

Fluid Implicit Particles on Coadjoint Orbits

MOHAMMAD SINA NABIZADEH, University of California San Diego, USA

RITOBAN ROY-CHOWDHURY, University of California San Diego, USA

HANG YIN, University of California San Diego, USA

RAVI RAMAMOORTHI, University of California San Diego, USA

ALBERT CHERN, University of California San Diego, USA

We propose Coadjoint Orbit FLIP (CO-FLIP), a high order accurate, structure preserving fluid simulation method in the hybrid Eulerian-Lagrangian framework. We start with a Hamiltonian formulation of the incompressible Euler Equations, and then, using a local, explicit, and high order divergence free interpolation, construct a modified Hamiltonian system that governs our discrete Euler flow. The resulting discretization, when paired with a geometric time integration scheme, is energy and circulation preserving (formally the flow evolves on a coadjoint orbit) and is similar to the Fluid Implicit Particle (FLIP) method. CO-FLIP enjoys multiple additional properties including that the pressure projection is exact in the weak sense, and the particle-to-grid transfer is an exact inverse of the grid-to-particle interpolation. The method is demonstrated numerically with outstanding stability, energy, and Casimir preservation. We show that the method produces benchmarks and turbulent visual effects even at low grid resolutions.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; • **Mathematics of computing** → *Interpolation*; *Discretization*; **Partial differential equations**.

Additional Key Words and Phrases: Geometric fluid mechanics, Hamiltonian mechanics, Structure preserving discretizations, Mimetic interpolation

ACM Reference Format:

Mohammad Sina Nabizadeh, Ritoban Roy-Chowdhury, Hang Yin, Ravi Ramamoorthi, and Albert Chern. 2024. Fluid Implicit Particles on Coadjoint Orbits. *ACM Trans. Graph.* 43, 6, Article 270 (December 2024), 38 pages. <https://doi.org/10.1145/3687970>

1 INTRODUCTION

Discretizing incompressible fluid dynamical systems has been a central focus in the development of computational fluid dynamics and fluid animations in computer graphics. The goal is to find a computationally tractable finite dimensional analogue of the infinite dimensional continuous fluid system that properly represents the simulated measurements and visual phenomena. Among these fluid phenomena, the one that has received considerable attention is the conservation of *vortices* in fluids at low or zero viscosity. They constitute the main visual elements in smoke visual effects as well as

Authors' addresses: Mohammad Sina Nabizadeh, University of California San Diego, 9500 Gilman Dr, MC 0404, La Jolla, CA, 92093, USA, mnabizad@ucsd.edu; Ritoban Roy-Chowdhury, University of California San Diego, 9500 Gilman Dr, MC 0404, La Jolla, CA, 92093, USA, rroychowdhury@ucsd.edu; Hang Yin, University of California San Diego, 9500 Gilman Dr, MC 0404, La Jolla, CA, 92093, USA, h7yin@ucsd.edu; Ravi Ramamoorthi, University of California San Diego, 9500 Gilman Dr, MC 0404, La Jolla, CA, 92093, USA, ravir@ucsd.edu; Albert Chern, University of California San Diego, 9500 Gilman Dr, MC 0404, La Jolla, CA, 92093, USA, alchern@ucsd.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2024 Copyright held by the owner/author(s).

ACM 0730-0301/2024/12-ART270

<https://doi.org/10.1145/3687970>



Fig. 1. Pyroclastic cloud simulated using our method (CO-FLIP). Note the intricate vortical structures present on the plume despite the low simulation grid resolution of $96 \times 192 \times 96$. Inset: Image of volcano eruption from Mount St. Helens, Washington, USA, on May 18th, 1980 (Photograph by Harris & Ewing, Inc. [Public domain], via Wikimedia Commons).

ingredients in turbulent flows. This conservation of vortices, better known as the *circulation conservation*, is, however, challenging to reproduce accurately and stably after discretizations.

In this paper, we revisit the mathematical foundation of the vortex-related conservation laws. Specifically, the theory involves describing incompressible fluid dynamics using Lie algebras and Hamiltonian formulations. The payoff of the theory is that it provides a concrete characterization for the circulation conservation: “*the circulation state stays on a coadjoint orbit*,” a proposition that will be explained later (Section 4). We propose a discretization framework that satisfies this proposition. It turns out the natural algorithm derived from the framework is reminiscent of a Fluid Implicit Particles (FLIP) method [Brackbill et al. 1988], a familiar method in the fluid simulation community in graphics. Our discrete fluid system is therefore referred

Table 1. Method acronyms used throughout the paper.

Method	Reference	Acronym
Polynomial Particle-in-Cell	[Fu et al. 2017]	PolyPIC
Fluid Implicit Particles	[Brackbill et al. 1988]	FLIP
Polynomial FLIP	[Fei et al. 2021]	PolyFLIP
Reflection	[Zehnder et al. 2018]	R
Covector Fluids	[Nabizadeh et al. 2022]	CF
Neural Flow Map	[Deng et al. 2023]	NFM
Coadjoint Orbit FLIP	Our method	CO-FLIP

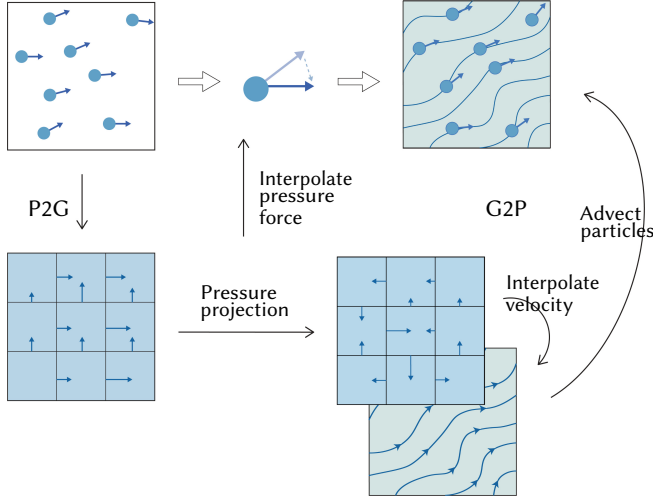


Fig. 2. The classical FLIP method for simulating incompressible fluids. Particles (top row) are moved, and their momentum updated, using a background velocity field which is reconstructed from the pressure projection of the transferred particle data to the grid (bottom row). The CO-FLIP algorithm modifies each stage: The particles move and their velocity covector is transformed (i) using a point-wise div-free interpolation (ii) of grid velocity data transferred from particles using the pseudoinverse of the div-free interpolation (iii), and, further, Galerkin pressure projected (iv). This time evolution is now a Hamiltonian ODE which we integrate geometrically (v).

to as Coadjoint Orbit Fluid Implicit Particles (CO-FLIP). Inheriting from the mathematical properties of the Hamiltonian formulation of fluid dynamics, CO-FLIP preserves energy and circulation even on a low resolution grid. Despite such low resolutions, our method produces intricate and fractal-shaped vortical structures (see Fig. 1). In addition, CO-FLIP exhibits outstanding stability compared to similar methods.

While the mathematical properties of CO-FLIP involve a deeper study in geometric fluid mechanics, our CO-FLIP method can be described using the familiar language of FLIP-based algorithms. See Table 1 for a list of acronyms used throughout the text.

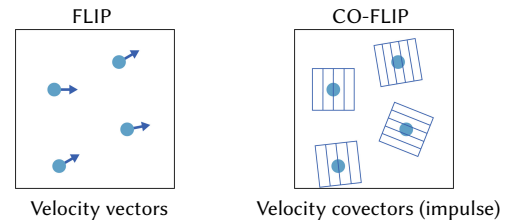
Quick Recall of the FLIP Method. FLIP [Brackbill et al. 1988] is one of the family of Particle in Cell (PIC) [Harlow 1962] methods or Material Point Methods (MPMs) [Sulsky et al. 1995]. They model incompressible and inviscid fluids governed by the Euler equations using a hybrid of Lagrangian and Eulerian coordinates. The fluid state is given by the positions and velocities of a large number of particles. There is also an auxiliary grid that discretizes the fluid domain. A so-called *particle-to-grid transfer* (P2G) interpolates the particle

velocities to the grid as an Eulerian velocity field. This grid velocity is typically stored in a Marker-And-Cell (MAC) staggered grid, with fluxes annotated on the grid's cell walls. The grid velocity data is then *pressure-projected* (PP), via solving a central-difference Poisson equation, to become a discrete divergence-free velocity field. The divergence-free grid velocity is interpolated (*grid-to-particle* (G2P)) into a continuous velocity field over the fluid domain. The particle positions are updated by flowing along this interpolated velocity field, and the particle velocity is updated by the interpolated pressure force computed in the pressure projection step. This concludes the entire time step of the FLIP algorithm. See Fig. 2.

Popular cousin methods include the classical PIC and Affine-PIC (APIC) [Jiang et al. 2015] methods. The classical PIC method replaces the particle velocity by the G2P velocity at every time step. APIC is a PIC method that additionally stores spatial gradients of velocities on particles for better quality P2G and G2P. FLIP is more advantageous to PIC or APIC as FLIP does not overwrite the particle velocity with a lower-resolution grid velocity. Thus FLIP is seen as a long-time method of characteristic, while PIC or APIC has their velocity state limited to a grid resolution comparable to purely grid-based semi-Lagrangian [Stam 1999] or finite volume methods [Mullen et al. 2009]. The downside of FLIP is that it has been observed to be more unstable compared to PIC. Common implementations of FLIP blend FLIP's particle velocity with PIC or APIC velocity for stability.

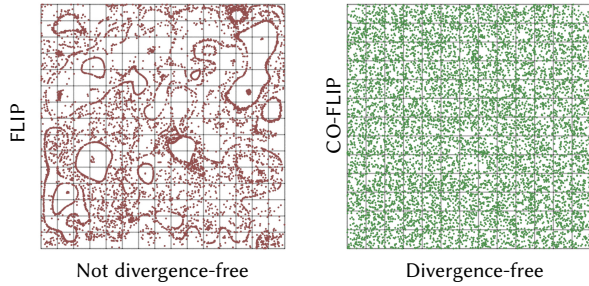
Our CO-FLIP Method. From a numerical viewpoint, to obtain CO-FLIP, we modify the FLIP method to satisfy the following properties. See Fig. 2 for a summary of the modifications.

- (i) The particle velocity is stored as a *velocity covector* [Nabizadeh et al. 2022], also known as an *impulse* [Feng et al. 2022; Deng et al. 2023]. As a particle position is updated by flowing along a G2P velocity, its velocity covector is transformed by the transpose of the G2P velocity gradient. As depicted in the inset, vectors are thought of as arrows with magnitude and direction, whereas covectors are taken as local level sets with magnitude (its compactness) and direction (its orientation).

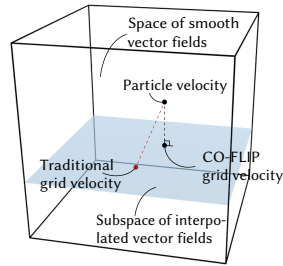


- (ii) The G2P interpolation is *mimetic*. That is, a discretely divergence-free MAC-grid velocity will interpolate into a continuously divergence-free vector field. This G2P interpolation is, therefore, an embedding from the finite dimensional space of grid velocity data into the infinite dimensional space of vector fields so that the discrete divergence-free subspace maps into the continuous divergence-free subspace. Using a specific usage of B-splines [Buffa et al. 2010; Schroeder et al. 2022; Roy-Chowdhury et al. 2024], we have an explicit local interpolation scheme that is mimetic and of arbitrarily high order. As shown in the inset, this results in point-wise divergence-free interpolated velocity

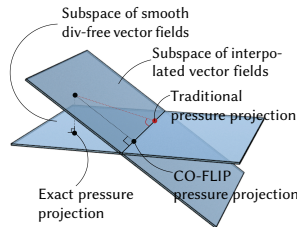
fields for the CO-FLIP method; as opposed to the traditional FLIP methods which show particle clumping.



(iii) The P2G interpolation is derived as the *pseudoinverse* of the mimetic G2P interpolation of (ii). If a particle velocity data is assigned by an interpolation from a grid velocity data, its P2G interpolation exactly reconstructs the grid velocity data (P2G \circ G2P is the identity map on the space of grid velocities). If the particle velocity is not in the range of G2P, the P2G operator orthogonally projects it on the image of G2P. In particular, the order of accuracy of the interpolation is directly governed by the B-spline order of (ii). There is no need for the APIC or PolyPIC [Fu et al. 2017] technique to include additional Taylor expansion modes on particles for improving the P2G accuracy. As illustrated in the inset, this means we orthogonally project to the subspace of interpolated velocities. In contrast, previous methods find a less accurate, non-orthogonal projection.



(iv) The discrete pressure projection step over the grid is *Galerkin*. Instead of solving the simplified central difference Poisson equation, we derive the pressure projection respecting the inner product structure induced by the continuous L^2 structure through the embedding (ii). In particular, this pressure projection is the exact L^2 projection to the divergence-free subspace of B-splines when it is mapped by the mimetic interpolation of (ii). Compared to traditional methods, this means we have a high order pressure projection with minimal error as demonstrated in the inset.



(v) The resulting time-evolution equation for the particles is a Hamiltonian ODE, allowing us to employ structure preserving geometric integrators.

Some of the above numerical ideas were known in the literature but have not been integrated into fluid simulations in a practical

manner. In that regard, we highlight a few additional numerical contributions.

- The divergence-free interpolation methods were recently introduced to computer graphics. Unlike these recent techniques requiring global linear solves [Chang et al. 2022; Lyu et al. 2024], our mimetic interpolation (ii) is local and explicit and does not require any linear solve.
- Pseudoinverse-based P2G (iii) discussed above is introduced in the original MPM work of [Sulsky et al. 1995] but was immediately simplified to the common PIC interpolations through mass lumping. Existing work XPIC [Hammerquist and Nairn 2017] and FMPM [Nairn and Hammerquist 2021] attempts to better approximate the pseudoinverse-based P2G through a truncated Neumann series. We show that the exact pseudoinverse-based P2G can be computed efficiently by a standard preconditioned conjugate gradient (PCG) solver.
- Structure-preserving time integrators [Mullen et al. 2009; Pavlov et al. 2011; Azencot et al. 2014] for (v) are usually implicit, involving expensive high-dimensional root finding with Newton solves. We show that these implicit methods can be bootstrapped from simple explicit integrators via a few fixed point iterations.

In addition to commonly tested fluid simulation benchmarks, we demonstrate the following conservation laws quantitatively. Some of these conservation laws have rarely been tested in previous numerical work. In addition to energy, momentum, and angular momentum conservation, in 2D $\iint (\text{curl } \vec{u})^k dA$ for every $k = 1, 2, 3, \dots$ is conserved under the Euler equation. In 3D, the helicity $\iiint \vec{u} \cdot (\text{curl } \vec{u}) dV$ is conserved. Here \vec{u} denotes the fluid velocity. These invariants of the Euler flow are known as the *Casimir invariants*. We point out that the conservation of these Casimir invariants is the quantitative measure for the circulation conservation.

Besides being a higher order method based on (ii)–(iv), the CO-FLIP algorithm shows superior invariant conservation, both in energy and Casimirs, and stability compared to a classical FLIP method. Moreover, CO-FLIP is capable of producing benchmark vortex phenomena at low grid resolutions (see Fig. 3). For instance, 64^3 , or even lower resolutions of 32^3 .

Glimpse of the Mathematics behind CO-FLIP. Beyond improving the results of FLIP, we elucidate that the CO-FLIP algorithm naturally arises when considering a structure-preserving discretization of the geometric mechanics formulation of the incompressible Euler equations.

In the continuous theory, the Euler equation is a Hamiltonian system. The phase space is given by the dual space of divergence-free vector fields, which is a Poisson space, and the Hamiltonian is given by the kinetic energy of the velocity field.

Towards the discrete theory, a mimetic interpolation (ii) gives us a finite-dimensional subspace of divergence-free vector fields to work with, which constitute grid velocities. We define a discrete Euler equation on the original continuous dual space of divergence-free vector fields, using the finite-dimensional Hamiltonian function defined over grid velocities. This discrete Euler flow has energy conservation and circulation conservation in the same way the continuous Euler flows obey these laws. Using a technique involving

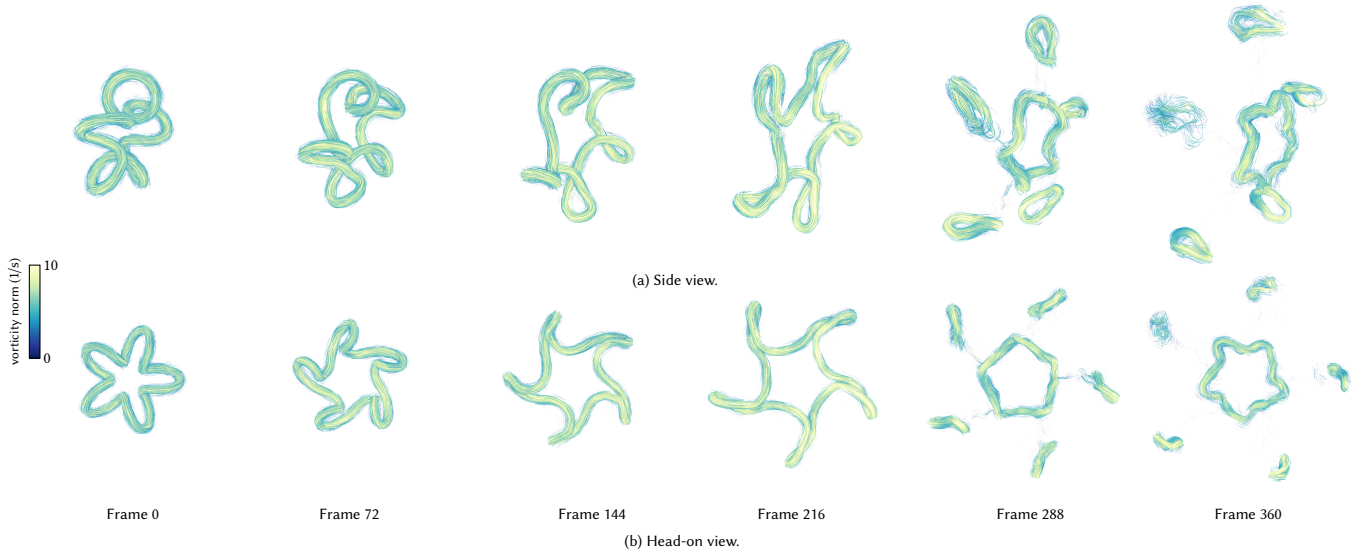


Fig. 3. The (1,5)-torus unknot. They are referred to as *unknot* as the numbers 1 and 5 are no longer coprime [Maggioni et al. 2010]. With time, the unknot moves forward and stretches apart until the vortex reconnection event happens. Our method at a low resolution of $64 \times 64 \times 64$ captures this phenomenon.

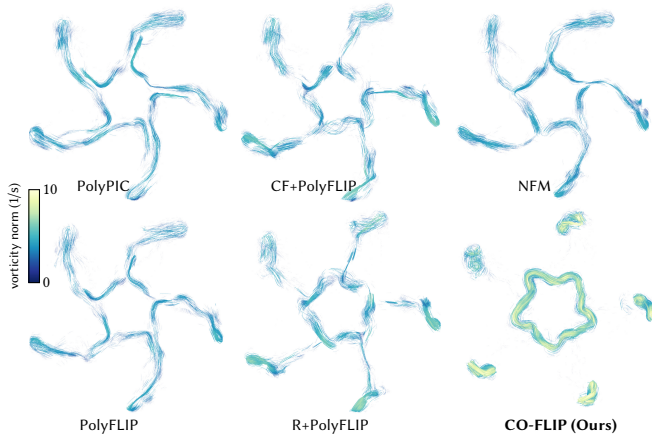


Fig. 4. Comparison of torus (1,5)-unknot experiment. Note that our method conserves energy and vortical structures throughout the simulation, while traditional methods lose both energy and helicity.

momentum maps, we can emulate this dynamical system on the Lagrangian coordinates, which leads to the particle-in-cell hybrid algorithm of CO-FLIP.

Overview. After a literature survey in Section 2, we first provide a high-level overview of the CO-FLIP algorithm in Section 3. Next, we elucidate the theoretical foundation of CO-FLIP in Section 4. In Section 5 we focus on the mimetic interpolations, their constructions, and their implications. The remainder of the paper consists of implementation details (Section 6) and numerical examples (Section 7).

2 RELATED WORK

2.1 Vorticity Conservation in Eulerian Fluid Simulation

Since the early work of Stam [1999] in the computer graphics field, the need for maintaining vorticity in the fluid simulations has received much attention. Indeed, the nonlinear advection operator present in the Navier-Stokes and Euler equations, when discretized on a grid, introduces much artificial viscosity. A variety of techniques have been developed to mitigate this dissipation. In particular, back-and-forth error compensation and correction (BFEC) and modified MacCormack methods [Dupont and Liu 2003; Kim et al. 2005; Selle et al. 2008], vorticity confinement method [Fedkiw et al. 2001], and advection-reflection [Zehnder et al. 2018] are few methods which tackle energy loss and restore the simulation vorticity otherwise lost.

Another class of approaches simulate the *vorticity equation*, which describes the evolution of the vorticity, instead of velocity [Selle et al. 2005; Elcott et al. 2007; Zhang et al. 2015]. By removing the splitting error between the projection and advection steps, these methods generally do well at conserving vorticity. However, they run into instability problems in three dimensions due to the stretching term, and further complicate handling of harmonic components [Yin et al. 2023].

Finally, a recent line of work has observed that the Euler equations can be recast as the Lie advection of the velocity covector [Nabizadeh et al. 2022] or impulse [Feng et al. 2022; Saye 2016; Yang et al. 2021]. Such methods provide a velocity-based formulation which similarly removes the splitting error without the complication of handling harmonic components.

In these semi-Lagrangian methods, Lagrangian features of the fluid are tracked over time, along the characteristics of the fluid flow. As such, an effective method of further removing dissipation is



Fig. 5. A nozzle shooting out smoke and hitting a Spot-shaped [Crane et al. 2013b] obstacle. Note the intricate vortical structures our method (CO-FLIP) can create at a low resolution of $128 \times 64 \times 64$.

to track characteristics further back in time. This idea of the long-term method of characteristic mappings (MCM) has been developed in multiple works [Tessendorf and Pelfrey 2011; Sato et al. 2018; Qu et al. 2019]. A recent work [Deng et al. 2023] uses a neural network to compress the large amount of velocity data needed to evaluate these long-term characteristics. In this work, we advocate an alternative that requires no compression but produces similarly accurate long-term characteristics. We, instead, opt to use particles to track Lagrangian features of the fluid. In particular, we track velocity covectors and deformation gradients on Lagrangian particles, so that the Lie advection of the velocity covector becomes solving an ODE on particles. This both minimizes splitting error, and tracks the long term features of fluids along characteristics, preserving circulation as well as energy.

2.2 Hybrid Eulerian-Lagrangian Methods

Particle-in-cell (PIC) algorithms for simulating fluids were first introduced by [Harlow 1962]. Since then, they have been used in a variety of scientific computing and computer graphics applications. In particular, Fluid Implicit Particles (FLIP) was developed by [Brackbill and Ruppel 1986; Brackbill et al. 1988] for use in magnetohydrodynamic calculations, and [Sulsky et al. 1995] developed the Material Point Method (MPM), which also tracks deformation gradients on particles to handle more sophisticated materials.

The material point method gained prominence in computer graphics after Stomakhin et al. [2013] used it for snow simulation, and subsequently has been extended to a wide variety of materials [Stomakhin et al. 2014; Ram et al. 2015; Klár et al. 2016; Jiang et al. 2017; Tampubolon et al. 2017] as well as scenarios like contact [Han et al. 2019] and fracture [Wolper et al. 2019]. There has also been significant work in improving the properties of the P2G transfer [Jiang et al. 2015; Fu et al. 2017; Hu et al. 2018; Fei et al. 2021] in order to reduce the dissipation involved in PIC, and the instabilities and noise in FLIP [Hammerquist and Nairn 2017]. The least-squares based P2G transfer proposed in this paper ((79)) uses a simple preconditioned conjugate gradient solver to alleviate both issues. First, the P2G transfer is directly defined as the pseudoinverse of G2P interpolation and, as such, matches its arbitrarily high order of accuracy. Unlike high-order PIC [Edwards and Bridson 2012], which used moving least squares reconstruction and WENO interpolation [Jiang and Shu 1996], our transfers canonically follow from the choice of B-spline

subspace. Second, the least-squares solve minimizes the instability problem involved with FLIP-based methods, which are due to divergent velocity modes on the particles that are in the kernel of the particle to P2G transfer, and therefore never get projected out [Hammerquist and Nairn 2017; Ding et al. 2020].

Particle-in-cell methods have also previously been used in graphics for fluid simulation. In graphics, FLIP for fluids was first suggested by [Zhu and Bridson 2005], and subsequent work has focused on maintaining incompressibility and volume conservation in the flow [Cornelis et al. 2014; Kugelstadt et al. 2019; Qu et al. 2022]. These approaches have all involved a more sophisticated pressure projection that directly involves particle positions in some capacity – they thus also ensure particles remain distributed evenly. In contrast, our Galerkin Pressure projection (Eq. (79)), does not involve particles directly and improves the incompressibility of fluids by finding the best divergence-free representative in the B-spline in the L^2 sense (See Appendix Section C.1). Additionally, the particles remain evenly distributed throughout the domain as they are advected by a point-wise divergence free interpolated velocity field, as discussed in Section 3.2.

2.3 Geometric Fluid Mechanics

Our algorithm is derived from physical principles arising in geometric fluid mechanics: the study of fluid mechanics from Lagrangian and Hamiltonian perspectives.

The story of geometric fluid mechanics begins in the 19th century, with Thomson [1868] circulation over closed loops is conserved, Helmholtz [1858] observing that vortex rings form stable structures under the Euler flow, and Clebsch [1859] showing that ideal fluids (at least with zero helicity) can be represented as a Lagrangian/Hamiltonian system. A truly general theory, however, only emerged after a breakthrough from Arnold [1966], who observed that the Euler flow is geodesic motion on the Lie group of volume-preserving diffeomorphisms. From this insight, much work has occurred in geometric fluid mechanics. In particular, the term helicity was coined by Moffatt [1969] and shown to be a topological invariant related to the knottedness of vortex lines [Moreau 1961; Moffatt 2014]. Significant advances were also made by Ebin and Marsden [1970], who addressed functional analytic issues in this representation and Marsden and Weinstein [1983] who unified this picture with Clebsch

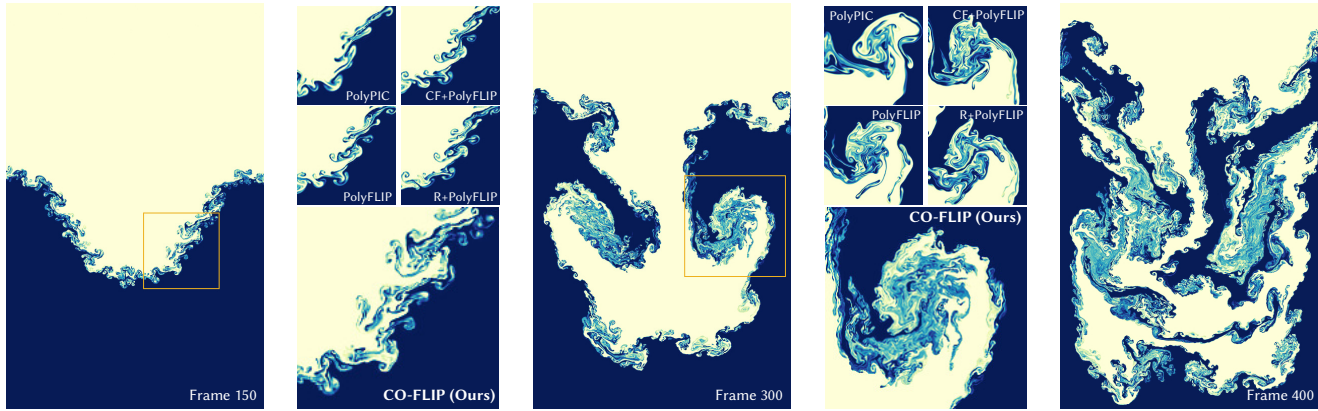


Fig. 6. Rayleigh-Taylor instability at frames 150 (left), 300 (middle), and 400 (right). The insets provide a closer look at how our method (CO-FLIP) produces and maintains an increasing amount of vortical structures compared with traditional methods. This behavior matches better the infinite fractal look results expected from buoyant fluids evolving with zero viscosity.

representations using the notion of *reduction* of a Hamiltonian system, and observed that Kelvin circulation conservation is a Noether current associated with particle relabeling symmetry.

The Hamiltonian view also suggests the covector or impulse formulation of Euler equation [Oseledets 1989; Cortez 1995], which have recently become influential in graphics following work by [Nabizadeh et al. 2022; Feng et al. 2022; Saye 2016]. The geometric formulation also gives a close relation between Euler flows and Schrödinger flows [Chern et al. 2016; Khesin et al. 2019] and Clebsch-based formulations, which have been used in graphics for vortical flow simulations [Yang et al. 2021; Xiong et al. 2022] and visualizations [Chern et al. 2017]. Recent developments in Geometric Fluid Mechanics include the classification of Casimir invariants in the Euler flow [Izosimov and Khesin 2017; Khesin et al. 2022]. These Casimirs are constant on coadjoint orbits and thus enable us to perform measurements that show coadjoint orbit conservation in Section 6.7.

For a broader discussion of geometric fluid mechanics, we refer the reader to the books by Arnold and Khesin [1998]; Holm [2011] or Morrison [1998].

2.4 Structure-Preserving Discretizations

The ideas of geometric fluid mechanics have inspired a wide array of both spatial and temporal discretizations, including ours, that aim to more faithfully reflect features of the underlying mathematical equations.

In computer graphics, Mullen et al. [2009] developed a reversible, energy preserving time integrator involving a Newton iteration. More generally, there are a wide variety of approaches to geometric numerical integration [Hairer et al. 2006] that can be used to conserve energy, symplectic or Poisson structures [Marsden et al. 1999], or remain on a Lie Group [Celledoni et al. 2014]. Such a Lie group integrator was applied in computer graphics by Azencot et al. [2014] for fluid simulation on surfaces. We adopt a method similar to Engö and Faltingsen [2001] for the time integration of Lie-Poisson

systems that preserves energy and remains on coadjoint orbits, see Section 3.4.

On the spatial discretization side, Pavlov et al. [2011] approached the problem by viewing discrete flow maps as stochastic, orthogonal matrices. The resulting discretization showed significantly better energy behavior than previous methods. Another line of structure-preserving spatial discretizations involves Nambu brackets, a generalization of Poisson brackets [Nambu 1973; Névir and Blender 1993], which have been extensively used in numerical weather prediction [Gassmann and Herzog 2008; Zängl 2013].

2.4.1 Mimetic Interpolation & Isogeometric Analysis. Our method takes advantage of a *mimetic interpolation* [Pletzer and Fillmore 2015; Pletzer and Hayek 2019] – that is, an interpolation that is consistent with a discrete derivative operator. For example, if grid data is discretely divergence free (such as that obtained after a pressure projection), then a mimetic interpolant should be pointwise divergence-free everywhere in the domain. Such interpolants were introduced to computer graphics by [Chang et al. 2022], but they have a long history in scientific computing. The particular mimetic interpolants that we adopt are based on B-splines, and have been used extensively in isogeometric analysis Buffa et al. [2010, 2011]; Evans and Hughes [2013] and finite element exterior calculus (FEEC) Arnold et al. [2006]. The idea is to observe that the mimetic interpolation properties of B-splines yield a discrete de Rham complex, which in turn yield natural finite element spaces for all of the involved quantities. Thus, FEEC is a generalization of discrete exterior calculus [Hirani 2003] using higher order function spaces. FEEC has given rise to several conservative simulation algorithms. For example, Zhang et al. [2022] developed a dual-field formulation of the Navier-Stokes equations that naturally discretizes to exactly conserve energy and helicity. We use a B-spline to represent grid data (see Section 5), as it guarantees that our particle advection action is volume preserving and that our pressure projection is weakly exact (Thm. 3.1).

Table 2. Notations and their meanings.

Notation	Meaning
M	Material space (Lagrangian viewpoint)
W	World space (Eulerian viewpoint)
(Σ, σ)	Symplectic manifold, and its associated 2-form
\mathcal{P}	Index set on the particles
$\mathfrak{X}(W), \mathfrak{X}^*(W)$	Space of vector fields on W , and its dual
$\mathfrak{X}_{\text{div}}(W), \mathfrak{X}_{\text{div}}^*(W)$	Space of div-free vector fields on W , and its dual
$\mathfrak{B}, \mathfrak{B}^*$	Space of discrete vector fields, and its dual
$\mathfrak{B}_{\text{div}}, \mathfrak{B}_{\text{div}}^*$	Space of discrete div-free vector fields, and its dual
I	Interpolation operator (aka G2P map)
I^+	Pseudoinverse of I using continuous metric
\hat{I}^+	I^+ using particle-based metric (aka P2G map)
I^\top	Adjoint of I
P	Pressure projection operator
d	Exterior derivative operator
$*$	Continuous Hodge star
\star	Galerkin Hodge star
$b, \#$	Musical isomorphisms
\mathcal{W}_p	p -th moment of the vorticity (2D Casimirs)
\mathcal{H}	Helicity (3D Casimir)
H	Hamiltonian function
$\text{SDiff}(W)$	Lie group of vol-preserving diffeomorphisms on W
Adv	Advection action on Σ by $\text{SDiff}(W)$
adv	Infinitesimal advection action induced by Adv
J_{adv}	Momentum map induced by adv

For a more thorough discussion of alternative approaches to mimetic interpolation, we refer the interested reader to [Schroeder et al. 2022; Roy-Chowdhury et al. 2024; Chang et al. 2022].

3 METHOD

In this section, we describe the CO-FLIP method for simulating incompressible and inviscid fluids. After a brief background of the governing PDE in Section 3.1, we describe a particle-in-cell framework, involving a finite dimensional velocity space as the grid structure in Section 3.2, and the particle dynamical system in Section 3.3. See Table 2 for notations used across the paper.

3.1 Euler Equation

Let $W \subset \mathbb{R}^n$, $n = 2$ or 3 , be a fluid domain in a Euclidean space. Let

$$\mathfrak{X}(W) := \{C^1 \text{ vector fields on } W\} \quad (1)$$

$$\mathfrak{X}_{\text{div}}(W) := \{\vec{v} \in \mathfrak{X}(W) \mid \nabla \cdot \vec{v} = 0, \vec{v} \cdot \vec{n}|_{\partial W} = 0\} \quad (2)$$

respectively denote the space of smooth (C^1) vector fields and the space of divergence-free vector fields satisfying the no-through boundary condition. Here \vec{n} denotes the normal vector of the domain boundary ∂W . The space $\mathfrak{X}_{\text{div}}$ models the space of velocity fields of a smooth incompressible flow. The space \mathfrak{X} is equipped with an L^2 inner product denoted by $\langle\langle \vec{v}, \vec{w} \rangle\rangle := \int_W \langle \vec{v}, \vec{w} \rangle d\mu$, where $d\mu$ is the Euclidean volume form on W .

