SIMDFKS: DIVERGENCE-FREE KERNELS AS SPATIAL REPRESENTATIONS FOR FLOW SIMULATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fluid dynamics has posed enduring challenges in computational physics and computer graphics, where resolving rich, multiscale flow structures demands spatial representations that are both expressive and computationally efficient. Recent studies have shown that kernel-based spatial representations, which approximate flow fields as superpositions of localized basis functions, can outperform implicit neural representations (INRs) in both flow reconstruction and simulation while retaining a compact continuous parameterization. In this work, we extend divergence-free kernels (DFKs) from flow reconstruction to forward fluid simulation, introducing a key reformulation that makes evolving the continuous representation under PDEs practical and stable. Building on this representation, our optimization-based framework evolves the kernel parameters under advection, external forces, and complex boundary conditions while preserving incompressibility by construction. Our reformulation retains the compactness and expressiveness of the original DFK formulation while substantially improving the efficiency and numerical accuracy of derivative-sensitive PDE optimization. Without requiring training on precomputed data, our method produces high-quality forward simulations of a variety of 2D and 3D fluid phenomena from given initial conditions. The same framework also enables simulation recovery, i.e., reconstructing a fluid-dynamics-consistent flow field sequence from sparse vorticity observations of results produced by an existing solver. This capability is not directly available to conventional forward numerical solvers.

1 INTRODUCTION

Fluid simulation remains a longstanding challenge in computational physics and computer graphics, demanding both expressive representations to capture multiscale spatial details and accurate evolutions to resolve chaotic non-linear temporal dynamics. Traditional solvers typically rely on explicit spatial discretizations, such as Eulerian grids or Lagrangian particles, where degrees of freedom (DoFs) represent physical quantities at discrete spatial locations throughout the simulation domain (Bridson, 2015). They have long served as the standard for forward simulation, as their explicit structure supports efficient and well-established numerical evolution. However, their accuracy and cost are closely tied to discretization and resolution: resolving finer flow structures requires rapidly growing spatial DoFs and introduces discretization-dependent errors. Moreover, their discretized representations are less naturally suited for inverse tasks with incomplete observations.

Such limitations have motivated compact continuous representations of flow fields, initially dominated by implicit neural representations (INRs), which use lightweight neural networks to map spatial coordinates to field values. In inverse tasks such as flow reconstruction, INRs provide compact and globally optimizable representations of dynamic flows with substantially fewer DoFs (Yu et al., 2023; Qiu et al., 2024; Tao et al., 2025; Zhang et al., 2026). Their compactness, spatial adaptivity, and continuous differentiability also make continuous representations highly appealing for forward simulation (Chen et al., 2023; Jain et al., 2024; Wang et al., 2024a; 2025a). However, unlike explicit discretizations with well-established update rules, forward simulation now requires evolving a globally parameterized continuous field according to the governing PDEs. Existing approaches remain less mature than classical solvers, exhibiting substantial trade-offs in accuracy, stability, and computational cost.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Recent studies suggest kernel-based continuous representations as a promising alternative to neural parameterizations. Gaussian Fluids (GSR) (Xing et al., 2025) represents velocity fields as superpositions of Gaussian kernels. With efficient analytic spatial derivatives and improved locality, GSR achieves higher fidelity and efficiency than INRs in forward simulation, particularly in preserving vortical structures. Yet its incompressibility is enforced only approximately through an optimization penalty. Divergence-free kernels (DFKs) instead provide a compact kernel representation that embeds incompressibility directly into the represented velocity field. They have shown strong performance in flow reconstruction (Ni et al., 2025; Tao et al., 2026), but have so far been limited to reconstruction. This raises an appealing possibility: a continuous representation that combines stronger physical structure for forward evolution with the optimization flexibility needed for inverse reconstruction. We therefore ask whether DFKs can advance continuous forward fluid simulation while retaining this inverse flexibility.

Turning DFKs into an effective forward representation, however, is not straightforward. For continuous representations originally designed for reconstruction, forward simulation is substantially more sensitive to accumulated numerical errors. Directly applying the original DFK formulation also introduces unnecessarily high-order derivatives for simulation, which not only incur substantial computational overhead but also make the optimization considerably more difficult. We therefore introduce a simple but essential single-curl reformulation for velocity, along with a carefully designed forward simulation pipeline which starts from a fitted initial condition and applies time integration optimizations for each following timestep. Beyond forward simulation, the same representation enables a more challenging simulation recovery task, where only partial flow observations are available throughout the time horizon. By coupling these observations through forward and backward filtering, our method jointly recovers the complete dynamics over the spatial domain. Our reformulation reduces the derivative order without losing exact fluid incompressibility of the original formulation. Together with our optimization strategies, the full method substantially improves simulation robustness, accuracy and efficiency over long-term evolution compared to the original DFKs as well as INR and GSR-based methods.

Our main contributions are summarized as follows:

- We introduce a **forward-oriented DFK representation** with a simple single-curl reformulation that improves robustness while preserving exact incompressibility.
- We develop a **forward simulation framework** based on vorticity transport and optimization, supporting advection, external forces, and complex boundary conditions.
- We extend the same representation to **simulation recovery**, using forward and backward transport to reconstruct complete dynamics from partial observations.

2 Related Work

Classical fluid simulations. Grid, particle, and hybrid discretizations form the basis of established fluid solvers (Bridson, 2015). Grid-based solvers commonly combine pressure projection (Chorin, 1968) with semi-Lagrangian advection (Stam, 1999). Smoothed particle hydrodynamics (SPH) uses moving particles (Koschier et al., 2022), with incompressible formulations using different variants of pressure correction (Solenthaler & Pajarola, 2009; Ihmsen et al., 2013; Bender & Koschier, 2015). FLIP and APIC combine particle transport with grid-based updates (Brackbill & Ruppel, 1986; Jiang et al., 2015). To retain vortical detail, methods use vorticity confinement (Fedkiw et al., 2001), improved advection (Selle et al., 2008; Zehnder et al., 2018), or explicit vortex particles, filaments, and sheets (Selle et al., 2005; Weißmann & Pinkall, 2010; Pfaff et al., 2012). Covector Fluids (Nabizadeh et al., 2022) preserves circulation during advection, while Eulerian vortex methods on flow maps (Wang et al., 2024b) transport vorticity before reconstructing velocity.

Divergence-free representations. Incompressibility can be embedded into continuous velocity representations through potentials (Chang et al., 2022; Lyu et al., 2024), compatible grid interpolation (Schroeder et al., 2022; Roy-Chowdhury et al., 2024; Nabizadeh et al., 2024), or constrained neural fields (Kim et al., 2019; Richter-Powell et al., 2022). Matrix-valued radial basis functions provide a meshfree alternative (Narcowich & Ward, 1994; Lowitzsch, 2005), supporting Stokes solvers and Leray projection (Wendland, 2009; Fuselier et al., 2016). Using compactly supported

Wendland functions (Wendland, 1995), Ni et al. (2025) optimize divergence-free kernel centers, radii, and strengths for flow reconstruction. We adopt a single-curl velocity formulation in place of their double-curl kernels, so vorticity requires second rather than third derivatives of the radial basis while velocity remains analytically divergence-free.

Physics-constrained flow reconstruction. Physical constraints help infer flow from incomplete observations by coupling observation fitting with incompressibility, simulation, and temporal transport (Gregson et al., 2014; Eckert et al., 2019; Franz et al., 2021). Physics-informed neural networks (Raissi et al., 2019) incorporate PDE residuals, enabling velocity and pressure recovery from transported scalars (Raissi et al., 2020). For smoke reconstruction, PINF and HyFluid combine neural rendering with fluid constraints (Chu et al., 2022; Yu et al., 2023), while PICT, NeuSmoke, and FlowCapX emphasize temporal transport and consistency (Wang et al., 2024c; Qiu et al., 2024; Tao et al., 2025). Kernel-based approaches include divergence-free velocity fitting with DFKs (Ni et al., 2025), physically guided Gaussian optimization in GauSmoke (Zhang et al., 2026), and DFK-driven transport of Gaussian density primitives in LagrangianSplats (Tao et al., 2026).

Memory-efficient representations for simulation. Sparse grids (Setaluri et al., 2014) and neural volume compression (Kim et al., 2024) reduce storage costs, while Neural Flow Maps (Deng et al., 2023) stores velocity histories compactly for long-range flow-map calculations in a grid-based solver. INSR (Chen et al., 2023) instead evolves a neural spatial representation through per-step optimization, without precomputed simulation data. Related neural solvers incorporate Monte Carlo pressure solves (Jain et al., 2024), multiresolution spatial encodings (Wang et al., 2024a), and divergence-free tangent fields for surface flows (Wang et al., 2025a). Gaussian Fluids (Xing et al., 2025) directly represents velocity with Gaussian kernels and advances their parameters by fitting vorticity, divergence, and boundary constraints using analytic spatial derivatives. We retain optimization-based temporal evolution while enforcing incompressibility through divergence-free kernels, eliminating the divergence penalty.

3 Problem Statement

Our goal is to compute the dynamics of an inviscid, incompressible flow, represented by a space-time velocity field $\boldsymbol{u} : \Omega \times [0, T] \to \mathbb{R}^d$. Here, $\Omega \subset \mathbb{R}^d$ is the fluid domain, with $d \in \{2, 3\}$, and $[0, T]$ is the time interval of interest. The velocity satisfies the incompressible Euler equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{g}, \quad \nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

where ρ is the constant fluid density, p is pressure, and $\boldsymbol{g}$ is an externally prescribed acceleration. Two types of boundary consitions are enforced to the velocity field as

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}_{\mathrm{b}}(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \Gamma_1, \tag{2}$$

$$\boldsymbol{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n}(\boldsymbol{x}, t) = f_{\mathrm{b}}(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \Gamma_2, \tag{3}$$

where $\boldsymbol{n}$ is the unit boundary normal, $\boldsymbol{u}_{\mathrm{b}}, f_{\mathrm{b}}$ are known functions relative to boundary movement, and $\Gamma_1, \Gamma_2 \subseteq \partial\Omega$ are sets where the no-slip and free-slip boundary conditions apply, respectively. We suppress the time dependence of Ω and Γ_1, Γ_2 when describing moving boundaries.

In our tasks, the dynamics of the flow are partially observed as

$$\mathcal{L}\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{h}_{\mathrm{given}}(\boldsymbol{x}, t), \quad \forall (\boldsymbol{x}, t) \in \mathbb{D}_{\mathrm{given}}, \tag{4}$$

where $\mathcal{L}$ is an observation operator, $\boldsymbol{h}_{\mathrm{given}}$ contains the corresponding known values, and $\mathbb{D}_{\mathrm{given}} \subset \Omega \times [0, T]$ is the observed region in the spatiotemporal domain. The choice of $\mathcal{L}$ specifies the observed quantity. When multiple quantities are supplied, Eq. (4) is imposed for each corresponding operator and data field. Forward simulation uses velocity and its spatial gradient at the initial time, corresponding to the identity and gradient operators, respectively (Sec. 4.2), while simulation recovery uses vorticity observations distributed across time, corresponding to the curl operator (Sec. 4.3). This formulation therefore unifies both settings, which we address in the next section using the same continuous DFK representation within a unified optimization framework.

4 DFKs FOR FLUID SIMULATION

We use divergence-free kernels (DFKs) as a continuous spatial representation whose parameters evolve through a sequence of optimizations. We first introduce the representation (Sec. 4.1), then describe forward simulation (Sec. 4.2) and incomplete flow observation recovery (Sec. 4.3).

4.1 DFKs AS SPATIAL REPRESENTATIONS

At discrete times $t_n = n\Delta t$ with step size Δt, we represent the velocity field $\boldsymbol{u}(\boldsymbol{x}, t_n)$ by $\boldsymbol{u}_{\boldsymbol{\theta}^n}(\boldsymbol{x})$, where $\boldsymbol{\theta}^n$ denotes the optimizable parameters of our DFKs at t_n. This separates the spatial representation from time integration (described in Sec. 4.2.2), following Chen et al. (2023). To describe the spatial representation at a fixed time, we omit the superscript n and write $\boldsymbol{\theta} = \{\boldsymbol{p}_i, h_i, \boldsymbol{a}_i\}_{i=1}^N$ for the parameters of N kernels. Each kernel has an optimizable center $\boldsymbol{p}_i \in \mathbb{R}^d$, support radius $h_i > 0$, and strength vector $\boldsymbol{a}_i \in \mathbb{R}^3$.

We use a unified three-dimensional curl notation: a planar flow is extended independently of x_3, with $\boldsymbol{u} = (u_1, u_2, 0)^\top$ and $\boldsymbol{\omega} = \nabla \times \boldsymbol{u} = (0, 0, \omega)^\top$. Thus, the two-dimensional case is a restriction of the same vector formulation. The kernels define a vector potential

$$\boldsymbol{A}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{N} \phi_i(\boldsymbol{x}) \boldsymbol{a}_i, \qquad \phi_i(\boldsymbol{x}) = h_i^2 \phi\left(\frac{\|\boldsymbol{x} - \boldsymbol{p}_i\|_2}{h_i}\right), \tag{5}$$

where ϕ is Wendland's compactly supported $\mathcal{C}^4$ radial basis function (Wendland, 1995). For planar flows, we take $\boldsymbol{a}_i = s_i \boldsymbol{e}_3$ and measure radial distance in (x_1, x_2), so $\boldsymbol{A}_{\boldsymbol{\theta}}$ is independent of x_3. Velocity and vorticity then follow from the same potential in either dimension:

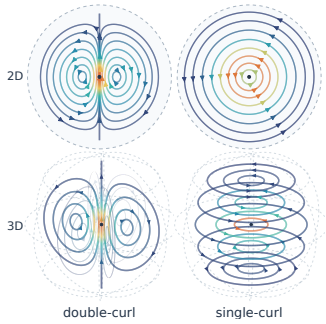
$$\boldsymbol{u}_{\boldsymbol{\theta}} = \boldsymbol{u}_{\text{base}} + \nabla \times \boldsymbol{A}_{\boldsymbol{\theta}}, \qquad \boldsymbol{\omega}_{\boldsymbol{\theta}} = \nabla \times \boldsymbol{u}_{\boldsymbol{\theta}} = \nabla \times \nabla \times \boldsymbol{A}_{\boldsymbol{\theta}}. \tag{6}$$

Here, $\boldsymbol{u}_{\text{base}} \in \mathbb{R}^d$ is a prescribed constant velocity fixed across space and time. Every parameter choice gives $\nabla \cdot \boldsymbol{u}_{\boldsymbol{\theta}} = 0$ analytically. Detailed formulations are specified in App. A.1.

