An Adjoint Method for Differentiable Fluid Simulation on Flow Maps

ZHIQI LI*, Georgia Institute of Technology, USA
JINJIN HE*, Georgia Institute of Technology, USA
BARNABÁS BÖRCSÖK, Georgia Institute of Technology, USA
TAIYUAN ZHANG, Dartmouth College, USA
DUOWEN CHEN, Georgia Institute of Technology, USA
TAO DU, Independent Researcher,
MING C. LIN, University of Maryland, USA
GREG TURK, Georgia Institute of Technology, USA
BO ZHU, Georgia Institute of Technology, USA

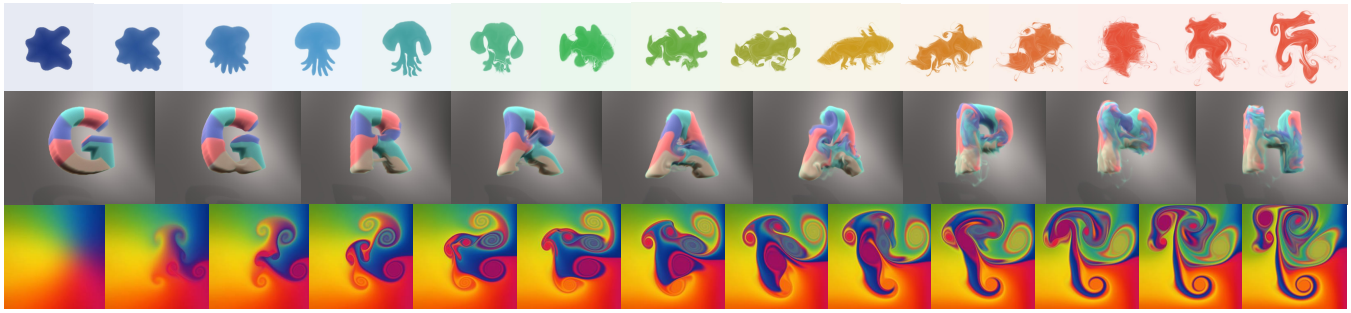


Fig. 1. Demonstration of our differentiable fluid simulation on flow maps using our adjoint solver: *Top:* A sequence of 2D fluid shape optimizations demonstrating smooth morphing between target silhouettes. *Middle:* 3D fluid control with multiple keyframes to guide a 3D letter morphing from "G" to "R" to "A" to "P" to "H". *Bottom:* A vortex dynamics inference task that predicts future flow evolution from a sequence of observed past images.

This paper presents a novel adjoint solver for differentiable fluid simulation based on bidirectional flow maps. Our key observation is that the forward fluid solver and its corresponding backward, adjoint solver share the same flow map as the forward simulation. In the forward pass, this map transports fluid impulse variables from the initial frame to the current frame to simulate vortical dynamics. In the backward pass, the same map propagates adjoint variables from the current frame back to the initial frame to compute gradients. This shared long-range map allows the accuracy of gradient computation to benefit directly from improvements in flow map construction. Building on this insight, we introduce a novel adjoint solver that solves the adjoint equations directly on the flow map, enabling long-range and accurate differentiation of incompressible flows without differentiating intermediate numerical steps or storing intermediate variables, as required in conventional adjoint methods. To further improve efficiency, we propose a long-short time-sparse flow map representation for evolving adjoint variables. Our approach has low memory usage, requiring only 6.53GB of data at a resolution of 192^3 while preserving high accuracy in tracking vorticity, enabling new differentiable simulation tasks that require precise identification, prediction, and control of vortex dynamics.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Fluid Simulation, Adjoint Method, Flow Map Method, Differentiable Fluid Simulation

ACM Reference Format:
Zhiqi Li, Jinjin He, Barnabás Börcsök, Taiyuan Zhang, Duowen Chen, Tao Du, Ming C. Lin, Greg Turk, and Bo Zhu. 2025. An Adjoint Method for Differentiable Fluid Simulation on Flow Maps. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3757377.3763903

*Joint first author

Authors' Contact Information: Zhiqi Li, zli3167@gatech.edu, Georgia Institute of Technology, USA; Jinjin He, jhe433@gatech.edu, Georgia Institute of Technology, USA; Barnabás Börcsök, borcsok@gatech.edu, Georgia Institute of Technology, USA; Taiyuan Zhang, taiyuan.zhang.gr@dartmouth.edu, Dartmouth College, USA; Duowen Chen, dchen322@gatech.edu, Georgia Institute of Technology, USA; Tao Du, taodu.eecs@gmail.com, Independent Researcher,; Ming C. Lin, lin@umd.edu, University of Maryland, USA; Greg Turk, turk@cc.gatech.edu, Georgia Institute of Technology, USA; Bo Zhu, bo.zhu@gatech.edu, Georgia Institute of Technology, USA.

SA Conference Papers '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/2025/12
https://doi.org/10.1145/3757377.3763903

1 Introduction

Accurately differentiating a dynamic fluid system, particularly computing the derivatives of fluid variables over a long time horizon, remains a fundamental challenge in both computer graphics and computational physics. The primary difficulty arises from the intrinsic flow nature of fluids: unlike solid systems with more constrained

configurations, fluid systems evolve freely over space and time under physical laws, giving rise to a high-dimensional, continuously deforming state space that significantly complicates the backward differentiation process. As the simulation progresses over longer time periods (either forward or backward), numerical errors accumulate at each timestep, further degrading the accuracy of gradient computation and making the long-horizon derivative estimation increasingly unreliable.

Two mainstream approaches have been developed for differentiating fluid systems governed by the Navier–Stokes equations in both computer graphics and computational physics. One class of methods directly differentiates the discrete numerical scheme used in the forward simulation. A representative example is the pioneering work by McNamara et al. [2004], which employed a classical advection-projection scheme [Stam 1999] in the forward process and computes gradients by sequentially differentiating the advection and projection steps, where an adjoint system is solved to account for the projection. This method has been highly successful within the graphics community and has inspired a substantial body of follow-up work (e.g., see Holl and Thuerey [2024]; Li et al. [2024c]; Takahashi et al. [2021]). Since this approach directly targets the discrete formulation used in the forward solver, it ensures that the computed gradients are fully consistent with the actual simulation steps, which is critical for using gradient information to guide optimization processes (e.g., in control, animation, or design problems). The other class of methods derives the adjoint system analytically at the level of the continuous governing equations, followed by discretization of the resulting adjoint PDEs. Such approaches have been widely adopted in computational fluid dynamics for solving inverse problems (e.g., see Gałecki and Szumbarski [2022]; Stück [2012]). However, to ensure that the discrete adjoint solution accurately corresponds to the derivative of the discrete forward simulation, these methods often rely on high-order discretization, particularly in their advection schemes and spatial operators, which limits their practicality in visual computing scenarios where computational efficiency and scalability are critical.

We propose a new adjoint solver that improves both the accuracy and efficiency of existing methods for differentiable incompressible flow simulation. Our approach is built upon the concept of long-range bidirectional flow maps, which have recently emerged as an effective modeling framework for simulating a wide range of fluid systems and their multiphysics couplings dominated by vortical dynamics (e.g., see [Chen et al. 2024b; Deng et al. 2023b; Li et al. 2024b; Zhou et al. 2024] for examples). The core idea of the flow-map method is to construct a mapping between the initial time and the current time that accurately transports physical quantities between corresponding spatial locations. The term "bidirectional" refers to the capability of transporting quantities both forward and backward in time, with the forward and backward maps forming a consistent and temporally symmetric pair. A key observation underlying our work is that this bidirectional flow map, originally introduced to enhance the accuracy of forward simulation, can be naturally repurposed to support the backward adjoint process. Sharing the same flow map across both forward and backward processes enables temporally symmetric transport of fluid quantities and their adjoints over extended time intervals, which is a proven strength of flow-map-based formulations. Leveraging this accuracy, our method eliminates the need to differentiate individual numerical steps, thereby avoiding the high memory and computational overhead associated with discrete differentiation, and instead enables direct solution of the continuous adjoint PDEs with improved scalability and precision.

Motivated by this idea, we developed a novel adjoint solver grounded in flow map theory to enable long-range, accurate differentiation of incompressible flow systems. Our system comprises three key components: (1) a forward incompressible fluid solver based on bidirectional flow maps discretized over a sequence of grid-aligned frames; (2) a backward adjoint solver that solves the adjoint equations using the same flow maps as in the forward process; and (3) an acceleration strategy based on a long-short time-sparse flow map representation to reduce computational cost without sacrificing accuracy. The forward and backward solvers not only share the same grid-based bidirectional flow maps but also apply the same numerical scheme to evolve fluid quantities and their adjoints, respectively, along opposite time directions.

2 Related Work

Differentiable Fluid Simulation. Differentiable fluid simulation in computer graphics typically computes gradients by differentiating the discretized forward simulation. The pioneering works by Treuille et al. [2003] and McNamara et al. [2004] differentiates the discretized advection-projection fluid simulation method [Stam 1999] and has inspired a line of subsequent works [Holl et al. 2020; Holl and Thuerey 2024; Li et al. 2024c; Pan and Manocha 2017; Takahashi et al. 2021]. Previous works have also studied differentiating smoothed-particle hydrodynamics (SPH) [Li et al. 2023b], reduced-mode fluids [Chen et al. 2024a], and the lattice-boltzmann method (LBM) [Ataei and Salehipour 2024]. These techniques have enabled progress in fluid control and optimization, but they can suffer from limited accuracy or high memory cost when applied to long-horizon simulations. In contrast, our work does not differentiate a discretized fluid simulator but directly discretize the continuous adjoint PDEs of the Navier-Stokes equations, enabling more accurate numerical solutions to the adjoint equations.

Adjoint Methods. The adjoint method is a standard mathematical tool for computing gradients in PDE-constrained optimization. Studies in computational fluid dynamics (CFD) typically apply it to derive the adjoint Navier-Stokes equations [Giles et al. 2003; Giles and Pierce 2000; Jameson 1988; Stück 2012], enabling sensitivity analysis through backward-time evolution. Beyond fluids, adjoint formulations underpin a broad class of differentiable physics frameworks, where gradients of physical systems are leveraged for inverse design, control, and optimization. Applications span elastic materials [Du et al. 2021; Geilinger et al. 2020; Hu et al. 2019b; Qiao et al. 2021a], cloth simulation [Li et al. 2022; Qiao et al. 2020], contact and collision [Huang et al. 2024a,b], magnetic shells [Chen et al. 2022], and topology optimization [Feng et al. 2023; Liu et al. 2018; Sigmund 2001; Zhu et al. 2017]. These differentiable physics systems enable diverse applications including robot design [Gjoka et al. 2024; Ma et al. 2021], surface optimization [He et al. 2024; Mehta et al. 2022; Montes Maestre et al. 2023], parameter identification

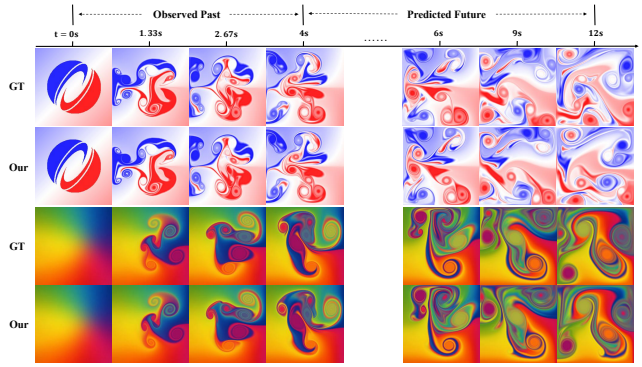


Fig. 2. **Vortex dynamics inference from velocity-field videos.** Training on the first 4 seconds infers 8 random vortices, maintaining accuracy over extended 12-second predictions.

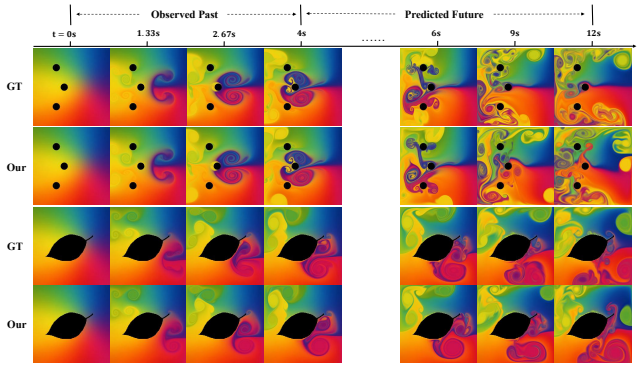


Fig. 3. **Vortex dynamics with obstacle interference.** Method successfully infers vortices with vortex-obstacle interactions, enabling accurate long-term flow prediction around geometric constraints.

[Hahn et al. 2019; Li et al. 2023a; Ma et al. 2022], microstructure discovery [Huang et al. 2024a; Sigmund 2001], and policy learning [Huang et al. 2021; Li et al. 2018; Qiao et al. 2021b; Zhou et al. 2023], typically using gradient-based optimizers like Adam [Kingma 2014] or LBFGS [Nocedal and Wright 1999].

Flow Map Methods. Flow map methods trace their origins to the method of characteristic mapping (MCM) by Wiggert and Wylie [1976], later developed in computer graphics by Tessendorf and Pelfrey [2011] and Qu et al. [2019]. Recent progress on representing a bidirectional map includes neural network-based storage compression [Deng et al. 2023b], buffer-free Eulerian representations [Li et al. 2025b], and the particle flow map method [Chen et al. 2025; Li et al. 2024b, 2025a; Wang et al. 2025; Zhou et al. 2024], which have further enhanced accuracy. Gauge-based fluid formulations [Buttke 1992; Cortez 1996] have been explored with various applications [Feng et al. 2022; Li et al. 2024a; Nabizadeh et al. 2022]. Despite their accuracy advantages, flow map methods suffer from high computational complexity, with traditional Eulerian flow map (EFM) method [Deng et al. 2023b] requiring $O(n^2)$ flow map evolution costs, recently addressed by time-sparse approaches [Sun et al. 2025].

