Adapted SIMPLE Algorithm for Incompressible SPH Fluids with a Broad Range Viscosity

Shusen Liu, Xiaowei He, Wencheng Wang, Member, IEEE and Enhua Wu, Member, IEEE

Abstract—In simulating viscous incompressible SPH fluids, incompressibility and viscosity are typically solved in two separate stages. However, the interference between pressure and shear forces could cause the missing of behaviors that include preservation of sharp surface details and remarkable viscous behaviors such as buckling and rope coiling. To alleviate this problem, we introduce for the first time the semi-implicit method for pressure linked equations (SIMPLE) into SPH to solve incompressible fluids with a broad range viscosity. We propose to link incompressibility and viscosity solvers, and impose incompressibility and viscosity constraints iteratively to gradually remove the interference between pressure and shear forces. We will also discuss how to solve the particle deficiency problem for both incompressibility and viscosity solvers. Our method is stable at simulating incompressible fluids whose viscosity can range from zero to an extremely high value. Compared to state-of-the-art methods, our method not only produces realistic viscous behaviors, but is also better at preserving sharp surface details.

Index Terms—smoothed particle hydrodynamics (SPH), SIMPLE algorithm, fluid simulation, incompressibility, viscosity.

1 INTRODUCTION

Viscous fluids can be found everywhere in our daily life, such as honey, oil and ketchup, etc. Although smoothed particle hydrodynamics (SPH) has been widely used to model low viscous or nearly inviscid incompressible fluids [1][2], how to correctly model highly or even dynamic viscous incompressible fluids still faces challenges. Since explicit methods can hardly be extended for modeling highly viscous fluids due to the time step restriction [3][4], recent works prefer to solve fluid viscosity implicitly [5][6][7]. According to the analysis in [7], an ideal implicit viscosity solver should satisfy several requirements as listed in their work. However, in simulating highly viscous incompressible fluids, an additional requirement, i.e., incompressibility and viscosity constraints should not interfere [6][8], is also necessary to be fulfilled.

In traditional projection-based particle fluid solvers, the most common way to solve the governing equations of viscous incompressible fluids is to apply operator splitting, where the influence of pressure and shear forces are considered independently [9][10]. While operator splitting has the advantage of reducing computational cost and simplifying code implementation, Larionov et al. [8] pointed out splitting viscosity from projection will cause the loss of behaviors that depend on coupling between pressure and shear forces, such as preservation of sharp surface details and the rope coiling effect. This is because when both incompressibility and viscosity constraints are solved independently but only once, at least one of the two constraints will be damaged by the other at the end of each simulation step. Therefore, referring to fluid-solid coupling method in [11], a natural question is why not select an existing incompressibility solver and an existing viscosity solver, then iterate over them multiple times to alleviate the interference artifact? Unfortunately, according to our test in Figure 7, Figure 8 and Figure 13(c), where the DFSPH incompressibility solver [12] and the implicit viscosity solver [7] are selected, this strategy does not work well. The reason could be that if incompressibility and viscosity constraints are solved independently, the interference between pressure and shear forces could introduce drift errors to each other. Peer et al. [6][13] proposed an alternative way to solve the problem caused by the interference. However, the constructed velocity field suffers from non-physical velocity damping [7]. Our tests show that their method fails to reproduce the correct coiling effect, as shown in Figure 8.

Motivated by the semi-implicit method for pressure linked equations (SIMPLE) [14], we propose a novel iterative strategy for simulating viscous incompressible SPH fluids, with which the interference artifact can be addressed. Unlike the above mentioned simple strategy, the incompressibility and viscosity solvers in our method are linked. In solving the momentum equation for viscosity, aside from the velocity field, a term involving the pressure gradient is also integrated. This guarantees that the effect of pressure is properly considered when fluid viscosity is solved. In solving the momentum equation for incompressibility, the linked viscosity term will be discarded due to the convergence characteristics of SIMPLE algorithm.

Therefore, a traditional projection-based incompressibil-
Fig. 1: Newtonian fluids. Three bunnies with different viscosities are dropped onto a bar. This example shows that our method is stable at simulating incompressible fluids with a viscosity that can range from zero to an extremely high value.

Fig. 2: Non-Newtonian fluids. (a) A low viscosity ($\mu = 50 Pa \cdot s$) Newtonian bunny (left) and a shear thickening bunny (right) with equal initial viscosities are dropped onto the ground. The motion of the shear thickening bunny slows down upon impact, but then restores as its shear rate decreases; (b) A high viscosity ($\mu = 5 \times 10^5 Pa \cdot s$) Newtonian bunny (left) and a shear thinning bunny (right) with equal initial viscosities are dropped onto the ground. The motion of the shear thinning bunny accelerates upon impact.

Fig. 3: Three bunnies with different viscosities are dropped onto a table to demonstrate their interactions.

