Gaussian Fluids: A Grid-Free Fluid Solver based on Gaussian Spatial Representation

JINGRUI XING, School of Intelligence Science and Technology, Peking University, China
BIN WANG, Independent Researcher, China
MENGYU CHU*, State Key Laboratory of General Artificial Intelligence, Peking University, China
BAOQUAN CHEN*, State Key Laboratory of General Artificial Intelligence, Peking University, China

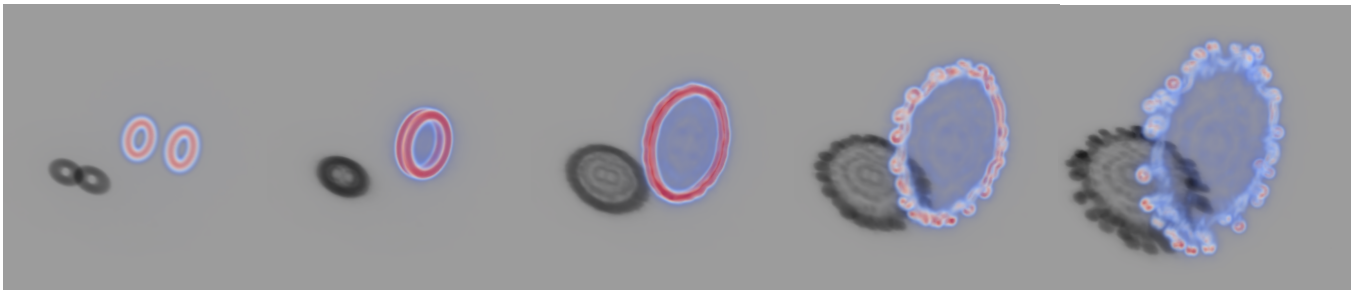


Fig. 1. Simulation of two vortex rings colliding using our grid-free fluid solver based on Gaussian Spatial Representation. The figures visualize the vorticity magnitude at frames 0, 74, 123, 178, and 242, showcasing long temporal stability, improved vorticity preservation, and adaptive spatial accuracy, capturing fine-scale details with intricate dynamics, such as the disturbed large vortex ring splitting into massive smaller ones.

We present a grid-free fluid solver featuring a novel Gaussian representation. Drawing inspiration from the expressive capabilities of 3D Gaussian Splatting in multi-view image reconstruction, we model the continuous flow velocity as a weighted sum of multiple Gaussian functions. This representation is continuously differentiable, which enables us to derive spatial differentials directly and solve the time-dependent PDE via a custom first-order optimization tailored to fluid dynamics. Compared to traditional discretizations, which typically adopt Eulerian, Lagrangian, or hybrid perspectives, our approach is inherently memory-efficient and spatially adaptive, enabling it to preserve fine-scale structures and vortices with high fidelity. While these advantages are also sought by implicit neural representations, GSR offers enhanced robustness, accuracy, and generality across diverse fluid phenomena, with improved computational efficiency during temporal evolution. Though our first-order solver does not yet match the speed of fluid solvers using explicit representations, its continuous nature substantially reduces spatial discretization error and opens a new avenue for high-fidelity simulation.We evaluate the proposed solver across a broad range of 2D and 3D fluid phenomena, demonstrating its ability to preserve intricate vortex dynamics, accurately capture boundary-induced effects such as Kármán vortex streets, and remain robust across long time horizons—all without additional parameter tuning. Our results suggest that GSR offers a compelling direction for future research in fluid simulation. The source code for our fluid solver is publicly available at https://github.com/xjr01/Gaussian-Fluids-Code.

*Corresponding authors

SIGGRAPH Conference Papers '25, August 10–14, 2025, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08…$15.00
https://doi.org/10.1145/3721238.3730620

CCS Concepts: • **Computing methodologies** → **Physical simulation**; • **Applied computing** → *Physics*.

Additional Key Words and Phrases: Fluid Simulation, Gaussian Spatial Representation

ACM Reference Format:
Jingrui Xing, Bin Wang, Mengyu Chu, and Baoquan Chen. 2025. Gaussian Fluids: A Grid-Free Fluid Solver based on Gaussian Spatial Representation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3721238.3730620

1 INTRODUCTION

Fluid phenomena have presented enduring challenges in computer graphics, characterized by the need for expressive representations to capture rich spatial details, alongside the demand for efficient and accurate temporal evolution to handle non-linear and chaotic dynamics. Traditional fluid solvers lean on grid-based or particle-based spatial representations, aligning with the Eulerian and Lagrangian perspectives. Hybrid approaches have also been proposed. These straightforward spatial discretizations enable efficient calculations for solving temporal dynamics.

Despite substantial progress, traditional solvers persistently face challenges due to limited expressiveness. Eulerian methods often suffer from numerical viscosity arising from the lack of continuity, while the limited accuracy of Lagrangian methods like the Smoothed Particle Hydrodynamics (SPH) impedes the capture of fine structures [Koschier et al. 2022]. Hybrid approaches aim to combine the strengths of both views, but introduce numerical error in data transferring between the two discretizations. Though these limitations can be alleviated with extensive memory usage, it would lead to the curse of dimensionality.

To deal with these challenges, we aim to utilize expressive, continuous spatial representations for fluid simulation. In this context, both Implicit Neural Representations (INR) and 3D Gaussian functions have recently shown promise in representing both function values and derivatives with high fidelity and spatial adaptiveness. While we focus on Gaussian functions due to their flexibility and efficiency, our endeavor aligns with the growing trend of employing INR in simulations, e.g. for fluid dynamics [Chen et al. 2023], garment untangling [Santesteban et al. 2022], and facial animation [Yang et al. 2023]. Yet, the continuous nature of these representations often incurs greater computational cost and optimization challenges, such as slow convergence [Wang et al. 2022] and difficulty in enforcing hard physical constraints [Chen et al. 2023]. There remains no consensus on an efficient, unified PDE-solving strategy or interactive editing workflow.

We propose Gaussian Spatial Representation (GSR), a novel continuous representation for fluid dynamics, paired with a first-order, physics-guided optimization framework. We carefully design a composite loss that enforces incompressibility and preserves vorticity. We also introduce an optimization strategy to resolve conflicts in gradient directions, enhancing stability and convergence. Our solution consistently outperforms INR-based approaches in stability, accuracy, and detail preservation.

In summary, GSR offers the following properties:

- Continuous Differentiability: Enables accurate and efficient computation of spatial derivatives for PDE terms.
- Compactness and Adaptivity: Significantly reduces memory footprint while retaining thin structures.
- Vorticity Fidelity: Maintains vortex structures while handles harmonic components more faithfully than vortex-only approaches.

We acknowledge that the first-order solver speed and strict constraint enforcement still lag behind well-established discretizations, but our results, achieved without per-scene parameter tuning, demonstrate a stable, spatially adaptive solver. It performs well across various 2D and 3D scenarios, with and without obstacles, charting a promising direction for continuous, grid-free fluid simulation.

2 RELATED WORK

Traditional Fluid Simulation Methods. We briefly summarize traditional simulation methods with a focus on single-phase flows. Since the introduction of the stable fluids algorithm [Stam 1999], various methods have been developed to solve the highly nonlinear Navier-Stokes equations following the operator-splitting technique. Grid-based methods primarily focus on accelerating the pressure projection step [Aanjaneya et al. 2017; McAdams et al. 2010] and reducing numerical dissipation with more accurate advection [Kim et al. 2005; Selle et al. 2008]. Recently, there has been growing interest in impulse-based fluid methods [Feng et al. 2023a; Nabizadeh et al. 2022; Selle et al. 2008], which refine advection through a reformulation of the governing equations into the impulse-velocity framework. These methods lead to better vorticity preservation, especially within Eulerian velocity-based solvers, and demonstrate improved stability in capturing fine-scale features.

Lagrangian alternatives such as Smoothed-particle hydrodynamics (SPH) methods [Müller et al. 2003] have been widely explored. These methods focus on enhancing incompressibility [Bender and Koschier 2016; Ihmsen et al. 2013], improving spatial adaptivity [Owen et al. 1998], and mitigating errors due to interpolation inconsistencies in sparsely sampled regions [Band et al. 2018; Westhofen et al. 2023]. A comprehensive overview of SPH techniques can be found in [Koschier et al. 2019]. Vortex methods typically use Lagrangian representations such as particles [Cottet et al. 2000], filaments [Weißmann and Pinkall 2010], and sheets [Brochu et al. 2012]. They reformulate the fluid equations into vorticity-velocity form and exhibit difficulties in geometric and boundary treatments. Many hybrid methods are proposed to combine the advantages among them.