3 Physical Model

3.1 Differentiable Fluid

Fluid Equations. We focus on the incompressible Navier–Stokes equations and the advection of a passive field for fluid simulation:

$$\left(\frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla)\right)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u} + \mathbf{f}, \qquad \nabla\cdot\mathbf{u} = 0, \tag{1}$$

$$\left(\frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla)\right)\xi = 0, \tag{2}$$

where p, $\mathbf{f}$, $\nu = \frac{\mu}{\rho}$ denote the pressure field, external force, and the kinematic viscosity, respectively. The scalar field ξ represents a passive quantity field advected by the fluid, such as smoke density or color. Given the velocity field $\mathbf{u}_{s'}$ and passive field $\xi_{s'}$ at arbitrary start time $s' \geq s$, Equation 1 determines their exact evolution for any $t \geq s'$ with well-defined boundary conditions, denoted as $(\mathbf{u}_t, \xi_t) = \mathbf{F}_{s'\to t}(\mathbf{u}_{s'}, \xi_{s'})$. The fluid simulation method approximates this process numerically as $(\hat{\mathbf{u}}_t, \hat{\xi}_t) = \hat{\mathbf{F}}_{s'\to t}(\mathbf{u}_{s'}, \xi_{s'})$, where $\hat{}$ denotes numerical approximation.

Adjoint Equations. In fluid-related optimization problems, we aim to minimize an objective functional

$$L(\mathbf{u}, \xi) = \int_s^r \int_{\mathbb{U}_t} J(\mathbf{u}, \xi, t)\, d\mathbf{x}dt, \tag{3}$$

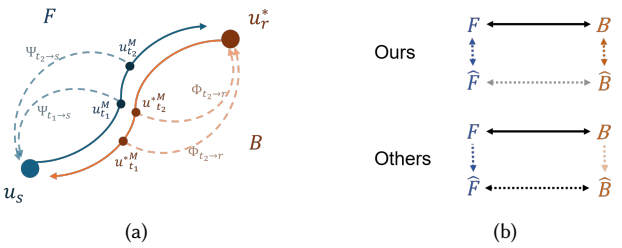
where the objective functional integrand J is a time-dependent functional of the velocity field $\mathbf{u}$ and the passive scalar ξ, for example, the terminal velocity loss $J(\mathbf{u}, \xi, t) = \delta(t-r)\|\mathbf{u} - \mathbf{u}_{\text{target}}\|_2^2$, where $\mathbf{u}_{\text{target}}$ and δ denote the target velocity field and the Dirac delta function, respectively. When applying common optimization methods such as gradient descent to minimize L, it is necessary to compute the gradient information $\mathbf{u}_t^* = \frac{\partial L}{\partial \mathbf{u}_t}$ and $\xi_t^* = \frac{\partial L}{\partial \xi_t}$, which are referred to as the adjoints of $\mathbf{u}_t$ and ξ_t, respectively. [Gałecki and Szumbarski 2022; Stück 2012] shows that $\mathbf{u}_t^*$ and ξ_t^* follow the equations:

$$\left(\frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla)\right)\mathbf{u}^* = \nabla\mathbf{u}^\top\mathbf{u}^* + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p^* + \nu\Delta\mathbf{u}^* - \frac{\partial J}{\partial \mathbf{u}}, \qquad \nabla\cdot\mathbf{u}^* = 0, \qquad \left(\frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla)\right)\xi^* = -\frac{\partial J}{\partial \xi}, \tag{4}$$

where p^* is the adjoint pressure. Here we assume that the external force $\mathbf{f}$ is independent of $\mathbf{u}$ and ξ, and more general cases can be derived using the adjoint method. Given the adjoint velocity field $\mathbf{u}_{r'}^*$ and passive field $\xi_{r'}^*$ at time $r' \leq r$, Equation 4 determine their exact solution for any $t \leq r'$ with boundary conditions, denoted as $(\mathbf{u}_t^*, \xi_t^*) = \mathbf{B}_{r'\to t}(\mathbf{u}_{r'}^*, \xi_{r'}^*)$. Differentiable fluid simulation aims at approximating this process numerically as $(\hat{\mathbf{u}}_t^*, \hat{\xi}_t^*) = \hat{\mathbf{B}}_{r'\to t}(\mathbf{u}_{r'}^*, \xi_{r'}^*)$.

3.2 Method Overview

We address fluid-related optimization problems using the flow map method, aiming to optimize parameters θ so that the resulting fluid



(a) (b)

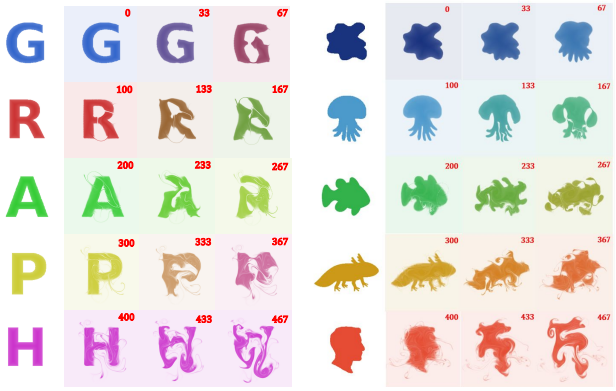
Fig. 4. **Method overview.** In (a), we illustrate the symmetry between the forward and backward passes. The forward pass maps $\mathbf{u}$ using the backward flow map Ψ, while the backward pass maps $\mathbf{u}^*$ using the forward flow map Φ. Both Ψ and Φ are opposite to the flow direction and require repeated long-range integration for accuracy, leading to the original EFM's $O(m^2)$ time complexity, where m is the flow map length. In (b), we compare our method with other differentiable approaches. Due to lower accuracy, existing methods indirectly approximate $\mathbf{B} \to \hat{\mathbf{B}}$ (semi-transparent one-way arrows) through approximating $\mathbf{F} \to \hat{\mathbf{F}}$ (dashed one-way arrows) and directly differentiating $\hat{\mathbf{F}}$ (dashed double arrows). Although $\hat{\mathbf{B}}$ is consistent with $\hat{\mathbf{F}}$, $\mathbf{B} \to \hat{\mathbf{B}}$ is inaccurate. In contrast, our method leverages the strict correspondence between $\mathbf{F}$ and $\mathbf{B}$. We only need to construct accurate approximations $\mathbf{F} \to \hat{\mathbf{F}}$ and $\mathbf{B} \to \hat{\mathbf{B}}$ respectively (dashed double arrows), and the consistency between $\hat{\mathbf{F}}$ and $\hat{\mathbf{B}}$ then naturally follows through transitivity (semi-transparent dashed double arrows), enabled by the higher accuracy of flow maps.

states $\mathbf{u}_t^\theta(\mathbf{x})$ and $\xi^\theta(\cdot)$ minimize the objective functional $L(\mathbf{u}^\theta, \xi^\theta)$ under specific control scenarios and subject to constraints of fluid dynamics (Equation 1). The process iteratively performs forward simulation $\hat{\mathbf{F}}_{s\to r}$, evaluates the functional $L(\mathbf{u}^\theta, \xi^\theta)$, computes the backward adjoint process $\hat{\mathbf{B}}_{r\to s}$ to obtain adjoints $\mathbf{u}_t^{*\theta}$ and $\xi_t^{*\theta}$, and updates the control parameters θ using these adjoints.

To obtain the scheme for $\hat{\mathbf{B}}_{r\to s}$, unlike previous differentiable fluid solvers that differentiate the discretized forward process $\hat{\mathbf{F}}_{s\to t}$, our approach directly computes the continuous backward process $\mathbf{B}_{r\to t}$ via flow maps, yielding a principled adjoint formulation and establishing a symmetric forward–backward framework that uses flow maps for the consistent evolution of states and adjoints. A high-level comparison and overview of this computation are shown in Fig. 4. We then introduce the flow map method (subsection 3.3), describe its use for forward (subsection 4.1) and adjoint computation (subsection 4.2), present a novel acceleration strategy (subsection 4.3), and finally assemble the complete numerical scheme (section 5), which we subsequently employ to solve fluid optimization problems (section 6).

3.3 Flow Map

In fluid simulation, the flow map method [Deng et al. 2023b; Zhou et al. 2024] enables accurate advection of physics quantities by constructing a mapping between the initial domain $\mathbb{U}_s$ and the current domain $\mathbb{U}_t, r > t > s$ where s and r are the initial time and the final time respectively. Consider a fluid moving with a velocity field $\mathbf{u}(\mathbf{x}, t), \mathbf{x} \in \mathbb{U}_t$. For any time $t_1 < t_2$, the forward flow map $\Phi_{t_1\to t_2} : \mathbb{U}_{t_1} \to \mathbb{U}_{t_2}$ and backward flow map $\Psi_{t_2\to t_1} : \mathbb{U}_{t_2} \to \mathbb{U}_{t_1}$ are defined as functions satisfying $\Phi_{t_1\to t_2}(\mathbf{x}_q(t_1)) = \mathbf{x}_q(t_2)$ and $\Psi_{t_2\to t_1}(\mathbf{x}_q(t_2)) = \mathbf{x}_q(t_1)$ for any fluid particle q moving with $\frac{d\mathbf{x}_q(t)}{dt} = \mathbf{u}(\mathbf{x}_q(t), t)$, where $\mathbf{x}_q(t)$ denotes position of particle q



(a) Letter Morphing (b) Life-Form Evolution

Fig. 5. **2D sequential optimizations.** A sequence of 2D morphing tasks, including letter morphing ('G'→'R'→'A'→'P'→'H') and life-form evolution, demonstrating smooth transitions between target silhouettes. Each row illustrates the progressive transformation between two consecutive keyframes, with target shapes shown on the left.

at time t. The Jacobian matrices of the flow maps are denoted as $\mathcal{F}_{t_1\to t_2}(\mathbf{x}) = \frac{\partial \Phi_{t_1\to t_2}(\mathbf{x})}{\partial \mathbf{x}}, \mathbf{x} \in \mathbb{U}_{t_1}$ and $\mathcal{T}_{t_2\to t_1}(\mathbf{x}) = \frac{\partial \Psi_{t_2\to t_1}(\mathbf{x})}{\partial \mathbf{x}}, \mathbf{x} \in \mathbb{U}_{t_2}$, respectively. From start time $s' \geq s$ selected arbitrarily, flow maps and their Jacobians follow evolution equations:

$$\begin{cases} \frac{\partial \Phi_{s'\to t}(\mathbf{x})}{\partial t} = \mathbf{u}(\Phi_{s'\to t}(\mathbf{x}), t), & \Phi_{s'\to s'}(\mathbf{x}) = \mathbf{x}, \\ \frac{\partial \mathcal{F}_{s'\to t}(\mathbf{x})}{\partial t} = \nabla \mathbf{u}(\Phi_{s'\to t}(\mathbf{x}), t)\mathcal{F}_{s'\to t}(\mathbf{x}), & \mathcal{F}_{s'\to s'}(\mathbf{x}) = \mathbf{I}, \end{cases} \tag{5}$$

$$\begin{cases} \frac{D\Psi_{t\to s'}(\mathbf{x})}{Dt} = 0, & \Psi_{s'\to s'}(\mathbf{x}) = \mathbf{x}, \\ \frac{D\mathcal{T}_{t\to s'}(\mathbf{x})}{Dt} = -\mathcal{T}_{t\to s'}(\mathbf{x})\nabla \mathbf{u}(\mathbf{x}, t), & \mathcal{T}_{s'\to s'}(\mathbf{x}) = \mathbf{I}. \end{cases} \tag{6}$$

In the flow map method, accurate calculation of advection depends on the accuracy of flow maps. While the forward flow map $\Phi_{s'\to t}$ and its Jacobian $\mathcal{F}_{s'\to t}$ can be integrated accurately on grids using high-order schemes, like the Fourth-order Runge-Kutta method (RK4) for computing Equation 5, the semi-Lagrangian treatment of advection terms $\frac{D}{Dt}$ in Equation 6 introduces dissipation and accumulates errors, making precise computation of Ψ and $\mathcal{T}$ challenging. To address this issue, [Deng et al. 2023b] observes that at any given time r', the backward flow map $\Psi_{r'\to s'}$ and its Jacobian $\mathcal{T}_{r'\to s'}$ can be interpreted as the result of evolving $\Psi_{r'\to t}$ and $\mathcal{T}_{r'\to t}$ backward in time from r' to s' with $\Delta t < 0$, which follows the dynamics of the reverse-time fluid motion without advection terms with start time r' (see Fig. 6 for illustration):

$$\begin{cases} \frac{\partial \Psi_{r'\to t}(\mathbf{x})}{\partial t} = \mathbf{u}(\Psi_{r'\to t}(\mathbf{x}), t), & \Psi_{r'\to r'}(\mathbf{x}) = \mathbf{x}, \\ \frac{\partial \mathcal{T}_{r'\to t}(\mathbf{x})}{\partial t} = \nabla \mathbf{u}(\Psi_{r'\to t}(\mathbf{x}), t)\,\mathcal{T}_{r'\to t}(\mathbf{x}), & \mathcal{T}_{r'\to r'}(\mathbf{x}) = \mathbf{I}. \end{cases} \tag{7}$$

Since Equation 7 excludes advection terms, high-order integration can accurately compute the backward flow map Ψ and its Jacobian $\mathcal{T}$. This approach is known as the Eulerian Flow Map method (EFM), and based on the accurately computed flow maps from Equation 5 and Equation 7, it achieves state-of-the-art performance in preserving vortex structures.