2 Related Works

Viscosity Models. Many forms of different viscosity models have been applied in computer graphics. We only review the most commonly used ones here. The first category uses the finite difference method to discretize the viscosity term \([15]\), which will be referred to as the local viscosity model considering its similarity to the Laplacian operator proposed by \([16]\). An explicit Euler integration method was used to impose viscosity in \([4, 17, 18]\) for the first time proposed an implicit method to solve viscosity. Compared to other viscosity models, the local model have a number of desirable features including being Galilean invariant, linear and angular momentum conservation. Besides, it does not suffer from the particle deficiency problem. The second category is labelled as the global viscosity model, which typically requires to take into account second-ring neighbors to calculate viscous forces. Since viscosity and incompressibility constraints can interfere, Peer et al. \([6]\) proposed to project the velocity gradient to a target that does not only encode the desired viscosity, but also preserves arbitrary velocity divergences. Takahashi et al. \([5]\) solved the exact viscosity implicitly to enable the usage of larger time steps and higher viscosities. Bender and Koschier \([12]\) also proposed an implicit viscosity solver by using a parallel Jacobi method to enforce a target strain rate. Peer and Teschner \([13]\) extended their earlier work to preserve the vorticity for highly viscous fluids. One disadvantage of the exact viscosity model is that artificial viscosity will be introduced to boundary due to the particle deficiency problem \([7]\). The last category is non-physical viscosity model, which is labelled as the XSPH artificial viscosity model. It was originally proposed by Monaghan \([19]\) to prevent

penetration for particles impinging on each other. Schechter and Bridson \([20]\) introduced this technique into computer graphics to stabilize the ghost SPH method. Macklin and Müller \([21]\) applied XSPH viscosity in a position-based fluid solver. Although Müller et al. \([22]\) proposed to use a slightly different formulation to model viscosity, their formulation still can be regarded as an XSPH artificial viscosity model as both models have the same physical meanings. Other works applying this method include \([23, 24, 25]\). Its wide usage attributes to the good performance in stability. However, the XSPH artificial viscosity model fails to reproduce certain physical features. For example, it cannot conserve angular momenta and introduces artificial viscosity even for rigid body rotation.

Incompressibility model. We categorize all incompressible solvers in SPH into two main classes. One aims to solve fluid incompressibility that targets a desired density. In the early work, incompressible fluids are assumed to be slightly compressible. Therefore, an equation of state (EOS) was applied to model the relationship between pressure and density \([4, 26, 27]\). However, since EOS-based solvers are subject to the limitation of small time steps, Solenthaler and Pajarola \([1]\) first proposed an iterative predictive-corrective scheme to enforce a constant density. Continuous efforts on accelerating the convergence rate were taken by several
later works. The other class aims to solve fluid incompressibility that targets the divergence-free condition. Cummins and Rudman first proposed an approximate projection method to enforce incompressibility. Motivated by the staggered-grid method, He et al. proposed a staggered particle method to simulate fluid-solid coupling. Bender and Koschier proposed to fulfill the divergence-free condition in a way similar to the iterative EOS-based solvers. Among all solvers, the velocity-constraint incompressibility model are preferable for simulating viscous fluids because undesired bulk viscosity could arise if the divergence-free condition is not fulfilled. For more discussions on incompressibility solvers, we refer to the review papers.

The linkage between solvers. In simulating highly viscous incompressible fluids with large time steps, the interference between pressure and viscosity solvers cannot be ignored. Under the Eulerian frame, this problem has been studied more thoroughly. However, only a few works have been done to address this issue under the Lagrangian frame. Peer et al. obtained the velocity constraint by projecting the velocity gradient to a target and reconstructing the velocity field. This approach is able to avoid the interference, yet a significant damping is introduced. Additionally, their approach cannot simulate low viscous fluids realistically. Aside from viscous incompressible fluids, the interference problem also exists in solid-fluid coupling or air-liquid coupling. Gissler et al. proposed to resolve the instability in fluid-rigid coupling or rigid-rigid coupling with SPH. Batté et al. reformulated solid-fluid coupling as a kinetic energy minimization problem, achieving a stable and accurate unified two-way coupling solver. Ajanjanya et al. adopted this similar idea, but proposed a monolithic solver to simulate bubbles. To realize Euler-Lagrange solid-fluid coupling, Robinson-Mosher et al. proposed a novel method that handles solid-fluid coupling in a fully implicit manner to obtain better stability. Robinson-Mosher et al. later extended their work to obtain a symmetric formulation of the coupled system, making it easier to solve.

3 Overview

The governing equations for a viscous incompressible fluid are expressed in the Lagrangian form as

\[ \rho \frac{Dv}{Dt} = -\nabla p + \nabla \cdot \tau + f, \quad (1) \]

\[ \nabla \cdot v = 0, \quad (2) \]

where \( v \) is velocity, \( \rho \) is density, \( p \) is pressure, \( \tau = \mu (\nabla v + (\nabla v)^T) \) is deviatoric stress tensor, and \( \mu \) is the dynamic viscosity coefficient. \( f \) is external force per unit volume. We assume \( \rho \) to be constant and \( \mu \) be varying in space. To better understand this interference issue between incompressibility and viscosity in operator splitting schemes, we take the prediction-correction scheme applied in for a demonstration. In the prediction step, an explicit integration without enforcing incompressibility is taken, i.e., only shear and body forces are used to update the velocity field. Equation should be fulfilled at the end of the prediction step, while the divergence-free condition is not fulfilled yet. Then, a pressure Poisson equation is solved to enforce the divergence-free condition in Equation. Unfortunately, after the velocity field is updated with the pressure force, Equation does not stand any longer due to the changes in the shear force and velocity.

