Final Report for Differentiable SPH

Yuqi Ren*
Institute for Interdisciplinary Information Sciences
Tsinghua University
Beijing 100084, China
ryq22@mails.tsinghua.edu.cn

Supervisor: Hao Su
University of California San Diego
haosu@ucsd.edu

Abstract

Differentiable Smoothed Particle Hydrodynamics (SPH) is a promising approach for gradient-based fluid optimization. To explore its capabilities, we implemented a complete end-to-end pipeline featuring a differentiable SPH simulator and renderer, and we verified its effectiveness on short-horizon fluid control tasks. However, when applied to long-horizon problems, we observed significant gradient errors and numerical instability, rendering optimization infeasible. This prompted a rigorous investigation into the source of this error accumulation. Through a comparative analysis of gradients derived from automatic differentiation, high-precision analytical methods, and finite differences, we identify the Hessian of the SPH smoothing kernel within the pressure force derivative as the primary cause.

1 Introduction

Differentiable simulation has become a cornerstone for solving complex optimization and inverse problems in fields ranging from robotics to computer graphics. Among various simulation techniques, Smoothed Particle Hydrodynamics (SPH) is a widely-used method for modeling fluids. The advent of differentiable SPH simulators allows for the direct computation of simulation gradients, unlocking powerful gradient-based optimization for tasks like fluid control, parameter estimation, and scene design.

However, a critical challenge severely hinders the broader adoption of this technology. While the forward (simulation) pass of SPH is often stable, the backward pass (gradient computation) suffers from exploding gradient error, especially over long simulation horizons. This phenomenon introduces significant noise, rendering the gradients unreliable for optimization. Consequently, integrating differentiable SPH with other components, such as a differentiable renderer for end-to-end inverse rendering, becomes practically infeasible.

In this research, we develop a complete end-to-end pipeline that integrates a high-performance, autograd-based SPH simulator with a custom differentiable renderer, demonstrating its effectiveness on simple and short-horizon fluid control tasks. Furthermore, we seek to understand why the gradients behaves so differently from the forward simulation and to pinpoint the specific components of the gradient calculation that lead to instability.

*Undergraduate student. Email: ryq22@mails.tsinghua.edu.cn

2 Related Work

2.1 SPH-based Simulation

Smoothed Particle Hydrodynamics (SPH) is a mesh-free, Lagrangian method originally developed for astrophysical simulations [Lucy, 1977, Gingold and Monaghan, 1977]. Unlike grid-based Eulerian methods, SPH discretizes the fluid continuum into a set of particles, each carrying physical properties such as mass, velocity, and pressure. These properties are interpolated from neighboring particles using a smoothing kernel function. Due to its natural ability to handle complex free surfaces and deformable boundaries, SPH has become a popular choice for fluid simulation in computer graphics.

A primary challenge in SPH is enforcing fluid incompressibility. Early methods approached this by computing particle pressures using an Equation of State (EOS). Standard SPH employs a low-stiffness EOS [Desbrun and Gascuel, 1996], while Weakly Compressible SPH (WCSPH) utilizes a much stiffer EOS to strictly limit density fluctuations, typically to under 1% [Monaghan, 2005, Becker and Teschner, 2007, Becker et al., 2009]. However, this high stiffness parameter, which is often difficult to tune, imposes restrictive time step constraints due to the Courant-Friedrichs-Lewy (CFL) condition.

An alternative approach enforces incompressibility is projecting the velocity field onto a divergence-free state, similar to Eulerian methods (e.g. [Enright et al., 2002]). This led to the family of Incompressible SPH (ISPH) methods. While ISPH methods achieve low density fluctuations, they involve formulating and solving a complex pressure Poisson equation. To balance computational cost and stability, [Solenthaler and Pajarola, 2009] introduced Predictive-Corrective SPH (PCISPH), an iterative method that controls density errors within a user-defined tolerance. PCISPH successfully combines the low computational cost of WCSPH with the large time steps characteristic of ISPH.

However, a key limitation of PCISPH is that it only enforces incompressibility at the density level. Subsequent methods were developed to address this shortcoming, notably Local Poisson SPH (LP-SPH) [He et al., 2012] and Divergence-Free SPH (DFSPH) [Bender and Koschier, 2015]. By enforcing a divergence-free velocity field in addition to maintaining density constancy, these approaches satisfy a stricter incompressibility constraint at both the density and velocity levels.

2.2 Differentiable Simulation

Differentiable simulation has recently emerged as a pivotal concept in computer graphics and machine learning. While significant progress has been made in developing differentiable simulators for rigid-body dynamics [Freeman et al., 2021, Geilinger et al., 2020, Qiao et al., 2021, Xu et al., 2021], soft-body dynamics [Du et al., 2021, Hahn et al., 2019, Hu et al., 2019b, Murthy et al., 2020], cloth simulation [Li et al., 2022b, Liang et al., 2019], and Eulerian fluid dynamics [Du et al., 2020, Holl et al., 2020, Li et al., 2022a, McNamara et al., 2004, Schenck and Fox, 2018, Li et al., 2024], their Lagrangian fluid counterparts remain comparatively underexplored.Furthermore, it is worth noting that some methods achieve differentiability by using neural networks to predict the solver's output [Holl et al., 2020, Schenck and Fox, 2018]. However, this black-box approach is typically slower and less generalizable than simulators that are made differentiable by computing analytical gradients directly through the physical equations [Li et al., 2024]. In the Lagrangian fluid domain, the work on DiffRF by [Li et al., 2023], while proposing a differentiable SPH-based simulator with localized gradient computation scheme, primarily focuses on the derivatives of fluid-rigid coupling forces. This scope does not require tracking the evolution of fluid state derivatives over time, which marks a key distinction from the aforementioned Eulerian differentiable fluid simulators that propagate gradients through the entire state.