The formulation of our spatial representation is inspired by Ni et al. (2025), with an essential modification. In Ni et al. (2025), a single kernel function written as

$$\left[(-\boldsymbol{I}\nabla^2 + \nabla\nabla^\top)\phi_i\right]\boldsymbol{a}_i = \nabla \times \nabla \times (\phi_i \boldsymbol{a}_i) \tag{7}$$

is used as basis function for the velocity field, while the same mathematical form is for the vorticity field in our method by taking the inner curl as our velocity basis. The inset figure illustrates the resulting flow structures for a single kernel, where colors of the streamlines indicate velocity magnitude, and dashed curves mark the kernel support. In 2D, double-curl produces two counter-rotating lobes, whereas single-curl produces concentric circular streamlines. In 3D, double-curl produces meridional recirculation, with flow along the axis returning around it, while single-curl produces azimuthal circulation about the axis. Our reformulation suits simulation tasks whose main optimization target is vorticity (see Eq. (12) below): evaluating it requires second derivatives of the radial basis instead of the third derivatives required by a double-curl velocity representation, removing one curl evaluation. It also places the double-curl construction on the quantity being fitted, motivated by the stronger reconstruction performance of double-curl kernels over single-curl kernels reported by Ni et al. (2025). Our ablation study (Sec. 5.4) will compare the single-curl and double-curl velocity representations in simulation accuracy and computational cost.



4.2 FORWARD SIMULATION

For forward simulation in two or three dimensions, $\mathbb{D}_{\text{given}} = \Omega \times \{0\}$, and the supplied data comprise initial velocity $\boldsymbol{u}_0$ and its Jacobian $\boldsymbol{G}_0 = \nabla \boldsymbol{u}_0$. These give two instances of Eq. (4) on the same initial slice: $\mathcal{L} = \text{Id}$ with $\boldsymbol{h}_{\text{given}} = \boldsymbol{u}_0$, and $\mathcal{L} = \nabla$ with $\boldsymbol{h}_{\text{given}} = \boldsymbol{G}_0$. We first fit these data as an initialization process (Sec. 4.2.1 below), then follow the time integration scheme of a traditional simulation solver by evolving the DFKs frame by frame (Sec. 4.2.2 below). For queries $\mathcal{Q}$, the

componentwise mean absolute discrepancy is $\mathcal{D}_{\mathcal{Q}}(f,g) = \frac{1}{m|\mathcal{Q}|}\sum_{\boldsymbol{x}\in\mathcal{Q}}\|f(\boldsymbol{x}) - g(\boldsymbol{x})\|_1$, where m counts physical scalar components (App. A.2). Alg. 1 in App. A.8 summarizes the forward procedure detailed below.

4.2.1 INITIALIZATION

We initialize kernels on a padded lattice with a common radius and zero strengths (App. A.2). For an unforced initial field, we fit velocity and its Jacobian through

$$\mathcal{L}_{\text{init}}(\boldsymbol{\theta}) = \mathcal{D}_{\mathcal{Q}}(\boldsymbol{u}_{\boldsymbol{\theta}}, \boldsymbol{u}_0) + \mathcal{D}_{\mathcal{Q}}(\nabla\boldsymbol{u}_{\boldsymbol{\theta}}, \boldsymbol{G}_0). \tag{8}$$

The Jacobian term constrains local flow structure, including vorticity. We jointly optimize all kernel parameters with Adam, using fresh spatial samples at each iteration. For some cases, we also perform a boundary-velocity fit or a smoke startup fit to the initial buoyancy response (Apps. A.2 and A.4).

4.2.2 TIME INTEGRATION

Each time step optimizes the DFK parameters to fit the evolution of vorticity together with the prescribed boundary velocities (the following paragraphs detail these stages). Taking the curl of Eq. (1) eliminates pressure and gives

$$\frac{\partial\boldsymbol{\omega}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{\omega} = (\boldsymbol{\omega}\cdot\nabla)\boldsymbol{u} + \nabla\times\boldsymbol{g}. \tag{9}$$

Vorticity transport. We construct a semi-Lagrangian target (Sawyer, 1963) from the frozen previous field $\boldsymbol{u}^n = \boldsymbol{u}_{\boldsymbol{\theta}^n}$. Let $\mathcal{B}_n(\boldsymbol{x})$ be the departure point obtained by tracing $\boldsymbol{x}$ backward for Δt through this field using fourth-order Runge–Kutta (RK4), and let $\boldsymbol{J}_n(\boldsymbol{x}) = D\mathcal{B}_n(\boldsymbol{x})$ be the spatial Jacobian of this backward map.

In 2D, the stretching term $(\boldsymbol{\omega}\cdot\nabla)\boldsymbol{u}$ vanishes, so unforced vorticity is constant along trajectories. In 3D, we follow Wang et al. (2024b) and transport vorticity as a material line element, accounting for vortex stretching and tilting through the inverse backward-map Jacobian. The transport target is

$$\widehat{\boldsymbol{\omega}}^{n+1}_{\text{adv}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{\omega}^n\big(\mathcal{B}_n(\boldsymbol{x})\big), & \text{in 2D,} \\ \boldsymbol{J}_n(\boldsymbol{x})^{-1}\boldsymbol{\omega}^n\big(\mathcal{B}_n(\boldsymbol{x})\big), & \text{in 3D.} \end{cases} \tag{10}$$

Exterior targets follow the scene-specific extrapolation rules in App. A.3.

External forces. An external acceleration contributes its curl to the vorticity target. We add this source explicitly at the query point and fit the new target through the vorticity loss:

$$\widehat{\boldsymbol{\omega}}^{n+1}(\boldsymbol{x}) = \widehat{\boldsymbol{\omega}}^{n+1}_{\text{adv}}(\boldsymbol{x}) + \Delta t\,(\nabla\times\boldsymbol{g}^n)(\boldsymbol{x}), \tag{11}$$

$$\mathcal{L}_{\omega}(\boldsymbol{\theta}) = \mathcal{D}_{\mathcal{Q}}(\boldsymbol{\omega}_{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}}^{n+1}). \tag{12}$$

For smoke, $\boldsymbol{g}^n$ is density-dependent buoyancy; otherwise $\boldsymbol{g}^n = \boldsymbol{0}$. Apps. A.4 and A.3 detail scalar advection and frozen-field tracing.

Boundary conditions. Vorticity fitting is coupled to the prescribed boundary velocities. With sampled boundary sets $\mathcal{Q}_1$ and $\mathcal{Q}_2$ at t_{n+1}, we define the penalties

$$\mathcal{L}_{\text{b1}}(\boldsymbol{\theta}) = \mathcal{D}_{\mathcal{Q}_1}\big(\boldsymbol{u}_{\boldsymbol{\theta}}, \boldsymbol{u}_{\text{b}}(\cdot, t_{n+1})\big), \tag{13}$$

$$\mathcal{L}_{\text{b2}}(\boldsymbol{\theta}) = \mathcal{D}_{\mathcal{Q}_2}\big(\boldsymbol{u}_{\boldsymbol{\theta}}\cdot\boldsymbol{n}(\cdot, t_{n+1}), f_{\text{b}}(\cdot, t_{n+1})\big). \tag{14}$$

These penalties constrain the boundary behavior while incompressibility is built into the representation (boundary sampling and case-specific conditions are detailed in App. A.3).

Optimization. Combining the vorticity and boundary penalties, our time-step objective is

$$\boldsymbol{\theta}^{n+1} \approx \arg\min_{\boldsymbol{\theta}} \left[\mathcal{L}_{\omega}(\boldsymbol{\theta}) + \lambda_1\mathcal{L}_{\text{b1}}(\boldsymbol{\theta}) + \lambda_2\mathcal{L}_{\text{b2}}(\boldsymbol{\theta})\right], \tag{15}$$

with absent boundary terms omitted. The boundary-loss weights λ_1 and λ_2 are specified in App. A.3. Consequently, a time step fits the evolved vorticity and boundary data directly in a divergence-free function space, without introducing a pressure variable or a pressure Poisson solve.

Starting from the preceding kernel configuration, Adam jointly updates centers, radii, and normalized strengths, holding the previous field fixed and excluding target construction from differentiation. Amplitudes are normalized during each fit and restored afterward (App. A.2). Smoke is subsequently advected with the fitted velocity (App. A.4).

4.2.3 Reinitialization

To restore spatial coverage, we periodically fit a fresh kernel set to the current velocity and Jacobian using Eq. (8), then insert kernels at old centers with large Jacobian errors and refit. This transfers the field without advancing time; App. A.5 specifies selection and fitting.

4.3 Simulation Recovery

For K frames, vorticity observations $\boldsymbol{\omega}^n_{\text{obs}}$ are given at $\mathcal{O}_n \subset \Omega$, which may be empty. Thus $\mathbb{D}_{\text{given}} = \bigcup_{n=0}^{K-1}(\mathcal{O}_n \times \{t_n\})$ and $\mathcal{L} = \nabla\times$. We recover the framewise parameters $\Theta = \{\boldsymbol{\theta}^n\}_{n=0}^{K-1}$ by alternating advective filtering and refitting.

At filtering round r, $\boldsymbol{u}^{n,(r)}$ and $\boldsymbol{\omega}^{n,(r)}$ denote the current velocity and vorticity at frame n. The map $\Psi^{(r)}_{n\to j}(\boldsymbol{x})$ traces a query $\boldsymbol{x} \in \Omega$ from frame n to j using chained one-frame RK4 steps through these velocities. For temporal half-window R, define the neighboring frames $\mathcal{N}_R(n) = \{j : 0 \le j < K,\ 1 \le |j-n| \le R\}$. The model-based target averages the current and transported vorticities:

$$\overline{\boldsymbol{\omega}}^{n,(r)}(\boldsymbol{x}) = \frac{\boldsymbol{\omega}^{n,(r)}(\boldsymbol{x}) + \sum_{j\in\mathcal{N}_R(n)} \boldsymbol{\omega}^{j,(r)}\left(\Psi^{(r)}_{n\to j}(\boldsymbol{x})\right)}{1 + |\mathcal{N}_R(n)|}. \tag{16}$$

If traces reach observed cells in neighboring frames, their mean observation-conditioned vorticity replaces this average, giving $\widehat{\boldsymbol{\omega}}^{n,(r)}$. With all targets fixed, each frame is refitted over the full-domain grid $\mathcal{G}$:

$$\boldsymbol{\theta}^{n,(r+1)} \approx \arg\min_{\boldsymbol{\theta}}\ \mathcal{D}_{\mathcal{Q}_n}\left(\boldsymbol{\omega}_{\boldsymbol{\theta}}, \widehat{\boldsymbol{\omega}}^{n,(r)}\right), \qquad \mathcal{Q}_n \subset \mathcal{G} \subset \Omega, \tag{17}$$

The refitted sequence supplies the next round, without differentiating through trajectories. App. A.6 and Alg. 2 detail the reference-assisted protocol. Recovered velocities remain analytically divergence-free by Eq. (6).

5 Results

Our forward simulations share one set of learning rates and iteration budgets across all 2D cases and another across both 3D cases (App. A.2). Reinitialization schedules are specified in App. A.5. Scene domains, initial conditions, and obstacle geometry are specified in Appendices B and C, and ablation protocols in App. D.

5.1 Quantitative Evaluation

We evaluate accuracy, representation size, and running time using forward simulation of the stationary Taylor–Green vortex. Initial fitting and subsequent evolution test the spatial approximation and transport components shared by both simulation tasks. We compare against an Eulerian MAC-grid solver, INSR (Chen et al., 2023), NMC (Jain et al., 2024), and Gaussian fluids (GSR) (Xing et al., 2025); detailed settings are given in App. B.

Table 1 shows that our method maintains the lowest velocity error with little accumulation over time, while using fewer velocity parameters than GSR and NMC and running fastest among the optimization-based methods.

Table 1: Taylor–Green comparison of velocity parameters, running time per frame, and velocity MSE at frames 0, 50, and 100. Bold and underlined values indicate the lowest and second-lowest values in each column, respectively.

Method	Velocity Params.	Running Time (s)	Frame 0	Frame 50	Frame 100
Eulerian	8320	**0.00155**	2.432×10^{-4}	9.757×10^{-3}	2.019×10^{-2}
INSR	**3330**	403	8.998×10^{-7}	1.715×10^{-5}	1.992×10^{-5}
NMC	25282	39	1.829×10^{-4}	6.492×10^{-4}	1.725×10^{-3}
GSR	4032	38	$\underline{9.957 \times 10^{-8}}$	$\underline{2.510 \times 10^{-7}}$	$\underline{2.181 \times 10^{-7}}$
ours	<u>3602</u>	<u>23</u>	$\mathbf{6.079 \times 10^{-8}}$	$\mathbf{6.132 \times 10^{-8}}$	$\mathbf{6.218 \times 10^{-8}}$

5.2 Validation on Forward Simulation

Vortex preservation. We validate the ability of our method to preserve vortices with three examples. We first compare against a semi-implicit Eulerian solver and INSR (Chen et al., 2023) on the Taylor vortex example, in which two initially separated vortices rotate in the same direction and develop thin filaments (initial conditions in App. C.1). Using the vortex-in-cell solution as the reference in Appendix Fig. 7, our method better preserves vorticity magnitude and produces less background noise than INSR, while both methods preserve the thin vortical structures well. Our method achieves nearly the same accuracy as the Eulerian solver while using only about 5% of its velocity-grid storage.

The 2D leapfrogging example initializes two pairs of counter-rotating vortices that repeatedly pass through one another (Appendix Fig. 8; detailed settings in App. C.1). The vortices remain distinct until the fifth leap, when like-signed cores merge, demonstrating sustained vortex preservation through repeated interactions.

Finally, the 3D leapfrogging example initializes two coaxial vortex rings of unequal radii moving in the same direction, with each passage of one ring through the other forming a leap (App. C.1). In Appendix Fig. 9, covector fluids (Nabizadeh et al., 2022), Neural Flow Maps (NFM) (Deng et al., 2023), Vortex Particle Flow Maps (VPFM) (Wang et al., 2025b), Gaussian fluids (Xing et al., 2025), and our method exhibit two, two, three, three, and four leaps, respectively, with merger or breakup during the final interaction. Our method preserves coherent vortex rings through more successive leapfrogging interactions while using only 11.3% of the velocity-grid storage.

Boundary handling. We demonstrate the capability of our method in handling boundary conditions through two examples in which the vorticity dynamics are dominated by solid boundaries. The Kármán vortex-street example drives a uniform inflow past a stationary circular obstacle in a channel, with a no-slip condition on the obstacle and impermeable upper and lower walls (detailed settings in App. C.2). Vorticity generated at the obstacle rolls up and sheds alternately from its two sides, forming a staggered vortex street downstream. Appendix Fig. 10 compares NMC (Jain et al., 2024), Gaussian fluids (Xing et al., 2025), and our method, using the traditional grid-based solver Stable and Circulation Preserving Fluids (SCPF) (Elcott et al., 2007) as the reference. In this experiment, Gaussian fluids and our method maintain stable vortex shedding, whereas the INR-based NMC develops spurious vorticity, indicating greater stability of the kernel-based representations. The two kernel methods exhibit different numerical viscosity, with our method producing a vortex street closer to the SCPF reference.