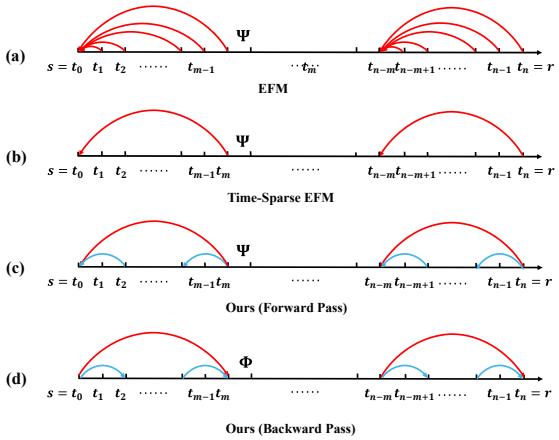


Fig. 6. **Illustration of long-range flow map evolution in different methods.** Let m be the reinitialization interval, typically $m = 15 \sim 60$. (a) EFM computes $\Psi_{t\to s'}$ at every time step by repeatedly evolving back to the previous reinit time. The number of steps crossed by each curve in the figure indicates the number of steps required for each flow map evolution, resulting in a total cost of $O(m^2)$. (b) Time-Sparse EFM computes $\Psi_{t\to s'}$ only at reinit steps, reducing the cost to $O(m)$. (c)(d) Our improved Time-Sparse EFM introduces shorter intermediate flow maps between reinit steps to improve accuracy at intermediate times, while maintaining the overall cost at $O(m)$—specifically, doubling the number of flow map steps compared to (b).

4 Differentiable Flow Maps

To implement differentiable flow maps, we compute $\hat{\mathbf{B}}$ by directly discretizing the backward process $\mathbf{B}$, rather than differentiating $\hat{\mathbf{F}}$ as in previous methods. Using flow maps, we first solve the Navier-Stokes Equation 1 forward from the start time s to the end time r, then solve the adjoint Navier-Stokes Equation 4 backward from r to s. These two processes are referred to as the forward and backward pass, which will be discussed below.

4.1 Forward Pass

By [Li et al. 2024b], Equation 1 can be accurately computed via flow maps, using the integral form of Equation 1

$$\begin{aligned}
&\mathbf{u}(\mathbf{x},t) = \mathcal{T}^{\top}_{t\to s}(\mathbf{x})\mathbf{u}(\Psi_{t\to s}(\mathbf{x}),s) + \mathcal{T}^{\top}_{t\to s}(\mathbf{x})\mathbf{\Gamma}_{s\to t}(\Psi_{t\to s}(\mathbf{x})),\\
&\mathbf{\Gamma}_{s\to t}(\mathbf{x}) = \int_s^t \mathcal{F}^{\top}_{s\to\tau}(\mathbf{x})\left(-\frac{1}{\rho}\nabla p + \frac{1}{2}\nabla\|\mathbf{u}\|^2 + \mathbf{f}\right)(\Phi_{s\to\tau}(\mathbf{x}),\tau)d\tau,\\
&\xi(\mathbf{x},t) = \xi(\Psi_{t\to s}(\mathbf{x}),s).
\end{aligned} \tag{8}$$

The detailed procedure is omitted here and provided in Appendix A of the supplementary material. We follow the same approach to accurately compute the evolution of the adjoint.

4.2 Backward Pass

Since the adjoint velocity field $\mathbf{u}^*_t$ also satisfies the incompressibility condition $\nabla\cdot\mathbf{u}^*_t = 0$, *Equation* 4 can similarly be solved using flow maps. Notably, both the forward Equation 1 and the adjoint Equation 4 are driven by the same velocity field, and the flow of the backward pass can be viewed as the time reversal of the forward pass.

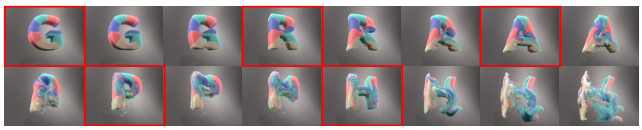


Fig. 7. **3D sequential optimization.** We perform 3D fluid control with multiple keyframes to guide a 3D shape morphing sequence from "G" to "R" to "A" to "P" to "H". The red boxes highlight keyframes where the fluid configuration successfully matches the target shapes.

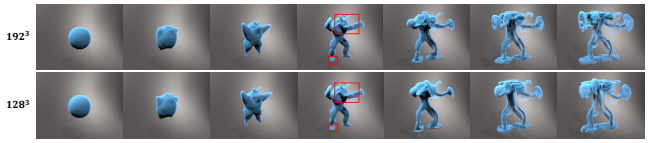


Fig. 8. **Armadillo shape morphing at different resolutions.** We compare shape morphing results at two resolutions: 192^3 (top) and 128^3 (bottom). The initial sphere is progressively optimized to match the Armadillo shape. Higher resolution preserves finer geometric details, particularly in regions highlighted by red boxes.

As a result, the forward flow maps $\Phi_{t_1\to t_2}$, $\mathcal{F}_{t_1\to t_2}$ and backward flow maps $\Psi_{t_2\to t_1}$, $\mathcal{T}_{t_2\to t_1}$, $t_1 < t_2$ used in the forward equations can serve as the backward and forward flow maps, respectively, in the backward pass of adjoint equations, allowing adjoint fields $\mathbf{u}^*_t$ and $\boldsymbol{\xi}^*_t$ to be expressed as:

$$\begin{aligned}
&\mathbf{u}^*(\mathbf{x},t) = \mathcal{F}^{\top}_{t\to r}(\mathbf{x})\mathbf{u}^*(\Phi_{t\to r}(\mathbf{x}),r) + \mathcal{F}^{\top}_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t}(\Phi_{t\to r}(\mathbf{x})),\\
&\mathbf{\Lambda}^u_{r\to t}(\mathbf{x}) = \int_r^t \mathcal{T}^{\top}_{r\to\tau}(\mathbf{x})\Big(2\nabla\mathbf{u}^{\top}\mathbf{u}^* + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u}^*\\
&\qquad\qquad - \frac{\partial J}{\partial\mathbf{u}}\Big)(\Psi_{r\to\tau}(\mathbf{x}),\tau)d\tau,\\
&\xi^*(\mathbf{x},t) = \xi^*(\Phi_{t\to r}(\mathbf{x}),r) + \Lambda^{\xi}_{r\to t}(\Phi_{t\to r}(\mathbf{x})),\\
&\Lambda^{\xi}_{r\to t}(\mathbf{x}) = -\int_r^t \frac{\partial J}{\partial\xi}(\Psi_{r\to\tau}(\mathbf{x}),\tau)d\tau.
\end{aligned} \tag{9}$$

Here, $\mathbf{u}^{*M}_{r\to t}(\mathbf{x}) = \mathcal{F}^{\top}_{t\to r}(\mathbf{x})\mathbf{u}^*(\Phi_{t\to r}(\mathbf{x}),r)$ is referred to as the long-range mapped adjoint velocity, and $\mathbf{\Lambda}^u_{r\to t}$ denotes the path integrator of the adjoint velocity. The long-range mapping allows $\mathbf{u}^*$ to avoid the error accumulation caused by advection. For $\xi^*(\mathbf{x},t)$, $\xi^{*M}_{r\to t}(\mathbf{x}) = \xi^*(\Phi_{t\to r}(\mathbf{x}),r)$ is the long-range mapped adjoint passive field and $\Lambda^{\xi}_{r\to t}$ is its path integrator.

For the adjoint calculation in the backward pass, we leverage the long-short term mapping conversion strategy introduced in [Chen et al. 2024b; Li et al. 2024b] to formulate our strategy to calculate the adjoint integration Equation 9. We discuss the details as follows.

Mapping and Conversion. We first compute the long-range mapped adjoint velocity $\mathbf{u}^{*M}_{r\to t}(\mathbf{x})$ using $\mathbf{u}^{*M}_{r\to t}(\mathbf{x}) = \mathcal{F}^{\top}_{t\to r}(\mathbf{x})\,\mathbf{u}^*(\Phi_{t\to r}(\mathbf{x}),r)$, and then convert the long-range mapped adjoint velocity to short-range advected adjoint velocity $\mathbf{u}^{*A}_{t'\to t}(\mathbf{x})$, where t' is the last time step of time t (see Supplementary Section B.1 for proof):

$$\mathbf{u}^{*A}_{t'\to t}(\mathbf{x}) = \mathbf{u}^{*M}_{r\to t}(\mathbf{x}) + \mathcal{F}^{\top}_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t'}(\Phi_{t\to r}(\mathbf{x})) + 2\nabla\mathbf{u}^{\top}\mathbf{u}^*\Delta t, \tag{10}$$

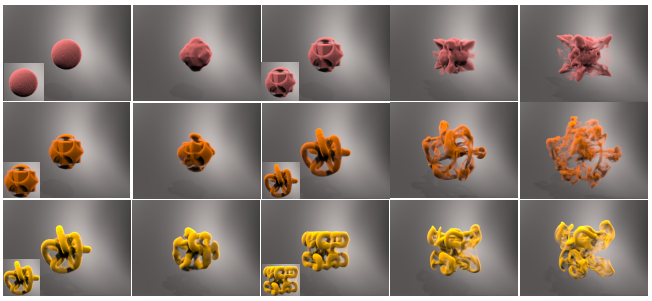


Fig. 9. **3D shape morphing with complex topologies.** A sequence of 3D shape morphing tasks demonstrating smooth transitions between complex topological structures at frame 0, 50, 100, 150, and 200. Each row illustrates a progressive transformation from a simple to a highly intricate shape, with insets showing the corresponding target geometries. The morphing process preserves topological features while gradually introducing geometric complexity and fine details.

where the advected adjoint velocity $\mathbf{u}^{*A}_{t'\to t}(\mathbf{x})$ is the velocity advected directly by the adjoint advection equation $\left(\frac{\partial}{\partial t}+(\mathbf{u}\cdot\nabla)\right)\mathbf{u}^*=\nabla\mathbf{u}^\top\mathbf{u}^*$ from the previous backward timestep t'. After mapped adjoint passive field $\xi^{*M}_{r\to t}(\mathbf{x})=\xi^*(\mathbf{\Phi}_{t\to r}(\mathbf{x}),r)$ is calculated, with path integrator $\mathbf{\Lambda}^{\xi}_{r\to t'}$ and current source term $\frac{\partial J}{\partial \xi}$, the current adjoint passive field is updated as:

$$\xi^*(\mathbf{x},t)=\xi^{*M}_{r\to t}(\mathbf{x})+\mathbf{\Lambda}^{\xi}_{r\to t'}(\Phi_{t\to r}(\mathbf{x}))-\frac{\partial J}{\partial \xi}(\mathbf{x})\Delta t. \tag{11}$$

Accumulated Effect. Then we proceed to compute the accumulated contributions from terms other than the mapping. After the $\xi^*\nabla\xi$ is calculated, together with the viscous term $\nu\Delta\mathbf{u}^*$ calculated from $\mathbf{u}^{*A}_{t'\to t}(\mathbf{x})$ and the source term of the objective functional $\frac{\partial J}{\partial \mathbf{u}}$, the unprojected velocity $\mathbf{u}^{*\mathrm{up}}_t(\mathbf{x})$ is calculated as

$$\mathbf{u}^{*\mathrm{up}}_t(\mathbf{x})=\mathbf{u}^{*A}_{t'\to t}(\mathbf{x})+(\xi^*\nabla\xi-\frac{1}{\rho}\nabla p^*+\nu\Delta\mathbf{u}^*-\frac{\partial J}{\partial \mathbf{u}})\Delta t. \tag{12}$$

Projection. To obtain the final adjoint velocity at the current timestep, an adjoint Poisson equation is solved with the adjoint non-through boundary condition [Stück 2012]:

$$\begin{aligned}\frac{\Delta t}{\rho}\Delta p^*&=\nabla\cdot\mathbf{u}^{*\mathrm{up}}_t,\\ \mathbf{u}^*_t\cdot\mathbf{n}&=0,\mathbf{x}\in\partial_b\mathbb{U}_t,\end{aligned} \tag{13}$$

where $\mathbf{n}$ is the normal vector of the solid boundary, and $\partial_b\mathbb{U}_t$ denotes the solid boundary of the domain. Then calculate the final adjoint velocity at current time by projection:

$$\mathbf{u}^*_t=\mathbf{u}^{*\mathrm{up}}_t-\frac{\Delta t}{\rho}\nabla p^*. \tag{14}$$

Path Integrator Update. Subsequently, it is necessary to accumulate the adjoint source term, adjoint viscous term and adjoint pressure gradient $-\nabla p^*$ into the path integrator $\mathbf{\Lambda}^u_{r\to t}$ for long-short term mapping conversion next step and accumulate $\frac{\partial J}{\partial \xi}$ into $\mathbf{\Lambda}^{\xi}_{r\to t}$. $\mathbf{\Lambda}^u_{r\to t}$ and $\mathbf{\Lambda}^{\xi}_{r\to t}$ are updated by their definition in Equation 9 with

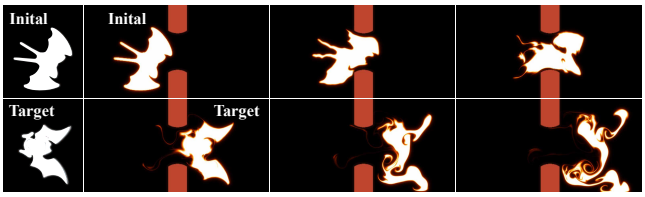


Fig. 10. **Fluid shape morphing with obstacle constraints.** We demonstrate a challenging smoke control task involving obstacle-aware shape matching. By optimizing the control forces, the bat-shaped fluid navigates through the gap and transforms into the target configuration.