There are several typical methods for differentiating a simulator, including finite differences, auto-differentiation, and the manual derivation of analytical gradients, as well as combinations thereof. The complex-step finite difference has recently seen increasing attention [Luo et al., 2019, Shen et al., 2021] for overcoming the drawbacks in the real number domain such as subtractive cancellation issues. However, its computational overhead becomes significant when the number of parameters is large. Auto-differentiation tools [Hu et al., 2019a,b] save the labor of manual derivation by storing the computational graph on a "tape" during the forward pass and then backpropagating through the graph to obtain gradients. To obtain the analytical gradient, [McNamara et al., 2004] develped the

adjoint method to efficiently compute the gradient of a fluid system for keyframe fluid animation control. Another mothod to differentiate a simulator is symbolic differentiation, which computes the derivative of a function by applying the chain rule repeatedly. However, the size of the symbolic gradients grow rapidly with the complexity of the function. Therefore, symbolic differentiation is typically only used when the dynamics are simple, or for a few select components of the simulator [Geilinger et al., 2020, Hu et al., 2019b, Werling et al., 2021].

3 Contributions

Despite the extensive body of research on SPH-based simulation, work in the specific area of differentiable SPH fluid simulation remains notably scarce. This research addresses this significant gap by providing a comprehensive analysis of gradient instability in differentiable SPH and presenting a functional end-to-end optimization pipeline. Our main contributions are:

- **Identification of Gradient Instability Source:** Through a rigorous ablation study and high-precision numerical experiments, we pinpoint the primary source of exploding gradients in analytical differentiable SPH. We demonstrate conclusively that the instability arises from the Hessian of the smoothing kernel, $\nabla(\nabla W)$, which is a component of the pressure force derivative.
- **End-to-End Differentiable Pipeline:** We develop and validate a complete, end-to-end differentiable pipeline for fluid control tasks. This system integrates a high-performance, autograd-based SPH simulator (NVIDIA Warp) with a custom-built differentiable renderer, enabling the optimization of physical parameters directly from image-space loss functions.
- **Comparative Gradient Analysis Framework:** We establish a robust framework for analyzing gradient error by comparing results from three distinct methods: high-performance automatic differentiation (NVIDIA Warp), high-precision analytical differentiation (C++/Eigen), and finite differences as a ground truth. This multi-faceted approach allows for a nuanced understanding of the trade-offs between performance, precision, and stability in different gradient computation schemes.

4 Methodology

The motion of an incompressible fluid is governed by the Navier-Stokes (NS) equations, which describe the conservation of momentum and mass:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nabla \cdot (\nu \nabla \mathbf{v}) + \frac{\mathbf{f}_{ext}}{\rho} \tag{1}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{2}$$

where $\frac{D}{Dt}$ is the material derivative, $\mathbf{v}$ is the velocity, ρ is the fluid density, p is the pressure, ν is the kinematic viscosity, and $\mathbf{f}_{ext}$ denotes external forces, such as gravity. To solve these equations numerically, we employ the PCISPH scheme to ensure stable and efficient enforcement of the incompressibility constraint.

4.1 Time Integrating

We utilize a semi-implicit Euler method for time integration, which offers a balance between stability and computational cost. The particle velocities and positions are updated as follows:

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t \tag{3}$$

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t + \Delta t)\Delta t \tag{4}$$

where $a(t)$ is the acceleration at time t and Δt is the time step.

4.2 PCISPH

The core of the PCISPH method is an iterative prediction-correction loop that adjusts particle pressures to minimize density fluctuations, thereby satisfying the incompressibility condition. The overall procedure is detailed in Algorithm 1, and the key calculations within this framework are described in the following sections.

4.2.1 Density Estimation

The density ρ_i of a particle i at location $\mathbf{x}_i$ is computed by summing up the weighted contributions of the neighboring particles j:

$$\rho_i = \sum_j m_j W(\mathbf{x}_{ij}, h) = \sum_j m_j W_{ij} \tag{5}$$

where m_j is the mass of neighboring particle j, W is the weighting kernel function with a smoothing length h, and $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

4.2.2 Pressure Correction

In the PCISPH method, the pressure correction term p_i for particle i is computed to counteract density errors:

$$p_i = \delta_i \rho^*_{err,i} \tag{6}$$

$$\delta_i = \frac{1}{\alpha_i \left(\sum_j \nabla W_{ij} \cdot \sum_j W_{ij} + \sum_j \left(\nabla W_{ij} \cdot W_{ij} \right) \right)} \tag{7}$$

where $\alpha_i = \frac{2m_i^2 \Delta t^2}{{\rho_0}_i^2}$, ρ_{0i} is the reference density, $\rho^*_{err,i} = \rho^*_i - \rho_{0i}$ is the predicted density error, and δ_i is a scaling factor which is evaluated for a prototype particle with a filled neighborhood.

4.2.3 Force Discretization

The only external force considered in our simulations is gravity:

$$\mathbf{f}_{ext,i} = m_i \mathbf{g} \tag{8}$$

The pressure force is derived from the pressure gradient term in the Navier-Stokes equation and is discretized symmetrically to conserve momentum:

$$\mathbf{f}_{pi} = -\sum_j m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} \tag{9}$$

The viscous force is modeled as:

$$\mathbf{f}_{vi} = 2(d+2)\nu \sum_j \frac{m_i m_j}{\rho_i \rho_j} \frac{\mathbf{v}_{ij} \cdot \mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \nabla W_{ij} \tag{10}$$

where d is the number of spatial dimensions and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$.