Our purpose is to guarantee the final status of \( p \) and \( v \) at the end of each simulation step fulfill both governing equations. Despite the dependence of \( \mu \) on space, we make an assumption that \( \nabla \cdot (\mu (\nabla v + (\nabla v)^T)) = \mu \nabla^2 v + \mu \nabla \cdot (\nabla v^T) \). The second term just cancels out for incompressible fluids due to the relationship \( \mu \nabla \cdot (\nabla v^T) = \mu \nabla (\nabla \cdot v) = 0 \). After taking an implicit backward Euler method in time, the momentum and continuity equations are simplified as

\[ \rho \frac{v^{n+1} - v^*}{\delta t} = -\nabla p^{n+1} + \mu \nabla^2 v^{n+1}, \quad (3) \]

\[ \nabla^2 p^{n+1} = \frac{\rho}{\delta t} \nabla \cdot v^* + \nabla \cdot (\mu \nabla^2 v^{n+1}), \quad (4) \]

where \( v^* \) is the intermediate velocity after adding external forces, \( \delta t \) is time step size, superscript \( n+1 \) indicates the new value at the end of each simulation step. The difficulty in solving Equation and lies in that the unknown pressure \( p^{n+1} \) and velocity \( \nabla v^{n+1} \) are coupled for both equations. Even worse, the value of \( \nabla \cdot (\mu \nabla^2 v^{n+1}) \) could be quite sensitive to particle distributions due to the third order derivative of \( v \).

To solve the above-mentioned issue, a well-known SIMPLE algorithm was originally proposed in the finite volume method (FVM). Before we go into the details on how to extend the SIMPLE algorithm to SPH, let us review the fundamental principles of the SIMPLE method. To solve Equation and the SIMPLE algorithm starts with a guessed pressure field \( p^* \). The momentum equation for viscosity can then be solved as follows

\[ \rho \frac{v' - v^*}{\delta t} = -\nabla p^* + \mu \nabla^2 v'. \quad (5) \]

However, unless the correct pressure is given, the resulting velocity field \( v' \) will not satisfy the divergence-free condition in Equation, i.e., \( \nabla \cdot v' \neq 0 \). Therefore, we should find a way to improve \( p^* \), so that the velocity field can get closer to satisfying the divergence-free condition. Let us denote the
is expressed as both sides, we have

\[ \text{SIMPLE algorithm as follows:} \]

\[ V'' = V' - \frac{\delta t}{\rho} \nabla p', \quad (6) \]

where \( V'' \) satisfies \( \nabla \cdot V'' = 0 \). By taking the divergence of both sides, we have

\[ \nabla \cdot \frac{\delta t}{\rho} \nabla p' = \nabla \cdot V', \quad (7) \]

which is the same as [29].

Compared to Equation 4 we have dropped the viscosity term from the above equation. As suggested by Patankar [14], the omission of the viscosity term is absolutely unacceptable if the ultimate solution would not be the true solution of the discretized momentum and continuity equations. Fortunately, as we solve Equation 5 and 7 iteratively, both the velocity and pressure fields can progressively converge to their final solutions (refer to Figure 13(a)). That is to say, the global solution is actually uninfluenced by the approximation of dropping the viscosity term. Besides, above simplification is essential to extend the SIMPLE algorithm to SPH because the viscosity term will otherwise introduce a third-order differential term, and the smoothing length. To derive SPH formulations for gradient and Laplacian operators, there in fact exist a number of different ways [37, 38]. To discretize the pressure gradient, the symmetric repulsive pressure gradient model is mostly used because of its momentum conservation feature. However, this model is sensitive to the tensile instability problem [31]. In order to capture realistic stretching behaviors in viscous fluids (e.g., Figure 4), we choose to discretize the gradient operator with the Taylor-series consistent model

\[ \nabla_i \phi = - \sum_j V_j \phi_{ij} \nabla_i W_{ij}, \quad (9) \]

where \( W_{ij} \) denotes the smoothing function \( W(\|x_i - x_j\|, h) \), \( h \) is the smoothing length. To derive SPH formulations for gradient and Laplacian operators, there in fact exist a number of different ways [37, 38]. To discretize the pressure gradient, the symmetric repulsive pressure gradient model is mostly used because of its momentum conservation feature. However, this model is sensitive to the tensile instability problem [31]. In order to capture realistic stretching behaviors in viscous fluids (e.g., Figure 4), we choose to discretize the gradient operator with the Taylor-series consistent model

\[ \nabla_i \phi = - \sum_j V_j \phi_{ij} \nabla_i W_{ij}, \quad (9) \]

where \( \phi_{ij} \) is short for \( \phi_i - \phi_j, \nabla_i W_{ij} = \frac{x_i - x_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} \) and \( r_{ij} = \|x_i - x_j\| \). Compared to the symmetric repulsive model, the asymmetric one is more accurate as it recovers...
Fig. 7: Comparison of the buckling effect between our approach and state-of-the-art method [7] with different number of global iterations. DFSPH-IV means we use DFSPH [12] to solve incompressibility and the implicit method [7] to solve viscosity. For DFSPH-IV, the numbers and depths of the “wrinkles” of buckling effects change with increasing iterations, which shows that its physical viscosity changes drastically with global iterations [40], due to the divergence.

4.1 Viscosity

To solve the momentum equation 5 both \( \nabla p^* \) and \( \nabla^2 \psi' \) should be discretized first. If a particle’s location is far from free surface boundary, i.e., its support domain is not truncated by the boundary, we can easily get the discretized forms by inserting \( p^* \) and \( \psi' \) into Equation 2 and 10. Unfortunately, if the particle is near the free surface boundary, both terms suffer from the particle deficiency problem. To investigate this problem, we illustrate a one-dimensional linear function in Figure 5(a). The corresponding analytic formula and numerical calculation of its first and second order derivatives are demonstrated in Figure 5(b). We can notice the particle approximation no longer reproduces the correct first and second order derivatives for boundary particles.