The incompressible and inviscid fluid with uniform density is governed by the *Euler equation* that describes the time evolution of a divergence-free velocity field $\vec{v}(t) \in \mathfrak{X}_{\text{div}}(W)$:

$$\frac{\partial}{\partial t} \vec{v} + \vec{v} \cdot \nabla \vec{v} = -\nabla p, \quad \vec{v}(t) \in \mathfrak{X}_{\text{div}}(W), \quad (3)$$

where the pressure force “ $-\nabla p$,” p being a scalar function, is a general element of the orthogonal complement $\mathfrak{X}_{\text{div}}(W)^\perp$ in $\mathfrak{X}(W)$ with respect to $\langle\langle \cdot, \cdot \rangle\rangle$ that keeps \vec{v} on $\mathfrak{X}_{\text{div}}(W)$.¹ The equivalent *covector*

¹We acknowledge that the long-time existence of the Euler equation in the C^1 class is not guaranteed [Elgindi et al. 2019; Chen and Hou 2022].



Fig. 7. Rocket launching out into space with hot fuel smoke in its wake. Smoke ejection is simulated using our method (CO-FLIP). Note the turbulent features of the fuel plume captured despite the low resolution of $64 \times 64 \times 64$. The colors represent an artistic visualization of rocket engine exhaust.

formulation of the Euler equation (3) describes the time evolution of the velocity covector (1-form) $\vec{v}^b \in \Omega^1(W)$ [Nabizadeh et al. 2022]:

$$\frac{\partial}{\partial t} \vec{v}^b + \mathcal{L}_{\vec{v}} \vec{v}^b = -d(p - \frac{1}{2} |\vec{v}|^2), \quad \vec{v}(t) \in \mathfrak{X}_{\text{div}}(W), \quad (4)$$

where $\mathcal{L}_{\vec{v}}$ is the Lie derivative operator for 1-forms and d is the exterior derivative. One can further rewrite (4) in a more general form for 1-forms $\vec{u}(t)^b \in \Omega^1(W)$ with $\vec{u}(t) \in \mathfrak{X}(W)$ not necessarily divergence-free:

$$\frac{\partial}{\partial t} \vec{u}^b + \mathcal{L}_{\vec{v}} \vec{u}^b = -dq, \quad \vec{v} = P_{\mathfrak{X}_{\text{div}}} \vec{u}, \quad (5)$$

where $P_{\mathfrak{X}_{\text{div}}} : \mathfrak{X}(W) \rightarrow \mathfrak{X}_{\text{div}}(W)$ is the orthogonal projection, and the scalar function q is arbitrary. Note that \vec{u} and \vec{v} differ only by a gradient of an arbitrary scalar function, which is known as a *gauge degree of freedom* in \vec{u} . The field \vec{u} in (5) is referred to as an *impulse* in the literature [Cortez 1995; Feng et al. 2022]. The 1-form \vec{u}^b is also regarded as the *circulation field* since the circulation $\oint_C \vec{u}^b$ on any closed curve C is well-defined exactly with the gauge degree of freedom in \vec{u} . Finally, in terms of vectors \vec{u} , (5) is equivalent to

$$\frac{\partial}{\partial t} \vec{u} + \vec{v} \cdot \nabla \vec{u} + (\nabla \vec{v})^\top \vec{u} = -\nabla q, \quad \vec{v} = P_{\mathfrak{X}_{\text{div}}} \vec{u}. \quad (6)$$

The CO-FLIP method is a particle-grid hybrid method for simulating (5) (or (6)). The advected quantity \vec{u} is stored on particles, and the velocity \vec{v} generating the advection is stored on the grid. We first describe the grid structure in Section 3.2, which allows us to write down the ODE system for the particles in Section 3.3.

3.2 Divergence-Consistent Grid

As part of the setup, we suppose we have a *divergence-free interpolation scheme* to represent divergence-free vector fields. Precisely, there is a finite N -dimensional vector space \mathfrak{B} , representing a space of discrete vector field data (e.g. over a grid covering W). The space

\mathfrak{B} is equipped with a discrete divergence operator

$$\mathbf{d}: \mathfrak{B} \xrightarrow{\text{linear}} \mathfrak{D} \quad (7)$$

where \mathfrak{D} denotes some space of discrete divergence data. The discrete divergence operator (7) defines the space

$$\mathfrak{B}_{\text{div}} := \ker(\mathbf{d}) \subset \mathfrak{B} \quad (8)$$

of discrete divergence-free vector fields. What is referred to as an *interpolation scheme* is an operator that realizes each element of \mathfrak{B} as a smooth vector field

$$\mathcal{I}: \mathfrak{B} \xrightarrow{\substack{\text{linear,} \\ \text{injective}}} \mathfrak{X}(W), \quad \mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix} \mapsto \mathcal{I}(\mathbf{f}) = \sum_{i=1}^N f_i \vec{b}_i \quad (9)$$

via a set of given basis vector fields $\vec{b}_1, \dots, \vec{b}_N \in \mathfrak{X}(W)$. A *divergence-free interpolation scheme* is an interpolation \mathcal{I} so that discrete divergence free vector fields are realized as smooth divergence free vector fields

$$\mathcal{I}(\mathfrak{B}_{\text{div}}) \subset \mathfrak{X}_{\text{div}}(W). \quad (10)$$

In other words, the discrete and continuous divergence operators are consistent through the interpolation.

In Section 5 we construct a divergence-consistent interpolation method from a discrete grid using B-splines. In general, such interpolation methods that are consistent with the exterior derivatives (grad, curl, div) are known as a **mimetic interpolation**, also detailed in Section 5. In this section, we assume a divergence-free interpolation scheme \mathcal{I} is given, so that we can describe the algorithm in terms of elementary linear algebra.

Once $(\mathfrak{B}, \mathfrak{B}_{\text{div}}, \mathcal{I})$ is given, we have an induced inner product structure on \mathfrak{B} from the L^2 inner product structure of $\mathfrak{X}(W)$ through the map $\mathcal{I}: \mathfrak{B} \rightarrow \mathfrak{X}(W)$:

$$\langle \mathbf{f}, \mathbf{g} \rangle_{\mathfrak{B}} = \mathbf{f}^\top \star \mathbf{g} := \langle \mathcal{I}(\mathbf{f}), \mathcal{I}(\mathbf{g}) \rangle = \int_W \langle \mathcal{I}(\mathbf{f}), \mathcal{I}(\mathbf{g}) \rangle d\mu \quad (11)$$

for all $\mathbf{f}, \mathbf{g} \in \mathfrak{B}$, where \star denotes the inner product matrix, which is an N -by- N symmetric positive definite matrix explicitly given by $\star_{ij} = \langle \vec{b}_i, \vec{b}_j \rangle$. Let $\mathbf{P}_{\mathfrak{B}_{\text{div}}}: \mathfrak{B} \rightarrow \mathfrak{B}_{\text{div}}$ denote the orthogonal projection to $\mathfrak{B}_{\text{div}} = \ker(\mathbf{d})$ with respect to $\langle \cdot, \cdot \rangle_{\mathfrak{B}}$, given by

$$\mathbf{P}_{\mathfrak{B}_{\text{div}}} = \text{id} - \star^{-1} \mathbf{d}^\top (\mathbf{d} \star^{-1} \mathbf{d}^\top)^{-1} \mathbf{d}, \quad (12)$$

derived in Appendix C.1. The discrete projection (12) is analogous to the continuous pressure projection $(\text{id} - \nabla \Delta^{-1} (\nabla \cdot))$. One may equivalently write (12) in terms of streamfunctions, which is detailed in Section 6.1.3. We refer $\mathbf{P}_{\mathfrak{B}_{\text{div}}}$ as the **discrete pressure projection operator**. Note that the divergence-consistency (10) of the interpolation \mathcal{I} implies that the discrete pressure projection is *exact* in the weak sense. That is, it results in the closest-point approximation to the exact pressure projection:

Theorem 3.1. *The resulting vector field $\mathcal{I}(\mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathbf{f})$ by the discrete pressure projection is, among $\mathcal{I}(\mathfrak{B}_{\text{div}})$, the closest element to the continuous pressure projection $\mathbf{P}_{\mathfrak{X}_{\text{div}}} \mathcal{I}(\mathbf{f})$ performed in $\mathfrak{X}(W)$. That is,*

$$\left(\mathcal{I}(\mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathbf{f}) - \mathbf{P}_{\mathfrak{X}_{\text{div}}} \mathcal{I}(\mathbf{f}) \right) \perp_{L^2} \mathcal{I}(\mathfrak{B}_{\text{div}}) \quad \text{for all } \mathbf{f} \in \mathfrak{B}. \quad (13)$$

PROOF. Appendix C.2. \square

3.3 Equations of Motion of CO-FLIP

CO-FLIP is a dynamical system for particles. Now, we describe the equations of motion for this particle system. Let \mathcal{P} denote the index set for the particles. The time-dependent state $(\vec{x}_p(t), \vec{u}_p(t))_{p \in \mathcal{P}}$ of the system consists of the *position* $\vec{x}: \mathcal{P} \rightarrow W$ of each particle, and the *impulse* $\vec{u}: \mathcal{P} \rightarrow \mathbb{R}^n$ at each particle.

The equations of motion for the particles are given by

$$\begin{cases} \frac{d}{dt} \vec{x}_p(t) = \vec{v}|_{\vec{x}_p(t)}, & p \in \mathcal{P} \end{cases} \quad (14a)$$

$$\begin{cases} \frac{d}{dt} \vec{u}_p(t) = -(\nabla \vec{v}|_{\vec{x}_p(t)})^\top \vec{u}_p(t), & p \in \mathcal{P} \end{cases} \quad (14b)$$

$$\begin{cases} \vec{v} = \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \hat{\mathcal{I}}^+(\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}} \in \mathfrak{X}_{\text{div}}(W), \end{cases} \quad (14c)$$

as a discretization of (6) (with $q = 0$) along advected particles. Here $\hat{\mathcal{I}}^+$ is the pseudoinverse of \mathcal{I} that takes in a state $(\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}}$ and returns an element in \mathfrak{B} by

$$\hat{\mathcal{I}}^+(\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}} := \operatorname{argmin}_{\mathbf{f} \in \mathfrak{B}} \sum_{p \in \mathcal{P}} |(\mathcal{I}(\mathbf{f}))_{\vec{x}_p} - \vec{u}_p|^2 \mu_p, \quad (15)$$

where μ_p is a particle volume preserved over time.

Theorem 3.2. $\hat{\mathcal{I}}^+$ is an exact left-inverse of \mathcal{I} in the sense that

$$\mathbf{f} = \hat{\mathcal{I}}^+ \left(\vec{x}_p, \mathcal{I}(\mathbf{f})_{\vec{x}_p} \right)_{p \in \mathcal{P}} \quad \begin{array}{l} \text{for all } \mathbf{f} \in \mathfrak{B} \text{ and particle} \\ \text{positions } (\vec{x}_p)_{p \in \mathcal{P}}, \end{array} \quad (16)$$

provided that $(\vec{x}_p)_{p \in \mathcal{P}}$ is sufficiently dense so that the minimizer for (15) is unique.

PROOF. Appendix C.3. \square

In the language of PIC and FLIP, the interpolation $\mathbf{f} \in \mathfrak{B} \mapsto (\vec{x}_p, \mathcal{I}(\mathbf{f})_{\vec{x}_p})_{p \in \mathcal{P}}$ is our grid-to-particle (G2P) transfer, while the least squares solve $\hat{\mathcal{I}}^+$ is our particle-to-grid (P2G) transfer. In particular, (16) states that P2G perfectly reconstructs the grid data if the particle data comes from a G2P transfer. We detail, in Section 7.1.1, an efficient manner of solving this problem using a preconditioned conjugate gradient (PCG) solver.

Also note that the C^0 -continuous gradient of velocity $\nabla \vec{v}$ in (14b) can also be evaluated exactly since \vec{v} is in the image of \mathcal{I} . For example if $\vec{v} = \mathcal{I}(\mathbf{f}) = \sum_{i=1}^N f_i \vec{b}_i$ then $\nabla \vec{v} = \sum_{i=1}^N f_i \nabla \vec{b}_i$.

This concludes the CO-FLIP dynamical system as an ODE (14).

3.4 Time Integration

The time integration of (14) is dissected into two levels. At the first level, we describe the time integration of (14a) and (14b) for a time-independent velocity field \vec{v} . Second, we describe the coupling with (14c).

3.4.1 Advection action. When a smooth divergence-free vector field $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$ is given and fixed over time, the ODE system (14a) and (14b) for $(\vec{x}_p(t), \vec{u}_p(t))_{p \in \mathcal{P}}$ can be integrated accurately, efficiently, and in parallel over $p \in \mathcal{P}$, using the 4th order Runge-Kutta (RK4) method. Conceptually, this first level ODE solve is the evaluation of a Lie group action on the state $(\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}}$ by an element of the Lie algebra (i.e. generator) $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$. Let us denote this advection

by a fixed $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$ over a given time span $\tau \geq 0$ by

$$\begin{aligned} \text{Adv}_{\vec{v}}^{\tau}: W \times \mathbb{R}^n &\rightarrow W \times \mathbb{R}^n, \\ \text{Adv}_{\vec{v}}^{\tau}(\vec{x}^0, \vec{u}^0) &:= (\vec{x}(\tau), \vec{u}(\tau)), \\ \text{where } \begin{cases} \frac{d}{dt}\vec{x}(t) &= \vec{v}|_{\vec{x}(t)} \\ \frac{d}{dt}\vec{u}(t) &= -(\nabla\vec{v}|_{\vec{x}(t)})^{\top}\vec{u}(t) \\ \vec{x}(0) &= \vec{x}^0, \vec{u}(0) = \vec{u}^0. \end{cases} \end{aligned} \quad (17)$$

The operator $\text{Adv}_{\vec{v}}^{\tau}$ also acts on the particle state space $(W \times \mathbb{R}^n)^{|\mathcal{P}|}$ by mapping the operation in parallel to each (\vec{x}_p, \vec{u}_p) for $p \in \mathcal{P}$. Note that we have group structure $\text{Adv}_{\vec{v}}^{\tau_1} \text{Adv}_{\vec{v}}^{\tau_2} = \text{Adv}_{\vec{v}}^{\tau_1 + \tau_2}$ and $\text{Adv}_{a\vec{v}}^{\tau} = \text{Adv}_{\vec{v}}^{a\tau}$.

3.4.2 Lie group integrator. At the second level, we couple the above group action (17) with (14c) using a Lie group integrator. That is, we construct a higher order time integration method only by compositions of group actions. For simplicity in notation, we abbreviate each state by $\mathbf{y} = (\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}} \in (W \times \mathbb{R}^n)^{|\mathcal{P}|}$, and we write (14c) as $\vec{v} = \mathcal{I}\mathcal{F}(\mathbf{y})$ where

$$\mathcal{F}: (W \times \mathbb{R}^n)^{|\mathcal{P}|} \rightarrow \mathfrak{B}_{\text{div}}, \quad \mathcal{F} = \mathbf{P}\mathfrak{B}_{\text{div}}\hat{\mathcal{I}}^+. \quad (18)$$

We adopt a trapezoidal rule. On a time axis $(n\Delta t)_{n=0,1,2,\dots}$ discretized by a uniform size $\Delta t > 0$, a given state $\mathbf{y}^{(n)}$ at the n -th time step determines the state $\mathbf{y}^{(n+1)}$ at the next time step by solving

$$\begin{cases} \mathbf{y}^{(n+1)} = \text{Adv}_{\mathcal{I}\mathbf{f}^*}^{\Delta t} \mathbf{y}^{(n)} \end{cases} \quad (19a)$$

$$\begin{cases} \mathbf{f}^* = \frac{1}{2} (\mathcal{F}(\mathbf{y}^{(n)}) + \mathcal{F}(\mathbf{y}^{(n+1)})). \end{cases} \quad (19b)$$

The intermediate grid velocity data $\mathbf{f}^* \in \mathfrak{B}_{\text{div}}$ represents the grid velocity at the midpoint, *i.e.* the $(n + 1/2)$ -th time step.

In Section 4.6 (Corollary 4.4) we show that (19) becomes an energy preserving integrator in the limit where the particle density approaches a continuum (the grid and temporal resolution stay fixed during this limit).

A simple method to evaluate the solution to (19) is by a fixed point iteration:

Algorithm 1 CO-FLIP Trapezoidal Integrator

Input: $\mathbf{y}^{(n)} = (\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}}^{(n)}$ \triangleright Particle position and impulse.
 1: $\mathbf{f}^* \leftarrow \mathcal{F}(\mathbf{y}^{(n)});$ \triangleright Projected particle-to-grid
 2: **do**
 3: $\mathbf{y}^{(n+1)} \leftarrow (19a);$ \triangleright Advect the particles.
 4: $\mathbf{f}^* \leftarrow (19b);$ \triangleright Averaged particle-to-grid velocity.
 5: **while** $\mathcal{F}(\mathbf{y}^{(n+1)})$ not converged.
Output: $\mathbf{y}^{(n+1)}$.

The stop criteria is determined either by a max number of iterations, or when $\|\mathcal{F}(\mathbf{y}^{(n+1)}) - \mathcal{F}(\mathbf{y}^{(n)})\| / \|\mathcal{F}(\mathbf{y}^{(0)})\|$ has converged to a desired tolerance. This algorithm is implementation-wise attractive since it only requires calling the explicit advection procedure (17) and evaluating \mathcal{F} (18), which are familiar subroutines in any PIC or FLIP implementation. It is in contrast to a full Newton's method for (19) that would involve computing the system's Hessian [Mullen et al. 2009; Azencot et al. 2014]. When only one iteration

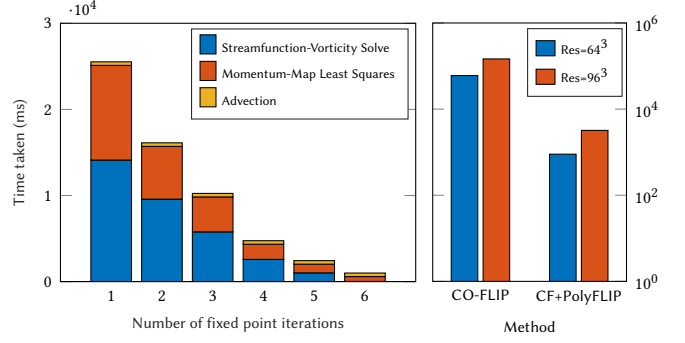


Fig. 8. The average time taken per fixed point iteration during the trapezoidal implicit time integration; for the Trefoil knot experiment in Fig. 24 at resolution 64^3 (left), and a comparison of the total time taken for different resolutions of CO-FLIP and CF+PolyFLIP (right). Note that the exponential drop in time taken to run the later iterations. Compared to an explicit midpoint solver (first two iterations only), these additional steps only incur 50% additional computational time. In comparison to traditional methods, our method takes longer to run, and it scales linearly as number of grid cells increase.

is performed, this is the forward Euler method or time-splitting method adopted in most particle-in-cell methods. Performing two iterations corresponds to the higher order time integration, such as explicit trapezoidal or midpoint schemes, adopted in [Narain et al. 2019; Nabizadeh et al. 2022]. We run the fixed point iteration until convergence, which typically takes up to 4–6 iterations (note that later iterations cost less, see Fig. 8).

3.5 Energy-Based Correction

The $\hat{\mathcal{I}}^+$ (15) and the \mathcal{F} (18) operators are projection operators, which take a vector field represented by the particles, and send to the image of $\mathcal{I}: \mathfrak{B} \rightarrow \mathfrak{X}(W)$ and $\mathcal{I}: \mathfrak{B}_{\text{div}} \rightarrow \mathfrak{X}_{\text{div}}(W)$. However, the projection $\hat{\mathcal{I}}^+$ (which is a part of \mathcal{F}) is orthogonal only with respect to the metric $\sum_{p \in \mathcal{P}} |\cdot|^2 \mu_p$ instead of the continuous L^2 metric $\int_W |\cdot|^2 d\mu$. Increasing the particle density essentially improves the approximation of $\sum_{p \in \mathcal{P}} |\cdot|^2 \mu_p$ to $\int_W |\cdot|^2 d\mu$. In other words, the difference between the two norms reflects a discretization error due to the particles not being infinitely dense.

Here, we propose a simple method to include a correction term that compensates this discretization error. We will utilize the facts that (i) we do have access to the exact L^2 norm when the data is on \mathfrak{B} (*cf.* (11)), and (ii) at the limit where particles become a continuum the method (19) is energy preserving (Corollary 4.4 in Section 4.6).

The general idea goes as follows. Let $\mathbf{f}^{(n)} \approx \mathcal{F}(\mathbf{y}^{(n)})$, $\mathbf{f}^{(n+1)} \approx \mathcal{F}(\mathbf{y}^{(n+1)})$ represent modified grid velocities projected from the particle data $\mathbf{y}^{(n)}$, $\mathbf{y}^{(n+1)}$. This modification is made minimally so that the kinetic energy is conserved: $|\mathbf{f}^{(n+1)}|_{\mathfrak{B}}^2 = |\mathbf{f}^{(n)}|_{\mathfrak{B}}^2$ for all n , where $|\cdot|_{\mathfrak{B}}^2$ is given by (11) which yields the exact L^2 norm of the interpolated grid velocities. Note that by the difference of squares

formula, the condition $|\mathbf{f}^{(n+1)}|_{\mathfrak{B}}^2 - |\mathbf{f}^{(n)}|_{\mathfrak{B}}^2 = 0$ translates to

$$\underbrace{\langle \mathbf{f}^{(n+1)} - \mathbf{f}^{(n)}, \mathbf{f}^{(n+1)} + \mathbf{f}^{(n)} \rangle_{\mathfrak{B}}}_{\Delta \mathbf{f}} = 0. \quad (20)$$

To impose this energy conservation condition, we project an original difference $(\Delta \mathbf{f})_{\text{original}} = \mathcal{F}(\mathbf{y}^{(n+1)}) - \mathbf{f}^{(n)}$ to the orthogonal complement of $\tilde{\mathbf{f}}^*$, the intermediate grid velocity projected from the particle data. This treatment corresponds to the following implicit scheme.

Initialize $\mathbf{f}^{(0)} = \mathcal{F}(\mathbf{y}^{(0)})$. Given $\mathbf{y}^{(n)}, \mathbf{f}^{(n)}$, determine $\mathbf{y}^{(n+1)}, \mathbf{f}^{(n+1)}$ by solving

$$\mathbf{y}^{(n+1)} = \text{Adv}_{\mathcal{I}\tilde{\mathbf{f}}^*}^{\Delta t} \mathbf{y}^{(n)} \quad (21a)$$

$$\tilde{\mathbf{f}}^* = \frac{1}{2} (\mathbf{f}^{(n)} + \mathbf{f}^{(n+1)}) \quad (21b)$$

$$\mathbf{f}^{(n+1)} = \mathbf{f}^{(n)} + \mathbf{P}_{\tilde{\mathbf{f}}^* \perp} (\mathcal{F}(\mathbf{y}^{(n+1)}) - \mathbf{f}^{(n)}) \quad (21c)$$

where $\mathbf{P}_{\tilde{\mathbf{f}}^* \perp} \mathbf{f} = \mathbf{f} - \tilde{\mathbf{f}}^* \langle \tilde{\mathbf{f}}^*, \mathbf{f} \rangle_{\mathfrak{B}} / |\tilde{\mathbf{f}}^*|_{\mathfrak{B}}^2$ is the orthogonal projection to the orthogonal complement of $\tilde{\mathbf{f}}^*$.

Similar to (19), the solution to (21) can be evaluated using a fixed point iteration with an implicit trapezoidal integrator:

Algorithm 2 CO-FLIP Integrator with Energy-Based Correction

Input: $\mathbf{y}^{(n)}, \mathbf{f}^{(n)}$ ▷ Particle state and grid state
 1: $\tilde{\mathbf{f}}^* \leftarrow \mathcal{F}(\mathbf{y}^{(n)});$ ▷ Projected particle-to-grid
 2: **do**
 3: $\mathbf{y}^{(n+1)} \leftarrow (21a);$ ▷ Advection of the particles.
 4: $\mathbf{f}^{(n+1)} \leftarrow \mathcal{F}(\mathbf{y}^{(n+1)});$ ▷ Projected particle-to-grid.
 5: $\tilde{\mathbf{f}}^* \leftarrow (21b);$ ▷ Average.
 6: $\mathbf{f}^{(n+1)} \leftarrow \mathbf{f}^{(n)} + \mathbf{P}_{\tilde{\mathbf{f}}^* \perp} (\mathbf{f}^{(n+1)} - \mathbf{f}^{(n)});$ ▷ Correction (21c).
 7: **while** $\mathcal{F}(\mathbf{y}^{(n+1)})$ not converged.
Output: $\mathbf{y}^{(n+1)}, \mathbf{f}^{(n+1)}$.

We let the fixed point iteration run until convergence with the same criteria as Algorithm 1. For our experiments, we keep the CFL number close to 0.5, which results in 4-6 iterations to exit for a tolerance of 10^{-9} in 2D (10^{-7} in 3D). As shown in Fig. 8, by reusing the previous iterations' results as guesses to the solvers for the next iteration, each step of the fixed-point iteration runs progressively faster than the one before. In practice, we observe an increase in computation time of roughly 50% compared to an explicit second-order method. This is a relatively small price to pay for the conservation of energy. Preserving energy informs us quantitatively that the algorithm is correctly simulating the \mathcal{I} -discrete Euler flow (Section 4.2). Note that energy conservation is only possible when $\tilde{\mathbf{f}}^*$ has sufficiently converged using our implicit integration; otherwise, the explicit scheme cannot benefit from the energy correction (see Fig. 9). We use Algorithm 2 for all of our experiments.

3.6 Discussion

We have described the essence of the CO-FLIP method. One only needs to provide a divergence-consistent interpolation scheme $\mathcal{I}: \mathfrak{B}_{\text{div}} \rightarrow \mathfrak{X}_{\text{div}}(W)$. Given such an interpolation method, the

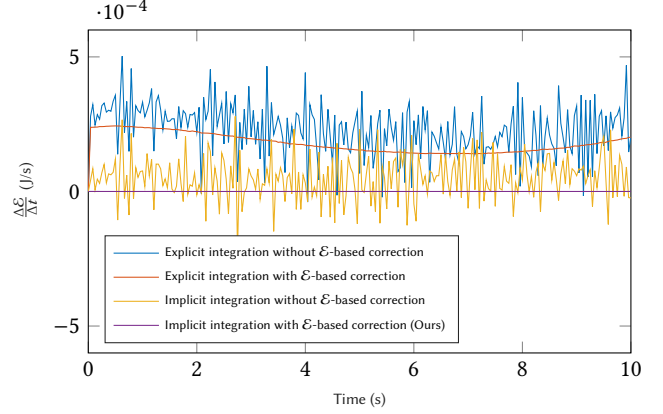


Fig. 9. Comparison of time integration method used for our method. This study compares energy change within a timestep depending on whether our method uses either an implicit or explicit trapezoidal integrator, with or without energy-based correction. Note that the energy-based correction only works as expected with an implicit integrator since $\tilde{\mathbf{f}}^*$ is correctly evaluated with convergence of the fixed point iteration.

rest of the template description canonically unfolds into a fluid simulation method.

In particular, the spatial accuracy of the method depends only on the choice of the grid interpolation \mathcal{I} (assuming the typical setup that the particles are sufficiently dense compared to the grid). A high-order interpolation scheme yields a globally high-order fluid solver. This is in contrast to most fluid solvers, where achieving a certain global accuracy requires designing multiple high-order subroutines, such as particle-to-grid transfer, grid-to-particle transfer, and pressure solves. We provide concrete constructions of divergence-consistent interpolations of arbitrary order in Section 5. In addition to having a high order of accuracy, the method preserves energy and circulations. This is a consequence of the associated Hamiltonian system governing the CO-FLIP dynamics.

Some practical implementation details are postponed to Section 6. We discuss an efficient construction of our high-order pressure projection, an adaptive resetting strategy, and details regarding minimizing the instability of FLIP-based methods. Additional details regarding the construction of the pseudoinverse and streamform-vorticity solves are presented in Section 7.1.

In the following Section 4, we detail the mathematical foundation of the CO-FLIP algorithm. In particular, we use the framework of geometric fluid mechanics to show that CO-FLIP satisfies the conservation of circulation and energy in a precise manner (Section 4.6). The theory relies on the background on Poisson structure and Lie algebras developed in Appendix A. Readers primarily interested in the implementation may skip to Section 5.

4 THEORY PART 1: A STRUCTURE PRESERVING FLUID SIMULATOR

In this section, we show that the CO-FLIP method of Section 3 naturally arises when considering discretizing the geometric theory of the incompressible Euler equation in a structure-preserving manner.

For further preliminary material regarding geometric mechanics, we refer the readers to Appendix A, and for a brief exposition on exterior calculus we refer the readers to [Wang et al. 2023; Yin et al. 2023; Crane et al. 2013a].

4.1 Background

In geometric mechanics, the incompressible Euler equation is formulated as a Hamiltonian system.

The phase space of the Hamiltonian system is given by the dual space $\mathfrak{X}_{\text{div}}^*(W)$ of the space $\mathfrak{X}_{\text{div}}(W)$ of divergence-free vector fields. Note that the dual space of any Lie algebra is equipped with a natural Poisson bracket, called the **Lie–Poisson bracket** (Section A.5). Since $\mathfrak{X}_{\text{div}}(W)$ is a Lie algebra, with the standard Lie bracket for vector fields, our phase space $\mathfrak{X}_{\text{div}}^*(W)$ is equipped with a Poisson bracket. The purpose of the Poisson bracket is that it can turn any smooth function defined over the phase space, called a Hamiltonian, into a vector field that describes a Hamiltonian dynamical system on the phase space. The incompressible Euler equation is the Hamiltonian system whose phase space is $\mathfrak{X}_{\text{div}}^*(W)$ with the Lie–Poisson bracket, and whose Hamiltonian is the kinetic energy of each state in $\mathfrak{X}_{\text{div}}^*(W)$.

The fact that the incompressible Euler flow is a Hamiltonian system using the Lie–Poisson bracket structure has some significant implications. A dual Lie algebra is foliated into lower dimensional **coadjoint orbits** (see Definition A.12), and any Hamiltonian vector field on a dual Lie algebra is tangential to each coadjoint orbit and therefore the flow preserves the coadjoint orbits. In particular, a state in $\mathfrak{X}_{\text{div}}^*(W)$ following the incompressible Euler flow must stay on the same coadjoint orbit in $\mathfrak{X}_{\text{div}}^*(W)$. This conservation of coadjoint orbit, besides energy conservation, is an important structure that we aim to preserve in a structure-preserving discretization.

Let us unpack the above Hamiltonian description with concrete expressions in terms of fluid mechanical variables.

4.1.1 Dual space of divergence free vector fields.

Proposition 4.1. $\mathfrak{X}_{\text{div}}^*(W) = \Omega^1(W)/d\Omega^0(W)$.

PROOF. Appendix C.4. \square

That is, each element of $\mathfrak{X}_{\text{div}}^*(W)$ is an equivalence class of 1-forms, where two 1-forms $\eta, \eta' \in \Omega^1(W)$ are defined to be equivalent if their difference is exact $\eta - \eta' \in \text{im}(d)$. Recall the usual interpretation that a 1-form is an object to be line-integrated along oriented curves in the domain W . Here, an element $[\eta] \in \mathfrak{X}_{\text{div}}^*(W) = \Omega^1(W)/d\Omega^0(W)$ can be evaluated only along *closed* oriented curves. Therefore, we call $[\eta]$ a *circulation field*.

The dual pairing between a circulation field $[\eta] \in \mathfrak{X}_{\text{div}}^*(W)$ and a divergence-free vector field $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$ is denoted and given by

$$\langle [\eta] | \vec{v} \rangle = \int_W \langle \eta | \vec{v} \rangle d\mu \quad (22)$$

where $\langle \cdot | \cdot \rangle$ is the pointwise dual pairing between a 1-form and a vector. One can verify that the right-hand side expression of (22) is independent of the representative $\eta \in [\eta]$ for the equivalence class $[\eta]$. The dual pairing expression is relevant when we perform calculus of variations. For each functional $F: \mathfrak{X}_{\text{div}}^*(W) \rightarrow \mathbb{R}$, the



Fig. 10. Two nozzles shooting out inks of different colors collide with one another. This highly turbulent behavior produces complex vortical structures that are captured using our method (CO-FLIP) at a low resolution of $64 \times 128 \times 64$.

variation $(\delta F/\delta[\eta])$ is a divergence-free vector field $(\delta F/\delta[\eta]) \in \mathfrak{X}_{\text{div}}(W)$ given so that

$$\frac{d}{d\varepsilon} \Big|_{\varepsilon=0} F([\eta] + \varepsilon[\dot{\eta}]) = \left\langle \left[\dot{\eta} \right] \left| \frac{\delta F}{\delta[\eta]} \right. \right\rangle \quad \text{for all } [\dot{\eta}] \in \mathfrak{X}_{\text{div}}^*(W). \quad (23)$$

Note that the L^2 inner product structure $\langle \langle \cdot, \cdot \rangle \rangle = \int \langle \cdot, \cdot \rangle d\mu$ on $\mathfrak{X}_{\text{div}}(W)$ induces an isomorphism between $\mathfrak{X}_{\text{div}}(W)$ and $\mathfrak{X}_{\text{div}}^*(W)$

$$b_{\mathfrak{X}_{\text{div}}}: \mathfrak{X}_{\text{div}}(W) \xrightarrow{\text{linear } \cong} \mathfrak{X}_{\text{div}}^*(W), \quad b_{\mathfrak{X}_{\text{div}}}\vec{u} = [\vec{u}^b] \quad (24)$$

where $(\cdot)^b$ is the pointwise flat operator that converts vectors to covectors using the metric on W , so that we have $\langle \langle \vec{u}, \vec{v} \rangle \rangle = \langle \langle b_{\mathfrak{X}_{\text{div}}}\vec{u} | \vec{v} \rangle \rangle$ for all $\vec{u}, \vec{v} \in \mathfrak{X}_{\text{div}}(W)$. The inverse map

$$\sharp_{\mathfrak{X}_{\text{div}}} := b_{\mathfrak{X}_{\text{div}}}^{-1}: \mathfrak{X}_{\text{div}}^*(W) \xrightarrow{\text{linear}} \mathfrak{X}_{\text{div}}(W) \quad (25)$$

is given by that $\sharp_{\mathfrak{X}_{\text{div}}}[\eta]$ is the unique divergence free vector field $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$ such that $[\vec{v}^b] = [\eta]$. Explicitly, this \vec{v} is the pressure projection of η^\sharp of any representative $\eta \in [\eta]$.

