Energy-Preserving Integrators for Fluid Animation

Patrick Mullen Caltech · Keenan Crane Caltech · Dmitry Pavlov Caltech · Yiying Tong MSU · Mathieu Desbrun Caltech

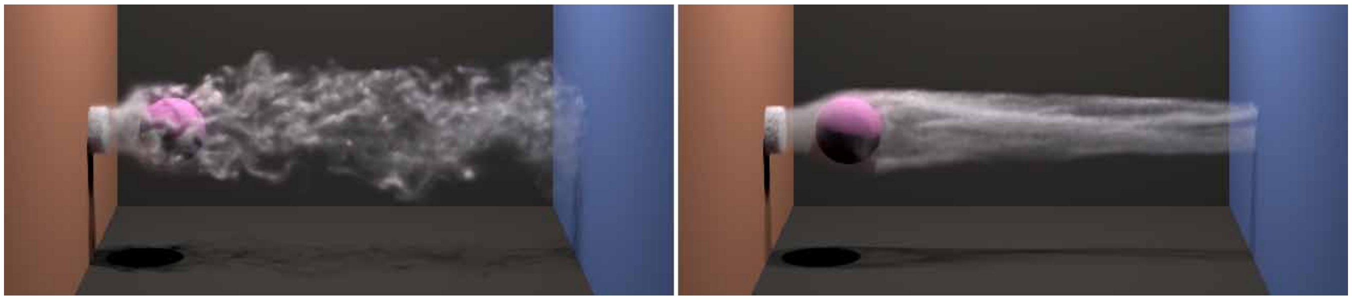


Figure 1: *By developing an integration scheme that exhibits zero numerical dissipation, we can achieve more predictable control over viscosity in fluid animation. Dissipation can then be modeled explicitly to taste, allowing for very low (left) or high (right) viscosities.*

Abstract

Numerical viscosity has long been a problem in fluid animation. Existing methods suffer from intrinsic artificial dissipation and often apply complicated computational mechanisms to combat such effects. Consequently, dissipative behavior cannot be controlled or modeled explicitly in a manner independent of time step size, complicating the use of coarse previews and adaptive-time stepping methods. This paper proposes simple, unconditionally stable, fully Eulerian integration schemes with no numerical viscosity that are capable of maintaining the liveliness of fluid motion without recourse to corrective devices. Pressure and fluxes are solved efficiently and simultaneously in a time-reversible manner on simplicial grids, and the energy is preserved exactly over long time scales in the case of inviscid fluids. These integrators can be viewed as an extension of the classical energy-preserving Harlow-Welch / Crank-Nicolson scheme to simplicial grids.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: Eulerian fluid animation, time integration, energy preservation.

1 Introduction

Physically-based animation of fluids is often modeled using the *incompressible Navier Stokes equations*. Numerically integrating these equations presents numerous practical challenges, however, and has been a focus of Computational Fluid Dynamics (CFD) for the past thirty years. When visual impact matters most, more elaborate CFD methods are generally considered unnecessary: simple Eulerian discretizations with explicit semi-Lagrangian or upwind advection have been the methods of choice in computer animation for the last few years.

A significant numerical difficulty in all CFD techniques is avoiding *numerical viscosity*, which gives the illusion that a simulated fluid is *more* viscous than intended. It is widely recognized that numerical viscosity has substantial visual consequences, hence several mechanisms (such as vorticity confinement and Lagrangian particles) have been devised to help cope with the loss of fine scale details. Common to all such methods, however, is the fact that the amount of energy lost is not purely a function of simulation duration, but also depends on the grid size and number of time steps performed over the course of the simulation. As a result, changing the time step size and/or spatial sampling of a simulation can significantly change its qualitative appearance. Such a dependence can make it difficult to compute coarse previews, for example.

In this paper, we propose a family of fully-Eulerian schemes that provide full control over the amount of dissipation, independent of temporal and spatial resolution. Control over viscosity stems from the use of a non-dissipative integration scheme for the Euler equations on arbitrary simplicial grids. We can then add implicit diffusion to model viscosity directly. We demonstrate the efficiency and robustness of these schemes via numerical comparison with current techniques.

1.1 Previous Work

Early work on fluid animation for computer graphics widely favored the use of Eulerian staggered grid discretizations [Foster and Metaxas 1997; Fedkiw et al. 2001], where the fluid velocity components are stored per face instead of being collocated at nodes. Semi-Lagrangian advection techniques [Stam 1999] have been prevalent due to their unconditional stability and ease of implementation. Their resulting excessive energy dissipation was later mitigated by the adoption of vorticity confinement to partially reinject lost vorticity into the flow [Steinhoff and Underhill 1994; Fedkiw et al. 2001], or through higher-order advection schemes using repeated semi-Lagrangian steps [Kim et al. 2007; Selle et al. 2008]. However, the stability of these methods is guaranteed only if spurious extrema are eliminated by a limiter. More recently, a case was made for explicit, third-order upwind-based advection [Molemaker et al. 2008] as a low dissipation technique at reasonable computational cost. Subscale modeling was also proposed [Schechter and Bridson 2008] to get energy cascading in line with the empirical behavior of statistically stationary isotropic turbulence despite numerical viscosity, although at a scale much larger than intended.

Another remedy to combat dissipation is to add Lagrangian machinery to the Eulerian solver. Selle *et al.* [2005] proposed to add vortex particles to track vorticity and inject a tuneable confinement force into the flow. Bridson *et al.* [2005] even advocated the substitution of the semi-Lagrangian advection scheme in stable fluids with the Lagrangian *fluid-implicit-particle* (FLIP) scheme [Brackbill and Ruppel 1986] that exhibits significantly less numerical dissipation than semi-Lagrangian advection after pressure projection. While fast, these mixed schemes require careful management of particle distribution in order to achieve good quality and performance.

Another line of research in computer animation of fluids focused on offering more versatile spatial discretizations to capture complex geometries with low node count. Spatially adaptive discretizations such as octrees were proposed to improve resolution of highly turbulent flows [Shi and Yu 2002; Losasso et al. 2004], albeit with significant computational overhead and grid-aligned aliasing artifacts. Integration schemes for simplicial and hybrid meshes through semi-Lagrangian backtracking were introduced, for which the computational overhead brought by non-regular mesh data structures were largely compensated for by the increased visual complexity per element [Feldman et al. 2005; Chentanez et al. 2007]. Even if circulation preservation can be enforced [Elcott et al. 2007], these schemes on arbitrary grids still suffer from noticeable energy dissipation, leading to an uncontrolled energy decay in the flow that depends on both the mesh size and the time step size.