There are Eulerian and Lagrangian hybrid methods [Foster and Metaxas 1996; Jiang et al. 2015; Zhu and Bridson 2005] as well as vorticity and velocity hybrid methods [Koumoutsakos et al. 2008; Pfaff et al. 2012]. In general, traditional methods face challenges due to the limited expressivity of their spatial representations and improvements have been proposed over the years. We focus on using continuous representation with more expressivity. Although our optimization-based approach has some limitations compared to established state-of-the-art methods using explicit representations, it offers advantages in detail preservation and spatial adaptivity, which we consider a promising direction for exploration.

Emerging Trends in Simulation using Continuous Representations. Continuous representations with accurate gradient estimations are attracting wide attention in graphics and vision tasks, such as scene reconstruction [Feng et al. 2024; Mildenhall et al. 2021], video compression [Chen et al. 2021], and physics spatial-temporal approximation [Karniadakis et al. 2021]. There is also a trend to apply INR and Gaussian functions in simulations. Many works apply them as mass distribution functions and incorporate them with existing numerical solvers (e.g., Material Point Methods) to support physics-based scene editing or animation [Feng et al. 2023b; Xie et al. 2023]. Deng et al. [2023] leverages INR to store flow maps and generate non-dissipative results using a grid-based fluid solver with long-term advection. However, fewer works explore how to solve time-dependent PDEs using these novel representations as spatial functions. The most relevant works to us are Implicit Neural Spatial Representations for PDEs (INSR) [Chen et al. 2023] and Neural Monte Carlo Fluids (NMC) [Jain et al. 2024], where the former applies INR as spatial representations for fluid and soft-body simulations and solves temporal PDEs via optimization, while the latter takes advantage of the continuity nature of INR by leveraging the walk on stars method for pressure solving and augment the INR to fit boundary conditions. Our method takes advantage of the locality and flexibility of Gaussian functions and achieves better efficiency and stability in temporal evolution.

3 BACKGROUND

The motion of incompressible fluid is governed by the Navier-Stokes equations with the divergence-free constraint:

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{g}, \\ \nabla\cdot\boldsymbol{u} &= 0, \end{aligned} \tag{1}$$

where $\boldsymbol{u}$ is the velocity field, ρ is density, p is pressure, ν is kinematic viscosity, and $\boldsymbol{g}$ is acceleration due to external force. To compute the time evolution of $\boldsymbol{u}$, numerical solvers typically employ an operator splitting scheme, consisting of two main steps: advection and projection. In the advection step, the equation $\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = 0$ is solved by transporting the velocity field to its new positions at the next time step, i.e. $\boldsymbol{u}^*(\boldsymbol{x}) \leftarrow \boldsymbol{u}^{n-1}(\Psi^{n-1}(\boldsymbol{x}))$, where Ψ^{n-1} maps a position at frame n to its corresponding position at frame $n-1$. In the projection step, the pressure field $p^*(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{\rho}$ is solved subject to the divergence-free constraint, i.e., $\nabla \cdot (\boldsymbol{u}^* - \nabla p^*) = 0$. The velocity field at the next frame is then updated as $\boldsymbol{u}^n(\boldsymbol{x}) \leftarrow \boldsymbol{u}^*(\boldsymbol{x}) - \nabla p^*(\boldsymbol{x})$. In addition to the two main steps, the external forces are usually applied to the velocity field before advection, while viscosity is introduced to the system between advection and projection.

4 GAUSSIAN SPATIAL REPRESENTATION

Math Definition. Motivated by the high expressiveness of 3D Gaussian Splatting [Kerbl et al. 2023] in reconstruction tasks, we propose Gaussian spatial representation (GSR), a continuous, differentiable, and memory-efficient spatial representation based on combined Gaussian functions. A d-dimension ($d \in \{2, 3\}$) Gaussian function $G_i : \mathbb{R}^d \to \mathbb{R}$ can be formulated as:

$$G_i(\boldsymbol{x}) = \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^\top \Sigma_i^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)\right\}, \tag{2}$$

where $\boldsymbol{\mu}_i$ is the position of the Gaussian particle i, and Σ_i is its covariance matrix. Since Σ_i^{-1} is positive definite, we can further decompose it into

$$\Sigma_i^{-1} = \boldsymbol{R}_i \boldsymbol{S}_i^{-1} \boldsymbol{S}_i^{-1} \boldsymbol{R}_i^\top, \tag{3}$$

where $\boldsymbol{R}_i$ is a d-dimension rotation matrix, and $\boldsymbol{S}_i$ is a positive diagonal matrix. The rotation $\boldsymbol{R}_i$ can be represented as an angle θ_i in 2D and a quaternion $\boldsymbol{r}_i$ in 3D. For convenience of implementation, we store the diagonal elements of $\boldsymbol{S}_i^{-1}$ (denoted as $\boldsymbol{s}_i^{-1}$) instead of $\boldsymbol{S}_i$ for particle i. A GSR is a vector field $\tilde{\boldsymbol{v}} : \mathbb{R}^d \to \mathbb{R}^m$ defined as the weighted sum of all Gaussian functions:

$$\tilde{\boldsymbol{v}}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{v}_i G_i(\boldsymbol{x}), \tag{4}$$

where $\boldsymbol{v}_i \in \mathbb{R}^m$ is the weight of Gaussian particle i, and N is the number of Gaussian particles. Hence, the parameters of a GSR are $\Theta = \{\boldsymbol{v}_i, \boldsymbol{\mu}_i, \theta_i, \boldsymbol{s}_i^{-1} : i = 1, \cdots, N\}$ in 2D and $\Theta = \{\boldsymbol{v}_i, \boldsymbol{\mu}_i, \boldsymbol{r}_i, \boldsymbol{s}_i^{-1} : i = 1, \cdots, N\}$ in 3D.

Efficiency Improvements. The GSR defined in Equation 4 initially requires $O(N)$ floating-point operations to evaluate the field at a single point, which becomes computationally prohibitive as the number of queries grows. To mitigate this, we apply a local restriction to each Gaussian function, exploiting its rapid decay with distance. This results in the clamped Gaussian function formulation:

$$\hat{G}_i(\boldsymbol{x}) = \begin{cases} G_i(\boldsymbol{x}) - c, & G_i(\boldsymbol{x}) \geq c \\ 0, & G_i(\boldsymbol{x}) < c \end{cases}, \tag{5}$$

where c is a small positive threshold. We subtract the Gaussian function by c to avoid discontinuity of the kernel function. Accordingly, we change the definition of GSR in Equation 4 into:

$$\tilde{\boldsymbol{v}}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{v}_i \hat{G}_i(\boldsymbol{x}). \tag{6}$$

A hash table is employed to store the Gaussian particles based on spatial locality, enabling fast retrieval of only the relevant Gaussian particles. This reduces the time complexity for each query to $O(1)$. Details can be found in supplemental files.

Advantage on Differentiability. The gradient of the GSR can be computed directly from its definition:

$$\nabla\tilde{\boldsymbol{v}}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{v}_i \nabla\hat{G}_i(\boldsymbol{x}) = -\sum_{i \in \mathcal{N}(\boldsymbol{x})} G_i(\boldsymbol{x}) \boldsymbol{v}_i (\boldsymbol{x} - \boldsymbol{\mu}_i)^\top \Sigma_i^{-1}, \tag{7}$$

where $\mathcal{N}(\boldsymbol{x}) = \{i : \hat{G}_i(\boldsymbol{x}) \geq 0\}$ is the set of the particle indices near $\boldsymbol{x}$. Though the GSR is not differentiable at the boundary of any clamped Gaussian function, i.e. $\tilde{\boldsymbol{v}}(\boldsymbol{x})$ is not differentiable when $\exists i \in \{1, \cdots, N\}$, $G_i(\boldsymbol{x}) = c$, Equation 7 is a good approximation for small cs. Furthermore, we can naturally derive the divergence and curl operators from this formulation. Unlike implicit neural representations that rely on auto-differentiation for computing differential quantities, GSR allows direct and efficient differentiation, with the same time complexity as evaluating the GSR field itself. This inherent advantage leads to faster optimization when enforcing physics-based constraints, making GSR a computationally superior choice for dynamic simulations.