To resolve the particle deficiency problem, one could introduce ghost air particles, as was done in [20]. However, creating ghost air particles requires more memory and computational resources.

Fortunately, a special treatment of free surface boundary conditions in fluids allows us to solve the particle deficiency problem efficiently with a semi-analytical approach [39, 41, 42]. Following their work, if liquid particles \( j^l \) and ghost air particles \( j^a \) are uniformly distributed, we can get the following relationship

\[
\sum_j V_j \nabla_i W_{ij} = \sum_j V_{jl} \nabla_i W_{ijl} + \sum_{j^a} V_{j^a} \nabla_i W_{ij^a} = \nabla 1 = 0,
\]

where \( \eta \) is a small value to prevent singularities and \( r_{ij} = x_i - x_j \).

Therefore, the pressure gradient can be discretized as

\[
\nabla_i p^* = \sum_j V_{jl} p^*_{jl} \nabla_i W_{ijl} + \sum_{j^a} V_{j^a} p^*_{j^a} \nabla_i W_{ij^a} - p^*_l \sum_j V_j \nabla_i W_{ij},
\]

from free surface boundary, i.e., its support domain is not truncated by the boundary, we can easily get the discretized first term. If a particle's location is far to zero, the pressure gradient can be derived as

\[
\nabla_i p^* = \begin{cases} \sum_j V_{jl} \left( p^*_{jl} - p^*_l \right) \nabla_i W_{ijl}, & i \notin B \\ \sum_{j^a} V_{j^a} p^*_{j^a} \nabla_i W_{ij^a}, & i \in B \end{cases}
\]

where \( B \) represents the boundary particles whose support domain intersect with the boundary as Figure 10. How to identify interior particles will be left to the next section.
Similarly, we discretize the Laplacian of the velocity field as
\[
\nabla_i^2 \mathbf{v}' = 2(d + 2) \sum_{j'} V_{j'} \frac{\mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2 + \eta^2} \nabla W_{ij}. \tag{14}
\]

The derivation of equation (14) can be found in [3] or [32]. Note that the contributions from virtual ghost air particles for both the pressure gradient and Laplacian of velocity have been semi-analytically eliminated. In fact, the formulation in Equation (14) is exactly the same as the one used in [7]. Yet our derivation has provided a better motivation for its effectiveness at reducing visual artifacts caused by the particle deficiency problem.

Based on the above discretization scheme, the momentum equation (5) can be divided into the following two steps
\[
\mathbf{v}^{**} = \mathbf{v}^* - \frac{\delta t}{\rho} \nabla p^*, \tag{15}
\]
\[
\mathbf{v}' = \mathbf{v}^{**} + \frac{\delta t \mu}{\rho} \nabla^2 \mathbf{v}'. \tag{16}
\]

Note Equation (16) is the same as the one solved in [7]. The calculated solution is then substituted into the pressure Poisson equation to enforce fluid incompressibility.

### 4.2 Incompressibility

To guarantee the convergence, the incompressibility solver should be properly derived from the pressure Poisson equation, and be able to impose the divergence-free condition on the velocity field (as discussed in Section 3). We therefore prefer to apply an approximate projection method to solve the fluid incompressibility.

Solving the pressure Poisson equation (7) involves discretizing both the gradient and Laplacian operators as well. Besides, free-surface boundary conditions, which affect the convergence in solving the pressure Poisson equation [43], should also be considered. In a projection based method, the pressure Poisson equation (7) is to be transformed into a linear system in the form of \( Ax = b \). The coefficient matrix \( A \) is symmetric, but cannot be guaranteed to be positive definite if no free boundary condition is imposed. For example, if a particle \( i \) has no neighbors, the corresponding diagonal entry of \( A \) is zero, indicating the coefficient matrix is singular. To resolve this problem, we introduce ghost air particles at the free surface boundary [42]. With enough ghost air particles, a particle at the free surface boundary should have full neighbors. By imposing the Dirichlet boundary condition, i.e., \( p_{ij}^a = 0 \) for ghost air particles, the coefficient matrix \( A \) is guaranteed to be positive definite. Therefore, typical iterative methods such as the conjugate gradient method can be applied to solve the linear system. In the following context, we will give more details on how to discretize Equation (7) and impose the free-surface boundary condition.

Integrating virtual ghost air particles, the velocity divergence can be easily derived as
\[
\nabla_i \cdot \mathbf{v}' = \sum_{j'} V_{j'} \left( \mathbf{v}_{ij}' - \mathbf{v}_i' \right) \cdot \nabla_i W_{ij}. \tag{17}
\]

However, discretizing the Laplacian of pressure will not be an easy task because we do not know how to calculate \( \nabla_i W_{ij} \) if ghost air particles are not created. Following [41], we apply the semi-analytical method to discretize the Laplacian of pressure. The neighbors of particle \( i \) could contain both particles \( j \) and ghost air particles \( j^a \). Since the pressure value of ghost air particles \( p_{ij}^a \) can be set to zero according to Dirichlet boundary conditions [44], we can discretize the Laplacian of pressure as
\[
\nabla_i^2 p_{ij}' = -\sum_{j'} a_{ij} p_{ij}' - \sum_{j^a} a_{ij} p_{ij}^a + \sum_{j'} b_{ij} p_{ij}', \tag{18}
\]
where \( a_{ij} = \frac{\mathbf{v}_i \cdot \nabla W_{ij}}{|\mathbf{r}_{ij}|^2 + \eta^2} \) is the coefficient. After rearrangement for different cases, the discretized Laplacian of pressure becomes
\[
\nabla_i^2 p_{ij}' = -\sum_{j'} a_{ij} p_{ij}' - \sum_{j^a} a_{ij} p_{ij}^a + \sum_{j'} b_{ij} p_{ij}'.
\]
**Fig. 12:** Rope coiling effect for a fluid with a viscosity of $\mu = 5000 \text{ Pa} \cdot \text{s}$. By taking more iterations for the outer loop, our method is able to preserve more sharp surface details without losing viscous behaviors. The radiiues of the rope coiling effects are almost the same, which means that the physical viscosities do not change with the iteration number \[45\].