4.3 Differentiable PCISPH

To enable gradient-based optimization of physical parameters, we formulate a differentiable version of the PCISPH solver. This involves deriving the analytical gradients of all relevant quantities with respect to a general optimization parameter vector $\mathbf{x}$. The complete algorithm for the differentiable PCISPH solver is illustrated in Algorithm 2.

Algorithm 1 PCISPH

for all i **do**
  find neighborhoods $N_i(t)$
for all i **do**
  compute forces $\mathbf{f}_{vi}(t)$, $\mathbf{f}_{ext,i}(t)$
  initialize pressure $p_i(t) = 0$
  initialize pressure force $\mathbf{f}_{pi}(t) = 0$
while $\left(\rho^*_{err,avg}(t + \Delta t)\right) < \eta \;||\; (iter < minIterations)$ **do**
  for all i **do**
    predict velocity $\mathbf{v}^*_i(t + \Delta t)$
    predict position $\mathbf{x}^*_i(t + \Delta t)$
  for all i **do**
    predict density $\rho^*_i(t + \Delta t)$
    predict density variation $\rho^*_{err,i}(t + \Delta t)$
    compute pressure $p_i(t)$
  for all i **do**
    compute pressure force $\mathbf{f}_{pi}(t)$
for all i **do**
  compute new velocity $\mathbf{v}_i(t + \Delta t)$
  compute new position $\mathbf{x}_i(t + \Delta t)$

4.3.1 Differentiate Pressure

The derivatives of density and pressure with respect to s are found by applying the chain rule:

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}\mathbf{s}} = \sum_j m_j \nabla W_{ij}^T \frac{\mathrm{d}\mathbf{x}_{ij}}{\mathrm{d}\mathbf{s}} \tag{11}$$

$$\frac{\mathrm{d}p_i}{\mathrm{d}\mathbf{s}} = \delta_i \frac{\mathrm{d}\rho_i}{\mathrm{d}\mathbf{s}} = \delta_i \sum_j m_j \nabla W_{ij}^T \frac{\mathrm{d}\mathbf{x}_{ij}}{\mathrm{d}\mathbf{s}} \tag{12}$$

4.3.2 Differentiate Force

The derivatives of the pressure and viscous forces are derived accordingly.

$$\begin{aligned}\frac{\mathrm{d}\mathbf{f}_{pi}}{\mathrm{d}\mathbf{s}} =& -\sum_j m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla(\nabla W)^T \frac{\mathrm{d}\mathbf{x}_{ij}}{\mathrm{d}\mathbf{s}} \\ &+ \sum_j m_i m_j \left(\frac{2p_i - \delta_i\rho_i}{\rho_i^3}\frac{\mathrm{d}\rho_i}{\mathrm{d}\mathbf{s}} + \frac{2p_j - \delta_j\rho_j}{\rho_j^3}\frac{\mathrm{d}\rho_j}{\mathrm{d}\mathbf{s}}\right) \nabla W_{ij}\end{aligned} \tag{13}$$

For convenience, we define a term $\gamma = 2(d + 2\mu)$. The derivative of the viscous force is then given by:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{f}_{vi}}{\mathrm{d}\mathbf{s}} =& -\gamma \sum_j \frac{m_i m_j}{\rho_i \rho_j}\left(\frac{1}{\rho_i}\frac{\mathrm{d}\rho_i}{\mathrm{d}\mathbf{s}} + \frac{1}{\rho_j}\frac{\mathrm{d}\rho_j}{\mathrm{d}\mathbf{s}}\right)\frac{\mathbf{v}_{ij}\cdot\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2}\nabla W_{ij} \\ &+ \gamma \sum_j \frac{m_i m_j}{\rho_i \rho_j}\frac{\mathbf{v}_{ij}\cdot\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2}\nabla(\nabla W_{ij})^T \frac{\mathrm{d}\mathbf{x}_{ij}}{\mathrm{d}\mathbf{s}} \\ &+ \gamma \sum_j \frac{m_i m_j}{\rho_i \rho_j}\left(\frac{\frac{\mathrm{d}\mathbf{v}_{ij}}{\mathrm{d}\mathbf{s}}\cdot\mathbf{x}_{ij} + \mathbf{v}_{ij}\cdot\frac{\mathrm{d}\mathbf{x}_{ij}}{\mathrm{d}\mathbf{s}}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} - 2\frac{(\mathbf{v}_{ij}\cdot\mathbf{x}_{ij})\left(\mathbf{x}_{ij}\cdot\frac{\mathrm{d}\mathbf{x}_{ij}}{\mathrm{d}\mathbf{s}}\right)}{\left(\|\mathbf{x}_{ij}\|^2 + 0.01h^2\right)^2}\right)\nabla W_{ij}\end{aligned} \tag{14}$$

