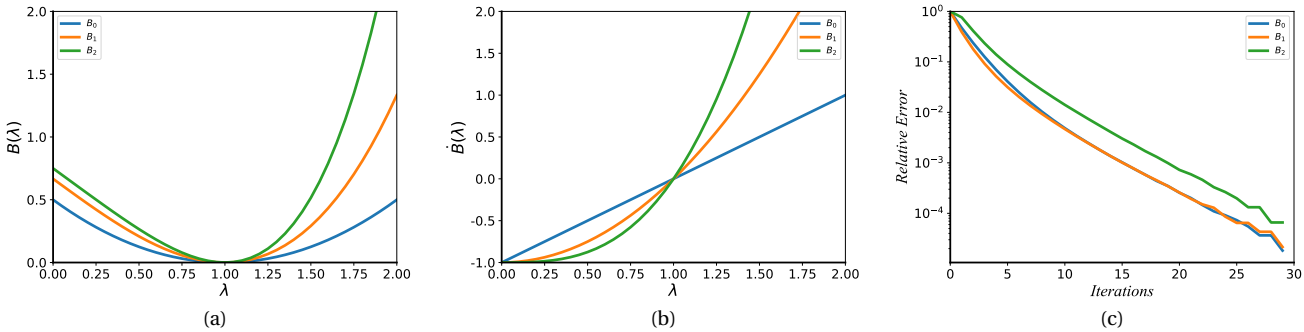


Figure 5: Three representative bulk energy functions are employed to assess convergence: (a) the original bulk energy functions, (b) their first-order derivatives, and (c) the convergence rates for all three bulk energy functions.

5 Results and Discussion

Our method is implemented using the open-source project PeriDyno[1], and we configure various scenarios to showcase its features. 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 incompressibility solving, are parallelized on the GPU using CUDA. To address complex solid-fluid interactions, we have integrated the semi-analytical boundary handling technique [6], enabling the coupling of fluid particles with boundary triangle meshes. We use the XSPH method [25] to model artificial viscosity and employ the surface tension method introduced by He et al. [12] for specific examples. The pseudocode of our SIISPH method is available in Algorithm 1, and the time-cost statistics for the scenarios can be found in Table 1.

5.1 Convergence analysis

It is essential to emphasize that the selection of the bulk energy function is not exclusive. 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 [32]. 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 $\lambda=1$.

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

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

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

where $\mathbf{x}^0$ is the initial particle position, $\mathbf{x}^k$ is the k-th iterate, $\mathbf{x}^N$ is the maximum 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. [19] is used for all three cases to avoid the overshooting problem. 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.

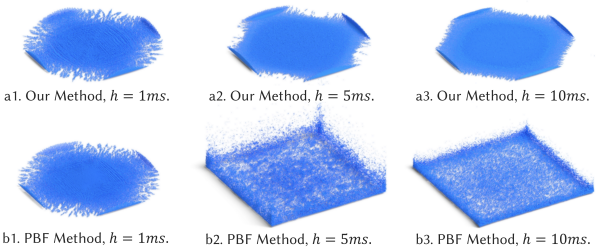


Figure 6: We compared our method to the PBF method [20] with different time steps, both employing 70,000 particles. Time steps are set to $1ms$, $5ms$, and $10ms$ from left to right. The particle sizes are consistently set at $0.01m$. The smoothing lengths are uniform at $1.2dx$, equivalent to $0.012m$. Both our method and the PBF method employed 3 iterations. The strengths of the artificial viscosity solver (XSPH) [25] are identically set at 0.3.