$$(a) \text{iter}_{\text{max}} = 1 \quad (b) \text{iter}_{\text{max}} = 2 \quad (c) \text{iter}_{\text{max}} = 4 \quad (d) \text{iter}_{\text{max}} = 8 \quad (e) \text{iter}_{\text{max}} = 16$$

**Fig. 13:** Evaluation of the global convergence rate. (a) In simulating an incompressible fluid with a viscosity of $\mu = 5000 \text{ Pa} \cdot \text{s}$, our method shows that the relative errors of both the viscosity and incompressibility solvers converge to zero synchronously as the number of global iteration increases. (b) Comparison of the convergence rates of our method in simulating fluids with different viscosity coefficients, note the global convergence rate depends mostly on the viscosity coefficient. (c) Instead, if we use DFSPH to solve fluid incompressibility, the algorithm fails to converge to a global solution. In all above tests, the thresholds for both the viscous and incompressible solvers are set to $10^{-5}$.

Pressure is written as

$$\nabla^2 p' = \begin{cases} 
\sum_{j'} a_{ij'} (p'_{j'} - p_i'), & i \notin B \\
-A_0 p_i' + \sum_{j'} a_{ij'} p_{j'}', & i \in B
\end{cases}, \quad (19)$$

where $A_0$ is a predefined value of $A_i = \sum_{j'} a_{ij'}$ calculated for a particle with full neighbors at the beginning of simulation (e.g., by choosing an interior particle whose support domain is not clumped by boundary), as shown in Fig 10(a). We identify a fluid particle $i$ as a boundary particle only if $A_i < A_0$ (Fig 10(b)). Therefore, the boundary particle set $B$ is defined as

$$B = \{i \mid A_i < A_0\}. \quad (20)$$

Another issue in solving fluid incompressibility with Equation 7 is the particle density drifting problem \[45\], because the velocity divergence-free condition cannot correct cumulative density errors. Motivated by the ECS (error compensating source) used in \[47\] and \[48\], we solve this problem by adding the following term to the right-hand side of Equation 7 for particles with $\rho_i > \rho_0$, where $\rho_0$ represents the reference density and $\alpha$ is a positive constant. Equation 21 is similar to the source term of the projection method which corrects the density error \[9\]. When compression occurs, this term can generate repulsive forces to correct cumulative density errors. $\alpha$ should ensure that the value of this term is close to the right-hand side of Equation 7, otherwise, it may be ineffective or cause instability. Therefore, we typically set $\alpha = 10^6$ in our current implementation.

In calculating the pressure force, although it is common to clump particle pressures to zero \[2\], we retain the negative pressure to capture the fingering behavior in viscous fluids. More details on how resolve the tensile instability problem can be found in \[31, 48\].

To evaluate the stability of our incompressibility solver, a dambreak case is simulated with four different time steps. As shown in Figure 11, consistent behaviors can be noticed for all simulations.

**4.3 Iterative Solver**

To demonstrate the interference problem between incompressibility and viscosity solvers, we simulated the rope...
Algorithm 1 Adapted SIMPLE algorithm for viscous incompressible SPH fluids

1: while \( t < t_{\text{stop}} \) do
2: \( \delta t \leftarrow CFL(v^i_t) \);
3: for all fluid particle \( i \) do
4: \( v^i \leftarrow v^i_n + \delta t F^{ext,i} \);
5: \( x^i \leftarrow x^i_n + \delta t v^i_n \);
6: for all fluid particle \( i \) do
7: \( N_i \leftarrow \text{Find neighbors using } x^i_n \);
8: Detect boundary particles;
9: Set global iteration number \( \text{iter} = 0 \);
10: while \( \text{iter} < \text{iter}_{\text{max}} \) do
11: for all fluid particle \( i \) do
12: Compute velocity divergence \( \nabla_i \cdot v^i \);
13: Add \( \Lambda_i \) to \( \nabla_i \cdot v^i \);
14: while \( \epsilon^p > \epsilon^0_p \) do
15: Take one CG iteration to update \( p^i \);
16: Compute the relative error \( \epsilon^p \) for Equation 7
17: for all fluid particle \( i \) do
18: Compute \( p^i + = p^i \);
19: for all fluid particle \( i \) do
20: Compute pressure gradient \( \nabla_i p^* \);
21: while \( \epsilon^p > \epsilon^0_p \) do
22: Take one CG iteration for viscosity;
23: Compute the relative error \( \epsilon^p \) for Equation 9
24: for all fluid particle \( i \) do
25: Set new velocity \( v^i_{n+1} = v^i_t \);

effect with two different settings. In Figure 3(a), the viscosity solver is scheduled to run first. We can notice the incompressibility solver causes significant loss of viscous behaviors, such as the rope coiling. In contrast, when the incompressibility solver is taken first, the volume of fluid cannot be well preserved, as demonstrated at bottom of Figure 3(b). To solve the interference problem, we propose to improve the original SIMPLE algorithm by solving incompressibility first. In case the divergence-free condition is violated by the viscosity solver, we take more overall iterations to alleviate the interference problem.