Algorithm 2 Differentiable PCISPH

for all i **do**
  find neighborhoods $N_i(t)$
for all i **do**
  compute forces $\mathbf{f}_{vi}(t)$, $\mathbf{f}_{ext,i}(t)$
  compute forces' gradients $\frac{\mathrm{d}\mathbf{f}_{vi}(t)}{\mathrm{d}\mathbf{s}}$, $\frac{\mathrm{d}\mathbf{f}_{ext,i}(t)}{\mathrm{d}\mathbf{s}}$
  initialize pressure $p_i(t) = 0$
  initialize pressure force $\mathbf{f}_{pi}(t) = 0$
while $\left(\rho^*_{err,avg}(t+\Delta t)\right) < \eta \;||\; (iter < minIterations)$ **do**
  for all i **do**
    predict velocity $\mathbf{v}^*_i(t+\Delta t)$
    predict position $\mathbf{x}^*_i(t+\Delta t)$
    compute predicted velocity's gradient $\frac{\mathrm{d}\mathbf{v}^*_i(t+\Delta t)}{\mathrm{d}\mathbf{s}}$
    compute predicted position's gradient $\frac{\mathrm{d}\mathbf{x}^*_i(t+\Delta t)}{\mathrm{d}\mathbf{s}}$
  for all i **do**
    predict density $\rho^*_i(t+\Delta t)$
    predict density variation $\rho^*_{err,i}(t+\Delta t)$
    compute pressure $p_i(t)$
    compute predicted density's gradient $\frac{\mathrm{d}\rho^*_i(t+\Delta t)}{\mathrm{d}\mathbf{s}}$
    compute pressure's gradient $\frac{\mathrm{d}p_i(t+\Delta t)}{\mathrm{d}\mathbf{s}}$
  for all i **do**
    compute pressure force $\mathbf{f}_{pi}(t)$
    compute pressure force's gradient $\frac{\mathrm{d}\mathbf{f}_{pi}(t)}{\mathrm{d}\mathbf{s}}$
for all i **do**
  compute new velocity $\mathbf{v}_i(t+\Delta t)$
  compute new position $\mathbf{x}_i(t+\Delta t)$
  compute new velocity's gradient $\frac{\mathrm{d}\mathbf{v}_i(t+\Delta t)}{\mathrm{d}\mathbf{s}}$
  compute new position's gradient $\frac{\mathrm{d}\mathbf{x}_i(t+\Delta t)}{\mathrm{d}\mathbf{s}}$

4.4 Multi-fluid PCISPH

To extend our model to incompressible multi-fluid systems, which is required in our latte art optimization task, we employ the Navier-Stokes-Cahn-Hilliard (NSCH) equations, following the approach of [Yang et al., 2015]. The NSCH equations for an n-phase fluid, neglecting the reactive stress term, are given as:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nabla\cdot(\nu\nabla\mathbf{v}) + \frac{\mathbf{f}_{ext}}{\rho}$$
$$\nabla\cdot\mathbf{v} = 0$$
$$\frac{Dc_k}{Dt} = \nabla\cdot(M\nabla\mu_k) \tag{15}$$
$$\mu_k = \frac{\partial F}{\partial c_k} - \epsilon^2\nabla^2 c_k - \frac{1}{n}\sum_{k'}\frac{\partial F}{\partial c_{k'}} \tag{16}$$

where c_k is the mass fraction of phase k, μ_k is its chemical potential, M is the degenerate mobility, ϵ is a parameter related to the diffuse-interface thickness, and F is the Helmholtz free energy. The spatial discretization of the Cahn-Hilliard terms is performed using SPH approximations similar to those used for the standard fluid dynamics terms:

$$\nabla \mu_i = \rho_i \sum_j m_j \left(\frac{\mu_i}{\rho_i^2} + \frac{\mu_j}{\rho_j^2} \right) \nabla W_{ij} \tag{17}$$

$$\nabla \cdot (M_i \nabla \mu_i) = \sum_j \frac{m_j}{\rho_j} (M_i + M_j) \mu_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \tag{18}$$

$$\nabla^2 \mathbf{c}_i = 2 \sum_j \frac{m_j}{\rho_j} \mathbf{c}_{ij} \frac{\mathbf{x}_{ij} \cdot \nabla W_{ij}}{\|\mathbf{x}_{ij}\|^2 + 0.01h^2} \tag{19}$$

The complete algorithm for the multi-fluid PCISPH solver is presented in Algorithm 3.

Algorithm 3 Multi-Phase PCISPH

for all i **do**
  find neighborhoods $N_i(t)$
for all i **do**
  compute mass $m_i(t)$
  compute density $\rho_i(t)$
  compute chemical potential $\mu_i(t)$
  compute new mass ratio $\mathbf{c}_i(t + \Delta t)$
for all i **do**
  compute new mass $m_i'(t)$
  compute new density $\rho_i'(t)$
  compute forces $\mathbf{f}_{vi}(t)$, $\mathbf{f}_{ext,i}(t)$
  initialize pressure $p_i(t) = 0$
  initialize pressure force $\mathbf{f}_{pi}(t) = 0$
while $\left(\rho^*_{err,avg}(t + \Delta t)\right) < \eta \;||\; (iter < minIterations)$ **do**
  for all i **do**
    predict velocity $\mathbf{v}_i^*(t + \Delta t)$
    predict position $\mathbf{x}_i^*(t + \Delta t)$
  for all i **do**
    predict density $\rho_i^*(t + \Delta t)$
    predict density variation $\rho^*_{err,i}(t + \Delta t)$
    compute pressure $p_i(t)$
  for all i **do**
    compute pressure forces $\mathbf{f}_{pi}(t)$
for all i **do**
  compute new velocity $\mathbf{v}_i(t + \Delta t)$
  compute new position $\mathbf{x}_i(t + \Delta t)$

5 Implementation

5.1 Framework

To build an end-to-end pipeline for optimizing latte art patterns, we utilize NVIDIA Warp, a Python framework for high-performance GPU computing. Warp's kernel-based programming model allows us to efficiently parallelize the SPH computations and utilize HashGrid data structure. Crucially, we utilize Warp's built-in automatic differentiation (AD) capabilities, specifically its Tape object, to compute gradients for optimization.

For the purpose of validating our gradients and analyzing their error, we also developed a C++/Eigen implementation. This allows us to derive and verify the analytical gradients with the precision offered by float64 or float80 data types, which is critical for gradient accuracy analysis.