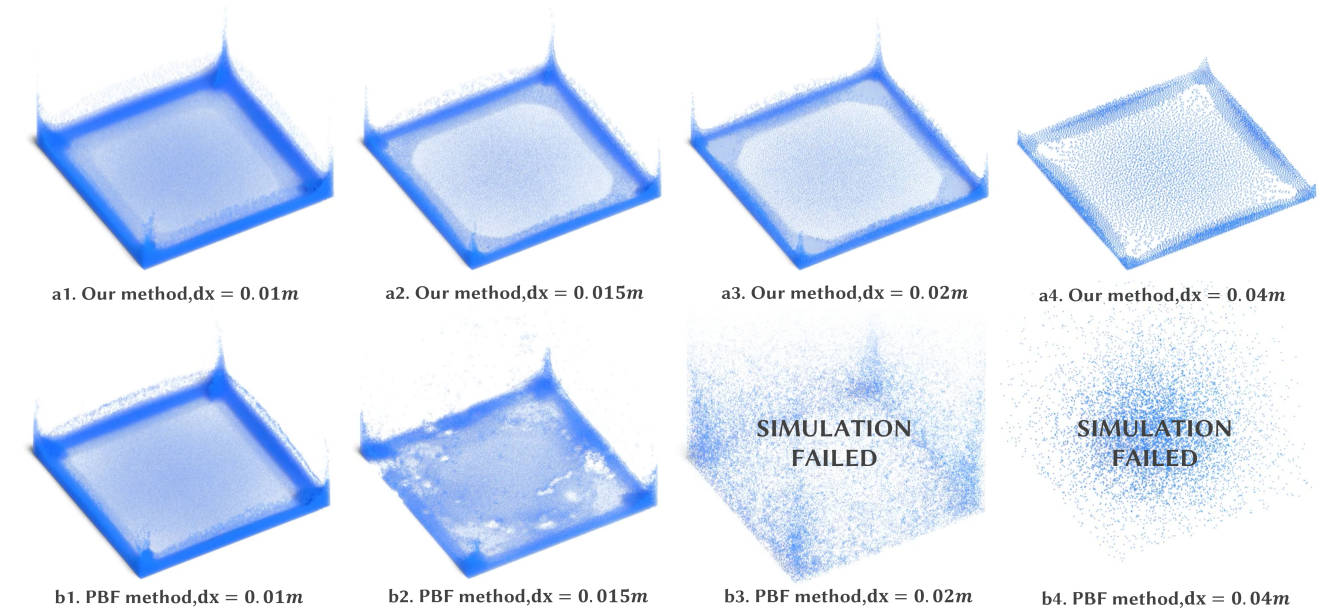
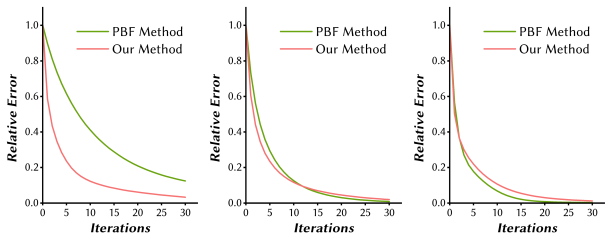


Figure 7: Comparison to the PBF method. The dam-break is respectively simulated using the PBF method [20] and our method with $0.01m$, $0.015m$, $0.02m$ and $0.04m$. The time steps are all set to $0.001s$, the smoothing lengths are defined as 1.2 times the particle size dx, and the gravity is $9.8m/s^2$.

5.2 Performance comparison

To demonstrate the adaptability of our method to varying particle sizes, we simulate a dam-break scenario. We compare our method with the PBF method, each using different particle sizes, as depicted in Figure 7. For a particle size of $0.01m$, we made minor adjustments to the step length of the Guess-Newton method within the PBF framework to ensure stable and consistent simulation results for both methods. Nevertheless, with an increase in particle size, the fluids simulated by the PBF method exhibit progressive destabilization. In contrast, our method maintains stability even with significantly larger particle sizes. To elucidate the cause of instability observed in the PBF method, we compare the convergence rates of both methods for scenarios with smaller particle sizes, as illustrated in Figure 8. It is noteworthy that as the particle size increases from $0.005m$ to $0.01m$, the convergence rate of our method remains stable, whereas that of the PBF method exhibits significant variation. As particle size exceeds $0.015m$, the PBF method's iterations finally exhibit oscillations or divergence. To better illustrate the versatility of our method across diverse particle sizes, Figure 1 shows a comparison encompassing a broader spectrum of particle sizes. In this setup, the test with the highest particle resolution comprises 23 million particles and maintains an average computation time of 1.67 seconds per frame. A side by side comparison shows our method consistently models fluid behavior across various spacings. Figure 3 presents a further comparison in fluid simulation, employing a particle size of $dx = 0.02m$. Our method consistently models fluid behaviors, whereas the PBF method fails to produce physically plausible results unless numerical parameters are meticulously fine-tuned.