The detailed algorithm is demonstrated in Algorithm 1. It contains two loops. The two inner loops are used to solve Equation 5 and 19 independently with an iterative solver. Figure 4(a) and 4(b) demonstrate the convergence rates of two commonly used iterative solvers, including both the Jacobi method and the conjugate gradient method. We can notice faster convergence rate of the conjugate gradient compared to the Jacobi method in solving both fluid incompressibility and viscosity. The outer loop is also taken iteratively, with a purpose to diminish the interference between pressure and viscosity.

Figure 12 demonstrates an intuitive comparison on the effect of increasing the global iteration number, where the iteration number is increased from 1 to 16. As the global iteration number increases, we can notice a better property of shape preserving without losing viscous behaviors.

To demonstrate whether our method converges to a global solution, Figure 13(a) plots the convergence curves of relative errors for both the incompressibility and viscosity solvers. Note the pressure and velocity fields are largely affected between each other at the beginning of the global iteration. However, as we take more global iterations, relative errors for both fields are gradually removed. In other words, the interference between pressure and viscosity can be alleviated with our method. Further experiments show that the global convergence rate mostly depends on the viscosity coefficient, as demonstrated in Figure 13(b). Both the time step size and the particle sampling distance only have a minor effect on the global convergence rate, as illustrated in Figure 14. Unfortunately, if we use DFSPH to solve fluid incompressibility, the algorithm fails to converge to a global solution, as shown in Figure 13(c). That is to say, the idea of coupling an arbitrary incompressibility and a viscosity solver may not be a serious solution to simulate viscous incompressible fluids.

To further evaluate the dependence of viscous behaviors on the SIMPLE iteration number, Figure 15 demonstrates four materials with different viscosities. As the iteration number is increased, it can be noted that patterns of all four materials tend to be stable. In other words, if a sufficient iteration number is taken, real physical behaviors for all viscous fluids should be reached. However, in the practical implementation, we suggest to only take a small fixed number of iterations for efficiency. According to the comparison, 10 iterations will be enough to achieve desirable visual effects for most cases. However, for fluids with a high viscosity, e.g., \( \mu > 10^4 \text{Pa} \cdot \text{s} \), more than 10 iterations can better preserve the shapes.

5 Results and Discussion
In this section, we simulate various fluids, including both Newtonian and non-Newtonian ones. All implementations were performed on an Intel CPU (i7-8700K 3.70GHz) with time-consuming parts (neighbor-list searching, incompressibility solver, viscosity solver and velocity updating etc.), parallelized on an NVIDIA GPU (Geforce RTX2080). Surface tension is based on the method of He et al. [42]. Surface
Kinematic Viscosity: 0.5 \text{m}^2/\text{s} (\mu = 500 \text{Pa} \cdot \text{s})

Kinematic Viscosity: 1 \text{m}^2/\text{s} (\mu = 1000 \text{Pa} \cdot \text{s})

Kinematic Viscosity: 5 \text{m}^2/\text{s} (\mu = 5000 \text{Pa} \cdot \text{s})

Kinematic Viscosity: 10 \text{m}^2/\text{s} (\mu = 10000 \text{Pa} \cdot \text{s})

Fig. 15: Evaluation of the dependence of viscous behaviors on the SIMPLE iteration number. The figure shows a snapshot of the viscous fluids 2.5 seconds after they first hit the ground. In this scenario, each fluid block consists of 8k particles and the corresponding SIMPLE iteration number is displayed on the board. A free-slip boundary condition is imposed on the ground. The surface tension is neglected to make surface details more obvious. Stable fluid patterns are reached as the iteration number is increased.

Comparison to other methods. Figure 7 shows a comparison of the buckling effect between our method and state-of-the-art method [2]. We can notice that when only taking one global iteration, the surface details of both fluids can be quickly blurred. Then, as the global iteration number increases, significant improvement on preserving surface details can be found with our method. However, if we solve incompressibility with DFSPH and viscosity with an implicit viscosity (IV) solver multiple times, a side effect can be observed. The material becomes stiffer with more global iterations (please count the wrinkle number). It indicates the combination of DFSPH and the viscosity solver does not converge to a global solution. The cause of this problem is that if incompressibility and viscosity solvers are not linked properly (as discussed in section 3), the final solution could deviate from the global solution. In contrast, our method does not suffer from this problem due to the linkage between pressure and shear forces.

Since earlier works [6, 13] have also been denoted to remove the interference between pressure and viscosity, we provide a more thorough comparison in Figure 8. As shown in Figure 8(a) and Figure 8(b), both the methods of Peer et al. 2015 [6] and Peer et al. 2016 [13] fail to capture the coiling effect correctly. This problem is caused by the missing of rotational motion due to the damping introduced by their reconstruction of the velocity field [7]. With Weiler et al. [7]'s method, the correct rope coiling effect is visible in Figure 8(c). However, if we want to enrich surface details by taking 5 SIMPLE iterations, Figure 8(d) shows the correct coiling effect is damaged. In other words, the DFSPH+IV strategy does not guarantee the convergence under multiple SIMPLE iterations. In contrast, Figure 8(e) and Figure 8(f) show that our method guarantees a consistent behavior with more surface details when more iterations are taken.