Energy preservation for inviscid fluids on unstructured grids had not received much attention in CFD in the past due to the prevalence of methods based on regular grids. This has changed recently with the introduction of discrete energy-conserving schemes on 2D and 3D unstructured grids [Perot 2000; Zhang et al. 2002; Mahesh et al. 2004]. The role of discrete differential operators for the curl and divergence (as used in [Elcott et al. 2007]) in ensuring kinetic energy conservation was acknowledged, and numerical benefits from non-dissipative advection were demonstrated. In particular, energy preservation guarantees that the velocity will always remain bounded, bringing unconditional stability to the schemes. Derivations of these new schemes were obtained through discretization of the vorticity form of the continuous Euler equations, but the final update rules require a fairly large stencil to compute advection. Moreover, it is not known whether these schemes can be derived from first principles (as was successfully done for Lagrangian integrators; see [Stern and Desbrun 2006] for a review).

1.2 Overview and Contributions

In this paper we present purely Eulerian integrators for fluid flow on simplicial grids. These integrators involve very sparse equations, are unconditionally stable, and can be made time-reversible, while exhibiting excellent long-term energy behavior in the sense that total kinetic energy of inviscid fluids is conserved over arbitrarily long durations without any parameter tuning. In the important case of viscous fluids, our schemes capture the correct energy decay, mostly independently of the time step and grid size. Efficiency is achieved via an application of the *Schur complement* to quickly solve the saddle-point problem arising from the implicit integration step. Consequently, our implicit schemes are nearly as efficient as other integrators for typical animation sequences, with the benefit of being robust to change of time and space resolution. We finally show that the resulting schemes share the same numerical benefits as the Harlow-Welch scheme with a Crank-Nicolson time discretization on regular grids (see, e.g., [Simo and Armero 1994]), and discuss that some of these integrators are, in fact, variational.

2 Discrete Setup of Fluid Motion

Before describing our Eulerian schemes for fluid animation, we establish the equations of motion of both viscid and inviscid fluids, and discuss the physical properties that our discrete time integrators should preserve.

2.1 Equations of Motion

Consider an inviscid, incompressible, and homogeneous fluid on domain M with velocity $\mathbf{u}$ and pressure p. Assuming constant unit density ($\rho = 1$), the motion of such a fluid is governed by the *Euler equations* which consist of a *momentum equation*

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p \qquad (1)$$

along with an *incompressibility constraint*

$$\nabla \cdot \mathbf{u} = 0. \qquad (2)$$

Along with boundary conditions, these equations define the fluid behavior and have been thoroughly studied both from a mechanics point of view [Chorin and Marsden 1979] and from a computational perspective [Gresho and Sani 2000].

Vorticity Formulation A particularly convenient expression in which the Euler equations can be rewritten is a function not only of the velocity $\mathbf{u}$, but also of its vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\omega} \times \mathbf{u} = -\nabla P \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \qquad (3)$$

where now P is the Bernoulli pressure, *i.e.* the former pressure p plus the kinetic energy density: $P = p + \mathbf{u}^2/2$.

Handling Viscosity For a viscous fluid, the incompressible Navier-Stokes equations are used instead. A diffusion term $\nu\Delta\mathbf{u}$ is added to the previous momentum equation as an extra body force, where ν controls the amount of viscosity in the flow. This term diffuses the momentum of the fluid, damping down turbulences.

2.2 Relevant Continuous Properties

Euler and Navier-Stokes flows have a number of properties that one may want to preserve in the discrete realm to ensure a close visual match between the discrete results and typical flows in nature.

Of particular visual significance is the fact that the Euler equations preserve kinetic energy in time. *Energy preservation* is, however, only rarely observed in practice by the fluid simulators used in computer animation, visually producing overly viscous animation. Additionally, this energy tends to seep towards small scales, a general mechanical property called *energy cascading*: the kinetic energy is thus conservatively transferred to smaller and smaller length scales statistically, meaning that a large vortex will, over time, erode into several smaller ones[1]. Faithfully reproducing this energy cascading in the discrete case is impossible due to the limited resolution of meshes, and can only be approximated through subscale modeling [Schechter and Bridson 2008]—although it is quite unclear that the type and scale of animations used in computer graphics can visually benefit from the restricted physical assumptions used in the modeling of forward cascading. Kelvin's circulation theorem (stating that *vorticity is advected along the flow*) was also pointed out in [Elcott et al. 2007] as a crucial property to preserve to visually capture the traditional turbulent behavior of nearly inviscid flows. A last property of the Euler equations that has been shown important to preserve independently of the numerical approximation is *time reversibility* [Duponcheel et al. 2008]: applying the integration with negated fluxes should exactly run the simulation backwards in time. Notice that this symmetry in time is what was sought after in the BFECC [Kim et al. 2007] and modified MacCormack [Selle et al. 2008] schemes: both schemes try to get the backward *and* forward semi-Lagrangian to numerically agree, leading to much decreased numerical viscosity. Finally, some applications are interested in the preservation of the total *enstrophy* of the fluid. However, the benefit of this to computer animation is currently less evident and our methods do not address the preservation of this quantity.

[1]Note that the energy may sometimes go back up to larger scales temporarily, as when two same-sign vortices merge.

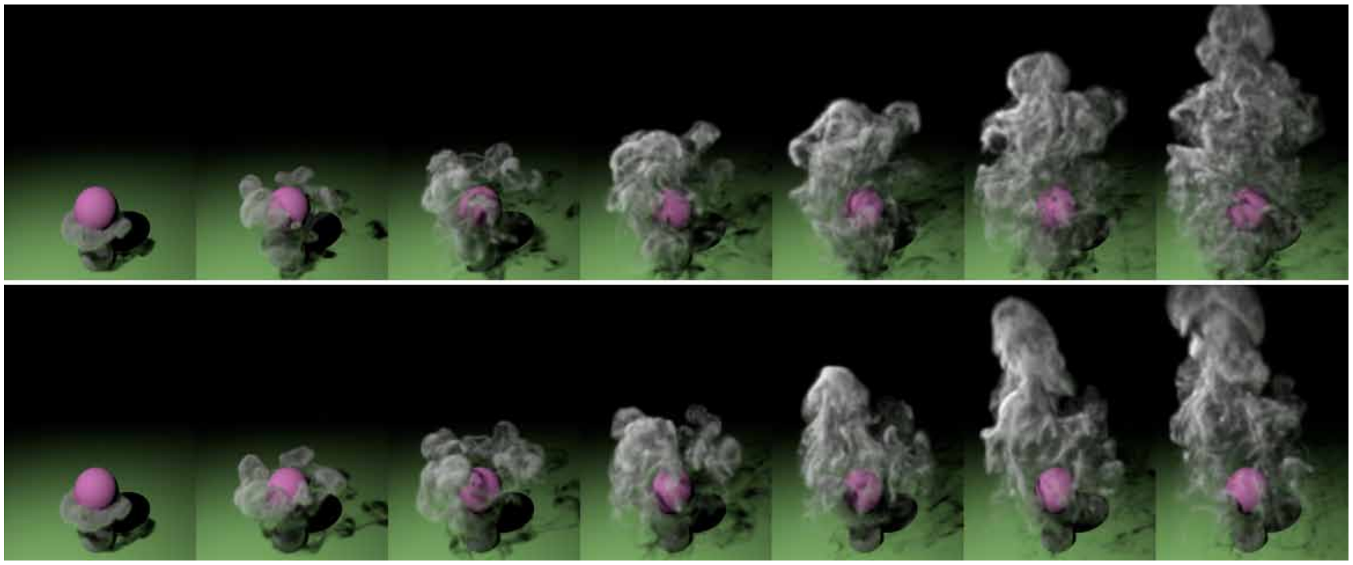


Figure 2: ***Flow Past Sphere****: Although completely inviscid flows may look unnatural (top), the absence of numerical viscosity gives animators more predictable control over fluid appearance (bottom); here, smoke rises in a closed box containing a round obstacle.*