We extend the boundary-handling validation to a moving obstacle by testing a cross rotating inside a closed square container, with free-slip, no-penetration boundary conditions on both the cross and the container walls (App. C.2). As the cross turns, our simulation captures the roll-up of tip vortices into four coherent cores surrounded by thin filaments, which persist as the obstacle continues to rotate (Appendix Fig. 11).

Incompressibility. We compare with Gaussian fluids (Xing et al., 2025), using covector fluids (Nabizadeh et al., 2022) as a numerical reference (Fig. 1). In a closed vessel of width 10, a piston of width 8 descends at speed 0.5, with no-slip conditions on the piston and impermeable, free-slip vessel walls. Incompressibility requires a combined upward flux $Q = 8 \times 0.5 = 4$ through the two side gaps at the piston's top edge. Our flux remains near 4 throughout the simulation; at frame 60, it is 3.9990, compared with 4.0000 for covector fluids and -0.1649 for Gaussian fluids. Our method also reproduces the reference's upward gap flow and recirculation above the piston,

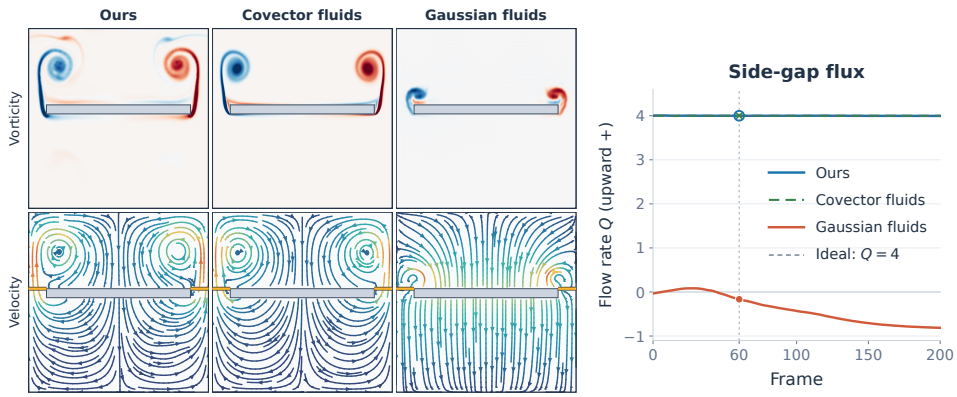


Figure 1: Piston results at frame 60 ($t = 3$). Left: vorticity and velocity streamlines for ours, covector fluids, and Gaussian fluids. Right: combined upward gap flux; the dashed line denotes the incompressible value $Q = 4$.

whereas Gaussian fluids exhibits predominantly downward flow. These results support the volume balance maintained by our divergence-free representation; the divergence penalty in Gaussian fluids is insufficient here. Geometry and flux quadrature are detailed in App. C.3.

External forces and smoke. By introducing external forces into the system, our method enables high-fidelity smoke simulations. In the 2D rising-smoke example, a circular puff rises under buoyancy in a closed square domain and develops a mushroom-shaped plume, with free-slip, no-penetration walls (App. C.4). Appendix Fig. 12 compares the normalized total smoke density for our method, covector fluids (Nabizadeh et al., 2022), and Gaussian fluids (Xing et al., 2025). Our method keeps the normalized total near one, with a maximum relative drift below 0.09%, compared with 8.52% for covector fluids and 2.87% for Gaussian fluids, demonstrating more accurate preservation of smoke content in this experiment.

In the smoky sphere example, a localized smoke puff rises under buoyancy toward a stationary sphere in a closed 3D domain, with free-slip, no-penetration boundaries on the sphere and outer walls (App. C.4). Our simulation captures the smoke wrapping around the obstacle and forming a widening cloud above it, with finer folds and billows developing during its ascent (Appendix Fig. 13).

5.3 Validation on Simulation Recovery

Spatial recovery. We recover 100 PFM Taylor-vortex fields (Zhou et al., 2024) from 10% vorticity observations per frame. Independent vorticity-only fits without temporal coupling capture the cores but leave isolated artifacts and an uneven filament. After 30 filtering rounds, ours better matches the reference (Fig. 2; protocol in App. A.6).

In Table 2, missing ω MSE measures vorticity error at unobserved grid cells; velocity MSE uses the full field, and relative velocity MSE divides it by the mean squared PFM velocity component. These grid metrics use PFM cell centers. Mapped L_1 is the mean absolute advective vorticity discrepancy between adjacent frames at uniformly sampled spatial queries. Ours improves all four metrics over the independent fit. Filtering and normalization use additional reference information, so this is reference-assisted recovery rather than reconstruction solely from retained samples (App. A.6).

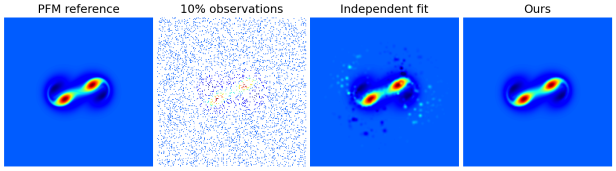


Figure 2: Spatial recovery at frame 100. Additional frames are shown in App. A.6.

Table 2: Spatial recovery errors over frames 100–199.

Metric	Independent fit	Ours
Missing ω MSE	1.907×10^{-1}	$\mathbf{4.088 \times 10^{-3}}$
Mapped L_1	1.481×10^{-1}	$\mathbf{3.649 \times 10^{-3}}$
Velocity MSE	7.287×10^{-3}	$\mathbf{4.053 \times 10^{-4}}$
Relative vel. MSE	4.326×10^{-2}	$\mathbf{2.406 \times 10^{-3}}$

Temporal super-resolution. We reveal the complete vorticity field only at frames 100, 125, 150, 175, and 200, leaving the other 96 PFM frames entirely unobserved. On these unobserved frames, vorticity and velocity MSEs are 3.470×10^{-3} and 3.375×10^{-4}, respectively. App. A.6 provides relative errors, the protocol, and the recovered sequence (Fig. 4).

5.4 Ablation Study: Effect of the Single-Curl Reformulation

We evaluate the effect of the single-curl reformulation in Equation 6 (**ours**) by comparing it with the original double-curl velocity representation of Ni et al. (2025) (**DFK-doublecurl**). The Taylor-vortex experiment evaluates runtime and convergence, and the piston experiment evaluates moving-boundary fitting.

Runtime and convergence. Table 3 compares both formulations on 10 Taylor-vortex time steps. For a fair comparison, we use the same optimization objectives, initial radii, kernel counts, iteration budgets, and device for both formulations, while tuning their learning rates separately. App. D.1 specifies the fitting and timing protocols. Results show our modified formulation of DFK brings a 49% of performance boost in initialization and 42% in time integration with an additional reduction of the converged loss value, as our formulation involves fewer derivative calculations. The loss curves in Appendix Fig. 6 further demonstrate faster convergence of our method, indicating that the single-curl formulation better suits the derivative-sensitive losses.

Boundary fitting. On the piston example, we compare DFK-doublecurl with radius factors $\eta = 9$ and 27, and ours with $\eta = 9$ (Eq. (27)). For a fair comparison, we match the scene, initial kernel count, time step, boundary weights, and evaluation protocol, while tuning learning rates separately for each formulation (App. D.2). Table 4 reports the full-velocity boundary loss $\mathcal{L}_{\mathrm{b1}}$ averaged over the sequence; sampling and averaging are specified in App. D.2.

Table 3: Taylor-vortex ablation.

Configuration	**Init time** (ms/iteration)	**Advance time** (s/time step)	$\mathcal{L}_{\mathrm{init}}$	$\mathcal{L}_{\omega}$
DFK-doublecurl	4.210	41.837	1.309×10^{-4}	4.063×10^{-4}
ours	**2.118**	**24.160**	$\mathbf{5.862 \times 10^{-5}}$	$\mathbf{6.385 \times 10^{-5}}$

Table 4: Piston ablation.

Configuration	$\mathcal{L}_{\mathrm{b1}}$
DFK-doublecurl ($\eta = 9$)	0.147385
DFK-doublecurl ($\eta = 27$)	0.010651
ours	**0.002879**

Under these simulation settings, our single-curl formulation reduces the mean boundary loss by 98.0% and 73.0% relative to DFK-doublecurl ($\eta = 9$) and DFK-doublecurl ($\eta = 27$), respectively, and has lower loss at every evaluated frame. These results also indicate that DFK-doublecurl requires larger radii to satisfy the boundary conditions, at the expense of additional computational time. In contrast, our formulation achieves the lowest boundary loss without scene-specific hyperparameter retuning.

6 Conclusion and Limitations

We presented SimDFKs, a continuous spatial representation for fluid simulation based on divergence-free kernels. The single-curl reformulation reduces evaluation cost, accelerates convergence, and improves boundary fitting over the original double-curl formulation, while maintaining analytically divergence-free velocities without projection or divergence penalties. Our method achieves the lowest velocity error and error growth among the compared methods. Experiments demonstrate vortex preservation, static and moving boundary handling, smoke simulation, and reference-assisted recovery from sparse vorticity observations, while retaining memory efficiency, spatial adaptivity, and continuous differentiability.

The main limitation is computational cost: repeated first-order optimization makes SimDFKs substantially slower than conventional solvers based on explicit spatial discretizations, as with INSR and Gaussian fluids. Higher-order optimizers tailored to DFKs could reduce the iterations needed for convergence. A second limitation is resolving thin layers of intense vorticity near solid boundaries. Smooth kernels can spread these structures over wider bands, producing visible discrepancies from conventional solvers in cases such as the Kármán vortex street. These unresolved boundary structures also make accurate simulation recovery more challenging.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with translation. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, interpreting results, and generating synthetic data sets, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformatting dataset are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, drafting the Related Work section of our paper, summarizing or analyzing existing literature in the Related Work section, editing our paper to improve readability, formatting references and designing the webpage in our supplemental material. We have reviewed all AI-assisted work. All authors have reviewed and refined the paper. We have checked all the figures, images, plots and tables and made sure they are consistent with our experimental results. We have checked all references in our paper (especially in the Related Work section) and made sure their description in the paper and formatting in BibTeX is correct. We have verified that AI-assisted editing has preserved our intended meaning. We have made sure that all videos in our supplemental webpage are generated with our method or the baseline and reference methods mentioned in the paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 describes the DFK representation and the forward-simulation and recovery procedures. Appendix A provides analytic kernel formulas, algorithm pseudocode, and numerical settings for sampling, optimization, boundary handling, and reinitialization. Scene configurations, initial conditions, baseline settings, and evaluation protocols are documented in Appendices B–D. Appendix A.6 specifies the recovery observation protocol, reference information used, and evaluation conventions. We will open-source our code upon acceptance of the paper.

REFERENCES

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

Jeremiah U Brackbill and Hans M Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational physics*, 65(2):314–343, 1986.

Robert Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, Boca Raton, FL, USA, 2nd edition, 2015.

Jumyung Chang, Ruben Partono, Vinicius C. Azevedo, and Christopher Batty. Curl-Flow: Boundary-respecting pointwise incompressible velocity interpolation for grid-based fluids. *ACM Transactions on Graphics*, 41(6):243:1–243:21, 2022. doi: 10.1145/3550454.3555498. URL https://doi.org/10.1145/3550454.3555498.

Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. Implicit neural spatial representations for time-dependent PDEs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 5162–5177. PMLR, 2023. URL https://proceedings.mlr.press/v202/chen23af.html.

Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

Mengyu Chu, Lingjie Liu, Quan Zheng, Aleksandra Franz, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics*, 41(4):119:1–119:14, 2022. doi: 10.1145/3528223.3530169. URL https://doi.org/10.1145/3528223.3530169.

Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. Fluid simulation on neural flow maps. *ACM Trans. Graph.*, 42(6), December 2023. ISSN 0730-0301. doi: 10.1145/3618392. URL https://doi.org/10.1145/3618392.

Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. ScalarFlow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics*, 38(6):239:1–239:16, 2019. doi: 10.1145/3355089.3356545. URL https://doi.org/10.1145/3355089.3356545.

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics*, 26(1):4, 2007. doi: 10.1145/1189762.1189766. URL https://www.geometry.caltech.edu/pubs/ETKSD07.pdf.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pp. 15–22, New York, NY, USA, 2001. Association for Computing Machinery. doi: 10.1145/383259.383260.

Aleksandra Franz, Barbara Solenthaler, and Nils Thuerey. Global transport for fluid reconstruction with learned self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1632–1642, 2021. doi: 10.1109/CVPR46437.2021.00168. URL https://arxiv.org/abs/2104.06031.

Edward J. Fuselier, Varun Shankar, and Grady B. Wright. A high-order radial basis function (RBF) Leray projection method for the solution of the incompressible unsteady Stokes equations. *Computers & Fluids*, 128:41–52, 2016. doi: 10.1016/j.compfluid.2016.01.009. URL https://doi.org/10.1016/j.compfluid.2016.01.009.

James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. From capture to simulation: Connecting forward and inverse problems in fluids. *ACM Transactions on Graphics*, 33(4):139:1–139:11, 2014. doi: 10.1145/2601097.2601147. URL https://doi.org/10.1145/2601097.2601147.

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

Pranav Jain, Ziyin Qu, Peter Yichen Chen, and Oded Stein. Neural monte carlo fluid simulation. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705250. doi: 10.1145/3641519.3657438. URL https://doi.org/10.1145/3641519.3657438.

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

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum*, 38(2):59–70, 2019. doi: 10.1111/cgf.13619. URL https://doi.org/10.1111/cgf.13619.

Doyub Kim, Minjae Lee, and Ken Museth. NeuralVDB: High-resolution sparse volume representation using hierarchical neural networks. *ACM Transactions on Graphics*, 43(2):20:1–20:21, 2024. doi: 10.1145/3641817. URL https://doi.org/10.1145/3641817.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. A survey on sph methods in computer graphics. *Computer Graphics Forum*, 41(2):737–760, 2022. doi: 10.1111/cgf.14508. URL https://doi.org/10.1111/cgf.14508.

Svenja Lowitzsch. Error estimates for matrix-valued radial basis function interpolation. *Journal of Approximation Theory*, 137(2):238–249, 2005. doi: 10.1016/j.jat.2005.09.008. URL https://doi.org/10.1016/j.jat.2005.09.008.

Luan Lyu, Xiaohua Ren, Wei Cao, Jian Zhu, Enhua Wu, and Zhi-Xin Yang. Wavelet potentials: An efficient potential recovery technique for pointwise incompressible fluids. *Computer Graphics Forum*, 43(2):e15023, 2024. doi: 10.1111/cgf.15023. URL https://doi.org/10.1111/cgf.15023.