4.1.2 Coadjoint orbits in $\mathfrak{X}_{\text{div}}^*(W)$.

Proposition 4.2. *The Hamiltonian flow on $\mathfrak{X}_{\text{div}}^*(W)$ for any Hamiltonian function $H: \mathfrak{X}_{\text{div}}^*(W) \rightarrow \mathbb{R}$ is given by*

$$\frac{\partial}{\partial t} [\eta] + \mathcal{L}_{\frac{\delta H}{\delta [\eta]}} [\eta] = [0]. \quad (26)$$

PROOF. Appendix C.5. □

Every Hamiltonian flow will lie on a coadjoint orbit (Definition A.12). The next proposition characterizes the coadjoint orbits in terms of circulation fields.

Proposition 4.3. *Two states $[\eta_0], [\eta_1] \in \mathfrak{X}_{\text{div}}^*(W)$ lie on the same coadjoint orbit if and only if there exists a volume-preserving map $\varphi: W \rightarrow W$ (isotopic to the identity map) so that $[\varphi^* \eta_1] = [\eta_0]$. Here φ^* denotes the pullback operator.*

PROOF. Appendix C.6 □

In other words, two circulation data share the same coadjoint orbit if and only if they are related by a volume-preserving transportation with Kelvin circulation preservation.

The significance about this proposition is that the Kelvin circulation theorem, known to be associated to the qualitative statement of “vorticity conservation” in Euler flows, is now concretely characterized as that the state should lie on a submanifold, being the coadjoint orbit, which is a structure solely determined by the Poisson structure equipped in every dual Lie algebra (Appendix A.4).

Furthermore, we can quantitatively detect whether a family of states lie in the same coadjoint orbit. A Casimir function (Definition A.10) is a function on $\mathfrak{X}_{\text{div}}^*(W)$ that is constant along each coadjoint orbit (using Proposition A.9). Elements on the same coadjoint orbit will take the same value for each Casimir function. For 2D fluids ($n = 2$) a basis for the Casimir functions are the p -th moment of the vorticity function

$$\mathcal{W}_p([\eta]) := \int_W w^p d\mu, \quad w = *(d\eta). \quad (27)$$

In 3D the regular Casimir is the helicity

$$\mathcal{H}([\eta]) := \int_W \eta \wedge \omega, \quad \omega = d\eta. \quad (28)$$

In particular, these Casimir functions are conserved for any Hamiltonian flow (26) on $\mathfrak{X}_{\text{div}}^*(W)$.

One example for a Hamiltonian flow on $\mathfrak{X}_{\text{div}}^*(W)$ is the incompressible Euler equation:

Proposition 4.4. *The incompressible Euler equation is the Hamiltonian flow $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{Euler}})$ for the Hamiltonian function*

$$H_{\text{Euler}}([\eta]) = \frac{1}{2} \langle \#_{\mathfrak{X}_{\text{div}}} [\eta], \#_{\mathfrak{X}_{\text{div}}} [\eta] \rangle. \quad (29)$$

PROOF. Appendix C.7 □

Next, as we consider a discretized Euler equation, we modify the Hamiltonian function to a lower dimensional function, but we keep the Poisson structure on $\mathfrak{X}_{\text{div}}^*(W)$ the same. By doing so, the modified equation will still stay on a coadjoint orbit in the true phase space $\mathfrak{X}_{\text{div}}^*(W)$ and therefore obey Kelvin’s circulation conservation.

4.2 Discrete Euler Flow

Our goal is to develop a finite dimensional (discrete) analog of the Hamiltonian system $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{Euler}})$.

Our discretization starts with a finite dimensional subspace in the space of divergence-free vector fields given by a divergence-consistent interpolation

$$I: \mathfrak{B} \hookrightarrow \mathfrak{X}(W), \quad (30)$$

$$I: \mathfrak{B}_{\text{div}} \hookrightarrow \mathfrak{X}_{\text{div}}(W). \quad (31)$$

Let us go towards replacing $\mathfrak{X}_{\text{div}}(W)$ with this finite dimensional representation $\mathfrak{B}_{\text{div}}$. Like the continuous theory, the dual space $\mathfrak{B}_{\text{div}}^*$ is the quotient space $\mathfrak{B}_{\text{div}}^* = \mathfrak{B}^*/\text{im}(\mathbf{d}^\top)$ where \mathbf{d} is the discrete divergence operator defining $\mathfrak{B}_{\text{div}} = \ker(\mathbf{d})$. As described in (11) the embedding induces an inner product structure on $\mathfrak{B}_{\text{div}}$ denoted by $\langle \mathbf{f}, \mathbf{g} \rangle_{\mathfrak{B}} = \langle I(\mathbf{f}), I(\mathbf{g}) \rangle = \mathbf{f}^\top \star \mathbf{g}$. Similar to the continuous theory the inner product \star induces a map $\#_{\mathfrak{B}_{\text{div}}}: \mathfrak{B}_{\text{div}}^* \rightarrow \mathfrak{B}_{\text{div}}$ given by $\#_{\mathfrak{B}_{\text{div}}} = \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1}$, the pressure projection of the metric dual of any representative in $\mathfrak{B}^*/\text{im}(\mathbf{d}^\top)$. Using the metric we define the *discrete* Hamiltonian function analogous to the Hamiltonian for the Euler equation:

Definition 4.1. *The I -discrete Hamiltonian for Euler fluid is given by*

$$H_{\text{D}}: \mathfrak{B}_{\text{div}}^* \rightarrow \mathbb{R}, \quad H_{\text{D}}([\mathbf{c}]) := \frac{1}{2} \langle \#_{\mathfrak{B}_{\text{div}}} \mathbf{c}, \#_{\mathfrak{B}_{\text{div}}} \mathbf{c} \rangle_{\mathfrak{B}}. \quad (32)$$

Here $[\mathbf{c}]$ represents a discrete circulation data.

4.2.1 Lack of Poisson Structure for $\mathfrak{B}_{\text{div}}^*$. To construct a discrete analogue of the Hamiltonian system $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{Euler}})$, it is tempting to consider a system $(\mathfrak{B}_{\text{div}}^*, H_{\text{D}})$ where the phase space is $\mathfrak{B}_{\text{div}}^*$, on which the discrete Hamiltonian H_{D} is given. Unfortunately, unlike the continuous theory, the space $\mathfrak{B}_{\text{div}}$ of divergence-free discrete vector fields is in general not a Lie algebra like $\mathfrak{X}_{\text{div}}$. It is unlikely that there exists a Lie algebraically closed set of vector fields suitable for fluid simulation, given that, at least in 2D, all Lie algebras of vector fields have been completely classified [González-López et al. 1992]. As a consequence, in general, there is no natural Poisson structure for $\mathfrak{B}_{\text{div}}^*$ to be a phase space. The core problem that the space of computational velocities not being Lie algebraically closed is referred to as that the velocity is *non-holonomically constrained* [Mullen et al. 2009; Pavlov et al. 2011; Liu et al. 2015]. We discuss the non-holonomicity in the literature of structure-preserving fluid simulation in Appendix B.

4.2.2 Our dynamical system. The lack of Poisson structure for $\mathfrak{B}_{\text{div}}^*$ does not prevent us from considering a Hamiltonian system on the larger space $\mathfrak{X}_{\text{div}}^*(W)$, the original dual Lie algebra of the continuous theory. The adjoint of the embedding (31) is a map of type

$$I^\top: \mathfrak{X}_{\text{div}}^*(W) \xrightarrow{\text{linear}} \mathfrak{B}_{\text{div}}^*. \quad (33)$$

This allows us to pullback the discrete Hamiltonian function (32) from $\mathfrak{B}_{\text{div}}^*$ to $\mathfrak{X}_{\text{div}}^*(W)$ by composition

$$(H_{\text{D}} \circ I^\top): \mathfrak{X}_{\text{div}}^*(W) \rightarrow \mathbb{R}. \quad (34)$$

Definition 4.2. *The I -discrete Euler flow is given by the Hamiltonian system $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{D}} \circ I^\top)$.*

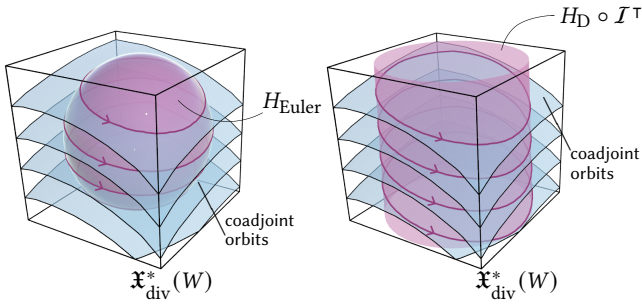


Fig. 11. Left: The Hamiltonian for Euler fluid H_{Euler} on $\mathfrak{X}_{\text{div}}^*(W)$. Right: The pullback of I -discrete Hamiltonian for Euler fluid $H_{\text{D}} \circ I^{\top}$ on $\mathfrak{X}_{\text{div}}^*(W)$. Note that Euler flow and the I -discrete Euler flow stay on the same coadjoint orbits (illustrated in blue). Their intersections with the coadjoint orbits of $\mathfrak{X}_{\text{div}}^*(W)$ are marked with pink lines with arrows.

Theorem 4.1. . The equation of motion for the I -discrete Euler flow on $\mathfrak{X}_{\text{div}}^*(W)$ is given by

$$\frac{\partial}{\partial t} [\eta] + \mathcal{L}_{\vec{v}} [\eta] = [0], \quad \text{where} \quad (35a)$$

$$\vec{v} = I \#_{\mathfrak{B}_{\text{div}}} I^{\top} [\eta] = I \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top} \eta. \quad (35b)$$

PROOF. Appendix C.8. \square

Corollary 4.1. The I -discrete Euler flow preserves the energy $H_{\text{D}} \circ I^{\top}$ and the coadjoint orbits in $\mathfrak{X}_{\text{div}}^*(W)$.

The I -discrete Euler flow only modifies the continuous Euler flow by changing the Hamiltonian function from H_{Euler} to $H_{\text{D}} \circ I^{\top}$, a function of “less complexity” filtered through a projection I^{\top} to a finite dimensional space $\mathfrak{B}_{\text{div}}^*$. However, the I -discrete Euler equation expressed for elements $[\eta](t) \in \mathfrak{X}_{\text{div}}^*(W)$ is still a continuous PDE (35) (see Fig. 11). How can we compute solutions to such PDEs? Next, we describe a technique using momentum maps to answer this question. The procedure leads to an interpretation of the I -discrete Euler flow in a manner closer to a Lagrangian–Eulerian hybrid simulation algorithm.

4.3 Method of Characteristics using an Auxiliary Symplectic Space

We now introduce a technique that states that we can emulate the Hamiltonian system $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{D}} \circ I^{\top})$ on an auxiliary symplectic space (See A.8 on which the dynamical system may be easier to compute). A special case of this technique is the method of characteristics for advective PDEs, leading to the hybrid formulation. Readers less familiar with Symplectic Manifolds could refer to the appendix Section A.3 for a brief introduction.

Note that $\mathfrak{X}_{\text{div}}(W)$ is the Lie algebra (tangent space at the identity) of the Lie group of volume-preserving diffeomorphisms on W

$$\text{SDiff}(W) = \{\varphi \in C^1(W; W) \mid \text{vol } U = \text{vol } \varphi(U) \text{ for all } U \subset W\}$$

Consider any symplectic manifold (Σ, σ) that is acted symplectomorphically by the Lie group $\text{SDiff}(W)$. That is, there is a group action, which we call **advection**, on Σ

$$\text{Adv}: \text{SDiff}(W) \times \Sigma \rightarrow \Sigma \quad (36)$$

that preserves the symplectic 2-form $\sigma \in \Omega^2(\Sigma)$

$$(\text{Adv}_{\varphi})^* \sigma = \sigma, \quad \text{for all } \varphi \in \text{SDiff}(W). \quad (37)$$

Here $(\text{Adv}_{\varphi})^*$ is the pullback operator by the map $\text{Adv}_{\varphi}: \Sigma \rightarrow \Sigma$. By considering the infinitesimal actions of Adv , we obtain an induced Lie algebra (anti)homomorphism between spaces of vector fields

$$\text{adv}: \mathfrak{X}_{\text{div}}(W) \xrightarrow[\text{(anti)hom}]{\text{Lie alg.}} \mathfrak{X}_{\sigma}(\Sigma) \quad (38)$$

where $\mathfrak{X}_{\sigma}(\Sigma)$ is the Lie algebra of symplectic-form-preserving vector fields $\mathfrak{X}_{\sigma}(\Sigma) := \{\vec{X} \in \mathfrak{X}(\Sigma) \mid \mathcal{L}_{\vec{X}} \sigma = 0\}$. The symplectomorphic action of adv induces a **momentum map** (Example A.7)

$$J_{\text{adv}}: \Sigma \rightarrow \mathfrak{X}_{\text{div}}^*(W), \quad (39)$$

which, by Proposition A.11, is an (anti-)Poisson map (Definition A.7). In other words, Σ and $\mathfrak{X}_{\text{div}}^*(W)$ share the same Poisson structure through the change of variables of J_{adv} . Therefore, the Hamiltonian system $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{D}} \circ I^{\top})$ is equivalent to $(\Sigma, H_{\text{D}} \circ I^{\top} \circ J_{\text{adv}})$ in the following sense.

Definition 4.3. $H_{\Sigma} := (H_{\text{D}} \circ I^{\top} \circ J_{\text{adv}}): \Sigma \rightarrow \mathbb{R}$.

Theorem 4.2. If a path $s: [0, T] \rightarrow \Sigma$ is a solution of the Hamiltonian dynamical system (Σ, H_{Σ}) , then $(J_{\text{adv}} \circ s): [0, T] \rightarrow \mathfrak{X}_{\text{div}}^*(W)$ is a solution to the I -discrete Euler flow $(\mathfrak{X}_{\text{div}}^*(W), H_{\text{D}} \circ I^{\top})$.

PROOF. Appendix C.9. \square

In other words, we can solve the I -discrete Euler equation (35) by solving the following equation of motion for $s(t) \in \Sigma$.

Theorem 4.3. The equation of motion for the Hamiltonian system (Σ, H_{Σ}) is given by

$$\left\{ \begin{array}{l} \frac{d}{dt} s(t) = \text{adv}_{\vec{v}(t)} s(t), \\ \vec{v}(t) = I \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top} J_{\text{adv}} s(t). \end{array} \right. \quad (40a)$$

$$\vec{v}(t) = I \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top} J_{\text{adv}} s(t). \quad (40b)$$

PROOF. Appendix C.10. \square

Corollary 4.2. Any solution s to (40) has the property that $J_{\text{adv}} s(t)$ and $J_{\text{adv}} s(0)$ lie on the same coadjoint orbit in $\mathfrak{X}_{\text{div}}^*(W)$ for all t .

In the work of Marsden and Weinstein [1983], the variable $s \in \Sigma$ is called a **Clebsch variable**, and the Poisson map J_{adv} is a Clebsch representation [Chern et al. 2016, 2017; Yang et al. 2021]. We will see in the following concrete instance of Σ that s serves as a Lagrangian marker. The change of variables from (35a) to (40) is similar to changing an Eulerian coordinate to a Lagrangian coordinate in the method of characteristics. We do emphasize that the construction of (40) is more delicate than just devising a Lagrangian coordinate. Properties such as Corollary 4.2 require the advection action on Σ to be symplectomorphic, and the transfer function J_{adv} to be the derived momentum map associated to the symplectomorphic action.

4.4 A Choice for the Symplectic Space Σ

Here we describe a construction of Σ and the actions Adv , adv on Σ , followed by deriving the associated momentum map J_{adv} .

4.4.1 Flow Action on Positions. Consider the space of material positions

$$Q = \{\vec{x}: M \rightarrow W \mid \vec{x} \text{ is volume preserving}\} \quad (41)$$

where M represents the material space (Lagrangian coordinate) equipped with a volume form. One may identify $M = W$ as the fluid domain at $t = 0$. We distinguish the notation for the space M from W for clarity. The Lie group $\text{SDiff}(W)$ of volume preserving maps on W acts on Q by flowing the particle

$$\text{Adv}^Q: \text{SDiff}(W) \times Q \rightarrow Q, \quad \text{Adv}_\varphi^Q(\vec{x}) = \varphi \circ \vec{x}. \quad (42)$$

4.4.2 Lifted Action on the Cotangent Bundle. Now, neither Q is a symplectic space nor Adv^Q is a symplectomorphic action. But starting from such a group action on the position space Q , we can always lift the action to an action on the cotangent bundle $\Sigma := T^*Q$. It turns out that the cotangent bundle is symplectic, and the lifted action is symplectomorphic. Precisely, the cotangent bundle is given by

$$\Sigma = T^*Q = \left\{ (\vec{x}, \vec{u}) \mid \vec{x}: M \rightarrow W, \vec{u}: M \rightarrow T^*W, \vec{u}_p \in T_{\vec{x}_p}^*W \right\}. \quad (43)$$

That is, each element of the cotangent bundle is an assignment of particle position $\vec{x}_p \in W$, and an assignment of a covector $\vec{u}_p \in T_{\vec{x}_p}^*W$ based at \vec{x}_p for each $p \in M$. The cotangent bundle has a canonical symplectic 2-form σ given by the exterior derivative $d\vartheta$ of the **Liouville 1-form** $\vartheta \in \Omega^1(\Sigma)$ given by

$$\vartheta_{(\vec{x}, \vec{u})}[\dot{(\vec{x}, \vec{u})}] = \int_M \langle \vec{u} | \dot{\vec{x}} \rangle d\mu, \quad \text{for } (\vec{x}, \vec{u}) \in T_{(\vec{x}, \vec{u})}\Sigma. \quad (44)$$

The lifted action $\text{Adv}: \text{SDiff}(W) \times \Sigma \rightarrow \Sigma$ from Adv^Q is given by

$$\text{Adv}_\varphi(\vec{x}, \vec{u}) = \left(\varphi \circ \vec{x}, (d\varphi^{-1})^* \vec{u} \right) \quad (45)$$

where $(d\varphi^{-1})^*$ is the adjoint of the inverse of the flow map φ . One may check that Adv preserves the symplectic structure by checking that it preserves the Liouville 1-form $\text{Adv}_\varphi^* \vartheta = \vartheta$ and that the exterior derivative and pullbacks commute.

By taking the direction derivative of (45) with respect to φ at the identity along a Lie algebra element $\vec{v} \in \mathfrak{X}_{\text{div}}$, we obtain the induced infinitesimal action

$$\text{adv}_{\vec{v}}(\vec{x}, \vec{u}) = \left(\vec{v}|_{\vec{x}}, -(\nabla \vec{v})_{\vec{x}}^T \vec{u} \right) \in T_{(\vec{x}, \vec{u})}\Sigma. \quad (46)$$

4.4.3 The Momentum Map. The momentum map $J_{\text{adv}}: \Sigma \rightarrow \mathfrak{X}_{\text{div}}^*(W)$ associated to this symplectomorphic action can be expressed explicitly using the Liouville form. At each $(\vec{x}, \vec{u}) \in \Sigma$, the covector field $J_{\text{adv}}(\vec{x}, \vec{u}) \in \mathfrak{X}_{\text{div}}^*(W)$, when paired with an arbitrary $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$, is given by

$$\langle J_{\text{adv}}(\vec{x}, \vec{u}) | \vec{v} \rangle = \vartheta_{(\vec{x}, \vec{u})}[\text{adv}_{\vec{v}}(\vec{x}, \vec{u})] = \int_M \langle \vec{u} | \vec{v} \circ \vec{x} \rangle d\mu. \quad (47)$$

From this expression, we conclude that

$$J_{\text{adv}}(\vec{x}, \vec{u}) = [\eta] \in \mathfrak{X}_{\text{div}}^*(W) = \Omega^1(W)/d\Omega^0(W) \quad (48)$$

where η is uniquely defined by that $\vec{u} = \eta \circ \vec{x}$. See Fig. 12 for illustration.

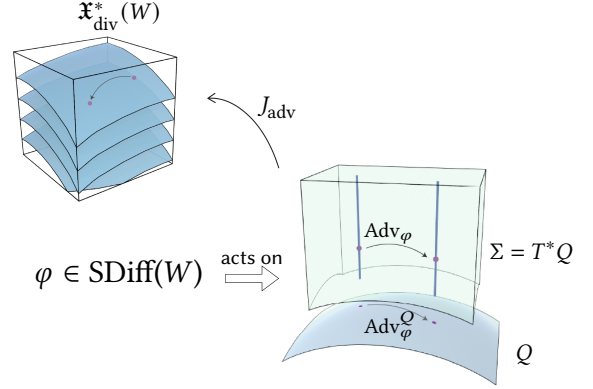


Fig. 12. Mapping between $\Sigma = T^*Q$, $\text{SDiff}(W)$, and $\mathfrak{X}_{\text{div}}^*(W)$.

4.5 Arrival at the CO-FLIP Equations

The CO-FLIP equation arises by substituting the setup of Section 4.4 in (40). To see it, let us unpack one final operator, $\star^{-1}I^\top$, in (40b).

4.5.1 The Map $\star^{-1}I^\top J_{\text{adv}}$. In (40b) the momentum map J_{adv} is used only in conjunction with subsequent operator

$$\star^{-1}I^\top: \Omega^1(W) \rightarrow \mathfrak{B}. \quad (49)$$

Here I^\top is the adjoint of $I: \mathfrak{B} \rightarrow \mathfrak{X}(W)$. The map $\star^{-1}I^\top$ becomes well-defined on $\mathfrak{X}_{\text{div}}^*(W) = \Omega^1(W)/d\Omega^0(W)$ if it is always followed by the pressure projection $\mathbb{P}_{\mathfrak{B}_{\text{div}}}$, which is indeed the case in (40b).

Remark 4.1. In fact, $I^\top J_{\text{adv}}: \Sigma \rightarrow \mathfrak{B}_{\text{div}}^*$, which is the map from the Lagrangian data Σ to the grid, is the momentum map associated to the action $(\text{adv} \circ I): \mathfrak{B}_{\text{div}} \rightarrow \mathfrak{X}_\sigma(\Sigma)$.

The following lemmas characterizes $\star^{-1}I^\top$ and $\star^{-1}I^\top J_{\text{adv}}$ as the pseudoinverse of I .

Lemma 4.1. The map $\star^{-1}I^\top: \Omega^1(W) \rightarrow \mathfrak{B}$ is the pseudoinverse of $b \circ I: \mathfrak{B} \rightarrow \Omega^1(W)$ with respect to the L^2 structure on $\mathfrak{X}(W)$ and its dual $\Omega^1(W)$:

$$\star^{-1}I^\top \eta = \underset{f \in \mathfrak{B}}{\text{argmin}} \int_W |I(f) - \eta^\#|^2 d\mu. \quad (50)$$

Lemma 4.2. The map $\star^{-1}I^\top J_{\text{adv}}: \Sigma \rightarrow \mathfrak{B}$ is given by

$$\star^{-1}I^\top J_{\text{adv}}(\vec{x}, \vec{u}) = \underset{f \in \mathfrak{B}}{\text{argmin}} \int_M |I(f) \circ \vec{x} - \vec{u}|^2 d\mu. \quad (51)$$

That is, $\star^{-1}I^\top J_{\text{adv}}$ is the L^2 -projection of a vector field, represented by the position-impulse field (\vec{x}, \vec{u}) , to the subspace \mathfrak{B} of interpolatable vector fields.

PROOF. Appendix C.11. □

Definition 4.4. We define $I^+ := \star^{-1}I^\top$ and with slight abuse of notation $I^+ := \star^{-1}I^\top J_{\text{adv}}$. In both cases, $I^+: (\cdot) \rightarrow \mathfrak{B}$ is an operator that L^2 -orthogonally projects a smooth vector field to the subspace \mathfrak{B} of interpolatable vector fields. The input vector field is either represented as a vector or covector (through identification of \flat/\sharp) on either the world W or material M coordinate (through identification of J_{adv})

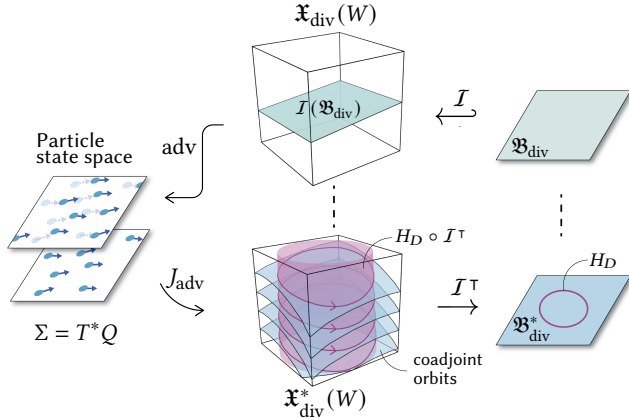


Fig. 13. Illustration of our CO-FLIP dynamical system solving the \mathcal{I} -discrete Euler equation; with finite particles approximating the phase space.

4.5.2 CO-FLIP on Continuous Material Space. Combining (40), (46), and (51), we obtain an equation of motion for $\vec{x}(t): M \rightarrow W$ and $\vec{u}(t): M \rightarrow T^*W$ (with $\vec{u}_p \in T^*_{\vec{x}_p} W$)

$$\begin{cases} \frac{d}{dt} \vec{x}_p(t) = \vec{v}|_{\vec{x}_p(t)} & p \in M & (52a) \\ \frac{d}{dt} \vec{u}_p(t) = -(\nabla \vec{v}|_{\vec{x}_p(t)})^\top \vec{u}_p(t) & p \in M & (52b) \\ \vec{v} = \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ (\vec{x}, \vec{u}). & & (52c) \end{cases}$$

4.5.3 Time Discretization of CO-FLIP on Continuous Material Space. Following the trapezoidal rule (19) we consider the time discretization of (52) as the following Lie group integrator:

$$\begin{cases} (\vec{x}, \vec{u})^{(n+1)} = \text{Adv}_{\vec{v}}^{\Delta t} (\vec{x}, \vec{u})^{(n)} & (53a) \\ \vec{v} = \mathcal{I} \left(\frac{1}{2} (\mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ (\vec{x}, \vec{u})^{(n)} + \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ (\vec{x}, \vec{u})^{(n+1)}) \right) & (53b) \end{cases}$$

which is a map $(\vec{x}, \vec{u})^{(n)} \mapsto (\vec{x}, \vec{u})^{(n+1)}$ on Σ . Here,

$$\text{Adv}_{\vec{v}}^{\Delta t} := \exp(\Delta t \text{adv}_{\vec{v}}) = \text{Adv}_{\varphi(\Delta t)} \quad (54)$$

is the symplectomorphic group action (45) where $\varphi(\Delta t) \in \text{SDiff}(W)$ is the flow map generated by \vec{v} over Δt time span.

One may also strip away the instantiation of the symplectic space Σ in Section 4.4, and observe that the time discretization (53) is a special case of the following Lie group integrator for (40):

$$\begin{cases} s^{(n+1)} = \exp(\Delta t \text{adv}_{\vec{v}}) s^{(n)} = \text{Adv}_{\varphi(\Delta t)} s^{(n)} & (55a) \\ \vec{v} = \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ \left(\frac{1}{2} (J_{\text{adv}} s^{(n)} + J_{\text{adv}} s^{(n+1)}) \right) & (55b) \end{cases}$$

which is a map $s^{(n)} \mapsto s^{(n+1)}$ generating a discrete sequence on Σ .

4.5.4 CO-FLIP on Finite Particles. In (52), the advection component (52a) and (52b) are ODEs that hold for $p \in \mathcal{P}$ for any subset $\mathcal{P} \subset M$. Let $\mathcal{P} \subset M$ be a finite set of points, representing computational particles. The restrictions $(\vec{x}, \vec{u}): \mathcal{P} \rightarrow T^*W$ of $(\vec{x}, \vec{u}): M \rightarrow T^*W$ are samples of the function $(\vec{x}, \vec{u}): M \rightarrow T^*W$ on the fixed set of points \mathcal{P} . The only modification on the system (52) when restricting

the material space M to the finite set \mathcal{P} is replacing $\mathcal{I}^+ = \star^{-1} \mathcal{I}^\top J_{\text{div}}$ by the following approximation of (51)

$$\hat{\mathcal{I}}^+ (\vec{x}, \vec{u}) := \underset{f \in \mathfrak{B}}{\text{argmin}} \sum_{p \in \mathcal{P}} |\mathcal{I}(f) \circ \vec{x}_p - \vec{u}_p^\#|^2 \mu_p. \quad (56)$$

The weight μ_p represents the particle volume. This modification results in the finite dimensional ODE system (14) and the algorithm (19) in Section 3, as illustrated in Fig. 13.

In the discretization from (51) to (56), the L^2 -integral $\int_M |\cdot|^2 d\mu$ is approximated by a finite sum $\sum_{p \in \mathcal{P}} |(\cdot)_p|^2 \mu_p$. Other linear algebraic properties about \mathcal{I}^+ remain true for $\hat{\mathcal{I}}^+$.

Theorem 4.4. *Suppose $\mathcal{P} \subset M$ is sufficiently dense so that the minimization (56) is unique. Then $\hat{\mathcal{I}}^+ (\vec{x}, \vec{u})$ is a projection on the subspace \mathfrak{B} from a smooth vector field represented by (\vec{x}, \vec{u}) . In particular, $\hat{\mathcal{I}}^+$ is an exact left-inverse of \mathcal{I} .*

PROOF. See Theorem 3.2. \square

Note that the projection $\hat{\mathcal{I}}^+$ is an orthogonal projection not with respect to $\int_M |\cdot|^2 d\mu$ but with respect to a perturbed metric $\sum_{p \in \mathcal{P}} |(\cdot)_p|^2 \mu_p$. In the infinite limit of $\mathcal{P} \rightarrow M$, the projection $\hat{\mathcal{I}}^+$ to \mathfrak{B} is progressively closer to being L^2 -orthogonal.

4.6 Properties of CO-FLIP

Corollary 4.3. *Solutions $(\vec{x}(t), \vec{u}(t))$ to the CO-FLIP equation (52) on M are \mathcal{I} -discrete Euler flow. In particular it preserves the energy $(H_D \circ \mathcal{I}^\top \circ J_{\text{adv}})(\vec{x}(t), \vec{u}(t))$ and $J_{\text{adv}}(\vec{x}(t), \vec{u}(t))$ stays on a coadjoint orbit.*

Theorem 4.5. *The integrator $s^{(n)} \rightarrow s^{(n+1)}$ by (55) exactly preserves the energy $\mathcal{E}(s) := H_D \circ \mathcal{I}^\top \circ J_{\text{adv}}(s)$ and exactly preserves the coadjoint orbit for $J_{\text{div}}(s)$.*

PROOF. Appendix C.12 \square

Corollary 4.4. *The integrator (53) exactly preserves the energy and the coadjoint orbit.*

Theorem 4.6. *The finite dimensional CO-FLIP ODE system (14) and the integrator (19) preserves the coadjoint orbit.*

PROOF. Appendix C.13 \square

The systems (14) and (19) preserves the coadjoint orbit. However the energy can drift depending on the approximation of the integral $\int_M |\cdot|^2 d\mu$ by the summation $\sum_{p \in \mathcal{P}} |(\cdot)_p|^2 \mu_p$, as described in Section 3.5.

Theorem 4.7. *The method (21) exactly preserves the coadjoint orbit and the energy.*

PROOF. Appendix C.14. \square

5 THEORY PART 2: MIMETIC INTERPOLATION

The theory of Section 4 and the resulting method of Section 3 rely heavily on a divergence-free interpolation scheme $\mathcal{I}: \mathfrak{B} \hookrightarrow \mathfrak{X}(W)$, $\mathcal{I}: \mathfrak{B}_{\text{div}} \hookrightarrow \mathfrak{X}_{\text{div}}(W)$. In this section, we describe the properties and construction of \mathcal{I} using B-splines.

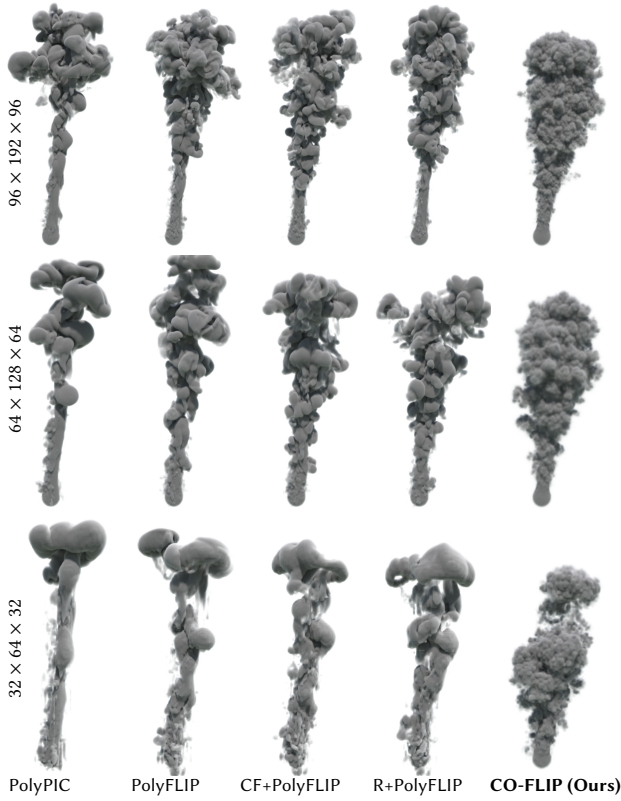


Fig. 14. Smoke plume 3D. At a fixed CFL number, with increasing spatiotemporal resolution, the vortical structures present in the smoke get denser and more abundant. Note that our method produces increased vortical features, reminiscing the fractal features seen in pyroclastic clouds. This behavior is present even at a very low resolution of $32 \times 64 \times 32$.

Highlights. A special quality of our divergence-free interpolation is that the evaluation is explicit and local. This is in contrast to the recent divergence-free interpolation methods [Chang et al. 2022; Lyu et al. 2024] introduced in fluid simulations in computer graphics that require a global streamfunction integration. Moreover, our interpolation scheme is accompanied with a family of interpolation schemes that are curl-consistent and gradient-consistent. These interpolations are useful for implementation details such as building various versions of pressure solvers.