We will thus focus on developing numerical schemes that, by pushing vorticity forward with the flow, will exactly achieve time reversibility and remove numerical energy dissipation for inviscid flows. As we generally want to animate viscous flows, proper treatment of the non-viscous part of the Navier-Stokes equations will ensure proper energy decay even for large time steps. Note that potential artifacts due to energy accumulating at fine scales will be removed with an appropriate viscosity coefficient ν.

2.3 Discretization of Physical Quantities

To derive a computational procedure to integrate Euler and/or Navier-Stokes equations, we must discretize both space and time, and define a discretization of our physical quantities as well.

Spatial Discretization We use a tetrahedral mesh $\mathcal{T}$ (assumed to be Delaunay and well-centered for simplicity) to discretize the spatial domain M; *i.e.*, the domain is decomposed into a series of tets $\{T_i\}, i = 1..|\mathcal{T}|$. We assume that these tets are all oriented so as to have positive volume, and their faces and edges are given arbitrary orientations. From this *primal* tet mesh, we construct a *Voronoi dual* $\mathcal{V}$ of the mesh. That is, we define a dual vertex c_i as the circumcenter of tet T_i; a dual edge h_{ij} is the edge between c_i and c_j, that is dual to face f_{ij}; a dual face s_e is a face of the Voronoi region associated to (and thus orthogonal to) edge e, bounded by a loop of dual edges h_{kl}; and finally, a dual volume is the Voronoi cell of a vertex of $\mathcal{T}$, formed by dual faces. Note that the dual cells are also given orientations. We will also use the intersections of the primal and dual elements, including c_{ij} (the circumcenter of face f_{ij}) and c_e (the midpoint of edge e). We will denote by $w_{e,j}$ the quad-shaped intersection of cell j and the dual face s_e (see Figure 3). Finally, $|.|$ will denote the Lebesgue measure (length, area, or volume) of the elements (edges, faces, or tets).

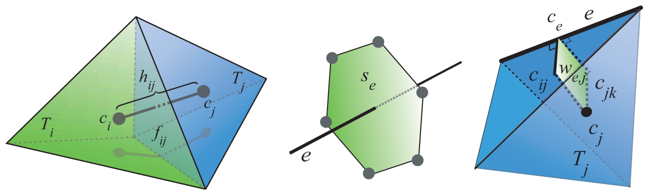


Figure 3: ***Spatial Discretization****: we will refer to primal and dual mesh elements using the notation depicted in this figure.*

Field Discretization As in [Feldman et al. 2003; Elcott et al. 2007], we adopt a flux based discretization of the velocity field. This particular discrete setup is well-known, on regular grids, for offering less aliasing than the node-based discretization, as well as for preventing spurious modes in Poisson problems. We will denote by F_{ij} the flux of the fluid through the face f_{ij} common to tet T_i and tet T_j (note that $F_{ji} = -F_{ij}$), while the discrete Bernouilli pressure on the dual node of tet T_i will be denoted P_i. The velocity field will thus be represented by all the fluxes F_{ij}, stored in a vector F of size equal to the number of faces. Similarly, P will represent the pressure field as a vector of all tet-based pressure values.

Discrete Operators We can discretize differential operators based on Stokes' theorem through two types of matrices [Elcott et al. 2007]: the adjacency matrices d_k representing the connectivity between k-simplices and $k+1$-simplices (with the sign of the entries determined by mutual orientation), and the diagonal Hodge star matrices $\star_k$ with diagonal entries containing the ratios between the corresponding dual $(n-k)$-element size and the primal k-element size. Equipped with these simple matrices, the **divergence** of $\mathbf{u}$ is directly $\nabla \cdot \mathbf{u} \equiv d_2 F$, *i.e.*, the divergence per tet T_i is the sum of the outward fluxes of T_i . Similarly, the **curl** of $\mathbf{u}$ is represented via its surface integrals over dual faces by $\nabla \times \mathbf{u} \equiv d_1^t \, \star_2 F$, *i.e.*, the curl per dual face ω_e is the sum of line integrals of $\mathbf{u}$ along the dual edges between adjacent tets around an edge e.

3 Discrete Energy-preserving Time integrator

In this section, we first present a time integration that preserves total energy while respecting time-symmetry for inviscid fluids, before describing our treatment of viscosity.

3.1 Discrete Euler equations

From the vorticity form of Euler equations given in Eq. (3), integrating the continuous terms over each face f_{ij} leads to:

$$\dot{F}_{ij} + \overbrace{\int_{f_{ij}} (\boldsymbol{\omega} \times \mathbf{u}) \cdot \mathbf{n} \, dA}^{\mathrm{Adv}(F)_{ij}} = -(\star_2)^{-1}_{ij} (P_j - P_i)$$

$$\sum_{j \in \mathcal{N}(i)} F_{ij} = 0$$

where $\mathcal{N}(i)$ are the cells sharing a face with i. Note that the diagonal Hodge star $(\star_2)_{ij} = |h_{ij}|/|f_{ij}|$ was used to turn the line integral of ∇P into a surface integral. Using $\text{Adv}(F)_{ij}$ as a shorthand for the area integral over face f_{ij} we just derived, we can write the discrete version of Eq. (3) as a function of F through:

$$\begin{cases} \dot{F} + \text{Adv}(F) = -\star_2^{-1}\, d_2^t P \\ d_2 F = 0 \end{cases} \tag{4}$$

In order to update the Eulerian fluxes F and construct an Euler fluid integrator, we must therefore design a numerical approximation of this "advection term" $\text{Adv}(F)$.