$\Delta t<0$ respetively as:

$$\begin{aligned}\mathbf{\Lambda}^u_{r\to t}(\mathbf{x})=\mathbf{\Lambda}^u_{r\to t'}(\mathbf{x})+\Delta t\mathcal{T}^{\top}_{r\to t}(\mathbf{x})\Big(&2\nabla\mathbf{u}^\top\mathbf{u}^*+\xi^*\nabla\xi\\&-\frac{1}{\rho}\nabla p+\nu\Delta\mathbf{u}^*-\frac{\partial J}{\partial \mathbf{u}}\Big)(\mathbf{\Psi}_{r\to t}(\mathbf{x}),t),\\ \mathbf{\Lambda}^{\xi}_{r\to t}(\mathbf{x})=\mathbf{\Lambda}^{\xi}_{r\to t'}(\mathbf{x})-\Delta t\frac{\partial J}{\partial \xi}&(\mathbf{\Psi}_{r\to t}(\mathbf{x}),t).\end{aligned} \tag{15}$$

4.3 Long-Short Time-Sparse EFM

Directly differentiating the flow map forward process poses several challenges. The original flow map methods, including the Eulerian Flow Map (EFM) [Deng et al. 2023b] and Particle Flow Map (PFM) [Zhou et al. 2024], are intensive in terms of both computational cost and memory consumption. Differentiating the forward process requires additional storage for intermediate states, and the 3–5 times backward pass computation further exacerbates the time and memory overhead. However, differentiable simulation requires repeated runs during optimization, making time efficiency critical. To avoid the expensive $O(m^2)$ flow map evolution in EFM, we adopt time-sparse EFM [Sun et al. 2025] for both the forward Equation 8 and backward Equation 9 computations. In this approach, long-range mapping is applied only at reinit steps, while intermediate steps use semi-Lagrangian for advection and meanwhile accumulate the path integral for later use in flow map calculation at reinit steps.

The key idea of time-sparse EFM is to tolerate error accumulation within each reinit interval due to semi-Lagrangian advection, and then correct it at reinit steps using more accurate long-range mapped velocities from the last reinit. As shown in Fig. 15 (right), while time-sparse EFM preserves vortices over long timescales, its energy curve shows a sawtooth pattern with clear decay between reinit steps. This is acceptable in the visual effects of computer graphics, where only selected frames are rendered. By aligning the reinit interval with the frame output interval, the visual result remains unaffected. However, in adjoint calculation in backward pass, this poses a major problem. Unlike visual effects, velocity $\mathbf{u}$ at every step contributes explicitly to the adjoint $\mathbf{u}^*$ through the $\nabla\mathbf{u}^\top\mathbf{u}^*$ term in Equation 4. Errors from intermediate steps accumulate via the path integral $\mathbf{\Lambda}_{r\to t}$, significantly degrading the accuracy of $\mathbf{u}^*$—as evidenced by the noticeable artifacts in the adjoint fields of the leapfrogging example shown in Fig. 15.

To address this issue, we propose **Long-Short Time-Sparse EFM**, an improved version of the original Time-Sparse EFM. We

Algorithm 1 Long-Short Time-Sparse EFM of Backward Pass

Initialize: short-range reinit time r'' to long-range reinit time r'.

for each time step t_c between r' and $r' + \Delta t^l_{\text{reinit}}$, **do**
  March $\Psi^l_{r'\to t_c}$,$\Psi^s_{r''\to t_c}$,$\mathcal{T}^l_{r'\to t_c}$,$\mathcal{T}^s_{r''\to t_c}$ one step; ▷ eq. 7
  if $\exists m \in \mathbb{Z}$, st. $t_c = r' + m\Delta t^s_{\text{reinit}}$ **then**
    Calculate $\Phi^s_{t_c\to r''}$ and $\mathcal{F}^s_{t_c\to r''}$ by integrating $\Phi^s_{t_c\to t}$ and $\mathcal{F}^s_{t_c\to t}$ from t_c to r''; ▷ eq. 5
    Do mapping and conversion with $\Phi^s_{t_c\to r''}$ and $\mathcal{F}^s_{t_c\to r''}$; ▷ eq. 10
    Reset $\Psi^s_{r''\to t_c}$ and $\mathcal{T}^s_{r''\to t_c}$ and set $r'' = t_c$;
  else
    Calculate advection with semi-Lagrangian method;
  Calculate source term and projection;
  Update $\Lambda^{u,l}_{r'\to t_c}$, $\Lambda^{\xi,l}_{r'\to t_c}$ and $\Lambda^{u,s}_{r''\to t_c}$, $\Lambda^{\xi,s}_{r''\to t_c}$; ▷ eq. 15
  if $t_c = r' + \Delta t^l_{\text{reinit}}$ **then**
    Calculate $\Phi^l_{t_c\to r'}$ and $\mathcal{F}^l_{t_c\to r'}$ by integrating $\Phi^l_{t_c\to t}$ and $\mathcal{F}^l_{t_c\to t}$ from t_c to r'; ▷ eq. 5
    Correct results by adding long-range mapping with $\Phi^l_{t_c\to r'}$, $\mathcal{F}^l_{t_c\to r'}$ and path integrators. ▷ eq. 9

observe that while Time-Sparse EFM maintains accuracy over long temporal distances using long-range flow maps, it suffers from significant errors over short distances. To remedy this, we introduce sparse, short-range flow maps between reinit steps to improve local accuracy. We call this scheme **Long-Short Time-Sparse EFM**. This scheme is used in both forward simulation and backward adjoint calculation. Here, we take the backward pass as an example. The method for the forward pass is similar and presented in Algorithm 3 of Supplementary Section A.

We reinitialize the long-range flow map every $\Delta t^l_{\text{reinit}} = n^l \Delta t$ and the short-range flow map every $\Delta t^s_{\text{reinit}} = n^s \Delta t$, typically with $n^l = 15 \sim 60$ and $n^s = 1 \sim 3$. In Algorithm 1, we illustrate one long-range reinit cycle starting from the previous long-range reinit time r'. Let r'' denote the most recent short-range reinit time, initialized as $r'' = r'$. We maintain two sets of flow maps, $\Psi^l_{r'\to t}$, $\mathcal{T}^l_{r'\to t}$, $\Phi^l_{t_c\to t}$, $\mathcal{F}^l_{t_c\to t}$ for long-range, $\Psi^s_{r''\to t}$, $\mathcal{T}^s_{r''\to t}$, $\Phi^s_{t_c\to t}$, $\mathcal{F}^s_{t_c\to t}$ for short-range and two path integrators $\Lambda^{u,l}_{r'\to t}$, $\Lambda^{\xi,l}_{r'\to t}$ and $\Lambda^{u,s}_{r''\to t}$, $\Lambda^{\xi,s}_{r''\to t}$ for long-range and short-range respectively, where $t_c < r'$ is the current time and t serves as the evolving time variable during integration.

In Algorithm 1, the blue-highlighted parts indicate the components added by Long-Short Time-Sparse EFM compared to Time-Sparse EFM, corresponding to the blue lines in Fig. 6 (c)(d). Our method retains the $O(m)$ time complexity of flow map evolution, with a one-time overhead in the number of evolution steps. As in the original Time-Sparse EFM, we use long-range flow maps to maintain accuracy over large time intervals, while short-range flow maps ensure accuracy between sparse long-range updates, preventing error accumulation of $\mathbf{u}^*$ caused by inaccurate intermediate velocities.

5 Numerical Algorithm

We use a grid $\mathbb{G}$ with spacing Δx, where $\mathbf{x}_g$ denotes the position of grid point $g \in \mathbb{G}$. We denote the field value at grid point g by

Algorithm 2 Long-Short Time-Sparse EFM for Adjoint Field

Initialize: $\mathbf{u}^*_{r'',g}$, $\mathbf{u}^*_{r',g}$ to initial adjoint velocity; $\mathcal{T}^s_{r''\to t_c,g}$, $\mathcal{F}^s_{t_c\to r'',g}$, $\mathcal{T}^l_{r'\to t_c,g}$, $\mathcal{F}^l_{t_c\to r',g}$ to $\mathbf{I}$; $\Psi^s_{r''\to t_c,g}$, $\Phi^s_{t_c\to r'',g}$, $\Psi^l_{r'\to t_c,g}$, $\Phi^l_{t_c\to r',g}$ to $\mathbf{x}_g$; $\Lambda^{u,l}_{r'\to t_c,g}$, $\Lambda^{\xi,l}_{r'\to t_c,g}$ and $\Lambda^{u,s}_{r''\to t_c,g}$, $\Lambda^{\xi,s}_{r''\to t_c,g}$ to 0; r', r'', t_c to r.

for each time step t_c from t_n to t_0 **do**
  Load midpoint velocity $\mathbf{u}^{\text{mid}}_g$;
  March $\Psi^l_{r'\to t_c,g}$,$\Psi^s_{r''\to t_c,g}$,$\mathcal{T}^l_{r'\to t_c,g}$,$\mathcal{T}^s_{r''\to t_c,g}$ one step; ▷ eq. 7
  if $c \pmod{n^s} = 0$ **then**
    Integrate $\mathcal{F}^s_{t_c\to r'',g}$ and $\Phi^s_{t_c\to r'',g}$ from t_c to r''; ▷ eq. 5
    Calculate mapped adjoint velocity $\mathbf{u}^{*M}_{r''\to t_c,g}$ and convert to one-step advected adjoint velocity $\mathbf{u}^{*A}_{t_{c+1}\to t_c,g}$; ▷ eq. 10
    Set initial time for short mapping r'' to t_c;
    Reinitialize $\mathcal{T}^s_{r''\to t_c,g}$ to $\mathbf{I}$, $\Psi^s_{r''\to t_c,g}$ to $\mathbf{x}_g$;
    Calculate $\xi^*_{t_c,g}$ by $\Phi^s_{t_c\to r'',g}$ and $\Lambda^{\xi,s}_{r''\to t_{c+1},g}$; ▷ eq. 11
  else
    Calculate $\mathbf{u}^{*A}_{t_{c+1}\to t_c,g}$ and $\xi^{*A}_{t_{c+1}\to t_c,g}$ by semi-Lagrangian;
    Calculate $\xi^*_{t_c,g} = \xi^{*A}_{t_{c+1}\to t_c,g} - \Delta t \frac{\partial J}{\partial \xi}$;
  Calculate adjoint viscous term $[\nu\Delta\mathbf{u}^*]_g$; ▷ eq. 18
  Calculate coupling term $[\xi^*\nabla\xi]_g$ and $[\nabla\mathbf{u}^\top\mathbf{u}^*]_g$; ▷ eq. 19
  Compute source term from objective functional $\frac{\partial J}{\partial \mathbf{u}}$;
  Compute unprojected $\mathbf{u}^{*\text{up}}_{t_c,g}$; ▷ eq. 12
  Calculate $\mathbf{u}^*_{t_c,g}$ using p^* from Possion equation; ▷ eq. 13,14
  Update both short and long path integrator $\Lambda^{u,l}_{r'\to t_c,g}$, $\Lambda^{\xi,l}_{r'\to t_c,g}$ and $\Lambda^{u,s}_{r''\to t_c,g}$, $\Lambda^{\xi,s}_{r''\to t_c,g}$; ▷ eq. 15
  if $c \pmod{n^l} = 0$ **then**
    Integrate $\mathcal{F}^l_{t_c\to r',g}$ and $\Phi^l_{t_c\to r',g}$ from t_c to r'; ▷ eq. 5
    Calculate accurate $\xi^*_{t_c,g}$ and $\mathbf{u}^*_{t_c,g}$ by mapping with $\mathcal{F}^l_{t_c\to r',g}$, $\Phi^l_{t_c\to r',g}$ and integrator $\Lambda^{u,l}_{r'\to t_c,g}$, $\Lambda^{\xi,l}_{r'\to t_c,g}$; ▷ eq. 9
    Set initial time for long mapping r' to t_c.

the subscript $_g$. With a fixed time step Δt, we define $t_i = i\Delta t$ for $i = 0, \ldots, n$, where $t_0 = s$ and $t_n = r$. Our complete algorithm consists of a forward pass for computing physical quantities and a backward pass for computing their adjoints. Algorithm 2 illustrates the backward pass, while the forward pass is presented in Algorithm 4 of Supplementary Section C, as forward pass calculation is not the main focus of this paper.

Interpolation. To compute the mapping, we interpolate the mapped field using a second-order kernel with a support radius of $1.5\Delta x$. For example, the computation of $\mathbf{u}^{*M}_{r'\to t_c}(\mathbf{x}) = \mathcal{F}_{t_c\to r'}(\mathbf{x})\mathbf{u}^*(\Phi_{t_c\to r'}(\mathbf{x}), r')$ is given by:

$$\mathbf{u}^{*M}_{r'\to t_c,g} = \mathcal{F}_{t_c\to r',g} \sum_{g'\in N(\Phi_{t_c\to r',g})} \mathbf{u}^*_{r',g'} w(\mathbf{x}_{g'} - \Phi_{t_c\to r',g}), \tag{16}$$

where $N(\Phi_{t_c\to r',g}) = \{g' \mid w(\mathbf{x}_{g'} - \Phi_{t_c\to r',g}) > 0\}$ denotes the set of grid points neighboring $\Phi_{t_c\to r',g}$.