5 ALGORITHM

Our work focuses on the dynamics of inviscid fluids with no external force, i.e. $\nu = 0$ and $\boldsymbol{g} = \boldsymbol{0}$ in Equation 1. Our fluid solver can be divided into initialization, physics-based optimization, and reseeding, as shown in Algorithm 1.

Algorithm 1 Fluid solver with GSR

```
u^0 ← Initialize(u)
for n ← 1, ··· , T do
    ũ^{n-1} ← Reseed(u^{n-1})
    //An initial guess for physics-based optimization:
    ũ* ← AdvectPositions(ũ^{n-1})
    //Physics-based optimization:
    u^n ← OptimizeLosses(ũ*, u^{n-1})
end for
```

5.1 Initialization

At the very beginning of the simulation, we initialize the GSR $\tilde{\boldsymbol{u}}^0(\boldsymbol{x})$ to fit a given velocity field $\boldsymbol{u}(\boldsymbol{x})$. GSR is capable of fitting any continuous vector field if trained properly with a sufficient number of kernels. For any vector field $\boldsymbol{v} : \mathbb{R}^d \to \mathbb{R}^m$ defined on domain $\mathcal{D} \subset \mathbb{R}^d$, our goal is to find a GSR $\tilde{\boldsymbol{v}} : \mathbb{R}^d \to \mathbb{R}^m$ as close to $\boldsymbol{v}(\boldsymbol{x})$ on $\mathcal{D}$ as possible. This can be interpreted as an optimization problem: $\arg\min_\Theta \frac{1}{d|\mathcal{D}|} \int_{\mathcal{D}} \|\boldsymbol{v}(\boldsymbol{x}) - \tilde{\boldsymbol{v}}(\boldsymbol{x})\|_1 \mathrm{d}V$. Directly evaluating the integral is difficult, thus we take the following value loss to approximate

it in a Monte-Carlo way:

$$\mathcal{L}_{\text{val}} = \frac{1}{Qd}\sum_{j=1}^{Q}\|\boldsymbol{v}(\boldsymbol{x}_j) - \tilde{\boldsymbol{v}}(\boldsymbol{x}_j)\|_1, \tag{8}$$

where $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_Q$ are uniformly randomly sampled from $\mathcal{D}$ in each iteration. To promote high-quality fitting of GSR, we employ an additional gradient loss:

$$\mathcal{L}_{\text{grad}} = \frac{1}{Qd^2}\sum_{j=1}^{Q}\|\nabla\boldsymbol{v}(\boldsymbol{x}_j) - \nabla\tilde{\boldsymbol{v}}(\boldsymbol{x}_j)\|_{\text{sum}}, \tag{9}$$

where $\|\cdot\|_{\text{sum}} = \sum_{k=1}^{d}\sum_{l=1}^{m}|[\cdot]_{kl}|$ is the sum of the absolute values of all elements. The gradient loss can supervise a local neighborhood, which makes a difference to the point-wise value loss, improving the training efficiency on the continuous field.

In addition to the value loss and the gradient loss, we leverage anisotropic loss $\mathcal{L}_{\text{aniso}}$ proposed by Xie et al. [2023] and volume loss $\mathcal{L}_{\text{vol}}$ proposed by Feng et al. [2024] as regularization terms:

$$\mathcal{L}_{\text{aniso}} = \frac{1}{N}\sum_{i=1}^{N}\max\left(\frac{\max(\boldsymbol{s}_i)}{\min(\boldsymbol{s}_i)} - r_{\text{aniso}}, 0\right), \text{ and} \tag{10}$$

$$\mathcal{L}_{\text{vol}} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\prod(\boldsymbol{s}_i)}{\frac{1}{N}\sum_{j=1}^{N}\prod(\boldsymbol{s}_j)} - 1\right)^2, \tag{11}$$

where $\max(\cdot)$ is the maximum element of a vector, $\prod(\cdot)$ is the product of all elements of a vector, r_{aniso} is the maximum threshold of the ratio between the major axis length and minor axis length of a Gaussian particle. We take $r_{\text{aniso}} = 1.5$ in all our experiments.

The total loss in the initialization process is:

$$\mathcal{L}_{\text{init}} = \mathcal{L}_{\text{val}} + \mathcal{L}_{\text{grad}} + \mathcal{L}_{\text{aniso}} + \mathcal{L}_{\text{vol}}. \tag{12}$$

We initialize the parameters of GSR adaptively according to the particle number, with details specified in the supplementary. We then run the optimization for enough iterations to get an initial GSR ready for the simulation.

5.2 Physics-Based Optimization

We formulate the time integration as an optimization problem with physics-based losses, training the GSR from an initial guess inspired by Lagrangian advection to better represents the velocity field at the next frame. Our optimization process leverages physics-guided gradients to ensure adherence to physical constraints. By employing a sampling-based strategy, our method effectively handles boundary conditions without requiring explicit cell-cutting operations, offering a seamless and flexible approach to simulate complex geometries. The details of the initial guess, losses, gradient adjustments, and boundary sampling are described as follows.

5.2.1 Advection-Based Initial Guess. To encourage fast convergence and temporal consistency, we make an initial guess for the physics-based optimization with an approximated velocity field advected from the last frame. Drawing inspiration from the advection step in Lagrangian methods, we treat the Gaussian particles as Lagrangian particles and update the position of each particle by moving it along the velocity for a time step h. Unlike Lagrangian approaches where each particle moves with its own velocity, our Gaussian particles are advected using velocities sampled from the Gaussian field according to Eq. 4. To improve accuracy, we apply the 4-th order Runge-Kutta convention. Denoting $\boldsymbol{\Phi}^{n-1} : \mathbb{R}^d \to \mathbb{R}^d$ as the mapping from a position at time step $n-1$ to the position after applying the RK4 time integration, the advection step is given by $\boldsymbol{\mu}_i^* \leftarrow \boldsymbol{\Phi}^{n-1}(\boldsymbol{\mu}_i^{n-1})$.

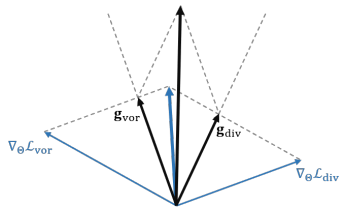


Fig. 2. The gradiant projection technique can turn the contradicted gradients into compatible ones, resulting in larger steps in gradient descent.

While this step does not precisely solve $\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = 0$, it provides a reliable initial state for subsequent steps. It maintains a relatively uniform particle distribution without clustering, as long as the velocity field remains approximately divergence-free, as ensured in later steps. We also use the advected positions as regularization in subsequent optimization by introducing a constraint specified in Section 5.2.5 to limit excessive drift of particles. Serving both as initialization and regularization, the advected positions accelerate optimization by guiding the solution toward a local optimum, improving temporal consistency and stability of the spatial representation. The effectiveness of this step is validated through an ablation study in Section 6.4.

5.2.2 The PDE Losses. We first introduce the vorticity loss and the divergence loss used in the physics-based optimization step:

$$\mathcal{L}_{\text{vor}} = \frac{1}{Q\hat{d}}\sum_{j=1}^{Q}\|\nabla\times\tilde{\boldsymbol{u}}^n(\boldsymbol{x}_j) - \omega(\boldsymbol{x}_j)\|_1, \tag{13}$$

$$\mathcal{L}_{\text{div}} = \frac{1}{Q}\sum_{j=1}^{Q}|\nabla\cdot\tilde{\boldsymbol{u}}^n(\boldsymbol{x}_j)|^2, \tag{14}$$

where $\hat{d} = 1$ in 2D examples and $\hat{d} = 3$ in 3D, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_Q$ are uniformly randomly sampled from $\mathcal{D}$ in each iteration of the optimization, $\omega(\boldsymbol{x})$ is the vorticity field advected from $\nabla\times\tilde{\boldsymbol{u}}^{n-1}(\boldsymbol{x})$. In 2D, the vorticity simply transports along the velocity field, i.e.

$$\omega(\boldsymbol{x}) = \nabla\times\tilde{\boldsymbol{u}}^{n-1}(\boldsymbol{\Psi}^{n-1}(\boldsymbol{x})). \tag{15}$$

In 3D, the vorticity field evolves according to $\frac{\mathrm{D}\omega}{\mathrm{D}t} = \nabla\boldsymbol{u}\cdot\omega$, which can be characterized by the bidirectional flow map as mentioned by Wang et al. [2024]:

$$\omega(\boldsymbol{x}) = \mathrm{d}\boldsymbol{\Phi}^{n-1}(\boldsymbol{\Psi}^{n-1}(\boldsymbol{x}))\nabla\times\tilde{\boldsymbol{u}}^{n-1}(\boldsymbol{\Psi}^{n-1}(\boldsymbol{x})), \tag{16}$$

where $\mathrm{d}\boldsymbol{\Phi}^{n-1}(\boldsymbol{x})$ denotes the Jacobian matrix of the forward mapping $\boldsymbol{\Phi}^{n-1}$ at $\boldsymbol{x}$.