3.2 Advection Term

To approximate this advection term, we proceed through local averages of the flux of $\boldsymbol{\omega}\times\mathbf{u}$ on face f_{ij} in a Finite Volume manner. Our procedure is conceptually simple: first we reconstruct a piecewise-constant velocity vector $\mathbf{U}_i$ per tet T_i based on the known fluxes of each of its four faces: since $\mathbf{U}$ is divergence-free there is a unique vector such that the four fluxes of the tet represent the area integral of this vector along the face normals. (Note that this vector can similarly be found through discrete Whitney basis functions to reconstruct the vector field [Elcott et al. 2007].) For each reconstructed vector $\mathbf{U}_i$, we can evaluate the face integral that defines $\text{Adv}(F)_{ij}$: treating the vorticity vector around edge e as a constant $\boldsymbol{\omega}_e = \omega_e/|s_e|.(\mathbf{e}/|\mathbf{e}|)$ in the region, we can integrate the flux of $\boldsymbol{\omega}_e\times\mathbf{U}_i$ over the region of f_{ij} inside the convex hull of the edge e and the circumcenter c_{ij} of the face. Summing the contributions from the two other edges of f_{ij} will provide us directly with the area integral. Finally, we antisymmetrize this evaluation to enforce that our discrete approximation of $\boldsymbol{\omega}\times\mathbf{u}$ also satisfies $(\boldsymbol{\omega}\times\mathbf{u})\times\mathbf{u}=0$; we will see that this will enforce energy preservation when we integrate our equations in time. This procedure provides a low-order approximation of the advection term $\text{Adv}(F)_{ij}$ on each f_{ij}. The reader can find the exact terms involved in this summation, as well as the explanations leading to energy preservation, in Appendix A.

Assembly Per Edge While this integration can be performed literally as explained above, we found it more efficient to reorganize the terms involved so as to ensure an easy implementation and an efficient evaluation. We instead proceed as follows:

For each edge e,

- evaluate the vorticity ω_e on edge e
- for every pair of consecutive faces f_{ij} and f_{jk} around e,
  - add contribution of F_{jk} and $\boldsymbol{\omega}_e$ on $\text{Adv}(F)_{ij}$:
  $$\text{Adv}(F)_{ij} \mathrel{-}= \frac{\omega_e}{|s_e|} F_{jk} \frac{2|w_{e,j}|}{3|T_j|}$$
  - add contribution of F_{ij} and $\boldsymbol{\omega}_e$ on $\text{Adv}(F)_{jk}$:
  $$\text{Adv}(F)_{jk} \mathrel{+}= \frac{\omega_e}{|s_e|} F_{ij} \frac{2|w_{e,j}|}{3|T_j|}$$

Note that Adv represents a flux change through oriented faces, thus $\text{Adv}(F)_{ji} = -\text{Adv}(F)_{ij}$; also, consecutive faces must be ordered consistently with the edge orientation and vorticity computation. Now that we have defined the advection operator, we can proceed to build Eqs. (4) required to evolve our fluid forward in time.

3.3 Time-Reversible Integration

Until now, we only focused on spatial integration of the Euler equations, and did not discuss which fluxes to use in the advection operator. One could use an explicit time integration by using the fluxes at time t^n in $\text{Adv}(F)_{ij}$ to compute the fluxes at time t^{n+1}. Conversely, an implicit integration could be used instead, assuring much better numerical stability at the price of a greater energy dissipation. As argued in Section 2.2, we would rather derive a time integration devoid of numerical viscosity to offer better control over the fluid viscosity once diffusion is included, as well as enforcing time-reversibility.

Therefore, we use a *midpoint* integration for the updates of flux in time, resulting in the following integrator:

$$\begin{cases} F^{n+1} = F^n - h\,\text{Adv}\left(\dfrac{F^n + F^{n+1}}{2}\right) - h\,\star_2^{-1}\, d_2^t P^{n+\frac{1}{2}}, \\ d_2 F^{n+1} = 0 \end{cases} \tag{5}$$

where $h = t^{n+1} - t^n$ is the time step size, F refers to the vector of all fluxes F_{ij}, and the superscript n (resp., $n+\frac{1}{2}$) is used to indicate evaluation at time t^n (resp., between time t^n and t^{n+1}).

Nonlinear Solve Evolving the fluid in time thus amounts to finding F^{n+1} and $P^{n+\frac{1}{2}}$ such that the following residual is zero:

$$R(F,P) := \begin{pmatrix} \frac{1}{h}(F - F^n) + \text{Adv}\left(\dfrac{F^n + F}{2}\right) + \star_2^{-1} d_2^t P \\ d_2 F \end{pmatrix}$$

This update equation allows us to derive the next set of fluxes F^{n+1} as a function of the current fluxes F^n through a non-linear solve. Notice however that this equation only involves *linear and quadratic terms* in F^{n+1}, so a simple nonlinear solver using Newton's method is sufficient. We thus repeatedly solve the following Newton steps:

$$\overbrace{\begin{pmatrix} \frac{1}{h}\text{Id} + \frac{\partial \text{Adv}}{\partial F} & \star_2^{-1} d_2^t \\ d_2 & \mathbf{0} \end{pmatrix}}^{J} \begin{pmatrix} \delta F \\ \delta P \end{pmatrix} = -R(F^{n+1}, P^{n+\frac{1}{2}})$$

until the norm of the residual R is below an accuracy threshold.

Improving Solver Performance Directly solving the system $J(\delta F^t, \delta P^t)^t = -R$ during each Newton step can be burdensome since J is asymmetric and needs to be assembled for each step. We take two measures to improve performance. First, we approximate the Jacobian matrix J by omitting any entries of its upper left block not on the diagonal. In practice we have found that taking this approach both reduces the amount of work required to setup each Newton step and greatly increases the sparsity of the system without significantly increasing the required number of steps for convergence. We are then left with a typical *saddle-point* problem:

$$\begin{bmatrix} A & \star_2^{-1} d_2^t \\ d_2 & 0 \end{bmatrix} \begin{bmatrix} \delta F \\ \delta P \end{bmatrix} = -\begin{bmatrix} R_F \\ R_P \end{bmatrix}$$

where A is the sum of Id/h and of the *diagonal part* of the Jacobian matrix $\partial\text{Adv}/\partial F$ containing the derivatives of the advection term with respect to fluxes. Notice in particular that, besides being diagonal, the matrix A has only positive terms for sufficiently small timesteps (or sufficiently small velocities); hence in practice, our matrix A is positive definite. Saddle point problems of this form can be efficiently solved using the *Schur complement method* [Benzi et al. 2005]. We can indeed solve the linear system:

$$d_2 A^{-1} \star_2^{-1} d_2^t\, \delta P = -d_2 A^{-1} R_F + R_P \tag{6}$$

for the change of pressure δP, then derive the new flux change through:

$$\delta F = A^{-1}\left(-R_F - \star_2^{-1} d_2^t \delta P\right).$$

Notice that the pressure update is solved via a Poisson equation (as $d_2 A^{-1} \star_2^{-1} d_2^t$ is a symmetric Laplacian matrix for the metric induced by $\star_2 A$), for which a preconditioned conjugate gradient is most appropriate; thus, applying the Schur complement amounts to solve a system similar to pressure projection to find δP, followed by a trivial backsubstitution to get δF (as A is diagonal). Overall the work done in each Newton step is thus identical to the work done for a single step of the stable fluids algorithm [Stam 1999].