5.2 Challenges

A primary challenge during implementation was managing numerical precision. The NVIDIA Warp framework, while highly efficient, has limited support for double-precision (float64) floating-point numbers in some of its core functions, particularly within the HashGrid data structure. This limitation makes it difficult to directly transfer and validate high-precision data types required for accurate gradient calculations.

To overcome this, we adopted a dual-pronged approach. The forward simulation and automatic differentiation for large-scale optimization tasks are performed in Warp using single-precision (float32) for maximum performance. For gradient validation and error analysis, we conduct smaller-scale tests using our C++/Eigen implementation, where we can compute the analytical gradients with high precision and establish a reliable ground truth. For multithreaded acceleration, we also use the OpemMP library.

5.3 Differentiable Rendering

To complete our end-to-end optimization pipeline, a differentiable renderer is required to connect the image-based loss with the fluid simulator. We implement this by first calculating a color field on a 3D grid, which utilizes the same spatial hashing structure as the simulator. The color at each grid node is computed in a manner analogous to SPH, by summing the weighted color contributions of all neighboring particles. From this 3D color grid, we then generate the final 2D image. The color of each pixel is determined through differentiable interpolation from the grid, ensuring that the entire rendering process is differentiable and that gradients can flow from the image-space loss back to the particle positions.

6 Experimental Results

6.1 Experimental Setup

All experiments were conducted on a system equipped with an NVIDIA RTX 4090 GPU and an Intel Core i9-13900K CPU. For optimization tasks, we utilized the Adam optimizer provided within the NVIDIA Warp framework, with the gravitational acceleration, $\mathbf{g}$, serving as the primary optimization parameter. For analyzing gradients error, we utilized C++ and Eigen. The specific simulation parameters, such as particle counts, time step sizes, are detailed in each of the following experimental sections.

6.2 Latte Art Optimization

This section demonstrates the effectiveness of our end-to-end differentiable pipeline, combining the differentiable simulator and renderer to solve a control problem.

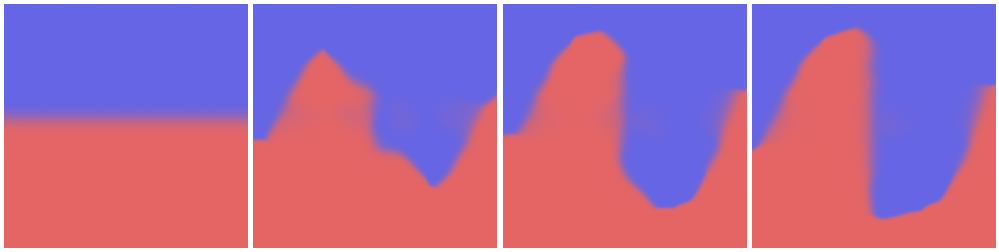
6.2.1 Differentiable Rendering

To verify that our differentiable renderer is functioning correctly, we perform an optimization task to arrange particle positions to match a target image. As shown in Figure 1, the optimization successfully rearranges an initial particle distribution into the desired pattern, demonstrating that meaningful gradients are being propagated from the image-space loss back to the particle positions.

6.2.2 Differentiable Simulation Validation

We first validate the gradients produced by Warp's automatic differentiation (autograd) system. The simulation involves approximately 8,000 particles with a time step of 0.0028s, using single-precision (float32) arithmetic. The loss is defined as the squared L2 norm between the final and target particle positions: $L = \sum \|\mathbf{x}_{final} - \mathbf{x}_{target,final}\|^2$. Figure 2 shows the convergence of the loss function during optimization, confirming that the gradients are effective for minimization.

However, we observe that the error of the autograd gradients accumulates over time. We conduct an experiment with approximately 8,000 particles and a time step of 0.0028s, optimizing based on an image-space loss: $L = \sum \|\mathbf{Image}_{final} - \mathbf{Image}_{target,final}\|^2$. Figure 3 compares the gradient



(a) Initial State (Step 0) (b) Step 120 (c) Step 180 (d) Final State (Step 240)

Figure 1: Optimization progress using the differentiable renderer to match a target image. The particles are successfully rearranged, confirming the renderer's functionality.

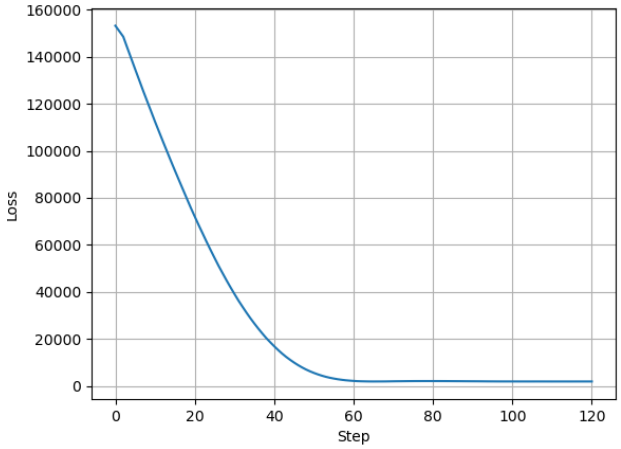


Figure 2: Loss curve for the latte art optimization task over 120 steps using the Adam optimizer, demonstrating effective gradient-based minimization.

values computed via autograd and finite differences (FD) over simulations of varying lengths. The FD gradients remain stable, whereas the autograd gradients diverge as the simulation progresses. This suggests that for long-horizon simulations, the accumulation of floating-point errors in autograd can become a significant issue.