Midpoint Velocity. Following [Deng et al. 2023b], the flow maps and their Jacobians, $\mathcal{F}$ and $\mathcal{T}$, are advected using the fourth-order

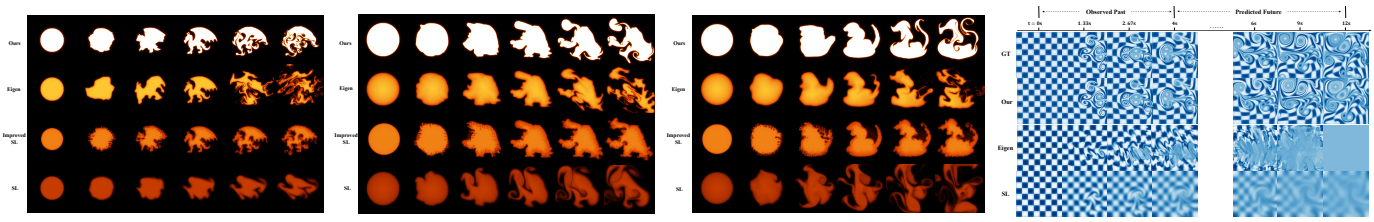


Fig. 11. **Comparison with different differentiable solvers.** We compare our method with EigenFluids and semi-Lagrangian (SL) methods on two tasks: 2D smoke control and vortex inference. *Columns 1–3:* 2D smoke control tasks (Dragon, Turtle, Snake). SL fails to achieve precise fluid control, while EigenFluids and SL with optimized control forces (Improved SL) can guide the fluid but suffer from poor advection, leading to unrealistic motion. In contrast, our approach achieves accurate control while preserving fine details and overall volume, resulting in more realistic fluid behavior. *Column 4:* Vortex inference task, where only our method succeeds, highlighting its superior accuracy.

Runge-Kutta method to solve Equation 5 and Equation 7, respectively. The midpoint velocity $\mathbf{u}_r^{\text{mid}}$ is computed according to Algorithm 2 in [Deng et al. 2023b] by marching the velocity field forward by half a time step. Since these midpoint velocities are also required during the backward pass for evolving the flow maps and solving Equation 9, we store them at each step to avoid redundant computation. Unlike those autodiff methods, which require retaining all intermediate velocities in the computation graph, our method allows storing them on disk or in memory without using GPU.

BFECC for Mapping. Since the flow maps we evolve are bidirectional, following [Deng et al. 2023b; Sun et al. 2025], we apply back and forth error compensation and correction (BFECC) during the mapping process using the flow maps. Taking the mapping of adjoint velocity $\mathbf{u}^*_{r'\to t_c}(\mathbf{x}) = \mathcal{F}_{t_c\to r'}(\mathbf{x})\mathbf{u}^*(\mathbf{\Phi}_{t_c\to r'}(\mathbf{x}), r')$ as an example, the BFECC procedure is as follows:

$$\begin{aligned}
\mathbf{u}^{*(1)}_{r'\to t_c}(\mathbf{x}) &= \mathcal{F}_{t_c\to r'}(\mathbf{x})\mathbf{u}^*(\mathbf{\Phi}_{t_c\to r'}(\mathbf{x}), r'),\\
\mathbf{u}^{*(2)}_{t_c\to r'}(\mathbf{x}) &= \mathcal{T}_{r'\to t_c}(\mathbf{x})\mathbf{u}^{*(1)}_{r'\to t_c}(\mathbf{\Psi}_{r'\to t_c}(\mathbf{x})),\\
\mathbf{u}^*_{r'\to t_c}(\mathbf{x}) &= \mathbf{u}^{*(1)}_{r'\to t_c}(\mathbf{x}) + \frac{1}{2}\mathcal{F}_{t_c\to r'}(\mathbf{x})(\mathbf{u}^*_{r'} - \mathbf{u}^{*(2)}_{t_c\to r'})(\mathbf{\Phi}_{t_c\to r'}(\mathbf{x})),
\end{aligned} \tag{17}$$

where the superscripts $^{(1)}$ and $^{(2)}$ denote intermediate steps in the BFECC process. Other terms, such as $\mathcal{F}_{t_c\to r'}(\mathbf{x})\Lambda^u_{r'\to t_c}(\mathbf{\Phi}_{t_c\to r'}(\mathbf{x}))$, which rely on flow maps for calculation, are similarly computed using this strategy.

Pressure Projection. For 2D simulations, we solve the Poisson equation using a multigrid preconditioned conjugate gradient solver (MGPCG) as in [Deng et al. 2023b; Zhou et al. 2024]. For 3D simulations, to improve computational efficiency, we adopt a fast matrix-free algebraic multigrid preconditioned conjugate gradient (AMGPCG) solver as used in [Sun et al. 2025].

Derivatives Calculation. On the grid, we compute the viscous term using second-order finite difference schemes, as

$$[\nu\Delta\mathbf{u}]_g = \nu \sum_{g'\in N_g} (\mathbf{u}'_{\mathbf{g}} - \mathbf{u}_{\mathbf{g}})/(|N_g|\Delta x^2), \tag{18}$$

where N_g represents the adjacent grid points of g with $|N_g| = 4$ for 2D and $|N_g| = 6$ for 3D. For the coupling terms $\nabla\mathbf{u}^\top\mathbf{u}^*$ and $\xi^*\nabla\xi$ in Equation 10 and Equation 12, we use the third-order kernel-based interpolation to calculate as

$$[\nabla\zeta]_g = \sum_{g'\in N_g^{w_3}} \zeta_g \nabla_{x_g} w_3(\mathbf{x}_g - \mathbf{x}_{g'}), \tag{19}$$

where ζ is a field (either $\mathbf{u}$ or ξ), and w_3 is a third-order kernel with compact support of radius $2\Delta x$. The neighbor set $N_g^{w_3} = \{g' \mid w_3(\mathbf{x}_{g'} - \mathbf{x}_g) > 0\}$ includes grid points within the kernel support of g. When computing $\nabla\mathbf{u}^\top\mathbf{u}^*$, for consistency with $\mathbf{u}^*$, $\mathbf{u}$ used in $\nabla\mathbf{u}^\top\mathbf{u}^*$ is calculated backward along with $\mathbf{u}^*$.

6 Results and Discussion

In this section, we first validate the accuracy of our method on specific flow fields and objective functionals whose adjoint velocities admit analytical solutions. We then demonstrate our approach on three representative tasks: vortex dynamics inference from videos, vortex control, and smoke control. For these tasks, we compare our method with prior differentiable solvers, highlighting both quantitative and qualitative improvements. For each task, we will present the objective functional integrand J (see Equation 3) along with experiment-specific optimization parameters.

Validation & Ablation Test. First, we validate on cases with known adjoints for correctness of adjoint calculation: for incompressible flow with rigid walls, choosing $J = \frac{1}{2}\delta(t-r)\|\mathbf{u}\|^2$ forces $\mathbf{u}^*_t = \mathbf{u}_t$ $(r \geq t \geq s)$, giving a direct velocity check. For all experiments, we set the simulation to run for 100-500 steps and performed 200–800 optimization iterations. Fig. 15 confirms the correctness of our results for single one vortex and leapfrog vortices, and also shows that using Time-Sparse EFM alone incurs large errors, highlighting the need for our Long-Short Time-Sparse EFM. **3D leapfrog.** Fig. 16 confirms our adjoint computation for 3-D leap-frogging vortex rings— a case standard autodiff fails. It also shows that the semi-Lagrangian method for forward or backward pass yields incorrect results. Next, we verify optimization capability through a known analytic flow. **Viscosity Coefficient Inferring.** On the domain $[0, 2\pi]^2$, the Navier–Stokes equations admit the Taylor-Green vortex $\mathbf{u} = (\cos x \sin y, -\sin x \cos y)e^{-2\nu t}$. Using its analytic velocity at $800\Delta t$ as the target $\mathbf{u}_{\text{target}}$ and treating the viscosity ν as the optimization parameter, we infer the viscosity by minimizing $J = \frac{1}{2}\,\delta(t-r)\big(\|\mathbf{u}_{\text{target}}\|^2 - \|\mathbf{u}\|^2\big)$ as shown in Fig. 14.

Examples. **(1) Vortex Dynamics Inference from Videos.** This task infers an initial velocity field, represented by 16 vortices, from a single RGB video of fluid motion (synthetic or real) [Deng et al. 2023a]. The optimization variables are $(c_{x,i}, c_{y,i}, w_i, r_i), i = 1, ..., 16$ which denote the position, strength, and radius of each vortex respectively. The objective minimizes the difference between simulated frames ξ_i and target video frames $\xi_{i,\text{gt}}$: $J = \frac{1}{2}\sum_k \sum_{i=1}^{3} \delta(t - t_k)|\xi_i(\mathbf{x}, t) - \xi_{i,\text{gt}}(\mathbf{x}, t)|^2$, where $\xi_i, i = 1, 2, 3$ represents the passive fields for the R, G, and B channels, and t_k denotes the time corresponding to the k-th frame. Optimization is initialized with 16 randomly placed vortices, and the RGB video is the only input of the optimization process. We show some beautiful results of infering initial vorticity field from warped **2D Logo** and **2D Gradient Background** (Fig. 2) examples, and more complex scenarios with obstacles such as **2D Three Cylinders** and **2D Leaf Obstacle** (Fig. 3). All demonstrate the robustness and generality of our method in complex environments, including multiple obstacles and real environment. This task requires strong vortex preservation, which can only be achieved using differentiable flow maps. **(2) 2D Vortex Control.** This task optimizes the initial positions $\mathbf{c}_{j,\text{init}}^{\text{ctrl}}$ of 1–2 controlled vortices to guide other vortices with position $\mathbf{c}_i(t)$ toward target locations $\mathbf{c}_i^{\text{target}}$ via vortex interactions, with the loss defined as $J = \sum_i \delta(t - r)\|\mathbf{c}_i(t) - \mathbf{c}_i^{\text{target}}\|_2^2$, where $\mathbf{c}_i$ denotes the vortex positions and $\mathbf{c}_i^{\text{target}}$ the target positions. As $\mathbf{c}_i$ is driven by $\mathbf{u}_t$, which depends on vortex interactions and initial positions, the functional J can be viewed as a functional of $\mathbf{u}_t$ as in Equation 3 and optimized over $\mathbf{c}_{j,\text{init}}^{\text{ctrl}}$. Since controlling vortex relies on preserving vortex structures, flow maps are particularly well-suited for this task. We present the scenarios of **Single Vortex** case (Fig. 12), **Multi-Vortex Single-Target** case and the **Multi-Vortex Multi-Target** case (Fig. 13). These examples, with increasing levels of difficulty, demonstrate our method's ability to preserve vortex structures and to compute accurate gradients for optimization in complex scenarios. **(3) 2D Smoke Control.** This task optimizes a time-varying control force $\mathbf{f}$ to influence the fluid and deform smoke into a target shape [Chen et al. 2024a; Treuille et al. 2003]. The force is modeled using m ($m \sim 3000$) Gaussian wind fields $\mathbf{f}_i = \mathbf{w}_i e^{-a\|\mathbf{x}-\mathbf{c}_i\|^2}$ with centers $\mathbf{c}_i$ and strength vector $\mathbf{w}_i$ as parameters, $i = 1, .., m$ [Treuille et al. 2003]. The objective function integrand is $J = \sum_{k\in K} \delta(t - t_k)|\xi(\mathbf{x}, t) - \xi^{\text{target}}(\mathbf{x}, t)|^2$, where t_k denotes the the time of $k-$th key frame. We showcase continuous keyframe optimization through the **2D GRAPH Formation** example (Fig. 5a), the **2D Life Evolution** example (Fig. 5b), and the **2D Bat Through Obstacles** example (Fig. 10), which involves optimization in the presence of obstacles. Accurate shape control and fluid realism are enabled by the flow map's precise passive field mapping and strong advection preservation. **(4) 3D Smoke Control.** We extend the above task to a 3D volume setting. We showcase the comparison of **3D Sphere to Armadillo** (Fig. 8) across different resolutions, the continuous 3D shape transitions in **3D GRAPH** (Fig. 7), and the complex topological changes in **3D Topological Morphing** (Fig. 9). These examples demonstrate that our method remains accurate in 3D, enabling the preservation of fine smoke strands and coherent flow structures.

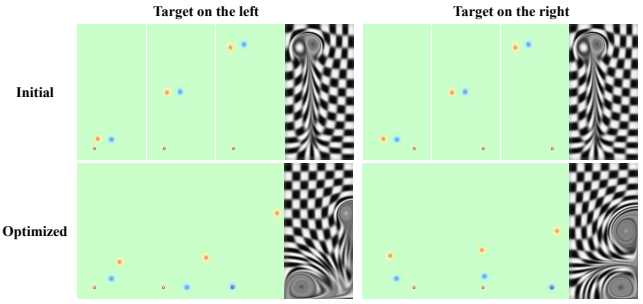


Fig. 12. **Single vortex control via position and vorticity optimization.** We optimize the position and vorticity of the red vortex to guide the blue vortex toward the target (indicated by the hollow circle). The top row shows results without optimization, while the bottom row shows optimized results that successfully steer the blue vortex to the target at frames 0, 400, and 800, with a checkerboard pattern at frame 800.

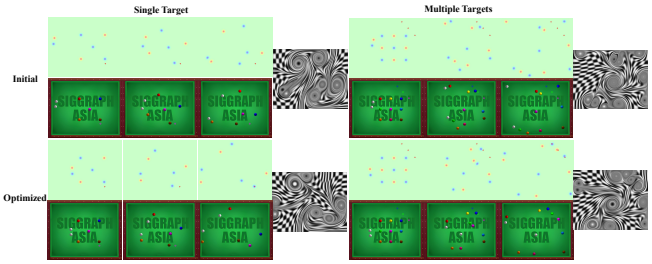


Fig. 13. **Multiple vortex control.** We optimize the positions and vorticities of the white spheres to accomplish complex control tasks involving single and multiple targets. Hollow circles denote the target positions. The top row shows results with randomly initialized white spheres, while the bottom row displays our optimized outcomes. Snapshots of vorticity and rendered results are provided at frames 0, 400, and 800, with the checkerboard pattern shown at frame 800.