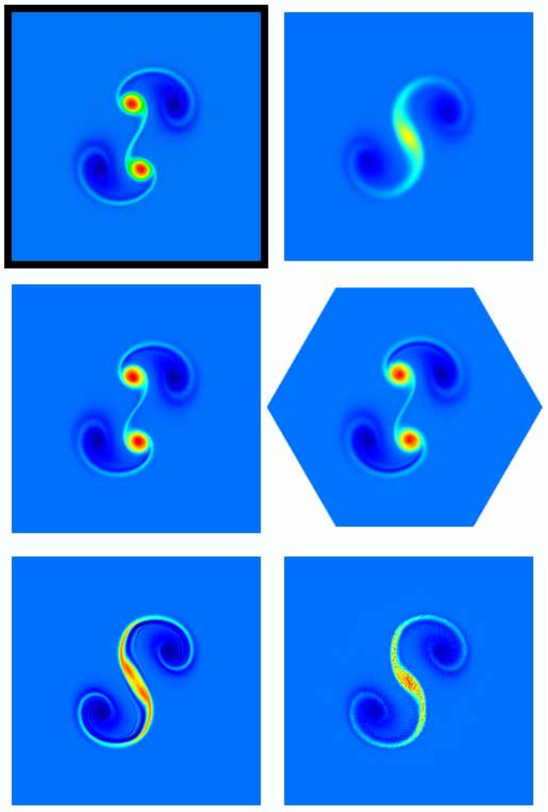


Figure 4: ***Taylor vortices on a periodic domain****: for the particular initial separation distance used here, two vortices of the same sign should split apart as in the reference solution (top left). Many schemes fail to reproduce these results. From left to right, top to bottom; Reference solution; Stable fluids [Stam 1999]; energy-preserving scheme (Harlow-Welch [Harlow and Welch 1965] w/ midpoint time discretization); our simplicial energy-preserving scheme; a MacCormack scheme [Selle et al. 2008]; FLIP [Zhu and Bridson 2005]. All results were computed on grids of around* 2^{16} *cells or triangles.*

Further Improvement If the time step needed for animation is small enough, we can further simplify the non-linear solver by only using the dominant term Id/h in matrix A. Discarding $\partial\mathrm{Adv}/\partial F$ renders the matrix A constant, removing the need to rebuild the matrix and allowing for the precomputation of preconditioners. For reasonably sized systems (in our experience, less than 150K tets when using Matlab) we can even LU factorize the Laplacian matrix $hd_2\star_2^{-1}d_2^t$ once at the beginning, and efficiently solve each Newton step through the Schur complement method in Eq. (6) by constant-time backsubstitution. This simplification is the method of choice in practice when possible. However, for large time steps with respect to the motion, this approach can fail to converge; we then either revert to the previous method and add to A the diagonal of the Jacobian of the advection term, or simply reduce our time step.

3.4 Other Time Integration Schemes

While this implicit, time-reversible energy preserving scheme is the one we used in all our examples, we can easily derive other fully Eulerian integrators with similar numerical properties.

Variational Integration Instead of a midpoint rule, we can maintain time reversibility by choosing a trapezoidal time update for which the momentum equation is expressed as:

$$\frac{1}{h}(F^{n+1}-F^n)+\frac{1}{2}\mathrm{Adv}(F^n)+\frac{1}{2}\mathrm{Adv}(F^{n+1})=-\star_2 d_2^t P^{n+\frac{1}{2}}.$$

The implementation of this different integrator is nearly identical to the midpoint case, and will thus not be detailed here. Unlike the previous integration scheme, this time-reversible update rule does not conserve energy exactly. Nevertheless, the energy remains nearly constant, basically oscillating around the initial energy. Such a behavior is typical of symplectic integrators, and in fact, we derived this integrator through first principles. This lengthy derivation will not be shown here, but we refer the reader to Appendix B for an overview of the geometric mechanical arguments leading to this variational time update.

Hybrid Time Integration One can also play around with the basic approach and derive other interesting integrators. The vorticity ω used in our derivation can in fact be evaluated at any point in time between t^n and t^{n+1}, and the update will *remain* exactly energy preserving—while still corresponding to an advection of the vorticity in the velocity field. We can exploit this property to now provide a time update that only requires a *linear* solve in F^{n+1} and $P^{n+\frac{1}{2}}$:

$$\frac{1}{h}(F^{n+1}-F^n)+\mathrm{Adv}(F^{n+\frac{1}{2}},\boldsymbol{\omega}^n)=-\star_2^{-1}d_2^t P^{n+\frac{1}{2}},$$
$$d_2F^{n+1}=0.$$

For a reasonably small timestep, this approach offers a fast alternative to midpoint integration. Our experience shows that for larger timesteps, the conditioning of the linear system goes down, and solving this linear system may not always be faster than the full-blown non-linear alternative. However, this offers a viable fallback solution if the Newton solver of the nonlinear time integrator fails.

3.5 Viscosity

In practice, fluid animation in computer graphics requires a small amount of viscosity to render the motion more realistic. We thus need to approximate the dissipation term $\nu\Delta\mathbf{u}$ to be added to the right hand side of Eq. (1) to transform the Euler equations into the Navier-Stokes equations. As we represent the divergence-free velocity field $\mathbf{u}$ by its fluxes on mesh faces, we can directly apply the discrete Laplacian operator as defined in [Elcott et al. 2007], which, with our notations, is expressed as $-d_1\star_1^{-1}d_1^t\star_2$. Therefore, we modify our integrator to include this term evaluated at the midpoint (we will denote $\bar{F}^{n+\frac{1}{2}}:=(F^n+F^{n+1})/2$), resulting in the following momentum update rule:

$$F^{n+1}=F^n-h\left[\mathrm{Adv}\Big(\bar{F}^{n+\frac{1}{2}}\Big)-\star_2^{-1}d_2^tP^{n+\frac{1}{2}}\right.$$
$$\left.-\nu d_1\star_1^{-1}d_1^t\star_2\bar{F}^{n+\frac{1}{2}}\right],\quad (7)$$

This simple dissipation model turns out to be mostly independent of the time step size and the spatial discretization. As we will see in Section 4, this is particularly convenient for computer animation, as one can easily and predictably adjust the viscosity of the fluid simulated without having to worry that the visual results are, in fact, dependent on the time step. Note that this also allows adaptive time stepping strategies to be used for more efficient computations without inducing motion artifacts.

3.6 Boundary Conditions

Basic boundary conditions can be dealt with quite simply in our Eulerian framework. First, we can control the normal component of

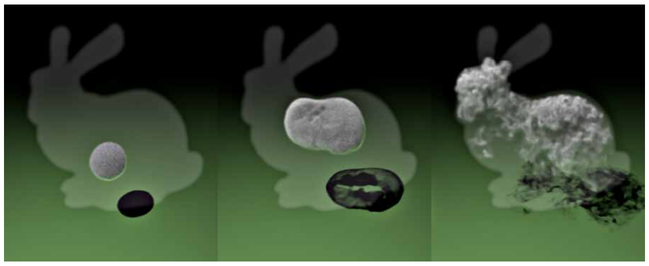


Figure 5: ***Smoking Bunny****: hot smoke rising in a bunny-shaped domain, then cooling down over time.*

the velocity along the boundary by specifying the desired fluxes F_{ij} on boundary faces. No-transfer conditions are thus implemented by simply setting fluxes to zero, while forced fluid influx or outflux is achieved by forcing non-zero F_{ij} values in the solver (note that to ensure divergence-freeness, the total flux through the boundary should still sum to zero). We can also prescribe *pressure* values on the outside of boundary faces to achieve "open" boundary conditions, where the boundary flux will now be determined by the gradient of the pressure across the boundary. Both types of boundary conditions fix the same degrees of freedom for the momentum update rule, although if only flux boundary conditions are used the pressure is arbitrary up to a constant, in which case the pressure of one tet may be fixed to keep the system definite.