6.3 Gradients Error Analysis

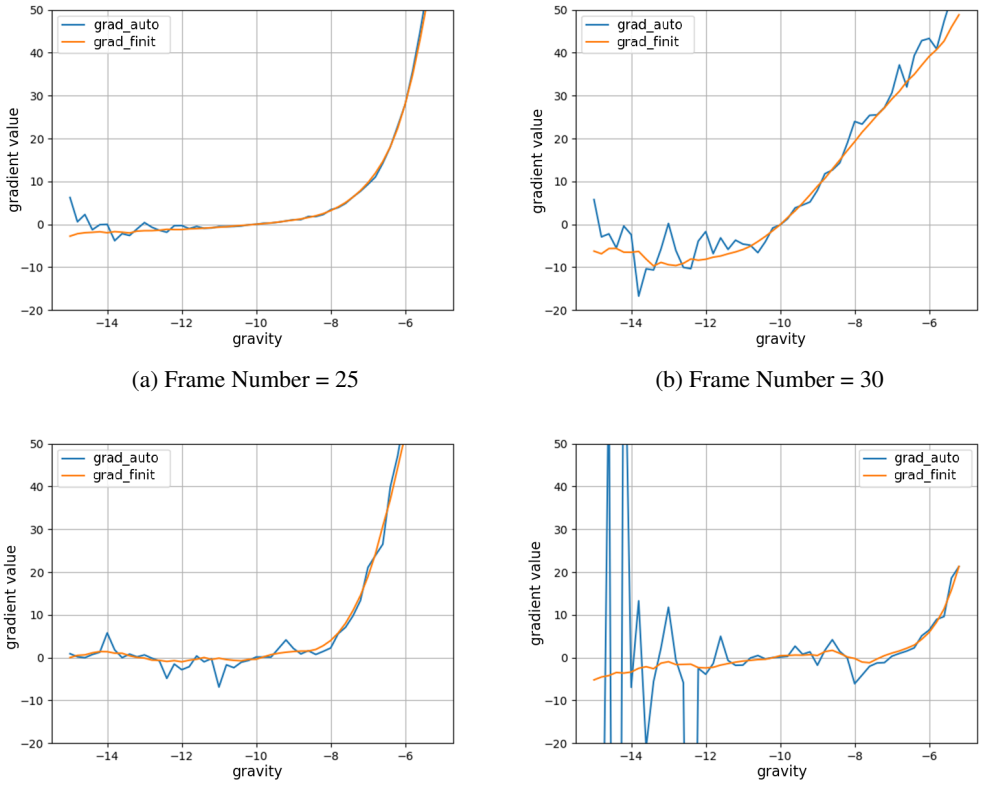
To better analyze the analytical gradient errors, particularly to study the contributions of different force derivatives and test various numerical precisions, we utilized our C++/Eigen implementation. This setup allows for high-precision computations in a simulation with 125 particles and a time step of 0.0021s.

6.3.1 Analytical vs. Finite Difference Gradients

We first compare our derived analytical gradients against finite difference gradients under varying gravity and numerical precision (float32, float64, float80). As shown in Figure 4, the finite difference gradients are stable and consistent across all precisions. In contrast, the analytical gradients exhibit significant variance, with the error increasing substantially at higher precisions and under stronger gravitational forces.

6.3.2 Ablation Study of Force Contributions

To identify the source of the analytical gradient error, we perform an ablation study where we systematically remove force components. We compare the full simulation (base case) against scenarios with no pressure force, no viscous force, and neither. Figures 5 and 6 show the gradient norm over the simulation frames for finite differences and our analytical method, respectively. The finite dif-



(a) Frame Number = 25

(b) Frame Number = 30

(c) Frame Number = 35

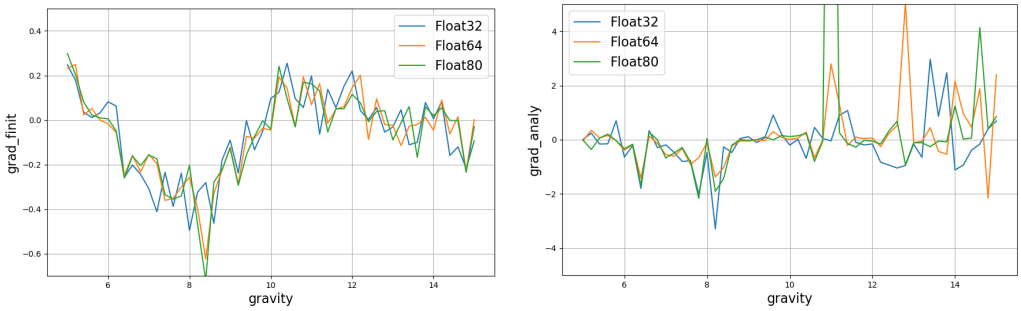
(d) Frame Number = 40

Figure 3: Comparison of gradient magnitudes from autograd vs. finite differences for simulations of increasing length. As the frame number increases, the autograd gradient error grows, while the finite difference gradient remains stable.

ference gradients remain stable in all cases. The analytical gradients, however, only become stable when the pressure force term is removed. This strongly indicates that the pressure force derivative is the primary source of numerical instability and error.