(a) $0.005m$ (b) $0.0075m$ (c) $0.010m$

Figure 8: We compare the convergence of the dam-break scenario in Figure 7, with different particle sizes: (a) particle size of $0.005m$; (b) particle size of $0.0075m$; (c) particle size of $0.010m$. We calculate the relative errors for both our method and the PBF method using Equation 18. It is observed that, in contrast to our method, the convergence speed of the PBF method is significantly influenced by the particle size.

In addition to particle size, the timestep size is another crucial numerical parameter that may influence simulation outcomes [3, 13, 27]. It is well known the PBF method is stable for simulating fluids with large time steps. However, there exists a crucial prerequisite for the PBF method, i.e., numerical parameters should be adjusted for each specific scenario. Figure 6 shows a comparison between our method and the PBF method using different timestep sizes. For a timestep size of $1ms$, we have made minor adjustments to numerical parameters for the PBF method to ensure stable and consistent simulation results for both methods. If we only increase the timestep size without changing other numerical parameters, such as the step length used in the Guass-Newton method, instability arises in the PBF method when the timestep size exceeds $5ms$. Conversely, our method consistently produces simulation results that remain relatively stable across varying timestep sizes.

5.3 More examples

Thanks to our method's excellent adaptability to various timestep and particle sizes, our semi-implicit approach can readily integrate with other techniques to simulate increasingly complex scenarios. As an illustration, Figure 9 simulated a scenario consisting of 2.3 million fluid particles and 1.5 million boundary triangles. We utilize Continuous Collision Detection (CCD) [30]

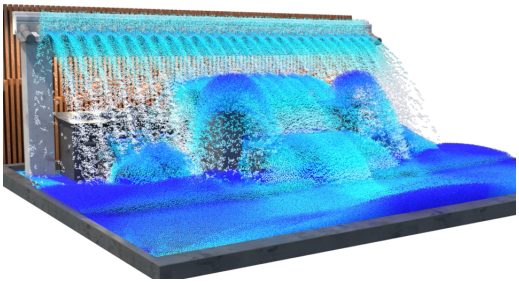


Figure 9: Fountain. This scenario consists of 2.5 million fluid particles and 1.5 million boundary triangle meshes, the particle sizes are set to $0.01m$.

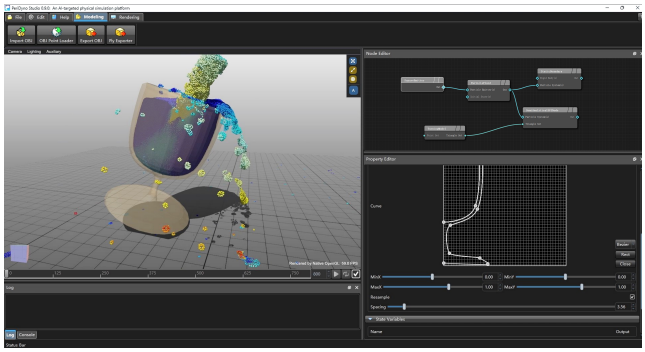


Figure 10: The real-time simulation system we have developed encompasses modules for real-time rendering, modeling, and other crucial components essential for conducting real-time simulations. This system enables the rapid creation of complex simulation scenarios through node-based connections, eliminating the necessity for manual coding.

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.