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. Covector fluids. *ACM Transactions on Graphics (TOG)*, 41(4):113:1–113:15, 2022. URL https://cseweb.ucsd.edu/~viscomp/projects/SIG22CovectorFluids/.

Mohammad Sina Nabizadeh, Ritoban Roy-Chowdhury, Hang Yin, Ravi Ramamoorthi, and Albert Chern. Fluid implicit particles on coadjoint orbits. *ACM Transactions on Graphics*, 43(6):270:1–270:38, 2024. doi: 10.1145/3687970. URL https://doi.org/10.1145/3687970.

Francis J. Narcowich and Joseph D. Ward. Generalized hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comput.*, 63(208):661–687, October 1994.

Xingyu Ni, Jingrui Xing, Xingqiao Li, Bin Wang, and Baoquan Chen. Representing flow fields with divergence-free kernels for reconstruction. *Proc. ACM Comput. Graph. Interact. Tech.*, 8(4), August 2025. doi: 10.1145/3747872. URL https://doi.org/10.1145/3747872.

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

Jiaxiong Qiu, Ruihong Cen, Zhong Li, Han Yan, Ming-Ming Cheng, and Bo Ren. Neusmoke: Efficient smoke reconstruction and view synthesis with neural transportation fields. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400711312. doi: 10.1145/3680528.3687667. URL https://doi.org/10.1145/3680528.3687667.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045. URL https://doi.org/10.1016/j.jcp.2018.10.045.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020. doi: 10.1126/science.aaw4741. URL https://doi.org/10.1126/science.aaw4741.

Jack Richter-Powell, Yaron Lipman, and Ricky T. Q. Chen. Neural conservation laws: A divergence-free perspective. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/f8d39584f87944e5dbe46ec76f19e20a-Abstract-Conference.html.

Ritoban Roy-Chowdhury, Tamar Shinar, and Craig Schroeder. Higher order divergence-free and curl-free interpolation on MAC grids. *Journal of Computational Physics*, 503:112831, 2024. doi: 10.1016/j.jcp.2024.112831. URL https://doi.org/10.1016/j.jcp.2024.112831.

John Stanley Sawyer. A semi-lagrangian method of solving the vorticity advection equation. *Tellus*, 15(4):336–342, 1963. doi: 10.3402/tellusa.v15i4.8862. URL https://doi.org/10.3402/tellusa.v15i4.8862.

Craig Schroeder, Ritoban Roy Chowdhury, and Tamar Shinar. Local divergence-free polynomial interpolation on MAC grids. *Journal of Computational Physics*, 468:111500, 2022. doi: 10.1016/j.jcp.2022.111500. URL https://doi.org/10.1016/j.jcp.2022.111500.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pp. 910–914, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 9781450378253. doi: 10.1145/1186822.1073282. URL https://doi.org/10.1145/1186822.1073282.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2):350–371, 2008.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. Spgrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics (TOG)*, 33 (6):1–12, 2014.

Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible SPH. *ACM Transactions on Graphics*, 28(3):40:1–40:6, 2009. doi: 10.1145/1531326.1531346. URL https://doi.org/10.1145/1531326.1531346.

Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pp. 121–128, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. doi: 10.1145/311535.311548. URL https://doi.org/10.1145/311535.311548.

Ningxiao Tao, Liru Zhang, Xingyu Ni, Mengyu Chu, and Baoquan Chen. FlowCapX: Physics-grounded flow capture with long-term consistency. *Computer Graphics Forum*, 44(7):e70274, 2025. doi: 10.1111/cgf.70274. URL https://doi.org/10.1111/cgf.70274.

Ningxiao Tao, Baoquan Chen, and Mengyu Chu. Lagrangiansplats: Divergence-free transport of gaussian primitives for fluid reconstruction. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, SIGGRAPH Conference Papers '26, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725548. doi: 10.1145/3799902.3811188. URL https://doi.org/10.1145/3799902.3811188.

Haoxiang Wang, Tao Yu, Tianwei Yang, Hui Qiao, and Qionghai Dai. Neural physical simulation with multi-resolution hash grid encoding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(6):5410–5418, 2024a. doi: 10.1609/aaai.v38i6.28349. URL https://doi.org/10.1609/aaai.v38i6.28349.

Haoxiang Wang, Tao Yu, Hui Qiao, and Qionghai Dai. Neural fluid simulation on geometric surfaces. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f58c24798220ba724fe05c0fa786227d-Abstract-Conference.html.

Sinan Wang, Yitong Deng, Molin Deng, Hong-Xing Yu, Junwei Zhou, Duowen Chen, Taku Komura, Jiajun Wu, and Bo Zhu. An eulerian vortex method on flow maps. *ACM Transactions on Graphics (TOG)*, 43(6):1–14, 2024b.

Sinan Wang, Junwei Zhou, Fan Feng, Zhiqi Li, Yuchen Sun, Duowen Chen, Greg Turk, and Bo Zhu. Fluid simulation on vortex particle flow maps. *ACM Trans. Graph.*, 44(4), July 2025b. ISSN 0730-0301. doi: 10.1145/3731198. URL https://doi.org/10.1145/3731198.

Yiming Wang, Siyu Tang, and Mengyu Chu. Physics-informed learning of characteristic trajectories for smoke reconstruction. In *ACM SIGGRAPH 2024 Conference Papers*, 2024c. doi: 10.1145/3641519.3657483. URL https://doi.org/10.1145/3641519.3657483.

Steffen Weißmann and Ulrich Pinkall. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450302104. doi: 10.1145/1833349.1778852. URL https://doi.org/10.1145/1833349.1778852.

Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(1):389–396, 1995. doi: 10.1007/BF02123482. URL https://doi.org/10.1007/BF02123482.

Holger Wendland. Divergence-free kernel methods for approximating the Stokes problem. *SIAM Journal on Numerical Analysis*, 47(4):3158–3179, 2009. doi: 10.1137/080730299. URL https://doi.org/10.1137/080730299.

Jingrui Xing, Bin Wang, Mengyu Chu, and Baoquan Chen. Gaussian fluids: A grid-free fluid solver based on gaussian spatial representation. In *ACM SIGGRAPH 2025 Conference Papers*, SIGGRAPH '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715402. doi: 10.1145/3721238.3730620. URL https://doi.org/10.1145/3721238.3730620.

Hong-Xing Yu, Yang Zheng, Yuan Gao, Yitong Deng, Bo Zhu, and Jiajun Wu. Inferring hybrid neural fluid fields from videos. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG)*, 37(4):1–8, 2018.

Wenran Zhang, Yuxiang Cai, Letian Huang, Dongwei Ye, Jie Guo, and Bo Ren. Gausmoke: Hybrid physics-optical gaussian splatting for sparse smoke reconstruction. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, SIGGRAPH Conference Papers '26, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725548. doi: 10.1145/3799902.3811148. URL https://doi.org/10.1145/3799902.3811148.

Junwei Zhou, Duowen Chen, Molin Deng, Yitong Deng, Yuchen Sun, Sinan Wang, Shiying Xiong, and Bo Zhu. Eulerian-lagrangian fluid simulation on particle flow maps. *ACM Transactions on Graphics*, 43(4):1–20, 2024.

A Method Details

A.1 Kernel Formulation and Analytic Derivatives

The scalar radial function in Eq. (5) is

$$\phi(r) = (1-r)_+^6(35r^2+18r+3), \qquad (z)_+ = \max(z,0), \tag{18}$$

following Wendland (1995). Write $\boldsymbol{y}_i = (\boldsymbol{x}-\boldsymbol{p}_i)/h_i$ and $r_i = \|\boldsymbol{y}_i\|_2$. In two dimensions, we append a zero third component to $\boldsymbol{y}_i$ and extend ϕ_i constantly along x_3; in three dimensions, the distance includes all three coordinates. Let $\boldsymbol{P}_2 = \mathrm{diag}(1,1,0)$ and $\boldsymbol{P}_3 = \boldsymbol{I}$. The radial derivatives are

$$a(r) = \frac{\phi'(r)}{r} = -56(1-r)^5(5r+1), \qquad q(r) = \frac{a'(r)}{r} = 1680(1-r)^4 \tag{19}$$

for $0 \le r < 1$, with values at $r = 0$ interpreted by continuity. The gradient and Hessian of the scaled basis are

$$\nabla\phi_i = h_i a(r_i)\boldsymbol{y}_i, \qquad \boldsymbol{H}_i = \nabla\nabla^\top\phi_i = a(r_i)\boldsymbol{P}_d + q(r_i)\boldsymbol{y}_i\boldsymbol{y}_i^\top. \tag{20}$$

Substituting into the curl formulation gives, in either dimension,

$$\boldsymbol{u}_i = h_i a(r_i)(\boldsymbol{y}_i \times \boldsymbol{a}_i), \qquad \boldsymbol{\omega}_i = \big[\boldsymbol{H}_i - \mathrm{tr}(\boldsymbol{H}_i)\boldsymbol{I}\big]\boldsymbol{a}_i. \tag{21}$$

The factor h_i^2 in Eq. (5) cancels the inverse-square radius factor introduced by the two derivatives in vorticity. In particular, $\boldsymbol{\omega}_i(\boldsymbol{p}_i) = 112\boldsymbol{a}_i$ in three dimensions and under the planar restriction $\boldsymbol{a}_i = s_i\boldsymbol{e}_3$, so the coefficient controls the center amplitude independently of the radius.

The analytic velocity Jacobian is $\nabla\boldsymbol{u}_i = -[\boldsymbol{a}_i]_\times\boldsymbol{H}_i$, where $[\boldsymbol{a}_i]_\times\boldsymbol{b} = \boldsymbol{a}_i \times \boldsymbol{b}$ and $\boldsymbol{H}_i$ is given by Eq. (20). This expression uses the same three-dimensional embedding as Eq. (21). In 2D cases, setting $\boldsymbol{a}_i = s_i\boldsymbol{e}_3$ and writing $\boldsymbol{A}_{\boldsymbol{\theta}} = \psi_{\boldsymbol{\theta}}\boldsymbol{e}_3$ recovers the stream-function expressions

$$\begin{aligned} \psi_{\boldsymbol{\theta}} &= \sum_i s_i\phi_i, \\ \boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{u}_{\mathrm{base}} &= (\partial_{x_2}\psi_{\boldsymbol{\theta}}, -\partial_{x_1}\psi_{\boldsymbol{\theta}}, 0)^\top, \\ \boldsymbol{\omega}_{\boldsymbol{\theta}} &= \big(0, 0, -(\partial_{x_1}^2 + \partial_{x_2}^2)\psi_{\boldsymbol{\theta}}\big)^\top. \end{aligned} \tag{22}$$

For the planar formulas below, we omit the zero velocity component and write $\omega = (\boldsymbol{\omega})_3$. With $\boldsymbol{J} = \left[\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right]$ and planar $\boldsymbol{y}_i$, a single kernel contributes

$$\boldsymbol{u}_i(\boldsymbol{x}) = s_i h_i a(r_i)\boldsymbol{J}\boldsymbol{y}_i, \tag{23}$$

$$\nabla\boldsymbol{u}_i(\boldsymbol{x}) = s_i\left[q(r_i)(\boldsymbol{J}\boldsymbol{y}_i)\boldsymbol{y}_i^\top + a(r_i)\boldsymbol{J}\right], \tag{24}$$

$$\omega_i(\boldsymbol{x}) = s_i c(r_i), \qquad c(r) = 56(1-r)^4(2+8r-40r^2). \tag{25}$$

Here, $c(r) = -\big(\phi''(r) + \phi'(r)/r\big)$. All three contributions are zero for $r_i \geq 1$. We use the Jacobian convention $[\nabla \boldsymbol{u}]_{ab} = \partial_{x_b} u_a$. The trace of Eq. (24) vanishes because $\boldsymbol{y}_i^\top \boldsymbol{J} \boldsymbol{y}_i = 0$ and $\operatorname{tr} \boldsymbol{J} = 0$. Its antisymmetric entries give Eq. (25). At the center, $\boldsymbol{u}_i(\boldsymbol{p}_i) = \boldsymbol{0}$, $\nabla \boldsymbol{u}_i(\boldsymbol{p}_i) = -56 s_i \boldsymbol{J}$, and $\omega_i(\boldsymbol{p}_i) = 112 s_i$. These finite expressions avoid division by r_i in field evaluation.

The analytic fields are differentiated with respect to kernel parameters to accumulate gradients for optimization. For example, the vorticity derivatives are

$$\frac{\partial \omega_i}{\partial s_i} = c(r_i), \qquad \nabla_{\boldsymbol{p}_i} \omega_i = -\frac{s_i}{h_i} \frac{c'(r_i)}{r_i} \boldsymbol{y}_i, \qquad \frac{\partial \omega_i}{\partial h_i} = -\frac{s_i r_i}{h_i} c'(r_i), \tag{26}$$

where $c'(r)/r = 1680(1-r)^3(8r-4)$ within the support. Velocity and Jacobian losses similarly differentiate Eq. (23) and Eq. (24). We use these analytic gradients to optimize the kernel centers, radii, and strengths with Adam.

A.2 Sampling, Initialization, and Optimization

In the componentwise mean absolute discrepancy $\mathcal{D}_{\mathcal{Q}}$ (Sec. 4.2), m counts physical scalar components: d for velocity, d^2 for its Jacobian, and 1 or 3 for vorticity in two or three dimensions, respectively. Zero components introduced solely by the planar embedding are omitted. For matrices, the norm sums the absolute values of all entries.

We use the same sampling and optimization procedure in 2D and 3D cases, with spatial queries drawn uniformly from the computational box. In 2D cases, spatial coordinates and velocities are rescaled by the same factor so that the shorter side of the initial rectangular domain has length 10; this leaves time, velocity gradients, and vorticity unchanged. Reinitialization samples a padded box in 2D and the computational box in 3D. In scenes with immersed solids, the continuous field extends into solid interiors and solid surfaces contribute separate boundary samples. Scene domains and initial conditions are specified in Appendices B and C.

We initialize centers on a regular lattice with three padding layers. Both 3D examples start from a 40^3 lattice, giving $N = 46^3$ kernels after padding. For N centers and padded-box volume V_{pad} (area when $d = 2$), the common initial radius is

$$h_0 = \eta \left(\frac{V_{\mathrm{pad}}}{N \kappa_d} \right)^{1/d}, \qquad \kappa_2 = \pi, \quad \kappa_3 = \frac{4\pi}{3}, \tag{27}$$

where κ_d is the volume of the unit ball. We use $\eta = 9$ for the 2D forward simulations and Taylor-vortex recovery, and $\eta = 15$ for both 3D examples, leapfrogging vortex rings and smoky sphere. Before each fitting iteration, we enforce the radius lower bound $h_{\min} = \chi h_0$, with $\chi = 0.5$ in 2D and $\chi = 0.042843$ in both 3D examples. A prescribed uniform background flow remains fixed during optimization. Recovery uses the same lattice in every frame and the kernel field alone, with $\boldsymbol{u}_{\mathrm{base}} = \boldsymbol{0}$.