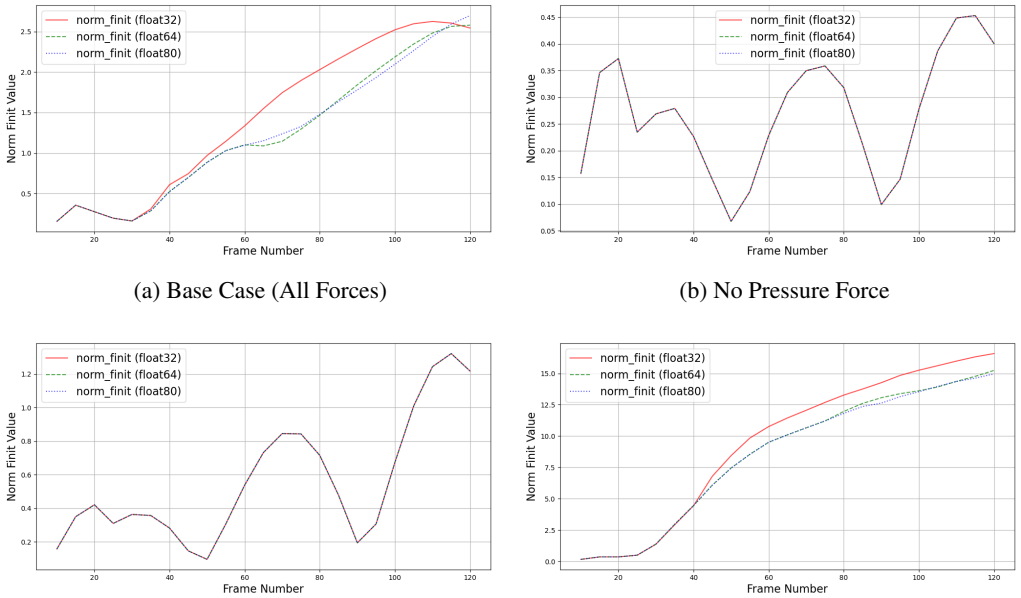
6.3.3 Analysis of the Pressure Gradient Term

The derivative of the pressure force consists of two main terms, as shown in Equation 13. To pinpoint the cause of the instability, we analyze the contribution of each term individually. Figure 7 plots the norm of the full pressure gradient, the first term alone, and the second term alone, all computed at float32 precision. The results clearly show that the first term, which contains the Hessian of the smoothing kernel $\nabla(\nabla W)$, is the dominant factor causing the gradient to diverge.



(a) Finite Difference Gradients (b) Analytical Gradients

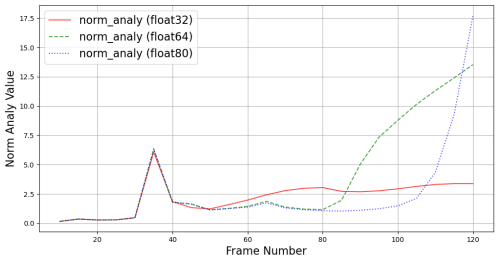
Figure 4: Comparison of gradient magnitudes vs. gravity. Finite difference gradients (left) are stable across precisions, while analytical gradients (right) are sensitive to both precision and the magnitude of external forces.



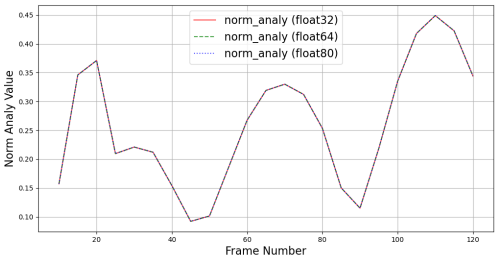
(a) Base Case (All Forces) (b) No Pressure Force

(c) No Pressure or Viscous Force (d) No Viscous Force

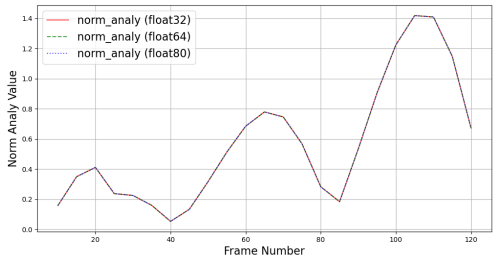
Figure 5: Finite difference gradient norm vs. frame number under different force configurations. The gradients remain stable regardless of which forces are included.



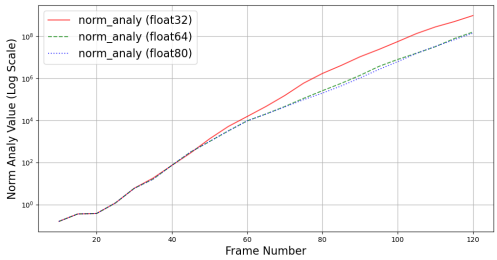
(a) Base Case (All Forces)



(b) No Pressure Force



(c) No Pressure or Viscous Force



(d) No Viscous Force

Figure 6: Analytical gradient norm vs. frame number under different force configurations. The gradient norm is stable only when the pressure force is removed.

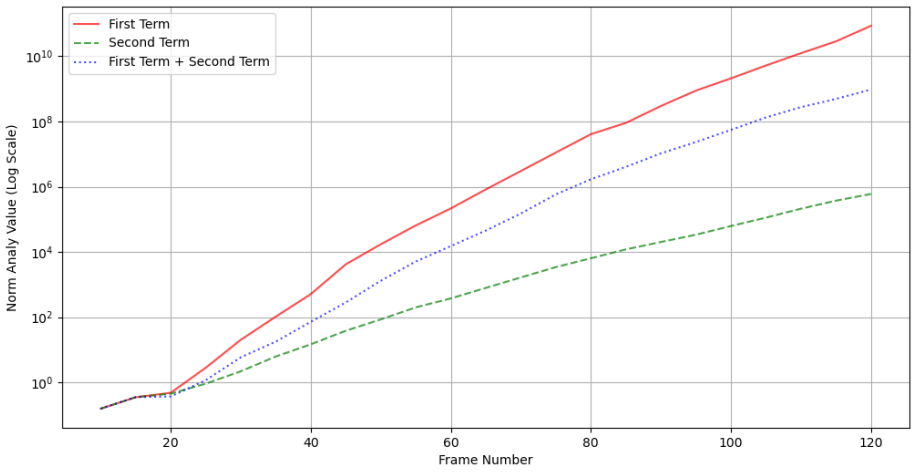


Figure 7: Analysis of the Pressure Gradient Term

7 Discussion

Our experimental results demonstrate both the potential and the pitfalls of differentiable SPH simulation. The latte art optimization confirms that an end-to-end, gradient-based pipeline is viable for fluid control, especially for short-horizon tasks. The convergence validates that both the autograd-based simulator and our custom renderer produce meaningful gradients.