5.4 A simulation pipeline based on PeriDyno

To facilitate the application of our method, we have developed a real-time simulation system utilizing a node-based CPU-GPU hybrid architecture. This system encompasses real-time rendering, modeling, and other vital components essential for conducting real-time simulations. Additionally, the system features a real-time interactive interface, as illustrated in Figure 10. Through this interface, users can create intricate simulation scenarios by constructing nodes and establishing connections between them. Specifically, the "Particle Fluid" node governs the animation pipeline of the Semi-Implicit Incompressible SPH method depicted in Algorithm 1; the "Static Boundary" generates simulation domain boundaries utilizing the signed distance field; the "Turning Model" node represents the goblet using multiple triangle meshes; the "Square Emitter" node generates fluid particles with prescribed velocities; and the "Semi-Analytical SFI Node" is responsible for coupling solid boundaries with fluids, based on the work of Chang et al. [6]. Each node in the system is equipped with its "property editor" and various computing modules that users can manipulate and define.

Owing to the robustness inherent in our semi-implicit method and the utilization of GPU parallelism in the majority of our algorithms, the real-time simulation system we have designed demonstrates exceptional speed and efficiency. This is substantiated by the performance statistics provided in Table 1 and the supplementary material. Consequently, our system is particularly well-suited for applications within the realm of Virtual Reality (VR).

Table 1: Statistics and timings per timestep for all examples.

Figure[2]	N_p[3]	dx[4]	t_{total}[5]
Figure 1(Left). Teaser	23.0M	0.1m	1.67s
Figure 1(Middle). Teaser	831k	0.3m	82.7ms
Figure 1(Right). Teaser	195k	0.5m	16.5ms
Figure 2. Waterfall	0∼1.03M	0.1m	0∼105ms
Figure 3. Seaside	600k	0.2m	69.7ms
Figure 7(a1). Dam break	531k	0.1m	35.9ms
Figure 7(a2). Dam break	135k	0.15m	13.5ms
Figure 7(a3). Dam break	6.44k	0.2m	6.44ms
Figure 7(a4). Dam break	4.20k	0.4m	4.21ms
Figure 9. Fountain	0∼2.09M	0.1m	0∼45.9ms
Figure 10. goblet	0∼20k	0.05m	0∼11.2ms

*2.*In these scenarios, the iteration numbers of our semi-implicit ISPH solvers are set to 3, the time step are $1ms$, and the smoothing length of kernel are 1.2 times of the particle size, i.e., 1.2dx;
3. N_p represents the number of particles;
4. dx represents the particle size (particle spacing);
5. t_{total} represents the average computational cost per time step.

6 Conclusion and Limitations

We have presented a semi-implicit successive substitution method to solve fluid incompressibility that is formulated as a nonlinear optimization problem of particle positions. Compared to the traditional PBD method, our method eliminates the need to adjust numerical parameters one by one when creating fluid assets for VR applications. 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. It is expected the proposed method makes particle-based fluid simulation more widely applicable for real-time applications.

The major limitation is that we only tested with a subset of all bulk energy functions, therefore it is not clear which one gives the best convergence rate. We believe the convergence rate can be accelerated by taking acceleration techniques used in traditional solvers, e.g., by taking the Chebyshev method [29]. Furthermore, our current implementation of the semi-implicit solver introduces no special treatments for particles that are near the boundary, therefore, an additional boundary handling step should be included to avoid particle penetration into the solid. We would also like to investigate on how to extend our method to handle one-way and two-way fluid-solid coupling in our future work.

References

[1] S. Band, C. Gissler, M. Ihmsen, J. Cornelis, A. Peer, and M. Teschner. Pressure boundaries for implicit incompressible sph. *ACM Transactions on Graphics (TOG)*, 37(2):1–11, 2018.

[2] M. Becker and M. Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 209–217, 2007.

[3] J. Bender and D. Koschier. Divergence-free sph for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1193–1206, 2016.

[4] K. Bodin, C. Lacoursiere, and M. Servin. Constraint fluids. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):516–526, 2011.

[5] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics*, 33(4):1–11, 2014.

[6] Y. Chang, S. Liu, X. He, S. Li, and G. Wang. Semi-analytical solid boundary conditions for free surface flows. In *Computer Graphics Forum*, vol. 39, pp. 131–141. Wiley Online Library, 2020.

[7] J. Cornelis, J. Bender, C. Gissler, M. Ihmsen, and M. Teschner. An optimized source term formulation for incompressible sph. *The Visual Computer*, 35(4):579–590, 2019.