Each fitting iteration draws fresh spatial queries, with a batch size of N in 2D cases and 8192 in 3D cases; recovery uses N samples per frame for each filtering refit (Appendix A.6). During time integration, traced targets are resampled with the queries and held fixed with respect to differentiation. The optimizer uses separate learning rates for centers, radii, and strengths, as listed in Table 5. Both 3D examples use the same learning rates and iteration budgets. We reduce the learning rates after 100 iterations without improvement in the objective, by a factor of 0.9 for initialization/refitting or 0.8 for time integration. Each fit uses a fixed iteration budget.

For the 2D forward simulations, the time step is 0.001 for Taylor–Green, 0.025 for leapfrogging vortices, 0.05 for the Kármán and piston scenes, and 0.01 for the Taylor vortex, rotating cross, and smoke scenes. Both 3D examples use $\Delta t = 0.02$.

The forward losses in the main text are written before amplitude normalization. At each time step, we compute a scalar normalization factor on a regular diagnostic grid $\mathcal{G}$:

$$S = \left[\frac{1}{|\mathcal{G}|} \sum_{\boldsymbol{x} \in \mathcal{G}} \left\| \boldsymbol{J}_n(\boldsymbol{x})^\top \widetilde{\boldsymbol{u}}^n\big(\mathcal{B}_n(\boldsymbol{x})\big) + \beta_d \Delta t \, \boldsymbol{g}^n(\boldsymbol{x}) \right\|_2^2 \right]^{1/2}, \tag{28}$$

where $\widetilde{\boldsymbol{u}}^n$ uses the scene's exterior velocity at departure points outside the region specified in Appendix A.3. The normalization uses $\beta_2 = 1$ and $\beta_3 = 0$: the force contributes to this numerical

Table 5: Adam settings for forward simulation and Taylor-vortex recovery. Reinitialization performs two fits with the listed per-fit budget. Recovery sampling and filtering settings are given in Appendix A.6; ablation learning rates are given in Appendix D.

Stage	Dimension	Centers	Radii	Strengths	Iterations
Initialization	2D	10^{-3}	10^{-3}	5×10^{-6}	10,000
	3D	10^{-3}	3×10^{-3}	5×10^{-4}	10,000
Time integration	2D	10^{-3}	10^{-2}	10^{-6}	5,000
	3D	5×10^{-5}	10^{-4}	5×10^{-5}	5,000
Reinitialization	2D	10^{-3}	10^{-3}	5×10^{-6}	10,000
	3D	10^{-3}	3×10^{-3}	5×10^{-4}	10,000
Recovery refit	2D	10^{-4}	10^{-4}	5×10^{-7}	3,000

scale only in 2D cases, while its curl contributes to the vorticity target in both dimensions. Kernel strengths, the fixed background velocity, and the vorticity and boundary targets are divided by this same S during fitting; the physical amplitude is restored afterward. Reinitialization uses a scale computed from the old field: componentwise root-mean-square velocity in 2D cases and root-mean-square speed in 3D cases. Recovery refit normalization is detailed in Appendix A.6. Spatial parameters and characteristic tracing remain in physical units.

In both 3D examples, $\mathcal{G}$ is a 128^3 grid over the full computational box for time integration and reinitialization. Exterior departure points contribute zero to the time-step scale, which is bounded below by 10^{-12}.

After ordinary initialization, the Kármán, piston, and rotating-cross scenes perform an additional fit using initial vorticity and boundary data. This uses Eq. (15) at $\Delta t = 0$ for 10,000 iterations, with $\lambda_1 = \lambda_2 = 20$ for Kármán and piston, and $\lambda_2 = 20$ with $\mathcal{L}_{\mathrm{b1}}$ omitted for rotating cross. Its learning rates for centers, radii, and strengths are 10^{-4}, 10^{-3}, and 10^{-5}, respectively.

A.3 Characteristic Tracing, Extrapolation, and Boundaries

For a frozen velocity field $\boldsymbol{v}$, denote its tracing map by $\mathcal{F}^{\tau}_{\boldsymbol{v}}(\boldsymbol{x})$, obtained by integrating

$$\frac{d\boldsymbol{X}}{d\tau} = \boldsymbol{v}(\boldsymbol{X}), \qquad \boldsymbol{X}(0) = \boldsymbol{x}. \tag{29}$$

We use classical RK4, evaluating the continuous kernel field at every stage. The backward map for forward simulation is $\mathcal{B}_n = \mathcal{F}^{-\Delta t}_{\boldsymbol{u}^n}$. Its spatial Jacobian, used in vorticity transport for 3D cases and in amplitude normalization, is computed with the same RK4 stages. Writing $\boldsymbol{F}(\tau) = D_{\boldsymbol{x}}\boldsymbol{X}(\tau)$ for the Jacobian of the traced position with respect to its starting point gives

$$\frac{d\boldsymbol{F}}{d\tau} = \nabla \boldsymbol{v}(\boldsymbol{X})\boldsymbol{F}, \qquad \boldsymbol{F}(0) = \boldsymbol{I}. \tag{30}$$

The velocity is frozen within the step, and forces are added explicitly. RK4 therefore describes the characteristic tracer, without implying fourth-order temporal accuracy of the complete simulation scheme.

Extrapolation stages. Let $\widetilde{\boldsymbol{u}}^n$ equal the previous DFK velocity inside the source frame's computational box and a prescribed exterior function $\boldsymbol{u}_{\mathrm{ext}}$ outside. Exterior Jacobian and vorticity values are given by the analytic gradient and curl of $\boldsymbol{u}_{\mathrm{ext}}$, respectively. Table 6 lists the functions for every case. Taylor–Green uses its analytic periodic continuation, the Kármán scene uses the uniform incoming flow, and the remaining cases use zero exterior values.

During time integration, extrapolation is applied at the departure point after RK4 tracing, using the exterior vorticity in Eq. (10). Both 3D examples use $\boldsymbol{u}_{\mathrm{ext}} = \boldsymbol{0}$ outside $[0,1]^3$, giving zero transported targets at exterior departure points. The same exterior region determines the velocity used in Eq. (28). The immersed obstacle in smoky sphere is handled through boundary fitting and zero concentration inside the solid (Appendix C.4). The RK4 stages evaluate the DFK field directly, including at intermediate exterior positions. Buoyancy is added at the receiving query point after this transport extrapolation.

During reinitialization, both velocity-and-Jacobian fits use the old DFK field at interior samples and $\boldsymbol{u}_{\mathrm{ext}}$, $\nabla \boldsymbol{u}_{\mathrm{ext}}$ at exterior samples. The fitting region is the padded box in 2D cases and the computational box in 3D cases, so exterior fitting targets arise only in 2D cases. Ordinary initialization

Table 6: Exterior velocity fields for the simulation scenes. Formulas are given in the original scene coordinates; planar coordinate and velocity rescaling follows Appendix A.2.

d	Case	$\boldsymbol{u}_{\mathrm{ext}}$	$\nabla \boldsymbol{u}_{\mathrm{ext}}$
2	Taylor–Green vortex	$\boldsymbol{u}_{\mathrm{TG}}$	$\nabla \boldsymbol{u}_{\mathrm{TG}}$
2	Taylor vortex	$\mathbf{0}$	$\mathbf{0}$
2	Leapfrogging vortices	$\mathbf{0}$	$\mathbf{0}$
2	Kármán vortex street	$(0.5, 0)^\top$	$\mathbf{0}$
2	Piston	$\mathbf{0}$	$\mathbf{0}$
2	Rising smoke	$\mathbf{0}$	$\mathbf{0}$
2	Rotating cross	$\mathbf{0}$	$\mathbf{0}$
3	Leapfrogging vortex rings	$\mathbf{0}$	$\mathbf{0}$
3	Smoky sphere	$\mathbf{0}$	$\mathbf{0}$

directly evaluates the supplied analytic initial field at its samples and does not use this exterior replacement. Smoke-density advection uses the scalar-grid clamping rule in Appendix A.4.

Here $\boldsymbol{u}_{\mathrm{TG}}$ is the analytic Taylor–Green field defined in Eq. (36). All other exterior functions have zero vorticity.

Boundary sampling. Boundary locations, normals, and prescribed velocities are sampled anew during fitting. The nominal boundary sample count per iteration is 512 in 2D cases and 1024 in 3D cases; sampling multiple surfaces separately can increase the total count. Moving-boundary samples use the obstacle geometry at t_{n+1}. Full velocity is prescribed on the Kármán obstacle and piston surfaces; the rotating-cross prescribes rigid-body normal velocity, and both 3D examples prescribe zero normal velocity. Box boundaries generally use normal-velocity constraints. These are penalty constraints on the velocity representation, including for the no-slip cases; the method does not resolve a viscous boundary layer.

Boundary-loss weights. In Eq. (15), λ_1 weights the full-velocity penalty $\mathcal{L}_{\mathrm{b1}}$, and λ_2 weights the normal-velocity penalty $\mathcal{L}_{\mathrm{b2}}$. Time integration uses $\lambda_1 = \lambda_2 = 1$ for the applicable terms, except in the rotating-cross scene, which uses $\lambda_2 = 5$ with $\mathcal{L}_{\mathrm{b1}}$ omitted. Absent boundary terms are omitted in all scenes. The additional initial boundary fits use the weights specified in Appendix A.2.

A.4 Smoke and External Forces

For smoke, we store concentration, denoted c_{s} to distinguish it from the constant fluid density ρ, on a separate regular grid. In the scenes' coordinate convention, buoyancy acts along the negative x_2 direction. Using the same three-dimensional embedding as the velocity field gives

$$\boldsymbol{g}^n(\boldsymbol{x}) = -\alpha c_{\mathrm{s}}^n(\boldsymbol{x}) \boldsymbol{e}_2, \qquad \nabla \times \boldsymbol{g}^n = \alpha \big(\partial_{x_3} c_{\mathrm{s}}^n,\, 0,\, -\partial_{x_1} c_{\mathrm{s}}^n\big)^\top, \tag{31}$$

with $\alpha = 50$ for both smoke examples and $\partial_{x_3} c_{\mathrm{s}} = 0$ in 2D. Concentration is interpolated bilinearly in 2D and trilinearly in 3D. Its spatial derivatives use interpolated grid finite differences in 2D and derivatives of the trilinear interpolant in 3D.

Initialization seeds the concentration once and fits the initial buoyancy response. In the 2D smoke case, the targets in Eq. (8) include $\Delta t\, \boldsymbol{g}^0$ and $\Delta t\, \nabla \boldsymbol{g}^0$, with fitting amplitudes normalized by the root-mean-square magnitude of $\Delta t\, \boldsymbol{g}^0$. In the 3D smoke case, we fit the vorticity target $\Delta t\, \nabla \times \boldsymbol{g}^0$ together with zero normal velocity on the box and obstacle surfaces, using Eq. (15) with $\lambda_2 = 1$ and the other boundary term absent. This fits the buoyancy response in the divergence-free representation; the force itself need not be divergence-free. The startup normalization scale is $\max(|\alpha \Delta t|, 10^{-12})$.

During each subsequent step, the current concentration supplies the source term in Eq. (11). After fitting the new velocity, the concentration grid is updated by semi-Lagrangian advection (Stam, 1999), using backward RK4 tracing and multilinear interpolation:

$$c_{\mathrm{s}}^{n+1}(\boldsymbol{x}_j) = \mathcal{I}[c_{\mathrm{s}}^n]\big(\mathcal{F}_{\boldsymbol{u}^{n+1}}^{-\Delta t}(\boldsymbol{x}_j)\big)\,, \tag{32}$$

where $\mathcal{I}$ is the corresponding multilinear interpolant and $\boldsymbol{x}_j$ is a grid node. Scalar sampling clamps query coordinates to the grid extent. The seeded smoke is advected without source reinjection. The velocity and vorticity remain kernel fields throughout; the regular grid stores the transported scalar.

The initial concentration profiles and obstacle geometry for the smoke experiments are specified in Appendix C.4.

A.5 Reinitialization and Local Kernel Retention

Reinitialization takes place before the transport fit at time steps $50, 100, \ldots$. It is disabled for the Taylor–Green experiment and during the first eight steps of smoky sphere, which then follows the same schedule. We fit a fresh padded lattice to the current field's velocity and full Jacobian, preserving the fixed background velocity and using the fitting regions and extrapolation rules in Appendix A.3. The current field remains the reference throughout reinitialization.

After this first fit, let $\mathcal{I}_\Omega$ index old kernel centers inside the case's simulation domain. At each such center, we compute the componentwise Jacobian error

$$\begin{aligned} e_i &= \frac{1}{d^2} \sum_{a,b=1}^{d} \left| \left[\nabla \boldsymbol{u}_{\mathrm{new}}(\boldsymbol{p}_i^{\mathrm{old}}) - \nabla \boldsymbol{u}_{\mathrm{old}}(\boldsymbol{p}_i^{\mathrm{old}}) \right]_{ab} \right|, \\ \overline{e} &= \frac{1}{|\mathcal{I}_\Omega|} \sum_{i \in \mathcal{I}_\Omega} e_i. \end{aligned} \tag{33}$$

For every $i \in \mathcal{I}_\Omega$ with $e_i > 2\overline{e}$, we insert a kernel with the selected center and radius into the fresh set. The inserted strength retains its old scalar value in 2D cases and starts from a zero vector in 3D cases. We then fit the combined representation to the same reference velocity and Jacobian, using Eq. (8) for both fits. Refitting settings are given in Table 5. This procedure can change the kernel count, and it approximates the old state through fitting; it does not impose exact conservation of energy or circulation.

A.6 Sequence Recovery and Observation Protocol

Sparse observations. The Taylor-vortex recovery experiment uses $K = 100$ consecutive fields produced by a particle flow maps (PFM) solver (Zhou et al., 2024), beginning at frame 100, with frame interval $\Delta t = 0.01$ and a 200×200 observation grid on $[-5, 5]^2$. We independently sample a blue-noise-style observation mask for each frame, retaining a fraction $p = 0.1$ of the grid locations with random seed 0. A local grid-neighbor check rejects continuous candidate locations within one pixel of nearby accepted candidates; accepted locations are assigned to the containing grid cells. Sampling stops after selecting $\lfloor pHW \rfloor$ distinct cells or considering $500\lfloor pHW \rfloor$ candidates, where $H \times W$ is the grid resolution.