Tangential velocity conditions may, however, seem less obvious since we encoded the fluid velocity only by its normal component to each face: while Harlow and Welch [1965] implement free-slip condition by mirroring the tangential velocity component across the wall and no-slip boundary condition by reverting the tangential component across the wall, this type of symmetrization of the velocity is no longer simple for simplicial grids. Fortunately, we can implement the same boundary conditions by simply acting on the vorticity instead. For instance, free-slip condition is achieved by setting the vorticity on boundary edges to 0 since if the half-Voronoi face (loop of dual edges) is completed by its mirror image across the boundary surface, copying the tangential velocity component on this mirror image will cancel out the local vorticity integral. Conversely, no-slip condition is achieved by setting the vorticity on boundary edges to be set to the sum of existing dual circulations $(\star_2)_{ij}F_{ij}$ along the half-Voronoi face: reverting the tangential velocity component on the mirrored half Voronoi face simply double the integral over the full Voronoi face, leaving only the terms including inside fluxes in the vorticity. Partial slip can then be implemented by combining these two and using only some percentage of the vorticity around boundary edges, allowing the modeling of varying "roughness" of boundary materials. Note the choice of the tangential conditions does not affect the energy preservation in the abscence of viscosity.

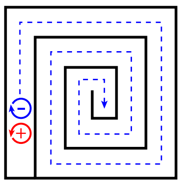
3.7 Discussion

One can easily show that the midpoint integration we introduced is energy preserving: indeed, $\bar{F}^{n+1/2} \star_2 \text{Adv}(\bar{F}^{n+1/2})$ is zero due to the antisymmetrization of the advection term (see Appendix A). Therefore, assuming the solver converges, this scheme preserves energy exactly up to machine accuracy and solver tolerance—resulting in stability as the velocity must thus remains bounded. This midpoint integrator on unstructured grids is, in fact, an extension of Harlow and Welch's scheme [1965] designed for regular grids: the same reasoning (either through the finite volume derivation presented in this paper, or via a variational derivation) leads to their skew-symmetric, conservative form of the advection when applied to regular hexahedral cells. Notice that this advection was used by Foster and Metaxas [1997], albeit with an explicit integration in time and an iterative pressure projection—both of which lead to a change of energy. Instead, our approach shares most of the well-studied numerical benefits of the original Harlow-Welch conservative scheme [Simo and Armero 1994]. Our integrator also involves a sparser system of equations than the most recent energy-preserving fluid integrators on unstructured meshes [Zhang et al. 2002; Mahesh et al. 2004], as only the topological one-ring of a face is involved in its update. While this property also means a lower order accuracy, it provides fast fluid animation at low cost. Further, the precise choice of mesh refinement or time step (which can be non-intuitive to an animator) becomes less critical in the design of an animation, and no extra parameters (like a number of Lagragian particles, or a coefficient of vorticity restitution) is needed. Note finally that simulating fluids in 2D basically follows the same procedure, with the edge/face/tet volumes replaced by vertex/edge/face volumes, and the $2/3$ coefficient in the advection replaced by $1/4$.

4 Results

To validate our approach, we ran a series of tests in 2D and 3D, for both visual and numerical evaluations. Note that we used Delaunay meshes for our domains, as this prevents having to deal with negative dual edge lengths or dual surface areas. One could locally revert to the barycentric dual as well, most likely at the price of accuracy.

Vortices in Periodic Domain A setup commonly used in CFD was used in 2D to evaluate the quality of the integrators presented in this paper. A periodic 2D domain was initialized with two Taylor vortex distributions of same sign for which pseudospectral solutions (with 3/2 dealiasing) are available in the literature. As Figure 4 shows, our results match the expected qualitative behavior (vortices separating) on any (reasonably good) triangle mesh. With the exception of the FLIP method which, while failing to capture the proper behavior, still results in visually pleasing results, none of the other methods were found to provide reasonable results on arbitrary grids.



Spiral Maze We also used a 2D example with many boundaries forming a spiral maze, to demonstrate how much diffusion previous methods engender: while the initial conditions are set up to create a vortex which should propagate through the maze, all our tests of other methods exhibit either significant dissipation, or unexpected behavior. However, our time-reversible scheme does advect the vortex along; no viscosity was added in this example to emphasize energy preservation.

3D Flows Examples we tried in 3D include particles blown in a flow past a sphere (Figure 2), and smoke in a $31K$-tet bunny-shaped domain (Figure 5; buoyancy is incorporated based on the local density of marker particles passively advected through the flow). These examples were chosen to offer easy comparison with previous methods. In these examples, we set our non-linear solver l_∞ threshold to 10^{-3}, and the typical range of Newton steps needed to reach convergence was between 3 and 15. We found that in practice our method was not substantially slower than existing schemes for velocity advection, and it is worth pointing out that rendering or advecting a density field in the flow still largely dominates the cost of animation. Therefore, it is well worth a little extra time on velocity integration, especially since it can have a profound effect on the appearance of the final animation. All of our 3D examples took no more than an average of 40 seconds per timestep, and we typically used 2-6 timesteps per frame. Additionally, we found that a naive adaptive time stepping method based on the CFL condition helps improve the computational complexity without detriment to the dynamics. A change of time step size in previous methods does, in contrast, significantly alter the results as it introduces a change of viscosity in time.