Newtonian fluids. A Newtonian fluid is a fluid whose viscous stress is linearly proportional to the local strain rate, i.e., \( \mu \) is a constant. To show that our method is stable at simulating Newtonian fluids with any viscosity, we drop a bunny of liquid with three different viscosities on a bar, as shown in Figure 1. The corresponding viscosity coefficients are set to 0 \text{Pa} \cdot \text{s}, 300 \text{Pa} \cdot \text{s} and an extremely high value of 5 \times 10^7 \text{Pa} \cdot \text{s}, respectively. Note for the non-viscous fluid, only one outer loop iteration is needed because no viscous forces exist to violate fluid incompressibility.

Non-Newtonian fluids. In non-Newtonian fluids, viscosity can change with varying shear rates. We implement the commonly used Cross model [50] to simulate non-Newtonian fluids. Its dynamic viscosity is defined as the
To guarantee momentum conservation, the viscosity term defined as
\[ \mu = \mu_\infty + \frac{\mu_0 - \mu_\infty}{1 + (k\gamma)^n}, \]
where \(\mu_0\) and \(\mu_\infty\) are two limiting values of viscosity at very low and high shear rates, \(k\) controls the shear rate and \(n\) controls the smoothness of transition, \(\gamma\) is the shear rate defined as
\[ \dot{\gamma} = \sqrt{0.5 \cdot \text{trace}(\mathbf{D})^2}, \quad \mathbf{D} = \nabla \mathbf{v} + (\nabla \mathbf{v})^T. \]  
To guarantee momentum conservation, the viscosity term for non-Newtonian fluids should be corrected as follows
\[ \mu \nabla^2 \mathbf{v}' = 2(d + 2) \sum_{ij} V^i_j \left[ \frac{\mu_i + \mu_j}{2} \frac{\mathbf{v}'_i \cdot \mathbf{r}_{ij} - \mathbf{r}_{ij} \cdot \mathbf{v}'_j}{\| \mathbf{r}_{ij} \|^2 + \eta^2} \right] \nabla W_{ij}. \]  
Figure 2 shows both shear thinning (\(\mu_0 = 50 \, Pa \cdot s, \mu_\infty = 5 \times 10^5 \, Pa \cdot s, n = -6, k = 10\)) and shear thickening (\(\mu_0 = 50 \, Pa \cdot s, \mu_\infty = 5 \times 10^5 \, Pa \cdot s, n = 6, k = 1\)) behaviors simulated with the Cross viscosity model. For comparison, a low viscosity as well as a high viscosity Newtonian fluid are also demonstrated. In this example, the two non-Newtonian fluids show dramatic change of the viscosity during their impact. The shear thickening fluid seems to be quickly hardened during the impact due to the increased viscosity and then softened gradually as the deformation slows down. In contrast, the shear thinning fluid crumbles as a result of material softening during the impact.

To further demonstrate the distinctive behaviors between Newtonian and non-Newtonian fluids, we setup another experiment for comparison, i.e., by dropping rigid objects into Newtonian fluids and non-Newtonian fluids respectively, as shown in Figure 19. Both the low viscosity Newtonian fluid and shear thickening fluid are simulated with an initial viscosity of \(\mu = 5 \, Pa \cdot s\). However, as the cube touches the liquid surface, the rapid change of deformation in the shear thickening fluid helps to increase shear forces, therefore slows the motion of the cube quickly. In contrast, when the simulation of a high viscosity Newtonian fluid and a shear-thickening fluid are started with an initial viscosity of \(\mu = 5 \times 10^5 \, Pa \cdot s\), the shear thinning fluid softens as the cube touch the liquid surface due to the rapid change of deformation. Note that for a shear thinning fluid, the value of \(\dot{\gamma}\) could occasionally be 0. To avoid being divided by zero, we simply neglect the last term of Equation 22 when \(\dot{\gamma}\) is smaller than a predefined threshold.

**Cream on Cookies.** To demonstrate the effectiveness of our method in removing the interference between pressure and shear forces, Figure 16 demonstrates a comparison of two simulations where on the left only one SIMPLE iteration is taken to simulate the cream while 5 iterations are taken on the right. According to the comparison, we can notice more surface details with more iterations. Besides, taking more

**TABLE 1:** Statistics and timings per timestep of our examples. \(i_{\text{iter max}}\) represents the number of SIMPLE iteration, \(t_p\) represents the average computational cost for solving Equation 7, \(t_v\) represents the average computational cost for solving Equation 5, \(t_{\text{total}}\) represents the average computational cost per time step.