Overview. We employ a family of **mimetic interpolation** schemes, widely studied in isogeometric analysis and finite element exterior calculus. These interpolation methods \mathcal{I}_k interpolate differential k forms and commute with the exterior derivative operators. Our divergence-free interpolation is the interpolation \mathcal{I}_{n-1} for $(n-1)$ -forms (flux forms), followed by the canonical conversion between a flux form and a vector field $i_{\vec{v}}(d\mu) \in \Omega^{n-1}(W) \leftrightarrow \vec{v} \in \mathfrak{X}(W)$. Closed flux form translates to divergence-free vector fields.

5.1 Mimetic interpolation

Let

$$C^0 \xrightarrow{d_0} C^1 \xrightarrow{d_1} C^2 \xrightarrow{d_2} \dots \xrightarrow{d_{n-1}} C^n \quad (57)$$

be a real-coefficient cochain complex, where each C^k is a finite dimensional real vector space.² Each cochain space C^k represents the space of discrete k -forms, and the operators d_k that satisfy the cochain condition $d_{k+1}d_k = 0$ are discrete exterior derivatives.

Definition 5.1 (Mimetic interpolation). A family of interpolation schemes $\mathcal{I}_k : C^k \rightarrow \Omega^k(W)$ is mimetic if it commutes with the exterior derivatives:

$$d_k(\mathcal{I}_k \mathbf{a}) = \mathcal{I}_{k+1}(d_k \mathbf{a}) \quad \text{for all } \mathbf{a} \in C^k, \text{ for all } k = 0, \dots, n-1. \quad (58)$$

Here $d_k : \Omega^k(W) \rightarrow \Omega^{k+1}(W)$ is the exterior derivative in the smooth theory.

Diagrammatically, a mimetic interpolation relates a finite dimensional cochain complex to the de Rham complex in the following commutative diagram

$$\begin{array}{ccccccc} C^0 & \xrightarrow{d_0} & C^1 & \xrightarrow{d_1} & C^2 & \xrightarrow{d_2} & \dots & \xrightarrow{d_{n-1}} & C^n \\ \downarrow \mathcal{I}_0 & & \downarrow \mathcal{I}_1 & & \downarrow \mathcal{I}_2 & & & & \downarrow \mathcal{I}_n \\ \Omega^0 & \xrightarrow{d_0} & \Omega^1 & \xrightarrow{d_1} & \Omega^2 & \xrightarrow{d_2} & \dots & \xrightarrow{d_{n-1}} & \Omega^n \end{array} \quad (59)$$

5.1.1 Basis Representation. Let $\beta_i^k \in \Omega^k(W)$, $i = 1, \dots, \dim(C^k)$, be a basis k -forms for $\mathcal{I}_k(C^k) \subset \Omega^k(W)$ so that \mathcal{I}_k can be expressed as a map that takes in a list of coefficients $\mathbf{a} = (a_1, \dots, a_{\dim(C^k)})$, representing an element of C^k , to a k -form $\mathcal{I}_k(\mathbf{a}) \in \Omega^k(W)$ by

$$\mathcal{I}_k(\mathbf{a}) = \sum_{i=1}^{\dim(C^k)} a_i \beta_i^k. \quad (60)$$

5.1.2 Galerkin Hodge Stars. The mimetic interpolations $\mathcal{I}_k : C^k \rightarrow \Omega^k(W)$ induces a metric on C^k from the L^2 structure on $\Omega^k(W)$. The L^2 inner product between $\omega_1, \omega_2 \in \Omega^k(W)$ is given by

$$\langle\langle \omega_1, \omega_2 \rangle\rangle_{\Omega^k(W)} = \int_W \omega_1 \wedge * \omega_2, \quad (61)$$

where $*$ is the Hodge star in the smooth theory. The induced inner product structure on C^k is a positive definite self-adjoint operator $\star_k : C^k \rightarrow C^{k*}$, where C^{k*} is the dual space of C^k , so that

$$\mathbf{a}^\top \star_k \mathbf{b} = \langle \mathbf{a} | \star_k \mathbf{b} \rangle = \langle\langle \mathcal{I}_k \mathbf{a}, \mathcal{I}_k \mathbf{b} \rangle\rangle_{\Omega^k(W)}, \quad \forall \mathbf{a}, \mathbf{b} \in C^k. \quad (62)$$

In basis expression, \star_k is the inner product matrix with entries

$$(\star_k)_{ij} = \langle\langle \beta_i^k, \beta_j^k \rangle\rangle_{\Omega^k(W)}. \quad (63)$$

5.1.3 General Construction. There are general constructions for mimetic interpolations starting with a scalar function interpolant $\mathcal{I}_0 : C^0 \rightarrow \Omega^0(W)$. An interpolant $\mathcal{I}_0 : C^0 \rightarrow \Omega^0(W)$ is said to satisfy **partition of unity** if there exists $\mathbf{a} \in C^0$ such that $\mathcal{I}_0 \mathbf{a} = 1$. Hiemstra et al. [2014] showed that any interpolant \mathcal{I}_0 which satisfies partition of unity would create a series of interpolants \mathcal{I}_k for all k -forms that satisfies the mimetic property. As such, there are many families of mimetic interpolations, such as the ones based on B-splines [Buffa

²Do not confuse the cochain space C^k with the space C^k of k -time continuously differentiable functions.

et al. 2011], Lagrange interpolating polynomials [Gerritsma 2010], and radial basis functions [Narcowich and Ward 1994; Drake et al. 2021].

We take B-splines as our choice of mimetic interpolation, similar to the works of [Buffa et al. 2011]. B-splines are piece-wise polynomial, local and compactly supported, non-negative, and are common in the MPM works in computer graphics [Jiang et al. 2016].

5.2 Univariate B-Spline Mimetic Interpolation

Mimetic interpolation for B-splines in multidimensions are built from mimetic univariate B-splines. We first give an overview of basic properties of univariate B-splines. See also [Hughes et al. 2005] for detailed theory of B-splines.

Basis splines (B-splines) are piecewise polynomial functions of a given degree with maximal smoothness and minimal support [Curry and Schoenberg 1947, 1966]. On a 1-dimensional parameter domain, consider a knot sequence $\xi_1 \leq \xi_2 \leq \dots$ partitioning the domain into pieces separated by the knots.

Definition 5.2 (B-splines). *The i -th degree- p B-spline B_i^p is a piecewise degree- p polynomial supported over interval $[\xi_i, \xi_{i+p+1}]$ defined recursively by the Cox-de Boor formula [Cox 1972; De Boor 1972]*

$$B_i^p(t) := \chi_i^p(t) B_{i+1}^{p-1}(t) + (1 - \chi_{i+1}^p(t)) B_{i+1}^{p-1}(t), \quad p \geq 1, \quad (64a)$$

$$B_i^0(t) := \begin{cases} 1, & t \in [\xi_i, \xi_{i+1}), \\ 0, & \text{otherwise,} \end{cases} \quad \chi_i^p(t) := \frac{t - \xi_i}{\xi_{i+p} - \xi_i}. \quad (64b)$$

When we use B-splines with a maximal polynomial degree p over a compact parameter domain $[0, 1]$, we set the knot set $\Xi = (\xi_1 \leq \xi_2 \leq \dots \leq \xi_{N+p+1})$ as

$$\begin{aligned} 0 &= \xi_1 = \dots = \xi_{p+1} < \xi_{p+2} < \dots \\ &\dots < \xi_N < \xi_{N+1} = \dots = \xi_{N+p+1} = 1. \end{aligned} \quad (65)$$

The number $N+p+1$ of knots correspond to the count that there are N B-splines of degree p , $\{B_1^p, \dots, B_N^p\}$, using which we represent an interpolated function from N coefficients $\mathbf{f} = (f_1, \dots, f_N) \in C^0 = \mathbb{R}^N$:

$$(\mathcal{I}_0 \mathbf{f})(t) := \sum_{i=1}^N f_i B_i^p(t). \quad (66)$$

In (65), the multiplicity of coinciding knots at the end points ensures that the boundary coefficients f_1, f_N are interpolated: $(\mathcal{I}_0 \mathbf{f})(0) = f_1$ and $(\mathcal{I}_0 \mathbf{f})(1) = f_N$.

The (exterior) derivative of a B-spline is remarkably a linear combination of lower order B-splines:

$$dB_i^p(t) = (D_i^{p-1}(t) - D_{i+1}^{p-1}(t)) dt, \quad (67)$$

where $D_i^{p-1}(t)$ is known as the Curry-Schoenberg basis, or M-splines [Curry and Schoenberg 1966], defined as follows.

Definition 5.3 (Curry-Schoenberg M-splines). *The i -th degree- $(p-1)$ Curry-Schoenberg M-spline D_i^{p-1} is a rescale of the corresponding B-spline, given by*

$$D_i^{p-1}(t) := \begin{cases} \frac{p}{\xi_{i+p} - \xi_i} B_i^p(t) & \xi_{i+p} > \xi_i \\ 0 & \xi_{i+p} = \xi_i. \end{cases} \quad (68)$$

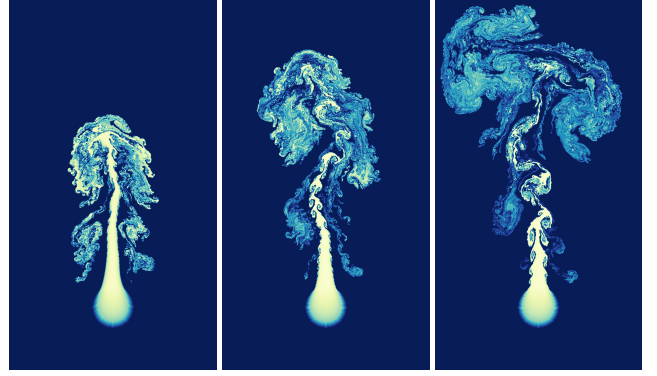


Fig. 15. Smoke plume in two dimensions. We demonstrate an abundance of vortical structures using our method (CO-FLIP), in the evolution of a buoyant material traveling upwards in a closed box.

In the setup of (65), the formula (67) holds for all $i = 1, \dots, N$, with $D_1^{p-1}(t) = D_{N+1}^{p-1}(t) = 0$.

In this one-dimensional setting, we obtain the mimetic interpolation property as we now show. Define the space C^1 of 1-cochains as \mathbb{R}^{N-1} . Let $\mathbf{d}_0: C^0 = \mathbb{R}^N \rightarrow C^1 = \mathbb{R}^{N-1}$ be the discrete differential operator given by

$$(\mathbf{d}_0 \mathbf{f})_i = f_i - f_{i-1}. \quad (69)$$

Define the interpolation for 1-forms $\mathcal{I}_1: C^1 \rightarrow \Omega^1([0, 1])$ as

$$\mathcal{I}_1 \mathbf{a} = \sum_{i=2}^N a_i D_i^{p-1}(t) dt. \quad (70)$$

Then we have the following mimetic interpolation property.

Proposition 5.1 (Mimetic interpolation in 1D). *$d(\mathcal{I}_0 \mathbf{f}) = \mathcal{I}_1(\mathbf{d}_0 \mathbf{f})$ for all $\mathbf{f} \in C^0 = \mathbb{R}^N$.*

PROOF. $d(\mathcal{I}_0 \mathbf{f}) = d(\sum_{i=1}^N f_i B_i^p) = \sum_{i=1}^N f_i dB_i^p \stackrel{(67)}{=} \sum_{i=1}^N f_i (D_i^{p-1} - D_{i+1}^{p-1}) dt = \sum_{i=2}^N (\mathbf{d}_0 \mathbf{f})_i D_i^{p-1} dt = \mathcal{I}_1(\mathbf{d}_0 \mathbf{f})$. \square

Remark 5.1. *The discrete 0-forms, 1-forms, and exterior derivative (69) are related to each other in a similar fashion as in Discrete Exterior Calculus (DEC). However, we only think of the data $\mathbf{f} \in C^0$ and $\mathbf{a} \in C^1$ as abstract coefficients, instead of the evaluation or integration of differential forms on a particular grid.*

Remark 5.2. *The parameter domain $[0, 1]$ is mapped to a physical domain $[x_{\min}, x_{\max}]$ linearly by an affine function $x: [0, 1] \rightarrow [x_{\min}, x_{\max}]$. This function $x(t)$ can be represented using B-spline $x(t) = (\mathcal{I}_0 \mathbf{x})(t)$ using a particular list of coefficients $\mathbf{x} = (x_1, \dots, x_N) \in C^0$. These coefficients x_i can be regarded as the effective grid points in the physical space, called the **Greville grid**. Note that \mathbf{x} is in general not uniformly spaced near the boundary, contrary to common computation grids adopted in fluid simulations.*

5.3 Multivariate B-Spline Mimetic Interpolation

We now consider B-splines in n dimension. We will demonstrate the construction for $n = 3$, whose pattern is easy to generalize to other

dimensions. For general constructions, see [Evans and Hughes 2013; Kapidani and Hernandez 2022].

Consider three knot sets Ξ_x, Ξ_y, Ξ_z , each of which is a list like (65) with size $N_x + p + 1, \dots, N_z + p + 1$. By the construction depicted in Section 5.2, we obtain three families $\{B_{x,1}^p, \dots, B_{x,N_x}^p\}, \{B_{y,1}^p, \dots, B_{y,N_y}^p\}, \{B_{z,1}^p, \dots, B_{z,N_z}^p\}$ of degree- p B-splines, and three families $\{D_{x,2}^{p-1}, \dots, D_{x,N_x}^{p-1}\}, \{D_{y,2}^{p-1}, \dots, D_{y,N_y}^{p-1}\}, \{D_{z,2}^{p-1}, \dots, D_{z,N_z}^{p-1}\}$ of degree- $(p-1)$ Curry–Schoenberg M-splines, respectively over Ξ_x, Ξ_y, Ξ_z .

Now consider an abstract grid with $N_x \times N_y \times N_z$ vertices. Similar to the discussion in Remark 5.1, this grid is a **grid of coefficients** that admits a discrete topology identical to a DEC grid. Define $C^0 = \mathbb{R}^{N_x N_y N_z} = \{(g_{ijk})_{1 \leq i \leq N_x, 1 \leq j \leq N_y, 1 \leq k \leq N_z}\}$ where elements are numbers assigned to the vertices of the grid. Likewise, define $C^1 = \mathbb{R}^{(N_x-1)N_y N_z} \oplus \mathbb{R}^{N_x(N_y-1)N_z} \oplus \mathbb{R}^{N_x N_y(N_z-1)}$ where elements $((h_{ijk}^x), (h_{ijk}^y), (h_{ijk}^z))$ are values assigned to edges, $C^2 = \mathbb{R}^{N_x(N_y-1)(N_z-1)} \oplus \mathbb{R}^{(N_x-1)N_y(N_z-1)} \oplus \mathbb{R}^{(N_x-1)(N_y-1)N_z}$ where elements $((f_{ijk}^x), (f_{ijk}^y), (f_{ijk}^z))$ are values assigned to faces, and $C^3 = \mathbb{R}^{(N_x-1)(N_y-1)(N_z-1)}$ where elements $(r_{ijk})_{2 \leq i \leq N_x, \dots, 2 \leq k \leq N_z}$ are values assigned to cells. By the boundary relationship between the vertices, edges, faces, and cells, construct the standard discrete de Rham complex in the DEC framework

$$C^0 \xrightarrow{d_0} C^1 \xrightarrow{d_1} C^2 \xrightarrow{d_2} C^3 \quad (71a)$$

$$d_0 \mathbf{g} = (\partial_x \mathbf{g}, \partial_y \mathbf{g}, \partial_z \mathbf{g}) \quad (71b)$$

$$d_1(\mathbf{h}^x, \mathbf{h}^y, \mathbf{h}^z) = (\partial_y \mathbf{h}^z - \partial_z \mathbf{h}^y, \partial_z \mathbf{h}^x - \partial_x \mathbf{h}^z, \partial_x \mathbf{h}^y - \partial_y \mathbf{h}^x) \quad (71c)$$

$$d_2(\mathbf{f}^x, \mathbf{f}^y, \mathbf{f}^z) = (\partial_x \mathbf{f}^x + \partial_y \mathbf{f}^y + \partial_z \mathbf{f}^z), \quad (71d)$$

where $\partial_x, \partial_y, \partial_z$ are the coordinate difference operators

$$(\partial_x \cdot)_{ijk} := (\cdot)_{ijk} - (\cdot)_{(i-1)jk}, \quad (\partial_y \cdot)_{ijk} := (\cdot)_{ijk} - (\cdot)_{i(j-1)k}, \quad (72a)$$

$$(\partial_z \cdot)_{ijk} := (\cdot)_{ijk} - (\cdot)_{ij(k-1)}. \quad (72b)$$

Note that C^2 is the MAC grid for velocity flux data on faces, and d_2 is the discrete divergence operator.

Define $\mathcal{I}_0: C^0 \rightarrow \Omega^0([0, 1]^3)$ by

$$(\mathcal{I}_0 \mathbf{g})(t_1, t_2, t_3) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \sum_{k=1}^{N_z} g_{ijk} B_{x,i}^p(t_1) B_{y,j}^p(t_2) B_{z,k}^p(t_3), \quad (73)$$

which is the tensor product B-spline interpolation for scalar functions. Now, instead of employing the common treatment of interpolating k -forms ($k \geq 1$) by the same tensor product B-spline basis such as in tri-quadratic or tri-cubic interpolation schemes, one replaces the B-spline functions by the Curry–Schoenberg spline functions for those directions where the edge/face/cell spans a width. That is, define $\mathcal{I}_1: C^1 \rightarrow \Omega^1([0, 1]^3)$ by

$$\begin{aligned} & (\mathcal{I}_1(\mathbf{h}^x, \mathbf{h}^y, \mathbf{h}^z))(t_1, t_2, t_3) \\ &= \sum_{i=2}^{N_x} \sum_{j=1}^{N_y} \sum_{k=1}^{N_z} h_{ijk}^x D_{x,i}^{p-1}(t_1) B_{y,j}^p(t_2) B_{z,k}^p(t_3) dt_1 \\ &+ \sum_{i=1}^{N_x} \sum_{j=2}^{N_y} \sum_{k=1}^{N_z} h_{ijk}^y B_{x,i}^p(t_1) D_{y,j}^{p-1}(t_2) B_{z,k}^p(t_3) dt_2 \\ &+ \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \sum_{k=2}^{N_z} h_{ijk}^z B_{x,i}^p(t_1) B_{y,j}^p(t_2) D_{z,k}^{p-1}(t_3) dt_3, \end{aligned} \quad (74)$$

$\mathcal{I}_2: C^2 \rightarrow \Omega^2([0, 1]^3)$ by

$$\begin{aligned} & (\mathcal{I}_2(\mathbf{f}^x, \mathbf{f}^y, \mathbf{f}^z))(t_1, t_2, t_3) \\ &= \sum_{i=1}^{N_x} \sum_{j=2}^{N_y} \sum_{k=2}^{N_z} f_{ijk}^x B_{x,i}^p(t_1) D_{y,j}^{p-1}(t_2) D_{z,k}^{p-1}(t_3) dt_2 \wedge dt_3 \\ &+ \sum_{i=2}^{N_x} \sum_{j=1}^{N_y} \sum_{k=2}^{N_z} f_{ijk}^y D_{x,i}^{p-1}(t_1) B_{y,j}^p(t_2) D_{z,k}^{p-1}(t_3) dt_3 \wedge dt_1 \\ &+ \sum_{i=2}^{N_x} \sum_{j=2}^{N_y} \sum_{k=1}^{N_z} f_{ijk}^z D_{x,i}^{p-1}(t_1) D_{y,j}^{p-1}(t_2) B_{z,k}^p(t_3) dt_1 \wedge dt_2, \end{aligned} \quad (75)$$

and $\mathcal{I}_3: C^3 \rightarrow \Omega^3([0, 1]^3)$ by

$$\begin{aligned} & (\mathcal{I}_3 \mathbf{r})(t_1, t_2, t_3) \\ &= \sum_{i=2}^{N_x} \sum_{j=2}^{N_y} \sum_{k=2}^{N_z} r_{ijk} D_{x,i}^{p-1}(t_1) D_{y,j}^{p-1}(t_2) D_{z,k}^{p-1}(t_3) dt_1 \wedge dt_2 \wedge dt_3. \end{aligned} \quad (76)$$

This construction ensures the mimetic property.

Theorem 5.1 (Mimetic interpolation in 3D). $d(\mathcal{I}_k \mathbf{a}) = \mathcal{I}_{k+1}(\mathbf{d}_k \mathbf{a})$ for all $\mathbf{a} \in C^k$, $k = 0, 1, 2$.

PROOF. Express the continuous exterior derivative of a differential form $\omega = \sum_I \omega_I dt_I$, where I is some multi-index, as $d\omega = \sum_I \frac{\partial \omega_I}{\partial t_1} dt_1 \wedge dt_I + \frac{\partial \omega_I}{\partial t_2} dt_2 \wedge dt_I + \frac{\partial \omega_I}{\partial t_3} dt_3 \wedge dt_I$. The partial derivatives boil down to differentiating univariate B-splines where (67) applies. Identical to the proof of Proposition 5.1, one performs a summation by parts to arrive at an expression in terms of the coordinate differences (72) of the coefficients. \square

5.3.1 Map to Physical Space. The formulas (73)–(76) interpolate a cochain $\in C^k$ to a differential form $\in \Omega^k([0, 1]^3)$ on the canonical parameter domain $[0, 1]^3$, which is yet to be a physical domain $W = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$. To interpolate cochains into the physical domain, it suffices to apply the linear change of variable as described in Remark 5.2. The explicit change of variables are

$$t_1 = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad t_2 = \frac{y - y_{\min}}{y_{\max} - y_{\min}}, \quad t_3 = \frac{z - z_{\min}}{z_{\max} - z_{\min}}, \quad (77)$$

$$dt_1 = \frac{dx}{x_{\max} - x_{\min}}, \quad dt_2 = \frac{dy}{y_{\max} - y_{\min}}, \quad dt_3 = \frac{dz}{z_{\max} - z_{\min}}. \quad (78)$$

5.3.2 Divergence-Free Interpolation. In 3D, we let $\mathfrak{B} = C^2$, discrete divergence operator $\mathbf{d} = d_2$, and $\mathcal{I} = \sharp \circ * \circ \mathcal{I}_2$. The map $\mathcal{I}_2: \mathfrak{B} \rightarrow \Omega^2(W)$ takes values in a continuous 2-form, the Hodge star $*$ converts it into a 1-form, and \sharp makes the final conversion into a vector field $\mathfrak{X}(W)$. The divergence-free subspace $\mathfrak{B}_{\text{div}} = \ker(d_2)$ maps to closed 2-forms thanks to the mimetic property; closed 2-form corresponds to divergence-free vector fields $\in \mathfrak{X}_{\text{div}}(W)$.

5.3.3 Choice of Knots. While the theory works for arbitrary knot sequence (65), we let the interior knots ξ_{p+2}, \dots, ξ_N be evenly spaced in $[0, 1]$ for each dimension. This choice is for the simplicity of index query. This uniform knot grid is called the **Bézier grid**, which is not to be confused with the Greville grid (*cf.* Remark 5.2). Bézier grid is used for deciding which B-splines have support at a given particle location, whereas the concept of Greville grids do not appear in implementation.

5.3.4 Choice of Degree. We choose the degree of the mimetic interpolation to be $p = 3$. That is, for the velocity interpolation \mathcal{I}_2 from C^2 to $\Omega^2(W)$, the interpolation basis is a mixture of cubic B-splines and quadratic B-splines. This is the minimal degree for the

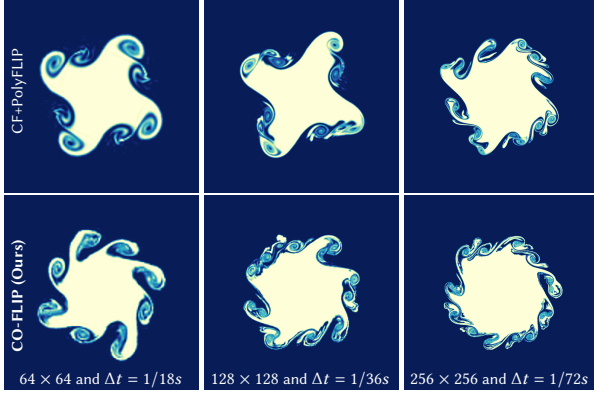


Fig. 16. 2D vortex sheet under spatiotemporal refinement. We compare CO-FLIP method (ours) against state-of-the-art method CF+PolyFLIP. Note that CO-FLIP produces more turbulent vortical structures at every resolution. Further, the lower resolution CO-FLIP results show similar qualitative behavior to higher resolutions of the previous method.

corresponding interpolated velocity to have a C^1 continuity, which is required for computing (14b).

5.3.5 Inclusion of Arbitrary Obstacles. While the mimetic interpolation detailed in this section works for boxed domain $W = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$, one can include arbitrary obstacles in it. This can be achieved by modifying the Galerkin Hodge star \star induced by $\mathcal{I} : \mathfrak{B} \rightarrow \mathfrak{X}(W)$. Set the kinetic energy $\frac{1}{2} \mathbf{f}^\top \star \mathbf{f} = \|\mathcal{I}(\mathbf{f})\|^2$ to infinity if the support of the vector field $\mathcal{I}(\mathbf{f})$ overlaps with the interior of the obstacle. This amounts to modifying the Galerkin Hodge star by surrounding it with a diagonal masking matrix $\tilde{\star} = \Lambda \star \Lambda$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_{\dim(\mathfrak{B})})$, $\lambda_i = \infty$ if the basis vector field $\tilde{b}_i \in \mathfrak{X}(W)$ (cf. (9)) overlaps with the obstacle, and otherwise $\lambda_i = 1$. In the CO-FLIP dynamics, the Galerkin Hodge star appears in the pressure projection (12) in the inverse form $\tilde{\star}^{-1} = \mathbf{V} \star^{-1} \mathbf{V}$, where $\mathbf{V} = \Lambda^{-1} = \text{diag}(V_1, \dots, V_{\dim(\mathfrak{B})})$, $V_i = 0$ in the obstacle and otherwise $V_i = 1$. According to how Hodge star appears in (12), this is effectively removing the columns of \mathbf{d} that is incident to an obstacle.

In practice, to include obstacles, we keep the Galerkin Hodge star unchanged and, instead, remove the entries from \mathbf{d} associated with $\mathfrak{B} = C^2$ grid elements incident to the obstacles.

6 IMPLEMENTATION DETAILS

This section provides practical notes on improving CO-FLIP in terms of computational robustness and efficiency. Additionally, notes on Casimir measurements are included.

6.1 Pressure Projection

For the clarity of exposition of this discussion, we will use the cochain notation in Section 5 for n -dimension. In particular $\mathfrak{B} = C^{n-1}$ and $\mathfrak{B}_{\text{div}} = \ker(\mathbf{d}_{n-1})$.

The pressure projection operator $\mathbf{P}_{\mathfrak{B}_{\text{div}}} : C^{n-1} \rightarrow \ker(\mathbf{d}_{n-1})$ (i.e. $\mathfrak{B} \rightarrow \mathfrak{B}_{\text{div}}$) acts on the space of MAC-grid flux data $\mathbf{f} \in C^{n-1}$.

However, unlike the classical central difference Poisson solve employed for most fluid simulators on MAC-grids, the Laplacian $(\mathbf{d}_{n-1} \star_{n-1}^{-1} \mathbf{d}_{n-1}^\top)$ in (12), or

$$\text{SOLVE } (\mathbf{d}_{n-1} \star_{n-1}^{-1} \mathbf{d}_{n-1}^\top) \mathbf{p} = \mathbf{d}_{n-1} \mathbf{f} \quad \text{FOR } \mathbf{p} \in C^{n*} \quad (79a)$$

$$\text{UPDATE } \mathbf{f}_{\text{new}} \leftarrow \mathbf{f} - \star_{n-1}^{-1} \mathbf{d}_{n-1}^\top \mathbf{p}, \quad (79b)$$

involves a *dense* inverse matrix \star_{n-1}^{-1} of a *non-diagonal* Galerkin Hodge star. Explicitly, building and storing this inverse Hodge star or the Laplace matrix would be memory infeasible.

Section 6.1.1–Section 6.1.3 are three ways to avoid explicitly inverting the Galerkin Hodge star.

6.1.1 Matrix-Free Pressure Projection. When solving this Poisson problem (79a) using a conjugate gradient (CG) method, one could use a matrix-free operation which only involves evaluating the the Laplace operator $(\mathbf{d}_{n-1} \star_{n-1}^{-1} \mathbf{d}_{n-1}^\top)$ acting on a vector. This at most involves a linear solve for evaluating \star_{n-1}^{-1} acting on a vector, which can once again be computed by a CG that only involves actions by \star_{n-1} .

Here \star_{n-1} can also be implemented matrix-free since most entries of this inner product matrix are identical due to the regularity of the grid (Section 5.3.3).

6.1.2 Fourier Domain. If the fluid domain W is a periodic domain, then the Fourier spectral method applies. The Fast Fourier Transform (FFT) diagonalizes all Galerkin Hodge stars \star_0, \dots, \star_n and discrete exterior derivatives $\mathbf{d}_0, \dots, \mathbf{d}_{n-1}$. Hence (79) only amounts to an elementwise arithmetic operation. FFT and its inverse are the only costs for the pressure solve.

6.1.3 Streamform Solve. The projection (79) can also be computed using a streamform (streamfunction). Let us assume that the fluid domain W is simply connected. We will discuss the case of non-simply-connected domains in Remark 6.1.

On simply connected domains, the projected flux $\mathbf{f}_{\text{new}} \in \ker(\mathbf{d}_{n-1})$ is in the image of \mathbf{d}_{n-2} ; i.e. $\mathbf{f} = \mathbf{d}_{n-2} \boldsymbol{\psi}$ for some **streamform** $\boldsymbol{\psi} \in C^{n-1}$ that vanishes at the boundary. For $n = 2$, the boundary condition is a zero Dirichlet boundary condition for the streamfunction $\boldsymbol{\psi} \in C^0$, and for $n = 3$, the stream 1-form vanishes on the edges that are contained in the boundary. This latter condition in 3D corresponds to the fact that the streamform as a vector field is normal to the boundary. These boundary values are easy to impose thanks to the boundary-interpolating property of the knot sequence (65). We let the boundary condition be denoted by $\boldsymbol{\psi}|_{\partial W} = \mathbf{0}$.

Now, the equation satisfied by $\boldsymbol{\psi}$ is the Poisson system

$$(\mathbf{d}_{n-2}^\top \star_{n-1} \mathbf{d}_{n-2}) \boldsymbol{\psi} = \mathbf{d}_{n-2}^\top \star_{n-1} \mathbf{f}, \quad \boldsymbol{\psi}|_{\partial W} = \mathbf{0}. \quad (80)$$

For $n = 2$ the equation (80) becomes $(\mathbf{d}_0^\top \star_1 \mathbf{d}_0) \boldsymbol{\psi} = \mathbf{d}_0^\top \star_1 \mathbf{f}$ which is a scalar Poisson equation with Dirichlet boundary, which has a unique solution. Note that this is a sparse linear solve as there is no involvement of the inverse of any Galerkin Hodge star.

In 3D, however, $(\mathbf{d}_1^\top \star_2 \mathbf{d}_1)$ has a kernel, being the image of \mathbf{d}_0 . A standard treatment in Discrete Exterior Calculus is to fix this gauge degree of freedom by replacing $(\mathbf{d}_1^\top \star_2 \mathbf{d}_1)$ (discrete δd) by the full Hodge Laplacian $(\mathbf{d}_1^\top \star_2 \mathbf{d}_1 + \star_1 \mathbf{d}_0 \star_0^{-1} \mathbf{d}_0^\top \star_1)$ (discrete $\delta d + d\delta$). Notice that this treatment would introduce the inverse of a Galerkin Hodge star. However, by the following theorem, we point out that

the gauge-fixing term $\star_1 \mathbf{d}_0 \star_0^{-1} \mathbf{d}_0^\top \star_1$ does not require that the stars are Galerkin. In fact, all the \star 's in the gauge-fixing term can be replaced by *any* symmetric positive definite matrix (e.g. the identity matrix) without changing the equality (80).

Theorem 6.1. *In 3D, the solution $\psi \in C^1$ to*

$$(\mathbf{d}_1^\top \star_2 \mathbf{d}_1 + \mathbf{A} \mathbf{d}_0 \mathbf{B} \mathbf{d}_0^\top \mathbf{A}) \psi = \mathbf{d}_1^\top \star_2 \mathbf{f}, \quad \psi|_{\partial W} = \mathbf{0} \quad (81)$$

satisfies $(\mathbf{d}_1^\top \star_2 \mathbf{d}_1) \psi = \mathbf{d}_1^\top \star_2 \mathbf{f}$, where \mathbf{A}, \mathbf{B} are arbitrary symmetric positive definite matrices.

PROOF. Suppose ψ is a solution to (81). Multiplying $\psi^\top \mathbf{A} \mathbf{d}_0 \mathbf{B} \mathbf{d}_0^\top$ from the left on (81) and noting that $\mathbf{d}_0^\top \mathbf{d}_1^\top = \mathbf{0}$, one obtains

$$\psi^\top \mathbf{A} \mathbf{d}_0 \mathbf{B} \mathbf{d}_0^\top \mathbf{A} \psi = \mathbf{0}. \quad (82)$$

This implies $\mathbf{d}_0 \mathbf{B} \mathbf{d}_0^\top \mathbf{A} \psi = \mathbf{0}$. Therefore (81) implies $(\mathbf{d}_1^\top \star_2 \mathbf{d}_1) \psi = \mathbf{d}_1^\top \star_2 \mathbf{f}$. \square

In practice we solve the following system in 3D.

$$\text{SOLVE } (\mathbf{d}_1^\top \star_2 \mathbf{d}_1 + \mathbf{d}_0 \mathbf{d}_0^\top) \psi = \mathbf{d}_1^\top \star_2 \mathbf{f}, \quad \psi|_{\partial W} = \mathbf{0}, \quad (83a)$$

$$\text{UPDATE } \mathbf{f}_{\text{new}} \leftarrow \mathbf{d}_1 \psi. \quad (83b)$$

Importantly this is a sparse linear solve as there is no more involvement of any inverse Galerkin Hodge star.

Remark 6.1 (Non-simply-connected domains). *If the domain is non-simply-connected, then we have $\mathbf{f}_{\text{new}} = \mathbf{d}_{n-2} \psi + \mathbf{h}$ for some cohomology component \mathbf{h} . Here $\mathbf{h} \in \mathcal{H}^{n-1} := \{\mathbf{d}_{n-2} \psi \mid \psi|_{\partial W} = \mathbf{0}\}^\perp \cap \ker(\mathbf{d}_{n-1})$ is some divergence-free form that is orthogonal to streamform fields. The space \mathcal{H}^{n-1} is finite dimensional. Elements of \mathcal{H}^{n-1} are close to harmonic: after interpolation \mathcal{I}_{n-1} they are exactly divergence-free but not exactly curl-free [Roy-Chowdhury et al. 2024]. Nevertheless this space serves as a representative for the cohomology space. To obtain this cohomology component $\mathbf{h} \in \mathcal{H}^{n-1}$ in $\mathbf{f}_{\text{new}} = \mathbf{d}_{n-2} \psi + \mathbf{h}$, simply orthogonally project \mathbf{f} to \mathcal{H}^{n-1} .*

Note that we do not need to evolve the cohomology component \mathbf{h} as described in [Yin et al. 2023]. In contrast to a vorticity method that requires evolving the cohomology component, the field \mathbf{f} comes from a particle covector field, whose advection encodes the correct dynamics in its cohomology.