Numerical Comparison of Dissipation As a stress test to estimate numerical energy dissipation of typical fluid integrators used in graphics, we computed the kinetic energy in time for the simulation of an inviscid flow on a periodic domain in Figure 6. While the decay rate varies significantly from method to method, no integrator but ours is devoid of numerically-induced energy dissipation. Notice that even with a very low Newton's accuracy threshold of 10^{-1}, our midpoint implicit Eulerian scheme still preserves energy remarkably well. We also tried to adjust the parameters involved in vorticity confinement and vortex-particle-enriched methods to limit energy loss. However, as Figure 7 indicates, finding good coefficients to eliminate dissipation as much as possible is very animation dependent and creates rather unpredictable energy behavior,

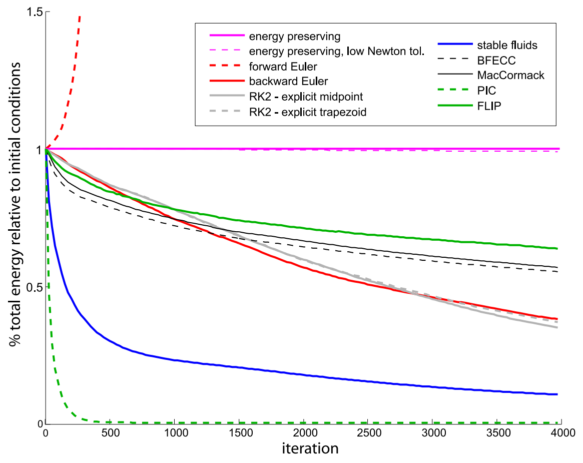


Figure 6: *Even for completely inviscid flows, time integration schemes used in computer animation dissipate a significant percentage of total energy over time. Our fully Eulerian scheme not only preserves energy exactly, but also demonstrates excellent energy behavior even for very approximate solutions (here we use an l_∞ tolerance of 10^{-1} in the Newton solver).*

making the parameter-tweaking process difficult and unintuitive. In contrast, our approach requires no parameter tuning.

Robustness to Time Step We also confirmed the robustness of our implicit integrators with respect to time step and grid sizes in the realistic context of viscous flows. Such improved numerics makes the design of fluid animations easier, as the viscosity parameter will have a predictive value on the results, instead of depending heavily on other simulation parameters (see Figure 1).

5 Conclusion

Fully Eulerian implicit integrators have been largely unexplored in graphics. We have presented evidence that they not only offer a robust computational tool for fluid integration, but possess numerical qualities highly desirable in animation: damping of the fluid flow is no longer a numerical artifact, but a controllable parameter.

Future Work Methods such as FLIP used in [Zhu and Bridson 2005] effectively eliminate numerical diffusion (before pressure projection) but *not* numerical energy dissipation, as demonstrated in Section 4. In contrast, our fully-Eulerian approach is devoid of numerical energy dissipation, but does not eliminate diffusion: vorticity diffusion is unavoidable in a purely Eulerian context, since discretization onto a fixed grid acts as a low-pass filter of the velocity field. Particle-based methods do not have this problem since Lagrangian particles carry information instead of diffusing it around.

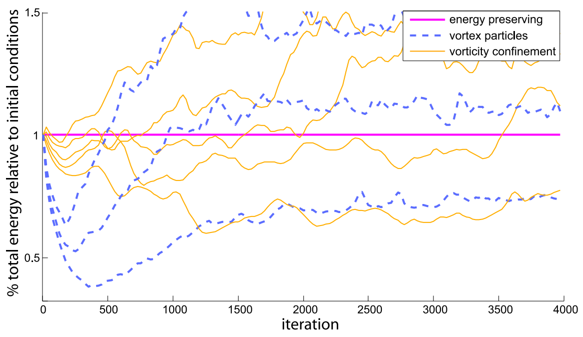


Figure 7: *For carefully chosen parameter values, some existing schemes can roughly preserve initial energy; even then, energy behavior is highly unpredictable. We show the energy curves resulting from several nearby parameter values for each scheme above.*

However, the nature of fluid flows makes the Eulerian approach particularly convenient, as no seeding or redistribution of particles is needed even for vastly turbulent flows. It may be worthwhile to develop Eulerian schemes with lower diffusion, while maintaining their energy preserving characteristics. Another interesting research direction is the design of better subscale modeling models; however, adding noise in the flows may well be more practical for graphics application than subscale modeling. Non-linear schemes can also be fairly well approximated through less computationally intensive explicit integrators [Simo and Armero 1994], although most likely at the cost of more stringent condition on time step size. Finally, the type of conservative integrators we discussed in this paper may be very appropriate for coarse-to-fine design of fluid animation, possibly offering an *Eulerian* extension to Lagrangian tracking methods [Bergou et al. 2007].

Acknowledgments. The authors wish to thank our collaborators Eva Kanso and Jerrold E. Marsden, as well as Kevin Beason for his renderer *Pane*. This research was partially funded by the NSF (CCF-0811373/0811313, DMS-0453145, and CMMI-0757106/0757123), the DOE (DE-FG02-04ER25657), and Pixar.

References

BENZI, M., GOLUB, G. H., AND LIESEN, J. 2005. Numerical solution of saddle point problems. *Acta Numerica 14*, 1–137.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. TRACKS: toward directable thin shells. *ACM Trans. on Graphics 26*, 3, art. 50.

BRACKBILL, J., AND RUPPEL, H. 1986. FLIP: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics 65*, 314–343.

CHENTANEZ, N., FELDMAN, B. E., LABELLE, F., O'BRIEN, J. F., AND SHEWCHUK, J. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Symposium on Computer Animation*, 219–228.

CHORIN, A., AND MARSDEN, J. 1979. *A Mathematical Introduction to Fluid Mechanics*, 3rd edition ed. Springer-Verlag.

DUPONCHEEL, M., ORLANDI, P., AND WINCKELMANS, G. 2008. Time-reversibility of the Euler equations as a benchmark for energy conserving schemes. *Journal of Computational Physics 227*, 19, 8736–8752.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics 26*, 1 (Jan.), art. 4.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH*, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Animating suspended particle explosions. *ACM Transactions on Graphics 22*, 3 (July), 708–715.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. *ACM Transactions on Graphics 24*, 3, 904–909.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proceedings of SIGGRAPH*, 181–188.

GRESHO, P. M., AND SANI, R. L. 2000. *Incompressible Flow and the Finite Element Method.* J. Wiley & Sons.

HARLOW, F. H., AND WELCH, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids 8*, 12 (Dec.), 2182–2189.

Kim, B., Liu, Y., Llamas, I., and Rossignac, J. 2007. Advections with significantly reduced dissipation and diffusion. *IEEE Trans. on Visualiz.and Comp. Graphics 13(1)*, 135–144.

Losasso, F., Gibou, F., and Fedkiw, R. 2004. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics 23*, 3 (Aug.), 457–462.

Mahesh, K., Constantinescu, G., and Moin, P. 2004. A numerical method for large-eddy simulation in complex geometries. *J. Comput. Phys. 197*, 1, 215–240.

Molemaker, J., Cohen, J. M., Patel, S., and yong Noh, J. 2008. Low viscosity flow simulations for animation. In *Symposium on Computer Animation*, 9–18.

Pavlov, D. 2009. *Structure-preserving Discretizations of Incompressible Fluids*. PhD dissertation in Mathematics, California Institute of Technology.

Perot, B. 2000. Conservation properties of unstructured staggered mesh schemes. *J. Comput. Phys. 159*, 1, 58–89.

Schechter, H., and Bridson, R. 2008. Evolving sub-grid turbulence for smoke animation. In *Symposium on Computer Animation*, 1–8.

Selle, A., Rasmussen, N., and Fedkiw, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics 24*, 3 (Aug.), 910–914.

Selle, A., Fedkiw, R., Kim, B., Liu, Y., and Rossignac, J. 2008. An unconditionally stable MacCormack method. *J. Sci. Comp. 35*, 350–371.