Table 1. Task categories and comparison of differentiable methods: differentiable semi-Lagrangian (SL), differentiable Eigenfluids (Eigen), and ours. In the table, ✓ and × indicate whether a solver can accomplish the task; bold entries denote the experiments selected for comparison, and underlined entries mark the best-performing solver for each task.

Task	Experiments	SL	Eigen	Ours
Vortex Dynamics Inference	Fig. **11 (col. 4)**, 2, 3	×	×	✓
Vortex Control	Fig. 12, 13	×	×	✓
Smoke Control	Fig. **11 (col. 1-3)**, 10, 5a, 5b, 9, 7, 8	✓	✓	✓

Comparison. We demonstrate the effectiveness of our adjoint method in four experiments and compare it with baseline methods, including differentiable semi-Lagrangian method [Li et al. 2024c; Treuille et al. 2003] and differentiable Eigenfluids [Chen et al. 2024a]. For non-open-source reasons, [Chen et al. 2024a] is reproduced by adapting the open-source code [Börcsök and Szécsi 2023] implemented in Φ_{Flow}[Holl and Thuerey 2024]. [Treuille et al. 2003] is implementation by DiffTaichi [Hu et al. 2019a]. As shown in Fig. 11, our method demonstrates significantly higher accuracy in both the **Vortex Dynamics Inference from Images** and **2D Smoke**

Table 2. Final volume conservation errors (%) for the 2D smoke control tasks in Fig. 11. Lower is better.

Volume Percentage	Ours	DiffEigen	Improved-SL	SL
Dragon	0.0016%	2.89%	6.54%	0.44%
Snake	0.0056%	1.50%	4.91%	3.28%
Turtle	0.0018%	2.84%	3.46%	2.02%

Table 3. Runtime (s) and GPU memory cost (GB) comparison between forward and backward passes for different examples. Here, (F) and (B) indicate whether the data corresponds to the forward pass or the backward pass, respectively. The reported times are per step and exclude the computation of external inputs and outputs, such as control forces. Poisson and Advection report the average cost of a single step of solving the Poisson equation and advecting the flow map, respectively. In 2D, we implemented a custom Poisson solver in Taichi, while in 3D we employed the high-performance MGPCG Poisson solver from [Sun et al. 2025], which makes the 3D Poisson solve significantly faster than its 2D counterpart.

Figure	Resolution	memory (F)	memory (F&B)	time (F)	time (B)	Poisson	Advection
Fig. 11 (col. 4)	256 × 256	1.6 GB	1.6 GB	0.17 s	0.18 s	0.16 s	0.004 s
Fig. 2	256 × 256	1.7 GB	1.7 GB	0.19 s	0.20 s	0.19 s	0.002 s
Fig. 3	256 × 256	1.7 GB	1.7 GB	0.18 s	0.21 s	0.19 s	0.01 s
Fig. 11 (col. 1-3)	256 × 256	1.6 GB	1.6 GB	0.16 s	0.21 s	0.18 s	0.01 s
Fig. 3	256 × 256	1.6 GB	1.6 GB	0.17 s	0.21 s	0.18 s	0.01 s
Fig. 5a & Fig. 5b	256 × 256	1.6 GB	1.6 GB	0.19 s	0.22 s	0.20 s	0.01 s
Fig. 8 Top	196 × 196 × 196	5.69 GB	6.53 GB	0.24 s	0.28 s	0.03 s	0.26 s
Fig. 8 bottom	128 × 128 × 128	2.10 GB	2.35 GB	0.08 s	0.09 s	0.06 s	0.02 s
Fig. 9	128 × 128 × 128	2.10 GB	2.35 GB	0.08 s	0.08 s	0.06 s	0.02 s
Fig. 7	128 × 128 × 128	2.10 GB	2.35 GB	0.09 s	0.10 s	0.03 s	0.05 s

Control tasks. In the 2D smoke control comparison, we further compare with a state-of-the-art method based on the semi-Lagrangian scheme with optimized control forces [Tang et al. 2021], and our approach still achieves superior results. Since neither Eigen Fluid nor Semi-Lagrangian methods preserve vortices, they are unable to accomplish the **2D Vortex Control** task. Moreover, as shown in [Chen et al. 2024a], Eigen Fluid also encounters difficulties in the **3D Smoke Control** scenario. A full summary of task categories and solver comparisons is provided in Table 1.

Volume Conservation. Benefiting from the accuracy of the flow map formulation, our method preserves smoke volume effectively throughout the control process. As shown in Table 2, our method achieves fluctuations below 0.006% across all 2D examples in Fig. 11, whereas competing approaches exhibit deviations ranging from 0.44% to 6.54%. In the 3D case (Fig. 7), our method further achieves a final-frame fluctuation of only 0.018%.

Time and Memory Cost. We report the runtime and GPU memory usage of our 3D examples to support our choice of numerically solving the adjoint Navier-Stokes equation instead of differentiating the forward process. As shown in Table 3, the backward pass has similar runtime and memory cost to the forward pass across different tasks, avoiding the significant increase in backpropagation that is often attributed to differentiating complex computation graph.

7 Conclusion and Future Work

This paper presents a differentiable flow map method to improve the accuracy and applicability of differentiable fluid simulation. Some limitations still remain. We focus on control forces and velocity fields, without tackling shape optimization and solid boundaries. The differentiation of incompressible flow with a free surface remains to be explored. Future work may also explore shape-based design tasks (e.g., [Li et al. 2024c]), perform shape optimization with real smoke images as targets, and experiment with more advanced optimization algorithms beyond quasi-Newton methods.

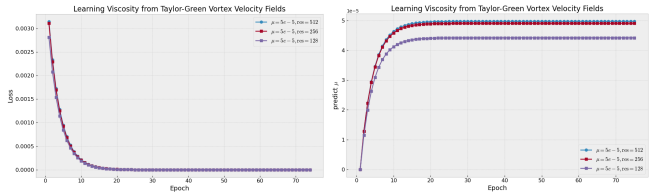


Fig. 14. **Taylor-Green velocity field viscous inference.** We use the analytical solution as the target to test if our method can infer the viscosity from the velocity field. Results at resolutions 128, 256, and 512 are shown.

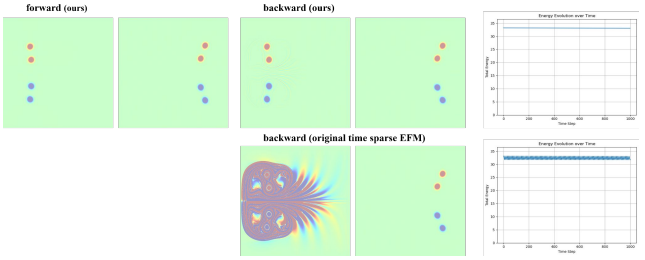


Fig. 15. **Ablation study.** Ablation studies on the leapfrog (left) and single-vortex (right) tests reveal that the original time-sparse EFM fails to compute accurate adjoints: the evolution of leapfrog vortices shows significant errors, while the single-vortex particles exhibit zigzag artifacts in the energy curves, thereby highlighting the necessity of our long-short time-sparse EFM.

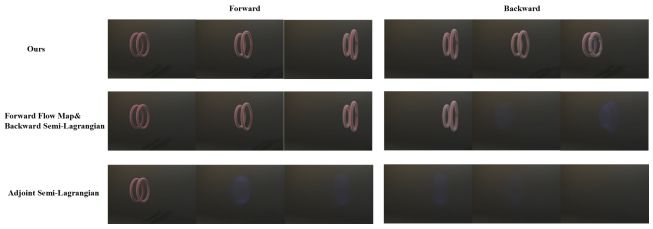


Fig. 16. **3D leapfrog comparison of forward and backward processes using different methods.** We test replacing both the forward and backward computations with semi-Lagrangian methods, and the results confirm that flow map methods are essential for both directions. This experiment also demonstrates the accuracy of our method in computing the adjoint.

Acknowledgments

We express our gratitude to the anonymous reviewers for their insightful feedback. Georgia Tech authors acknowledge NSF IIS #2433322, ECCS #2318814, CAREER #2433307, IIS #2106733, OISE #2433313, and CNS #1919647 for funding support. We credit the Houdini education license for video animations.

References

Mohammadmehdi Ataei and Hesam Salehipour. 2024. XLB: A differentiable massively parallel lattice Boltzmann library in Python. *Computer Physics Communications* 300 (2024), 109187.

Barnabás Börcsök and László Szécsi. 2023. Controlling 2D Laplacian Eigenfluids. In *Proceedings of CESCG 2023: The 27th Central European Seminar on Computer Graphics*. https://github.com/bobarna/eigenfluid-control.

TF Buttke. 1992. Lagrangian numerical methods which preserve the Hamiltonian structure of incompressible fluid flow. (1992).

Duowen Chen, Zhiqi Li, Taiyuan Zhang, Jinjin He, Junwei Zhou, Bart G van Bloemen Waanders, and Bo Zhu. 2025. Fluid Simulation on Compressible Flow Maps. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–17.

Duowen Chen, Zhiqi Li, Junwei Zhou, Fan Feng, Tao Du, and Bo Zhu. 2024b. Solid-Fluid Interaction on Particle Flow Maps. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–20.

Xuwen Chen, Xingyu Ni, Bo Zhu, Bin Wang, and Baoquan Chen. 2022. Simulation and optimization of magnetoelastic thin shells. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–18.

Yixin Chen, David Levin, and Timothy Langlois. 2024a. Fluid Control with Laplacian Eigenfunctions. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Ricardo Cortez. 1996. An impulse-based approximation of fluid motion due to boundary forces. *J. Comput. Phys.* 123, 2 (1996), 341–353.

Yitong Deng, Hong-Xing Yu, Jiajun Wu, and Bo Zhu. 2023a. Learning Vortex Dynamics for Fluid Inference and Prediction. *International Conference on Learning Representations (ICLR 2023)* (2023).

Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023b. Fluid Simulation on Neural Flow Maps. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–21.

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

Fan Feng, Jinyuan Liu, Shiying Xiong, Shuqi Yang, Yaorui Zhang, and Bo Zhu. 2022. Impulse Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* (2022).

Fan Feng, Shiying Xiong, Ziyue Liu, Zangyueyang Xian, Yuqing Zhou, Hiroki Kobayashi, Atsushi Kawamoto, Tsuyoshi Nomura, and Bo Zhu. 2023. Cellular topology optimization on differentiable Voronoi diagrams. *Internat. J. Numer. Methods Engrg.* 124, 1 (2023), 282–304.

Jakub Gałecki and Jacek Szumbarski. 2022. Adjoint-based optimal control of incompressible flows with convective-like energy-stable open boundary conditions. *Computers & Mathematics with Applications* 106 (2022), 40–56.

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

Michael B Giles, Mihai C Duta, Jens-Dominik Muller, and Niles A Pierce. 2003. Algorithm developments for discrete adjoint methods. *AIAA journal* 41, 2 (2003), 198–205.

Michael B Giles and Niles A Pierce. 2000. An introduction to the adjoint approach to design. *Flow, turbulence and combustion* 65, 3 (2000), 393–415.

Arvi Gjoka, Espen Knoop, Moritz Bächer, Denis Zorin, and Daniele Panozzo. 2024. Soft pneumatic actuator design using differentiable simulation. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

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

Jinjin He, Taiyuan Zhang, Hiroki Kobayashi, Atsushi Kawamoto, Yuqing Zhou, Tsuyoshi Nomura, and Bo Zhu. 2024. Multi-level Partition of Unity on Differentiable Moving Particles. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–21.

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

Philipp Holl and Nils Thuerey. 2024. Φ_{Flow} (PhiFlow): Differentiable Simulations for PyTorch, TensorFlow and Jax. In *International Conference on Machine Learning*. PMLR.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019a. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 201.

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

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. 2021. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311* (2021).

Zizhou Huang, Daniele Panozzo, and Denis Zorin. 2024a. Optimized shock-protecting microstructures. *ACM Trans. Graph.* 43, 6, Article 181 (Nov. 2024), 21 pages. https://doi.org/10.1145/3687765

Zizhou Huang, Davi Colli Tozoni, Arvi Gjoka, Zachary Ferguson, Teseo Schneider, Daniele Panozzo, and Denis Zorin. 2024b. Differentiable solver for time-dependent deformation problems with contact. *ACM Transactions on Graphics* 43, 3 (2024), 1–30.

Antony Jameson. 1988. Aerodynamic design via control theory. *Journal of scientific computing* 3, 3 (1988), 233–260.

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. 2023a. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512* (2023).

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

Yifei Li, Yuchen Sun, Pingchuan Ma, Eftychios Sifakis, Tao Du, Bo Zhu, and Wojciech Matusik. 2024c. Neuralfluid: Neural fluidic system design and control with differentiable simulation. *arXiv preprint arXiv:2405.14903* (2024).

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. 2018. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566* (2018).

Zhiqi Li, Barnabás Börcsök, Duowen Chen, Yutong Sun, Bo Zhu, and Greg Turk. 2024a. Lagrangian Covector Fluid with Free Surface. In *ACM SIGGRAPH 2024 (Conference Track)*.

Zhiqi Li, Duowen Chen, Candong Lin, Jinyuan Liu, and Bo Zhu. 2024b. Particle-Laden Fluid on Flow Maps. *arXiv preprint arXiv:2409.06246* (2024).

Zhiqi Li, Candong Lin, Duowen Chen, Xinyi Zhou, Shiying Xiong, and Bo Zhu. 2025a. Clebsch Gauge Fluid on Particle Flow Maps. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–12.

Zhiqi Li, Ruicheng Wang, Junlin Li, Duowen Chen, Sinan Wang, and Bo Zhu. 2025b. EDGE: Epsilon-Difference Gradient Evolution for Buffer-Free Flow Maps. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–11.

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

Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

Pingchuan Ma, Tao Du, Joshua B Tenenbaum, Wojciech Matusik, and Chuang Gan. 2022. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. *arXiv preprint arXiv:2205.05678* (2022).