5.2.3 Gradient Projection. We apply the gradient projection technique, a gradient strategy typically applied in *Multi-Task Learning (MTL)* [Dong et al. 2022; Liu et al. 2025; Yu et al. 2020], to the vorticity loss and the divergence loss in the pressure solve training process, as their gradients may contradict. In the backward stage,

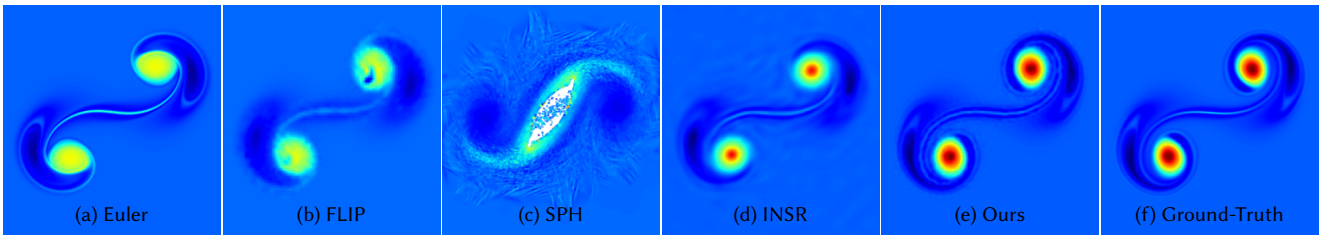


Fig. 3. Our method compared with traditional Eulerian and Lagrangian discretizations and INSR on the Taylor vortex example. We show frame 300 while more are given in figure 9.

after computing the gradients of the two losses $\nabla_\Theta \mathcal{L}_{\text{vor}}$ and $\nabla_\Theta \mathcal{L}_{\text{div}}$, we check if their dot product is negative. If so, this indicates a contradiction between the two gradients, meaning that following one gradient would increase the other loss. When this occurs, let $\boldsymbol{t}_1, \boldsymbol{t}_2$ be the normalized vector of $\nabla_\Theta \mathcal{L}_{\text{vor}}$ and $\nabla_\Theta \mathcal{L}_{\text{div}}$, respectively. We then modify the gradients of the two losses to:

$$\boldsymbol{g}_{\text{vor}} = \nabla_\Theta \mathcal{L}_{\text{vor}} - (\nabla_\Theta \mathcal{L}_{\text{vor}} \cdot \boldsymbol{t}_2)\boldsymbol{t}_2, \tag{17}$$

$$\boldsymbol{g}_{\text{div}} = \nabla_\Theta \mathcal{L}_{\text{div}} - (\nabla_\Theta \mathcal{L}_{\text{div}} \cdot \boldsymbol{t}_1)\boldsymbol{t}_1. \tag{18}$$

Moving along $\boldsymbol{g}_{\text{vor}}$ would not affect $\mathcal{L}_{\text{div}}$ and vice versa, hence increasing the efficiency of each gradient descent step, as shown in Figure 2. This technique follows the MTL strategy proposed by Yu et al. [2020], which in our case can reduce the ripple artifact in the vorticity field due to sub-optimal optimization processes converging to local minima, as later shown by an ablation test in Section 6.4.

5.2.4 Boundary Handling. There are two types of boundary conditions across all experiments in this paper. The First type can be formulated as $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_{\text{b}}(\boldsymbol{x})$ while the second $(\boldsymbol{u} \cdot \boldsymbol{n})(\boldsymbol{x}) = f(\boldsymbol{x})$, where $\boldsymbol{x} \in \partial\mathcal{D}$, $\boldsymbol{n}$ is the boundary normal, $\boldsymbol{u}_{\text{b}}$ and f are given functions. They stand for the "no-slip" and "free-slip" conditions, respectively. We handle the two types of boundary conditions by introducing two boundary losses:

$$\mathcal{L}_{\text{b1}} = \frac{1}{Q_{\text{b1}} d} \sum_{j=1}^{Q_{\text{b1}}} \|\tilde{\boldsymbol{u}}^n(\boldsymbol{y}_j) - \boldsymbol{u}_{\text{b}}(\boldsymbol{y}_j)\|_1, \tag{19}$$

$$\mathcal{L}_{\text{b2}} = \frac{1}{Q_{\text{b2}}} \sum_{j=1}^{Q_{\text{b2}}} |\tilde{\boldsymbol{u}}^n(\boldsymbol{z}_j) \cdot \boldsymbol{n}_j - f(\boldsymbol{z}_j)|, \tag{20}$$

where $\boldsymbol{y}_1, \cdots \boldsymbol{y}_{Q_{\text{b1}}}, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_{Q_{\text{b2}}}$ are uniformly randomly sampled from the corresponding type of the domain boundary in each iteration of the optimization, $\boldsymbol{n}_j = \boldsymbol{n}(\boldsymbol{z}_j)$ are the normal vectors at the sampled points.

5.2.5 The Position Penalty. To sufficiently exploit the initial distribution provided by the previous advection step and prevent the particles from clustering, we add another regularization term to constrain their positions:

$$\mathcal{L}_{\text{pos}} = \frac{1}{Nd} \sum_{i=1}^{N} \|\boldsymbol{\mu}_i^n - \boldsymbol{\mu}_i^*\|^2. \tag{21}$$

The total loss optimizing the temporal evolution is the weighted combination of the vorticity loss, the divergence loss, the boundary loss, and the regularization terms:

$$\mathcal{L} = \mathcal{L}_{\text{vor}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}} + \lambda_{\text{b1}}\mathcal{L}_{\text{b1}} + \lambda_{\text{b2}}\mathcal{L}_{\text{b2}} + \lambda_{\text{aniso}}\mathcal{L}_{\text{aniso}} + \lambda_{\text{vol}}\mathcal{L}_{\text{vol}} + \lambda_{\text{pos}}\mathcal{L}_{\text{pos}}.$$

We then use the Adam to optimize the total loss, with the gradient of $\mathcal{L}_{\text{vor}}$ and $\mathcal{L}_{\text{div}}$ replaced by $\boldsymbol{g}_{\text{vor}}$ and $\boldsymbol{g}_{\text{div}}$ respectively.

5.3 Reseeding

Although the anisotropic regularization term is applied during projection, some Gaussian particles may inevitably become excessively elongated due to turbulent fluid motion. To address this, we introduce a reseeding procedure at the beginning of each time step. We split particles whose maximum scale is at least twice its minimum scale, i.e. split particle i if $\max(\boldsymbol{s}_i) \geq r_{\text{aniso}} \min(\boldsymbol{s}_i)$. The positions of the two new particles resulting from the split are sampled from the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_i^{n-1}, \Sigma_i^{n-1})$. Their maximum scales are halved, while the other parameters of the new particles are inherited from particle i. This is followed by a local fitting procedure where we optimize only the parameters of the new particles and their neighbors using the same loss function as during initialization (i.e. $\mathcal{L}_{\text{init}}$ from Equation 12). Note the number of split particles is influenced by λ_{aniso}, as a larger λ_{aniso} imposes a stronger penalty over the particles' anisotropy in the previous time step, reducing the number of elongated particles.

6 RESULTS

Table 1. Performance comparison. Timestep is measured in seconds. Running time (in seconds) indicates the solvers' average time cost. Memory usage indicates the average size (in KB) taken by the spatial representation.