Shi, L., and Yu, Y. 2002. Visual smoke simulation with adaptive octree refinement. *Computer Graphics and Imaging*.

Simo, J., and Armero, F. 1994. Unconditional stability and long-term behavior of transient algorithms for the incompressible Navier-Stokes and Euler equations. *Computer Methods in Applied Mechanics and Engineering 111*, 1-2, 111–154.

Stam, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH*, 121–128.

Steinhoff, J., and Underhill, D. 1994. Modification of the euler equations for *Vorticity Confinement*. *Physics of Fluids 6*, 8 (Aug.), 2738–2744.

Stern, A., and Desbrun, M. 2006. Discrete geometric mechanics for variational time integrators. In *ACM SIGGRAPH Course Notes*, 75–80.

Zhang, X., Schmidt, D., and Perot, B. 2002. Accuracy and conservation properties of a 3d unstructured staggered mesh scheme for fluid dynamics. *J. Comput. Phys. 175*, 2, 764–791.

Zhu, Y., and Bridson, R. 2005. Animating sand as a fluid. In *Proceedings of ACM SIGGRAPH*, 965–972.

A Finite Volume Discretization of Advection Term

We briefly provide a derivation of the advection term in this section. With first-order accuracy, we rewrite the area integral involved as:

$$\int_{f_{ij}} (\boldsymbol{\omega}\times\mathbf{u})\cdot\mathbf{n}\,dA \simeq -\int_{f_{ij}} \nabla\times(\mathbf{u}\cdot(x-c_{ij})\boldsymbol{\omega})\cdot\mathbf{n}\,dA$$
$$= -\int_{\partial f_{ij}} (\mathbf{u}\cdot(x-c_{ij}))\boldsymbol{\omega}\cdot\,\mathbf{dl}.$$

Consequently, $\mathrm{Adv}(F)_{ij}$ can be approximated by summing, over the edges of face f_{ij}, the product of $\boldsymbol{\omega}$ dotted with an edge e and $\mathbf{u}$ dotted with $h_{e,ij} = c_{ij} - c_e$ (midpoint quadrature). Thus, using the vorticity ω_e approximated on face s_e dual to edge e, the contribution from flux F_{jk} to $\dot{F}_{ij}$ is

$$\frac{\omega_e|e|}{|s_e|}(\mathbf{u}\cdot h_{e,ij}) = \frac{\omega_e|e|}{|s_e|\sin(\alpha)}\left(\frac{F_{jk}}{|f_{jk}|} - \cos(\alpha)\frac{F_{ij}}{|f_{ij}|}\right)|h_{e,ij}|,$$

where α is the dihedral angle of tet T_j at edge e: indeed, $\mathbf{u}$ projected onto the direction of $h_{e,ij}$ is determined by the projection of $\mathbf{u}$ onto the plane orthogonal to e, *i.e.*, by its projections $F_{ij}/|f_{ij}|$ and $F_{jk}/|f_{jk}|$ on two non-colinear directions within that plane. We then convert the above expression to $\star_2\mathrm{Adv}(F)$ by multiplying through $|h_{ij,j}|/|f_{ij}|$:

$$\frac{\boldsymbol{\omega}_e}{|s_e|}\left(F_{jk}\frac{4|f_{ij,e}||h_{ij,j}|}{3|T_j||e|} - F_{ij}\frac{2|f_{ij,e}||h_{ij,j}|}{|f_{ij}|^2\cot\alpha}\right),$$

where $f_{ij,e}$ is the triangle formed by e and the circumcenter c_{ij} of the face, and $h_{ij,j}$ is the partial dual edge in cell j.

Notice that in the continuous limit, one should have $\int(\boldsymbol{\omega}\times\mathbf{u})\times\mathbf{u}=0$ for arbitrary $\boldsymbol{\omega}$ and $\mathbf{u}$. We enforce this property at the discrete level by further ensuring that its discrete equivalent, $F\star_2\mathrm{Adv}(F,\omega) = 0$, is automatically satisfied. It is easy to see that if the discrete version of $\int(\boldsymbol{\omega}\times\mathbf{u})\times\mathbf{u}=0$ is to hold for all $\boldsymbol{\omega}$ and $\mathbf{u}$, we must *antisymmetrize* the contribution from F_{ij} to $\star_2\mathrm{Adv}(F)_{jk}$ and from F_{jk} to $\star_2\mathrm{Adv}(F)_{ij}$. Thus, the contribution of $\boldsymbol{\omega}_e$ and F_{jk} to $\star_2\mathrm{Adv}(F)_{ij}$ is rectified to

$$2\frac{\boldsymbol{\omega}_e}{|s_e|}F_{jk}\frac{|f_{ij,e}||h_{ij,j}| + |f_{ik,e}||h_{ik,j}|}{3|T_j||e|} = \frac{\boldsymbol{\omega}_e}{|s_e|}F_{jk}\frac{2|w_{e,j}|}{3|T_j|},$$

where $w_{e,j}$ is the quad-shaped partial dual face of e in cell j.

This discretization of the advection term now enforces energy preservation. Indeed, if we denote $\bar{F}^{n+\frac{1}{2}} := (F^n + F^{n+1})/2$ as before, our antisymmetrization of the finite-volume approximation of the change of flux directly yields:

$$\bar{F}^{n+\frac{1}{2}} \star_2 \mathrm{Adv}(\bar{F}^{n+\frac{1}{2}}) = 0.$$

Multiplying the time update in Eq. (5) by $\bar{F}^{n+\frac{1}{2}}$ proves that this condition enforces that the energy at time t^{n+1} is equal to the energy at time t^n.

B Discrete Geometric Mechanical Derivation

Our trapezoidal-based numerical integrator (Section 3.4) can also be derived from first principles. We only sketch the derivation here—a full derivation can be found in [Pavlov 2009]. While time integrators for fluids are often derived by approximating equations of motion, we instead discretize the configuration space of incompressible fluids and then derive the equations of motion through the principle of stationary action. Our approach uses an Eulerian representation of *discrete, volume-preserving diffeomorphisms* that encodes the displacement of a fluid from its initial configuration using matrices whose rows and columns sum to one. From this particular discretization of the configuration space, which forms a finite-dimensional Lie group, one can derive a discrete equivalent to the Eulerian velocity through its Lie algebra, i.e., through matrices whose rows and columns sum to zero. After imposing non-holonomic constraints on the velocity field to ensure physical transfer only between neighboring cells during each time update, we apply Lagrange-d'Alembert principle (a variant of Hamilton's principle for non-holonomic systems) to obtain the discrete equations of motion for our fluid representation. The update rule obtained this way (using a simple finite-volume advection operator to define kinematic advection) corresponds to what we detail in Section 3.4. The resulting Eulerian variational Lie-group integrator is structure-preserving, and as such, has numerous numerical properties, from momentum preservation (through a discrete Noether theorem) to good long-term energy behavior.