Pingchuan Ma, Tao Du, John Z Zhang, Kui Wu, Andrew Spielberg, Robert K Katzschmann, and Wojciech Matusik. 2021. Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

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

Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. 2022. A level set theory for neural implicit evolution under explicit flows. In *European conference on computer vision*. Springer, 711–729.

Juan Sebastian Montes Maestre, Yinwei Du, Ronan Hinchet, Stelian Coros, and Bernhard Thomaszewski. 2023. Differentiable stripe patterns for inverse design of structured surfaces. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector fluids. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

Jorge Nocedal and Stephen J Wright. 1999. *Numerical optimization*. Springer.

Zherong Pan and Dinesh Manocha. 2017. Efficient solver for spacetime control of smoke. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.

Yiling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. 2021a. Differentiable simulation of soft multi-body systems. *Advances in Neural Information Processing Systems* 34 (2021), 17123–17135.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. 2020. Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168* (2020).

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

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Ole Sigmund. 2001. A 99 line topology optimization code written in Matlab. *Structural and multidisciplinary optimization* 21, 2 (2001), 120–127.

Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 121–128.

Arthur Stück. 2012. *Adjoint Navier-Stokes methods for hydrodynamic shape optimisation*. Technische Universität Hamburg.

Yuchen Sun, Junlin Li, Ruicheng Wang, Sinan Wang, Zhiqi Li, Bart G. van Bloemen Waanders, and Bo Zhu. 2025. Leapfrog Flow Maps for Real-Time Fluid Simulation. *ACM Transactions on Graphics* 43, 6 (Nov. 2025), 1–12. https://doi.org/10.1145/3687916

Tetsuya Takahashi, Junbang Liang, Yi-Ling Qiao, and Ming C Lin. 2021. Differentiable fluids with solid coupling for learning and control. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 6138–6146.

Jingwei Tang, Vinicius C. Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. 2021. Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 339–353.

Jerry Tessendorf and Brandon Pelfrey. 2011. The characteristic map for fast and efficient vfx fluid simulations. In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies. Ottawa, Canada.*

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*. 716–723.

Mengdi Wang, Fan Feng, Junlin Li, and Bo Zhu. 2025. Cirrus: Adaptive Hybrid Particle-Grid Flow Maps on GPU. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–17.

DC Wiggert and EB Wylie. 1976. Numerical predictions of two-dimensional transient groundwater flow by the method of characteristics. *Water Resources Research* 12, 5 (1976), 971–977.

Junwei Zhou, Duowen Chen, Molin Deng, Yitong Deng, Yuchen Sun, Sinan Wang, Shiying Xiong, and Bo Zhu. 2024. Eulerian-Lagrangian Fluid Simulation on Particle Flow Maps. *ACM Transactions on Graphics (SIGGRAPH 2024)* (2024).

Xian Zhou, Bo Zhu, Zhenjia Xu, Hsiao-Yu Tung, Antonio Torralba, Katerina Fragkiadaki, and Chuang Gan. 2023. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. *International Conference on Learning Representations (ICLR 2023)* (2023).

Bo Zhu, Mélina Skouras, Desai Chen, and Wojciech Matusik. 2017. Two-scale topology optimization with microstructures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.

Supplementary Material: An Adjoint Method for Differentiable Fluid Simulation on Flow Maps

ZHIQI LI*, Georgia Institute of Technology, USA
JINJIN HE*, Georgia Institute of Technology, USA
BARNABÁS BÖRCSÖK, Georgia Institute of Technology, USA
TAIYUAN ZHANG, Dartmouth College, USA
DUOWEN CHEN, Georgia Institute of Technology, USA
TAO DU, Independent Researcher,
MING C. LIN, University of Maryland, USA
GREG TURK, Georgia Institute of Technology, USA
BO ZHU, Georgia Institute of Technology, USA

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

ACM Reference Format:
Zhiqi Li, Jinjin He, Barnabás Börcsök, Taiyuan Zhang, Duowen Chen, Tao Du, Ming C. Lin, Greg Turk, and Bo Zhu. 2025. Supplementary Material: An Adjoint Method for Differentiable Fluid Simulation on Flow Maps. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3757377.3763903

A Forward Pass Calculation

For the forward pass, we accurately compute Equation 1 using the flow map by reformulating it in a path-integral form, following previous flow map methods [Chen et al. 2025, 2024; Li et al. 2024a,b, 2025a,b; Wang et al. 2025a,b]:

$$\begin{aligned}\mathbf{u}(\mathbf{x},t) &= \mathcal{T}_{t\to s}^{\top}(\mathbf{x})\mathbf{u}(\mathbf{\Psi}_{t\to s}(\mathbf{x}),s) + \mathcal{T}_{t\to s}^{\top}(\mathbf{x})\mathbf{\Gamma}_{s\to t}(\mathbf{\Psi}_{t\to s}(\mathbf{x})),\\ \mathbf{\Gamma}_{s\to t}(\mathbf{x}) &= \int_s^t \mathcal{F}_{s\to\tau}^{\top}(\mathbf{x})\left(-\frac{1}{\rho}\nabla p + \frac{1}{2}\nabla\|\mathbf{u}\|^2 + \mathbf{f}\right)(\mathbf{\Phi}_{s\to\tau}(\mathbf{x}),\tau)d\tau.\end{aligned} \tag{20}$$

The first term $\mathbf{u}^M_{s\to t}(\mathbf{x}) = \mathcal{T}_{t\to s}(\mathbf{x})^{\top}\mathbf{u}(\mathbf{\Psi}_{t\to s}(\mathbf{x}),s)$ in $\mathbf{u}(\mathbf{x},t)$ is referred to as the long-range mapped velocity, as it is directly obtained by mapping the accurate initial velocity $\mathbf{u}_s$ through the long-term flow map $\mathbf{\Psi}_{t\to s}$, which prevents the advection of velocity from being affected by accumulated errors. The second term $\mathbf{\Gamma}_{s\to t}$ is called the path integrator, since it accumulates the integral along the trajectory $\mathbf{S}_{\mathbf{x}_0}(t) = \mathbf{\Phi}_{s\to t}(\mathbf{x}_0)$ of any material point $\mathbf{x}_0 \in \mathbb{U}_s$ under the flow map. The passive field can also be obtained through long-range mapping based on the flow map

$$\xi(\mathbf{x},t) = \xi(\mathbf{\Psi}_{t\to s}(\mathbf{x}),s). \tag{21}$$

The above computation is generally carried out using the long-short term conversion technique proposed in [Li et al. 2024b]. With t_c denoting the current time and s the initial time:

(1) (**Mapping**) Calculate mapped velocity $\mathbf{u}^M_{s\to t_c}$ as

$$\mathbf{u}^M_{s\to t_c} = \mathcal{T}^{\top}_{t_c\to s}(\mathbf{x})\mathbf{u}(\mathbf{\Psi}_{t_c\to s}(\mathbf{x}),s). \tag{22}$$

(2) (**Conversion**) Convert $\mathbf{u}^M_{s\to t_c}$ to advected velocity $\mathbf{u}^A_{t_c'\to t_c}(\mathbf{x})$ using

$$\begin{aligned}\mathbf{u}^A_{t_c'\to t_c}(\mathbf{x}) = \mathbf{u}^M_{s\to t_c}(\mathbf{x}) &+ \mathcal{T}^{\top}_{t_c\to s}(\mathbf{x})\mathbf{\Gamma}_{s\to t_c'}(\mathbf{\Psi}_{t_c\to s}(\mathbf{x}))\\ &+ \frac{1}{2}\nabla\|\mathbf{u}\|^2\Delta t,\end{aligned} \tag{23}$$

where t_c' is the last step of t_c.

(3) (**Force Effect**) Compute the viscous force $\nu\Delta\mathbf{u}$ and external $\mathbf{f}$, and obtain the unprojected velocity as

$$\mathbf{u}^{\mathrm{up}}_{t_c} = \mathbf{u}^A_{t_c'\to t_c} + (\nu\Delta\mathbf{u} + \mathbf{f})\Delta t. \tag{24}$$

(4) (**Projection**) Perform velocity projection

$$\mathbf{u}_{t_c} = \mathbf{u}^{\mathrm{up}}_{t_c} - \frac{\Delta t}{\rho}\nabla p, \tag{25}$$

where p is solved from Poisson equation $\frac{\Delta t}{\rho}\Delta p = \nabla\cdot\mathbf{u}^{\mathrm{up}}_{t_c}$ with non-through boundary $\mathbf{u}_{t_c}\cdot\mathbf{n} = 0$.

(5) (**Path Integrator Update**) Update the path integrator as

$$\begin{aligned}\mathbf{\Gamma}_{s\to t_c}(\mathbf{x}) = \mathbf{\Gamma}_{s\to t_c'}(\mathbf{x}) &+ \Delta t\mathcal{F}^{\top}_{\mathbf{s}\to t_c}(\mathbf{x})\Big(\frac{1}{2}\nabla\|\mathbf{u}\|^2\\ &- \frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u}\Big)(\mathbf{\Phi}_{s\to t_c}(\mathbf{x}),t_c).\end{aligned} \tag{26}$$

Similar to the backward pass, the forward pass also suffers from the issue that repeated evolution of $\mathbf{\Psi}$ leads to an $O(m^2)$ time complexity. Therefore, we also apply our long-short time-sparse EFM in the forward pass.

We reinitialize the long-range flow map every $\Delta t^l_{\mathrm{reinit}} = n^l\Delta t$ and the short-range flow map every $\Delta t^s_{\mathrm{reinit}} = n^s\Delta t$, typically with $n^l = 15 \sim 60$ and $n^s = 1 \sim 3$. In Algorithm 3, we illustrate one long-range reinit cycle starting from the previous long-range reinit time

*Joint first author

Authors' Contact Information: Zhiqi Li, zli3167@gatech.edu, Georgia Institute of Technology, USA; Jinjin He, jhe433@gatech.edu, Georgia Institute of Technology, USA; Barnabás Börcsök, borcsok@gatech.edu, Georgia Institute of Technology, USA; Taiyuan Zhang, taiyuan.zhang.gr@dartmouth.edu, Dartmouth College, USA; Duowen Chen, dchen322@gatech.edu, Georgia Institute of Technology, USA; Tao Du, taodu.eecs@gmail.com, Independent Researcher,; Ming C. Lin, lin@umd.edu, University of Maryland, USA; Greg Turk, turk@cc.gatech.edu, Georgia Institute of Technology, USA; Bo Zhu, bo.zhu@gatech.edu, Georgia Institute of Technology, USA.

SA Conference Papers '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/2025/12
https://doi.org/10.1145/3757377.3763903

Algorithm 3 Long-Short Time-Sparse EFM of Forward Pass

Initialize: short-range reinit time s'' to long-range reinit time s'.

for each time step t_c between s' and $s' + \Delta t^l_{\text{reinit}}$, **do**
  March $\mathbf{\Phi}^l_{s'\to t_c}, \mathbf{\Phi}^s_{s''\to t_c}, \mathcal{F}^l_{s'\to t_c}, \mathcal{F}^s_{s''\to t_c}$ one step; ▷ eq. 5
  if $\exists m \in \mathbb{Z}$, st. $t_c = s' + m\Delta t^s_{\text{reinit}}$ **then**
    Calculate $\mathbf{\Psi}^s_{t_c\to s''}$ and $\mathcal{T}^s_{t_c\to s''}$ by integrating $\mathbf{\Psi}^s_{t_c\to t}$ and $\mathcal{T}^s_{t_c\to t}$ from t_c to s''; ▷ eq. 7
    Do mapping and conversion with $\mathbf{\Psi}^s_{t_c\to s''}$ and $\mathcal{T}^s_{t_c\to s''}$; ▷ eq. 22, 23
    Reset $\mathbf{\Phi}^s_{s''\to t_c}$ and $\mathcal{F}^s_{s''\to t_c}$ and set $s'' = t_c$;
  else
    Calculate advection with semi-Lagrangian method;
  Calculate force term and projection;
  Update $\mathbf{\Gamma}^l_{s'\to t_c}$ and $\mathbf{\Gamma}^s_{s''\to t_c}$; ▷ eq. 26
  if $t_c = s' + \Delta t^l_{\text{reinit}}$ **then**
    Calculate $\mathbf{\Psi}^l_{t_c\to s'}$ and $\mathcal{T}^l_{t_c\to s'}$ by integrating $\mathbf{\Psi}^l_{t_c\to t}$ and $\mathcal{T}^l_{t_c\to t}$ from t_c to s'; ▷ eq. 7
    Correct results by adding long-range mapping with $\mathbf{\Psi}^l_{t_c\to s'}$, $\mathcal{T}^l_{t_c\to s'}$ and path integrators. ▷ eq. 20

s'. Let s'' denote the most recent short-range reinit time, initialized as $s'' = s'$. We maintain two sets of flow maps, $\mathbf{\Phi}^l_{s'\to t}$, $\mathcal{F}^l_{s'\to t}$, $\mathbf{\Psi}^l_{t_c\to t}$, $\mathcal{T}^l_{t_c\to t}$ for long-range, $\mathbf{\Phi}^s_{s''\to t}$, $\mathcal{F}^s_{s''\to t}$, $\mathbf{\Psi}^s_{t_c\to t}$, $\mathcal{T}^s_{t_c\to t}$ for short-range and two path integrators $\mathbf{\Gamma}^l_{s'\to t}$ and $\mathbf{\Gamma}^s_{s''\to t}$ for long-range and short-range respectively, where $t_c > s'$ is the current time and t serves as the evolving time variable during integration.