6.1.4 Acceleration for Linear Solves. To speed up the involved CG solve for (83a), we employ a preconditioner. We use a geometric multi-grid (GMG) for the Poisson problem on 1-cochains C^1 defined over edges. Similar speed-ups have previously been proposed for solvers involving the face elements [Aanjaneya et al. 2019]. For this, we require two operators: the *prolongator*, which involves the operation of assigning values to a finer resolution grid, and the *restrictor*, which entails the inverse operation, where coarser edges are used to reconstruct values on a finer mesh. These prolongator and restrictor operators can be constructed by the B-spline interpolation and its pseudoinverse, which can inherit any order of accuracy from the choice of mimetic B-spline interpolations. This method is reminiscent of the works of [Bell 2008] where they perform such operations for Algebraic multigrid methods, and to the work of [De Goes et al. 2016] of extending DEC machinery to subdivision surfaces. For simplicity and to increase the sparsity of coarser-level Laplacians, we choose the lowest accuracy for a mimetic prolongation (i.e. B-spline degree $p = 1$). Additionally, we perform a

row-wise lumping of the restrictor (aka pseudoinverse interpolation) operation to lower the cost of restriction operator. Note that such approximations do not lower the accuracy of the CG solve as these approximations are used only for the preconditioner. With the preconditioner added, the number of iterations for the PCG solver to converge is roughly reduced by three times.

6.2 Adding Pressure Force Back to Particles

In the classical FLIP method, one adds the interpolated pressure force inferred from the pressure projection step to the particle's momentum/impulse. We can do so in the continuous theory, since the particle impulse \vec{u} has a gauge degree of freedom by addition of the gradient of a scalar field (cf. (6) and (26)). Adding the interpolated pressure force to particles can keep $(\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}}$ close to being divergence-free. Close to being divergence-free helps maintaining the stability of the overall method.

Here, we describe two methods (Sections 6.2.1 and 6.2.2) of adding pressure force back to particles. Let us denote the pressure force on the grid by $\tau := \mathbf{f}_{\text{new}} - \mathbf{f} \in C^{n-1}$ during a pressure solve (by (79) or (83)).

6.2.1 Divergence-Consistent Interpolation of Pressure Force. We can add the pressure force on the particle impulse data $(\vec{u}_p)_{p \in \mathcal{P}}$ by interpolating the pressure force as a flux form

$$\vec{u}_p \leftarrow \vec{u}_p + (\mathcal{I}_{n-1} \tau)_{\vec{x}_p}. \quad (84)$$

Remarkably, this modification (84) *does not* modify the vector field $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$ in (14c), using the fact (16) that $\hat{\mathcal{I}}^+$ is a left-inverse of \mathcal{I} . Spelling it out, observe that the modification (84) modifies $\hat{\mathcal{I}}^+(\vec{x}_p, \vec{u}_p)_{p \in \mathcal{P}}$ by $\tau = -\star_{n-1}^{-1} \mathbf{d}_{n-1}^\top \mathbf{p} \in \mathfrak{B}_{\text{div}}^\perp$ (cf. (79b)) which is exactly removed by $\mathbf{P}_{\mathfrak{B}_{\text{div}}}$. As a consequence, the addition of (84) does not affect energy conservation.

The caveat of (84) is that $\mathcal{I}_{n-1} \tau$ as a continuous vector field is not an exact gradient of a continuous function (or a curl-free field). Note that the mimetic guarantee for \mathcal{I}_{n-1} is that it commutes with the exterior derivative for flux form, which is the divergence operator. In particular (84) may modify the state of $J_{\text{adv}}(\vec{x}, \vec{u}) \in \mathfrak{X}_{\text{div}}^*(W) = \Omega^1(W)/d\Omega_0(W)$ in such a way that changes the coadjoint orbit.

In practice, on a simulation grid with a decent resolution, the high-order interpolation $\mathcal{I}_{n-1} \tau$ of a discrete gradient field $\tau = -\star_{n-1}^{-1} \mathbf{d}_{n-1}^\top \mathbf{p}$ is still close to a continuous gradient field. We observe that its effects on the coadjoint orbit preservation and Casimir conservation are minimal (see Figures 21 and 23).

6.2.2 Curl-Consistent Interpolation of Pressure Force. Another option for adding pressure force τ back to the particles is to treat the force as a discretely exact 1-form. In that case, its interpolation by \mathcal{I}_1 , which commutes with curl, will be exact (and curl free). Therefore, the addition of pressure does not affect the coadjoint orbit. To do so for a given $\tau \in C^{n-1}$, first interpolate it to the particles $\vec{t}_p = (\mathcal{I}_{n-1} \tau)_{\vec{x}_p}$ and then inverse-interpolate back to a discrete 1-form $\tilde{\tau} := \hat{\mathcal{I}}_1^+(\vec{x}_p, \vec{t}_p)_{p \in \mathcal{P}} \in C^1$. This $\tilde{\tau} \in C^1$ is the discrete 1-form that best shares the same interpolated vector field with the original $\tau \in C^{n-1}$. Note that $\tilde{\tau}$ is not necessarily exact (\mathbf{d}_0 of a scalar field). So, perform a Poisson reconstruction $\tilde{\mathbf{p}} = (\mathbf{d}_0^\top \star_1 \mathbf{d}_0)^{-1} \mathbf{d}_0^\top \star_1 \tilde{\tau}$ to

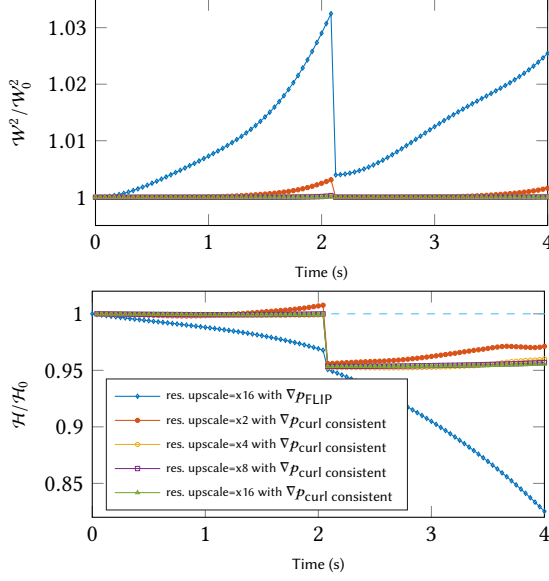


Fig. 17. Enstrophy for two dimensions, and helicity for three dimensions under refinement. Note that with increasing the resolution at which the Casimir measurement takes place, we better see the preservation of them. This is only expected while particle information is only perturbed with the curl consistent pressure force. With the traditional FLIP-based pressure force, we see the Casimirs changing; though, this error is very small. Finally, we emphasize that the jump in the middle is due to global resetting of the particle data.

find the closest exact 1-form $\mathbf{d}_0\tilde{\mathbf{p}}$ to $\tilde{\tau}$. In three dimensions, the reconstruction problem additionally has a Dirichlet boundary condition of $\tilde{\mathbf{p}} = 0$, which ensures that helicity remains unchanged. Finally, add the pressure force

$$\tilde{\mathbf{u}}_p \leftarrow \tilde{\mathbf{u}}_p + (J_1 \mathbf{d}_0 \tilde{\mathbf{p}})_{\tilde{\mathbf{x}}_p}. \quad (85)$$

The modification (85) modifies $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ by an exact form, and therefore, it does not modify the coadjoint in the limit when the particles represent a continuum. Note, however, that the curl-consistent interpolated pressure force is less computationally efficient as it involves two additional global solves, for the pseudo-inverse and Poisson solves. As such, we use the div-consistent pressure force for all of our simulations, despite its minimal effects on Casimir conservation.

6.3 Stability Discussions

The CO-FLIP algorithm is conditionally stable. It substantially improves on the well-known instability problems of the traditional FLIP method. Here, we discuss the various factors that affects stability.

6.3.1 Stability Given by Pressure Force. A known problem in FLIP methods is the frequency gap between the high resolution particles and low resolution grids. The particle data can develop high frequency patterns, whereas an interpolated pressure force feeding back to the particles can only regulate them up to the resolution of the grid. While this problem with FLIP is generally present, we observe

that utilizing mimetic interpolation can significantly improve the regularity of particle data. We observe that the divergence-consistent mimetic interpolation of pressure force (84) of Section 6.2.1 is better at maintaining the stability of FLIP-based methods. In our experiments, as seen in Fig. 18, we show that the addition of the pressure force allows for a far more infrequent global resetting frequency of the particle data. Empirically, we found the reset frequency can be set to once every 200 frames. In contrast, the state-of-the-art frequency in [Deng et al. 2023] is every 20 frames. Additionally, as discussed in Section 6.6, the adaptive resetting of the particles further lowers the error at very infrequent global reset frequencies.

6.3.2 Particle Distribution Over Time. The stability of the CO-FLIP algorithm is further improved by using the mimetic interpolation scheme to get point-wise divergence-free velocity queries which we can integrate to volume preserving flow maps for the particles. Concretely, a uniform independent and identical distribution (i.i.d.) of particles would remain uniformly i.i.d. through the dynamics of the incompressible fluid. However, to reduce the variance on the Monte Carlo integration needed for \hat{I}^+ (see (56)), one could use a stratified sampling of particles per grid cell. Such a stratified sampling would not remain stratified throughout the advection of particles, thus becoming an ill-suited distribution for the Monte Carlo integration of \hat{I}^+ . This would increase the error gap between the discrete operator \hat{I}^+ and the continuous operator I^+ , which would worsen the errors in the simulation and create room for instabilities to grow. As such, on global resets of the particle information, we additionally redistribute the particles in a per-grid-cell-stratified fashion.

6.3.3 Particle-per-Cell Count. Other than the particle distribution, the number of particles per cell (PPC) also affects the variance of the Monte Carlo integration. In principal, one needs a dense enough sample for the least-square system to have a unique solution. However, we empirically choose higher PPC counts to ensure a lower variance in the integration error of the discrete \hat{I}^+ operation. We select minimum and maximum PPC counts, and adaptively distribute particles in the domain based on the magnitude of local vorticity compared to the global maximum; this adaptive nature also helps with the performance (see Section 7.1.1). For 3D experiments, the PPC count is in the range [8 – 27] ([4 – 16] for 2D). In our experiments, this provides stable results throughout the simulated frames; however, below the mentioned range, we see unstable noisy behavior in the velocity field.

6.3.4 Choosing the CFL Number. The CFL number [Courant et al. 1928] we choose for the time integration dictates how good the fixed-point iteration algorithm's initial guess is, which relates to how fast the fixed-point iteration converges. At times, the fixed-point iteration fails to converge due to this. In these cases, we observe an unstable behavior from the solver. Additionally, so long as the fixed-point iteration converges in Algorithm 2, energy can be preserved by using (21); otherwise, one would observe an energy drift. Note, however, than one may not want to increase CFL number too high as it increase the error in the time integration; even if the fixed-point iteration converges. As such, an optimal CFL number is preferred where there is balance between performance and accuracy. We

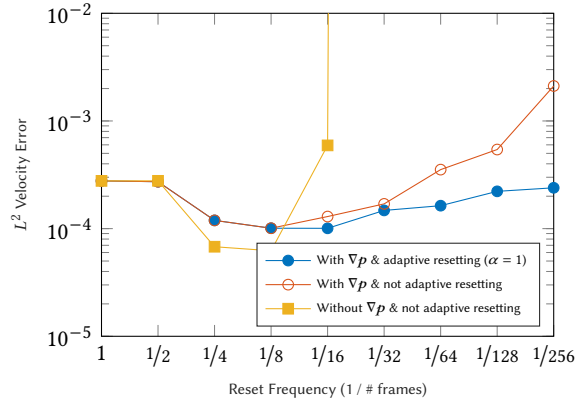


Fig. 18. Comparison of adding or not adding pressure force back to the particles, and whether adaptive resetting is used. Once pressure force is added to the particles, we can avoid blow-ups in lower global reset frequencies. Additionally, the error in simulation can be further minimized in highly infrequent global resets by using adaptive resets on the particles.

empirically find CFL numbers below one to be stable, but we choose roughly 0.5 to be optimal for accuracy.

6.4 Initialization

To initialize the simulation the user can provide a velocity field or a vorticity field. For a given velocity field $\vec{u}_{\text{given}} \in \mathfrak{X}_{\text{div}}(W)$, simply assign it to particles $\vec{u}_p, p \in \mathcal{P}$. One may also filter this velocity through $\hat{I}_{n-1}\hat{I}_{n-1}^+$ so its level of detail is conformed to the resolution of the grid.

For a given vorticity field (scalar field in 2D and vector field in 3D) w , assign the vorticity on each particle, transfer it to the grid by the pseudoinverse of the C^{n-2} mimetic interpolation

$$\mathbf{w} = \star_{n-2} \hat{I}_{n-2}^+ (\vec{x}_p, w_p)_{p \in \mathcal{P}} \in C^{(n-2)*}, \quad (86)$$

and solve the streamform $\psi \in C^{n-2}$, (80) for 2D and (83a) for 3D, with the right-hand side set as $\mathbf{w} \in C^{(n-2)*}$. Finally set $\mathbf{f} = \mathbf{d}_{n-2}\psi$ followed by interpolating \mathbf{f} to particle velocity using \hat{I}_{n-1} .

Note that one should not simply assign values at the grid faces since these faces may not represent the expected physical locations in the domain (cf. Remark 5.1 and Remark 5.2).

6.5 Particle Advection

Here we detail how we execute the particle advection action (17). In (17) we assume a given vector field $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$, and we transport the data (\vec{x}, \vec{u}) by the flow generated by \vec{v} . While solving the ODE system (17) is feasible, we implement the computational setup differently. Instead of updating (\vec{x}_p, \vec{u}_p) at each time t , we store a rest-frame (X_p, \vec{U}_p) and update a flow map and its inverse Jacobian per particle. To access (\vec{x}_p, \vec{u}_p) at the current frame, we use the flow map information to transform the rest-frame (\vec{X}_p, \vec{U}_p) . The main reason for doing so is that we can access the Jacobian at each particle, which is a cumulative record of the quality of the map. Based on the quality of the particle-wise Jacobian, we can reset the rest frame for a particle. This procedure is detailed in Section 6.6.

6.5.1 Data Structure. Each particle $p \in \mathcal{P}$ is assigned with a rest-frame position and impulse $(\vec{X}_p \in W, \vec{U}_p \in T_{\vec{X}}^*W = \mathbb{R}^n)$, and the timestamp $T_p \in \mathbb{R}$ of the rest-frame. Each particle is also equipped with a position vector $\vec{x}_p \in W$ that represents the particle's current location, and an $n \times n$ matrix Ψ_p . The pair of positions \vec{X}_p and \vec{x}_p represents the flow map at this particle from time T_p to the current time t . The matrix Ψ_p is the inverse Jacobian of the flow map.

To access the current impulse \vec{u}_p from this data structure, evaluate $\vec{u}_p = \Psi_p^T \vec{U}_p$.

6.5.2 Reset. When a reset is called for a particle p , such as at the initial frame, we set $\vec{X}_p \leftarrow \vec{x}_p$, $\Psi_p \leftarrow \text{id}$, $T_p \leftarrow t$, and $\vec{U}_p \leftarrow \vec{u}_p^{\text{given}}$, where \vec{u}_p^{given} is the given impulse covector to be assigned to this particle for the reset.

6.5.3 Advection. The advection step (17) for (\vec{x}_p, \vec{u}_p) is equivalent to marching (\vec{x}_p, Ψ_p) (while fixing $\vec{X}_p, \vec{U}_p, T_p$) by the following ODE

$$\begin{cases} \frac{d}{dt} \vec{x}_p(t) = \vec{v}|_{\vec{x}_p(t)} \\ \frac{d}{dt} \Psi_p(t) = -\Psi_p(t)(\nabla \vec{v}|_{\vec{x}_p(t)}) \end{cases} \quad (87)$$

which we solve by RK4. It is easy to verify that $\vec{u}_p(t) = \Psi_p^T(t) \vec{U}_p$ satisfies (17).

6.5.4 Trace-Free and Unimodularity. Since the velocity field \vec{v} from our mimetic interpolation is pointwise divergence free, the Jacobian $\nabla \vec{v}$ is pointwise trace-free $\nabla \vec{v} \in \mathfrak{sl}(n)$.³ Therefore, the solution $\Psi_p(t)$ to $\frac{d}{dt} \Psi_p(t) = -\Psi_p(t)(\nabla \vec{v}|_{\vec{x}_p(t)})$ with $\Psi_p(t = T_p) = \text{id}$ is a unimodular ($\det = 1$, or volume-preserving) matrix $\Psi_p(t) \in \text{SL}(n)$.⁴ The unimodularity of $\Psi_p(t)$ remains true over different frames even when the flow velocity \vec{v} has changed. This is because the algebraic property that $\det = 1$ is closed under matrix compositions ($\text{SL}(n)$ is a group).

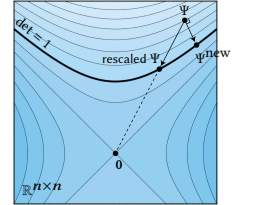
Projection to $\text{SL}(n)$. Although RK4 gives us a highly accurate integration of (87), over a long time $\det(\Psi_p)$ may drift away from 1. To enforce $\det(\Psi_p) = 1$ we employ the following projection to snap Ψ_p back to $\text{SL}(n)$ after each RK4 step [Hairer et al. 2006, Example 4.5]: Update Ψ_p by Ψ_p^{new} , where Ψ_p^{new} solves

$$\begin{cases} \Psi_p^{\text{new}} = \Psi_p + \lambda \Psi_p^{-T}, & \lambda \in \mathbb{R} \\ \det(\Psi_p^{\text{new}}) = 1. \end{cases} \quad (88a) \quad (88b)$$

One can see that (88) is indeed an orthogonal projection to the level-set of determinant in $\mathbb{R}^{n \times n}$ by noting that the cofactor matrix $\text{cof}(\Psi_p) = \det(\Psi_p) \Psi_p^{-T}$ is the gradient (with respect to the Frobenius inner product) of the determinant function at $\Psi_p \in \mathbb{R}^{n \times n}$. Compared to other alternatives, such as normalizing the matrix by its determinant, our projection is less aggressive in modifying the matrix (see inset). Solving (88) amounts to finding $\lambda \in \mathbb{R}$ so that $\det(\Psi_p + \lambda \Psi_p^{-T}) = 1$, which is a 1D root finding problem. We use

³ $\mathfrak{sl}(n)$ is the space of trace-free $n \times n$ matrices. It is the Lie algebra of $\text{SL}(n)$.

⁴The special linear group $\text{SL}(n)$ is the Lie group of $n \times n$ matrices with $\det = 1$.



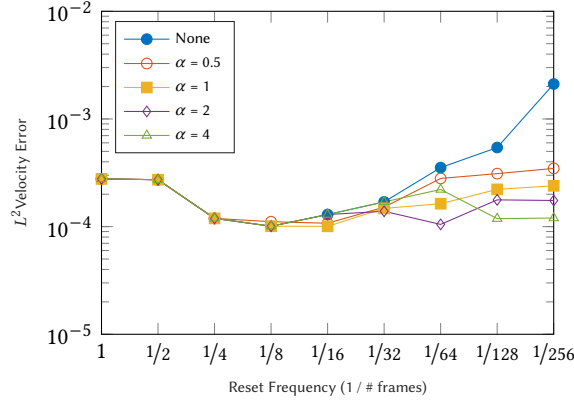


Fig. 19. Comparison of different choices of adaptive resetting cutoff (α) for particles. Note that there is little difference between the cutoff values, but all results show a lower error than those without an adaptive reset.

the following fixed point iteration (quasi-Newton iteration) with initial guess $\lambda_0 = 0$:

$$\lambda_{k+1} = \lambda_k - \frac{\det(\Psi_p + \lambda_k \Psi_p^{-T}) - 1}{\text{tr}((\Psi_p^T \Psi_p)^{-1}) \det(\Psi_p + \lambda_k \Psi_p^{-T})}. \quad (89)$$

The iteration involves 2×2 or 3×3 matrix inversion, which is computationally cheap. In practice, we observe that 1–2 iterations are sufficient.

6.6 Particle-wise Adaptive Reset

The impulse/covector variable \vec{u}_p is stretched by the deformation of the flow (17). However, in a fluid flow, the amount of shearing deformation in a flow map is generally unbounded. This can lead to numerical instability. In the literature, fluid methods that are based on using a long-time flow map to transport momentum or impulse [Qu et al. 2019; Nabizadeh et al. 2022; Deng et al. 2023] reset the flow map globally every few frames to avoid instability. We call the number of frames before a global reset the *reset period*. [Nabizadeh et al. 2022] has a reset period of 5. [Deng et al. 2023] resets every 20 frames. Classical FLIP is also blended with APIC or PIC, with the typical blending factor empirically chosen around roughly 0.99. In extreme cases, semi-Lagrangian, PIC, or APIC methods correspond to resetting the flow map at every frame. Although resetting the flow map stabilizes the fluid simulator, it removes the Lagrangian nature of the method of characteristics.

We point out that the amount of stretching throughout the domain is not uniform. Some parts of the domain may deform heavily and require flow-map resets, whereas the flow map in the other parts is still regular and does not require resets. Indeed, the flow map does not need to be reset globally: in Section 6.5.2, the particle-wise reset can be called *asynchronously*. This motivates an adaptive strategy to the reset.

Per particle $p \in \mathcal{P}$, the reset function Section 6.5.2 is called if the deformation gradient $F_p = \Psi_p^{-1}$ of the particle has its largest

singular value exceeding a threshold. At the reset, we set the rest-frame particle impulse covector \vec{U}_p (cf. Section 6.5.2) as the current interpolated grid velocity field at the particle’s location.

Concretely, for the thresholding, we inspect the following *cumulative Finite Time Lyapunov Exponent*

$$S_p := \ln(\sigma_{\max}(F_p)), \quad (90)$$

where σ_{\max} evaluates the largest singular value. It measures the maximal stretching of the deformation since the previous reset call. Note that S_p is closely related to the Finite Time Lyapunov Exponent (FTLE), which is defined by $\text{FTLE}_p := S_p/T_p$ where T_p is the duration of the deformation as defined in Section 6.5. Empirically, as seen in Fig. 19, we found that limiting $S_p < 1$ (i.e. reset the particle’s flow map if $S_p \geq 1$) leads to stable results without sacrificing accuracy.

We also include an optional global reset like previous methods. As discussed in Section 6.3.1, adding the pressure force back to the particles alone can improve stability so that the reset period can be as much as 256 frames. Our adaptive flow-map resetting further enhances the stability of CO-FLIP. See Fig. 18 for comparison.

6.7 Casimir Measurements

The coadjoint orbit preservation of the CO-FLIP method states that the image $J_{\text{adv}}(\vec{x}, \vec{u}) \in \mathfrak{X}_{\text{div}}^*(W)$ of the fluid state in the infinite dimensional space $\mathfrak{X}_{\text{div}}^*(W)$ of continuous 1-forms stays on a coadjoint orbit, as previously explained in Section 4.1.1. A quantitative measurement of this quality is measuring the Casimirs (cf. A.10). A Casimir is a function defined over $\mathfrak{X}_{\text{div}}^*(W)$ that is constant on each coadjoint orbit. Therefore, a necessary condition for the coadjoint orbit conservation is the preservation of each Casimir function. In 2D a basis for the Casimir functions are the p -th moment of the vorticity function [Khesin and Chekanov 1989]

$$\mathcal{W}^p([\eta]) := \int_W w^p d\mu, \quad w = *(d\eta), \quad [\eta] \in \mathfrak{X}_{\text{div}}^*(W). \quad (91)$$

In 3D the only regular Casimir is the helicity [Khesin et al. 2022]

$$\mathcal{H} := \int_W \eta \wedge \omega, \quad \omega = d\eta, \quad [\eta] \in \mathfrak{X}_{\text{div}}^*(W). \quad (92)$$

However, the Casimirs are defined on the continuous space $\mathfrak{X}_{\text{div}}^*(W)$, whose evaluation can only be approximated. In our validations, we measure the Casimirs using a sequence of grids with progressively higher resolutions to obtain a sequence of progressively more accurate approximations of the Casimirs. The conservation of the Casimirs is observed in the continuous limit of the approximated Casimirs (see Fig. 17).

Note that the procedures of adding divergence-consistent pressure force back to the particles (Section 6.2.1) and the adaptive resetting (Section 6.6) can affect the exactness of Casimir preservation. For validating the exact Casimir preservation in Fig. 17 we employed the curl-consistent pressure feedback (Section 6.2.2) and disabled adaptive resets. To prevent instability we included a global reset every $2 \text{ s} \times 24 \text{ fps} = 48$ frames.

In a more realistic simulation setup we include the more stable divergence-consistent pressure force (Section 6.2.1) and the adaptive resetting (Section 6.6). As shown in Fig. 21 the 2D Casimirs drift only mildly, compared to other methods, over the course of $500 \text{ s} \times 48 \text{ fps} = 24,000$ frames. In 3D, the helicity is conserved better than competing

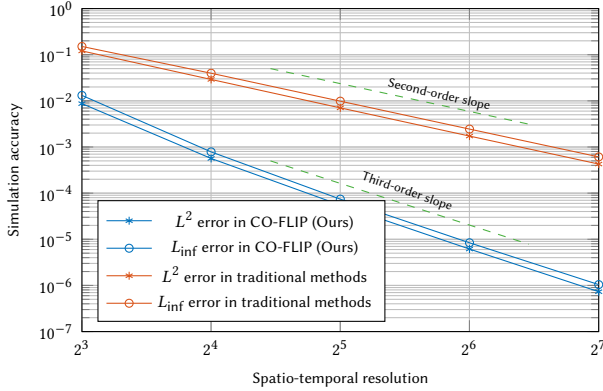


Fig. 20. The 2D convergence plot for the Taylor-Green experiment ran for one second at fixed $CFL = 0.4$. Note that our method achieves third-order accuracy using quadratic B-splines compared to second order accuracy of previous methods.

methods over the course of $10\text{ s} \times 48\text{ fps} = 480$ frames, as shown in Fig. 23.

7 RESULTS

In this section, we detail the experiments conducted to demonstrate our method. We implement our method (CO-FLIP) as detailed in Section 3. In addition to adding our method, we implement established methods such as PolyPIC [Fu et al. 2017] and PolyFLIP [Fei et al. 2021]. We chose the PolyFLIP method over the basic FLIP algorithm because it provides better stability by providing a smaller kernel of P2G transfer. We run these methods with second-order explicit midpoint time integration to make comparisons fair. Additionally, we combine the second-order advection-reflection [Narain et al. 2019] and covector fluids [Nabizadeh et al. 2022] with the PolyFLIP method to build Eulerian-Lagrangian methods referred to as R+PolyFLIP and CF+PolyFLIP, respectively. These methods lower the separation error (i.e. how much advection and pressure projection operators commute) when solving for the Euler equations [Zehnder et al. 2018; Nabizadeh et al. 2022]. We build all the above-mentioned methods on top of the codebase open-sourced by [Nabizadeh et al. 2022]. While the codebase offers many implementations of semi-Lagrangian methods, we instead focus on extending the backend toolset with various performance improvements and implementing the hybrid Eulerian-Lagrangian methods we desire. Readers should refer to the supplementary material for all code and data. Lastly, for the NFM experiments, we use the codebase shared by [Deng et al. 2023] with no additional modifications.

7.1 Performance Considerations

All the methods are parallelized to run on the CPU. We expect all parallelization to be easily transferred to the GPU for additional speed-ups. The experiments ran on a desktop machine with an Intel i9-13900K processor with 64GB of memory. See Table 3 for a summary of experiment statistics, and Fig. 8 for how long our method takes per iteration for the Trefoil knot experiment. Our

Table 3. Experiment statistics.

Figure name (figure number)	Domain size	Grid resolution	Avg. particles	Δt
Pyroclastic plume (Fig. 1)	$5 \times 10 \times 5\text{ m}^3$	$96 \times 192 \times 96$	3×10^7	1/288 s
Unknot (Fig. 4)	$5 \times 5 \times 5\text{ m}^3$	$64 \times 64 \times 64$	2×10^6	1/48 s
Spot obstacle (Fig. 5)	$10 \times 5 \times 5\text{ m}^3$	$128 \times 64 \times 64$	6×10^6	1/96 s
Rayleigh-Taylor Instability (Fig. 6)	$0.2 \times 0.4\text{ m}^2$	256×512	3×10^6	1/288 s
Rocket (Fig. 7)	$5 \times 5 \times 5\text{ m}^3$	$64 \times 64 \times 64$	1×10^6	1/48 s
Ink jet (Fig. 10)	$5 \times 10 \times 5\text{ m}^3$	$64 \times 128 \times 64$	1×10^7	1/48 s
3D smoke plume (Fig. 14)	$5 \times 10 \times 5\text{ m}^3$	$64 \times 128 \times 64$	5×10^6	1/192 s
2D smoke plume (Fig. 15)	$0.5 \times 1.0\text{ m}^2$	256×512	4×10^6	1/288 s
Leapfrogging vortices (Fig. 22)	$2\pi \times 2\pi\text{ m}^2$	256×256	1×10^6	1/96 s
Twisted torus (Fig. 23)	$5 \times 5 \times 5\text{ m}^3$	$64 \times 64 \times 64$	4×10^6	1/48 s
Trefoil knot (Fig. 25)	$5 \times 5 \times 5\text{ m}^3$	$64 \times 64 \times 64$	9×10^5	1/72 s
Leapfrogging rings (Fig. 26)	$10 \times 5 \times 5\text{ m}^3$	$128 \times 64 \times 64$	7×10^6	1/72 s

method's additional cost compared with traditional methods is due to the use of high-order methods. The majority of the time is spent solving the global problems of streamform-vorticity solve, and pseudoinverse P2G map. We conduct a performance study under the same conditions such as same particle-per-cell count, and CFL number. Compared to traditional methods, CO-FLIP is slower in wall clock time by an order of magnitude, given our modest optimizations. However, our results provide higher accuracy, and preservation of energy and Casimirs. Note that increasing the resolution of traditional methods, for achieving lower error, is not always practical, as they would hit system memory constraints. This is because the PolyPIC methods require $8\times$ additional data per particle compared to our method, and that they are only second order accurate compared to the third order accuracy of CO-FLIP. Finally, our method's performance scales linearly, similar to traditional methods, with increasing spatiotemporal resolution. Below, we discuss optimizations for the global solves that impact our method.

7.1.1 Pseudoinverse P2G Solve. We use a preconditioned conjugate gradient (PCG) solver to tackle the pseudoinverse P2G problem. This solver has three major components: (1) a preconditioner for the solver, (2) the interpolation operator (i.e. G2P transfer), and (3) the interpolation transpose operator (i.e. weighted sum of the particles values back to the grid). We use the *incomplete Cholesky (IC) with limited memory* implementation from the Eigen library [Lin and Moré 1999; Guennebaud et al. 2010] to speed up the solver. The IC preconditioner requires the construction of matrix $\hat{I}^\top \hat{I}$, which we build explicitly, without dealing with the much larger matrix \hat{I} . The sparsity pattern of $\hat{I}^\top \hat{I}$ is exactly the same as the one for \star_2 , and for the IC preconditioner, we only require its lower/upper triangular part to be constructed. For this, we run through the particles and subsequently go through the neighboring grid elements using a double for-loop to deposit their contribution to said elements. For the purpose of this paper, we only parallelize the for-loop over particles in the three axes of the domain (in \mathbb{R}^3), but more advanced binning strategies could further allow for parallelization and performance boost. Additionally, evaluating the IC preconditioner itself incurs an overhead, but in turn, it drastically reduces the number of iterations during the least-squares pseudoinverse solve. As discussed in [Gould and Scott 2017], if the IC preconditioner is evaluated at every step, the combined time taken to evaluate the preconditioner and run the solver is no different than using a simple diagonal preconditioner. To

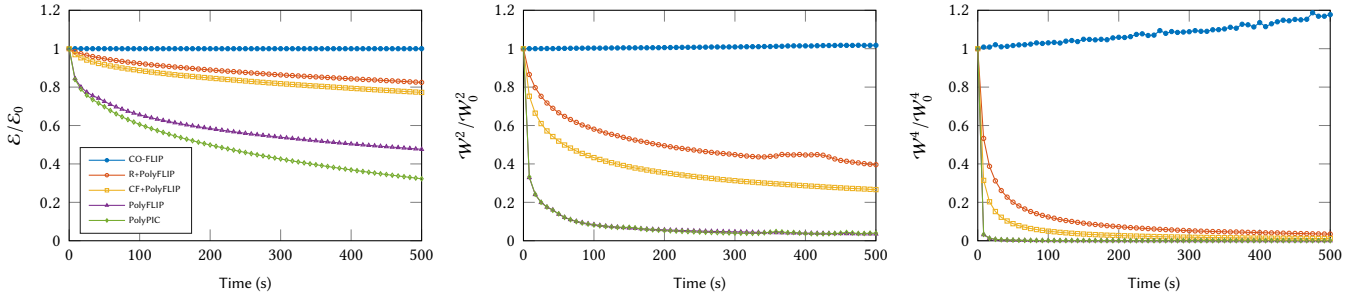


Fig. 21. Energy, enstrophy (second moment of vorticity), and fourth moment of vorticity plots over time for the 2D leapfrogging vortices experiment. Note that our method exactly conserves energy over time, minimal relative error in the measured values for Casimirs, compared to other methods. For clarity purposes, the plots show data samples at $\times 400$ times lower frequency than the simulation frame rate.

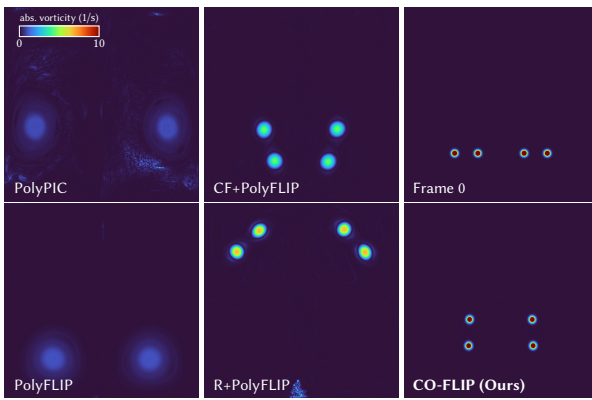


Fig. 22. 2D leapfrogging vortices. Our method (CO-FLIP) retains the core vortex structure with minimal dissipation after 500 seconds of simulation. Other methods lose both energy and vorticity as time progresses.

alleviate this problem and continue to benefit from the IC preconditioner’s performance improvement, we note that the preconditioner for one step could be reused for some future steps with a minimal drop in performance. We find empirically that re-evaluating the preconditioner every five timesteps shows an improvement in total runtime.