For a selected grid cell (i, j), the observation is associated with the center coordinate

$$\boldsymbol{x}_{ij} = \left(x_{\min} + \frac{j + 1/2}{W}(x_{\max} - x_{\min}),\ y_{\min} + \frac{i + 1/2}{H}(y_{\max} - y_{\min}) \right). \tag{34}$$

The PFM reference values and observed samples use the cell centers in Eq. (34). Our reported checkpoints were generated with endpoint-inclusive query coordinates, $j/(W-1)$ and $i/(H-1)$, before a subsequent cell-center correction to the filtering code. We evaluate those checkpoints at the PFM cell centers. This half-cell offset in the filtering targets is a limitation of the reported run. Full reference vorticity and velocity fields provide evaluation targets; the filtering stage uses additional reference information as specified below.

Advective filtering and refitting. All framewise kernel strengths are initially set to zero. For round r, let $\boldsymbol{u}^{n,(r)}$ and $\omega^{n,(r)}$ denote the physical fields at its start. With a half-window $R = 5$, set $\mathcal{N}_R(n) = \{j : 0 \le j < K,\ 1 \le |j - n| \le R\}$. To transport a query $\boldsymbol{x}$ from frame n to $j = n + \sigma d$, where $\sigma \in \{-1, 1\}$ and $d \ge 1$, we recursively apply one signed RK4 step through each intervening field:

$$\Psi^{(r)}_{n \to n}(\boldsymbol{x}) = \boldsymbol{x}, \qquad \Psi^{(r)}_{n \to n+\sigma d} = \mathcal{F}^{\sigma \Delta t}_{\boldsymbol{u}^{n+\sigma(d-1),(r)}} \circ \Psi^{(r)}_{n \to n+\sigma(d-1)}. \tag{35}$$

Here each $\mathcal{F}$ is a frozen-field RK4 step as in Eq. (29). On the filtering query grid $\mathcal{G}$, we evaluate the model-based average in Eq. (16). At a query whose traced location falls in an observed cell of at least one neighbor, we replace this average by the mean of those neighbors' reference vorticities bilinearly interpolated at the traced locations. The mask test uses the nearest cell, whereas bilinear interpolation reads the *full* PFM vorticity grid, including nearby unobserved cells. In the first round, these observation-conditioned replacements supply the initial nonzero targets and therefore provide information beyond the selected sparse samples.

All targets are computed before any frame is refitted, giving a simultaneous filtering round. Each frame is then fitted to its fixed target with the full-domain objective in Eq. (17), starting from its

existing kernels. Each refit uses 3000 Adam iterations, N grid samples drawn uniformly with replacement from all HW filtering-grid points per iteration, and learning rates 10^{-4}, 10^{-4}, and 5×10^{-7} for centers, radii, and strengths, with radius lower bound $0.5h_0$. Refits are normalized by the root-mean-square vorticity of the *full* PFM reference sequence, including unobserved locations, and physical strengths are restored afterward. The experiment performs 30 rounds; we report the result after round 30 without a further anchoring round. The full-reference normalization and interpolation mean that this filtered result is not a strictly sparse-observation-only reconstruction.

Sparse-in-time observations. We separately evaluate temporal sparsity using $K = 101$ PFM fields over frames 100–200. The observation mask contains the complete 200×200 vorticity field at frames 100, 125, 150, 175, and 200, and is empty at the other 96 frames. We use the same initialization, $R = 5$, the observation-conditioned replacement in Eq. (16), 3000 refitting iterations per frame and round, and the same refit learning rates, and run 50 simultaneous rounds. The current frame is not overwritten directly by its observations; anchor information enters through traced neighboring contributions. For this experiment, the refit scale is the root-mean-square vorticity over the observed anchor fields only. Hidden-frame reference vorticity and velocity are used only for offline evaluation. Filtering uses the same endpoint-inclusive query coordinates as the reported spatially sparse run, while all metrics are evaluated at the PFM cell centers.

We evaluate vorticity mean-square error over unobserved grid cells and full-field velocity mean-square error at the same cell centers. Relative velocity MSE divides the velocity MSE by the mean squared PFM velocity component. We also measure adjacent-frame advective vorticity discrepancy (mapped L_1) at 40,000 uniformly drawn spatial queries using the physical fields.

For temporal super-resolution, errors are evaluated over the 96 unobserved frames. Vorticity MSE is 3.470×10^{-3} (relative MSE 1.244×10^{-3}), and velocity MSE is 3.375×10^{-4} (relative MSE 2.003×10^{-3}).

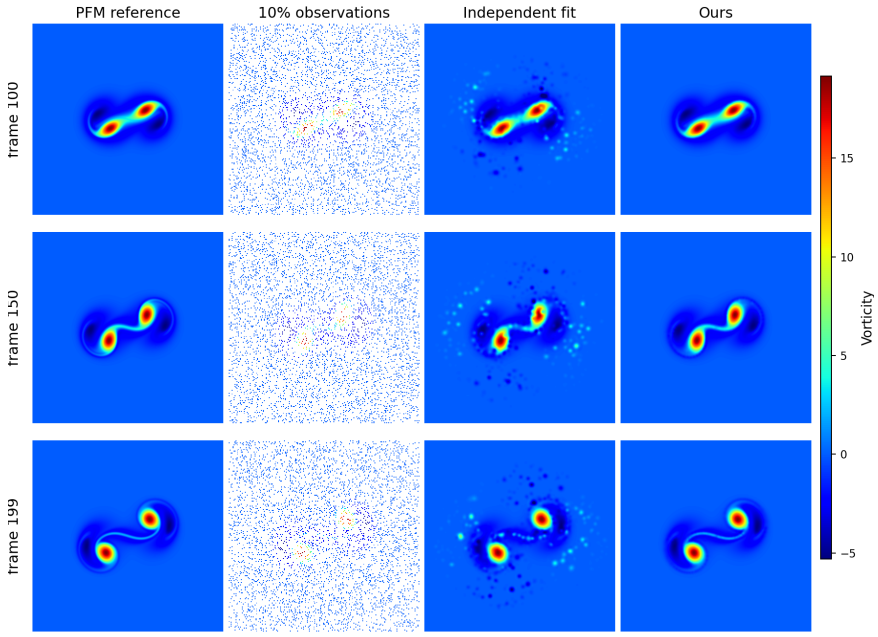


Figure 3: Taylor-vortex vorticity recovery from the PFM reference sequence at frames 100, 150, and 199 (top to bottom). Columns show the reference field, the retained 10% observations, the independent fit, and ours after 30 filtering rounds. All panels share the color scale at right.

A.7 Compact-Support Evaluation

Compact support restricts the field at a query point to kernels satisfying $r_i < 1$. To identify these kernels efficiently, we group centers spatially using bins of width at least $h_{\text{hash}} = 1.25h_0$. For kernels with radius at most h_{hash}, the 3^d neighboring bins contain all possible contributors. Kernels excluded from these bins, including those with larger radii, are checked directly. Every candidate must satisfy $r_i < 1$ before contributing to the field. The spatial groups are updated as kernel pa-

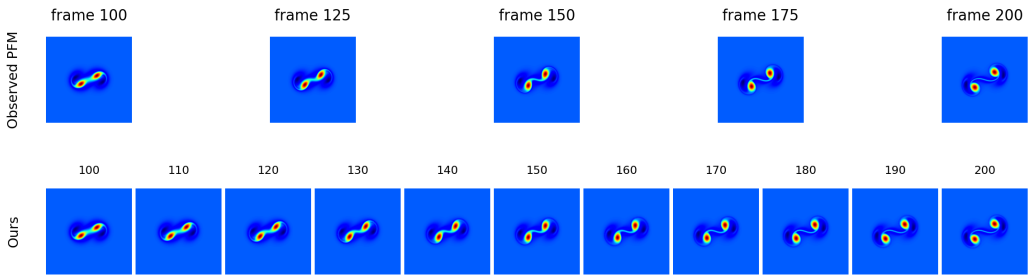


Figure 4: Temporal super-resolution of Taylor-vortex vorticity. Top: the five observed PFM fields at frames 100, 125, 150, 175, and 200. Bottom: recovered fields at frames 100, 110, . . . , 200.

rameters change; they accelerate neighborhood queries without discretizing the continuous velocity representation.

A.8 Algorithm Summaries

Algorithms 1 and 2 summarize the order of the components described in Sections 4.2 and 4.3, respectively.

Algorithm 1 Forward simulation

Fit $\boldsymbol{\theta}^0$ to the initial data ▷ Section 4.2.1
for each time step n **do**
  $\boldsymbol{\theta}^\star \leftarrow \boldsymbol{\theta}^n$
  if reinitialization is due **then**
    Refit $\boldsymbol{\theta}^n$ with fresh kernels to obtain $\boldsymbol{\theta}^\star$ ▷ Section 4.2.3
  end if
  Fit $\boldsymbol{\theta}^{n+1}$ to transport and boundary targets, starting from $\boldsymbol{\theta}^\star$
    ▷ Section 4.2.2; force $\boldsymbol{g}^n$ enters the transport target
  if smoke is present **then**
    Advect c_{s}^n with $\boldsymbol{u}_{\boldsymbol{\theta}^{n+1}}$ to obtain c_{s}^{n+1} ▷ Appendix A.4
  end if
end for

Algorithm 2 Simulation recovery

Initialize K framewise kernel fields and retain physical units
for each filtering round **do**
  Trace grid queries to neighboring frames using Eq. (35)
  Form model averages and observation-conditioned reference targets as in Eq. (16)
  Refit each field to its fixed target, then restore physical strengths
end for

B Taylor–Green Quantitative Evaluation

B.1 Flow Definition and Error Metric

The experiment uses the planar domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ and the analytic fields

$$\boldsymbol{u}_{\mathrm{TG}}(\boldsymbol{x}) = \begin{bmatrix} \sin x_1 \cos x_2 \\ -\cos x_1 \sin x_2 \end{bmatrix}, \qquad \omega_{\mathrm{TG}}(\boldsymbol{x}) = \partial_{x_1} u_2 - \partial_{x_2} u_1 = 2 \sin x_1 \sin x_2. \tag{36}$$

With no external forcing, this is a steady solution of the incompressible Euler equations, with $p/\rho = (\cos 2x_1 + \cos 2x_2)/4$ up to a constant. Figure 5 visualizes its velocity and vorticity. All methods use 100 time steps of $\Delta t = 0.001$, so the evaluated frames 0, 50, and 100 correspond to physical times 0, 0.05, and 0.1. Our method fits the initial velocity and its Jacobian before evolving the kernels.

For a method's $N_q = 3{,}600$ evaluation locations $\{\boldsymbol{x}_j\}$, we compute the componentwise velocity error

$$\mathrm{MSE}_n = \frac{1}{2N_q} \sum_{j=1}^{N_q} \|\boldsymbol{u}^n(\boldsymbol{x}_j) - \boldsymbol{u}_{\mathrm{TG}}(\boldsymbol{x}_j)\|_2^2, \qquad n \in \{0, 50, 100\}. \tag{37}$$

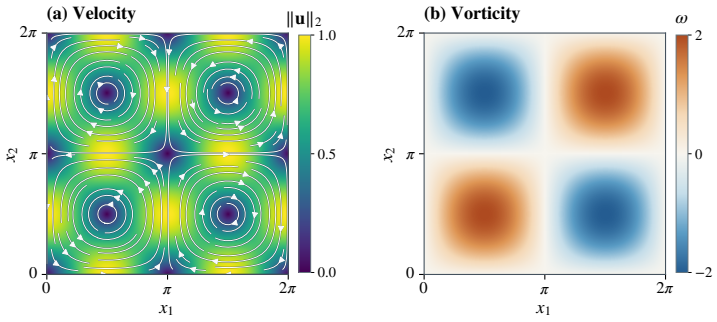


Figure 5: Analytic Taylor–Green reference fields on $[0, 2\pi]^2$. **(a)** Streamlines and arrows show the velocity direction, over a background colored by speed $\|\boldsymbol{u}_{\mathrm{TG}}\|_2$. **(b)** Signed scalar vorticity ω_{TG}, with a diverging color scale centered at zero. These fields are stationary and provide the reference for all evaluated frames.

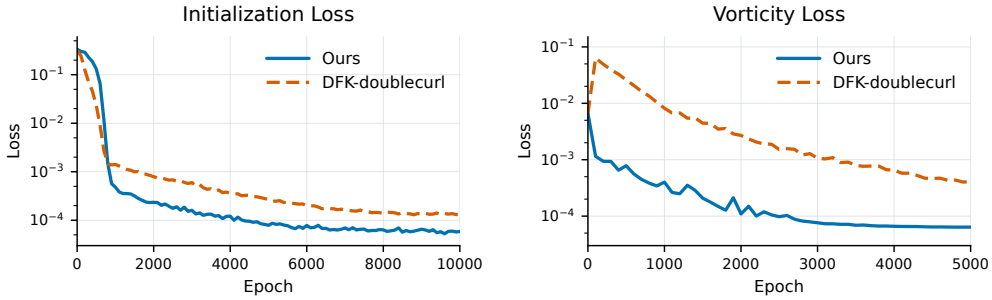


Figure 6: Loss curves of our method and DFK-doublecurl during initialization (left) and time integration (right) for the Taylor-vortex ablation in Appendix D.1.

Eulerian, GSR, and our method use a 60×60 grid with coordinates $x_{1,i} = (i + 1/2)2\pi/60$ and $x_{2,j} = (j + 1/2)2\pi/60$, for $i, j = 0, \ldots, 59$. Eulerian evaluation excludes the outer two cells on each side of the 64×64 grid and uses the two planar velocity components at the remaining locations. INSR and NMC use their respective evaluation locations, which need not coincide with this grid. For INSR's normalized coordinates and velocities $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{u}})$, the physical quantities are $\boldsymbol{x} = \pi(\widehat{\boldsymbol{x}} + \mathbf{1})$ and $\boldsymbol{u} = \pi\widehat{\boldsymbol{u}}$; NMC samples are already in physical coordinates. Each prediction is compared with the analytic field at its own location, without interpolation between sampling sets. All reported MSEs use single-precision field evaluations and average over both velocity components.

B.2 Representations and Parameter Counts

We count the scalar parameters of one velocity field, excluding auxiliary quantities used for optimization, time integration, and field evaluation. The INSR and NMC network architectures follow their Taylor–Green configurations (Chen et al., 2023; Jain et al., 2024). Both use SIREN MLPs with two spatial inputs, sine activations $\sin(30z)$ in the hidden layers, a linear output layer, and biases in every linear layer.

INSR's velocity network has four hidden layers of width 32 and two outputs, giving

$$(2 + 1)32 + 3(32 + 1)32 + (32 + 1)2 = 3{,}330$$

parameters. Its separate pressure network has the same hidden architecture and one output, adding 3,297 parameters; the combined count is 6,627. NMC's velocity network has seven hidden layers of width 64 and two outputs, giving

$$(2 + 1)64 + 6(64 + 1)64 + (64 + 1)2 = 25{,}282$$

parameters.