[8] M. Desbrun and M.-P. Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation'96: Proceedings of the Eurographics Workshop in Poitiers, France, August 31–September 1, 1996*, pp. 61–76. Springer, 1996.

[9] H. Gotoh and A. Khayyer. Current achievements and future perspectives for projection-based particle methods with applications in ocean engineering. *Journal of Ocean Engineering and Marine Energy*, 2:251–278, 2016.

[10] X. He, N. Liu, S. Li, H. Wang, and G. Wang. Local Poisson SPH For Viscous Incompressible Fluids. *Computer Graphics Forum*, 2012. doi: 10.1111/j.1467-8659.2012.03074.x

[11] X. He, N. Liu, G. Wang, F. Zhang, S. Li, S. Shao, and H. Wang. Staggered meshless solid-fluid coupling. *ACM Transactions on Graphics (TOG)*, 31(6):1–12, 2012.

[12] X. He, H. Wang, F. Zhang, H. Wang, G. Wang, and K. Zhou. Robust simulation of sparsely sampled thin features in sph-based free surface flows. *ACM Transactions on Graphics (TOG)*, 34(1):1–9, 2014.

[13] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner. Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, 2014. doi: 10.1109/TVCG.2013.105

[14] N. Kang and D. Sagong. Incompressible sph using the divergence-free condition. In *Computer graphics forum*, vol. 33, pp. 219–228. Wiley Online Library, 2014.

[15] M. H. Kee, K. Um, W. Jeong, and J. Han. Constrained projective dynamics: real-time simulation of deformable objects with energy-momentum conservation. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021.

[16] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *Eurographics 2019-Tutorials*, 2019.

[17] A. Latif. *Banach Contraction Principle and Its Generalizations*, pp. 33–64. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-01586-6_2

[18] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008. doi: 10.1109/TVCG.2008.37

[19] Z. Lu, X. He, Y. Guo, X. Liu, and H. Wang. Projective peridynamic modeling of hyperelastic membranes with contact. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[20] M. Macklin and M. Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[21] M. Macklin, M. Müller, and N. Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, pp. 49–54, 2016.

[22] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 154–159. Citeseer, 2003.

[23] K. Raveendran, C. Wojtan, and G. Turk. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 33–42, 2011.

[24] R. Schaback. Convergence analysis of the general gauss-newton algorithm. *Numerische Mathematik*, 46:281–309, 1985.

[25] H. Schechter and R. Bridson. Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[26] R. W. Siregar, M. Ramli, et al. Analysis local convergence of gauss-newton method. In *IOP Conference Series: Materials Science and Engineering*, vol. 300, p. 012044. IOP Publishing, 2018.

[27] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. vol. 28, pp. 1–6. ACM New York, NY, USA, 2009.

[28] T. Takahashi, Y. Dobashi, T. Nishita, and M. C. Lin. An efficient hybrid incompressible sph solver with interface handling for boundary conditions. In *Computer Graphics Forum*, vol. 37, pp. 313–324. Wiley Online Library, 2018.

[29] H. Wang and Y. Yang. Descent methods for elastic body simulation on the gpu. *ACM Transactions on Graphics (TOG)*, 35(6):212, 2016.

[30] Z. Wang, M. Tang, R. Tong, and D. Manocha. Tightccd: Efficient and robust continuous collision detection using tight error bounds. *Computer Graphics Forum*, 34(7):289–298, 2015. doi: 10.1111/cgf.12767

[31] M. Weiler, D. Koschier, and J. Bender. Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games*, pp. 79–84, 2016.

[32] L. Xu, X. He, W. Chen, S. Li, and G. Wang. Reformulating hyperelastic materials with peridynamic modeling. *Computer Graphics Forum*, 37(7):121–130, 2018. doi: 10.1111/cgf.13553

[33] S. Yang, X. He, H. Wang, S. Li, G. Wang, E. Wu, and K. Zhou. Enriching sph simulation by approximate capillary waves. In *Symposium on Computer Animation*, pp. 29–36, 2016.