Example	Timestep	Method	Running Time	Particle No.	Mem. Usage
Taylor-Green	0.001	INSR	403	-	32.1
		NMC	39	-	103.8
		Ours	38	576	17.9
Taylor vortex	0.01	INSR	378	-	32.1
		Ours	63	5041 ~ 5511	148.5
Leapfrog 2D	0.025	Ours	48	4846 ~ 5041	137.0
Vortices pass	0.01	Ours	47	5041 ~ 5273	143.1
Karman vortex street	0.05	NMC	81	-	134.3
		Ours	214	20408 ~ 24001	598.6
Leapfrog 3D	0.02	Ours	228	64000†	3252.2
Ring collide	0.02	Ours	206	64000†	3252.2
Smoking bunny	0.02	Ours	201	64000†	3252.2

† In all 3D examples, particle splitting does not occur due to sufficient initial particles.

We evaluate our method on a diverse set of 2D and 3D examples, with all results provided in the supplemental video. Notably, unlike most methods based on first-order optimization, our approach

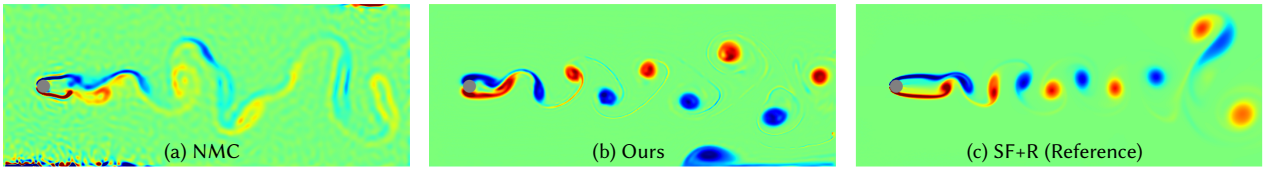


Fig. 4. Karman vortex street example by Neural Monte Carlo (NMC), our method, and stable fluids with reflection projection (SF+R) proposed by Zehnder et al. [2018]. The sub-figures display the vorticity fields of frame 152, 199 and 199 of the simulation results, respectively.

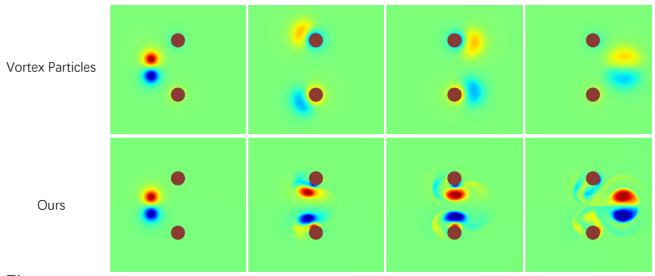


Fig. 5. Simulations of the vortices pass example produced by vortex particles with conserved harmonic components and our method. A pair of vortices pass through a gap between two spherical smooth objects. The images from left to right are showing frames 0, 150, 270 and 400, respectively.

does not require hyperparameter tuning across different scenarios, highlighting the robustness of GSR. To handle examples of varying scales, we rescale the entire fluid domain, boundary geometries, and initial velocity to a canonical size. With this adjustment, a single parameter setting suffices for all 2D examples and another for all 3D examples. Further details on hyperparameter settings, the normalization strategy, and scene configurations are available in the supplemental document.

We first show the numerical accuracy and convergence rate of GSR with a quantitative study on an example with analytical solution. We then validate its effectiveness by demonstrating the adaptive spatial accuracy on classical fluid phenomena. Next, we evaluate the stability of our optimization-based solver, in together with its boundary handling. Moreover, we demonstrate that GSR effectively preserves vortices and is capable of handling complex dynamic behaviors with highly intricate fluid simulations. Finally, multiple ablations are conducted to assess the contribution of key components in our method. The performance for all examples is provided.

6.1 Quantitative Study

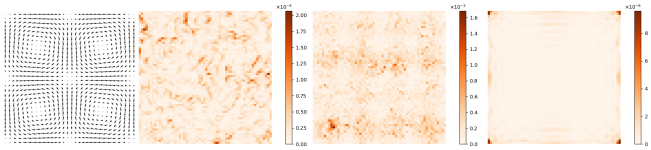
We conduct a quantitative study using the Taylor-Green vortex experiment, analyzing the numerical accuracy of our method and the convergence rates of the optimizations during initialization and time integration. In this example, the initial velocity field is set to

$$\boldsymbol{u}(x, y) = \begin{bmatrix} \sin x \cos y \\ -\cos x \sin y \end{bmatrix} \tag{22}$$

defined on the fluid domain $\mathcal{D} = [0, 2\pi] \times [0, 2\pi]$. We then run the simulation for 100 frames with a 0.001 seconds timestep.

Since the velocity field will remain constant on inviscid incompressible fluids, we measure the numerical error of different solvers by the mean squared error (MSE) between the simulated velocity fields and $\boldsymbol{u}(x, y)$ sampled on a 60×60 uniform grid at certain frames. The results in Figure 6 show that our method has significantly lower numerical error than INR-based methods, even with less memory usage as in Table 1. We also conduct a comparison against the semi-implicit Eulerian method run on a 64×64 MAC-grid (taking up 49KB each frame), which has the highest error among all as in Figure 6.

We plot the loss curves of the optimizations during initialization and a time integration process with time step size 0.01 seconds from the 100-th frame in Figure 6, which indicates fast convergence of our method. Figure 7 shows the loss curves for 400 iterations of the initialization optimization and 4100 iterations of the time-integration optimization. During simulation, we apply an early stop on the time-integration optimization if both vorticity and divergence loss do not change significantly for 500 iterations.



Frame	Eulerian	INSR	NMC	Ours
0	2.432×10^{-4}	8.998×10^{-7}	1.829×10^{-4}	9.957×10^{-8}
50	9.757×10^{-3}	1.715×10^{-5}	6.492×10^{-4}	2.510×10^{-7}
100	2.019×10^{-2}	1.992×10^{-5}	1.725×10^{-3}	2.181×10^{-7}

Fig. 6. Quantitative comparison using the Taylor-Green vortex example. Top-left: velocity field of the example. Top-right: the MSE fields at frame 100 of different methods. Bottom: MSE between the simulated velocity fields of different methods and the ground truth.

6.2 Validation

Efficacy of the GSR. We validate the efficacy of our proposed GSR by comparing the simulation results of Taylor vortex generated by our method, traditional methods with explicit representations, and INSR as another continuous representation. As shown in Figure 3,

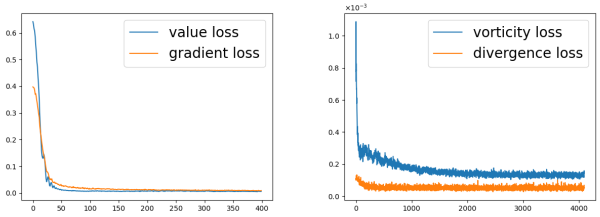


Fig. 7. Loss curves in the optimizations during initialization (left) and time integration (right).

the GSR preserves vorticity significantly better than all other methods, offering the best approximation to the ground truth, generated using a vortex-based 2D fluid solver. More frames are visualized in Figure 9. Compared to the Eulerian method with a 512×512 MAC grid, the GSR preserves the thin structure of the Taylor vortex with significantly reduced memory. When compared to the FLIP method with a 128×128 grid and 65536 particles, GSR demonstrates superior detail preservation and stability, using fewer than 5600 particles, highlighting the advantages of Gaussian kernels over traditional Lagrangian particles in both accuracy and stability. Our method also show far less numerical dissipation free of artifacts caused by particle deficiency compared to the SPH method. While both INR and GSR represent fluid details with lightweight data structures, our method outperforms INSR by better preserving vorticity and maintaining a cleaner background due to the locality of the Gaussian kernels. Additionally, our method is more time-efficient, as shown in Table 1, due to the high efficiency of differentiating GSR.