The second major component of the pseudoinverse solve is the interpolation operator, which is embarrassingly parallelizable because there is no data race between particles looking up their values from the grid. The third component, the transpose of this operator, is more complicated. This is because many particles could be in the neighborhood of a grid element, causing a data race. One approach to this problem is through *scattering* [Gao et al. 2018]. Here, each particle’s information is transferred to its neighboring grid element. A trivial method of introducing parallelization for the scattering technique is through atomic writes. For this paper, we employ this method. However, as the number of particles in the system grows, it is slower to resolve write conflicts. To resolve this problem, there are a few works that optimize and alleviate this problem on both CPU [Fang et al. 2018; Hu et al. 2018] and GPU [Gao et al. 2018]

by employing a method of binning using a form of data structure (e.g. SPGrids). This optimization level is beyond the scope of the current paper, but we expect our method to similarly improve if such speed-up methods are used. Finally, to further lower write conflicts and improve the performance, we opted to use an adaptive set of particles throughout the domain. Concretely, the number of particles per cell is selected based on the relative magnitude of local vorticity compared to its maximum in the entire domain. This would additionally lower the memory footprint of the particles. Overall, the use of the IC preconditioner, as detailed above, improves the runtime of the P2G solve by roughly $\times 50$.

7.1.2 Streamform-Vorticity Solve. We employ a PCG solver based on streamform and vorticity to find divergence-free velocity fields. Since the Laplacian matrix comes from a high-order Galerkin Hodge star, it has a more dense sparsity pattern than the simple finite-difference Laplacian matrix. This more accurate Laplacian matrix allows for a more accurate pressure projection operation but incurs an overhead to the solver. We detail various ways of improving this overhead.

As mentioned in Section 6.1.4, we employ a geometric multigrid (GMG) preconditioner for the solution based on the edges (nodes in 2D) of the systems. The preconditioner takes an input matrix and applies the prolongation and restrictor operators on either side to construct the progressively smaller matrices for the coarser levels. As a result of these GMG preconditioners, the PCG solver can converge in much fewer iterations [Saad 2003]. However, there is a tradeoff between the complexity of the GMG preconditioner and the total time the solver takes to converge. As such, we opt to build the GMG preconditioner using a smaller Laplacian matrix from a lower-order Hodge star. Note that the solver still uses the high-order Laplacian matrix, and only the preconditioner is approximated further. This introduces no error in the solver as the preconditioner is an approximation anyway. Our experiments show that a Laplacian matrix built from the second-order Galerkin Hodge star produces the best total runtime. While the number of iterations is larger than using the GMG preconditioner built from the Laplacian matrix based on the third-order Hodge star, each iteration takes less time, reducing the total time.

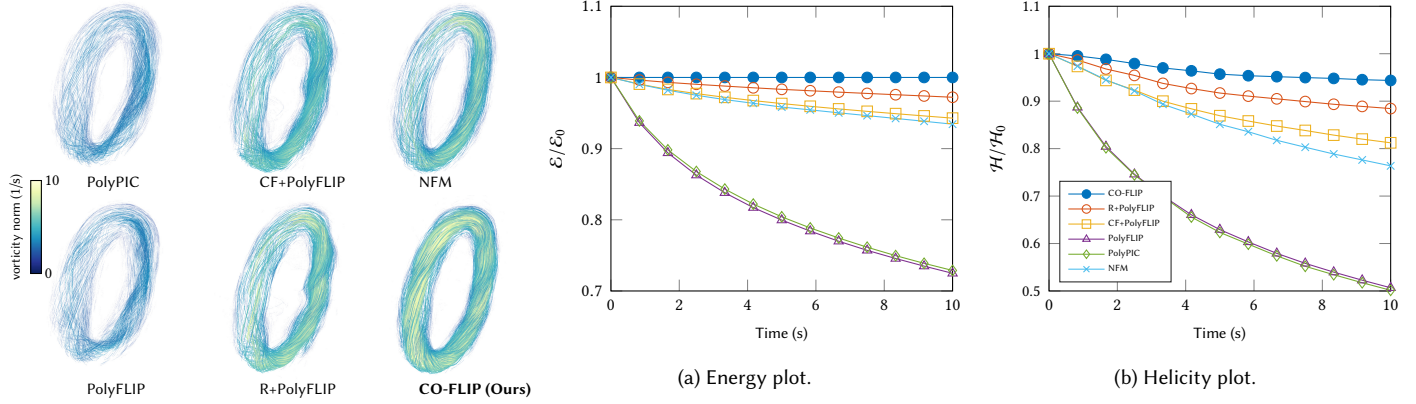


Fig. 23. Twisted torus comparison, and its associated energy and helicity plots. Note that our method exactly preserves energy over time at a low resolution of $64 \times 64 \times 64$. The measured helicity shows minimal relative error compared with other methods. For clarity purposes, the plots show data samples at x20 times lower frequency than the simulation frame rate.

Another avenue for optimization is in the inner problems that must be solved at each level of the GMG grids. Traditionally, without the presence of obstacles or multiphase fluids, the Laplacian decouples, which results in simplified solves during the inner iterative solves of the preconditioners, may use red-black Gauss-Seidel parallelization for these solves for acceleration. However, in the presence of Laplacian matrices made from high-order Hodge stars, they do not decouple, making the inner solves more complicated. Inspired by the results in [Adams et al. 2003; Saad 2003], we note that instead of using multicolor Gauss-Seidel methods, one may use Chebyshev acceleration (see [Saad 2003, algorithm 12.1]) to make fast a weighted Jacobi iteration solver by choosing an optimal weight. For this, one needs only an approximation of the largest and smallest eigenvalues of the matrix [Adams et al. 2003] at each grid level, which we get by using the Spectra [Qiu et al. 2010] library. Overall, we see a x3 reduction in the number of iterations needed for the PCG solver to converge. We observe that in some experiments, despite the reduction in the number of iterations, the solver takes a longer time to complete compared to a CG solver with fast parallelized matrix-vector multiplication and no preconditioner. We believe the slowdown is due to an unoptimized implementation of the matrix-vector multiplication borrowed from the Eigen library, which would be resolved if one uses explicit matrix-free multiplications instead.

7.1.3 Miscellaneous improvements. For all the Hodge stars that must be inverted, we evaluate an IC preconditioner on the first simulation frame to speed up the solver. This lowers the number of iterations needed to converge by roughly five times. Additionally, we use parallelized matrix-free operations for all matrix-vector multiplications involving the Hodge star operators, which further enhance the performance over sparse matrix-vector multiplication solutions offered by the Eigen library.

7.2 Validation

We validate our method through a collection of experiments. We showcase how our energy conserves energy and the Casimirs throughout the simulation.

7.2.1 2D Taylor-Green Vortices Convergence. Here, we validate the order of accuracy of our method using the Taylor-Green vortices experiment [Taylor and Green 1937], an analytical solution to the Euler equations. At a fixed $CFL = 0.4$, we run the experiment using our method and traditional methods for one second. We measure the relative L^2 and L_{inf} errors under progressively higher spatio-temporal resolutions. As seen in Fig. 20, our method shows third-order convergence when using quadratic B-splines as the interpolation operator. The traditional methods remain only at second-order, as expected. Studying the plot, our method requires a lower spatio-temporal resolution to achieve a fixed error rate. For instance, if the user is looking for an error of 10^{-5} , our method can achieve this error at a 64^2 resolution, while traditional methods must run at 1024^2 to match this accuracy.

7.2.2 2D and 3D Casimirs under refinement. As discussed in Section 6.7, we use a set of auxiliary measurement grids. We progressively increase the resolution of the measurement grid while running the experiment based on a fixed-resolution simulation grid. The Hamiltonian is measured on the simulation grid.

In 2D, we initialize the velocity field analytically with the Taylor-Green velocity field in an 8×8 simulation grid. We measure enstrophy, which is the Casimir related to the second moment of vorticity. As seen in Fig. 17, enstrophy is preserved progressively better under finer measurement grids. Additionally, Fig. 17 demonstrates that even without the curl consistent interpolation, i.e. using the traditional FLIP pressure force, the Casimirs are only changed by a small amount.

In 3D, we use an ABC flow velocity field pressure projected to a closed box. Note that, after pressure projection the velocity field no longer exactly follows the ABC analytical equation, but provides

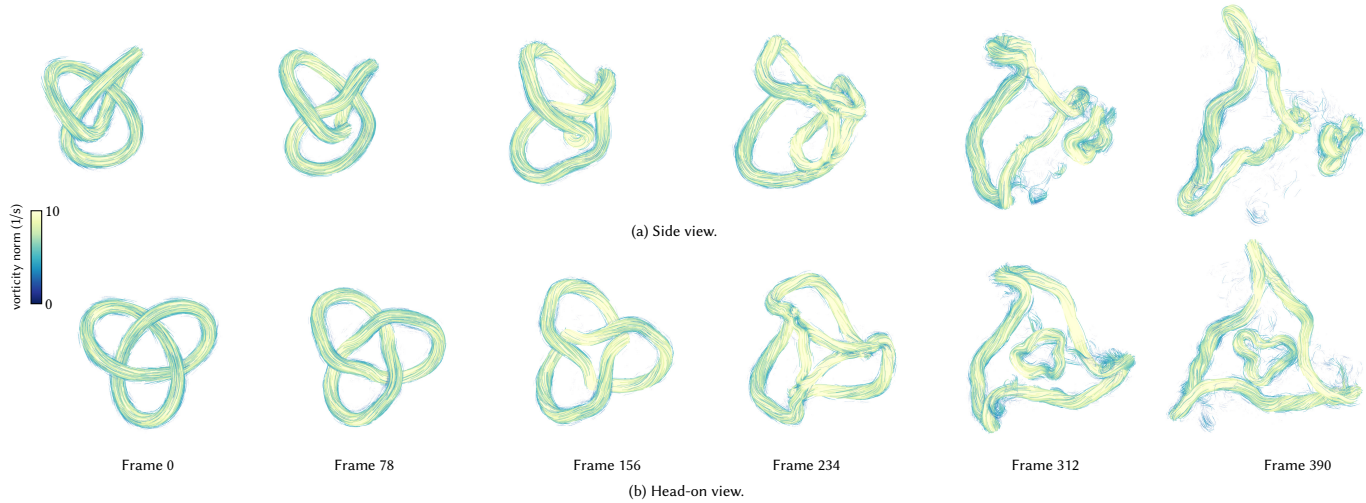


Fig. 24. Trefoil knot, also known as the (2,3)-torus knot. As time passes, the knot moves forward and stretches until the vortex reconnection even when it breaks into one big and one smaller part [Kleckner and Irvine 2013]. Our captures this behavior at a low resolution of $64 \times 64 \times 64$.

a close-to-ABC flow; which is smooth velocity field. Similar to the 2D experiment, Fig. 17 shows that helicity is preserved under fine enough measurements. Note that the coarse measurement shows a larger error, and should only be taken as an approximate measurement for the helicity Casimir.

7.2.3 2D Vortex leapfrogging vortices. For the 2D vortex leapfrogging experiment, the vortices are expected to run indefinitely as they leap through one another. Inspecting the final frame in Fig. 22, we see that our method (CO-FLIP) maintains the core vortical structure of the vortices despite 500 seconds of simulation, while other methods have drastically lost their energy and vorticity. As seen in Fig. 21, our method conserves the energy close to floating point accuracy, while all previous methods drastically lose their energy over time. From this plot, we also see that the Casimirs for our method only change slightly. For the second moment of vorticity \mathcal{W}^2 , our method has a relative error of 2%, which is drastically lower than other methods (the best one being at 60% error). Similarly, for the fourth moment of vorticity \mathcal{W}^4 , our method has a relative error of 17% compared to 97% (at best) for other methods.

7.2.4 2D Vortex sheet under refinement. To study cascading vortical structures, we initialize the velocity field with a rigidly rotating disk with angular velocity 0.25 rad/s. Due to the sudden drop in velocity at the border of the disk, a vortex sheet is created. Together with the finite resolution of the grid, this would create an instability that produces cascading vortices. See Fig. 16. Note that with our method the energy remains constant regardless of the spatiotemporal resolution. With increasing the resolution, our method consistently produces richer vortical structures, which qualitatively mimics the results of previous methods at a higher resolution. Additionally, the structures get progressively more turbulent.

7.2.5 3D Twisted Torus. We further validate our method in three dimensions. When filaments in a closed vortex tube are twisted, they generate a velocity field with non-zero helicity [Kleckner and Irvine

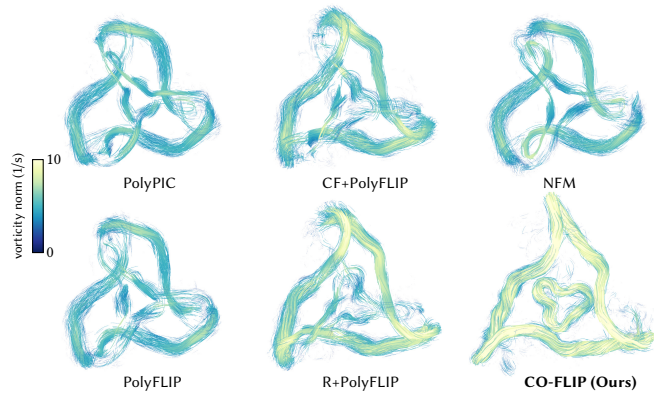


Fig. 25. Comparisons of trefoil knot between different methods. Note that our method shows a superior preservation of energy and vortical structures, while successfully showcasing the vortex reconnection event.

2013]. We use the codebase open-sourced by [Chern et al. 2017] to generate such a vortex tube. Concretely, we restrict the initial vorticity field to the inside of a torus with a major radius $1.5m$ and an aspect ratio of 6. Given two variables $\hat{\omega}_1$ (for vorticity per unit length through the torus tube) and ω_2 (for vorticity responsible for creating twists, which is perpendicular to the former component). We set the total vorticity field as $\hat{\omega}_1(\vec{n}(\vec{x}) \times \vec{d}(\vec{x})) + \omega_2\vec{n}(\vec{x})$, where \vec{x} is a location in the domain, $\vec{d} = \vec{x} - \vec{r}_{\text{core}}$, and \vec{n} is the tangential unit vector to the torus at minor radius \vec{r}_{core} from the core. We refer to this tube as a *twisted torus*. As seen in Fig. 23, our method preserves the twisted vortical structures at the end of the simulation, while all other methods have lost their structure. Solely inspecting the energy plot, one may conclude that such methods provide a high-quality result. But when inspected visually, as seen in Fig. 23, we see a large



Fig. 26. Leapfrogging vortex rings in 3D. At a low resolution of $128 \times 64 \times 64$, our method (CO-FLIP) leaps five times before the vortex rings mix. At an even lower resolution of $64 \times 32 \times 32$, CO-FLIP manages to leap almost three times, which shows better behavior than some traditional works at higher resolutions. Previous methods at best (i.e. R+PolyFLIP method) leap twice.

loss of vorticity in other methods as time progresses. This can be explained by complementing the analysis with the helicity plot. Our method conserves energy exactly, and only shows a small error (5%) in maintaining helicity, while other methods lose their helicity more significantly (around 13% at best for R+PolyFLIP).

7.2.6 3D Unknot. Here, we focus on a (1,5)-torus knot, which we refer to as an “unknot” following the convention from [Maggioni et al. 2010]. We set up our experiment using the following construction for a (p,q)-torus knot formula in \mathbb{R}^3 :

$$x = r \cos(p\theta), y = r \sin(p\theta), z = -\sin(q\theta), \quad (93)$$

where $r = \cos(q\theta) + 2$ and $\theta \in [0, 2\pi]$. As the vortex unknot travels forward in time, the structure stretches to a point where a reconnection event occurs, and five other smaller rings are created from the bigger unknot (see Fig. 3). As seen in Fig. 4, our method

(CO-FLIP) captures this even on a low resolution of 64^3 , while all other methods fail to achieve a clean separation. Additionally, note that the vortex strength is kept at the initial maximum for our results, while all other methods have lost most of their vorticity.

7.2.7 3D Trefoil Knot. We further validate our experiments by running the trefoil knot experiment, which sets up a (2,3)-torus knot in space. We expect that the knot will evolve and break into two separate knots. While other methods fail to fully undergo the vortex reconnection event, our method achieves this and maintains its vorticity strength, as seen from the color of the vortex filaments in Fig. 25. This evolution is captured by our method in Fig. 24 at a low resolution of 64^3 .

7.2.8 3D Vortex Leapfrogging. Similar to the two-dimensional case, the vortex leapfrogging is expected to continue indefinitely under no

viscous forces. However, the 3D experiment has proven to be more difficult to maintain this ever leaping behavior. As shown in Fig. 26, at low resolutions such as $128 \times 64 \times 64$, previous methods fail to recover the correct behavior and, at best, leap twice (see R+PolyFLIP). Our method (CO-FLIP) achieves five leaps before merging into one another. Note that at even lower resolutions, previously not seen before, our method manages to leap once before merging together. Based on this trend, we expect that higher resolutions would allow for higher leap counts.

7.3 Demonstrations

We also demonstrate that our method can create visually stunning simulations of phenomena seen in nature that are often of interest to the graphics community using relatively low resolutions (See Table 3).

7.3.1 2D Rayleigh Taylor Instability. Fig. 6 shows that our method in simulating 2D Rayleigh Taylor instability. We set up an experiment with one Rayleigh Taylor finger. Heavy fluid sits above light fluid and travels downward, creating fractal-shaped vortices. We set a small viscosity of $10^{-7} m^2/s$ and buoyancy acceleration $0.15 m/s^2$, and run the simulation with a small sinusoidal perturbation at the interface given by $\frac{L}{2} + 0.05W(\cos(\frac{2\pi}{W}) - 1)$, where L, W are the length and height of the domain. Note that our method can create fractal-shaped vortical structures that better match the expected output at very high Reynolds numbers. While other methods produce a modest amount of vorticities, our method produces abundantly more features at the same resolution.

7.3.2 2D Smoke plume. To additionally showcase the intricate vortical structures produced by our CO-FLIP algorithm, we set up an experiment where buoyant hot travels upward with acceleration $0.1 m/s^2$ in two dimensions. We apply a small viscosity of $10^{-6} m^2/s$. Fig. 15 shows how over time, our method produces an increasing amount of turbulent fractal-shaped vortices.

7.3.3 3D Smoke plume. Fig. 14 shows an experiment with a smoke plume under buoyancy $1 m/s^2$ and a small viscosity of $10^{-4} m^2/s$. Our method captures more detail and vortices than other methods at a low resolution of $64 \times 128 \times 64$. Our results at this resolution are comparable to traditional methods running at double the resolution. Even at a lower resolution of $32 \times 64 \times 32$, previously considered as too coarse, our method can create highly energetic results. As the resolution drops, CO-FLIP provides a consistent qualitative visual result with a rich level of vortical structures. This is because CO-FLIP does not have any numerical dissipation. Regardless of the resolution, the difference in the behavior is due to how far the I -discrete Euler equations are from the continuous Euler equations.

7.3.4 3D Pyroclastic. We further demonstrate the effectiveness of our high-order structure-preserving solver through the pyroclastic cloud Fig. 1. By injecting hot, buoyant smoke in the shape of a flat slab under buoyancy acceleration of $0.1 m/s^2$, our method produces results that are comparable in terms of the amount of vortical details to real footage of volcanic clouds emerging after their eruption. Note that our method runs at only $96 \times 192 \times 96$ simulation, which was previously considered too low for such effects. As shown in Fig. 27,



Fig. 27. A comparison of the pyroclastic experiment for different resolutions. Note as the spatiotemporal resolution increases, at a constant CFL number, we see more dense and energetic vortical structures emerge.

even at a lower resolution of $64 \times 128 \times 64$, we see energetic dense vortical structures.

7.3.5 3D Inkjet. We simulate the effect of injecting ink with a high speed of $2 m/s$ into a medium in Fig. 10. The ink evolves under a buoyancy acceleration of $0.1 m/s^2$, and a small viscosity of $1 \times 10^{-4} m^2/s$. Note the interplay of two smoke jets as they reach one another, mix, and produce many vortical structures while running at a low resolution of $64 \times 128 \times 64$.

7.3.6 3D Rocket. To further showcase our method, we run an experiment inspired by rocket launch footage. We stage a nozzle that steadily emits hot smoke into the scene at a speed of $0.9 m/s$, where the effective buoyancy acceleration is $0.55 m/s^2$ point upwards. We additionally use a small amount of viscosity of $1 \times 10^{-4} m^2/s$. Fig. 7 demonstrates the abundance of turbulent fuel and smoke left in the wake of the rocket flying upwards. Note that the rocket model and the smoke are not coupled; this is purely for the final presentation of the simulation. The simulation runs at a low resolution of 64^3 .

7.3.7 3D Spot Obstacle. We implement arbitrary boundary handling as described in Section 5.3.5. Fig. 5 shows how our method produces intricate vortical structures at a low resolution of $128 \times 64 \times 64$, as the smoke jet coming out of the nozzle interacts with the obstacle shaped by the Spot model [Crane et al. 2013b]. The nozzle jet speed for this experiment is set to $2 m/s$, and we use a small amount of viscosity of $1 \times 10^{-4} m^2/s$.

8 CONCLUSION AND FUTURE WORK

This paper details the construction of a high-order energy- and Casimirs-preserving discretization of the Euler equations for the dynamics of inviscid incompressible fluids in a FLIP-based hybrid Eulerian-Lagrangian framework. By a few key novel modifications to the FLIP algorithm, we establish a discrete Hamiltonian dynamical system. The Hamiltonian flow is based on an energy defined on the discrete grid velocity field, which sets motion on a symplectic space represented by a particle system. We employ a mimetic B-spline interpolation scheme which gives us pointwise divergence-free velocities as well as volume-preserving actions on the particles. This

action is symplectomorphic on the particle system, and as such induces a momentum map back to the grid, which perfectly matches the P2G transfer operator in the FLIP algorithm. Since momentum maps are Poisson maps, the actions preserve the coadjoint orbits, and thereby Casimirs, in the infinite-dimensional dual space of divergence-free velocity fields. Finally, the pressure projection uses an inner product structure given by the induced metric through the interpolation operator. This choice for the pressure projection is crucial for discrete energy conservation. As each component of the solver is canonically derived from the interpolation, the entire system can be made arbitrarily high-order by only specifying the order of accuracy of the interpolation operator.

We demonstrate that our high-order solver is capable of producing and maintaining high levels of detail in fluid flows. Even using a low resolution grid, the state on the coadjoint orbit can still develop turbulent flow with high resolution vortical structures and maintain concentrated vortex filaments. In contrast, competing numerical methods cannot capture the same level of detail at low resolutions. The mathematical reason is that, in our method, the grid resolution only affects the Hamiltonian, whereas the Poisson structure that describes the advective nature for the momentum is still the same as the one in the continuous theory. We also elucidate that the Casimir preservation is expected to be at the infinite-dimensional dual space of continuous velocities instead of on the dual space of the finite dimensional grid velocities suggested by previous structure-preserving discretizations of Euler equations.

The implicit time integration in the method facilitates long-term stability. Previous stability issues in FLIP-based algorithms are minimized when the mimetic interpolation is consistently applied to pressure force and the P2G transfer.

While our method provides higher orders of accuracy and structure preservation in simulating fluids, it comes with a higher computational cost. Currently, the performance bottleneck lies in the pseudoinverse solve in the P2G transfer. For future work, we aim to construct more efficient solvers that can tackle the bottlenecks of the algorithm. Another avenue is to extend the CO-FLIP machinery to free-surface flows, requiring careful handling of the Galerkin Hodge star near the free-surface boundary. We are also excited to explore other possible fluid solvers by following a similar derivation of CO-FLIP (Section 4) but with a different mimetic velocity representation and a different auxiliary symplectic space.

ACKNOWLEDGMENTS

This work was funded in part by the Ronald L. Graham Chair, NSF Career Award 2239062, and the UC San Diego Center for Visual Computing. Additional support was provided by SideFX Software and Activision Blizzard. The work used compute resources provided by the National Research Platform (funded by NSF grants 2100237 and 2120019), which Khai Vu helped the authors to set up. Moreover, the authors would like to thank Mukund Varma for sharing his desktop machine as an additional computational resource. Finally, the authors would like to thank Professors Melvin Leok and Craig Schroeder, and Dr. Alexey Stomakhin for insightful discussions.

REFERENCES

- Mridul Aanjaneya, Chengguizi Han, Ryan Goldade, and Christopher Batty. 2019. An Efficient Geometric Multigrid Solver for Viscous Liquids. *Proceedings of the ACM in Computer Graphics and Interactive Techniques 2*, 2, Article 14 (July 2019), 21 pages.
- Mark Adams, Marian Brezina, Jonathan Hu, and Ray Tuminaro. 2003. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *J. Comput. Phys.* 188, 2 (2003), 593–610.
- Douglas N Arnold, Richard S Falk, and Ragnar Winther. 2006. Finite element exterior calculus, homological techniques, and applications. *Acta numerica* 15 (2006), 1–155.
- Vladimir I. Arnold. 1966. Sur la géométrie différentielle des groupes de Lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. In *Annales de l'institut Fourier*, Vol. 16. 319–361.
- Vladimir I. Arnold and Boris A. Khesin. 1998. *Topological Methods in Hydrodynamics*. Springer.
- Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Maks Ovsjanikov. 2013. An operator approach to tangent vector field processing. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 73–82.
- Omri Azencot, Maks Ovsjanikov, Frédéric Chazal, and Mirela Ben-Chen. 2015. Discrete derivatives of vector fields on surfaces—an operator approach. *ACM Transactions on Graphics (TOG)* 34, 3 (2015), 1–13.
- Omri Azencot, Steffen Weißmann, Maks Ovsjanikov, Max Wardetzky, and Mirela Ben-Chen. 2014. Functional Fluids on Surfaces. In *Proceedings of the Symposium on Geometry Processing (Cardiff, United Kingdom) (SGP '14)*. Eurographics Association, Goslar, DEU, 237–246. <https://doi.org/10.1111/egf.12449>
- William N Bell. 2008. Algebraic multigrid for discrete differential forms. (2008).
- J.U. Brackbill and H.M. Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.* 65, 2 (1986), 314–343. [https://doi.org/10.1016/0021-9991\(86\)90211-1](https://doi.org/10.1016/0021-9991(86)90211-1)
- Jeremiah U Brackbill, Douglas B Kothe, and Hans M Ruppel. 1988. FLIP: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications* 48, 1 (1988), 25–38.
- Annalisa Buffa, Judith Rivas, Giancarlo Sangalli, and Rafael Vázquez. 2011. Isogeometric discrete differential forms in three dimensions. *SIAM J. Numer. Anal.* 49, 2 (2011), 818–844.
- Annalisa Buffa, Giancarlo Sangalli, and Rafael Vázquez. 2010. Isogeometric analysis in electromagnetics: B-splines approximation. *Computer Methods in Applied Mechanics and Engineering* 199, 17–20 (2010), 1143–1152.
- Elena Celledoni, Håkon Marthinsen, and Brynjulf Owren. 2014. An introduction to Lie group integrators—basics, new developments and applications. *J. Comput. Phys.* 257 (2014), 1040–1061.
- Jumyung Chang, Ruben Partono, Vinicius C Azevedo, and Christopher Batty. 2022. Curl-Flow: Boundary-Respecting Pointwise Incompressible Velocity Interpolation for Grid-Based Fluids. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 243:1–243:21.
- Jiajie Chen and Thomas Y Hou. 2022. Stable nearly self-similar blowup of the 2D Boussinesq and 3D Euler equations with smooth data I: Analysis. *arXiv preprint arXiv:2210.07191* (2022).
- Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2017. Inside fluids: Clebsch maps for visualization and processing. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 142:1–142:11.
- Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. 2016. Schrödinger's smoke. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 77:1–77:13.
- A. Clebsch. 1859. Ueber die Integration der hydrodynamischen Gleichungen. *Journal für die reine und angewandte Mathematik* 56 (1859), 1–10. English translation by D. H. Delphenich, http://www.neo-classical-physics.info/uploads/3/4/3/6/34363841/clebsch_-_clebsch_variables.pdf.
- Jens Cornelis, Markus Ihmsen, Andreas Peer, and Matthias Teschner. 2014. IISPH-FLIP for incompressible fluids. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 255–262.
- Ricardo Cortez. 1995. Impulse-based methods for fluid flow. <https://doi.org/10.2172/87798>
- Richard Courant, Kurt Friedrichs, and Hans Lewy. 1928. Über die partiellen Differenzgleichungen der mathematischen Physik. *Mathematische annalen* 100, 1 (1928), 32–74.
- Maurice G Cox. 1972. The numerical evaluation of B-splines. *IMA Journal of Applied mathematics* 10, 2 (1972), 134–149.
- Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses (Anaheim, California) (SIGGRAPH '13)*. Association for Computing Machinery, New York, NY, USA, Article 7, 126 pages. <https://doi.org/10.1145/2504435.2504442>
- Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013b. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.
- Haskell Brooks Curry and Isaac Jacob Schoenberg. 1947. On spline distributions and their limits—the polya distribution functions. In *Bulletin of the American Mathematical Society*, Vol. 53. AMER MATHEMATICAL SOC 201 CHARLES ST, PROVIDENCE, RI 02940-2213, 1114–1114.
- Haskell Brooks Curry and Isaac J Schoenberg. 1966. On Pólya frequency functions IV: the fundamental spline functions and their limits. *J. Analyse math* 17, 71 (1966), 107.