GSR uses 576 Gaussian particles, each with two position components, two scales, one rotation angle, and two velocity coefficients, for $576 \times 7 = 4{,}032$ parameters. Our method uses 900 DFK particles, including the padding kernels, each with two position components, one radius, and one scalar strength, for $900 \times 4 = 3{,}600$ optimizable parameters. The table additionally counts the two fixed components of $\boldsymbol{u}_{\mathrm{base}} = \mathbf{0}$, giving 3,602. These particle counts are unchanged at the three evaluated frames. For the 64×64 Eulerian MAC grid, the number of face-centered velocity values

is $(64+1)64+64(64+1)=8{,}320$; its auxiliary pressure grid is excluded from the velocity count, as is INSR's pressure network.

B.3 Computational Settings and Timing

INSR, NMC, GSR, and our method are evaluated on an NVIDIA RTX 4090 GPU. Our time-step fits use $5{,}000$ Adam iterations, with learning rates 10^{-3}, 10^{-2}, and 10^{-6} for centers, radii, and strengths, respectively. Optimization over frames 1–100 takes an average of 23.316 seconds per frame, reported as 23 seconds in Table 1; initialization, visualization, and data output are excluded. The remaining GPU timings are the measured average per-frame times of the corresponding methods.

The Eulerian baseline uses MantaFlow 0.13 with second-order MacCormack advection, pressure projection with conjugate-gradient tolerance 10^{-3}, and a one-cell open boundary on each side of the grid. Its time step is 0.001. We measure its simulation time on an AMD Ryzen 9 7950X CPU using one thread. After an untimed 100-step warmup, we average 15 repetitions of 100 steps, excluding initialization, visualization, vorticity postprocessing, and data output. Each repetition starts from the same initial velocity field. The resulting mean is 0.0015457 seconds per frame, reported as 0.00155.

C Validation Experiment Settings

This appendix specifies the domains, initial conditions, and obstacle geometry for the examples in Section 5.2. Coordinates and velocities below are given before the planar rescaling described in Appendix A.2. Shared fitting, transport, and reinitialization settings are given in Appendix A. The 3D examples are leapfrogging vortex rings and smoky sphere, both on $\Omega=[0,1]^3$.

C.1 Vortex Preservation

Taylor vortex. The comparison in Fig. 7 starts from two identical vortices rotating in the same direction on $\Omega=[-5,5]^2$, centered at $\boldsymbol{c}_1=(-0.8,0)^\top$ and $\boldsymbol{c}_2=(0.8,0)^\top$. Their initial velocity is

$$\boldsymbol{u}^0(\boldsymbol{x})=\sum_{j=1}^{2}\frac{U}{a}\exp\left[\frac{1}{2}\left(1-\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|_2^2}{a^2}\right)\right]\mathcal{R}(\boldsymbol{x}-\boldsymbol{c}_j),\qquad U=3,\quad a=0.5,\tag{38}$$

where $\mathcal{R}(\boldsymbol{\delta})=(-\delta_2,\delta_1)^\top$, and U is the peak azimuthal speed of each isolated vortex, attained at radius a. The initial fit uses this velocity and its analytic Jacobian. The Taylor-vortex ablation and the PFM reference sequence used for recovery start from the same analytic initial condition.

Our forward simulation initializes kernel centers on a 71×71 lattice with three padding layers, giving $N=77^2=5929$ kernels. We use the radial basis in Eq. (18) and the radius factor $\eta=9$ in Eq. (27), giving an initial radius $h_0\approx0.715966$. Initial strengths and the fixed background velocity are zero. Time integration uses $\Delta t=0.01$.

The semi-implicit Eulerian baseline uses a 512×512 MAC grid. Counting the two face-centered velocity components in single precision gives $2\times512\times513=525{,}312$ scalar values, requiring $2{,}101{,}248$ bytes (approximately 2 MB) per velocity field; pressure and auxiliary solver state are excluded. The reference vorticity field contains 512×512 double-precision samples, requiring 2 MB.

The INSR comparison reuses the Taylor-vortex sequence from Xing et al. (2025). Its average spatial-representation file size is 32.1 KB, as reported in that experiment. Our representation files average 103.9 KB across all 409 saved states. These file sizes include the overhead of storing the representation. We use 1 KB $=1024$ bytes and 1 MB $=1024^2$ bytes.

Leapfrogging vortices in 2D. The domain is $\Omega=[-5,5]^2$. Four vortices have centers $\boldsymbol{c}_j=(b_j,-3)^\top$, with $(b_1,b_2,b_3,b_4)=(-3,-1,1,3)$, signed strengths $(\gamma_1,\gamma_2,\gamma_3,\gamma_4)=(0.5,0.5,-0.5,-0.5)$, and common core radius $a=0.3$. Their initial velocity is

$$\boldsymbol{u}^0(\boldsymbol{x})=\sum_{j=1}^{4}\gamma_j\frac{1-\exp\left[-((r_j+\varepsilon)/a)^2\right]}{(r_j+\varepsilon)^2}\mathcal{R}(\boldsymbol{x}-\boldsymbol{c}_j),\qquad r_j=\|\boldsymbol{x}-\boldsymbol{c}_j\|_2,\quad\varepsilon=10^{-6}.\tag{39}$$

Kernel centers start on a 71×71 lattice with three padding layers, giving $N=77^2=5929$ kernels, with $\eta=9$ and $\Delta t=0.025$. The sequence in Fig. 8 spans frames 0–1599, corresponding to $t\in[0,39.975]$. Vorticity is sampled on an 801×801 grid over the full domain. All displayed frames share the fixed linear color range $[-11.11022,11.11802]$, determined from frame 0.

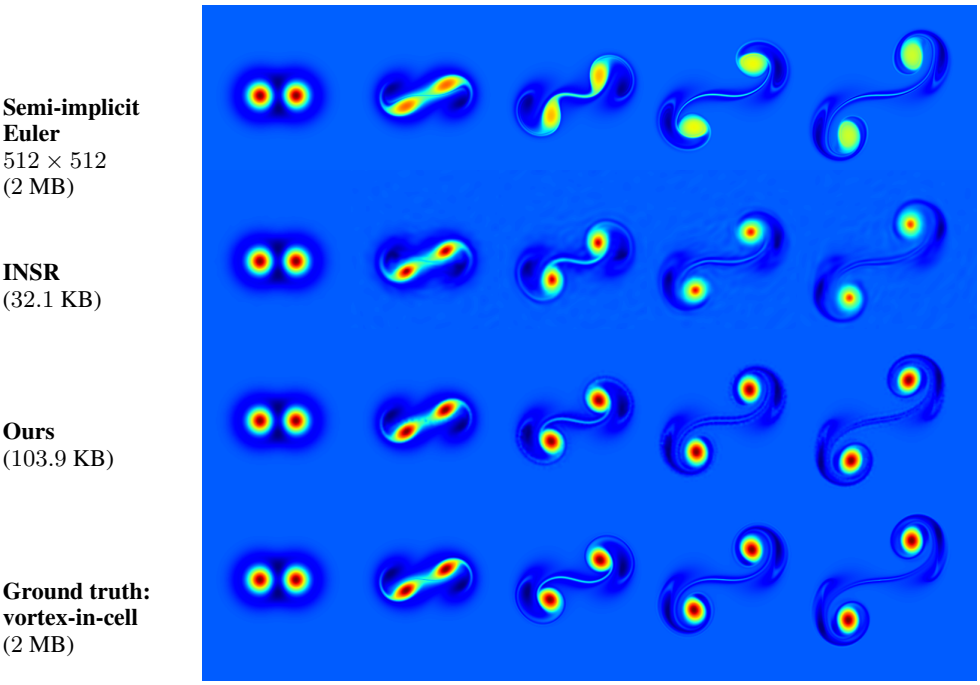


Figure 7: Taylor-vortex vorticity at frames 0, 100, 200, 300, and 399 (left to right). Rows show the semi-implicit Eulerian solver, INSR, ours, and the vortex-in-cell reference.

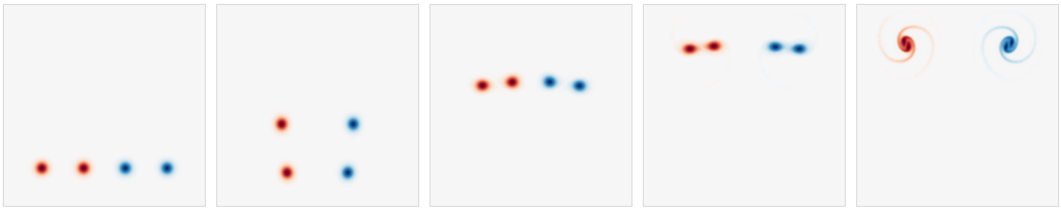


Figure 8: Leapfrogging vortices in 2D. Our vorticity fields at frames 0, 300, 930, 1420, and 1599 (left to right).

Leapfrogging vortex rings in 3D. The comparison in Fig. 9 uses two coaxial rings in $\Omega = [0,1]^3$, centered at $\boldsymbol{c}_1 = (0.75, 0.5, 0.5)^\top$ and $\boldsymbol{c}_2 = (0.85, 0.5, 0.5)^\top$, with radii $R_1 = 1/6$ and $R_2 = 0.7/6$. Both have normal $-\boldsymbol{e}_1$, core thickness $a = 0.12/6$, and strength coefficient $\gamma = 0.1/6$. Each ring is represented in the analytic initial field by $m = 500$ equally spaced vortex elements. With $\theta_k = 2\pi k/m$ for $k = 0, \dots, m-1$, define

$$\begin{aligned} \boldsymbol{q}_{jk} &= \boldsymbol{c}_j + R_j(\cos\theta_k\, \boldsymbol{e}_2 - \sin\theta_k\, \boldsymbol{e}_3), \\ \boldsymbol{t}_k &= -\sin\theta_k\, \boldsymbol{e}_2 - \cos\theta_k\, \boldsymbol{e}_3, \\ F_a(r) &= \frac{1 - \exp[-(r/a)^3]}{r^3}. \end{aligned} \tag{40}$$

The initial velocity is

$$\boldsymbol{u}^0(\boldsymbol{x}) = \sum_{j=1}^{2} \frac{\gamma R_j}{2m} \sum_{k=0}^{m-1} F_a(\|\boldsymbol{x} - \boldsymbol{q}_{jk}\|_2)\big[\boldsymbol{t}_k \times (\boldsymbol{x} - \boldsymbol{q}_{jk})\big]. \tag{41}$$

For distances below 10^{-6}, F_a is replaced by its limiting value a^{-3} and its radial derivative is set to zero. The initial fit uses the velocity and all nine entries of its analytic Jacobian. The box walls use free-slip, no-penetration boundary conditions, with boundary samples distributed over the six faces in proportion to their areas. Our method initializes a 40^3 lattice with three padding layers, giving $N = 46^3 = 97{,}336$ kernels, and uses $\eta = 15$ and $\Delta t = 0.02$; its learning rates and iteration budgets are given in Appendix A.2. Covector fluids, NFM, and VPFM use a spatial resolution of 128^3. The comparison covers frames 0–700, corresponding to $t \in [0, 14]$ for our simulation.

The average spatial-representation file size is 3.176 MB for Gaussian fluids across its 864 saved states and 2.745 MB for our method. For covector fluids, NFM, and VPFM, we count one single-precision velocity field on a 128^3 MAC grid: $3 \times 129 \times 128^2$ face-centered values require 25,362,432

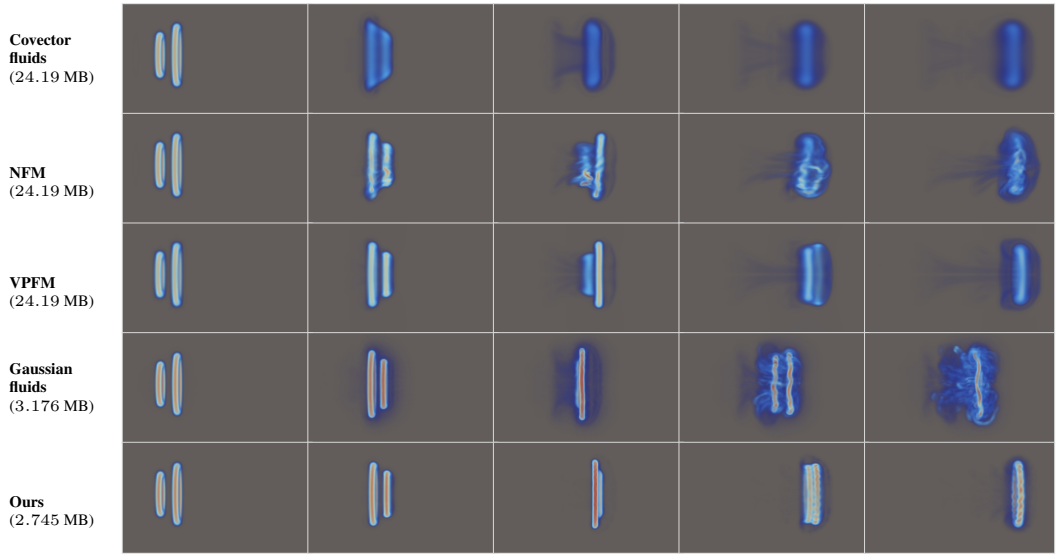


Figure 9: Leapfrogging vortex rings in 3D at frames 0, 150, 350, 550, and 700 (left to right), with methods ordered as labeled.

bytes, or 24.19 MB. The grid count excludes auxiliary solver state and temporal history; in particular, the VPFM number does not include its particle and flow-map state.

C.2 Boundary Conditions

Kármán vortex street. The domain is $\Omega = [-1.10321, 1.906778] \times [-0.598466, 0.60349]$. The initial velocity target and fixed background flow are $(0.5, 0)^\top$, with zero initial vorticity before the boundary fit. A stationary circular obstacle is centered at $(-0.80356845, -0.00502235)^\top$, with radius approximately 0.04553178. Its surface prescribes zero full velocity. The outer box prescribes horizontal normal flow of magnitude 0.5 at the left and right faces and zero normal flow at the upper and lower faces. Our method initializes centers on a 150×60 lattice with three padding layers, giving $N = 156 \times 66 = 10{,}296$ kernels, and uses $\eta = 9$. The initial boundary fit follows Appendix A.2.

All four methods use $\Delta t = 0.05$. Figure 10 shows NMC at frame 152 ($t = 7.6$) and Gaussian fluids, ours, and SCPF at frame 200 ($t = 10$). The SCPF reference uses a uniform 501×200 staggered grid with cell width $h = 0.00600796$; its discretized upper boundary is $y = 0.60312601$. It starts from the same uniform velocity, followed by boundary enforcement and an incompressibility projection.