Effectiveness of Our Projection Optimization. Our projection step demonstrates stability in both obstacle-free and boundary-driven scenarios. We evaluate the efficiency of our optimization by comparing it to other optimization-based methods, INSR and NMC, on two classic phenomena: the Taylor vortex (without obstacles) and the Karman vortex street (boundary-driven). Compared to INSR, our method introduces less numerical error, yielding more accuracy, as shown in Figure 9. Unlike INSR, our projection step avoids solving for the pressure field or Poisson equation, eliminating the calculation for third-order derivatives in optimization and providing a six-fold performance boost, as shown in Table 1. Additionally, in the Karman vortex street example (Figure 4), our method demonstrates improved stability and accuracy, effectively handling boundary conditions in a long time horizon. While NMC encounters numerical instability at the domain boundary, our method successfully generates stable vortex shedding, closely matching the reference produced by stable fluids with reflection projection.

Harmonic Component Validation. Our method better handles the harmonic components without extra efforts comparing to Lagrangian vortex methods. Yin et al. [2023] noted that the nontrivial harmonic component of the velocity field evolves over time in non-simply connected fluid domains. Figure 5 illustrates a pair of vortices passing through a gap between two spherical obstacles with a no-slip boundary, where correctly modeling the harmonic component is key for successful traversal. We apply the Kirchoff point vortex dynamics and Kelvin's method of reflection for handling circular obstacles in the simulation by vortex particles, while maintaining conserved harmonic components using the same way as [Yin et al. 2023]. Unlike the vortex particles, our method successfully models the dynamics of the pair of vortices passing through the gap, indicating proper treatment of the harmonic components. However, our solution cannot strictly guarantee the satisfaction of the boundary condition or the divergence-free constraint, which we will discuss further in the supplementary material.

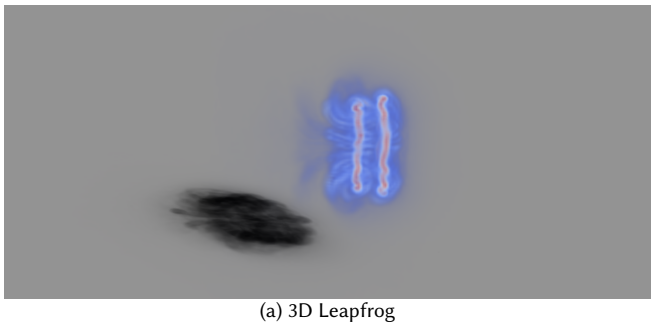
6.3 More Examples

Leapfrog 2D. Four vortices are placed at the bottom of the domain, with the left two vortices spin counterclockwise (i.e. with positive vorticity) and the right two spin clockwise (i.e. with negative vorticity). As shown in Figure 10, our solver is able to accurately simulate the alternating forward dynamics of two pairs of vortices and their eventual merging into a single vortex.

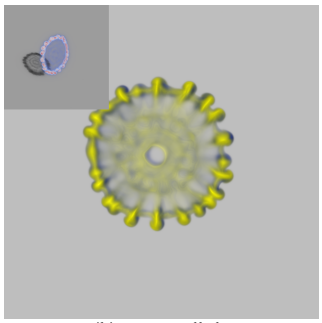
Leapfrog 3D. Figure 11 shows an example initialized with two parallel vortex rings facing the same direction. As they move forward, the vortex rings pass through each other, marking one leap. Our method successfully maintains the ring shapes after 3 leaps. However, numerical error is accumulated throughout the simulation, which is then converted by the physics-based optimization into low-divergence velocity components, resulting in small vortex-ring artifacts near the end of our 3D simulations. Reducing the accumulated error is worth future exploration.

Ring Collide. In this example, two parallel vortex rings face each other at the initial frame. As shown in Figure 1, the rings expand as they approach each other, followed by shredding into many small vortex rings upon collision. Figure 12 shows the rendering of passive smoke rings advected by the velocity field. We also compare it with a real-world recording in our video.

Smoking Bunny. To demonstrate that our method is capable of handling boundary with a complex geometry, we release two vortex rings towards the face of the Stanford Bunny. As shown in Figure 13, the vortex rings deforms and breaks down as they hit the uneven surface, immersing the bunny in a foggy environment.



(a) 3D Leapfrog



(b) Ring Collide

(c) Bunny

Fig. 8. We test our method in complex scenes, demonstrating the advantages of stability and spatial details.

6.4 Ablation Tests

Particle Splitting. Figure 14 compares the method without particle splitting with the full method. The former fails to preserve the thin filament of the Taylor vortex due to limited expressiveness. Our reseeding strategy effectively addresses this without introducing significant overhead, as shown by the particle count in Table 1. Meanwhile, we do not observe any change in convergence rate of the following optimization caused by the reseeding process.

Gradient Projection. Without the gradient projection technique outlined in Section 5.2.3, directly combining the losses can be difficult to optimize. The gradient of the divergence loss may increase the vorticity loss, leading to ripple artifacts in the vorticity field, as shown in Figure 15a. However, by applying the gradient projection, we achieve a clean result, as seen in Figure 15b.

Advection-Based Initial Guess. The initial guess improves both solver accuracy and convergence speed of our method. Figure 16 compares our method with and without the initial guess, in which case we directly optimize the GSR from its configuration at the end of the last time step. For an intuitive comparison, we set up a background grid moving at the inflow speed with two columns marked in a darker color. In Figure 16a, the flow rate is overly rapid, as evidenced by the vortex originally inside the darker zone moving out of it. In contrast, the flow rate in Figure 16b closely matches the inflow. Additionally, without advection, the projection step requires an average of 4661.3 iterations, compared to 3902.5 iterations with advection.

6.5 Performance

We run all experiments on a NVIDIA GeForce RTX 4090 GPU. Table 1 shows the time cost and memory usage for all examples, along with a comparison to INR-based methods. Our method exhibits comparable time and memory usage, while delivering superior quality and stability. Note the running times shown in Table 1 are all much longer than those used by traditional Lagrangian and Eulerian methods, since both GSR and INR-based methods require costly first-order optimizations.

7 CONCLUSION

We have presented a novel grid-free fluid solver based on a Gaussian spatial representation. Compared to established grid-based fluid solvers, our framework offers the advantages of enhanced spatial details and effective vorticity preservation over time. When compared to continuous implicit representations, our method excels in handling boundary phenomena. Additionally, our approach is more stable, scalable, and efficient, supporting long time horizons with consistent parameters, making it a promising tool for both 2D and 3D fluid simulations.

Despite these strengths, our method has some limitations. First, relying on soft constraints to solve the Navier-Stokes equations introduces small residual errors in divergence and boundary conditions, which may lead to discrepancies in global fluid behavior when compared to grid-based solvers. Additionally, while harmonic components are implicitly preserved, they are not explicitly modeled with time consistency, which may result in inaccuracies when simulating dynamics in non-simply connected domains. Future work will focus on incorporating hard constraints and improving the modeling of harmonic components. Furthermore, we plan to explore inverse problems, such as key-frame-based fluid control, leveraging the efficient gradient calculation capabilities of our representation.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments.

REFERENCES

Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Trans. Graph.* 36, 4, Article 140 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073625

Stefan Band, Christoph Gissler, Andreas Peer, and Matthias Teschner. 2018. MLS pressure boundaries for divergence-free and viscous SPH fluids. *Computers & Graphics* 76 (2018), 37–46.

Jan Bender and Dan Koschier. 2016. Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2016), 1193–1206.

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Citeseer, 87–95.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. 2021. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems* 34 (2021), 21557–21568.

Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. 2023. Implicit neural spatial representations for time-dependent PDEs. In *International Conference on Machine Learning*. PMLR, 5162–5177.

Georges-Henri Cottet, Petros D Koumoutsakos, et al. 2000. *Vortex methods: theory and practice*. Vol. 313. Cambridge university press Cambridge.

Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Trans. Graph.* 42, 6 (2023).

Xin Dong, Ruize Wu, Chao Xiong, Hai Li, Lei Cheng, Yong He, Shiyou Qian, Jian Cao, and Linjian Mo. 2022. GDOD: Effective Gradient Descent using Orthogonal Decomposition for Multi-Task Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 386–395. https://doi.org/10.1145/3511808.3557333

Fan Feng, Jinyuan Liu, Shiying Xiong, Shuqi Yang, Yaorui Zhang, and Bo Zhu. 2023a. Impulse Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023), 3081–3092. https://doi.org/10.1109/TVCG.2022.3149466

Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, and Yin Yang. 2024. Gaussian Splashing: Dynamic Fluid Synthesis with Gaussian Splatting. arXiv:2401.15318v1 [cs.GR] https://arxiv.org/abs/2401.15318v1

Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. 2023b. PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF. arXiv:2311.13099 [cs.CV]

Nick Foster and Dimitri Metaxas. 1996. Realistic animation of liquids. *Graphical models and image processing* 58, 5 (1996), 471–483.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2013. Implicit incompressible SPH. *IEEE transactions on visualization and computer graphics* 20, 3 (2013), 426–435.