- Carl De Boor. 1972. On calculating with B-splines. *Journal of Approximation theory* 6, 1 (1972), 50–62.
- Fernando De Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. 2016. Subdivision exterior calculus for geometry processing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.
- Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–21.
- Ounan Ding, Tamar Shinar, and Craig Schroeder. 2020. Affine particle in cell method for MAC grids and fluid simulation. *J. Comput. Phys.* 408 (2020), 109311.
- Kathryn P Drake, Edward J Fuselier, and Grady B Wright. 2021. A partition of unity method for divergence-free or curl-free radial basis function approximation. *SIAM Journal on Scientific Computing* 43, 3 (2021), A1950–A1974.
- Todd F Dupont and Yingjie Liu. 2003. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.* 190, 1 (2003), 311–324.
- David G Ebin and Jerrold Marsden. 1970. Groups of diffeomorphisms and the motion of an incompressible fluid. *Annals of Mathematics* (1970), 102–163.
- Essex Edwards and Robert Bridson. 2012. A high-order accurate particle-in-cell method. *Internat. J. Numer. Methods Engrg.* 90, 9 (2012), 1073–1088.
- Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics (TOG)* 26, 1 (2007), 4–es.
- Tarek M Elgindi, Tej-Eddine Ghoul, and Nader Masmoudi. 2019. On the stability of self-similar blow-up for $C^{1,\alpha}$ solutions to the incompressible Euler equations on \mathbb{R}^3 . *arXiv preprint arXiv:1910.14071* (2019).
- Kent Engo and Stig Faltinsen. 2001. Numerical Integration of Lie–Poisson Systems While Preserving Coadjoint Orbits and Energy. *SIAM journal on numerical analysis* 39, 1 (2001), 128–145.
- John A Evans and Thomas JR Hughes. 2013. Isogeometric divergence-conforming B-splines for the unsteady Navier–Stokes equations. *J. Comput. Phys.* 241 (2013), 141–167.
- Yu Fang, Yuanming Hu, Shi-Min Hu, and Chenfanfu Jiang. 2018. A temporally adaptive material point method with regional time stepping. In *Computer graphics forum*, Vol. 37. Wiley Online Library, 195–204.
- Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 15–22.
- Yun (Raymond) Fei, Qi Guo, Rundong Wu, Li Huang, and Ming Gao. 2021. Revisiting Integration in the Material Point Method: A Scheme for Easier Separation and Less Dissipation. *ACM Trans. Graph.* 40, 4, Article 109 (Aug. 2021), 16 pages. <https://doi.org/10.1145/3450626.3459678>
- Fan Feng, Jinyuan Liu, Shiyong Xiong, Shuqi Yang, Yaorui Zhang, and Bo Zhu. 2022. Impulse Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* (2022).
- Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 222:1–222:12.
- Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018. GPU optimization of material point methods. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–12.
- Almut Gassmann and Hans-Joachim Herzog. 2008. Towards a consistent numerical compressible non-hydrostatic model using generalized Hamiltonian tools. *Quarterly Journal of the Royal Meteorological Society* 134, 635 (2008), 1597–1613.
- Marc Gerritsma. 2010. Edge functions for spectral element methods. In *Spectral and High Order Methods for Partial Differential Equations: Selected papers from the ICOSAHOM’09 conference, June 22–26, Trondheim, Norway*. Springer, 199–207.
- Artemio González-López, Niky Kamran, and Peter J. Olver. 1992. Lie Algebras of Vector Fields in the Real Plane. s3-64, 2 (1992), 339–368. <https://doi.org/10.1112/plms/s3-64.2.339>
- Nicholas Gould and Jennifer Scott. 2017. The state-of-the-art of preconditioners for sparse linear least-squares problems. *ACM Transactions on Mathematical Software (TOMS)* 43, 4 (2017), 1–35.
- Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. <http://eigen.tuxfamily.org>.
- Ernst Hairer, Marlis Hochbruck, Arieh Iserles, and Christian Lubich. 2006. Geometric numerical integration. *Oberwolfach Reports* 3, 1 (2006), 805–882.
- Chad C Hammerquist and John A Nairn. 2017. A new method for material point method particle updates that reduces noise and enhances stability. *Computer methods in applied mechanics and engineering* 318 (2017), 724–738.
- Xuchen Han, Theodore F. Gast, Qi Guo, Stephanie Wang, Chenfanfu Jiang, and Joseph Teran. 2019. A Hybrid Material Point Method for Frictional Contact with Diverse Materials. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 17 (jul 2019), 24 pages. <https://doi.org/10.1145/3340258>
- Francis H Harlow. 1962. *The particle-in-cell method for numerical solution of problems in fluid dynamics*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
- Hermann von Helmholtz. 1858. Über Integrale der hydrodynamischen Gleichungen, welche den Wirbelbewegungen entsprechen. *Journal für die reine und angewandte Mathematik* 55 (1858), 25–55. English translation by P.G. Tait, 1867, <http://www.biodiversitylibrary.org/item/121849#page/499/mode/1up>.
- René R Hiemstra, Deepesh Toshniwal, RHM Huijsmans, and Marc I Gerritsma. 2014. High order geometric methods with exact conservation properties. *J. Comput. Phys.* 257 (2014), 1444–1471.
- Anil Nirmal Hirani. 2003. *Discrete exterior calculus*. California Institute of Technology.
- Darryl D Holm. 2011. *Geometric mechanics-Part I: Dynamics and symmetry*. World Scientific Publishing Company.
- Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.
- Thomas JR Hughes, John A Cottrell, and Yuri Bazilevs. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering* 194, 39–41 (2005), 4135–4195.
- Anton Izosimov and Boris Khesin. 2017. Classification of Casimirs in 2D hydrodynamics. *arXiv preprint arXiv:1702.01843* (2017).
- Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans. Graph.* 36, 4, Article 152 (jul 2017), 14 pages. <https://doi.org/10.1145/3072959.3073623>
- Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 51:1–51:10.
- Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, 1–52.
- Guang-Shan Jiang and Chi-Wang Shu. 1996. Efficient implementation of weighted ENO schemes. *Journal of computational physics* 126, 1 (1996), 202–228.
- Bernard Kapidani and Rafael Vázquez Hernandez. 2022. High Order Geometric Methods With Splines: An Analysis of Discrete Hodge-Star Operators. *SIAM Journal on Scientific Computing* 44, 6 (2022), A3673–A3699.
- BA Khesin and Yu V Chekanov. 1989. Invariants of the Euler equations for ideal or barotropic hydrodynamics and superconductivity in D dimensions. *Physica D: Nonlinear Phenomena* 40, 1 (1989), 119–131.
- Boris Khesin, Gerard Misiolek, and Klas Modin. 2019. Geometry of the Madelung transform. *Archive for Rational Mechanics and Analysis* 234 (2019), 549–573.
- Boris Khesin, Daniel Peralta-Salas, and Cheng Yang. 2022. The helicity uniqueness conjecture in 3D hydrodynamics. *Trans. Amer. Math. Soc.* 375, 02 (2022), 909–924.
- ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jaroslav R Rossignac. 2005. *Flowfixer: Using BFEC for fluid simulation*. Technical Report. Georgia Institute of Technology.
- Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. 2016. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.
- Dustin Kleckner and William TM Irvine. 2013. Creation and dynamics of knotted vortices. *Nature physics* 9, 4 (2013), 253–258.
- Tassilo Kugelstadt, Andreas Longva, Nils Thuerey, and Jan Bender. 2019. Implicit density projection for volume conserving liquids. *IEEE Transactions on Visualization and Computer Graphics* 27, 4 (2019), 2385–2395.
- John M Lee. 2013. Smooth manifolds. In *Introduction to smooth manifolds*. Springer, 1–31.
- Chih-Jen Lin and Jorge J Moré. 1999. Incomplete Cholesky factorizations with limited memory. *SIAM Journal on Scientific computing* 21, 1 (1999), 24–45.
- Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced variational fluid simulation. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 244:1–244: 12.
- Luan Lyu, Xiaohua Ren, Wei Cao, Jian Zhu, Enhua Wu, and Zhi-Xin Yang. 2024. Wavelet Potentials: An Efficient Potential Recovery Technique for Pointwise Incompressible Fluids. In *Computer Graphics Forum*. Wiley Online Library, e15023.
- Francesca Maggioni, Sultan Alamri, Carlo F Barenghi, and Renzo L Ricca. 2010. Velocity, energy, and helicity of vortex knots and unknots. *Physical Review E* 82, 2 (2010), 026309.
- Jerrold Marsden and Alan Weinstein. 1983. Coadjoint Orbits, Vortices, and Clebsch Variables for Incompressible Fluids. *Physica D: Nonlinear Phenomena* 7, 1 (1983), 305–323.
- Jerrold E Marsden, Sergey Pekarsky, and Steve Shkoller. 1999. Discrete euler-poincaré and lie-poisson equations. *Nonlinearity* 12, 6 (1999), 1647.
- Jerrold E Marsden, Tudor S Ratiu, and Robert Hermann. 1997. Introduction to mechanics and symmetry. *SIAM Rev.* 39, 1 (1997), 152–152.
- Henry Keith Moffatt. 1969. The degree of knottedness of tangled vortex lines. *Journal of Fluid Mechanics* 35, 1 (1969), 117–129.
- H Keith Moffatt. 2014. Helicity and singular structures in fluid dynamics. *Proceedings of the National Academy of Sciences* 111, 10 (2014), 3663–3670.
- Jean Jacques Moreau. 1961. Constantes d’un îlot tourbillonnaire en fluide parfait barotrope. *Comptes rendus hebdomadaires des séances de l’Académie des sciences* 252

- (1961), 2810–2812.
- Philip J Morrison. 1998. Hamiltonian description of the ideal fluid. *Reviews of modern physics* 70, 2 (1998), 467.
- Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiyong Tong, and Mathieu Desbrun. 2009. Energy-preserving integrators for fluid animation. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 38:1–38:8.
- Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector fluids. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 113:1–113:16.
- John A. Nairn and Chad C. Hammerquist. 2021. Material point method simulations using an approximate full mass matrix inverse. *Computer Methods in Applied Mechanics and Engineering* 377 (2021), 113667. <https://doi.org/10.1016/j.cma.2021.113667>
- Yoichiro Nambu. 1973. Generalized hamiltonian dynamics. *Physical Review D* 7, 8 (1973), 2405.
- Rahul Narain, Jonas Zehnder, and Bernhard Thomaszewski. 2019. A second-order advection-reflection solver. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 2, 2 (2019), 16:1–16:14.
- Francis J Narcowich and Joseph D Ward. 1994. Generalized Hermite interpolation via matrix-valued conditionally positive definite functions. *Math. Comp.* 63, 208 (1994), 661–687.
- P N vir and R Blender. 1993. A Nambu representation of incompressible hydrodynamics using helicity and enstrophy. *Journal of Physics A: Mathematical and General* 26, 22 (1993), L1189.
- nLab authors. 2024. Derivations of smooth functions are vector fields. <https://ncatlab.org/nlab/show/derivations+of+smooth+functions+are+vector+fields>. Revision 14.
- VI Oseledets. 1989. On a new way of writing the Navier-Stokes equation. The Hamiltonian formalism. *Russ. Math. Surveys* 44 (1989), 210–211.
- Dmitry Pavlov, Patrick Mullen, Yiyong Tong, Eva Kanso, Jerrold E Marsden, and Mathieu Desbrun. 2011. Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena* 240, 6 (2011), 443–458.
- Alexander Pletzer and David Fillmore. 2015. Conservative interpolation of edge and face data on n dimensional structured grids using differential forms. *J. Comput. Phys.* 302 (2015), 21–40.
- Alexander Pletzer and Wolfgang Hayek. 2019. Mimetic interpolation of vector fields on Arakawa C/D grids. *Monthly Weather Review* 147, 1 (2019), 3–16.
- Yixuan Qiu et al. 2010. Spectra. <https://spectralib.org/>.
- Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang. 2022. The power particle-in-cell method. *ACM Transactions on Graphics* 41, 4 (2022).
- Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 128:1–128:12.
- Daniel Ram, Theodore Gast, Chenfanfu Jiang, Craig Schroeder, Alexey Stomakhin, Joseph Teran, and Pirouz Kavehpour. 2015. A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 157–163.
- Ritoban Roy-Chowdhury, Tamar Shinar, and Craig Schroeder. 2024. Higher order divergence-free and curl-free interpolation on MAC grids. *J. Comput. Phys.* (2024), 112831.
- Yousef Saad. 2003. *Iterative methods for sparse linear systems*. SIAM.
- Takahiro Sato, Christopher Batty, Takeo Igarashi, and Ryoichi Ando. 2018. Spatially adaptive long-term semi-Lagrangian method for accurate velocity advection. *Computational Visual Media* 4, 3 (2018), 223–230.
- Robert Saye. 2016. Interfacial gauge methods for incompressible fluid dynamics. *Science advances* 2, 6 (2016), e1501869.
- Craig Schroeder, Ritoban Roy Chowdhury, and Tamar Shinar. 2022. Local divergence-free polynomial interpolation on MAC grids. *J. Comput. Phys.* 468 (2022), 111500.
- Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35, 2 (2008), 350–371.
- Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A Vortex Particle Method for Smoke, Water and Explosions. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 910–914. <https://doi.org/10.1145/1073204.1073282>
- Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 121–128.
- Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.
- Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.
- Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer physics communications* 87, 1-2 (1995), 236–252.
- Andre Pradhana Tampubolon, Theodore Gast, Gergely Kl r, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.
- Geoffrey Ingram Taylor and Albert Edward Green. 1937. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* 158, 895 (1937), 499–521.
- Jerry Tessendorf and Brandon Pelfrey. 2011. The characteristic map for fast and efficient vfx fluid simulations. In *Computer Graphics International Workshop on VFX, Computer Animation, and Stereo Movies*. Ottawa, Canada.
- William Thomson. 1868. On Vortex Motion. *Earth and Environmental Science Transactions of the Royal Society of Edinburgh* 25, 1 (1868), 217–260.
- Stephanie Wang, Mohammad Sina Nabizadeh, and Albert Chern. 2023. Exterior Calculus in Graphics: Course Notes for a SIGGRAPH 2023 Course. In *ACM SIGGRAPH 2023 Courses* (Los Angeles, California) (SIGGRAPH ’23). Association for Computing Machinery, New York, NY, USA, Article 8, 126 pages. <https://doi.org/10.1145/3587423.3595525>
- Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. 2019. CD-MPM: continuum damage material point methods for dynamic fracture animation. *ACM Trans. Graph.* 38, 4, Article 119 (jul 2019), 15 pages. <https://doi.org/10.1145/3306346.3322949>
- Shiyong Xiong, Zhecheng Wang, Mengdi Wang, and Bo Zhu. 2022. A Clebsch method for free-surface vortical flow simulation. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.
- Shuqi Yang, Shiyong Xiong, Yaorui Zhang, Fan Feng, Jinyuan Liu, and Bo Zhu. 2021. Clebsch gauge fluid. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 99:1–99:11.
- Hang Yin, Mohammad Sina Nabizadeh, Baichuan Wu, Stephanie Wang, and Albert Chern. 2023. Fluid Cohomology. *ACM Trans. Graph.* 42, 4, Article 126 (jul 2023), 25 pages. <https://doi.org/10.1145/3592402>
- Gunther Z ngl. 2013. ICON: The icosahedral nonhydrostatic modelling framework of DWD and MPI-M. In *Proc. ECMWF Seminar on Numerical Methods for Atmosphere and Ocean Modelling*.
- Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 85:1–85:8.
- Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the missing vorticity in advection-projection fluid solvers. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 52:1–52:8.
- Yi Zhang, Artur Palha, Marc Gerritsma, and Leo G Rebholz. 2022. A mass-, kinetic energy- and helicity-conserving mimetic dual-field discretization for three-dimensional incompressible Navier-Stokes equations, part I: Periodic domains. *J. Comput. Phys.* 451 (2022), 110868.
- Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 965–972.

A PRELIMINARIES

In this appendix, we review the essentials of Lie algebra, symplectic structure, Poisson structure, and Hamiltonian systems in a self-contained manner. A more complete introduction to these topics can be found in textbooks such as [Marsden et al. 1997; Holm 2011]. Our introduction uses notations in exterior calculus [Yin et al. 2023, Appendix A; Wang et al. 2023; Crane et al. 2013a] and terminologies in group theory.

Most of the technical terminologies in the group theoretic Hamiltonian fluid dynamics are borrowed from geometric Hamiltonian mechanics. In this paradigm, a Hamiltonian dynamical system is described by an energy function defined on a *phase space*. This energy function is called the *Hamiltonian*.

In the most general setting, the type of a phase space is a *Poisson space*, which is a manifold or a vector space equipped with a *Poisson structure*. The Poisson structure allows turning the Hamiltonian into a vector field (analogous to how a metric turns a loss function into a gradient field in numerical optimization). The vector field generated by the Hamiltonian and the Poisson structure is called the *symplectic gradient field* of the Hamiltonian. The Hamiltonian dynamical system is the negative symplectic gradient flow on the phase space (analogous to gradient descents).

Two most important examples of Poisson spaces are *symplectic manifolds* and *dual spaces of Lie algebras*. Let us briefly review the definitions and notations of these concepts.

A.1 Lie Algebra

Definition A.1 (Lie algebra). A **Lie algebra** $(V, [\cdot, \cdot])$ is a vector space V equipped with a skew-symmetric bilinear function $[\cdot, \cdot]: V \times V \xrightarrow{\text{bilinear}} V$ satisfying the Jacobi identity: $[\vec{a}, [\vec{b}, \vec{c}]] + [\vec{b}, [\vec{c}, \vec{a}]] + [\vec{c}, [\vec{a}, \vec{b}]] = 0$ for all $\vec{a}, \vec{b}, \vec{c} \in V$.

Elementary examples of Lie algebra products include the 3D cross product (\mathbb{R}^3, \times) and matrix commutator $(\mathbb{R}^{n \times n} = \mathfrak{gl}(\mathbb{R}^n), [\mathbf{A}, \mathbf{B}] := \mathbf{AB} - \mathbf{BA})$.

Definition A.2 (Adjoint representation). The currying⁵ of the Lie bracket $[\cdot, \cdot]$ of a Lie algebra V is called the **adjoint representation**

$$\text{ad}: V \rightarrow \underbrace{V \rightarrow V}_{\mathfrak{gl}(V)}, \quad \text{ad}_{\vec{u}}(\vec{v}) := [\vec{u}, \vec{v}]. \quad (94)$$

As a direct consequence of the Jacobi identity for $[\cdot, \cdot]$:

Proposition A.1. $\text{ad}: (V, [\cdot, \cdot]) \rightarrow (\mathfrak{gl}(V), [\cdot, \cdot]_{\text{commutator}})$ is a Lie algebra homomorphism.

PROOF. By currying the Jacobi identity $[\vec{a}, [\vec{b}, \vec{c}]] + [\vec{b}, [\vec{c}, \vec{a}]] + [\vec{c}, [\vec{a}, \vec{b}]] = 0$ we obtain $\text{ad}_{\vec{a}} \text{ad}_{\vec{b}} \vec{c} - \text{ad}_{\vec{b}} \text{ad}_{\vec{a}} \vec{c} - \text{ad}_{[\vec{a}, \vec{b}]} \vec{c} = 0$. Therefore $\text{ad}_{[\vec{a}, \vec{b}]} = [\text{ad}_{\vec{a}}, \text{ad}_{\vec{b}}]_{\text{commutator}}$. \square

Here, a **Lie algebra homomorphism** between two Lie algebras $(U, [\cdot, \cdot]), (V, [\cdot, \cdot])$ is referred to as a linear map $A: U \rightarrow V$ so that $A([\vec{u}_1, \vec{u}_2]) = [A\vec{u}_1, A\vec{u}_2]$. A **Lie algebra anti-homomorphism** is a linear map satisfying $A([\vec{u}_1, \vec{u}_2]) = -[A\vec{u}_1, A\vec{u}_2]$ instead.

The classes of Lie algebras that will be relevant in our context are the following two examples.

Example A.1. The tangent space $\mathfrak{g} = T_e G$ to a Lie group G at the identity $e \in G$ is a Lie algebra. The Lie bracket is given by the mixed 2nd derivative of the conjugation expression $(g \in G, h \in G) \mapsto ghg^{-1} \in G$ at the identity

$$[\vec{u}, \vec{v}] := \frac{\partial}{\partial g} \left(\frac{\partial}{\partial h} (ghg^{-1}) \Big|_{h=e} \right) \Big|_{g=e} \llbracket \vec{v} \rrbracket. \quad (95)$$

The next example says that the space of vector fields on a domain is naturally a Lie algebra. To concisely describe it, the space $\Gamma(TM)$ of tangent vector fields on a manifold M is identified with the space $\text{Der}(C^\infty(M))$ of **derivations** for scalar functions on M . Here, a derivation is any linear operator $D: C^\infty(M) \xrightarrow{\text{linear}} C^\infty(M)$ satisfying the Leibniz rule $D(fg) = D(f)g + fD(g)$. Every tangent vector field \vec{u} can be taken as a derivation, being the directional derivative operator: $(\vec{u} \triangleright) \in \text{Der}(C^\infty(M))$, $\vec{u} \triangleright f := df \llbracket \vec{u} \rrbracket$. Conversely, each derivation can be regarded as a directional derivative operator along a vector field [nLab authors 2024]. In graphics, this representation of vector fields has been used for tangent vector field processing and solving PDEs on surfaces [Azencot et al. 2013, 2014, 2015].

Example A.2. The space $\mathfrak{X}(M) = \text{Der}(C^\infty(M))$ of tangent vector fields on a manifold M is a Lie algebra, with $[\vec{u}, \vec{v}] \triangleright := (\vec{u} \triangleright)(\vec{v} \triangleright) - (\vec{v} \triangleright)(\vec{u} \triangleright)$.

The following proposition combines Example A.1 and Example A.2 in the context of group actions. A **group action** of a Lie group G on a manifold M is a (anti-)homomorphism Φ from G to smooth maps

⁵As the currying operation in functional programming.

from M to M (with composition “ \circ ” of smooth maps as the group operation). Spelling it out, we have type $\Phi: G \xrightarrow{\text{hom}} (M \rightarrow M)$. That is, $\Phi(g_1 g_2)(x) = \Phi(g_1)\Phi(g_2)(x)$, $g_1, g_2 \in G, x \in M$. We call such Φ a **left action**. If $\Phi: G \xrightarrow{\text{anti-hom}} (M \rightarrow M)$ is an anti-homomorphism, i.e. $\Phi(g_1 g_2)(x) = \Phi(g_2)\Phi(g_1)(x)$, we call Φ a **right action**.

Note that if $\Phi: G \xrightarrow{(\text{anti-})\text{hom}} (M \rightarrow M)$ is a group action, then its differential at the identity is a linear map from $\mathfrak{g} = T_e G$ to the space of tangent vector fields $\mathfrak{X}(M)$ on M , representing an infinitesimal motion.

Proposition A.2. The differential $d\Phi|_e$ of a left action $\Phi: G \xrightarrow{\text{hom}} (M \rightarrow M)$ at the identity is a Lie algebra anti-homomorphism

$$d\Phi|_e: \mathfrak{g} \xrightarrow[\text{Lie alg}]{\text{anti-hom}} \mathfrak{X}(M). \text{ Likewise, the differential of a right action } \Phi: G \xrightarrow{\text{anti-hom}} (M \rightarrow M) \text{ at the identity is a Lie algebra homomorphism } d\Phi|_e: \mathfrak{g} \xrightarrow[\text{Lie alg}]{\text{hom}} \mathfrak{X}(M).$$

PROOF. [Lee 2013, Theorem 20.15]. \square

A.2 Poisson Structure

A Hamiltonian dynamical system is a 1st order ODE on a phase space. This phase space has a structure that turns a Hamiltonian function into a vector field describing the ODE. In the most general setting, this structure is the Poisson structure. There are two special examples of spaces that are naturally endowed with Poisson structures. One example is any symplectic space (Section A.3). The second example is the dual vector space of any Lie algebra (Section A.5).

Definition A.3. A **Poisson manifold** (or **Poisson space**) is a manifold (or vector space) P equipped with a Lie algebra structure on the scalar functions $\{\cdot, \cdot\}: C^\infty(P) \times C^\infty(P) \xrightarrow{\text{Lie bracket}} C^\infty(P)$, satisfying that $\{f, \cdot\}: C^\infty(P) \rightarrow C^\infty(P)$ is a derivation for every $f \in C^\infty(P)$.

Definition A.4. The vector field $\vec{X}^f \in \Gamma(TP)$ associated with the derivation $\{f, \cdot\} = \vec{X}^f \triangleright (\cdot)$ is called the **symplectic gradient field** (or the **Hamiltonian vector field**) of $f \in C^\infty(P)$, denoted by $\text{sgrad } f := \vec{X}^f$.

Definition A.5 (Hamiltonian system). A **Hamiltonian dynamical system** (P, H) for a Hamiltonian function $H \in C^\infty(P)$ on a Poisson manifold P is a time-dependent position $x(t) \in P$ governed by the ODE $\frac{d}{dt} x(t) = -\text{sgrad } H|_{x(t)}$.

A.2.1 Properties of sgrad. The operator sgrad is derived directly from the Poisson structure $\{\cdot, \cdot\}$. It turns each Hamiltonian function into a Hamiltonian dynamical system. The following is an important property of sgrad .

Proposition A.3. The symplectic gradient operator

$$\text{sgrad}: (C^\infty(P), \{\cdot, \cdot\}) \rightarrow (\mathfrak{X}(P), [\cdot, \cdot]) \quad (96)$$

is a Lie algebra homomorphism.

PROOF. Observe that sgrad is the currying of the Poisson bracket $(\{f, \cdot\} = \text{sgrad}(f) \triangleright)$ analogous to that ad is the currying of a Lie bracket. The proof follows by replicating the proof of Proposition A.1. \square

Definition A.6. The space $\mathfrak{ham}(P)$ of Hamiltonian vector fields on P is the Lie subalgebra $\mathfrak{ham}(P) = \text{im}(\text{sgrad}) \subset \mathfrak{X}(P)$.

Note that the “integration” operator (defined up to a Casimir function which we will introduce later in Definition A.10)

$$\text{sgrad}^{-1}: (\mathfrak{ham}(P), [\cdot, \cdot]) \rightarrow (C^\infty(P), \{\cdot, \cdot\}) \quad (97)$$

is also a Lie algebra homomorphism. This will be convenient when we discuss momentum maps.

A.2.2 Poisson Maps. A Poisson map is a Poisson-structure preserving map. As such Hamiltonian systems map to each other.

Definition A.7. A map $\phi: P_1 \rightarrow P_2$ between two Poisson manifolds $(P_1, \{\cdot, \cdot\}_1)$ and $(P_2, \{\cdot, \cdot\}_2)$ is called a **Poisson map** if $\{f \circ \phi, g \circ \phi\}_1 = \{f, g\}_2 \circ \phi$ for all $f, g \in C^\infty(P_2)$. The map is called an **anti-Poisson map** if $\{f \circ \phi, g \circ \phi\}_1 = -\{f, g\}_2 \circ \phi$.

Proposition A.4. Let $\phi: (P_1, \{\cdot, \cdot\}_1) \rightarrow (P_2, \{\cdot, \cdot\}_2)$ be a Poisson map. Let $H \in C^\infty(P_2)$ be a Hamiltonian function defining a Hamiltonian system (P_2, H) . If $x(t) \in P_1$ is a solution to the dynamical system $(P_1, H \circ \phi)$ then $(\phi \circ x)(t)$ is a solution to the system (P_2, H) . If ϕ is an anti-Poisson map instead, then $\phi \circ x$ is a solution to the system (P_2, H) but reversed in time.

A.3 Symplectic Manifolds are Poisson

In elementary classical mechanics, the phase space of a mechanical system is introduced as the space of positions q and the corresponding momenta p . These position–momentum (qp) phase spaces are *symplectic spaces* or, more generally, *symplectic manifolds*. Indeed, symplectic manifolds are special cases of Poisson manifolds.

Definition A.8. A **symplectic manifold** is a manifold Σ equipped with a non-degenerate 2-form $\sigma \in \Omega^2(\Sigma)$.

By currying, the symplectic structure σ gives rise to a “flat” operator (analogous to the \flat, \sharp for an inner product structure)

$$\hat{b}_x: T_x \Sigma \xrightarrow{\text{linear}} T_x^* \Sigma, \quad \hat{b}_x(\vec{v})(\cdot) = \sigma_x(\vec{v}, \cdot), \quad x \in \Sigma, \vec{v} \in T_x \Sigma. \quad (98)$$

The non-degeneracy of σ means that \hat{b} is invertible, defining $\hat{b}_x^{-1}: T_x^* \Sigma \rightarrow T_x \Sigma$. Using a symplectic structure, one defines the symplectic gradient of a function $f \in C^\infty(\Sigma)$ by

$$\text{sgrad} f = \hat{b}^{-1}(df), \quad \text{equivalently} \quad i_{\text{sgrad} f} \sigma = df, \quad (99)$$

where $i_{(\cdot)}(\cdot)$ is the interior product. Every symplectic structure gives rise to a natural Poisson structure:

$$\{f, g\} = -\sigma(\text{sgrad} f, \text{sgrad} g) \quad (100a)$$

$$= -df \llbracket \text{sgrad} g \rrbracket = dg \llbracket \text{sgrad} f \rrbracket. \quad (100b)$$

Definition A.9. A vector field $\vec{X} \in \mathfrak{X}(\Sigma)$ is **symplectomorphic** if $\mathcal{L}_{\vec{X}} \sigma = 0$. That is, the flow $\varphi^t: \Sigma \rightarrow \Sigma$ generated by \vec{X} ($\frac{\partial}{\partial t} \varphi^t = \vec{X} \circ \varphi^t, \varphi^0 = \text{id}_\Sigma$) preserves the symplectic form $\varphi^{t*} \sigma = \sigma$.

Proposition A.5. All Hamiltonian vector fields are symplectomorphic (Liouville theorem). On simply connected symplectic manifolds, all symplectomorphic vector fields are also Hamiltonian.

PROOF. By Cartan’s formula $\mathcal{L}_{\text{sgrad} f} \sigma \stackrel{d\sigma=0}{=} di_{\text{sgrad} f} \sigma \stackrel{(99)}{=} ddf = 0$. Conversely, if $\mathcal{L}_{\vec{X}} \sigma = 0$, then $di_{\vec{X}} \sigma = 0$. On simply-connected

Σ this implies that there exists f so that $df = i_{\vec{X}} \sigma$; that is, $\vec{X} = \text{sgrad} f$. \square

Example A.3 (qp-space). The space of positions $\mathbf{q} \in \mathbb{R}^m$ and momenta $\mathbf{p} \in \mathbb{R}^m$ of a generic finite dimensional mechanical system form a symplectic space $\Sigma = \mathbb{R}^{2m} \ni (\mathbf{q}, \mathbf{p})$ with symplectic form $\sigma = \sum_{i=1}^m dp_i \wedge dq_i$. Note that $\sigma = d\vartheta$ where $\vartheta = \sum_{i=1}^m p_i dq_i$ is called the Liouville 1-form. Now, given any function $H \in C^\infty(P)$, the Hamiltonian dynamical system (Definition A.5) is given by $\frac{d}{dt} q_i(t) = \frac{\partial H}{\partial p_i}, \frac{d}{dt} p_i(t) = -\frac{\partial H}{\partial q_i}$. One checks that this $\text{sgrad} H$ satisfies $i_{\text{sgrad} H} \sigma = dH = \sum_{i=1}^m \frac{\partial H}{\partial q_i} dq_i + \frac{\partial H}{\partial p_i} dp_i$.

Example A.4 (Cotangent bundles). The coordinate-free version of Example A.3 is the cotangent bundle $\Sigma = T^*Q$ over a space Q of positions. Each element $(q, p) \in T^*Q$ takes the form of $q \in Q$ and $p \in T_q^*Q$. Let $\pi: T^*Q \rightarrow Q, (q, p) \mapsto q$, denote the projection from the bundle to the base. Then we can define the Liouville 1-form $\vartheta \in \Omega^1(\Sigma)$ by $\vartheta_{(q,p)}(\vec{X}) = \langle p | d\pi(\vec{X}) \rangle$ for $\vec{X} \in T_{(q,p)}(\Sigma)$. The symplectic form σ for Σ is given by $\sigma = d\vartheta$.

A.3.1 Lifted Map and Lifted Vector Field. There is a general way for constructing a symplectomorphic map on a cotangent bundle T^*Q . Suppose $\varphi: Q \rightarrow Q$ is a map on Q . Then its **lifted map**, given by $\tilde{\varphi}: T^*Q \rightarrow T^*Q, \tilde{\varphi}(q, p) = (\varphi(q), (\varphi^{-1})^* p)$, is a symplectomorphism. That is, $\tilde{\varphi}^* \sigma = \sigma$.

Similarly, each vector field $\vec{X} \in \mathfrak{X}(Q)$ can be lifted into a symplectomorphic vector field $\vec{\tilde{X}} \in \mathfrak{X}(T^*Q)$, whose flow map is the lifted map of the flow map generated by \vec{X} . Concretely, $\vec{\tilde{X}} = -\text{sgrad} H$ where $H: T^*Q \rightarrow \mathbb{R}, H(q, p) = \langle p | \vec{X} \rangle$.

A.4 Poisson Manifolds have Symplectic Foliations

We have seen that every symplectic manifold is Poisson. What about the converse? It turns out that every Poisson manifold is foliated into (i.e. decomposed into a disjoint union of) symplectic submanifolds, as discussed below.

If one is given a Poisson manifold $(P, \{\cdot, \cdot\})$, one can reconstruct the \hat{b} operator so that the symplectic gradient of Definition A.4 can be represented by $\text{sgrad} f = \hat{b} df$. If the \hat{b} operator from a Poisson structure is invertible, then the Poisson manifold is symplectic with symplectic structure $\hat{b} = \hat{b}^{-1}$. A more general Poisson manifold may have a non-invertible \hat{b} . Remarkably, even when \hat{b} is not invertible, the subspace field $\text{im}(\hat{b}_x) \subset T_x P$ is integrable; i.e. P is foliated into submanifolds with $\text{im}(\hat{b}_x)$ being the tangent spaces of these submanifolds. These submanifolds are symplectic since \hat{b} is invertible on their tangent spaces. Each of these submanifolds is called a **symplectic leaf**.

Note that since $\text{sgrad} f = \hat{b} df$, we have $\text{sgrad} f \in \text{im}(\hat{b})$ tangential to the symplectic leaves. Therefore, every Hamiltonian dynamical system on a Poisson manifold stays in a symplectic leaf.

Definition A.10 (Casimir). A function $C: P \rightarrow \mathbb{R}$ on a Poisson manifold P is called a **Casimir** if it is constant on each symplectic leaf.

Proposition A.6. Each Casimir function is conserved under any Hamiltonian dynamical system on a Poisson manifold.

PROOF. Every Hamiltonian flow stays in a symplectic leaf, on which any Casimir function is constant. \square

A.5 Dual Lie Algebras are Poisson

Lie algebra: $(V, [\cdot, \cdot])$. Dual Lie algebra: its dual vector space V^* . The Lie–Poisson structure:

$$\{\cdot, \cdot\}: C^\infty(V^*) \times C^\infty(V^*) \rightarrow C^\infty(V^*) \quad (101a)$$

$$\{f, g\}_\alpha := \left\langle \alpha \mid [df_\alpha, dg_\alpha] \right\rangle, \quad \alpha \in V^*, f, g \in C^\infty(V^*). \quad (101b)$$

Here $df_\alpha \in T_\alpha^*V^*$ (and dg_α) is the differential of f at α , which can be viewed as an element of V (by $T_\alpha^*V^* = V^{**} = V$). Thus it makes sense to take $[\cdot, \cdot]$ between df_α, dg_α , returning another element in V , which is finally paired with the covector $\alpha \in V^*$.

Example A.5 (Dual space of functions on a Poisson space). *Here is an example of a dual Lie algebra. Recall that $C^\infty(P)$ of a Poisson space $(P, \{\cdot, \cdot\})$ is a Lie algebra with Lie bracket being $\{\cdot, \cdot\}$. Its dual space, $C^\infty(P)^*$, is the space of distributions. A subset of distributions is the space of point measures, identified with P itself, $P \subset C^\infty(P)^*$ with the dual pairing being the point evaluation. The Lie–Poisson structure (101) on $P \subset C^\infty(P)^*$ as in the dual Lie algebra is just the original Poisson structure on P .*

Proposition A.7. *The symplectic gradient $\text{sgrad } f \in \mathfrak{X}(V^*)$ of a function $f: V^* \rightarrow \mathbb{R}$ on the dual Lie algebra V^* with the Lie–Poisson bracket (101) is given by*

$$\text{sgrad } f|_\alpha = \text{ad}_{df_\alpha}^\top \alpha, \quad \alpha \in V^* \quad (102)$$

PROOF. Using the adjoint representation of Definition A.2, rewrite (101b) as $\{f, g\}_\alpha = \langle \alpha \mid \text{ad}_{df_\alpha}(dg_\alpha) \rangle = \langle \text{ad}_{df_\alpha}^\top \alpha \mid dg_\alpha \rangle = \text{ad}_{df_\alpha}^\top \triangleright g$. Eq. (102) follows by Definition A.4. \square

Definition A.11 (Coadjoint action). *Let $\vec{v} \in V$ be an element of a Lie algebra V . The operator $\text{ad}_{\vec{v}}^\top: V^* \rightarrow V^*$ is an **infinitesimal coadjoint action** by \vec{v} .*

Definition A.12 (Coadjoint orbits). *Let $\alpha_0 \in V^*$ be a point in a dual Lie algebra V^* . Consider the set O_{α_0} of all possible endpoints $\gamma(1)$ of arbitrary paths $\gamma: [0, 1] \rightarrow V^*$ starting at $\gamma(0) = \alpha_0$ so that $\gamma'(t) = \text{ad}_{\vec{v}(t)}^\top(\gamma(t))$ for some $\vec{v}(t) \in V$. That is, O_{α_0} is the set of points that can be reached by infinitesimal coadjoint actions. The set $O_{\alpha_0} \subset V^*$ is called the **coadjoint orbit** of α_0 .*

The following proposition is an immediate consequence of (102).

Proposition A.8. *Any path of a Hamiltonian dynamical system on any dual Lie algebra stays on a coadjoint orbit.*

PROOF. Hamiltonian dynamical systems (Definition A.5) is in the form of coadjoint actions by (102). Therefore they generate paths that lie in coadjoint orbits. \square

In fact, coadjoint orbits are exactly the symplectic leaves.

Proposition A.9. *Coadjoint orbits coincide with the symplectic leaves in any dual Lie algebra.*

PROOF. At each point $\alpha \in V^*$, the subspace $\{\text{sgrad } f|_\alpha \mid f \in C^\infty(V^*)\}$ coincides with $\text{im}(\hat{\mu}_\alpha)$ and, by (102), $\{\text{ad}_{\vec{v}}^\top \alpha \mid \vec{v} \in V\}$. $\text{im}(\hat{\mu}_\alpha)$

integrates into symplectic leaves, and $\{\text{ad}_{\vec{v}}^\top \alpha \mid \vec{v} \in V\}$ integrates into coadjoint orbits. \square

Note that these Poisson structures and coadjoint orbit structures in a dual Lie algebra are canonically derived directly from the Lie algebra structure. An immediate consequence is the following proposition.