However, our core finding is the identification of a fundamental numerical instability in the analytical gradient (also in autograd). The ablation study decisively isolates the pressure force derivative as the source of this instability, with further analysis pinpointing the Hessian of the smoothing kernel, $\nabla(\nabla W)$, as the specific culprit. This term, which is absent in the forward pass, introduces high-frequency errors that cause the gradient to explode during backpropagation. This explains the divergence between the unstable analytical method and the stable finite difference method. The primary limitation of this work is that while we identify the source of the error, we do not yet propose a solution.

8 Conclusion and Future Work

In this research, we developed an end-to-end differentiable pipeline for SPH-based fluid control and conducted a rigorous analysis of gradient computation. Our key contribution is the identification of the pressure force term's derivative, specifically the kernel Hessian $\nabla(\nabla W)$, as the primary source of instability in analytical gradient calculations.

This work reveals that neither standard autograd nor direct analytical differentiation is sufficient for robust, long-horizon differentiable SPH. Based on this, future work will focus on the following directions:

- **Stabilizing the Pressure Gradient:** Develop methods to stabilize or bypass the problematic $\nabla(\nabla W)$ term, perhaps through alternative SPH pressure formulations or regularization techniques applied only during the backward pass.
- **Hybrid Differentiation Methods:** Investigate hybrid approaches that combine the strengths of different methods, such as using analytical gradients for stable terms and a more robust approximation for the pressure term.
- **Application to Long-Horizon Problems:** Once a more stable gradient computation method is developed, apply the pipeline to more challenging, long-horizon control and inverse problems to validate its effectiveness.

By addressing this fundamental instability, this research paves the way for making differentiable SPH a more reliable and widely applicable tool.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. Hao Su, for his invaluable guidance and support throughout this research. And I also extend my thanks to Xiaodi and Fanbo for their insightful advice and fruitful discussions during the course of this project.

References

Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217, 2007.

Markus Becker, Hendrik Tessendorf, and Matthias Teschner. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):493–503, 2009.

Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 147–155, 2015.

Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation96: Proceedings of the Eurographics Workshop in Poitiers, France, August 31–September 1, 1996*, pages 61–76. Springer, 1996.

Tao Du, Kui Wu, Andrew Spielberg, Wojciech Matusik, Bo Zhu, and Eftychios Sifakis. Functional optimization of fluidic devices with differentiable stokes flow. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (ToG)*, 41 (2):1–21, 2021.

Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 736–744, 2002.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

David Hahn, Pol Banzet, James M Bern, and Stelian Coros. Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

Xiaowei He, Ning Liu, Sheng Li, Hongan Wang, and Guoping Wang. Local poisson sph for viscous incompressible fluids. In *Computer Graphics Forum*, volume 31, pages 1948–1958. Wiley Online Library, 2012.

Philipp Holl, Vladlen Koltun, Kiwon Um, and Nils Thuerey. phiflow: A differentiable pde solving framework for deep learning via physical simulations. In *NeurIPS workshop*, volume 2, 2020.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019a.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE, 2019b.

Yifei Li, Tao Du, Sangeetha Grama Srinivasan, Kui Wu, Bo Zhu, Eftychios Sifakis, and Wojciech Matusik. Fluidic topology optimization with an anisotropic mixture model. *ACM Transactions on Graphics (TOG)*, 41(6):1–14, 2022a.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)*, 42(1):1–20, 2022b.

Yifei Li, Yuchen Sun, Pingchuan Ma, Eftychios Sifakis, Tao Du, Bo Zhu, and Wojciech Matusik. Neuralfluid: Nueral fluidic system design and control with differentiable simulation. *Advances in Neural Information Processing Systems*, 37:84944–84967, 2024.

Zhehao Li, Qingyu Xu, Xiaohan Ye, Bo Ren, and Ligang Liu. Difffr: Differentiable sph-based fluid-rigid coupling for rigid body control. *ACM Transactions on Graphics (TOG)*, 42(6):1–17, 2023.

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *Advances in neural information processing systems*, 32, 2019.

Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, 82:1013–1024, 1977.

Ran Luo, Weiwei Xu, Tianjia Shao, Hongyi Xu, and Yin Yang. Accelerated complex-step finite difference for expedient deformable simulation. *ACM Transactions on Graphics (TOG)*, 38(6): 1–16, 2019.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3):449–456, 2004.

Joe J Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.

J Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International conference on learning representations*, 2020.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pages 8661–8671. PMLR, 2021.

Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR, 2018.

Siyuan Shen, Yang Yin, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. High-order differentiable autoencoder for nonlinear model reduction. *arXiv preprint arXiv:2102.11026*, 2021.

Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 papers*, pages 1–6. 2009.

Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. *arXiv preprint arXiv:2103.16021*, 2021.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An end-to-end differentiable framework for contact-aware robot design. *arXiv preprint arXiv:2107.07501*, 2021.

Tao Yang, Jian Chang, Bo Ren, Ming C Lin, Jian Jun Zhang, and Shi-Min Hu. Fast multiple-fluid simulation using helmholtz free energy. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015.