Pranav Jain, Ziyin Qu, Peter Yichen Chen, and Oded Stein. 2024. Neural Monte Carlo Fluid Simulation. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/3641519.3657438

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics* 3, 6 (2021), 422–440.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-1gaussian-splatting/

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. 2005. FlowFixer: using BFECC for fluid simulation. In *Proceedings of the First Eurographics conference*

on Natural Phenomena. 51–56.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2019. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. *Eurographics 2019 - Tutorials* (2019). https://doi.org/10.2312/EGT.20191035

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2022. A Survey on SPH Methods in Computer Graphics. *Computer Graphics Forum* 41, 2 (2022). https://doi.org/10.1111/cgf.14508

Petros D Koumoutsakos, Georges-Henri Cottet, and Diego Rossinelli. 2008. Flow simulations using particles-Bridging Computer Graphics and CFD. In *SIGGRAPH 2008-35th International Conference on Computer Graphics and Interactive Techniques*. ACM, 1–73.

Qiang Liu, Mengyu Chu, and Nils Thuerey. 2025. ConFIG: Towards Conflict-free Training of Physics Informed Neural Networks. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=APojAzJQiq

A. McAdams, E. Sifakis, and J. Teran. 2010. A parallel multigrid Poisson solver for fluids simulation on large grids. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar, DEU, 65–74.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Citeseer, 154–159.

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector Fluids. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 113:1–113:15.

J. Michael Owen, Jens V. Villumsen, Paul R. Shapiro, and Hugo Martel. 1998. Adaptive Smoothed Particle Hydrodynamics: Methodology. II. *The Astrophysical Journal Supplement Series* 116, 2 (June 1998), 155–209. https://doi.org/10.1086/313100

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

Igor Santesteban, Miguel Otaduy, Nils Thuerey, and Dan Casas. 2022. ULNeF: untangled layered neural fields for mix-and-match virtual try-on. *Advances in Neural Information Processing Systems* 35 (2022), 12110–12125.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35 (2008), 350–371.

Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. https://doi.org/10.1145/311535.311548

Sinan Wang, Yitong Deng, Molin Deng, Hong-Xing Yu, Junwei Zhou, Duowen Chen, Taku Komura, Jiajun Wu, and Bo Zhu. 2024. An Eulerian Vortex Method on Flow Maps. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–13.

Sifan Wang, Xinling Yu, and Paris Perdikaris. 2022. When and why PINNs fail to train: A neural tangent kernel perspective. *J. Comput. Phys.* 449 (2022), 110768.

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–12.

Lukas Westhofen, Stefan Jeske, and Jan Bender. 2023. A comparison of linear consistent correction methods for first-order SPH derivatives. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (2023), 1–20.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2023. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. *arXiv preprint arXiv:2311.12198* (2023).

Lingchen Yang, Gaspard Zoss, Prashanth Chandran, Paulo Gotardo, Markus Gross, Barbara Solenthaler, Eftychios Sifakis, and Derek Bradley. 2023. An Implicit Physical Face Model Driven by Expression and Style. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

Hang Yin, Mohammad Sina Nabizadeh, Baichuan Wu, Stephanie Wang, and Albert Chern. 2023. Fluid Cohomology. *ACM Trans. Graph.* 42, 4, Article 126 (July 2023), 25 pages. https://doi.org/10.1145/3592402

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 5824–5836. https://proceedings.neurips.cc/paper_files/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Trans. Graph.* 37, 4, Article 85 (July 2018), 8 pages. https://doi.org/10.1145/3197517.3201324

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.

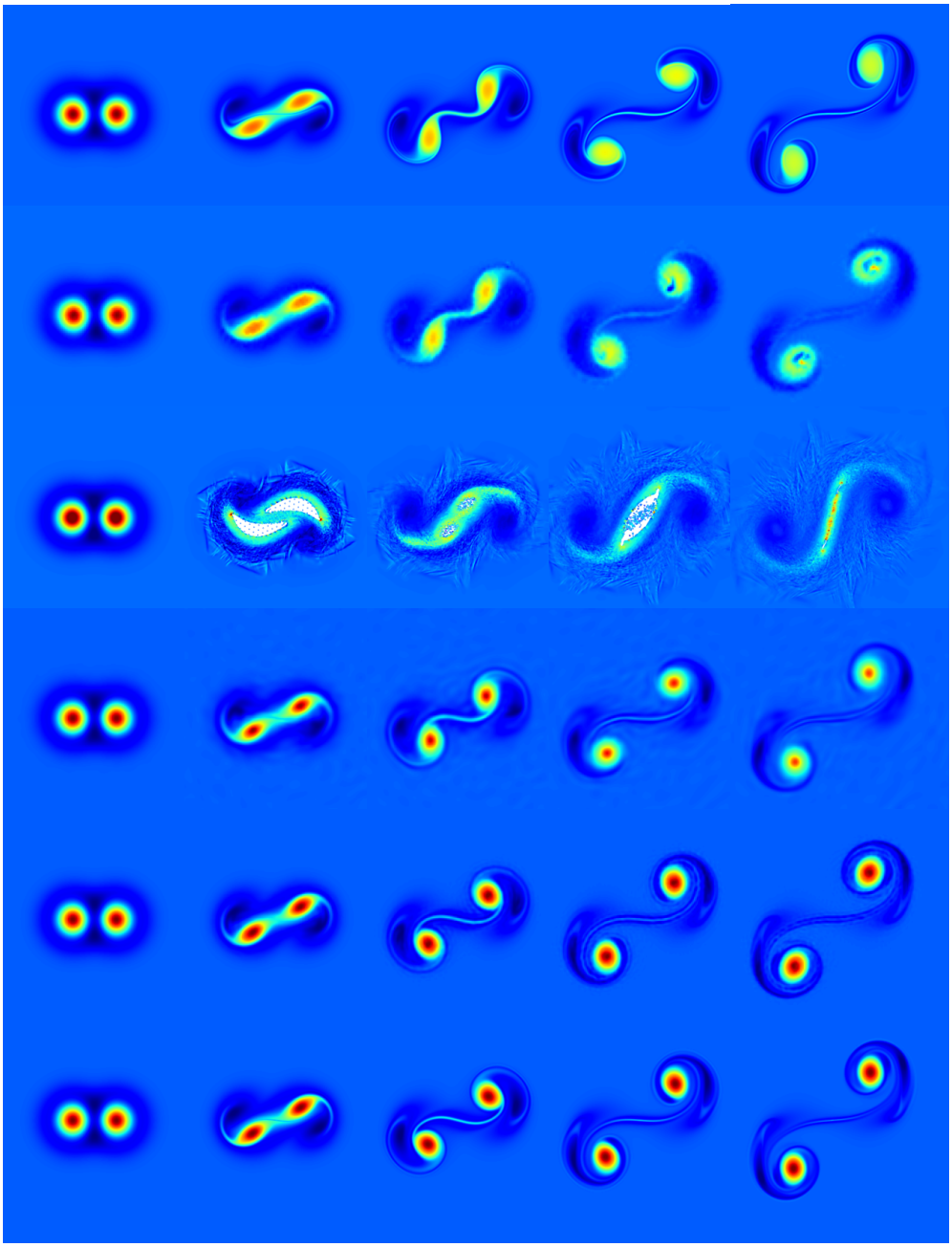


Fig. 9. We compare our method with different traditional spatial discretizations and INSR using the Taylor vortex example. We show the average file size of the according representation of the velocity fields on the first column, enclosed by parenthesis. Images show the vorticity field at frame 0, 100, 200, 300, and 399, respectively. Note the initial frame of the SPH results is sightly different from others since the vorticity on each particle is calculated with the differential operators in the SPH convention.