<table>
<thead>
<tr>
<th>Name</th>
<th>total particles</th>
<th>(\mu (Pa \cdot s))</th>
<th>surface tension</th>
<th>(\delta t [ms])</th>
<th>(i_{\text{iter max}})</th>
<th>(t_p [ms])</th>
<th>(t_v [ms])</th>
<th>(t_{\text{total}} [ms])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 12(a)</td>
<td>22k</td>
<td>5000</td>
<td>0.0005</td>
<td>1.0</td>
<td>1</td>
<td>4.1</td>
<td>5.2</td>
<td>5.3</td>
</tr>
<tr>
<td>Figure 12(b)</td>
<td>22k</td>
<td>5000</td>
<td>0.0005</td>
<td>1.0</td>
<td>2</td>
<td>4.1</td>
<td>5.2</td>
<td>4.5</td>
</tr>
<tr>
<td>Figure 12(c)</td>
<td>22k</td>
<td>5000</td>
<td>0.0005</td>
<td>1.0</td>
<td>1</td>
<td>4.1</td>
<td>5.2</td>
<td>6.4</td>
</tr>
<tr>
<td>Figure 12(d)</td>
<td>22k</td>
<td>5000</td>
<td>0.0005</td>
<td>1.0</td>
<td>8</td>
<td>4.1</td>
<td>5.2</td>
<td>0.11</td>
</tr>
<tr>
<td>Figure 12(e)</td>
<td>22k</td>
<td>5000</td>
<td>0.0005</td>
<td>1.0</td>
<td>10</td>
<td>4.1</td>
<td>5.2</td>
<td>0.19</td>
</tr>
<tr>
<td>Figure 16(left)</td>
<td>10k</td>
<td>5 \times 10^6</td>
<td>0</td>
<td>1.0</td>
<td>1</td>
<td>16</td>
<td>37</td>
<td>1.3</td>
</tr>
<tr>
<td>Figure 16(right)</td>
<td>10k</td>
<td>5 \times 10^6</td>
<td>0</td>
<td>1.0</td>
<td>5</td>
<td>16</td>
<td>37</td>
<td>0.34</td>
</tr>
<tr>
<td>Figure 12(a)</td>
<td>1068k</td>
<td>5 \times 10^4, 5 \times 10^5</td>
<td>0, 0.0005</td>
<td>1.0</td>
<td>5</td>
<td>90</td>
<td>203</td>
<td>24</td>
</tr>
<tr>
<td>Figure 17(b)</td>
<td>54k</td>
<td>5 \times 10^7</td>
<td>0</td>
<td>1.0</td>
<td>5</td>
<td>71</td>
<td>102</td>
<td>0.15</td>
</tr>
<tr>
<td>Figure 18</td>
<td>979k</td>
<td>5 \times 10^8</td>
<td>1.2~1.9(CFL)</td>
<td>4</td>
<td>82</td>
<td>171</td>
<td>1.9</td>
<td>1.9</td>
</tr>
<tr>
<td>Figure 19</td>
<td>103k</td>
<td>dynamic</td>
<td>0</td>
<td>1.0</td>
<td>2</td>
<td>17</td>
<td>41</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Fig. 16: Creams on cookies. Compared to a one-iteration strategy on the left, we can notice more surface details on the right when five SIMPLE iterations are taken.
iterations also helps preserve the final shape for a longer time.

More demonstrations. In Figure 9, three bunnies with different viscosities are dropped onto a table. Different viscous behaviors as well as interactions between them can be noticed. In Figure 18, five armadillos are dropped sequentially onto the ground. This example shows that our method preserves shapes and surface details well for a high viscosity fluid ($\mu = 5 \times 10^5 \text{Pa} \cdot \text{s}$). Figure 18 shows an armadillo consisting of 462k particles dropped onto a bench. By including surface tension forces, our method is able to reproduce remarkable viscous behaviors, such as rope coiling and buckling, as well as other realistic small-scale thin features, such as filaments and sheets. In Figure 17(b), a real scenario of making ice cream is simulated to demonstrate the rope coiling effect ($\mu = 5 \times 10^4 \text{Pa} \cdot \text{s}$). Due to our SIMPLE method, the ice cream is able to preserve the shape well. In Figure 17(a), four different Newtonian fluids were simulated by our method. The maple cream on the triangle cake is simulated with a viscosity of $300 \text{Pa} \cdot \text{s}$; chocolate cream with $\mu = 200 \text{Pa} \cdot \text{s}$; faint yellow cream: $\mu = 5 \times 10^4 \text{Pa} \cdot \text{s}$; white cream: $\mu = 5 \times 10^5 \text{Pa} \cdot \text{s}$.) are simulated. (b) Ice-cream. Our method captures the rope coiling effect ($\mu = 5 \times 10^4 \text{Pa} \cdot \text{s}$), (c) A shear thickening fluid with $\mu_0 = 5 \text{Pa} \cdot \text{s}, \mu_\infty = 5 \times 10^5 \text{Pa} \cdot \text{s}, n = 6$ and $k = 1$; (d) A high viscosity Newtonian fluid with $\mu = 5 \times 10^5 \text{Pa} \cdot \text{s}$; (e) A shear thinning fluid with $\mu_0 = 5 \text{Pa} \cdot \text{s}, \mu_\infty = 5 \times 10^5 \text{Pa} \cdot \text{s}, n = -10$ and $k = 50$.

6 Conclusions

In this paper, we presented a SIMPLE algorithm adapted for SPH in simulating viscous incompressible fluids. The main purpose is to alleviate the interference between pressure and shear forces, especially for highly viscous incompressible fluids. We have demonstrated a variety of different examples to show the effectiveness of our method in alleviating the interference artifact. Our method is able to model remarkable viscous behaviors, such as coiling and buckling, with a good property of preserving sharp surface details. Besides, our method is stable at simulating incompressible Newtonian and non-Newtonian fluids with any given viscosity.

For our further work, we will first consider how to improve the performance. Besides, we will consider extending our method to simulate more kinds of viscous incompressible fluids, especially for those with thin features. Finally, we will investigate how to alleviate the interference between viscoelasticity and incompressibility for viscoelastic incompressible fluids.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments. The project was supported by the National Key R&D Program of China (No.2017YFB1002701),
the National Natural Science Foundation of China (No.62072446, No.618707657, No.61672502, No.62072449, No.61632003), Youth Innovation Promotion Association, CAS (No.2019109), the Strategic Priority Research Program, CAS(No.19080102) and University of Macau Grant (MYRG2019-00006-FST).

References