B Missing Proofs

B.1 Proof of Equation 9

Similar to [Li et al. 2024b; Nabizadeh et al. 2022], we reformulate Equation 4 as $\left(\frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla)\right)\mathbf{u}^* + \nabla\mathbf{u}^\top\mathbf{u}^* = 2\nabla\mathbf{u}^\top\mathbf{u}^* + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p^* + \nu\Delta\mathbf{u}^* - \frac{\partial J}{\partial\mathbf{u}}$ and express it in covector form using the Lie derivative

$$\left(\frac{\partial}{\partial t} + \mathcal{L}_{\mathbf{u}}\right)\mathbf{u}^{*\flat} = \lambda^\flat, \tag{27}$$

where $\lambda(\mathbf{x},t) = \left(2\nabla\mathbf{u}^\top\mathbf{u}^* + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p^* + \nu\Delta\mathbf{u}^* - \frac{\partial J}{\partial\mathbf{u}}\right)(\mathbf{x},t)$, $\mathcal{L}_{\mathbf{u}}\mathbf{u}^{*\flat}$ denotes the Lie derivative of $\mathbf{u}^{*\flat}$, and $\flat$ is the musical isomorphism mapping a vector to a covector.

By integrating this equation of Lie derivative from r to t, we obtain

$$\begin{aligned}\mathbf{u}^{*\flat}_t &= \mathbf{\Phi}^*_{t\to r}\mathbf{u}^{*\flat}_r + \int_r^t (\mathbf{\Psi}_{r\to\tau}\circ\mathbf{\Phi}_{t\to r})^*\lambda^\flat_\tau d\tau \\ &= \mathbf{\Phi}^*_{t\to r}\mathbf{u}^{*\flat}_r + \mathbf{\Phi}^*_{t\to r}\int_r^t \mathbf{\Psi}^*_{r\to\tau}\lambda^\flat_\tau d\tau,\end{aligned} \tag{28}$$

where $\mathbf{\Psi}^*_{r\to\tau}$ and $\mathbf{\Phi}^*_{t\to r}$ are the pullbacks of the covector induced by $\mathbf{\Psi}_{r\to\tau}$ and $\mathbf{\Phi}_{t\to r}$, respectively. Convert the above expression back to vector form, and note that $\mathbf{\Phi}^*_{t\to r}\mathbf{v}^\flat$ and $\mathbf{\Psi}^*_{r\to\tau}\mathbf{v}^\flat$ corresponds to $\nabla\mathbf{\Phi}^\top_{t\to r}\mathbf{v}$ and $\nabla\mathbf{\Psi}^\top_{r\to\tau}\mathbf{v}$ respectively for arbitrary vector field $\mathbf{v}$. Then we get

$$\begin{aligned}\mathbf{u}^*(\mathbf{x},t) &= \mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{u}^*(\mathbf{\Phi}_{t\to r}(\mathbf{x}),r) + \mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t}(\mathbf{\Phi}_{t\to r}(\mathbf{x})),\\ \mathbf{\Lambda}^u_{r\to t}(\mathbf{x}) &= \int_r^t \mathcal{T}^\top_{r\to\tau}(\mathbf{x})\Big(2\nabla\mathbf{u}^\top\mathbf{u}^* + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p + \nu\Delta\mathbf{u}^* \\ &\qquad - \frac{\partial J}{\partial\mathbf{u}}\Big)(\mathbf{\Psi}_{r\to\tau}(\mathbf{x}),\tau)d\tau,\end{aligned} \tag{29}$$

For ξ^*, by directly integrating both sides of $\left(\frac{\partial}{\partial t} + (\mathbf{u}\cdot\nabla)\right)\xi^* = -\frac{\partial J}{\partial\xi}$ from r to t along the trajectory $\mathbf{S}_{\mathbf{x}}(\tau) = \mathbf{\Psi}_{r\to\tau}\circ\mathbf{\Phi}_{t\to r}(\mathbf{x})$, we obtain

$$\xi^*(\mathbf{x},t) - \xi^*(\mathbf{\Phi}_{t\to r}(\mathbf{x}),r) = -\int_r^t \frac{\partial J}{\partial\xi}(\mathbf{\Psi}_{r\to\tau}\circ\mathbf{\Phi}_{t\to r}(\mathbf{x}),\tau)d\tau \tag{30}$$

which gives the integral representation of ξ^*.

B.2 Proof of Equation 10

Following the approach of [Li et al. 2024b], we prove Equation 10, which establishes the relation between the short-range advected adjoint velocity $\mathbf{u}^{*A}_{t'\to t}$ and the long-range mapped adjoint velocity $\mathbf{u}^{*M}_{r\to t}$. First, we decompose $\mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t}(\mathbf{\Phi}_{t\to r}(\mathbf{x}))$ as

$$\begin{aligned}\mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t}(\mathbf{\Phi}_{t\to r}(\mathbf{x})) &= \mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t'}(\mathbf{\Phi}_{t\to r}(\mathbf{x})) \\ &+ \mathcal{F}^\top_{t\to r}(\mathbf{x})\int_{t'}^t \mathcal{T}^\top_{r\to\tau}(\mathbf{x})\lambda(\mathbf{\Psi}_{r\to\tau}(\mathbf{\Phi}_{t\to r}(\mathbf{x})),\tau)d\tau,\end{aligned} \tag{31}$$

where $\lambda(\mathbf{x},t) = \left(2\nabla\mathbf{u}^\top\mathbf{u}^* + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p^* + \nu\Delta\mathbf{u}^* - \frac{\partial J}{\partial\mathbf{u}}\right)(\mathbf{x},t)$. Here, the second part $\mathcal{F}^\top_{t\to r}(\mathbf{x})\int_{t'}^t \mathcal{T}^\top_{r\to\tau}(\mathbf{x})\lambda(\mathbf{\Psi}_{r\to\tau}(\mathbf{\Phi}_{t\to r}(\mathbf{x})),\tau)d\tau$ on the right-hand side of the above equation is equal to $\lambda(\mathbf{x},t)\Delta t$ when using the Euler scheme to compute with identities $\mathcal{F}^\top_{t\to r}\mathcal{T}^\top_{r\to t} = \mathbf{I}$ and $\mathbf{\Psi}_{r\to t}(\mathbf{\Phi}_{t\to r}(\mathbf{x})) = x$, where $\mathbf{I}$ is the identity matrix. Utilizing the relationship $\mathbf{u}^*(\mathbf{x},t) = \mathbf{u}^{*M}_{r\to t}(\mathbf{x}) + \mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t}(\mathbf{\Phi}_{t\to r}(\mathbf{x})) = \mathbf{u}^{*A}_{t'\to t}(\mathbf{x}) + \xi^*\nabla\xi - \frac{1}{\rho}\nabla p^* + \nu\Delta\mathbf{u}^* - \frac{\partial J}{\partial\mathbf{u}}$, we can derive the formula for converting $\mathbf{u}^{*M}_{r\to t}$ to $\mathbf{u}^{*A}_{t'\to t}$ as

$$\mathbf{u}^{*A}_{t'\to t}(\mathbf{x}) = \mathbf{u}^{*M}_{r\to t}(\mathbf{x}) + \mathcal{F}^\top_{t\to r}(\mathbf{x})\mathbf{\Lambda}^u_{r\to t'}(\mathbf{\Phi}_{t\to r}(\mathbf{x})) + 2\nabla\mathbf{u}^\top\mathbf{u}^*\Delta t. \tag{32}$$

C Numerical Algorithm for Forward Pass

To provide a clearer illustration of the forward pass, we present its detailed numerical algorithm in Algorithm 4. The discretization is the same as in the backward pass. The computation procedure is generally consistent with [Sun et al. 2025], mainly differing in the use of the Long-Short Time-Sparse EFM method.

In Algorithm 4, the midpoint computation strategy follows [Deng et al. 2023; Sun et al. 2025] and is consistent with that in the backward pass. The viscous term is computed in the same manner as the adjoint viscous term (Equation 18). The term $[\nabla\frac{1}{2}\|\mathbf{u}\|_2^2]_g$ is evaluated using a similar formulation as in Equation 19, but with a second-order kernel.

References

Duowen Chen, Zhiqi Li, Taiyuan Zhang, Jinjin He, Junwei Zhou, Bart G van Bloemen Waanders, and Bo Zhu. 2025. Fluid Simulation on Compressible Flow Maps. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–17.

Duowen Chen, Zhiqi Li, Junwei Zhou, Fan Feng, Tao Du, and Bo Zhu. 2024. Solid-Fluid Interaction on Particle Flow Maps. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–20.

Algorithm 4 Long-Short Time-Sparse EFM for Original Field

Initialize: $\mathbf{u}_{s',g}$, $\mathbf{u}_{s'',g}$ to initial velocity; $\mathcal{T}^s_{t_c\to s'',g}$, $\mathcal{F}^s_{s''\to t_c,g}$, $\mathcal{T}^l_{t_c\to s',g}$, $\mathcal{F}^l_{s'\to t_c,g}$ to $\mathbf{I}$; $\mathbf{\Psi}^s_{t_c\to s'',g}$, $\mathbf{\Phi}^s_{s''\to t_c,g}$, $\mathbf{\Psi}^l_{t_c\to s',g}$, $\mathbf{\Phi}^l_{s'\to t_c,g}$ to $\mathbf{x}_g$; $\mathbf{\Gamma}^l_{s'\to t_c,g}$ and $\mathbf{\Gamma}^s_{s''\to t_c,g}$ to 0; s', s'', t_c to s.

for each time step t_c from t_0 to t_n **do**
  Calculate midpoint velocity $\mathbf{u}_g^{\text{mid}}$;
  March $\mathbf{\Phi}^l_{s'\to t_c,g}$,$\mathbf{\Phi}^s_{s''\to t_c,g}$,$\mathcal{F}^l_{s'\to t_c,g}$,$\mathcal{F}^s_{s''\to t_c,g}$ one step; ▷ eq. 5
  if $c \pmod{n^s} = 0$ **then**
    Integrate $\mathcal{T}^s_{t_c\to s'',g}$ and $\mathbf{\Psi}^s_{t_c\to s'',g}$ from t_c to s''; ▷ eq. 7
    Calculate mapped velocity $\mathbf{u}^M_{s''\to t_c,g}$ and convert to one-step advected velocity $\mathbf{u}^A_{t_{c-1}\to t_c,g}$; ▷ eq. 22, 23
    Set initial time for short mapping s'' to t_c;
    Reinitialize $\mathcal{F}^s_{s''\to t_c,g}$ to $\mathbf{I}$, $\mathbf{\Phi}^s_{s''\to t_c,g}$ to $\mathbf{x}_g$;
    Calculate $\xi_{t_c,g}$ by $\mathbf{\Psi}^s_{t_c\to s'',g}$; ▷ eq. 21
  else
    Calculate $\mathbf{u}^A_{t_{c-1}\to t_c,g}$ and $\xi^A_{t_{c-1}\to t_c,g}$ by semi-Lagrangian;
  Compute viscous term $[\nu\Delta\mathbf{u}]_g$, force term $\mathbf{f}_g$ and $[\nabla\frac{1}{2}\|\mathbf{u}\|_2^2]_g$ on the grid;
  Compute unprojected velocity $\mathbf{u}^{\text{up}}_{t_c,g}$; ▷ eq. 24
  Calculate $\mathbf{u}_{t_c,g}$ using p from Poission equation; ▷ eq. 25
  Update both short and long path integrator $\mathbf{\Gamma}^l_{s'\to t_c,g}$ and $\mathbf{\Gamma}^s_{s''\to t_c,g}$; ▷ eq. 26
  if $c \pmod{n^l} = 0$ **then**
    Integrate $\mathcal{T}^l_{t_c\to s',g}$ and $\mathbf{\Psi}^l_{t_c\to s',g}$ from t_c to s'; ▷ eq. 7
    Calculate accurate $\mathbf{u}_{t_c,g}$ by mapping with $\mathcal{T}^l_{t_c\to s',g}$, $\mathbf{\Psi}^l_{t_c\to s',g}$ and integrator $\mathbf{\Lambda}^{u,l}_{r''\to t_c,g}$; ▷ eq. 20
    Set initial time for long mapping s' to t_c.

Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–21.

Zhiqi Li, Barnabás Börcsök, Duowen Chen, Yutong Sun, Bo Zhu, and Greg Turk. 2024a. Lagrangian Covector Fluid with Free Surface. In *ACM SIGGRAPH 2024 (Conference Track)*.

Zhiqi Li, Duowen Chen, Candong Lin, Jinyuan Liu, and Bo Zhu. 2024b. Particle-Laden Fluid on Flow Maps. *arXiv preprint arXiv:2409.06246* (2024).

Zhiqi Li, Candong Lin, Duowen Chen, Xinyi Zhou, Shiying Xiong, and Bo Zhu. 2025a. Clebsch Gauge Fluid on Particle Flow Maps. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–12.

Zhiqi Li, Ruicheng Wang, Junlin Li, Duowen Chen, Sinan Wang, and Bo Zhu. 2025b. EDGE: Epsilon-Difference Gradient Evolution for Buffer-Free Flow Maps. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–11.

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector fluids. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

Yuchen Sun, Junlin Li, Ruicheng Wang, Sinan Wang, Zhiqi Li, Bart G. van Bloemen Waanders, and Bo Zhu. 2025. Leapfrog Flow Maps for Real-Time Fluid Simulation. *ACM Transactions on Graphics* 43, 6 (Nov. 2025), 1–12. https://doi.org/10.1145/3687916

Mengdi Wang, Fan Feng, Junlin Li, and Bo Zhu. 2025a. Cirrus: Adaptive Hybrid Particle-Grid Flow Maps on GPU. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–17.

Sinan Wang, Junwei Zhou, Fan Feng, Zhiqi Li, Yuchen Sun, Duowen Chen, Greg Turk, and Bo Zhu. 2025b. Fluid Simulation on Vortex Particle Flow Maps. *arXiv preprint arXiv:2505.21946* (2025).