The visualization uses fixed linear vorticity limits: NMC $[-16.6806, 16.6767]$, Gaussian fluids $[-16.8691, 16.0311]$, ours $[-13.7113, 13.2970]$, and SCPF $[-15, 15]$. Values outside these ranges are saturated for display.

Rotating cross. The domain is $[-5, 5]^2$. At $t = 0$, the obstacle is the union of $[-3.5, 3.5] \times [-0.25, 0.25]$ and $[-0.25, 0.25] \times [-3.5, 3.5]$. It rotates about the origin with angular velocity 1, giving rotation angle $\theta(t) = t$ and rigid-body velocity $\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{x}) = (-x_2, x_1)^\top$. The initial velocity and vorticity targets are zero before the initial boundary fit. The obstacle constrains the normal component of this rigid-body velocity, and the outer box prescribes zero normal velocity.

We initialize kernel centers on a 71×71 lattice with three padding layers, giving $N = 77^2 = 5929$ kernels, and use $\eta = 9$ and $\Delta t = 0.01$. The initial boundary fit follows Appendix A.2; during time integration, the normal-velocity boundary loss uses $\lambda_2 = 5$, with $\mathcal{L}_{\mathrm{b1}}$ omitted. Figure 11 shows frames 0, 100, 185, 320, and 501, corresponding to $t = 0$, 1, 1.85, 3.2, and 5.01.

C.3 Incompressibility

The piston experiment uses $\Omega = [-5, 5]^2$. The piston is a rectangle of width 8 and height 0.5, with center $\boldsymbol{c}(t) = (0, 2 - 0.5t)^\top$ and prescribed velocity $\boldsymbol{u}_{\mathrm{s}} = (0, -0.5)^\top$. Its surface constrains the full velocity, while the outer box prescribes zero normal velocity. Our method uses zero initial velocity and vorticity targets before the initial boundary fit described in Appendix A.2.

Our method starts with 5929 kernels, radius factor $\eta = 9$, and zero background velocity, and advances for 200 steps with $\Delta t = 0.05$, covering $t \in [0, 10]$. Kernel centers are optimized during

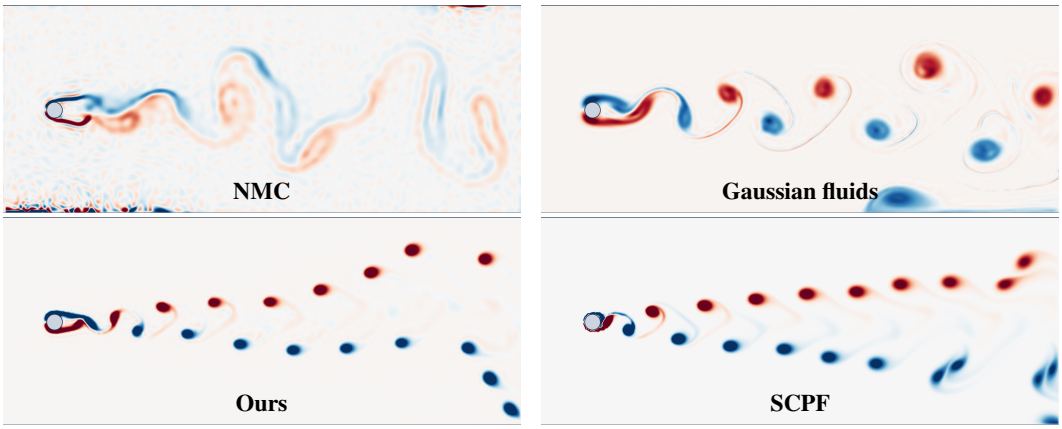


Figure 10: Kármán-vortex vorticity for NMC, Gaussian fluids, ours, and SCPF.

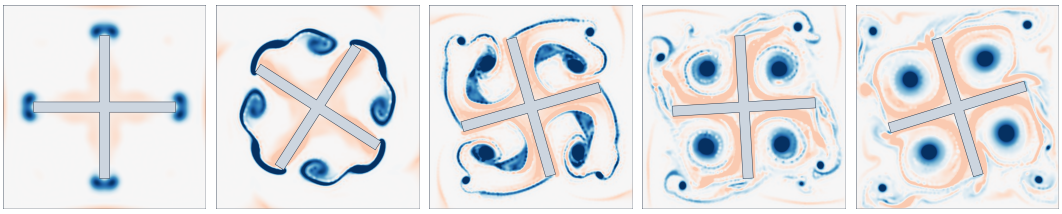


Figure 11: Rotating-cross vorticity at frames 0, 100, 185, 320, and 501 (left to right).

fitting without a separate advective predictor. The boundary-loss weights are $\lambda_1 = \lambda_2 = 20$ in the initial boundary fit and $\lambda_1 = \lambda_2 = 1$ during time integration.

The covector-fluids comparison uses a 400×400 staggered grid with spacing $h = 0.025$, second-order semi-Lagrangian covector transport, and a pressure projection obtained by a direct solve of the discrete Neumann problem. Each output interval of 0.05 contains five substeps of 0.01, giving 201 saved states over $t \in [0, 10]$. The initial zero velocity is projected with the moving-wall flux before the first saved state. The piston has prescribed normal velocity and tangential no-slip conditions imposed by odd reflection across its surface, including velocity constraints at the corners. The outer walls have zero normal velocity and constant tangential extension. The fixed-size piston is translated to the nearest grid faces, with positional error at most $h/2$ during substeps; its position agrees with the prescribed trajectory at every saved time.

The downward piston displacement requires an upward gap flux $Q = 8 \times 0.5 = 4$. We use 200 uniformly spaced midpoint samples on each unit-length side gap at the piston's top edge. With upward positive, the combined flux is approximated by $Q \approx 0.005 \sum_{k=1}^{400} u_y(\boldsymbol{x}_k)$. All three methods use these same 400 quadrature locations. Our method and Gaussian fluids are evaluated directly at these locations; covector velocities are bilinearly interpolated from their staggered grids, with constant tangential extension at the outer walls. For covector fluids, the resulting flux agrees with the sum over the native gap faces to numerical precision and differs from 4 by less than 4×10^{-8} over the saved states. This discrete flux balance does not imply that the interpolated velocity is pointwise divergence-free between grid locations.

The velocity streamlines in Fig. 1 are traced through bilinearly interpolated velocity samples on an 801×801 visualization grid at frame 60 ($t = 3$), with the piston interior masked. The covector velocity is first interpolated from its staggered grid onto this visualization grid. Line colors vary linearly with speed normalized by each method's maximum sampled fluid speed: 3.41986 for ours, 4.26007 for covector fluids, and 0.775531 for Gaussian fluids. Flux evaluation is independent of the visualization grid. The vorticity panels use linear red–blue scales fixed to the initial fluid-region ranges: $[-12.5212, 12.0269]$ for ours and the covector reference, and $[-3.67235, 3.72368]$ for Gaussian fluids. Values outside each range saturate at the endpoint colors. Covector vorticity is the discrete curl on the 401×401 grid vertices, with the piston masked. Its entire vorticity sequence retains the same fixed color scale as ours.

C.4 SMOKE

Both smoke examples start from a concentration profile

$$c_{\mathrm{s}}^{0}(\boldsymbol{x}) = \max\left(1 - \frac{\|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{s}}\|_2}{r_{\mathrm{s}}}, 0\right), \tag{42}$$

followed by the initial buoyancy fit in Appendix A.4. The concentration is seeded once, with no subsequent reinjection.

Rising smoke in 2D. The domain is $\Omega = [-5, 5]^2$, the source center is $\boldsymbol{x}_{\mathrm{s}} = (0, 2)^\top$, and its radius is $r_{\mathrm{s}} = 2$. The box walls use free-slip, no-penetration conditions. All three methods in Fig. 12 use 400×400 smoke-density samples and $\Delta t = 0.01$. Our method initializes $N = 77^2 = 5929$ kernels with $\eta = 9$ and uses buoyancy coefficient $\alpha = 50$, with the fitting and transport procedure in Appendix A.4.

We compare total smoke density normalized by each method's frame-0 value,

$$R(t) = \frac{\sum_{i,j} c_{\mathrm{s}}(\boldsymbol{x}_{ij}, t)}{\sum_{i,j} c_{\mathrm{s}}(\boldsymbol{x}_{ij}, 0)}. \tag{43}$$

Covector fluids uses cell-centered density samples, whereas Gaussian fluids and our method use endpoint-inclusive nodal grids. All samples, including boundary nodes, have equal weight in each sum. The fixed sample-area factor cancels in the ratio, with $R(t) = 1$ indicating unchanged total sampled density. We compare frames 0–99, corresponding to $t \in [0, 0.99]$, and measure the maximum relative drift as $\max_{0 \le n \le 99} |R(n\Delta t) - 1|$. The density snapshots show frame 0 and the three methods at frame 80 ($t = 0.8$), with $x_{2,\mathrm{display}} = -x_2$ to display buoyancy upward.

Smoky sphere in 3D. On $\Omega = [0, 1]^3$, the smoke source has center $\boldsymbol{x}_{\mathrm{s}} = (0.5, 0.8, 0.5)^\top$ and radius $r_{\mathrm{s}} = 0.15$; the spherical obstacle has center $(0.5, 0.4, 0.5)^\top$ and radius 0.1. Concentration is stored on a 128^3 nodal grid including the domain endpoints, and set to zero inside or on the sphere after initialization and each advection step. Both the sphere and box walls use free-slip, no-penetration conditions. Initialization and time integration use 1024 boundary samples per iteration: three quarters on the box faces in proportion to their areas and one quarter uniformly on the sphere, with $\lambda_2 = 1$ and $\mathcal{L}_{\mathrm{b1}}$ omitted. Interior fitting queries cover the full box, including the obstacle. We initialize $N = 46^3 = 97{,}336$ kernels with $\eta = 15$ and use $\Delta t = 0.02$. Figure 13 uses a fixed opacity mapping and displays $x_{2,\mathrm{display}} = 1 - x_2$ to show buoyancy upward.

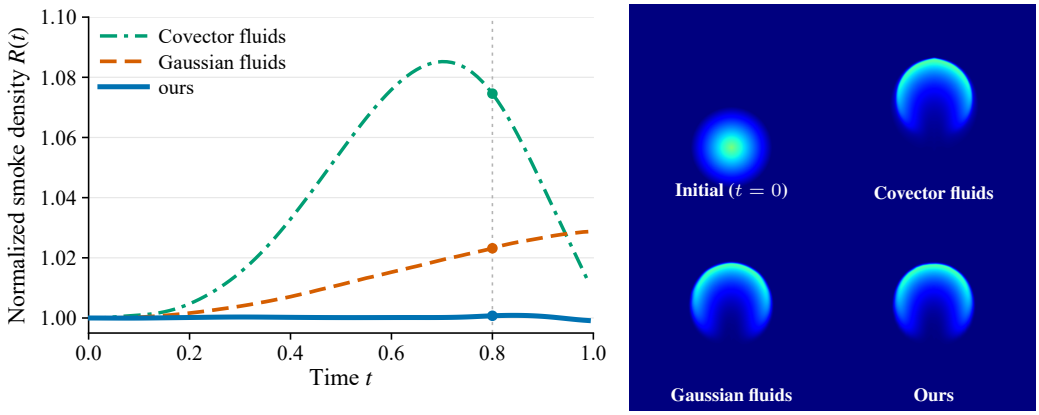


Figure 12: Rising-smoke comparison. Left: normalized total smoke density. Right: the initial field and frame-80 results.



Figure 13: Smoky-sphere density fields at frames 0, 20, 40, 70, and 99 (left to right).

D Ablation Study Settings

D.1 Taylor Vortex

Both formulations use the Taylor-vortex setup in Appendix C.1 and the Adam learning rates in Table 7. Initialization minimizes Eq. (8) for $10{,}000$ iterations; time integration uses Eq. (15) with $\lambda_2 = 1$ and $\mathcal{L}_{\mathrm{b1}}$ omitted for $5{,}000$ iterations per step over the first 10 steps. Both schedules use patience 100 iterations, with reduction factors 0.9 for initialization and 0.8 for time integration.

To exclude one-time startup costs, initialization time is averaged over its last $9{,}900$ iterations, and time integration over the final nine steps. We evaluate $\mathcal{L}_{\mathrm{init}}$ at the end of initialization and $\mathcal{L}_{\omega}$ at the end of the first time step. Figure 6 shows the corresponding loss curves.

Table 7: Adam learning rates for initialization and time integration in the Taylor-vortex ablation. Values are specified at the start of each fit.

Stage	Configuration	Centers	Radii	Strengths
Initialization	DFK-doublecurl	10^{-3}	5×10^{-3}	5×10^{-6}
	ours	10^{-3}	10^{-3}	5×10^{-6}
Time integration	DFK-doublecurl	10^{-4}	10^{-4}	10^{-4}
	ours	10^{-3}	10^{-2}	10^{-6}

D.2 Piston

Using the scene and shared simulation settings in Appendix C.3, Table 4 compares DFK-doublecurl with $\eta = 9$ or 27 and ours with $\eta = 9$.

Both formulations use Adam with the time-integration learning rates in Table 8, reset at each time step. Within each step, the learning-rate schedule uses reduction factor 0.8 and patience 100.

Table 8: Adam learning rates during time integration in the piston ablation. Values are reset at the start of each time step.

Configuration	Centers	Radii	Strengths
DFK-doublecurl ($\eta = 9$)	10^{-4}	10^{-4}	10^{-5}
DFK-doublecurl ($\eta = 27$)	10^{-4}	10^{-4}	10^{-5}
ours	10^{-3}	10^{-2}	10^{-6}

All three configurations are evaluated at the same $N_{\mathrm{b}} = 3400$ points fixed in the piston's local coordinates and translated with it at each frame. Uniform arc-length spacing 0.005 gives 1600 points per horizontal edge and 100 per vertical edge, located at segment midpoints to avoid corners. Only the moving piston surface is evaluated; the outer box is excluded.

Writing these local points as $\boldsymbol{q}_j$, we evaluate the full-velocity loss in physical units as

$$\mathcal{L}_{\mathrm{b1}}(t) = \frac{1}{2N_{\mathrm{b}}} \sum_{j=1}^{N_{\mathrm{b}}} \left\| \boldsymbol{u}_{\boldsymbol{\theta}(t)}\big(\boldsymbol{q}_j + \boldsymbol{c}(t)\big) - \boldsymbol{u}_{\mathrm{s}} \right\|_1 . \tag{44}$$

The factor $1/2$ averages the two velocity components, consistently with $\mathcal{D}_{\mathcal{Q}}$ in Sec. 4.2. The reported statistic is the equal-weight mean $201^{-1} \sum_{k=0}^{200} \mathcal{L}_{\mathrm{b1}}(0.05k)$, including the initial frame and excluding intermediate refitting states. No optimization weights or amplitude normalization are applied to these evaluation losses.