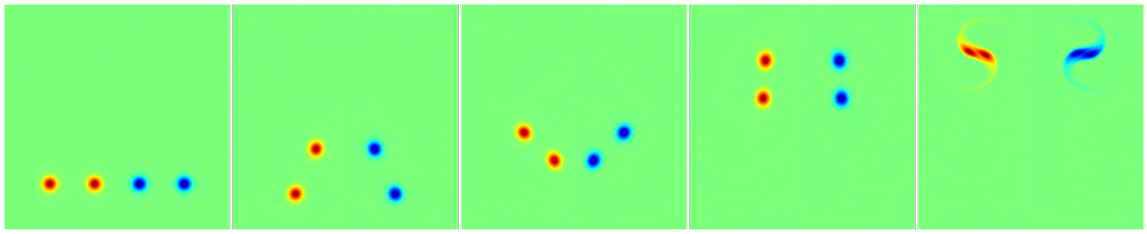


Fig. 10. Our simulation results on the 2D leapfrog example. The figures are showing frames 0, 165, 456, 1050 and 1500 from left to right.

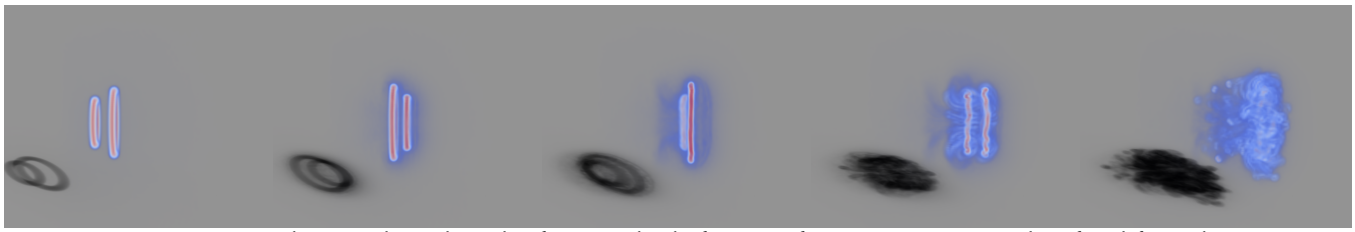


Fig. 11. Our simulation results on the 3D leapfrog example. The figures are frames 0, 148, 332, 548, and 860 from left to right.

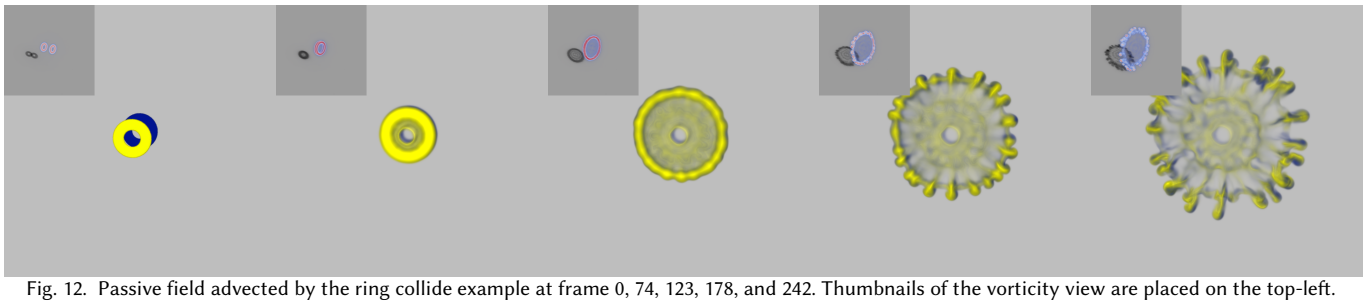


Fig. 12. Passive field advected by the ring collide example at frame 0, 74, 123, 178, and 242. Thumbnails of the vorticity view are placed on the top-left.

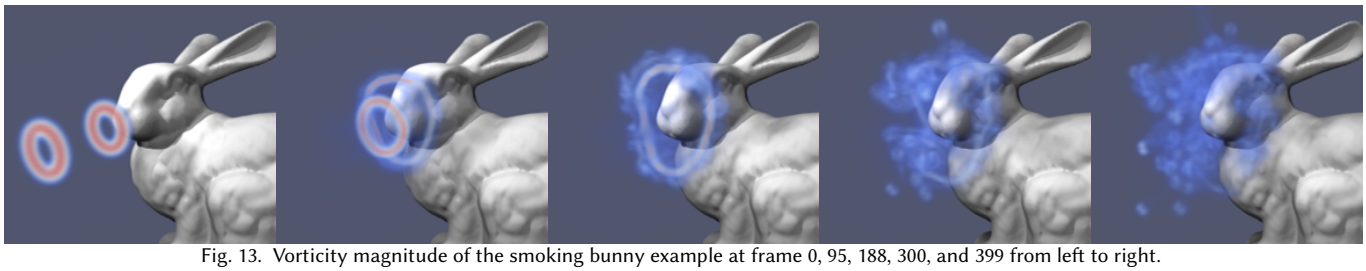
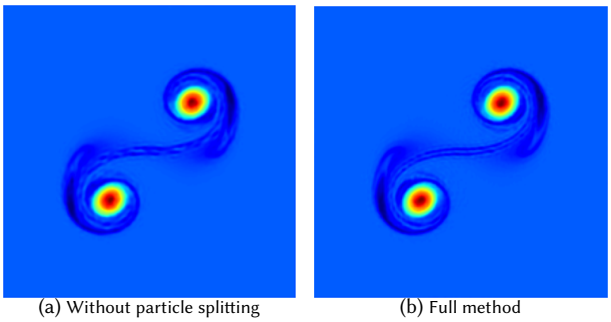
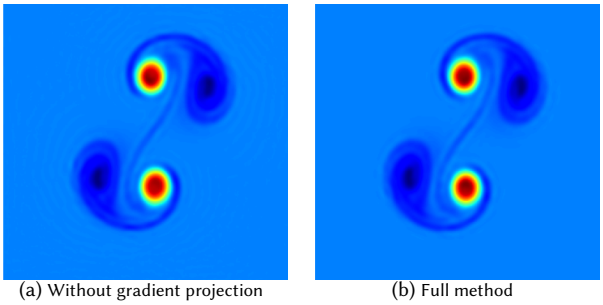


Fig. 13. Vorticity magnitude of the smoking bunny example at frame 0, 95, 188, 300, and 399 from left to right.



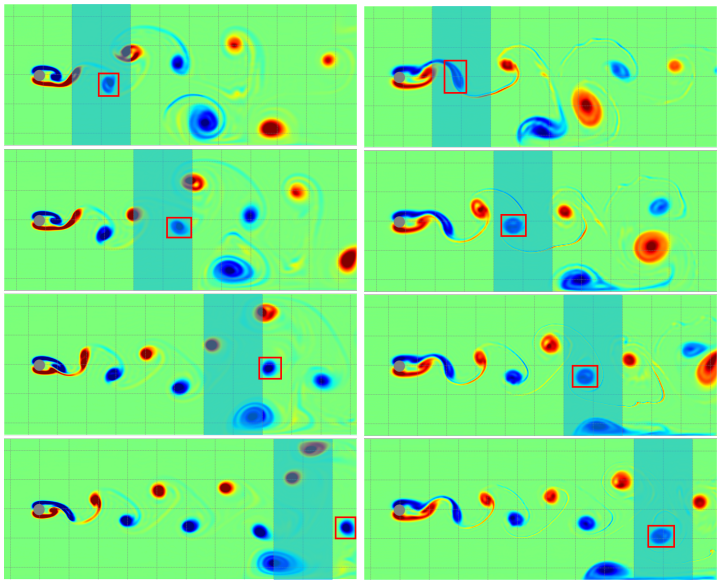
(a) Without particle splitting (b) Full method

Fig. 14. Ablation test on particle splitting. We show vorticity fields at frame 380.



(a) Without gradient projection (b) Full method

Fig. 15. Ablation test on the gradient projection technique. We show vorticity fields at frame 36 with a time step of 0.1 seconds.



(a) Without initial guess (b) Full method

Fig. 16. Ablation test on the advection-based initial guess. The images from top to bottom are simulation results at frames 127, 150, 174 and 198, respectively. The background grid in gray dashed line is moving forward at the inflow speed. We track one vortex (marked by the red boxes) to illustrate the flow rate of the simulation w.r.t. the speed of the grid.