Proposition A.10. *Let U, V be two Lie algebras. If $A: U \rightarrow V$ is a Lie algebra homomorphism, then $A^\top: V^* \rightarrow U^*$ is a Poisson map. Likewise if $A: U \rightarrow V$ is a Lie algebra anti-homomorphism, then $A^\top: V^* \rightarrow U^*$ is an anti-Poisson map.*

A.6 Momentum Maps

Let P be a Poisson space. Let V be a linear subspace of Hamiltonian vector fields (Definition A.6) on P with the inclusion map

$$I: V \xrightarrow{\text{linear}} \mathfrak{ham}(P). \quad (103)$$

Definition A.13 (Momentum map). *A **momentum map** associated to $I: V \hookrightarrow \mathfrak{ham}(P)$ is a function $J: P \rightarrow V^*$ so that $\langle J(\cdot) \mid \vec{v} \rangle: P \rightarrow \mathbb{R}$ is the Hamiltonian function that generates the Hamiltonian vector field $I(\vec{v})$ for all \vec{v} . That is,*

$$-\text{sgrad} \langle J \mid \vec{v} \rangle = I(\vec{v}), \quad \vec{v} \in V. \quad (104)$$

An explicit formula for a momentum map is

$$J = -I^\top \circ \text{sgrad}^{-\top}. \quad (105)$$

To understand, recall sgrad^{-1} is a map of type $\text{sgrad}^{-1}: \mathfrak{ham}(P) \rightarrow C^\infty(P)$, which is a Lie algebra homomorphism (cf. (97)). In particular, $\text{sgrad}^{-\top}$ is a Poisson map from distributions $C^\infty(P)^*$ to $\mathfrak{ham}(P)^*$ (Proposition A.10). Here we restrict $\text{sgrad}^{-\top}$ to $P \subset C^\infty(P)^*$; see Example A.5.

Proposition A.11 (Momentum maps for Lie subalgebras are Poisson). *If V is an (anti-)Lie subalgebra of $\mathfrak{ham}(P)$, i.e. I is a Lie algebra (anti-)homomorphism, then the momentum map J (105) is an (anti-)Poisson map.*

PROOF. J is a composition of a Poisson map $\text{sgrad}^{-\top}$ and an (anti-)Poisson map I^\top (Proposition A.10). \square

Example A.6 (Momentum maps for lifted vector fields on cotangent bundles). *Let P be a cotangent bundle T^*Q . Let $V \subset \mathfrak{X}(Q)$ be a subspace of vector fields on Q and let $I: V \rightarrow \mathfrak{ham}(T^*Q)$, $\vec{v} \mapsto \tilde{\vec{v}}$, be lifted vector fields (Section A.3.1). Then we can write down the associated momentum map J explicitly as $\langle J(q, p) \mid \vec{v} \rangle = \vartheta_{(q,p)}[\![I(\vec{v})]\!] = \langle p \mid \vec{v} \rangle$, where ϑ is the Liouville form (Example A.4).*

Example A.7 (Momentum map associated to symplectomorphic group actions). *Let (Σ, σ) be a symplectic manifold. Let $\Phi: G \xrightarrow{\text{hom}} (\Sigma \rightarrow \Sigma)$ be a left action on Σ by a Lie group G so that $\Phi_g^* \sigma = \sigma$ for all $g \in G$. That is, the group action produces symplectomorphic maps $\Phi_g: \Sigma \rightarrow \Sigma$. Then $d\Phi|_e: \mathfrak{g} \rightarrow \mathfrak{X}(\Sigma)$ is a Lie algebra anti-homomorphism (Proposition A.2). Moreover, the image of $d\Phi|_e$ is always a symplectomorphic vector field (Definition A.9). Assuming Σ being simply-connected, by Proposition A.5, these symplectomorphic vector fields are Hamiltonian vector fields. That is, we have a Lie algebra anti-homomorphism $I := d\Phi|_e: \mathfrak{g} \rightarrow \mathfrak{ham}(\Sigma)$. Therefore*

the associated momentum map $J: \Sigma \rightarrow \mathfrak{g}^*$ is an anti-Poisson map (Proposition A.11).

Combining (104) and (99), we have that the momentum maps for symplectomorphic actions satisfy

$$-d\langle J|\vec{v} \rangle = i_{I(\vec{v})}\sigma, \quad \vec{v} \in \mathfrak{g}. \quad (106)$$

Example A.8 (Momentum map associated to group actions on cotangent bundles). Let $\Phi: G \xrightarrow{\text{hom}} (Q \rightarrow Q)$ be a left action on Q by a Lie group G . Then $d\Phi|_e: \mathfrak{g} \rightarrow \mathfrak{X}(Q)$ is a Lie algebra anti-homomorphism (Proposition A.2). The vector fields can be lifted to the cotangent bundle using Section A.3.1 and Example A.6, giving us a Lie algebra anti-homomorphism $I := d\Phi|_e: \mathfrak{g} \rightarrow \mathfrak{ham}(T^*Q)$. Therefore, the associated momentum map $J: T^*Q \rightarrow \mathfrak{g}^*$ is an anti-Poisson map (Proposition A.11).

B RELATION TO NON-HOLONOMICITY

Here we expand the discussion on the relation between the lack of Poisson structure for the dual space $\mathfrak{B}_{\text{div}}^*$ in Section 4.2.1 and the non-holonomicity problem in the literature.

The problem here, that the space of discrete vector fields is not Lie algebraically closed, has been observed in previous work in computer graphics that searches for structure preserving discretizations of the Euler equation. In the setup of [Pavlov et al. 2011; Mullen et al. 2009], the discretization is constructed by replacing the function space $C^\infty(W)$ by a finite dimensional function space \mathbb{R}^N (e.g. the piecewise linear functions over a simplicial mesh with N vertices). This leads to that the Lie algebra $\mathfrak{X}_{\text{div}}(W)$ is replaced by a finite dimensional Lie algebra $\mathfrak{so}(N)$ of stochastic matrices as the generators of volume-preserving diffeomorphisms. Although $\mathfrak{so}(N)$ is a Lie algebra, the only vector fields that are computationally meaningful (representing discrete directional derivative operators) are restricted to a subspace $S \subset \mathfrak{so}(N)$ characterized by an additional sparsity condition on the stochastic matrices. This sparsity constraint is referred to as a *non-holonomic constraint* [Pavlov et al. 2011, § 2.4, § 4.1], which is equivalent to that the subspace S is not Lie algebraically closed. The same non-holonomic constraint is present in the spectral variational integrator of [Liu et al. 2015, § 3.2]. In these works, the resulting equations of motion are no longer pure coadjoint actions on the circulation variables like (26). Instead, these equations are in their *weak form*, where the circulation variable is restricted to stay on S^* and the coadjoint equation is only tested against vectors in S . In particular, the coadjoint orbit conservation is not guaranteed, as remarked at the end of § 4 of [Pavlov et al. 2011]. This is consistent with our observation that the dual space S^* of a non-holonomic velocity space S is not a Poisson space, since otherwise S^* would have been foliated into coadjoint orbits conserved by the methods of [Mullen et al. 2009; Pavlov et al. 2011; Liu et al. 2015].

One structure-preserving method in computer graphics that is able to circumvent the non-holonomic projection is the 2D vorticity method of Azencot *et al.* [2014]. This method follows the setup of [Pavlov et al. 2011] that the space of velocity is a non-holonomic $S \subset \mathfrak{so}(N)$. However, instead of formulating a dynamical system for elements in S^* , the phase space is the space of vorticity function \mathbb{R}^N , which is acted by unitary matrices generated by elements in

$S \subset \mathfrak{so}(N)$. In particular, we obtain exact conservation on the ℓ^2 -norm of the \mathbb{R}^N vorticity, and indeed the ℓ^2 -norm on \mathbb{R}^N is one of the Casimirs. Our method uses a similar technique (Section 4.3) of maintaining an advected object to obtain coadjoint orbit preservation, but this time on the original fluid's Lie algebra $\mathfrak{X}_{\text{div}}$ rather than on $\mathfrak{so}(N)$ and the method works in both 2D and 3D.

C PROOFS

C.1 Derivation of (12)

The orthogonal projection $\mathfrak{g} = \mathbf{P}_{\mathfrak{B}_{\text{div}}}$ of $\mathfrak{f} \in \mathfrak{B}$ to $\mathfrak{B}_{\text{div}} = \ker(\mathbf{d})$ is the solution to

$$\text{minimize}_{\mathfrak{g}} \frac{1}{2} (\mathfrak{g} - \mathfrak{f})^\top \star (\mathfrak{g} - \mathfrak{f}) \quad \text{subject to} \quad \mathbf{d}\mathfrak{g} = \mathbf{0} \quad (107)$$

The minimizing condition is given by $\star(\mathfrak{g} - \mathfrak{f}) + \mathbf{d}^\top \mathfrak{p} = \mathbf{0}$ for some Lagrange multiplier \mathfrak{p} . Solving the system

$$\begin{cases} \star(\mathfrak{g} - \mathfrak{f}) + \mathbf{d}^\top \mathfrak{p} = \mathbf{0} \\ \mathbf{d}\mathfrak{g} = \mathbf{0} \end{cases} \quad (108)$$

yields $\mathfrak{p} = (\mathbf{d} \star^{-1} \mathbf{d}^\top)^{-1} \mathbf{d}\mathfrak{f}$ and $\mathfrak{g} = \mathfrak{f} - \star^{-1} \mathbf{d}^\top \mathfrak{p} = \mathfrak{f} - \star^{-1} \mathbf{d}^\top (\mathbf{d} \star^{-1} \mathbf{d}^\top)^{-1} \mathbf{d}\mathfrak{f}$. Therefore, $\mathbf{P}_{\mathfrak{B}_{\text{div}}} = \text{id} - \star^{-1} \mathbf{d}^\top (\mathbf{d} \star^{-1} \mathbf{d}^\top)^{-1} \mathbf{d}$. \square

C.2 Proof of Theorem 3.1

In this proof we will repeatedly using the linear algebraic fact that $\langle \mathbf{P}\mathbf{a}, \mathbf{b} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle$ if \mathbf{P} is an orthogonal projection to a subspace B respecting the inner product structure $\langle \cdot, \cdot \rangle$, and \mathbf{b} lies in the subspace B .

Let $\mathcal{I}\mathfrak{g} \in \mathcal{I}(\mathfrak{B}_{\text{div}})$ be an arbitrary interpolated divergence-free vector field. Then

$$\langle \mathcal{I}\mathbf{P}_{\mathfrak{B}_{\text{div}}}(\mathfrak{f}) - \mathbf{P}_{\mathfrak{X}_{\text{div}}}\mathcal{I}(\mathfrak{f}), \mathcal{I}(\mathfrak{g}) \rangle \quad (109)$$

$$= \langle \mathcal{I}\mathbf{P}_{\mathfrak{B}_{\text{div}}}(\mathfrak{f}), \mathcal{I}(\mathfrak{g}) \rangle - \langle \mathbf{P}_{\mathfrak{X}_{\text{div}}}\mathcal{I}(\mathfrak{f}), \mathcal{I}(\mathfrak{g}) \rangle \quad (110)$$

$$= \langle \mathbf{P}_{\mathfrak{B}_{\text{div}}}(\mathfrak{f}), \mathfrak{g} \rangle_{\mathfrak{B}} - \langle \mathcal{I}(\mathfrak{f}), \mathcal{I}(\mathfrak{g}) \rangle \quad (111)$$

$$= \langle \mathfrak{f}, \mathfrak{g} \rangle_{\mathfrak{B}} - \langle \mathfrak{f}, \mathfrak{g} \rangle_{\mathfrak{B}} = 0, \quad (112)$$

where we have used that $\mathbf{P}_{\mathfrak{X}_{\text{div}}}$ is an orthogonal projection respecting $\langle \cdot, \cdot \rangle$, $\mathbf{P}_{\mathfrak{B}_{\text{div}}}$ is an orthogonal projection with respect to $\langle \cdot, \cdot \rangle_{\mathfrak{B}}$, and $\langle \cdot, \cdot \rangle_{\mathfrak{B}} = \langle \mathcal{I}(\cdot), \mathcal{I}(\cdot) \rangle$. Therefore, $\mathcal{I}\mathbf{P}_{\mathfrak{B}_{\text{div}}}(\mathfrak{f}) - \mathbf{P}_{\mathfrak{X}_{\text{div}}}\mathcal{I}(\mathfrak{f})$ is orthogonal to $\mathcal{I}(\mathfrak{B}_{\text{div}})$. \square

C.3 Proof of Theorem 3.2

In (15) if $\vec{u}_{\mathfrak{p}}$ is a sampling of a vector field $\vec{u} = \mathcal{I}(\mathfrak{f})$ in $\mathcal{I}(\mathfrak{B})$ at the particles, then the minimization problem $\min_{\mathfrak{f}' \in \mathfrak{B}} \sum_{\mathfrak{p} \in \mathfrak{B}} |\mathcal{I}(\mathfrak{f}')_{\vec{x}_{\mathfrak{p}}} - \mathcal{I}(\mathfrak{f})_{\vec{x}_{\mathfrak{p}}}|^2$ can attain zero by $\mathfrak{f}' = \mathfrak{f}$ uniquely, assuming that the minimization is unique guaranteed by sufficiently many samples. Therefore $\hat{\mathcal{I}}^+(\vec{x}, \mathcal{I}(\mathfrak{f}))$ recovers \mathfrak{f} . \square

C.4 Proof of Proposition 4.1

Represent div as a map to the space $\Omega^n(W)$ of measures $\mathfrak{X}(W) \xrightarrow{\text{div}} \Omega^n(W)$, by $\text{div} \vec{v} = \mathcal{L}_{\vec{v}}(d\mu) = d_{\vec{v}}(d\mu) \in \Omega^n(W)$ where $d\mu$ is the volume form. Then $0 \rightarrow \mathfrak{X}_{\text{div}}(W) \hookrightarrow \mathfrak{X}(W) \xrightarrow{\text{div}} \Omega^n(W)$ is an exact sequence. Take the dual of this sequence yields another exact sequence $C^0(W) \xrightarrow{d} \Omega^1(W) \rightarrow \mathfrak{X}_{\text{div}}^*(W) \rightarrow 0$. One can check that $\pm d$ is the adjoint of div . By the first-isomorphism theorem we conclude $\mathfrak{X}_{\text{div}}^*(W) = \Omega^1(W)/d\Omega^0(W)$. \square

C.5 Proof of Proposition 4.2

The (minus) Lie–Poisson bracket (101) on $\mathfrak{X}_{\text{div}}^*(W)$ is given by that for each functionals $H, G: \mathfrak{X}_{\text{div}}^*(W) \rightarrow \mathbb{R}$ and $[\eta] \in \mathfrak{X}_{\text{div}}^*(W)$

$$\{H, G\}_{[\eta]} = - \left\langle [\eta] \left| \left[\frac{\delta H}{\delta [\eta]}, \frac{\delta G}{\delta [\eta]} \right] \right. \right\rangle \quad (113)$$

$$= - \int_W \left\langle \eta \left| \left[\frac{\delta H}{\delta [\eta]}, \frac{\delta G}{\delta [\eta]} \right] \right. \right\rangle d\mu \quad (114)$$

$$\begin{aligned} &= - \underbrace{\int_W \mathcal{L}_{\frac{\delta H}{\delta [\eta]}} \left\langle \eta \left| \frac{\delta G}{\delta [\eta]} \right. \right\rangle d\mu + \int_W \left\langle \mathcal{L}_{\frac{\delta H}{\delta [\eta]}} \eta \left| \frac{\delta G}{\delta [\eta]} \right. \right\rangle d\mu}_{= \int_W \langle \eta | \frac{\delta G}{\delta [\eta]} \rangle \mathcal{L}_{\frac{\delta H}{\delta [\eta]}}(d\mu) = 0} \\ & \quad (115) \end{aligned}$$

Therefore the symplectic gradient (Definition A.4) sgrad H of H is

$$\text{sgrad } H|_{[\eta]} = [\mathcal{L}_{\frac{\delta H}{\delta [\eta]}}] = \mathcal{L}_{\frac{\delta H}{\delta [\eta]}}[\eta]. \quad (116)$$

Hence the Hamiltonian dynamical system (Definition A.5) $\frac{\partial}{\partial t}[\eta] = -\text{sgrad } H|_{[\eta]}$ is given by $\frac{\partial}{\partial t}[\eta] + \mathcal{L}_{\frac{\delta H}{\delta [\eta]}}[\eta] = [0]$. \square

C.6 Proof of Proposition 4.3

Based on the calculation in the proof of Proposition 4.2 (Appendix C.5), we have that the coadjoint action $\text{ad}_{\vec{v}}^{\top}: \mathfrak{X}_{\text{div}}^*(W) \rightarrow \mathfrak{X}_{\text{div}}^*(W)$ for each $\vec{v} \in \mathfrak{X}_{\text{div}}(W)$ is given by

$$\text{ad}_{\vec{v}}^{\top}[\eta] = -\mathcal{L}_{\vec{v}}[\eta]. \quad (117)$$

We want to show, by Definition A.12, that (i) the resulting $[\eta_1]$ that solves $\frac{\partial}{\partial t}[\eta_t] + \mathcal{L}_{\vec{v}(t)}[\eta_t] = [0]$ with given $[\eta_0]$ and $\vec{v}: [0, 1] \in \mathfrak{X}_{\text{div}}(W)$ can be expressed by $[\eta_0] = [\varphi^* \eta_1]$ for some $\varphi: W \rightarrow W$ isotopic to the identity map, and (ii) the converse statement.

For (i), consider $\varphi_t: W \rightarrow W$, $\frac{\partial}{\partial t} \varphi_t = \vec{v}(t) \circ \varphi_t$, $\varphi_0 = \text{id}_W$, be the flow map integrated by the given time-dependent $\vec{v}(t)$ from $t = 0$ to $t = 1$. Then $[\eta_t] := [(\varphi_t^{-1})^* \eta_0]$ is the solution to $\frac{\partial}{\partial t}[\eta_t] + \mathcal{L}_{\vec{v}(t)}[\eta_t] = [0]$. Therefore, at $t = 1$ we have $[\eta_1] = [(\varphi^{-1})^* \eta_0]$, which implies $[\eta_0] = [\varphi^* \eta_1]$.

Conversely (ii), suppose $\varphi: W \rightarrow W$ is isotopic to id_W . Then there exists a smooth family of maps $\varphi_t: W \rightarrow W$ so that $\varphi_0 = \text{id}_W$ and $\varphi_1 = \varphi$. Let $\vec{v}(t)$ be the family of vector fields so that $\frac{\partial}{\partial t} \varphi_t = \vec{v}(t) \circ \varphi_t$. Then $[\eta_t] = [(\varphi_t^{-1})^* \eta_0]$, with $[\eta_0] = [\varphi^* \eta_1]$ as a result, is the solution to $\frac{\partial}{\partial t}[\eta_t] + \mathcal{L}_{\vec{v}(t)}[\eta_t] = [0]$. Therefore $[\eta_0]$ can reach $[\eta_1]$ by a continuous sequence of coadjoint action, and thus they lie on the same coadjoint orbit. \square

C.7 Proof of Proposition 4.4

The variation of (29) is given by

$$\frac{\delta H_{\text{Euler}}}{\delta [\eta]} = \sharp_{\mathfrak{X}_{\text{div}}}[\eta] = \vec{v} \in \mathfrak{X}_{\text{div}}(W) \quad (118)$$

which is the pressure projected η^{\sharp} (cf. (25)). Substituting (118) to (26) yields the Euler equation (4) or (5). \square

C.8 Proof of Theorem 4.1

The Hamiltonian (34) (with (32)) is given by

$$(H_{\text{D}} \circ I^{\top})([\eta]) = \frac{1}{2} \langle \sharp_{\mathfrak{B}_{\text{div}}} I^{\top}[\eta], \sharp_{\mathfrak{B}_{\text{div}}} I^{\top}[\eta] \rangle_{\mathfrak{B}} \quad (119)$$

$$= \frac{1}{2} \langle \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top}[\eta] \rangle^{\top} \star \langle \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top}[\eta] \rangle. \quad (120)$$

Its variation is given by

$$\frac{\delta(H_{\text{D}} \circ I^{\top})}{\delta [\eta]} = I \star^{-1} \mathbf{P}_{\mathfrak{B}_{\text{div}}}^{\top} \star \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top}[\eta]. \quad (121)$$

Note that since $\mathbf{P}_{\mathfrak{B}_{\text{div}}}$ is an orthogonal projection with respect to the inner product structure \star , we have the self-adjoint property $\star^{-1} \mathbf{P}_{\mathfrak{B}_{\text{div}}}^{\top} \star = \mathbf{P}_{\mathfrak{B}_{\text{div}}}$. Then by idempotence of projection $\mathbf{P}_{\mathfrak{B}_{\text{div}}}^2 = \mathbf{P}_{\mathfrak{B}_{\text{div}}}$, we obtain

$$\frac{\delta(H_{\text{D}} \circ I^{\top})}{\delta [\eta]} = I \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top}[\eta]. \quad (122)$$

Substituting (122) to (26) we obtain (35). \square

C.9 Theorem 4.2

This is a direct application of Proposition A.4 on $J_{\text{adv}}: \Sigma \rightarrow \mathfrak{X}_{\text{div}}^*(W)$ being an (anti-)Poisson map.

C.10 Proof of Theorem 4.3

By (106), with (38) $\text{adv}: \mathfrak{X}_{\text{div}}(W) \rightarrow \mathfrak{ham}(\Sigma) \subset \mathfrak{X}(\Sigma)$ being the infinitesimal group action, we have

$$-d\langle J_{\text{adv}}(\cdot) | \vec{w} \rangle = i_{\text{adv}_{\vec{w}}(s)} \sigma \quad \forall \vec{w} \in \mathfrak{X}_{\text{div}}(W). \quad (123)$$

Note that we may express the left-hand side as $-\langle dJ_{\text{adv}} | \vec{w} \rangle$. Now let us take the derivative of the Hamiltonian $H_{\Sigma} = H_{\text{D}} \circ I^{\top} \circ J_{\text{adv}}$:

$$dH_{\Sigma} = \frac{\delta(H_{\text{D}} \circ I^{\top})}{\delta [\eta]} \Big|_{[\eta]=J_{\text{adv}}(s)} dJ_{\text{adv}}|_s \quad (124)$$

$$= \langle dJ_{\text{adv}} | \vec{v} \rangle, \quad \vec{v} = I \mathbf{P}_{\mathfrak{B}_{\text{div}}} \star^{-1} I^{\top} J_{\text{adv}}(s) \quad (125)$$

using (122). Now apply (123), which yields

$$dH_{\Sigma} = -i_{\text{adv}_{\vec{v}}(s)} \sigma. \quad (126)$$

Therefore by (99) we have

$$\text{sgrad } H_{\Sigma}|_s = -\text{adv}_{\vec{v}} s. \quad (127)$$

Hence the Hamiltonian flow (Definition A.5) for s is given by (40). \square

C.11 Proofs of Lemma 4.1 and Lemma 4.2

The minimizer $\mathbf{f} \in \mathfrak{B}$ for $\int_W |I(\mathbf{f}) - \eta^{\sharp}|^2 d\mu$ satisfies the optimality condition $\langle I(\mathbf{f}) - \eta^{\sharp}, I(\mathbf{f}) \rangle = 0$ for all $\mathbf{f} \in \mathfrak{B}$. Since $I: (\mathfrak{B}, \langle \cdot, \cdot \rangle_{\mathfrak{B}}) \rightarrow (\mathfrak{X}(W), \langle \cdot, \cdot \rangle)$ is an isometry, and $\sharp: \Omega^1(W) \rightarrow \mathfrak{X}(W)$ respects the L^2 product $\langle \cdot, \cdot \rangle$, the condition becomes that for all $\mathbf{f} \in \mathfrak{B}$

$$0 = \langle \mathbf{f}, \mathbf{f} \rangle_{\mathfrak{B}} - \langle \eta | I(\mathbf{f}) \rangle \quad (128)$$

$$= \mathbf{f}^{\top} \star \mathbf{f} - \langle I^{\top} \eta | \mathbf{f} \rangle \quad (129)$$

$$= \mathbf{f}^{\top} (\star \mathbf{f} - I^{\top} \eta). \quad (130)$$

Therefore $\mathbf{f} = \star^{-1} I^{\top} \eta$, showing (50).

The proof for (51) is the same as the one for (50) with the substitution $\eta = J_{\text{adv}}(\vec{x}, \vec{u})$. \square

C.12 Proof of Theorem 4.5

The proof has two parts. First we show that states $J_{\text{adv}}(s^{(n)})$ stay on the same coadjoint orbit. Second, we show that $\mathcal{E}^{(n)} = \mathcal{E}(s^{(n)}) := H_D \circ I^\top \circ J_{\text{adv}}(s^{(n)})$ is n-invariant.

C.12.1 Coadjoint Orbit Conservation. To show the coadjoint orbit conservation we need the following lemma.

Lemma C.1. *For each $\vec{v}_1, \vec{v}_2 \in \mathfrak{X}_{\text{div}}(W)$ and $s \in \Sigma$, we have*

$$i_{\text{adv}_{\vec{v}_2}} s i_{\text{adv}_{\vec{v}_1}} s \sigma = \langle J_{\text{adv}}(s) | [\vec{v}_1, \vec{v}_2] \rangle. \quad (131)$$

PROOF. The formula (131) we want to show is an equality between functions defined over $s \in \Sigma$. That is we want to show the functional relation $i_{\text{adv}(\vec{v}_2)} i_{\text{adv}(\vec{v}_1)} \sigma = \langle J_{\text{adv}} | [\vec{v}_1, \vec{v}_2] \rangle$, where $\text{adv}: \mathfrak{X}_{\text{div}}(W) \rightarrow \mathfrak{X}(\Sigma)$ is a Lie algebra homomorphism that results in a vector field. To show this functional relation, it suffices to show that their differentials are equal: We claim that $di_{\text{adv}(\vec{v}_2)} i_{\text{adv}(\vec{v}_1)} \sigma = d \langle J_{\text{adv}} | [\vec{v}_1, \vec{v}_2] \rangle$. Using (106) and that adv is a Lie algebra homomorphism, we have

$$d \langle J_{\text{adv}} | [\vec{v}_1, \vec{v}_2] \rangle = -i_{\text{adv}([\vec{v}_1, \vec{v}_2])} \sigma \quad (132)$$

$$= -i_{[\text{adv}(\vec{v}_1), \text{adv}(\vec{v}_2)]} \sigma \quad (133)$$

$$= -\mathcal{L}_{\text{adv}(\vec{v}_1)} i_{\text{adv}(\vec{v}_2)} \sigma + i_{\text{adv}(\vec{v}_2)} \mathcal{L}_{\text{adv}(\vec{v}_1)} \sigma \quad (134)$$

$$= -di_{\text{adv}(\vec{v}_1)} i_{\text{adv}(\vec{v}_2)} \sigma, \quad (135)$$

where the last equality uses the Cartan formula, $d\sigma = 0$, and that $\text{adv}(\vec{v}_1), \text{adv}(\vec{v}_2)$ are symplectomorphic $\mathcal{L}_{\text{adv}(\vec{v}_1)} \sigma = \mathcal{L}_{\text{adv}(\vec{v}_2)} \sigma = 0$. \square

Now we are ready to show that (40) preserves the coadjoint orbit. The update (35a) $s^{(n+1)} = \text{Adv}_{\varphi(\Delta t)} s^{(n)}$ is the solution to the following ODE for a fixed \vec{v}

$$\frac{\partial}{\partial t} s_t = \text{adv}_{\vec{v}} s_t, \quad s_0 = s^{(n)}, \quad s^{(n+1)} = s_1. \quad (136)$$

We show that during this update, $[\eta_t] := J_{\text{adv}}(s_t)$ evolves tangential to the coadjoint orbit. Taking the time derivative, we get

$$\frac{\partial}{\partial t} [\eta_t] = \frac{\partial}{\partial t} J_{\text{adv}}(s_t) = dJ_{\text{adv}}|_{s_t} \left[\frac{\partial s_t}{\partial t} \right] = dJ_{\text{adv}}|_{s_t} [\text{adv}_{\vec{v}} s_t]. \quad (137)$$

Take an arbitrary test vector field $\vec{\xi} \in \mathfrak{X}_{\text{div}}(W)$ to pair with this formula

$$\left\langle \frac{\partial}{\partial t} [\eta_t] | \vec{\xi} \right\rangle = \left\langle dJ_{\text{adv}}|_{s_t} [\text{adv}_{\vec{v}} s_t] | \vec{\xi} \right\rangle \quad (138)$$

$$= i_{\text{adv}_{\vec{v}} s_t} d \langle J_{\text{adv}} | \vec{\xi} \rangle \quad (139)$$

$$\stackrel{(106)}{=} -i_{\text{adv}_{\vec{v}} s_t} i_{\text{adv}_{\vec{\xi}}} \sigma \quad (140)$$

$$\stackrel{(131)}{=} \langle J_{\text{adv}}(s_t) | [\vec{v}, \vec{\xi}] \rangle = \langle [\eta_t] | [\vec{v}, \vec{\xi}] \rangle \quad (141)$$

$$= \langle -\mathcal{L}_{\vec{v}} [\eta_t] | \vec{\xi} \rangle. \quad (142)$$

For this to hold for all $\vec{\xi} \in \mathfrak{X}_{\text{div}}(W)$ we conclude that $\frac{\partial}{\partial t} [\eta_t] + \mathcal{L}_{\vec{v}} [\eta_t] = [0]$. Therefore $[\eta_t] = J_{\text{adv}}(s_t)$ evolves by coadjoint action and therefore stays on the same coadjoint orbit. \square

C.12.2 Energy Conservation. Our approach is similar to Engø and Faltsen [2001]. Continuing the setup of Section C.12.1, we let s_t be evolved by (136) that connects $s^{(n)}$ and $s^{(n+1)}$. Let $[\eta_t] = J_{\text{adv}}(s_t)$.

By Section C.12.1 we know $\frac{\partial}{\partial t} [\eta_t] + \mathcal{L}_{\vec{v}} [\eta_t] = [0]$, where \vec{v} is fixed. Then (35) simplifies to

$$\begin{cases} [\eta_1] = \exp(-\Delta t \mathcal{L}_{\vec{v}}) [\eta_0] \\ \vec{v} = \frac{1}{2} \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ ([\eta_1] + [\eta_0]). \end{cases} \quad (143a)$$

$$\vec{v} = \frac{1}{2} \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ ([\eta_1] + [\eta_0]). \quad (143b)$$

The energies at frame n and $n+1$ are respectively

$$\mathcal{E}^{(n)} = \frac{1}{2} \|\mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ [\eta_0]\|_{\mathfrak{B}}^2 = \frac{1}{2} \|\mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ [\eta_0]\|^2 \quad (144)$$

$$\mathcal{E}^{(n+1)} = \frac{1}{2} \|\mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ [\eta_1]\|_{\mathfrak{B}}^2 = \frac{1}{2} \|\mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ [\eta_1]\|^2. \quad (145)$$

Our goal is to show that their difference is zero. Their difference, by the difference-of-squares formula, is given by

$$\begin{aligned} \mathcal{E}^{(n+1)} - \mathcal{E}^{(n)} &= \frac{1}{2} \langle \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ ([\eta_1] + [\eta_0]), \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ ([\eta_1] - [\eta_0]) \rangle \\ &\stackrel{(143b)}{=} \langle \vec{v}, \mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}} \mathcal{I}^+ ([\eta_1] - [\eta_0]) \rangle. \end{aligned} \quad (146)$$

Now, by Theorem 3.1 and $\vec{v} \in \mathcal{I}(\mathfrak{B}_{\text{div}})$, we can replace $\mathcal{I} \mathbf{P}_{\mathfrak{B}_{\text{div}}}$ by $\mathbf{P}_{\mathfrak{X}_{\text{div}}} \mathcal{I}$, giving us

$$\mathcal{E}^{(n+1)} - \mathcal{E}^{(n)} = \langle \vec{v}, \mathbf{P}_{\mathfrak{X}_{\text{div}}} \mathcal{I}^+ ([\eta_1] - [\eta_0]) \rangle \quad (147)$$

$$= \langle \vec{v}, \sharp([\eta_1] - [\eta_0]) \rangle = \langle \vec{v} | [\eta_1] - [\eta_0] \rangle \quad (148)$$

The second equality uses the fact that $\mathbf{P}_{\mathfrak{X}_{\text{div}}}$ and $\mathcal{I} \mathcal{I}^+$ are L^2 orthogonal projection operators respectively to $\mathfrak{X}_{\text{div}}(W) \subset \mathfrak{X}(W)$ and $\mathcal{I}(\mathfrak{B}) \subset \mathfrak{X}(W)$, and \vec{v} is an element of both subspaces.

We are left to show $\langle \vec{v} | [\eta_1] - [\eta_0] \rangle = 0$. By (143a) we have

$$[\eta_1] - [\eta_0] = (\exp(-\Delta t \mathcal{L}_{\vec{v}}) - 1) [\eta_0] \quad (149)$$

$$= \left((1 - \Delta t \mathcal{L}_{\vec{v}} + \frac{\Delta t^2}{2!} \mathcal{L}_{\vec{v}}^2 - \frac{\Delta t^3}{3!} \mathcal{L}_{\vec{v}}^3 + \dots) - 1 \right) [\eta_0] \quad (150)$$

$$= \left(-\Delta t \mathcal{L}_{\vec{v}} + \frac{\Delta t^2}{2!} \mathcal{L}_{\vec{v}}^2 - \frac{\Delta t^3}{3!} \mathcal{L}_{\vec{v}}^3 + \dots \right) [\eta_0] \quad (151)$$

$$= \underbrace{\mathcal{L}_{\vec{v}} \left(-\Delta t + \frac{\Delta t^2}{2!} \mathcal{L}_{\vec{v}} - \frac{\Delta t^3}{3!} \mathcal{L}_{\vec{v}}^2 + \dots \right)}_{=[\vec{\eta}]} [\eta_0]. \quad (152)$$

Therefore,

$$\langle \vec{v} | [\eta_1] - [\eta_0] \rangle = \langle \vec{v} | \mathcal{L}_{\vec{v}} [\vec{\eta}] \rangle = -\underbrace{\langle \vec{v}, \vec{v} \rangle}_{=0} \langle [\vec{\eta}] \rangle = 0. \quad (153)$$

\square

C.13 Proof of Theorem 4.6

The advection in (14a) is a coadjoint action, and therefore it preserves the coadjoint orbit. The advection in (19a) preserves the coadjoint orbit by the same argument as Section C.12.1. \square

C.14 Proof of Theorem 4.7

By construction $(\mathbf{f}^{(n+1)} - \mathbf{f}^{(n)}) \perp (\mathbf{f}^{(n+1)} + \mathbf{f}^{(n)})$. Therefore $|\mathbf{f}^{(n+1)}|_{\mathfrak{B}}^2 - |\mathbf{f}^{(n)}|_{\mathfrak{B}}^2 = 0$. \square